Barrie Sosinsky

Cloud Computing Bible



Explore the cloud with this complete guide

Understand all platforms and technologies

Use Google, Amazon, or Microsoft Web services

The book you need to succeed!

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Cloud Computing Bible

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Cloud Computing Bible

Barrie Sosinsky



Wiley Publishing, Inc.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Cloud Computing Bible

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2011 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-90356-8

10 9 8 7 6 5 4 3

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, 201-748-6011, fax 201-748-6008, or online at `http://www.wiley.com/go/permissions`.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Library of Congress Control Number: 2010941512

Trademarks: Wiley, the Wiley logo, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

This book is dedicated to my sister Gina Sosinsky, with love.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

About the Author

Barrie Sosinsky has written about computers and technology for more than 25 years beginning with writing about personal computers for the Boston Computer Society in the early 1980s. He has published books on operating systems, applications, databases, desktop publishing, and networking for publishers such as Que, Sybex, Ventana, IDG, Wiley, and others and seen the industry change and reinvent itself several times. His last book was Wiley's *Networking Bible*.

At heart Barrie is a PC enthusiast. He loves building computers, finding and learning about new applications that allow him to do new things, and keeping up with the latest advances in the field of computer technology, which he believes is just in its infancy. Having lived long enough to see the Boston Red Sox win not one but two World Series, he remains committed to living long enough to see grandchildren and to see someone clone a wooly mammoth. To this list (replacing the Red Sox) he adds the new milestone of holding a universal translator in his hands; a device he believes will appear within this decade.

Barrie lives in Medfield Massachusetts about 25 miles southwest of Boston with his six cats Stormy, Shadow, Smokey, Scamper, Slate, and Spat as writing companions; Scout the wonder dog; his son Joseph; his daughter Allie; and his wife Carol; surrounded by pine trees, marauding deer, wild turkeys, and the occasional fox and coyote.

You can reach Barrie at `bsosinsky@mindspring.com`, where he welcomes your comments and suggestions.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Credits

Senior Acquisitions Editor
Stephanie McComb

Project Editor
Martin V. Minner

Technical Editor
Benjamin M. Schupak

Copy Editor
Gwenette Gaddis

Editorial Director
Robyn Siesky

Editorial Manager
Rosemarie Graham

Business Manager
Amy Knies

Senior Marketing Manager
Sandy Smith

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Barry Pruett

Project Coordinator
Patrick Redmond

Graphics and Production Specialists
Nikki Gately
Andrea Hornberger

Quality Control Technician
Lindsay Littrell

Proofreading and Indexing
Evelyn Wellborn
Sherry Massey

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Preface

Cloud Computing Bible is Wiley's general introduction to an important topic in large book format. A Bible is a book that is meant to be read by knowledgeable readers who are not subject matter experts in a topic but want to have an in-depth introduction to the various individual subjects contained within. It is assumed that the reader of this book may be a generalist, a developer, a system architect, a programmer, or perhaps something else, and therefore the content in a Bible must contain information for each member of this book's audience.

Cloud computing is a vast topic that encompasses many different subjects. To adequately describe what cloud computing offers, we must discuss infrastructure, service-oriented architectures, social networking, unique protocols, open and standard Application Programming Interfaces (APIs), and dozens of other topics. Even a large book can address many of these topics in only an introductory manner. However, this book tries to give you at least the basic information you need on all the related topics, as well as pointers to additional information sources.

In the last several years, many books have been published on cloud computing. Each book has attempted to present some element of the topic for a particular audience. In this book, I do not make the assumption that you are a particular type of reader, nor do I assume that you are approaching the topic with a fresh view. This Bible was written to serve as the introductory course in the topic at a university level, but it is not a textbook. You can pick up and read this book at any particular chapter because the material doesn't build upon itself.

Many topics in this book are unique to this book and are based on published information that is both current and timely. In researching this book, I attempted to bring into the discussion all the new trends, experiments, and products that have made cloud computing such a dynamic area.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Acknowledgments

I want to acknowledge the editorial team at Wiley for giving me the chance to do this book and to work with them again. I also want to thank my literary agent, Matt Wagner, for his assistance; without his vision, this book would not have been created. Finally, I want to thank my family for allowing me the time I spent away from them writing this book.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Contents at a Glance

Preface xi
Acknowledgments xiii

Part I: Examining the Value Proposition 1
Chapter 1: Defining Cloud Computing 3
Chapter 2: Assessing the Value Proposition 23
Chapter 3: Understanding Cloud Architecture 45
Chapter 4: Understanding Services and Applications by Type 65

Part II: Using Platforms 91
Chapter 5: Understanding Abstraction and Virtualization 93
Chapter 6: Capacity Planning 113
Chapter 7: Exploring Platform as a Service 133
Chapter 8: Using Google Web Services 151
Chapter 9: Using Amazon Web Services 179
Chapter 10: Using Microsoft Cloud Services 205

Part III: Exploring Cloud Infrastructures. 229
Chapter 11: Managing the Cloud 231
Chapter 12: Understanding Cloud Security 249

Part IV: Understanding Services and Applications 269
Chapter 13: Understanding Service Oriented Architecture 271
Chapter 14: Moving Applications to the Cloud 297
Chapter 15: Working with Cloud-Based Storage 311
Chapter 16: Working with Productivity Software 339
Chapter 17: Using Webmail Services 361
Chapter 18: Communicating with the Cloud 383
Chapter 19: Using Media and Streaming 403

Part V: Using the Mobile Cloud 423
Chapter 20: Working with Mobile Devices 425
Chapter 21: Working with Mobile Web Services 445

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Contents

About the Author viii

Preface xi

Acknowledgments xiii

Part I: Examining the Value Proposition 1

Chapter 1: Defining Cloud Computing. 3

Defining Cloud Computing 4
Cloud Types 5
The NIST model 5
The Cloud Cube Model 6
Deployment models 7
Service models 9
Examining the Characteristics of Cloud Computing 13
Paradigm shift 13
Benefits of cloud computing 16
Disadvantages of cloud computing 18
Assessing the Role of Open Standards 19
Summary 22

Chapter 2: Assessing the Value Proposition 23

Measuring the Cloud's Value 24
Early adopters and new applications 26
The laws of cloudonomics 27
Cloud computing obstacles 29
Behavioral factors relating to cloud adoption 31
Measuring cloud computing costs 33
Avoiding Capital Expenditures 35
Right-sizing 36
Computing the Total Cost of Ownership 37
Specifying Service Level Agreements 39
Defining Licensing Models 41
Summary 42

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Chapter 3: Understanding Cloud Architecture 45
Exploring the Cloud Computing Stack 46
Composability 46
Infrastructure 48
Platforms 49
Virtual Appliances 51
Communication Protocols 55
Applications 58
Connecting to the Cloud 58
The Jolicloud Netbook OS 60
Chromium OS: The Browser as an Operating System 61
Summary 64

Chapter 4: Understanding Services and Applications by Type 65
Defining Infrastructure as a Service (IaaS) 66
IaaS workloads 67
Pods, aggregation, and silos 69
Defining Platform as a Service (PaaS) 70
Defining Software as a Service (SaaS) 71
SaaS characteristics 72
Open SaaS and SOA 73
Salesforce.com and CRM SaaS 75
Defining Identity as a Service (IDaaS) 76
What is an identity? 76
Networked identity service classes 78
Identity system codes of conduct 79
IDaaS interoperability 79
User authentication 81
Authorization markup languages 84
Defining Compliance as a Service (CaaS) 87
Summary 88

Part II: Using Platforms 91

Chapter 5: Understanding Abstraction and Virtualization 93
Using Virtualization Technologies 94
Load Balancing and Virtualization 95
Advanced load balancing 96
The Google cloud 97
Understanding Hypervisors 100
Virtual machine types 100
VMware vSphere 103

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Understanding Machine Imaging 107
Porting Applications 108
The Simple Cloud API 109
AppZero Virtual Application Appliance 109
Summary 111

Chapter 6: Capacity Planning 113

Capacity Planning 114
Defining Baseline and Metrics 115
Baseline measurements 115
System metrics 118
Load testing 121
Resource ceilings 122
Server and instance types 126
Network Capacity 128
Scaling 131
Summary 132

Chapter 7: Exploring Platform as a Service 133

Defining Services 134
Salesforce.com versus Force.com: SaaS versus PaaS 135
Application development 137
Using PaaS Application Frameworks 138
Drupal 140
Eccentex AppBase 3.0 141
LongJump 143
Squarespace 145
WaveMaker 146
Wolf Frameworks 147
Summary 150

Chapter 8: Using Google Web Services 151

Exploring Google Applications 152
Surveying the Google Application Portfolio 153
Indexed search 154
The dark Web 155
Aggregation and disintermediation 156
Productivity applications and services 156
Enterprise offerings 159
AdWords 161
Google Analytics 162
Google Translate 163
Exploring the Google Toolkit 165
The Google APIs 166
Working with the Google App Engine 173
Summary 178

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Chapter 9: Using Amazon Web Services. 179
Understanding Amazon Web Services 180
Amazon Web Service Components and Services 181
Working with the Elastic Compute Cloud (EC2) 185
Amazon Machine Images 186
Pricing models 187
System images and software 188
Creating an account and instance on EC2 190
Working with Amazon Storage Systems 196
Amazon Simple Storage System (S3) 196
Amazon Elastic Block Store (EBS) 197
CloudFront 199
Understanding Amazon Database Services 200
Amazon SimpleDB 200
Amazon Relational Database Service (RDS) 201
Choosing a database for AWS 202
Summary 202

Chapter 10: Using Microsoft Cloud Services 205
Exploring Microsoft Cloud Services 206
Defining the Windows Azure Platform 209
The software plus services approach 210
The Azure Platform 210
The Windows Azure service 211
Windows Azure AppFabric 214
Azure Content Delivery Network 216
SQL Azure 217
Windows Azure pricing 217
Windows Live services 219
Using Windows Live 221
Windows Live Essentials 224
Windows Live Home 226
Windows Live for Mobile 227
Summary 228

Part III: Exploring Cloud Infrastructures 229

Chapter 11: Managing the Cloud 231
Administrating the Clouds 232
Management responsibilities 234
Lifecycle management 237
Cloud Management Products 237

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Emerging Cloud Management Standards 242
DMTF cloud management standards 243
Cloud Commons and SMI 244
Summary 248

Chapter 12: Understanding Cloud Security 249

Securing the Cloud 250
The security boundary 251
Security service boundary 253
Security mapping 255
Securing Data 256
Brokered cloud storage access 256
Storage location and tenancy 259
Encryption 260
Auditing and compliance 261
Establishing Identity and Presence 263
Identity protocol standards 264
Windows Azure identity standards 266
Presence 267
Summary 268

Part IV: Understanding Services and Applications 269

Chapter 13: Understanding Service Oriented Architecture 271

Introducing Service Oriented Architecture 272
Event-driven SOA or SOA 2.0 277
The Enterprise Service Bus 278
Service catalogs 281
Defining SOA Communications 282
Business Process Execution Language 286
Business process modeling 287
Managing and Monitoring SOA 292
SOA management tools 292
SOA security 293
The Open Cloud Consortium 294
Relating SOA and Cloud Computing 295
Summary 296

Chapter 14: Moving Applications to the Cloud 297

Applications in the Clouds 298
Functionality mapping 299
Application attributes 300
Cloud service attributes 302

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

System abstraction 303
Cloud bursting 304
Applications and Cloud APIs 308
Summary 309

Chapter 15: Working with Cloud-Based Storage 311

Measuring the Digital Universe 312
Cloud storage in the Digital Universe 314
Cloud storage definition 315
Provisioning Cloud Storage 316
Unmanaged cloud storage 316
Managed cloud storage 320
Creating cloud storage systems 321
Virtual storage containers 322
Exploring Cloud Backup Solutions 325
Backup types 325
Cloud backup features 327
Cloud attached backup 331
Cloud Storage Interoperability 332
Cloud Data Management Interface (CDMI) 333
Open Cloud Computing Interface (OCCI) 335
Summary 337

Chapter 16: Working with Productivity Software 339

Using Productivity Applications 340
Characteristics of productivity software 341
Online Office Systems 342
Acrobat.com 343
Glide Digital 345
Google Docs 347
Microsoft Office Web Apps 351
ThinkFree Office 354
Zoho Office Suite 355
Summary 360

Chapter 17: Using Webmail Services 361

Exploring the Cloud Mail Services 362
Google Gmail 366
Mail2Web 369
Windows Live Hotmail 370
Yahoo! Mail 373
Working with Syndication Services 375
The RSS and Atom Protocols 376
Newsreaders 377
News Aggregators 378
Summary 382

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Chapter 18: Communicating with the Cloud 383

Exploring Instant Messaging 384
Instant messaging clients 384
Instant messaging interoperability 387
Micro-blogs or Short Message Services 388
Exploring Collaboration Technologies 391
Using Social Networks 392
Features 393
List of social networking sites 394
Privacy and security 398
Interaction and interoperability 399
Summary 401

Chapter 19: Using Media and Streaming 403

Understanding the Streaming Process 404
Protocols in Use 405
The cloud computing advantages 405
Audio Streaming 407
Working with VoIP Applications 409
Skype 409
Google Voice and Google Talk 411
Video Streaming 412
Television in the cloud 413
Streaming video formats 416
YouTube 417
YouTube technologies 418
YouTube and the law 420
Summary 421

Part V: Using the Mobile Cloud 423

Chapter 20: Working with Mobile Devices 425

Defining the Mobile Market 426
Connecting to the cloud 427
Adopting mobile cloud applications 428
Feature phones and the cloud 429
Using Smartphones with the Cloud 430
Android 431
Apple iPhone 433
The App Store 434
MobileMe 435
iPhone apps hosting services 436

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Research In Motion BlackBerry 43
Symbian 44
Windows Mobile 44
Summary 44

Chapter 21: Working with Mobile Web Services 44

Understanding Service Types 44
Mobile interoperability 44
Performing Service Discovery 45
Context-aware services 45
MEMS 45
Location awareness 45
Push services 45
The BlackBerry Push Service 45
The Lemonade Profile 45
Using SMS 45
Defining WAP and Other Protocols 46
Performing Synchronization 46
Summary 46

Index 46

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Introduction

In the five months that I have been researching and writing *Cloud Computing Bible,* it has become clear to me that most people recognize that cloud computing is a big deal, even if they are not really clear why that is so. Every day newspaper and magazine articles and radio and TV stories report on cloud computing. The phrase "in the cloud" has entered into our colloquial language. You may have heard that the United States government has initiated a "cloud initiative," or that nearly 75 percent of the developers at Microsoft are currently working on "cloud-related" products, or that a phone or service stores its data in the cloud. The cloud is therefore this amorphous entity that is supposed to represent the future of modern computing.

In reality, the cloud is something that you have been using for a long time now; it is the Internet, along with all the associated standards and protocols that provide a set of Web services to you. When you draw the Internet as a cloud, you are representing one of the essential characteristics of cloud computing: abstraction. In the cloud, resources are pooled and partitioned as needed, and communications are standards-based.

The Internet was begun as a network of networks, with an architecture that was redundant and could survive massive disruption. What the original system architects of the Internet could not have anticipated is that the size of resources attached to it would become massively scalable, which is the second characteristic of cloud computing.

Google's infrastructure, for example, which is described in this book in Chapter 9, spans 30 datacenters around the world with over a million computers; infrastructure that Google now leases out to developers upon which applications may be staged. So the third and equally as important characteristic of cloud computing is that the cloud is a "utility" and that services are provided using a pay-as-you-go model.

A computing utility has been a dream of computer scientists and industry luminaries for several decades. With a utility model of computing, an application can start small and grow to be enormous overnight. This democratization of computing means that any application has the potential to scale, and that even the smallest seed planted in the cloud may be a giant.

Cloud computing will affect your life in the following ways in the next ten years:

- Applications in the cloud will replace applications that are local to your devices.
- Information will become cheaper, more ubiquitous, and easier to find because the cloud makes it cheaper to scale applications and connections to always-on networks such as wireless carriers that make the information always available.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- The cloud will enable new social services by connecting users via social networks that are constructed using multiple cloud services.
- New applications will be easier to create and will be based on standard modular parts.
- It will lessen the role that proprietary operating systems have in our daily computing.
- You will be connected through the cloud wherever you are and at all times.

Frankly, it is hard to predict what new capabilities the cloud may enable. The cloud has a trajectory that is hard to plot and a scope that reaches into so many aspects of our daily life that innovation can occur across a broad range.

Many technologically savvy people have told me they don't understand what the fuss about cloud computing is; in fact, they believe there is nothing new about cloud computing, at least from a technological standpoint. Indeed, they have a point. The technologies that enable cloud computing—system and resource virtualization, thin clients (browsers, for example), virtual private networks and tunneling, and others—are all technologies that existed before anyone ever began to talk about cloud computing. That is all true. Cloud computing is a revolutionary way of architecting and implementing services based on evolutionary changes. *Cloud Computing Bible* attempts to explain how this all came about.

How to Read This Book

Cloud Computing Bible is made up of 21 chapters in five parts. To read this book and get the most out of it, you should know about basic computer operations and theory. You should be able to turn a computer on and know what operating system is running, how processing and input/output is used, and be able to connect with a browser to different Web sites. You should understand the basic user interface elements used by many browsers, such as Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, or Google Chrome.

These are basic skills without which it would be hard to effectively maximize the value contained in this book. If you don't have these skills, Wiley publishes a number of introductory computer books that will give them to you.

It doesn't matter which type of computer operating system you use because most of cloud computing is operating-system-neutral. Indeed, as time goes by, it may not matter whether you use a computer at all. Mobile devices such as smartphones and tablets are on their way to displacing computers in many venues. If you have some familiarity with smartphones, that would be helpful in understanding the last part of this book on mobile-based cloud applications, but it isn't a necessity.

Part I of the book, called "Examining the Value Proposition," defines what cloud computing is and why you should be interested in it. This vocabulary, along with description of cloud architectures

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

and types, will allow you to discuss cloud computing in a standard way and serves to give you a framework over which you can place all the different service types that make cloud computing such a rich area.

Part II, called "Using Platforms," looks at the fundamental features that make a cloud computing application unique. You get a background in the concepts of abstraction and virtualization, along with methods for examining how applications are scaled. This part contains several chapters of vendor-specific services that are illustrative of different cloud computing models. In several chapters, I discuss vendors that are thought leaders in different fields of cloud computing. For infrastructure, I've chosen to highlight Amazon Web Services, and for platforms and services, you learn about the efforts of Google and Microsoft in cloud computing.

Part III, "Exploring Cloud Infrastructures," contains two chapters about managing the cloud and working with the cloud securely. The cloud builds on standard distributed networking technologies, applied over systems with large resources, often over federated systems and services.

In Part IV, "Understanding Services and Applications," the first two chapters describe Service Oriented Architecture and transactions—both of which are important principles in building cloud applications so they are efficient and interoperable—and moving applications to the cloud. The remaining chapters in Part IV describe different types of applications in common use in the cloud today. Those applications are the most highly developed ones in the cloud and have the largest number of users and services. The examples chosen are online backup and storage, Webmail, online productivity applications, messaging, and online media, particularly using streaming technologies.

The book rounds out with two chapters on "Using the Mobile Cloud," Part V. These chapters describe the rise of the smartphone and its predecessor, the feature phone. These phones are supported by a host of Web services. Since 2008, more traffic has been flowing over wireless networks than wired networks, so it would be hard to underestimate how much impact mobile devices have on the cloud. For vast portions of the world, the cell phone is the only computer most people will know. Mobile Web services use different protocols and technologies and can take into account location and other user profile information that can use the cloud to create a rich user experience.

Please dive into whatever chapter interests you. I hope you enjoy reading about cloud computing as much as I enjoyed writing about it.

Icons

The icons in this book offer you a chance to learn a little more about a topic, refer to a discussion elsewhere in the book, address a problem, or get a little more help. This book offers the following icons:

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 1

Defining Cloud Computing

Cloud computing refers to applications and services that run on a distributed network using virtualized resources and accessed by common Internet protocols and networking standards. It is distinguished by the notion that resources are virtual and limitless and that details of the physical systems on which software runs are abstracted from the user.

In an effort to better describe cloud computing, a number of cloud types have been defined. In this chapter, you learn about two different classes of clouds: those based on the deployment model and those based on the service model. The deployment model tells you where the cloud is located and for what purpose. Public, private, community, and hybrid clouds are deployment models.

Service models describe the type of service that the service provider is offering. The best-known service models are Software as a Service, Platform as a Service, and Infrastructure as a Service—the SPI model. The service models build on one another and define what a vendor must manage and what the client's responsibility is.

Cloud computing represents a real paradigm shift in the way in which systems are deployed. The massive scale of cloud computing systems was enabled by the popularization of the Internet and the growth of some large service companies. Cloud computing makes the long-held dream of utility computing possible with a pay-as-you-go, infinitely scalable, universally available system. With cloud computing, you can start very small and become big very fast. That's why cloud computing is revolutionary, even if the technology it is built on is evolutionary.

Not all applications benefit from deployment in the cloud. Issues with latency, transaction control, and in particular security and regulatory compliance are of particular concern.

IN THIS CHAPTER

Defining cloud computing

Learning about cloud types

Understanding the paradigm shift that is cloud computing

Comparing the benefits and disadvantages of cloud systems

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Defining Cloud Computing

Cloud computing takes the technology, services, and applications that are similar to those on the Internet and turns them into a self-service utility. The use of the word "cloud" makes reference to the two essential concepts:

- **Abstraction:** Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.
- **Virtualization:** Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.

Computing as a utility is a dream that dates from the beginning of the computing industry itself. A set of new technologies has come along that, along with the need for more efficient and affordable computing, has enabled an on-demand system to develop. It is these enabling technologies that are the focal point of this book.

Many people mistakenly believe that cloud computing is nothing more than the Internet given a different name. Many drawings of Internet-based systems and services depict the Internet as a cloud, and people refer to applications running on the Internet as "running in the cloud," so the confusion is understandable. The Internet has many of the characteristics of what is now being called cloud computing. The Internet offers abstraction, runs using the same set of protocols and standards, and uses the same applications and operating systems. These same characteristics are found in an intranet, an internal version of the Internet. When an intranet becomes large enough that a diagram no longer wishes to differentiate between individual physical systems, the intranet too becomes identified as a cloud.

Cloud computing is an abstraction based on the notion of pooling physical resources and presenting them as a virtual resource. It is a new model for provisioning resources, for staging applications, and for platform-independent user access to services. Clouds can come in many different types, and the services and applications that run on clouds may or may not be delivered by a cloud service provider. These different types and levels of cloud services mean that it is important to define what type of cloud computing system you are working with.

To help clarify how cloud computing has changed the nature of commercial system deployment, consider these three examples:

- **Google:** In the last decade, Google has built a worldwide network of datacenters to service its search engine. In doing so Google has captured a substantial portion of the world's advertising revenue. That revenue has enabled Google to offer free software to users based on that infrastructure and has changed the market for user-facing software. This is the classic Software as a Service case described in Chapter 8.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Azure Platform:** By contrast, Microsoft is creating the Azure Platform. It enables .NET Framework applications to run over the Internet as an alternate platform for Microsoft developer software running on desktops, which you will learn about in Chapter 10.
- **Amazon Web Services:** One of the most successful cloud-based businesses is Amazon Web Services, which is an Infrastructure as a Service offering that lets you rent virtual computers on Amazon's own infrastructure. AWS is the subject of Chapter 9.

These new capabilities enable applications to be written and deployed with minimal expense and to be rapidly scaled and made available worldwide as business conditions permit. This is truly a revolutionary change in the way enterprise computing is created and deployed.

Cloud Types

To discuss cloud computing intelligently, you need to define the lexicon of cloud computing; many acronyms in this area probably won't survive long. Most people separate cloud computing into two distinct sets of models:

- **Deployment models:** This refers to the location and management of the cloud's infrastructure.
- **Service models:** This consists of the particular types of services that you can access on a cloud computing platform.

This is a very useful demarcation that is now widely accepted.

The NIST model

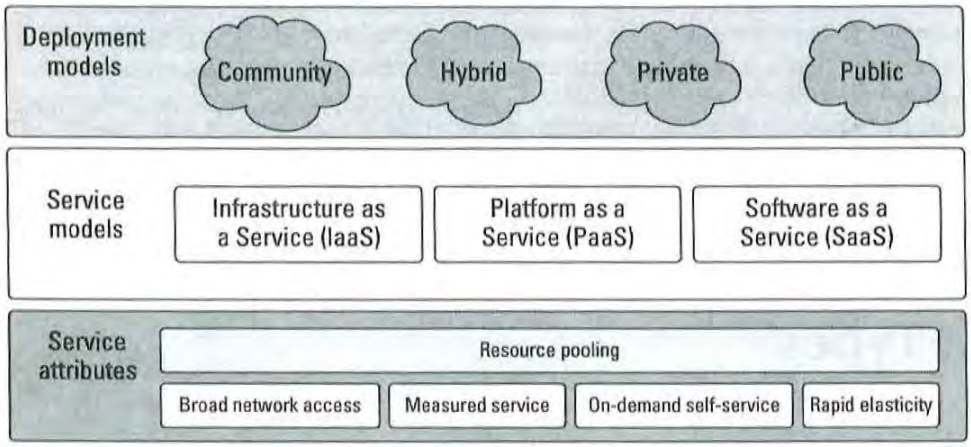
The United States government is a major consumer of computer services and, therefore, one of the major users of cloud computing networks. The U.S. National Institute of Standards and Technology (NIST) has a set of working definitions (`http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc`) that separate cloud computing into service models and deployment models. Those models and their relationship to essential characteristics of cloud computing are shown in Figure 1.1.

The NIST model originally did not require a cloud to use virtualization to pool resources, nor did it absolutely require that a cloud support multi-tenancy in the earliest definitions of cloud computing. Multi-tenancy is the sharing of resources among two or more clients. The latest version of the NIST definition does require that cloud computing networks use virtualization and support multi-tenancy.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 1.1

The NIST cloud computing definitions



Because cloud computing is moving toward a set of modular interacting components based on standards such as the Service Oriented Architecture (described in Chapter 13), you might expect that future versions of the NIST model may add those features as well. The NIST cloud model doesn't address a number of intermediary services such as transaction or service brokers, provisioning, integration, and interoperability services that form the basis for many cloud computing discussions. Given the emerging roles of service buses, brokers, and cloud APIs at various levels, undoubtedly these elements need to be added to capture the whole story.

The Cloud Cube Model

The Open Group maintains an association called the Jericho Forum (`https://www.opengroup.org/jericho/index.htm`) whose main focus is how to protect cloud networks. The group has an interesting model that attempts to categorize a cloud network based on four dimensional factors. As described in its paper called "Cloud Cube Model: Selecting Cloud Formations for Secure Collaboration" (`http://www.opengroup.org/jericho/cloud_cube_model_v1.0.pdf`), the type of cloud networks you use dramatically changes the notion of where the boundary between the client's network and the cloud begins and ends.

The four dimensions of the Cloud Cube Model are shown in Figure 1.2 and listed here:

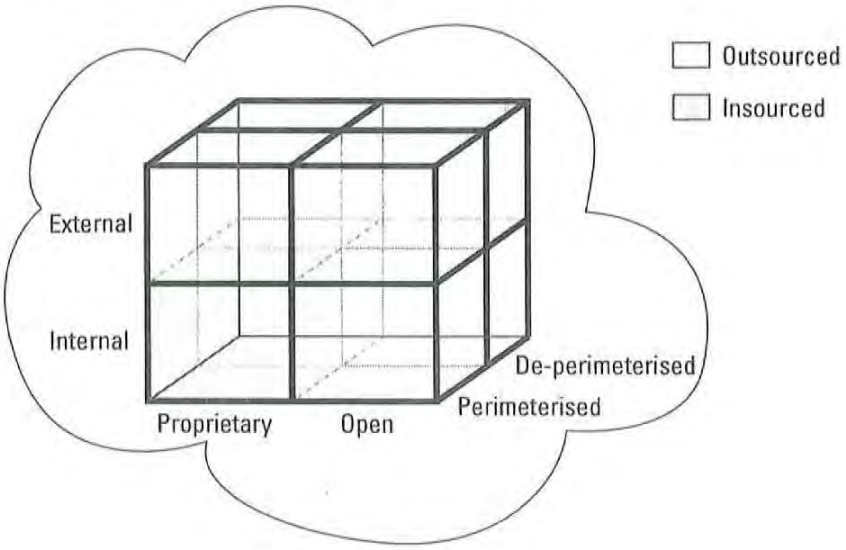
- **Physical location of the data:** Internal (I) / External (E) determines your organization's boundaries.
- **Ownership:** Proprietary (P) / Open (O) is a measure of not only the technology ownership, but of interoperability, ease of data transfer, and degree of vendor application lock-in.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Security boundary:** Perimeterised (Per) / De-perimiterised (D-p) is a measure of whether the operation is inside or outside the security boundary or network firewall.
- **Sourcing:** Insourced or Outsourced means whether the service is provided by the customer or the service provider.

FIGURE 1.2

The Jericho Forum's Cloud Cube Model



Taken together, the fourth dimension corresponds to two different states in the eight possible cloud forms: Per (IP, IO, EP, EO) and D-p (IP, IO, EP, EO). The sourcing dimension addresses the deliverer of the service. What the Cloud Cube Model is meant to show is that the traditional notion of a network boundary being the network's firewall no longer applies in cloud computing.

Deployment models

A deployment model defines the purpose of the cloud and the nature of how the cloud is located.

The NIST definition for the four deployment models is as follows:

- **Public cloud:** The public cloud infrastructure is available for public use alternatively for a large industry group and is owned by an organization selling cloud services.
- **Private cloud:** The private cloud infrastructure is operated for the exclusive use of an organization. The cloud may be managed by that organization or a third party. Private clouds may be either on- or off-premises.
- **Hybrid cloud:** A hybrid cloud combines multiple clouds (private, community of public) where those clouds retain their unique identities, but are bound together as a unit. A

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

hybrid cloud may offer standardized or proprietary access to data and applications, as well as application portability.

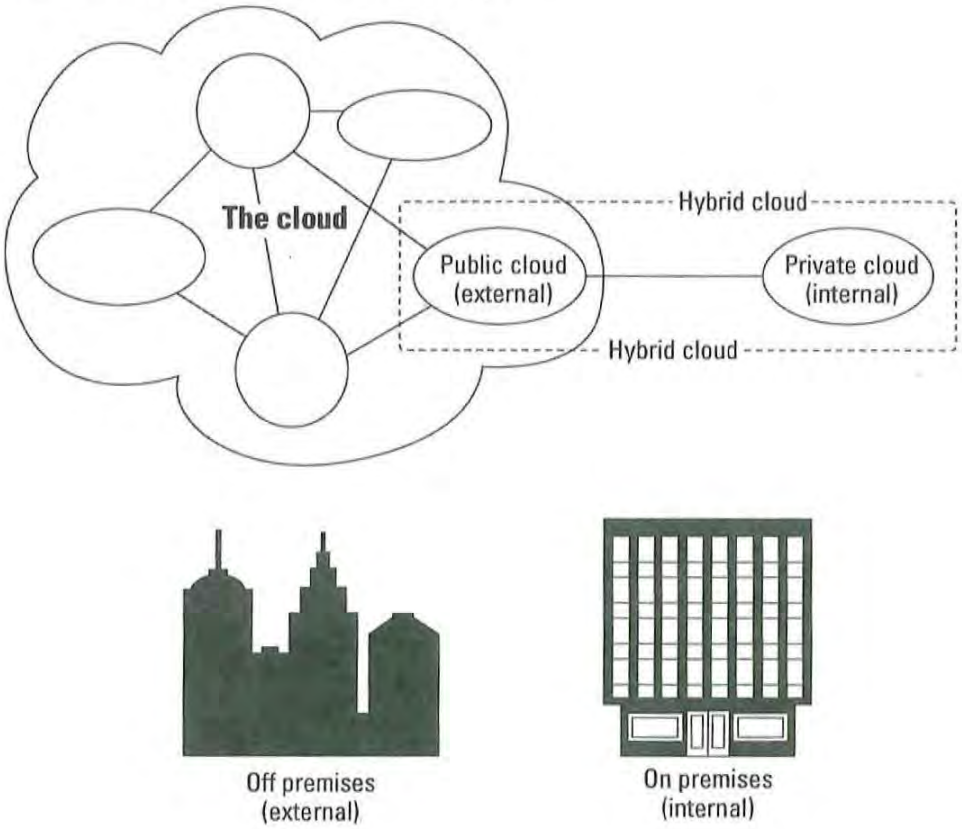
- **Community cloud:** A community cloud is one where the cloud has been organized to serve a common function or purpose.

  It may be for one organization or for several organizations, but they share common concerns such as their mission, policies, security, regulatory compliance needs, and so on. A community cloud may be managed by the constituent organization(s) or by a third party.

Figure 1.3 shows the different locations that clouds can come in. In the sections that follow, these different cloud deployment models are described in more detail.

FIGURE 1.3

Deployment locations for different cloud types



The United States Government, under the auspices of the General Services Administrator (GSA), launched a cloud computing portal called Apps.gov, as shown in Figure 1.4, with the purpose of providing cloud services to federal agencies. Described under the "U.S. Federal Cloud Computing

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Initiative (`http://www.scribd.com/doc/17914883/US-Federal-Cloud-Computing-Initiative-RFQ-GSA`), the goal of the initiative is to make large portions of the federal government's apparatus available under a cloud computing model. This is a good example of a community cloud deployment, with the government being the community.

FIGURE 1.4

Apps.gov is the U.S. government's cloud computing system for its various agencies.



Apps.gov is also making available connections to free media services from its cloud, such as Twitter and YouTube. An example of this connection in practice is the YouTube channel created by the White House for citizens' outreach. You can find the White House channel at `http://www.youtube.com/whitehouse` and the general U.S. Government YouTube channel at `http://www.youtube.com/usgovernment`. You can see YouTube in action when you visit WhiteHouse.gov and click the video link that usually appears on that home page.

Service models

In the deployment model, different cloud types are an expression of the manner in which infrastructure is deployed. You can think of the cloud as the boundary between where a client's network, management, and responsibilities ends and the cloud service provider's begins. As cloud

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

computing has developed, different vendors offer clouds that have different services associated with them. The portfolio of services offered adds another set of definitions called the service model.

There are many different service models described in the literature, all of which take the following form:

XaaS, or "<*Something*> as a Service"

Three service types have been universally accepted:

- **Infrastructure as a Service:** IaaS provides virtual machines, virtual storage, virtual infrastructure, and other hardware assets as resources that clients can provision.

  The IaaS service provider manages all the infrastructure, while the client is responsible for all other aspects of the deployment. This can include the operating system, applications, and user interactions with the system.

- **Platform as a Service:** PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures.

  The client can deploy its applications on the cloud infrastructure or use applications that were programmed using languages and tools that are supported by the PaaS service provider. The service provider manages the cloud infrastructure, the operating systems, and the enabling software. The client is responsible for installing and managing the application that it is deploying.

- **Software as a Service:** SaaS is a complete operating environment with applications, management, and the user interface.

  In the SaaS model, the application is provided to the client through a thin client interface (a browser, usually), and the customer's responsibility begins and ends with entering and managing its data and user interaction. Everything from the application down to the infrastructure is the vendor's responsibility.

The three different service models taken together have come to be known as the SPI model of cloud computing. Many other service models have been mentioned: StaaS, Storage as a Service; IdaaS, Identity as a Service; CmaaS, Compliance as a Service; and so forth. However, the SPI services encompass all the other possibilities.

It is useful to think of cloud computing's service models in terms of a hardware/software stack. One such representation called the Cloud Reference Model is shown in Figure 1.5. At the bottom of the stack is the hardware or infrastructure that comprises the network. As you move upward in the stack, each service model inherits the capabilities of the service model beneath it. IaaS has the least levels of integrated functionality and the lowest levels of integration, and SaaS has the most.

Examples of IaaS service providers include:

- Amazon Elastic Compute Cloud (EC2)
- Eucalyptus
- GoGrid

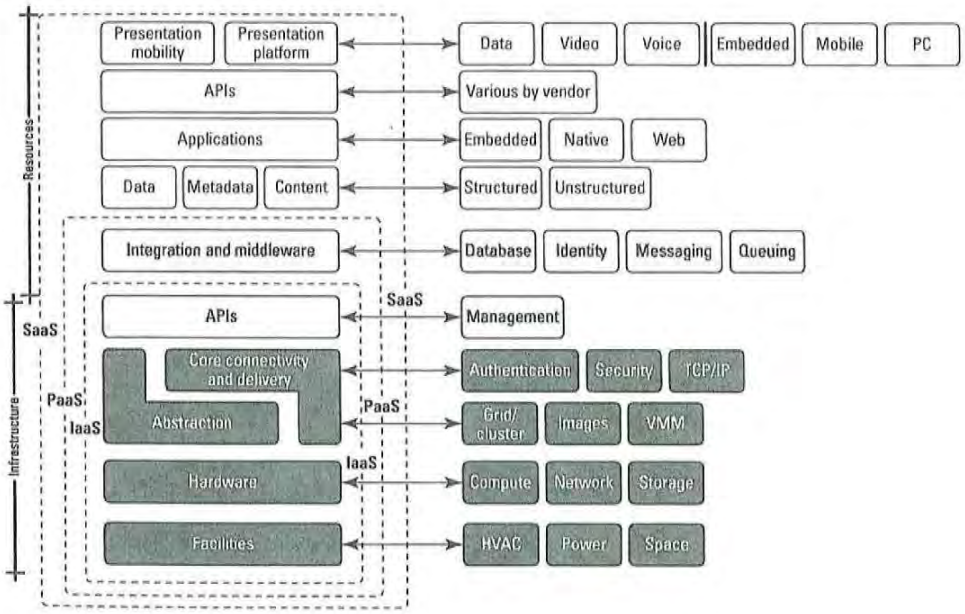
Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- FlexiScale
- Linode
- RackSpace Cloud
- Terremark

All these vendors offer direct access to hardware resources. On Amazon EC2, considered the classic IaaS example, a client would provision a computer in the form of a virtual machine image, provision storage, and then go on to install the operating system and applications onto that virtual system. Amazon has a number of operating systems and some enterprise applications that they offer on a rental basis to customers in the form of a number of canned images, but customers are free to install whatever software they want to run. Amazon's responsibilities as expressed in its Service Level Agreement, which is published on Amazon's Web site, contractually obligates Amazon to provide a level of performance commensurate with the type of resource chosen, as well as a certain level of reliability as measured by the system's uptime.

FIGURE 1.5

The Cloud Reference Model



A PaaS service adds integration features, middleware, and other orchestration and choreography services to the IaaS model. Examples of PaaS services are:

- Force.com
- GoGrid CloudCenter

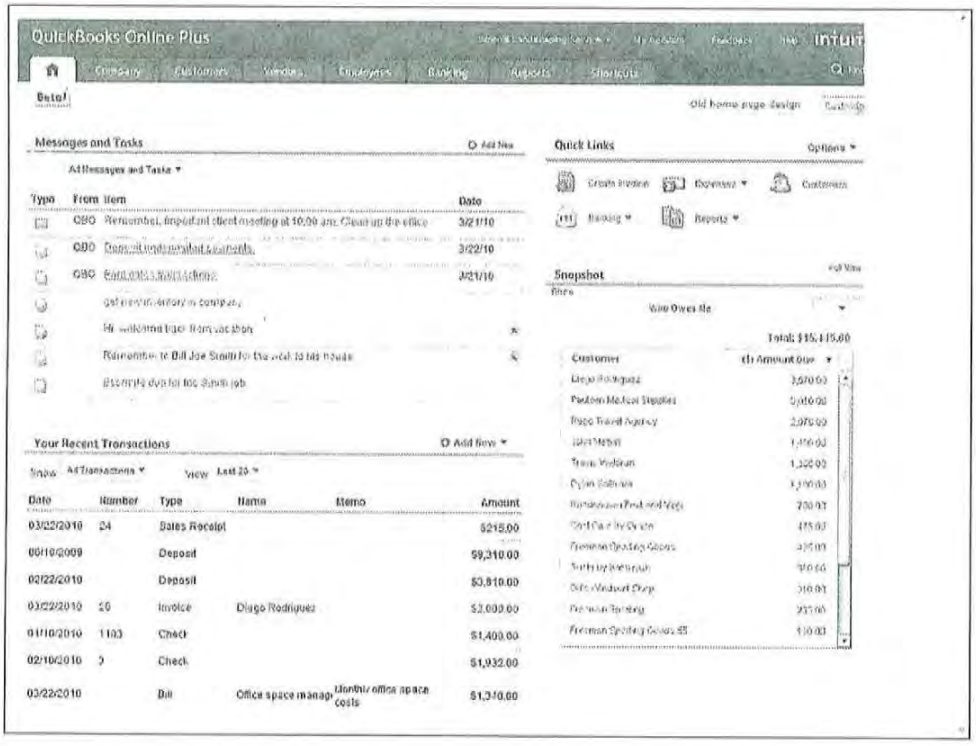
Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- Google AppEngine
- Windows Azure Platform

When a cloud computing vendor offers software running in the cloud with use of the application on a pay-as-you-go model, it is referred to as SaaS. With SaaS, the customer uses the application as needed and is not responsible for the installation of the application, its maintenance, or its upkeep. A good example of an SaaS offering is an online accounting package, with the online versions of Quicken and Quickbooks a prime example. Figure 1.6 shows a home page for QuickBooks Online plus on the Intuit.com Web site.

FIGURE 1.6

A home page for a Quickbooks customer on the Intuit.com Web site is an example of an SaaS service.



A client using an SaaS service might—as is the case for Quickbooks online—log into the service from his browser, create an account, and enter data into the system. Intuit.com has a service agreement that not only covers the performance of the hardware and software, but extends to protecting the data that they store for clients, and other fundamental characteristics.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Other good examples of SaaS cloud service providers are:

- GoogleApps
- Oracle On Demand
- SalesForce.com
- SQL Azure

These service model classifications start to get confusing rather quickly when you have a cloud service provider that starts out offering services in one area and then develops services that are classified as another type. For example, SalesForce.com started out as a Customer Relationship Management SaaS platform that allowed clients to add their own applications. Over time SalesForce.com opened an API called the Force API that allowed developers to create applications based on the SalesForce.com technologies. Force.com is thus their PaaS service.

As another example, take the PaaS offering that is the Windows Azure Platform. Windows Azure Platform allows .NET developers to stage their applications on top of Microsoft's infrastructure so that any application built with the .NET Framework can live locally, in Microsoft's cloud network, or some combination thereof. As Microsoft adds enterprise applications to its cloud service portfolio, as it has in the case of SQL Azure (and many other enterprise applications to come), these offerings fall under the rubric of being an SaaS service model.

Because a discussion of service models forms the basis for Chapter 4, I refer you to that chapter for a more in-depth discussion of this topic.

Examining the Characteristics of Cloud Computing

Cloud computing builds on so many older concepts in computer technology that it can be hard for people newly introduced to the concept to grasp that it represents a paradigm shift in computing. It's an evolutionary change that enables a revolutionary new approach to how computing services are produced and consumed.

Paradigm shift

When you choose a cloud service provider, you are renting or leasing part of an enormous infrastructure of datacenters, computers, storage, and networking capacity. Many of these datacenters are multi-million-dollar investments by the companies that run them. To give you some sense of scale, it has been estimated that a state-of-the-art microchip fabrication facility can cost anywhere from $2 to $5 billion. By comparison, a state of the art cloud computing datacenter can run in the range of $100 million. Most of the large cloud computing service providers have multiple datacenters located all over the world. An accurate count can be difficult to obtain, but in Chapter 9

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

the location of some 20 datacenters in Amazon Web Service's cloud are detailed. Google's cloud includes perhaps some 35 datacenters worldwide.

In the 1960s, military initiative aimed at miniaturizing electronics funded many of the semiconductor production lines that led to advanced microprocessors, dense memory arrays, and the sophisticated integrated circuit technology that makes computers, mobile devices, and so much more possible today. In the 1990s, the commercialization of the Internet gave rise to some very large companies that were forced to build very large computing infrastructures to support their businesses.

Amazon.com's infrastructure was built to support elastic demand so the system could accommodate peak traffic on a busy shopping day such as "Black Monday." Because much of the capacity was idle, Amazon.com first opened its network to partners and then as Amazon Web Services to customers.

Google's business has also grown exponentially and required the building of datacenters worldwide. One of its datacenters in Dalles, Oregon, built in 2006 on the banks of the Columbia River, is shown in Figure 1.7. It is the size of an American football field.

FIGURE 1.7

The Google Dalles, Oregon, datacenter shown in Google Earth is an industrial-sized information technology utility.

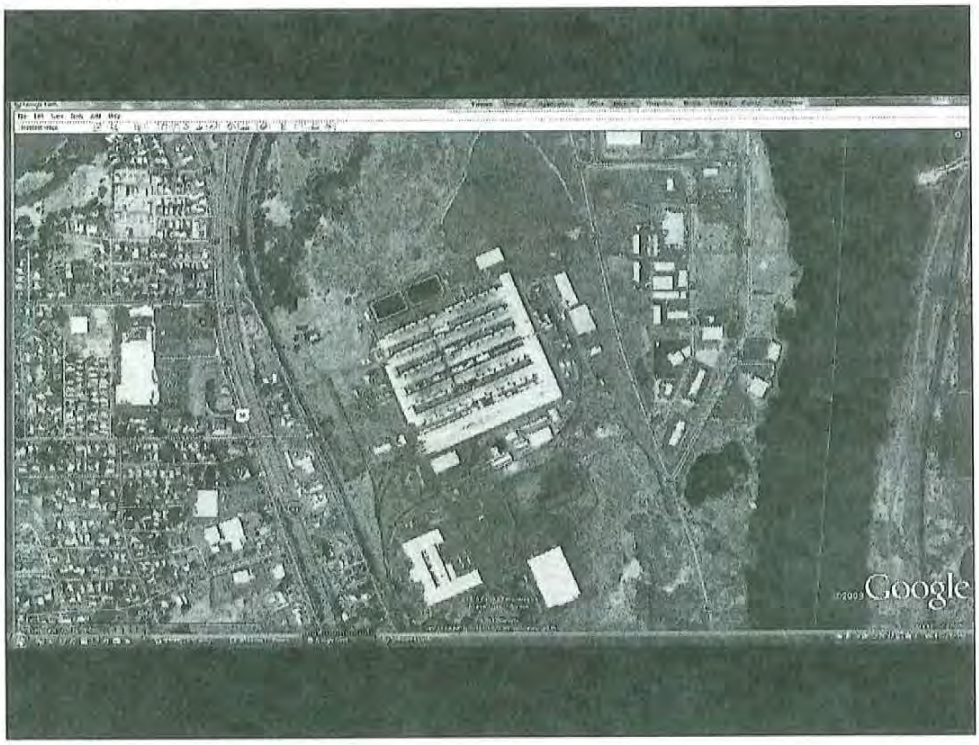


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

As these various datacenters grew in size, businesses have developed their datacenters as "greenfield" projects. Datacenters have been sited to do the following:

- Have access to low cost power
- Leverage renewable power source
- Be near abundant water
- Be sited where high-speed network backbone connections can be made
- Keep land costs modest and occupation unobtrusive
- Obtain tax breaks
- Optimize the overall system latency

These characteristics make cloud computing networks highly efficient and capture enough margin to make utility computing profitable.

It has been estimated that the Internet consumes roughly 10 percent of the world's total power, so these companies are very big energy consumers. In some cases, such as Google, these companies may also become some of the major energy producers of the 21st century. Essentially what has happened is that the Internet has funded the creation of the first information technology utilities. That's why cloud computing is such a big deal.

According to the research firm IDC, the following areas were the top five cloud applications in use in 2010:

- Collaboration applications
- Web applications/Web serving
- Cloud backup
- Business applications
- Personal productivity applications

The last five years have seen a proliferation of services and productivity applications delivered online as cloud computing applications. Examples of the impact of cloud computing abound in your everyday life, although many people do not make the connection to what was once a straightforward client/server Internet deployment. Movement of these applications to the cloud has been transparent, and in many cases the older on-premises deployment is supported by the same applications hosted in the cloud.

For example, many people have used ChannelAdvisor.com for their auction listings and sales management. That site recently expanded its service to include a CRM connector to Salesforce.com. One of the largest call center operations companies is a cloud-based service, Liveops.com. Figure 1.8 shows the Liveops home page.

Cloud computing has shifted the economics of software delivery in a manner similar to the way that music downloads have shifted the delivery of commercial music. The cost advantages of cloud

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

computing have enabled new software vendors to create productivity applications that they can make available to people at a much smaller cost than would be possible for shrink-wrapped software. Given the general demise of the big-box computer store along with many other traditional retail models, it has become increasingly difficult for vendors to get shelf space. You can visit your local Wal-Mart to get some sense of this issue.

In Chapter 16, "Working with Productivity Software," some of these applications are described. This new model of computer application delivery has allowed vendors like Google to offer complete office suites to individuals for free, supported by its advertiser subscription model. Even Google's business offerings have had some major successes against industry leader Microsoft Office. Last year, Los Angeles County switched to Google Docs.

FIGURE 1.8

Liveops.com is a cloud computing call center service.



Benefits of cloud computing

"The NIST Definition of Cloud Computing" by Peter Mell and Tim Grance (version 14, 10/7/2009) described previously in this chapter (refer to Figure 1.1) that classified cloud computing into the three SPI service models (SaaS, IaaS, and PaaS) and four cloud types (public, private, community, and hybrid), also assigns five essential characteristics that cloud computing systems must offer:

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **On-demand self-service:** A client can provision computer resources without the need for interaction with cloud service provider personnel.
- **Broad network access:** Access to resources in the cloud is available over the network using standard methods in a manner that provides platform-independent access to clients of all types.

  This includes a mixture of heterogeneous operating systems, and thick and thin platforms such as laptops, mobile phones, and PDA.
- **Resource pooling:** A cloud service provider creates resources that are pooled together in a system that supports multi-tenant usage.

  Physical and virtual systems are dynamically allocated or reallocated as needed. Intrinsic in this concept of pooling is the idea of abstraction that hides the location of resources such as virtual machines, processing, memory, storage, and network bandwidth and connectivity.
- **Rapid elasticity:** Resources can be rapidly and elastically provisioned.

  The system can add resources by either scaling up systems (more powerful computers) or scaling out systems (more computers of the same kind), and scaling may be automatic or manual. From the standpoint of the client, cloud computing resources should look limitless and can be purchased at any time and in any quantity.
- **Measured service:** The use of cloud system resources is measured, audited, and reported to the customer based on a metered system.

  A client can be charged based on a known metric such as amount of storage used, number of transactions, network I/O (Input/Output) or bandwidth, amount of processing power used, and so forth. A client is charged based on the level of services provided.

While these five core features of cloud computing are on almost anybody's list, you also should consider these additional advantages:

- **Lower costs:** Because cloud networks operate at higher efficiencies and with greater utilization, significant cost reductions are often encountered.
- **Ease of utilization:** Depending upon the type of service being offered, you may find that you do not require hardware or software licenses to implement your service.
- **Quality of Service:** The Quality of Service (QoS) is something that you can obtain under contract from your vendor.
- **Reliability:** The scale of cloud computing networks and their ability to provide load balancing and failover makes them highly reliable, often much more reliable than what you can achieve in a single organization.
- **Outsourced IT management:** A cloud computing deployment lets someone else manage your computing infrastructure while you manage your business. In most instances, you achieve considerable reductions in IT staffing costs.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Simplified maintenance and upgrade:** Because the system is centralized, you can easily apply patches and upgrades. This means your users always have access to the latest software versions.
- **Low Barrier to Entry:** In particular, upfront capital expenditures are dramatically reduced. In cloud computing, anyone can be a giant at any time.

This very long list of benefits should make it obvious why so many people are excited about the idea of cloud computing. Cloud computing is not a panacea, however. In many instances, cloud computing doesn't work well for particular applications.

Disadvantages of cloud computing

While the benefits of cloud computing are myriad, the disadvantages are just as numerous. As a general rule, the advantages of cloud computing present a more compelling case for small organizations than for larger ones. Larger organizations can support IT staff and development efforts that put in place custom software solutions that are crafted with their particular needs in mind.

When you use an application or service in the cloud, you are using something that isn't necessarily as customizable as you might want. Additionally, although many cloud computing applications are very capable, applications deployed on-premises still have many more features than their cloud counterparts.

All cloud computing applications suffer from the inherent latency that is intrinsic in their WAN connectivity. While cloud computing applications excel at large-scale processing tasks, if your application needs large amounts of data transfer, cloud computing may not be the best model for you.

Additionally, cloud computing is a stateless system, as is the Internet in general. In order for communication to survive on a distributed system, it is necessarily unidirectional in nature. All the requests you use in HTTP: PUTs, GETs, and so on are requests to a service provider. The service provider then sends a response. Although it may seem that you are carrying on a conversation between client and provider, there is an architectural disconnect between the two. That lack of state allows messages to travel over different routes and for data to arrive out of sequence, and many other characteristics allow the communication to succeed even when the medium is faulty. Therefore, to impose transactional coherency upon the system, additional overhead in the form of service brokers, transaction managers, and other middleware must be added to the system. This can introduce a very large performance hit into some applications.

If you had to pick a single area of concern in cloud computing, that area would undoubtedly be privacy and security. When your data travels over and rests on systems that are no longer under your control, you have increased risk due to the interception and malfeasance of others. You can't count on a cloud provider maintaining your privacy in the face of government actions.

In the United States, an example is the National Security Agency's program that ran millions of phone calls from AT&T and Verizon through a data analyzer to extract the phone calls that matched its security criteria. VoIP is one of the services that is heavily deployed on cloud computing systems. Another example is the case of Google's service in China, which had been subject to a

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

filter that removed content to which the Chinese government objected. After five years of operation, and after Google detected that Chinese hackers were accessing Gmail accounts of Chinese citizens, Google moved their servers for Google.ch to Hong Kong.

So while the cloud computing industry continues to address security concerns, if you have an application that works with sensitive data, you need to be particularly aware of the issues involved. Chapter 12, "Understanding Cloud Security," expands upon these points in more detail.

These days most organizations are faced with regulatory compliance issues of various kinds. In the United States, companies must comply with the accounting requirements of the Sarbanes-Oxley Act; health care providers comply with the data privacy rules of HIPAA, and so on. In Europe, the European Common Market has a raft of its own legislation for companies to deal with. Rules apply to data at rest, and different rules may apply to data in transit. If you stage your cloud computing deployment across states and countries, the bad news is that you may end up having to comply with multiple jurisdictions. Don't expect much support from the cloud system provider or from the governments involved. The laws of most regulatory agencies place the entire burden on the client. So when it comes to compliance, cloud computing is still the "Wild West" of computing.

Assessing the Role of Open Standards

When you consider the development of cloud computing to date, it is clear that the technology is the result of the convergence of many different standards. Cloud computing's promise of scalability completely changes the manner in which services and applications are deployed. Without standards, the industry creates proprietary systems with vendor lock-in, sometimes referred to as "stovepipe" clouds. Because clients do not want to be locked into any single system, there is a strong industry push to create standards-based clouds.

The cloud computing industry is working with these architectural standards:

- Platform virtualization of resources
- Service-oriented architecture
- Web-application frameworks
- Deployment of open-source software
- Standardized Web services
- Autonomic systems
- Grid computing

These standards help to enable different business models that cloud computing vendors can support, most notably Software as a Service (SaaS), Web 2.0 applications, and utility computing. These businesses require open standards so that data is both portable and universally accessible.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The race to create the first generation of open cloud platform technologies that will compete with proprietary technologies offered by companies such as Microsoft (Azure Platform) and VMware (vSphere) is already underway. Rackspace.com, one of the large IaaS cloud service providers, announced in July 2010 that it is initiating an open-source project called OpenStack that will begin with the code used to run its Cloud Files and Cloud Servers technologies. NASA has also donated some of the Nebula Cloud Platform technology that it developed. The software developed will be released under the Apache 2.0 license. Founding members of this project include AMD, Citrix, Dell, Intel, NTT Data, and several other cloud service providers. OpenStack.org's home page (`http://www.openstack.org/`) is shown in Figure 1.9.

The first two deliverables of the project are a distributed object store based on Rackspace Cloud Files and a scalable machine provisioning technology based on NASA Nebula and Rackspace Cloud Servers. OpenStack Compute software will automatically create large groups of virtual private servers on industry-standard systems. OpenStack Storage is the software that will create redundant object-based storage using clusters of commodity servers and storage systems.

FIGURE 1.9

OpenStack.org is an industry group seeking to create open cloud standards based on Rackspace.com and NASA technologies.

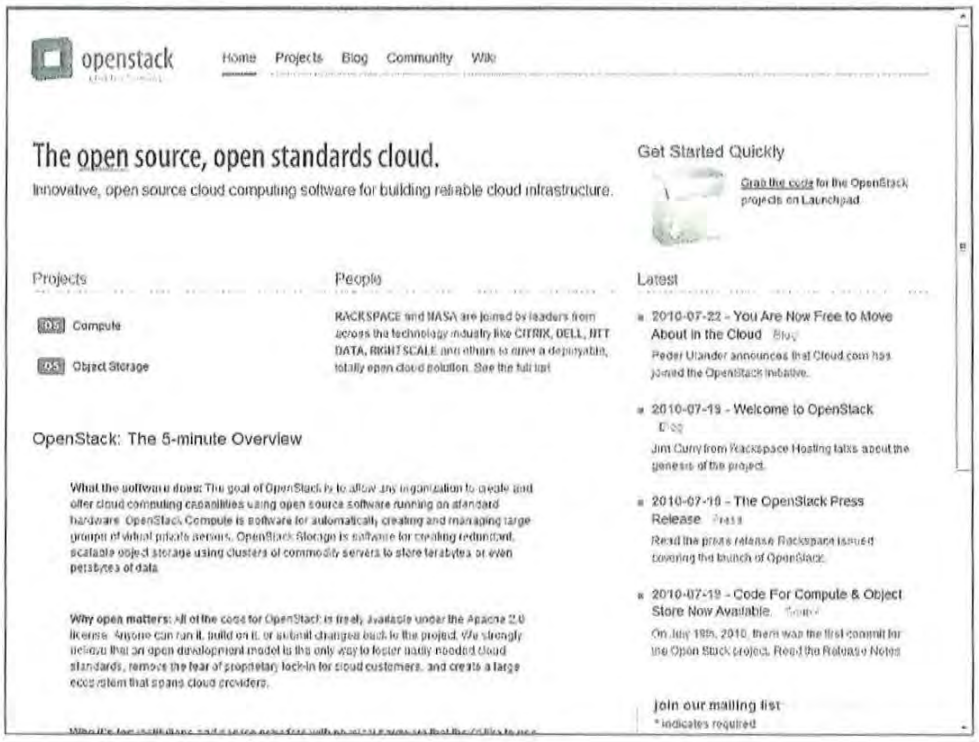


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Eucalyptus (`http://open.eucalyptus.com/`) is a Linux-based software platform for creating cloud computing IaaS systems based on computer clusters. The project has an interface that can connect to Amazon's compute and storage cloud systems (EC2 and S3), and it maintains a private cloud as a sandbox for developers to work in. Eucalyptus works with a number of technologies for system virtualization, including VMware, Xen, and KVM. Eucalyptus is an acronym taken from the expression "Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems." Most of the major Linux vendors support this project, which is based on the original work of Rich Wolski at the University of California at Santa Barbara. The company Eucalyptus Systems was formed in 2009 to support the commercialization of the Eucalyptus Cloud Computing Platform.

OpenStack and Eucalyptus are by no means unique; several other projects are underway to create open-source cloud platforms. There also are numerous research projects in the area. The IEEE Technical Committee on Services Computing (`http://tab.computer.org/tcsc/`) sponsors a conference in this area called CLOUD and has some working groups and publications in this area. Figure 1.10 shows the home page of TCSC.

FIGURE 1.10

The home page of the IEEE Technical Committee on Services Computing (`http://tab.computer.org/tcsc/`), which works with the cloud computing community to create standards

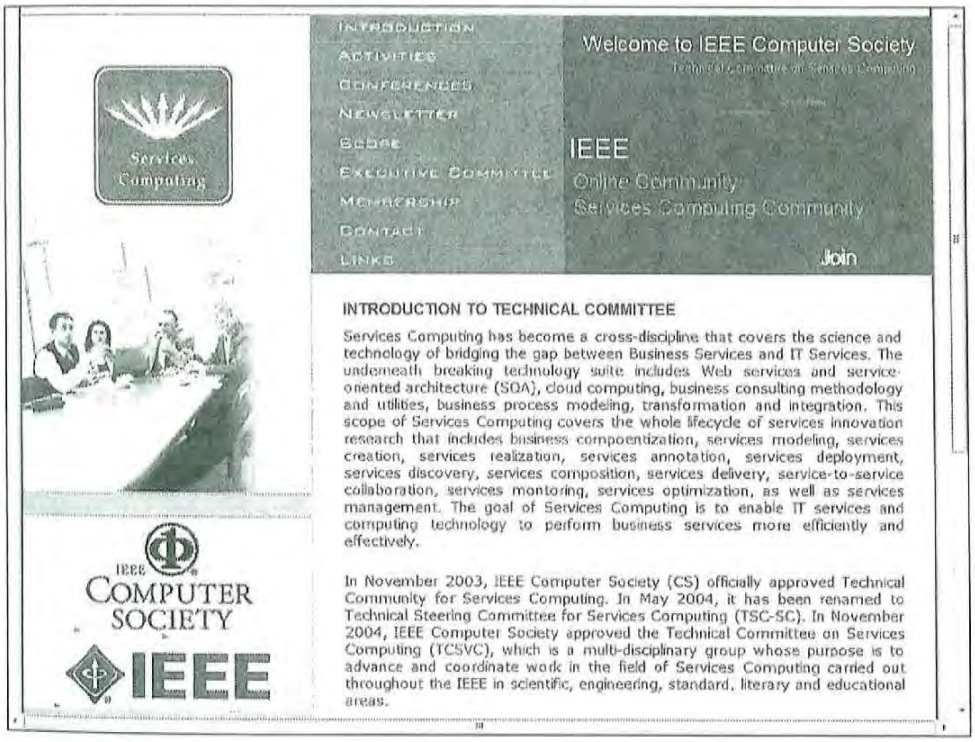


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Summary

In this chapter, you learned what is meant by the term "cloud computing." Cloud computing defines systems that are virtualized and where resources are pooled so that customers can provision computing resources as needed.

To understand cloud computing and fully appreciate its subtleties, you need to categorize the different cloud types. The two types of cloud models are those based on where the cloud is deployed and those based on the types of services that clouds offer. In this chapter, you were introduced to these cloud types.

Cloud computing has a number of benefits and has attracted great industry and general interest. Cloud computing allows systems to be created cheaply with little upfront costs and to be scaled to massive sizes, when needed. Not all applications and services benefit from cloud computing, and I presented some of the factors that help you differentiate between successful deployments and those that are not.

In Chapter 2, "Assessing the Value Proposition," the economics of cloud computing are discussed.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER

2

Assessing the Value Proposition

In this chapter, the various attributes of cloud computing that make it a unique service are described. These attributes—scalability, elasticity, low barrier to entry, and a utility type of delivery—completely change how applications are created, priced, and delivered. I describe the factors that have led to this new model of computing. Early adopters of these services are those enterprises that can best make use of these characteristics.

To get a sense for the value of cloud computing, this chapter compares it to on-premises systems. From this perspective, a number of benefits for cloud computing emerge, along with many obstacles. I describe these factors in some detail. Aside from technological reasons, behavior considerations associated with cloud adoption are discussed.

Cloud computing is particularly valuable because it shifts capital expenditures into operating expenditures. This has the benefit of decoupling growth from cash on hand or from requiring access to capital. It also shifts risk away from an organization and onto the cloud provider.

This chapter describes how to begin to measure the costs of cloud computing and some of the tools that you can use to do so. The concept of optimization known as right-sizing is described, and cloud computing has some unique new capabilities in this area.

Service Level Agreements (SLAs) are an important aspect of cloud computing. They are essentially your working contract with any provider. Cloud computing is having impact on software licensing, which although not entirely settled is also described in this chapter.

IN THIS CHAPTER

Discovering the attributes that make cloud computing unique

Applying cloud computing when it is the best option

Measuring the costs associated with cloud computing systems

Learning about Service Level Agreements and Licensing

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Measuring the Cloud's Value

Cloud computing presents new opportunities to users and developers because it is based on the paradigm of a shared multitenant utility. The ability to access pooled resources on a pay-as-you-go basis provides a number of system characteristics that completely alter the economics of information technology infrastructures and allows new types of access and business models for user applications.

Any application or process that benefits from economies of scale, commoditization of assets, and conformance to programming standards benefits from the application of cloud computing. Any application or process that requires a completely customized solution, imposes a high degree of specialization, and requires access to proprietary technology is going to expose the limits of cloud computing rather quickly. Applications that work with cloud computing are ones that I refer to as "low touch" applications; they tend to be applications that have low margins and usually low risk. The "high touch" applications that come with high margins require committed resources and pose more of a risk; those applications are best done on-premises.

A cloud is defined as the combination of the infrastructure of a datacenter with the ability to provision hardware and software. A service that concentrates on hardware follows the Infrastructure as a Service (IaaS) model, which is a good description for the Amazon Web Service described in Chapter 9. When you add a software stack, such as an operating system and applications to the service, the model shifts to the Software as a Service (SaaS) model. Microsoft's Windows Azure Platform, discussed in Chapter 10, is best described as currently using SaaS model. When the service requires the client to use a complete hardware/software/application stack, it is using the most refined and restrictive service model, called the Platform as a Service (PaaS) model. The best example of a PaaS offering is probably SalesForce.com. The Google App Engine discussed in Chapter 11 is another PaaS. As the Windows Azure Platform matures adding more access to Microsoft servers, it is developing into a PaaS model rather quickly.

Cross-Ref

Chapter 4, "Understanding Services and Applications by Type," describes a number of these XaaS service models. Cloud computing is in its wild and wooly frontier days, so it's best to take a few of the lesser known acronyms with a grain of salt. ■

A cloud is an infrastructure that can be partitioned and provisioned, and resources are pooled and virtualized. If the cloud is available to the public on a pay-as-you-go basis, then the cloud is a public cloud, and the service is described as a utility. If the cloud is captive in an organization's infrastructure (network), it is referred to as a private cloud. When you mix public and private clouds together, you have a hybrid cloud. Any analysis of the potential of cloud computing must account for all these possibilities.

These are the unique characteristics of an ideal cloud computing model:

- **Scalability:** You have access to unlimited computer resources as needed.

  This feature obviates the need for planning and provisioning. It also enables batch processing, which greatly speeds up high-processing applications.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Elasticity:** You have the ability to right-size resources as required.

  This feature allows you to optimize your system and capture all possible transactions.

- **Low barrier to entry:** You can gain access to systems for a small investment.

  This feature offers access to global resources to small ventures and provides the ability to experiment with little risk.

- **Utility:** A pay-as-you-go model matches resources to need on an ongoing basis.

  This eliminates waste and has the added benefit of shifting risk from the client.

It is the construction of large datacenters running commodity hardware that has enabled cloud computing to gain traction. These datacenters gain access to low-cost electricity, high-network bandwidth pipes, and low-cost commodity hardware and software, which, taken together, represents an economy of scale that allows cloud providers to amortize their investment and retain a profit. It has been estimated that it costs around $100 million to create a datacenter with sufficient scale to be a cloud provider. At this scale, the resources for a large datacenter have been estimated to be between 35 percent and 20 percent lower than the pricing that is offered to medium-sized datacenters.

Note

Members of the UC Berkeley Reliable Adaptive Distribute System Laboratory have published a white paper summarizing the benefits of Cloud Computing called "Above the Clouds: A Berkeley View of Cloud Computing," which can be found at `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`**. This lab was funded by contributions from the major cloud providers and from the NSF, but it's a useful source of analytical data.** ■

The virtualization of pooled resources—processors or compute engines, storage, and network connectivity—optimizes these investments and allows the cloud provider to pass along these economies to customers. Pooling also blurs the differences between a small deployment and a large one because scale becomes tied only to demand.

Companies become cloud computing providers for several reasons:

- **Profit:** The economies of scale can make this a profitable business.
- **Optimization:** The infrastructure already exists and isn't fully utilized.

  This was certainly the case for Amazon Web Services.

- **Strategic:** A cloud computing platform extends the company's products and defends their franchise.

  This is the case for Microsoft's Windows Azure Platform.

- **Extension:** A branded cloud computing platform can extend customer relationships by offering additional service options.

  This is the case with IBM Global Services and the various IBM cloud services.

- **Presence:** Establish a presence in a market before a large competitor can emerge.

  Google App Engine allows a developer to scale an application immediately. For Google, its office applications can be rolled out quickly and to large audiences.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Platform:** A cloud computing provider can become a hub master at the center of many ISV's (Independent Software Vendor) offerings.

  The customer relationship management provider SalesForce.com has a development platform called Force.com that is a PaaS offering.

The development of cloud computing has been likened to the situation that has faced hardware companies that rely on proprietary silicon to produce their products: the AMDs, nVidias, eVGAs, and Apples of the world. Because a semiconductor fabrication facility costs several billion dollars to create, these companies were at a severe disadvantage to companies such as Intel or NEC, which could build their own fabs or fabrication facilities. (A *fab* is a facility that is a self-contained semiconductor assembly line.) Companies such as TSMC (Taiwan Semiconductor Manufacturing Company) have come along that provide fabrication based on customer designs, spreading their risk and optimizing their operation. Cloud computing is much the same.

Early adopters and new applications

Cloud computing is still in its infancy, but trends in adoption are already evident. In his white paper "Realizing the Value Proposition of Cloud Computing: CIO's Enterprise IT Strategy for the Cloud," Jitendra Pal Thethi, a Principle Architect for Infosys' Microsoft Technology Group, lists the following business types as the Top 10 adopters of cloud computing:

1. Messaging and team collaboration applications
2. Cross enterprise integration projects
3. Infrastructure consolidation, server, and desktop virtualization efforts
4. Web 2.0 and social strategy companies
5. Web content delivery services
6. Data analytics and computation
7. Mobility applications for the enterprise
8. CRM applications
9. Experimental deployments, test bed labs, and development efforts
10. Backup and archival storage

You can download Thethi's paper from: `http://www.infosys.com/cloud-computing/white-papers/Documents/realizing-value-proposition.pdf`.

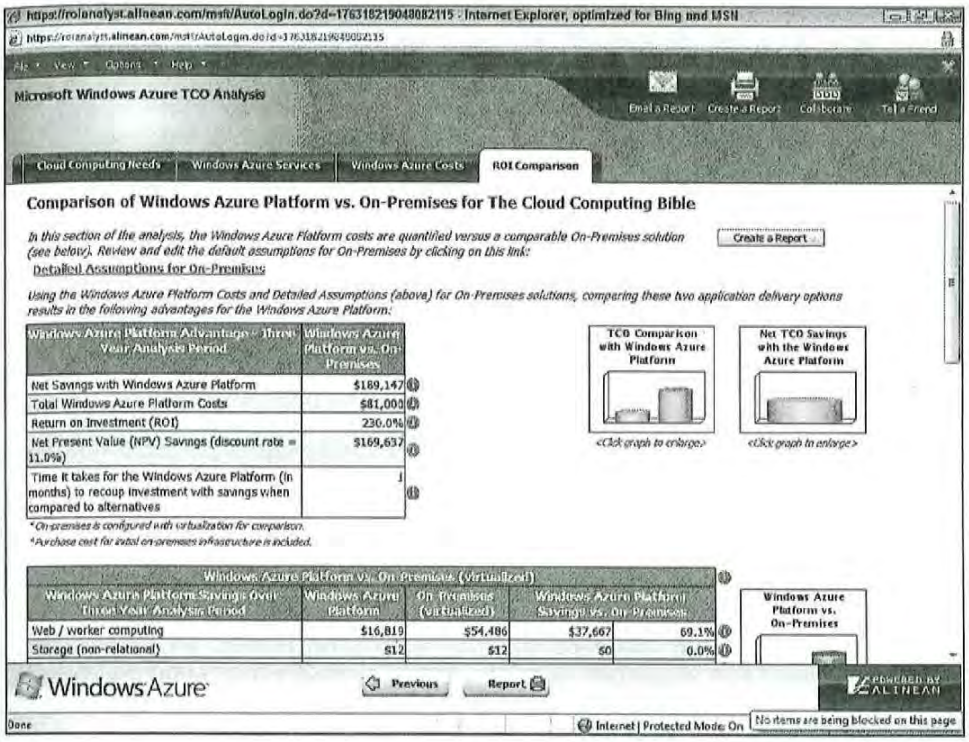
As a group, early adopters are categorized by their need for ubiquity and access to large data sets.

Around 2000-2001, some companies began using the Internet to stage various types of user-facing applications such as office suites, accounting packages, games, and so forth. The first attempts by large ISPs to create utility computing date to that period. By 2005-2006, several Internet sites had become sufficiently large that they had developed extensive infrastructure for their own sites. The excess capacity in these sites began to be offered to partners and eventually to the general public.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 2.3

The Microsoft Azure Platform ROI wizard provides a quick and dirty analysis of your TCO for a cloud deployment on Windows Azure in an attractive report format.



Specifying Service Level Agreements

A Service Level Agreement (SLA) is the contract for performance negotiated between you and a service provider. In the early days of cloud computing, all SLAs were negotiated between a client and the provider. Today with the advent of large utility-like cloud computing providers, most SLAs are standardized until a client becomes a large consumer of services.

Caution

Some SLAs are enforceable as contracts, but many are really agreements that are more along the lines of an Operating Level Agreement (OLA) and may not have the force of law. It's good to have an attorney review these documents before you make a major commitment to a cloud provider. ■

SLAs usually specify these parameters:

- Availability of the service (uptime)
- Response times or latency

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- Reliability of the service components
- Responsibilities of each party
- Warranties

If a vendor fails to meet the stated targets or minimums, it is punished by having to offer the client a credit or pay a penalty. In this regard, an SLA should be like buying insurance, and like buying insurance, getting the insurer to pay up when disaster strikes can be the devil's work.

Microsoft publishes the SLAs associated with the Windows Azure Platform components at `http://www.microsoft.com/windowsazure/sla/`, which is illustrative of industry practice for cloud providers. Each individual component has its own SLA. The summary versions of these SLAs from Microsoft are reproduced here:

- **Windows Azure SLA:** "Windows Azure has separate SLA's for compute and storage. For compute, we guarantee that when you deploy two or more role instances in different fault and upgrade domains, your Internet facing roles will have external connectivity at least 99.95% of the time. Additionally, we will monitor all of your individual role instances and guarantee that 99.9% of the time we will detect when a role instance's process is not running and initiate corrective action."

Cross-Ref

The different components of the Windows Azure Platform are discussed in detail in Chapter 10. ■

- **SQL Azure SLA:** "SQL Azure customers will have connectivity between the database and our Internet gateway. SQL Azure will maintain a "Monthly Availability" of 99.9% during a calendar month. "Monthly Availability Percentage" for a specific customer database is the ratio of the time the database was available to customers to the total time in a month. Time is measured in 5-minute intervals in a 30-day monthly cycle. Availability is always calculated for a full month. An interval is marked as unavailable if the customer's attempts to connect to a database are rejected by the SQL Azure gateway."
- **AppFabric SLA:** "Uptime percentage commitments and SLA credits for Service Bus and Access Control are similar to those specified above in the Windows Azure SLA. Due to inherent differences between the technologies, underlying SLA definitions and terms differ for the Service Bus and Access Control services. Using the Service Bus, customers will have connectivity between a customer's service endpoint and our Internet gateway; when our service fails to establish a connection from the gateway to a customer's service endpoint, then the service is assumed to be unavailable. Using Access Control, customers will have connectivity between the Access Control endpoints and our Internet gateway. In addition, for both Service Bus and Access Control, the service will process correctly formatted requests for the handling of messages and tokens; when our service fails to process a request properly, then the service is assumed to be unavailable. SLA calculations will be based on an average over a 30-day monthly cycle, with 5-minute time intervals. Failures seen by a customer in the form of service unavailability will be counted for the purpose of availability calculations for that customer."

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

You can find Google's App Engine for Business SLA at `http://code.google.com/appengine/business/sla.html`. The SLA for Amazon Web Service Elastic Computer Cloud (EC2) is published at `http://aws.amazon.com/ec2-sla/`, and the SLA for Amazon Simple Storage Service (S3) may be found at `http://aws.amazon.com/s3-sla/`.

Some cloud providers allow for service credits based on their ability to meet their contractual levels of uptime. For example, Amazon applies a service credit of 10 percent off the charge for Amazon S3 if the monthly uptime is equal to or greater than 99 percent but less than 99.9 percent. When the uptime drops below 99 percent, the service credit percentage rises to 25 percent and this credit is applied to usage in the next billing period. Amazon Web Services uses an algorithm that calculates uptime based on the following formula:

Uptime = Error Rate/Requests

as measured for each 5-minute interval during a billing period. The error rate is based on internal server counters such as "InternalError" or "ServiceUnavailable." There are exclusions that limit Amazon's exposure.

Service Level Agreements are based on the usage model. Most cloud providers price their pay-as-you-go resources at a premium and issue standard SLAs only for that purpose. You can also purchase subscriptions at various levels that guarantee you access to a certain amount of purchased resources. The SLAs attached to a subscription often offer different terms. If your organization requires access to a certain level of resources, then you need a subscription to a service. A usage model may not provide that level of access under peak load conditions.

Defining Licensing Models

When you purchase shrink-wrapped software, you are using that software based on a licensing agreement called a EULA or End User License Agreement. The EULA may specify that the software meets the following criteria:

- It is yours to own.
- It can be installed on a single or multiple machines.
- It allows for one or more connections.
- It has whatever limit the ISV has placed on its software.

In most instances, the purchase price of the software is directly tied to the EULA.

For a long time now, the computer industry has known that the use of distributed applications over the Internet was going to impact the way in which companies license their software, and indeed it has. The problem is that there is no uniform description of how applications accessed over cloud networks will be priced. There are several different licensing models in play at the moment—and no clear winners.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

It can certainly be argued that the use of free software is a successful model. The free use of software in the cloud is something that the open-source community can support, it can be supported as a line extension of a commercial product, or it can be paid for out of money obtained from other sources such as advertising. Google's advertising juggernaut has allowed it to create a portfolio of applications that users can access based on their Google accounts. Microsoft's free software in Windows Live is supported by its sales of the Windows operating system and by sales of the Microsoft Office suite.

In practice, cloud-based providers tend to license their applications or services based on user or machine accounts, but they do so in ways that are different than you might expect based on your experience with physical hardware and software. Many applications and services use a subscription or usage model (or both) and tie it to a user account. Lots of experimentation is going on in the publishing industry on how to price Internet offerings, and you can find the same to be true in all kinds of computer applications at the moment. Some services tie their licenses into a machine account when it makes sense. An example is the backup service Carbonite, where the service is for backing up a licensed computer. However, cloud computing applications rarely use machine licenses when the application is meant to be ubiquitous. If you need to access an application, service, or Web site from any location, then a machine license isn't going to be practical.

The impact of cloud computing on bulk software purchases and enterprise licensing schemes is even harder to gauge. Several analysts have remarked that the advent of cloud computing could lead to the end of enterprise licensing and could cause difficulties for software vendors going forward. It isn't clear what the impact on licensing will be in the future, but it is certainly an area to keep your eyes on.

Summary

In this chapter, you learned about the features that make cloud computing unique. It is both a new model and a new platform for computing. The idea of computing as a utility is as old as the computer industry itself, but it is the advent of low-cost datacenters that have really enabled this platform to thrive.

A cloud's unique features are scalability, elasticity, low barrier to entry, and a utility delivery of services. These features completely change the way in which applications and services are used. Cloud computing will enable the development of new types of applications, several of which were described in this chapter. Undoubtedly many new application types will arise that will be a complete surprise to us all. However, cloud computing has some limitations that were discussed as well.

To help you get a handle on cloud computing, this chapter stresses measurements of costs in comparison to private or on-premises systems. Cloud computing shifts capital expenditures into operating expenditures.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Cloud computing changes the nature of the service provider and its relationship to its client. You see this expressed in the Service Level Agreements (SLAs) and software licensing that are part of this new developing industry. There are many changes to come in these areas.

Chapter 3, "Understanding Cloud Architecture," is an in-depth look at the superstructure and plumbing that makes cloud computing possible. In that chapter, you learn about the standards and protocols used by the cloud computing industry.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER

3

Understanding Cloud Architecture

Cloud computing is a natural extension of many of the design principles, protocols, plumbing, and systems that have been developed over the past 20 years. However, cloud computing describes some new capabilities that are architected into an application stack and are responsible for the programmability, scalability, and virtualization of resources. One property that differentiates cloud computing is referred to as composability, which is the ability to build applications from component parts.

A platform is a cloud computing service that is both hardware and software. Platforms are used to create more complex software. Virtual appliances are an important example of a platform, and they are becoming a very important standard cloud computing deployment object.

Cloud computing requires some standard protocols with which different layers of hardware, software, and clients can communicate with one another. Many of these protocols are standard Internet protocols. Cloud computing relies on a set of protocols needed to manage interprocess communications that have been developed over the years. The most commonly used set of protocols uses XML as the messaging format, the Simple Object Access Protocol (SOAP) protocol as the object model, and a set of discovery and description protocols based on the Web Services Description Language (WSDL) to manage transactions.

Some completely new clients are under development that are specifically meant to connect to the cloud. These clients have as their focus cloud applications and services, and are often hardened and more securely connected. Two examples presented are Jolicloud and Google Chrome OS. They represent a new client model that is likely to have considerable impact.

IN THIS CHAPTER

Using the cloud computing stack to describe different models

Understanding how platforms and virtual appliances are used

Learning how cloud communications work

Discovering the new world of the cloud client

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Exploring the Cloud Computing Stack

Cloud computing builds on the architecture developed for staging large distributed network applications on the Internet over the last 20 years. To these standard networking protocols, cloud computing adds the advances in system virtualization that became available over the last decade. The cloud creates a system where resources can be pooled and partitioned as needed. Cloud architecture can couple software running on virtualized hardware in multiple locations to provide an on-demand service to user-facing hardware and software. It is this unique combination of abstraction and metered service that separates the architectural requirements of cloud computing systems from the general description given for an n-tiered Internet application.

Many descriptions of cloud computing describe it in terms of two architectural layers:

A client as a front end

The "cloud" as a backend

This is a very simplistic description because each of these two components is composed of several component layers, complementary functionalities, and a mixture of standard and proprietary protocols. Cloud computing may be differentiated from older models by describing an encapsulated information technology service that is often controlled through an Application Programming Interface (API), thus modifying the services that are delivered over the network.

A cloud can be created within an organization's own infrastructure or outsourced to another datacenter. While resources in a cloud can be real physical resources, more often they are virtualized resources because virtualized resources are easier to modify and optimize. A compute cloud requires virtualized storage to support the staging and storage of data. From a user's perspective, it is important that the resources appear to be infinitely scalable, that the service be measurable, and that the pricing be metered.

Composability

Applications built in the cloud often have the property of being built from a collection of components, a feature referred to as composability. A composable system uses components to assemble services that can be tailored for a specific purpose using standard parts. A composable component must be:

- **Modular:** It is a self-contained and independent unit that is cooperative, reusable, and replaceable.
- **Stateless:** A transaction is executed without regard to other transactions or requests.

It isn't an absolute requirement that transactions be stateless, some cloud computing applications provide managed states through brokers, transaction monitors, and service buses. In rarer cases, full transactional systems are deployed in the clouds, but these systems are harder to architect in a distributed architecture.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Although cloud computing doesn't require that hardware and software be composable, it is a highly desirable characteristic from a developer or user's standpoint, because it makes system design easier to implement and solutions more portable and interoperable.

There is a tendency for cloud computing systems to become less composable for users as the services incorporate more of the cloud computing stack. From the standpoint of an IaaS (Infrastructure as a Service) vendor such as Amazon Web Services, GoGrid, or Rackspace, it makes no sense to offer non-standard machine instances to customers, because those customers are almost certainly deploying applications built on standard operating systems such as Linux, Windows, Solaris, or some other well-known operating system.

In the next step up the cloud computing stack, PaaS (Platform as a Service) vendors such as Windows Azure or Google AppEngine may narrow the definition of standard parts to standard parts that work with their own platforms, but at least from the standpoint of the individual platform service provider, the intent is to be modular for their own developers.

When you move to the highest degree of integration in cloud computing, which is SaaS (Software as a Service), the notion of composability for users may completely disappear. An SaaS vendor such as Quicken.com or Salesforce.com is delivering an application as a service to a customer, and there's no particular benefit from the standpoint of the service provider that the customer be able to compose its own custom applications. A service provider reselling an SaaS may have the option to offer one module or another, to customize the information contained in the module for a client, to sell the service under their own brand, or to perform some other limited kind of customization, but modifications are generally severely limited.

This idea that composability diminishes going up the cloud computing stack is from the user's point of view. If you are a PaaS or SaaS service provider and your task is to create the platform or service presented to the developer, reseller, or user, the notion of working with a composable system is still a very powerful one. A PaaS or SaaS service provider gets the same benefits from a composable system that a user does—these things, among others:

- Easier to assemble systems
- Cheaper system development
- More reliable operation
- A larger pool of qualified developers
- A logical design methodology

You encounter the trend toward designing composable systems in cloud computing in the widespread adoption of what has come to be called the Service Oriented Architecture (SOA). The essence of a service oriented design is that services are constructed from a set of modules using standard communications and service interfaces. An example of a set of widely used standards describes the services themselves in terms of the Web Services Description Language (WSDL), data exchange between services using some form of XML, and the communications between the services using the SOAP protocol. There are, of course, alternative sets of standards.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Cross-Ref

SOA is described in detail in Chapter 13. ■

What isn't specified is the nature of the module itself; it can be written in any programming language the developer wants. From the standpoint of the system, the module is a black box, and only the interface is well specified. This independence of the internal workings of the module or component means it can be swapped out for a different model, relocated, or replaced at will, provided that the interface specification remains unchanged. That is a powerful benefit to any system or application provider as their products evolve.

Infrastructure

Most large Infrastructure as a Service (IaaS) providers rely on virtual machine technology to deliver servers that can run applications. Virtual servers described in terms of a machine image or instance have characteristics that often can be described in terms of real servers delivering a certain number of microprocessor (CPU) cycles, memory access, and network bandwidth to customers. Virtual machines are containers that are assigned specific resources. The software that runs in the virtual machines is what defines the utility of the cloud computing system.

Figure 3.1 shows the portion of the cloud computing stack that is defined as the "server." In the diagram, the API is shown shaded in gray because it is an optional component that isn't always delivered with the server. The VMM component is the Virtual Machine Monitor, also called a hypervisor. This is the low-level software that allows different operating systems to run in their own memory space and manages I/O for the virtual machines.

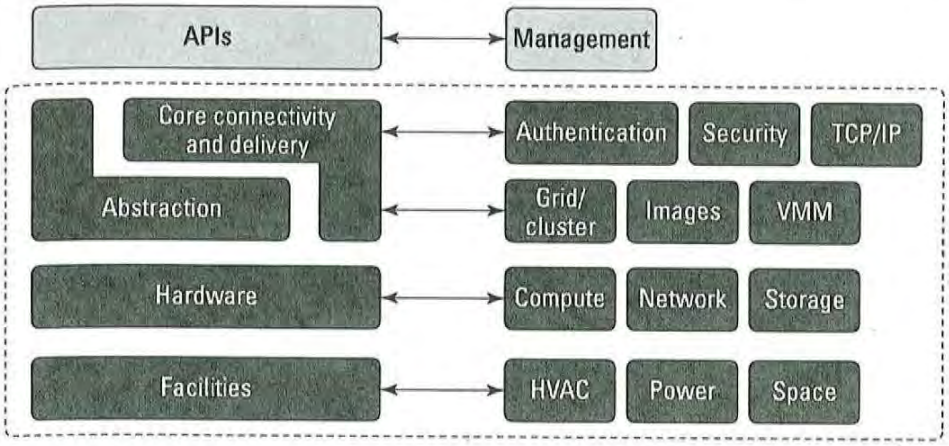
The notion of a virtual server presents to an application developer a new way of thinking about and programming applications. For example, when a programmer is creating software that requires several different tasks to be performed in parallel, he might write an application that creates additional threads of execution that must be managed by the application. When a developer creates an application that uses a cloud service, the developer can attach to the appropriate service(s) and allow the application itself to scale the program execution. Thus, an application such as a three-dimensional rendering that might take a long time for a single server to accomplish can be scaled in the cloud to many servers at once for a short period of time, accomplishing the task at a similar or lower price but at a much faster rate.

In future applications, developers will need to balance the architectural needs of their programs so their applications create new threads when it is appropriate or create new virtual machines. Applications will also need to be mindful of how they use cloud resources, when it is appropriate to scale execution to the cloud, how to monitor the instances they are running, and when not to expand their application's usage of the cloud. This will require a new way of thinking about application development, and the ability to scale correctly is something that will have to be architected into applications from the ground up.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 3.1

This architectural diagram illustrates the portion of the cloud computing stack that is designated as the server.



Platforms

A platform in the cloud is a software layer that is used to create higher levels of service. As you learned in Chapter 1, many different Platform as a Service (PaaS) providers offer services meant to provide developers with different capabilities. In Chapter 7, PaaS is explored more thoroughly, but for now it is useful to cite three of the major examples that are provided in this book:

- Salesforce.com's Force.com Platform
- Windows Azure Platform
- Google Apps and the Google AppEngine

These three services offer all the hosted hardware and software needed to build and deploy Web applications or services that are custom built by the developer within the context and range of capabilities that the platform allows. Platforms represent nearly the full cloud software stack, missing only the presentation layer that represents the user interface. This is the same portion of the cloud computing stack that is a virtual appliance and is shown in Figure 3.2. What separates a platform from a virtual appliance is that the software that is installed is constructed from components and services and controlled through the API that the platform provider publishes.

It makes sense for operating system vendors to move their development environments into the cloud with the same technologies that have been successfully used to create Web applications. Thus, you might find a platform based on a Sun xVM hypervisor virtual machine that includes a NetBeans Integrated Development Environment (IDE) and that supports the Sun GlassFish

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Web stack programmable using Perl or Ruby. For Windows, Microsoft would be similarly interested in providing a platform that allowed Windows developers to run on a Hyper-V VM, use the ASP.NET application framework, support one of its enterprise applications such as SQL Server, and be programmable within Visual Studio—which is essentially what the Azure Platform does. This approach allows someone to develop a program in the cloud that can be used by others.

Platforms often come replete with tools and utilities to aid in application design and deployment. Depending upon the vendor, you may find developer tools for team collaboration, testing tools, instrumentation for measuring program performance and attributes, versioning, database and Web service integration, and storage tools. Most platforms begin by establishing a developer community to support the work done in the environment.

Note

To see the entire cloud computing stack, refer to Figure 1.5 in Chapter 1. ■

FIGURE 3.2

A virtual appliance is software that installs as middleware onto a virtual machine.

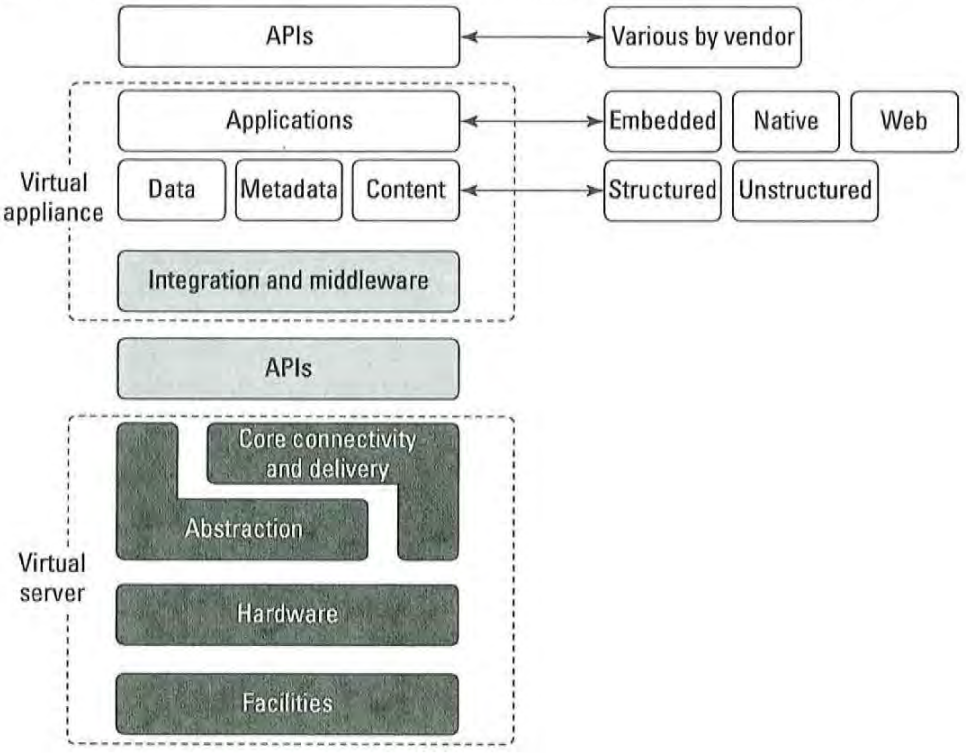
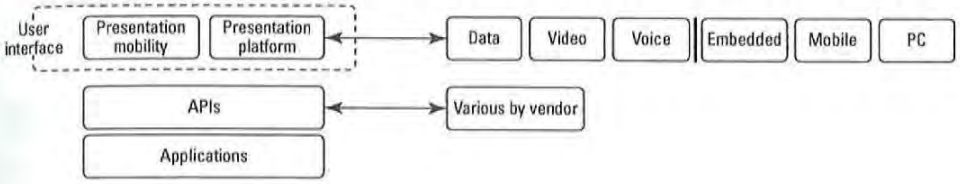


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Just as a virtual appliance may expose itself to users through an API, so too an application built in the cloud using a platform service would encapsulate the service through its own API. Users would then interact with the platform, consuming services through that API, leaving the platform to manage and scale the service appropriately. Many platforms offer user interface development tools based on HTML, JavaScript, or some other technology. As the Web becomes more media-oriented, many developers have chosen to work with rich Internet environments such as Adobe Flash, Flex, or Air, or alternatives such as Windows Silverlight. A user interface abstracts away the platform API, making those services managed through the UI. Figure 3.3 shows the top portion of the cloud computing stack, which includes the API and the presentation functionality.

FIGURE 3.3

The top of the cloud computing interface includes the user interface and the API for the application layer.



The Application Programming Interface is one of the key differentiators separating cloud computing from the older models of Internet applications, because it is the means for instantiating resources needed to support applications. An API can control data flow, communications, and other important aspects of the cloud application. Unfortunately, each cloud vendor has their own cloud API, none of them are standard, and the best you can hope for is that eventually the major cloud vendor's APIs will interoperate and exchange data. For now, the use of proprietary APIs results in vendor lock-in, which is why you are advised to choose systems that implement APIs based on open standards.

Virtual Appliances

Applications such as a Web server or database server that can run on a virtual machine image are referred to as virtual appliances. The name *virtual appliance* is a little misleading because it conjures up the image of a machine that serves a narrow purpose. Virtual appliances are software installed on virtual servers—application modules that are meant to run a particular machine instance or image type. A virtual appliance is a platform instance. Therefore, virtual appliances occupy the middle of the cloud computing stack (refer to Figure 3.2).

A virtual appliance is a common deployment object in the cloud, and it is one area where there is considerable activity and innovation. One of the major advantages of a virtual appliance is that you can use the appliances as the basis for assembling more complex services, the appliance being one of your standardized components. Virtual appliances remove the need for application configuration and maintenance from your list of system management chores.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

You run across virtual appliances in IaaS systems such as Amazon's Elastic Compute Cloud (EC2), which is discussed in detail in Chapter 9. Amazon Machine Images are virtual appliances that have been packaged to run on the gird of Xen nodes that comprise the Amazon Web Service's EC2 system. Shown in Figure 3.4, the AMI library (`http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=171`) includes a variety of operating systems both proprietary and open source, a set of enterprise applications such as Oracle BPM, SQL Server, and even complete application stacks such as LAMP (Linux, Apache, MySQL, and PHP). Amazon has negotiated licenses from these vendors that are part of your per-use pricing when you run these applications on their servers.

Virtual appliances are far easier to install and run than an application that you must set up yourself. However, virtual appliances are also much larger than the application themselves would be because they are usually bundled with the operating system on which they are meant to run. An application that is 50 or 100MB might require a virtual appliance that is 500MB to 1GB in size. Usually, when a virtual appliance is created, the operating system is stripped of all excess functionality that isn't required by the appliance, because the appliance is meant to be used as is.

FIGURE 3.4

Amazon Machine Images are a collection of virtual appliances that you can install on their Xen hypervisor servers.

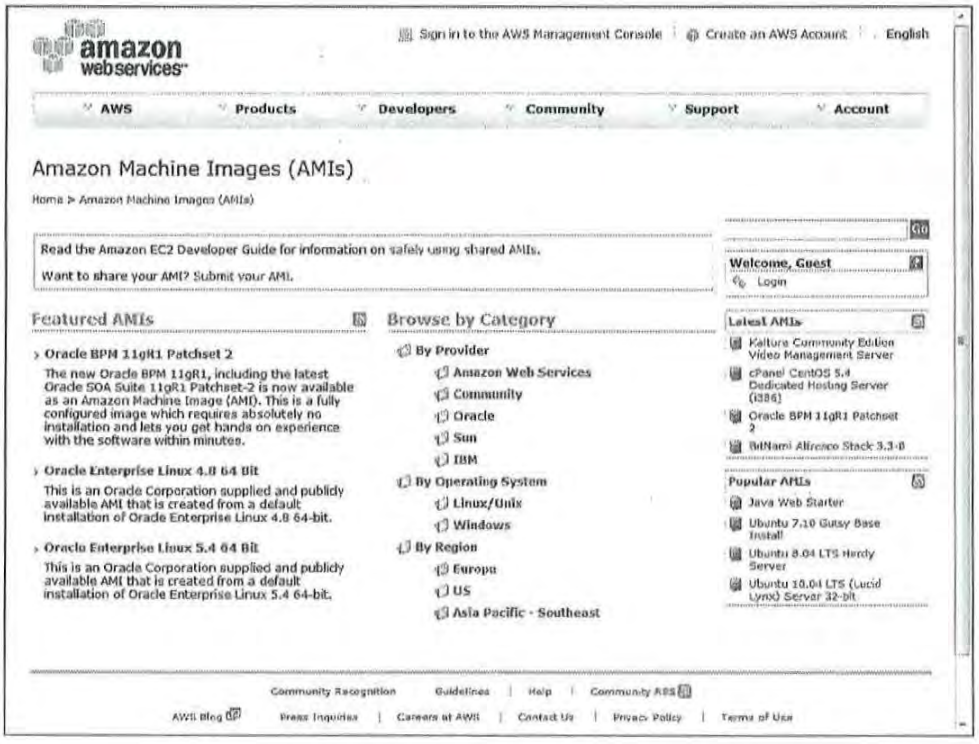


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Virtual appliances have begun to affect the PC industry in much the same way that application stores have affected the cell phone industry. You can find various Web sites that either sell or distribute ready-to-use virtual appliances in various forms. Perhaps the best developed of these marketplaces is VMware's Virtual Appliances site (`http://www.vmware.com/appliances/`) shown in Figure 3.5. These appliances are certified by VMware to be ready to use in the enterprise.

Among the other places you can find virtual appliances are at the Web sites of the various operating system vendors, such as Ubuntu, Xen (`http://www.xen.org/`), and others, including these:

- **Bagvapp** (`http://bagside.com/bagvapp/`) offers virtual appliances, including ones based on Windows, all of which run on VMware Player.
- **HelpdeskLive** (`http://helpdesklive.info/download/VirtualBox%20VDI%20free%20images.html`) offers various Linux distributions upon which you can build a virtual machine.
- **Jcinacio** (`http://www.jcinacio.com/`) has Ubuntu appliances.

FIGURE 3.5

VMware's Virtual Appliance marketplace (`http://www.vmware.com/appliances/`) sells virtual appliances that run on VMware's hypervisor in cloud computing applications.



Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Jumpbox** (`http://www.jumpbox.com`) offers open source virtual appliances installed by them as a managed service. Jumpbox offers virtual appliances for many applications including Bugzilla, DokuWiki, Drupal, Joomla!, Nagios, OpenVPN, PostgreSQL, Redmine, WordPress, and many others. Figure 3.6 shows the Jumpbox home page.
- **QEMU** (`http://www.qemu.org/`) is a CPU emulator and virtual machine monitor.
- **Parallels** (`http://ptn.parallels.com/ptn`) hosts a variety of appliances that includes Linux distros, server software, and other products.
- **ThoughtPolice** (`http://www.thoughtpolice.co.uk/vmware/`) offers appliances based on a variety of Linux distributions.
- **VirtualBox** (`http://www.virtualbox.org/`) is a virtual machine technology now owned by Oracle that can run various operating systems and serves as a host for a variety of virtual appliances.
- **Vmachines** (`http://www.vmachines.net/`) is a site with desktop, server, and security-related operating systems that run on VMware.

FIGURE 3.6

Jumpbox (`http://www.jumpbox.com/`) is an open-source virtual appliance installation and management service.



Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Converting a virtual appliance from one platform to another isn't an easy proposition. Efforts are underway to create file format standards for these types of objects that make this task easier. The best known of these file formats is the Open Virtualization Format (OVF), the work of the Distributed Management Task Force (DMTF) group. Nearly all major virtualization platform vendors support OVF, notably VMware, Microsoft, Oracle, and Citrix.

Cross-Ref

Chapter 5 describes virtual appliances in more detail and gives some examples of vendors offering these types of applications for cloud computing infrastructure development. ■

Communication Protocols

Cloud computing arises from services available over the Internet communicating using the standard Internet protocol suite underpinned by the HTTP and HTTPS transfer protocols. The other protocols and standards that expose compute and data resources in the cloud either format data or communications in packets that are sent over these two transport protocols.

In order to engage in interprocess communication (IPC) processes, many client/server protocols have been applied to distributed networking over the years. Various forms of RPC (Remote Procedure Call) implementations (including DCOM, Java RMI, and CORBA) attempt to solve the problem of engaging services and managing transactions over what is essentially a stateless network. The first of the truly Web-centric RPC technologies was XML-RPC, which uses platform-independent XML data to encode program calls that are transported over HTTP, the networking transport to which nearly everyone is connected.

Note

You can find a full description of the common Internet protocol standards in *Networking Bible* by Barrie Sosinsky, Wiley, 2009. These protocols form the basis for much of the discussion in any good networking textbook. ■

As Internet computing became more firmly entrenched over the last decade, several efforts began to better define methods for describing and discovering services and resources. The most widely used message-passing standard at the moment is the Simple Object Access Protocol (SOAP), which essentially replaces XML-RPC. SOAP uses XML for its messages and uses RPC and HTTP for message passing. SOAP forms the basis for most of the Web services stacks in use today. If you examine the XML file used in a SOAP transaction, you find that it contains a message and the instructions on how to use the message. The message has a set of rules that are translated into application instances and datatypes, and it defines the methods that must be used to initiate procedure calls and then return a response.

Several standards have emerged to allow the discovery and description of Web-based resources. The most commonly used model for discovery and description used with SOAP messaging is the Web Services Description Language (WSDL), a World Wide Web Consortium (`http://www.w3.org/2002/ws/desc/`) Internet standard. WSDL lets a Web service advertise itself in terms of a collection of endpoints or ports associated with a specific network address (URL) that can be

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

addressed using XML messages to provide a service. In WSDL, a service is a container that performs a set of functions that are exposed to Web protocols. Taken together, the protocol and port are a binding to which messages are passed and operations are performed. A bound service is one that responds to any valid HTTP request sent to it. The important thing to remember about WSDL is that it defines a Web service's public interface.

Cross-Ref

Chapter 13, which describes Service Oriented Architecture, continues the discussion of these various components in building an application in the cloud from a component-based architecture. ■

Using WSDL and SOAP, a number of extensions were created that allow various Web services to describe additional sets of properties and methods that they could provide. These extensions fall under the name WS-*, or the "WS-star" specifications. A number of WS-* extensions are in common use, with the following being the most widely used:

- WS-Addressing
- WS-Discovery
- WS-Eventing
- WS-Federation
- WS-MakeConnection
- WS-Messaging
- WS-MetadataExchange
- WS-Notification
- WS-Policy
- WS-ResourceFramework
- WS-Security
- WS-Transfer
- WS-Trust

These different specifications provide a standard means of adding metadata to a SOAP message by modifying the message header while maintaining the message body structure. In this way, a standard method for metadata exchange is piggybacked onto the WSDL XML message. Each of these different WS-* specifications is in a different state of development.

You use these various WS-* services in your daily work. For example, the Web Services Dynamic Discovery specification (WS-Discovery) is a specification for multicast discovery on a LAN (Local Area Network) that is extended to Web services, most often as SOAP over UDP (User Datagram Protocol). When you open the Network Neighborhood in Windows and use the People Near Me feature, WS-Discovery goes into action and shows you discoverable resources.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

These WS-* services carried over XML messages using the SOAP protocol access remote server applications in ways that are becoming increasingly complex. Whereas earlier methods for client/server provided a means through a gateway like CGI to access media content on servers, the current data communications burden servers with accepting and processing very complex requests or engaging their clients in sophisticated negotiations that seek to minimize the amount of processing that must be done and the information that must be exchanged as the response. None of this type of rich media servicing was ever envisaged in the construction of the Internet, and all of it is essentially a kludge.

Over the years, a variety of platform-specific RPC specifications, such as DCOM (Distributed Common Object Model) and CORBA (Common Object Response Broker Architecture), were developed to allow software components that ran on different computers to interoperate with one another. As SOAP and WS-* were developing, those protocols began to build into their specifications server application features from these other technologies in a more platform-independent protocol. What was really needed was a method for standardizing resources on the Web, which is where the idea of REST comes in.

REST stands for Representational State Transfer, and it owes its original description to the work of Roy Fielding, who was also a co-developer of the HTTP protocol. REST assigns a global identifier to a resource so there is a uniform method for accessing information sources. That identifier is a URI expressed in HTTP form. Given a resource then at a known address, various network clients in the form of what are called *user agents* can then communicate with that resource using HTTP commands (requests) to exchange information in the form of documents or files. Typical data transfers might use XML, text, an image file, a JSON document, or some other standard or agreed upon format to perform the data exchange. A transaction following the rules of REST is therefore considered to be RESTful, and this is the basis for cloud computing transactions to be initiated, processed, and completed in most modern implementations.

While REST is heavily used, it is not the only data interchange standard that is used by cloud services. Another example of a data exchange standard is the Atom or Atom Publishing Protocol (APP). Atom is a syndication format that allows for HTTP protocols to create and update information. Microsoft's ADO.NET Data Services Framework is another system for transferring data using a RESTful transaction and standard HTTP commands.

Cloud services span the gamut of computer applications: audio and video streaming, instant messaging, and so forth. Each of these areas uses protocols developed for network use and adapted for use by Web services. The impact of cloud computing on network communication is to encourage the use of open-network protocols in place of proprietary protocols.

For example, in the area of instant messaging, which is a major cloud service, the SIMPLE protocol (which stands for the Session Initiated Protocol for Instant Messaging and Presence Leveraging Extensions) is in widespread use today. SIMPLE is based on the IETF Session Initiation Protocol (SIP) standard and is an open standard. Although most early IM (Instant Messaging) services used proprietary standards, many IM services now support SIMPLE because without this support, it would be hard for the different services to interoperate. Similarly, in the area of VoIP, you find that the open XMPP or the Extensible Messaging and Presence Protocol is used.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Applications

Although the cloud computing stack encompasses many details that describe how clouds are constructed, it is not a perfect vehicle for expressing all the considerations that one must account for in any deployment. An important omission arises from the nature of distributed Web applications and the design of Internet protocols as a stateless service. The Internet was designed to treat each request made to a server as an independent transaction. Therefore, the standard HTTP commands are all atomic in nature: GET to read data, PUT to writedata, and so on.

While stateless servers are easier to architect and stateless transactions are more resilient and can survive outages, much of the useful work that computer systems need to accomplish are stateful. Here's the classic example. When you go to a reservation system to purchase something, you query inventory, reserve the item, and then pay for it. In a multiuser system, if you don't have a stateful system, you cannot know whether the item you reserved has already been taken by another user before you can enter your payment for the item. Should you decide you don't want the item at some later time, it is much easier to restore the item to inventory and return payments or make other adjustments if you can roll back all the steps as a transactional unit.

Much of the really hard development efforts that have gone into making the Web useful in commerce have centered around creating mechanisms to change a set of stateless transactions into stateful ones. The development of transaction servers, message queuing servers, and other middleware is meant to bridge this problem. Cloud computing is no exception to this problem, and to an extent it amplifies the problem by not only making transactions stateless but also virtualizing resources so transactions are always occurring in physically different locations. In cloud computing, a variety of constructs are brought to bear to solve these issues, but these are the two most important concepts:

- The notion of orchestration—that process flow can be choreographed as a service
- The use of what is referred to as a service bus that controls cloud components

In time, other methods for establishing transactional integrity may be developed that are better suited to cloud computing, but these are the standard methods that are now part of the Service Oriented Architecture. Chapter 13 takes up this topic in more detail, but it is something you should certainly keep in mind as you read through the intervening chapters.

Connecting to the Cloud

Clients can connect to a cloud service in a number of different ways. These are the two most common means:

- A Web browser
- A proprietary application

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

These applications can be running on a server, a PC, a mobile device, or a cell phone. What these devices have in common with either of these application types is that they are exchanging data over an inherently insecure and transient medium. There are three basic methods for securely connecting over a connection:

- Use a secure protocol to transfer data such as SSL (HTTPS), FTPS, or IPsec, or connect using a secure shell such as SSH to connect a client to the cloud.
- Create a virtual connection using a virtual private network (VPN), or with a remote data transfer protocol such as Microsoft RDP or Citrix ICA, where the data is protected by a tunneling mechanism.
- Encrypt the data so that even if the data is intercepted or sniffed, the data will not be meaningful.

The best client connections use two or more of these techniques to communicate with the cloud. In current browser technology, clients rely on the Web service to make available secure connections, but in the future, it is likely that cloud clients will be hardened so the client itself enforces a secure connection.

If you've ever logged into a hotel connection and browsed the network, you may find that often you can access systems on the network that haven't been protected with a firewall; an improperly configured firewall connection to the cloud is even worse. That has led people to drag portable routers with them, which provide a personal hardware firewall; many of these devices have VPN built directly into them.

Other solutions include using VPN software; here are three recommended solutions:

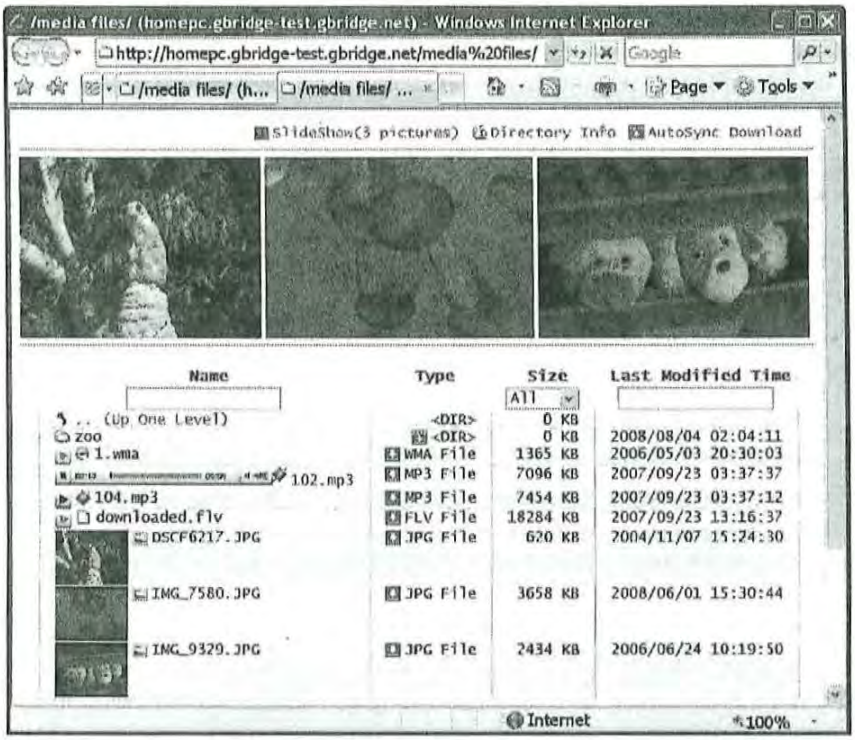
- Hotspot VPN (`http://www.hotspotvpn.com/`)
- AnchorFree Hotspot Shield (`http://hotspotshield.com/`)
- Gbridge (`http://www.gbridge.com/`), a third-party VPN based on Google's GoogleTalk infrastructure

Gbridge is an interesting solution that illustrates the use of VPN over a cloud connection. To use this product, you need to log into the GoogleTalk (or Gtalk) network and connect to another computer using your Google account. Gbridge allows additional people to join a connection when invited and supports collaborative features such as desktop sharing using the Virtual Network Computing (VNC) software, chat, live folder browsing, folder synchronization, and automated backup. Gbridge also works with applications deployed using Google Apps, allowing you to securely connect to these applications using a VPN. Figure 3.7 shows browsing a folder over a VPN connection using Gbridge's SecureShares feature.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 3.7

Gbridge provides a means for securely connecting one computer to another using Gtalk. Shown here is the SecureShares folder-browsing feature.



The Jolicloud Netbook OS

The popularity of ultralight netbooks and mobile phones has greatly expanded the potential audience of dedicated cloud computing devices, but until recently these devices ran standard operating systems such as Windows, Linux, and Macintosh on the PC and Android, IOS, and Windows Mobile (among others) on cell phones. The primary differentiation between these devices is whether or not they are capable of running video and animation (particularly Adobe Flash).

None of these portable devices has been optimized to connect securely to the cloud by narrowing the operating system functions to harden these devices. The French firm Jolicloud (`http://www.jolicloud.com/`) has recently released a lightweight version of Linux designed specifically to run connected to the cloud as a dedicated cloud client. Jolicloud 1.0 ("The Anywhere OS") can be loaded onto a netbook as the only operating system, or it can be set up as a dual boot system that shares files with a Windows partition.

Jolicloud concentrates on building a social platform with automatic software updates and installs. The application launcher is built in HTML 5 and comes preinstalled with Gmail, Skype, Twitter,

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Firefox, and other applications. Any HTML 5 browser can be used to work with the Jolicloud interface. Jolicloud maintains a library or App Directory of over 700 applications as part of an app store. When you click to select an application, the company both installs and updates the application going forward, just as the iPhone manages applications on that device. Figure 3.8 shows the Jolicloud interface.

FIGURE 3.8

The Jolicloud cloud client operating system is a social networking platform for netbooks with a dedicated application store.



When you install Jolicloud on multiple devices, the system automatically synchronizes your applications so you are working with the same content on all your devices. You can manage your devices from any cloud-connected device. Your files are also unified in a single location, and the operating system provides access to shared storage cloud services such as box.net, Dropbox, and drop.io, among other services.

Chromium OS: The Browser as an Operating System

The Google Chrome OS is a Linux open-source operating system designed to be a robust cloud client. Unlike many other Linux distributions, Google's Chrome is not a software installation, but is shipped installed on validated hardware from Google-approved OEMs (Original Equipment Manufacturers), just as the Android operating system is shipped on a variety of phones. The intent is to have a tightly coupled hardware offering that supports features in the Chrome OS and that would be highly efficient. Early designs have shown Chrome running on tablet designs that would position it as an Apple iPad competitor running on netbook-type devices in the $300-400 range.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Note

An OEM or original equipment manufacturer builds systems from components and sells them under a brand name. ■

The expectation is that the first versions of Chrome systems will appear in late 2010, perhaps in both consumer and enterprise offerings. There is also an open-source version of this cloud client called Chromium (`http://www.chromium.org/chromium-os`), which shares the same code base. The Chromium architecture is built as a three-tier system with a hardware layer, the browser and window manager, and a set of system software and utilities.

The Chrome OS has been described as a hardened operating system because it incorporates a sandbox architecture for running applications and also performs automatic updates. Also included in the system is a version of remote desktop connection software that creates an encrypted connection like Microsoft's RDP, Citrix's ICA, or a VNC client. The Chrome OS hardware specification includes a Trusted Platform Module, which provides for a "trusted bootpath" along with a hardware switch that can be used to boot the system into a developer model. In that mode, some of the security features are turned off, allowing the user to reset the system.

Demonstrations of early prototypes of Chrome and Chromium OS systems have shown that they are capable of nearly instantaneous startup. Chromium Linux kernel has adopted the Upstart (`http://upstart.ubuntu.com/`) event-based replacement for the `init` daemon, which is used to launch services concurrently, restore stalled jobs, and perform delayed system startup. The fast boot time is possible because the device is devoid of most of the devices in modern PCs. Chromium has also adopted a set of security routines in firmware that run during startup and store the information necessary to perform verified system restoration.

Essentially, the Chrome OS looks like the Chrome browser. Chrome is interesting because Google has essentially stripped down the operating system to run one specific application that connects to the Internet. The user interface is similar to the Chrome Web browser and includes a media player that plays MP3, views JPEGs, and plays media content both online and offline. Adobe Flash is integrated directly into the Chrome OS, just as it is in the Chrome browser. When you launch Chrome, you see links to the major Google cloud applications such as Gmail, Google Apps, and YouTube, along with other major sites such as Facebook, Hulu, Pandora, Twitter, and others. Figure 3.9 shows an early demonstration of the Chrome OS, its multi-tab interface, and its application launcher utility. From this same demonstration is shown the Google Reader application with a page from *Alice in Wonderland,* displayed in Figure 3.10.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Google will include on Chromium the Google Cloud Print service that allows an application to print to any connected printer without accessing a printer driver. This system frees Chromium from having to develop hardware- and OS-specific print subsystems. Instead, a proxy is installed in Chrome that registers a printer with the service, and this proxy manages print jobs for the user.

The Chrome OS devices that appear, as well as the competitors such as the Apple iPad and a host of other similar devices from other system vendors, signal a sea change in the manner in which users access the cloud, and they represent the cloud's impact on the manner in which many users perform their daily work. It's anyone's guess how impactful these introductions will be, but it is clear that they are not simply another competitor to Windows and Macintosh desktop-oriented systems. They represent the move into a cloud-based future where applications run and data is stored remotely. Their success is likely to be contingent upon how fast consumers and businesses become comfortable with the idea of outsourcing these functions. In time, the transition is probably inevitable because the economies of scale and efficiency that cloud computing offers is too compelling to ignore.

FIGURE 3.9

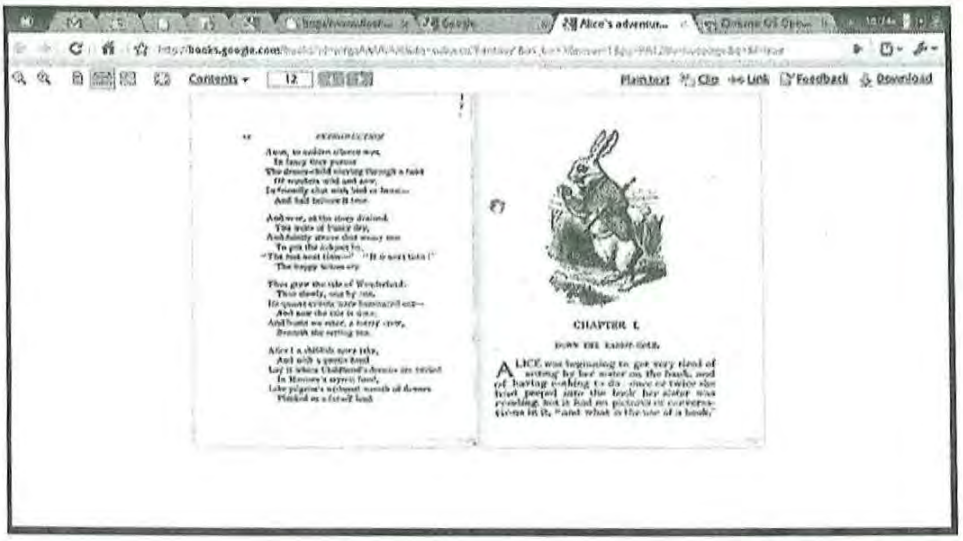
The Chrome OS operating system's application launcher from an early demo of the product found at `http://www.youtube.com/watch?v=ANMrzw7JFzA&feature=channel`



Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 3.10

From the same source as Figure 3.9, this figure illustrates how the Google Reader appears within the Chrome OS as a tab.



Summary

In this chapter, you learned about the various architectural layers that make up the cloud computing stack. Cloud computing may be seen to be an extension of older Internet standards, but it includes some new architecture. In particular, infrastructure components were described, as were platform components. Platform services are software-based and can include a class of applications referred to as virtual appliances.

The focus of the chapter was how all the different parts of cloud computing work together. Many of the important standards are aimed at managing transactions in the cloud and making components work together. This chapter took an initial look at SOAP, XML, and the various WSDL services.

The chapter ended by considering the development of a new class of cloud-connected clients. These clients, which are often hardware and software solutions like the Google Chrome OS, may have major impact in changing the model by which people connect to the Internet and get their work done. They have many advantages, but we will have to wait to see how readily consumers and businesses are willing to adopt this new computing paradigm.

In the next chapter, each of the different Internet service models is more closely described.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER



Understanding Services and Applications by Type

This chapter describes some of the different types of cloud computing models, categorized as a set of service models. You may think of cloud computing applications as being composed of a set of layers upon which distributed applications may be built or hosted. These layers include Infrastructure, Platform, and Software. Depending on the type and level of service being offered, a client can build on these layers to create cloud-based applications.

The service models described here—Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS)—are useful in categorizing not only cloud computing capabilities, but specific vendor offerings, products, and services. Infrastructure as a Service allows for the creation of virtual computing systems or networks.

Software as a Service represents a hosted application that is universally available over the Internet, usually through a browser. With Software as a Service, the user interacts directly with the hosted software. SaaS may be seen to be an alternative model to that of shrink-wrapped software and may replace much of the boxed software that we buy today.

Platform as a Service is a cloud computing infrastructure that creates a development environment upon which applications may be build. PaaS provides a model that can be used to create or augment complex applications such as Customer Relation Management (CRM) or Enterprise Resource Planning (ERP) systems. PaaS offers the benefits of cloud computing and is often componentized and based on a service-oriented architecture model.

As cloud computing matures, several service types are being introduced and overlaid upon these architectures. The most fully developed of these service types is Identity as a Service (IDaaS). Identity as a Service provides authentication and authorization services on distributed networks. Infrastructure and

IN THIS CHAPTER

Learning about different service types

Creating clouds with Infrastructure as a Service

Working with Software as a Service

Developing applications on a Platform as a Service

Securing cloud transactions with Identity as a Service

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

supporting protocols for IDaaS are described in this chapter. Other service types such as Compliance as a Service (CaaS), provisioning, monitoring, communications, and many vertical services yet to be fully developed are touched upon in this chapter.

Defining Infrastructure as a Service (IaaS)

You can broadly partition cloud computing into four layers that form a cloud computing ecosystem, as shown in Figure 4.1. The Application layer forms the basis for Software as a Service (SaaS), while the Platform layer forms the basis for Platform as a Service (PaaS) models that are described in the next two sections. Infrastructure as a Service (IaaS) creates what may be determined to be a utility computing model, something that you can tap into and draw from as you need it without significant limits on the scalability of your deployment. You pay only for what you need when you need it. IaaS may be seen to be an incredibly disruptive technology, one that can help turn a small business into a large business nearly overnight. This is a most exciting prospect; one that is fueling a number of IaaS startups during one of the most difficult recessions of recent memory.

Infrastructure as a Service (IaaS) is a cloud computing service model in which hardware is virtualized in the cloud. In this particular model, the service vendor owns the equipment: servers, storage, network infrastructure, and so forth. The developer creates virtual hardware on which to develop applications and services. Essentially, an IaaS vendor has created a hardware utility service where the user provisions virtual resources as required.

FIGURE 4.1

The cloud computing ecosystem

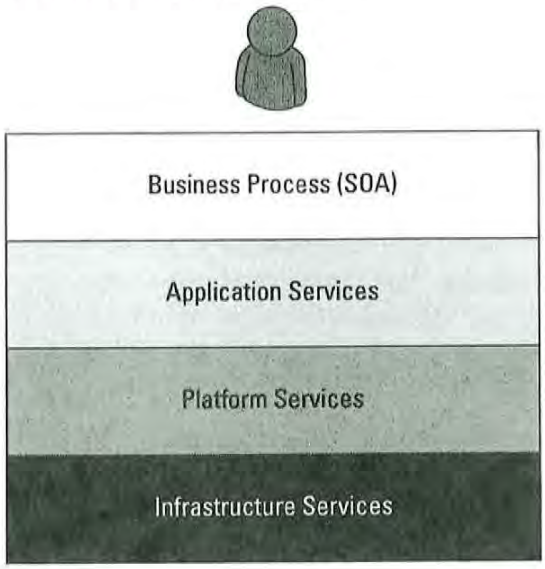


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The developer interacts with the IaaS model to create virtual private servers, virtual private storage, virtual private networks, and so on, and then populates these virtual systems with the applications and services it needs to complete its solution. In IaaS, the virtualized resources are mapped to real systems. When the client interacts with an IaaS service and requests resources from the virtual systems, those requests are redirected to the real servers that do the actual work.

IaaS workloads

The fundamental unit of virtualized client in an IaaS deployment is called a *workload*. A workload simulates the ability of a certain type of real or physical server to do an amount of work. The work done can be measured by the number of Transactions Per Minute (TPM) or a similar metric against a certain type of system. In addition to throughput, a workload has certain other attributes such as Disk I/Os measured in Input/Output Per Second IOPS, the amount of RAM consumed under load in MB, network throughput and latency, and so forth. In a hosted application environment, a client's application runs on a dedicated server inside a server rack or perhaps as a standalone server in a room full of servers. In cloud computing, a provisioned server called an instance is reserved by a customer, and the necessary amount of computing resources needed to achieve that type of physical server is allocated to the client's needs.

Figure 4.2 shows how three virtual private server instances are partitioned in an IaaS stack. The three workloads require three different sizes of computers: small, medium, and large.

A client would reserve a machine equivalent required to run each of these workloads. The IaaS infrastructure runs these server instances in the data center that the service offers, drawing from a pool of virtualized machines, RAID storage, and network interface capacity. These three layers are expressions of physical systems that are partitioned as logical units. LUNs, the cloud interconnect layer, and the virtual application software layer are logical constructs. LUNs are logical storage containers, the cloud interconnect layer is a virtual network layer that is assigned IP addresses from the IaaS network pool, and the virtual application software layer contains software that runs on the physical VM instance(s) that have been partitioned from physical assets on the IaaS' private cloud.

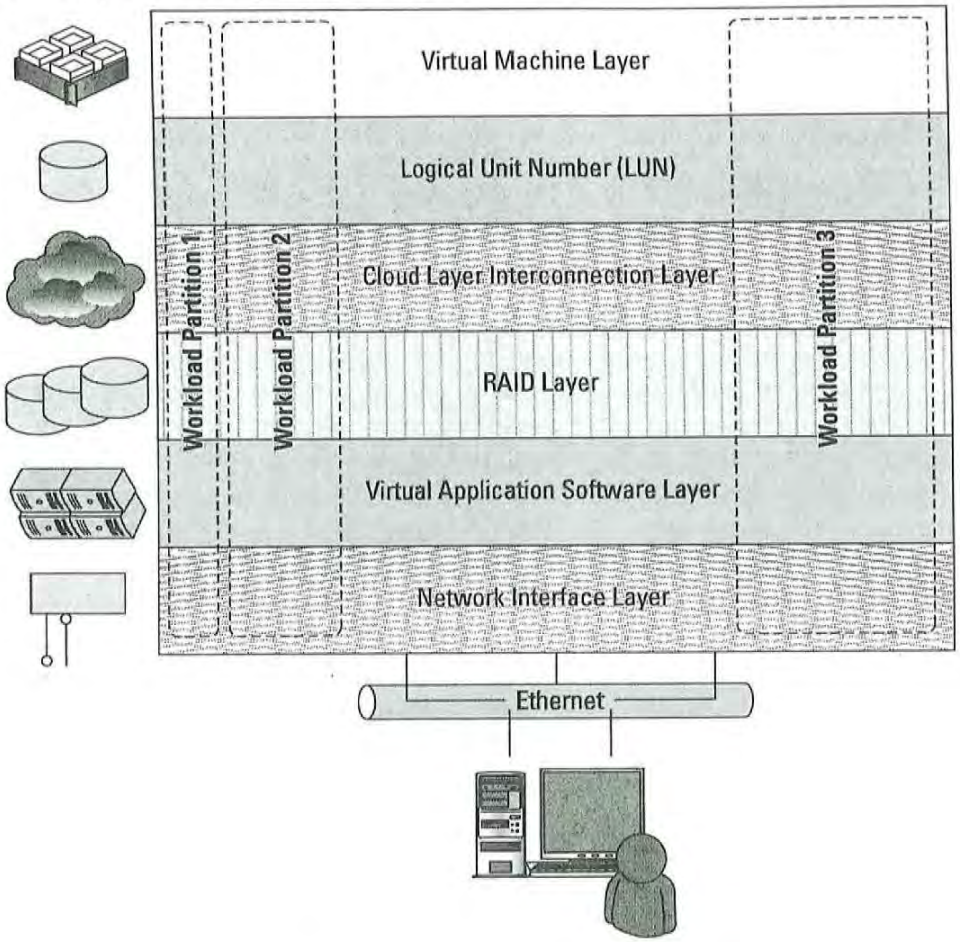
From an architectural standpoint, the client in an IaaS infrastructure is assigned its own private network. The Amazon Elastic Computer Cloud (EC2), described in detail in Chapter 8, behaves as if each server is its own separate network—unless you create your own Virtual Private Cloud (an EC2 add-on feature), which provides a workaround to this problem. When you scale your EC2 deployment, you are adding additional networks to your infrastructure, which makes it easy to logically scale an EC2 deployment, but imposes additional network overhead because traffic must be routed between logical networks. Amazon Web Service's routing limits broadcast and multicast traffic because Layer-2 (Data Link) networking is not supported. Rackspace Cloud (`http://www.rackspacecloud.com/`) follows the AWS IP assignment model.

Other IaaS infrastructures such as the one Cloudscaling.com (`http://www.cloudscaling.com`) offers or a traditional VMWare cloud-assigned networks on a per-user basis, which allows for Level 2 networking options. The most prominent Level 2 protocols that you might use are tunneling options, because they enable VLANs.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 4.2

A virtual private server partition in an IaaS cloud



Consider a transactional eCommerce system, for which a typical stack contains the following components:

- Web server
- Application server
- File server
- Database
- Transaction engine

This eCommerce system has several different workloads that are operating: queries against the database, processing of business logic, and serving up clients' Web pages.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The classic example of an IaaS service model is Amazon.com's Amazon Web Services (AWS). AWS has several data centers in which servers run on top of a virtualization platform (Xen) and may be partitioned into logical compute units of various sizes. Developers can then apply system images containing different operating systems and applications or create their own system images. Storage may be partitions, databases may be created, and a range of services such a messaging and notification can be called upon to make distributed application work correctly.

Cross-Ref

Amazon Web Services offers a classic Service Oriented Architecture (SOA) approach to IaaS. You learn more about AWS in Chapter 9; a description of the Service Oriented Architecture approach to building distributed applications is described in Chapter 13. ■

Pods, aggregation, and silos

Workloads support a certain number of users, at which point you exceed the load that the instance sizing allows. When you reach the limit of the largest virtual machine instance possible, you must make a copy or clone of the instance to support additional users. A group of users within a particular instance is called a *pod*. Pods are managed by a Cloud Control System (CCS). In AWS, the CCS is the AWS Management Console.

Sizing limitations for pods need to be accounted for if you are building a large cloud-based application. Pods are aggregated into pools within an IaaS region or site called an *availability zone*. In very large cloud computing networks, when systems fail, they fail on a pod-by-pod basis, and often on a zone-by-zone basis. For AWS' IaaS infrastructure, the availability zones are organized around the company's data centers in Northern California, Northern Virginia, Ireland, and Singapore. A failover system between zones gives IaaS private clouds a very high degree of availability. Figure 4.3 shows how pods are aggregated and virtualized in IaaS across zones.

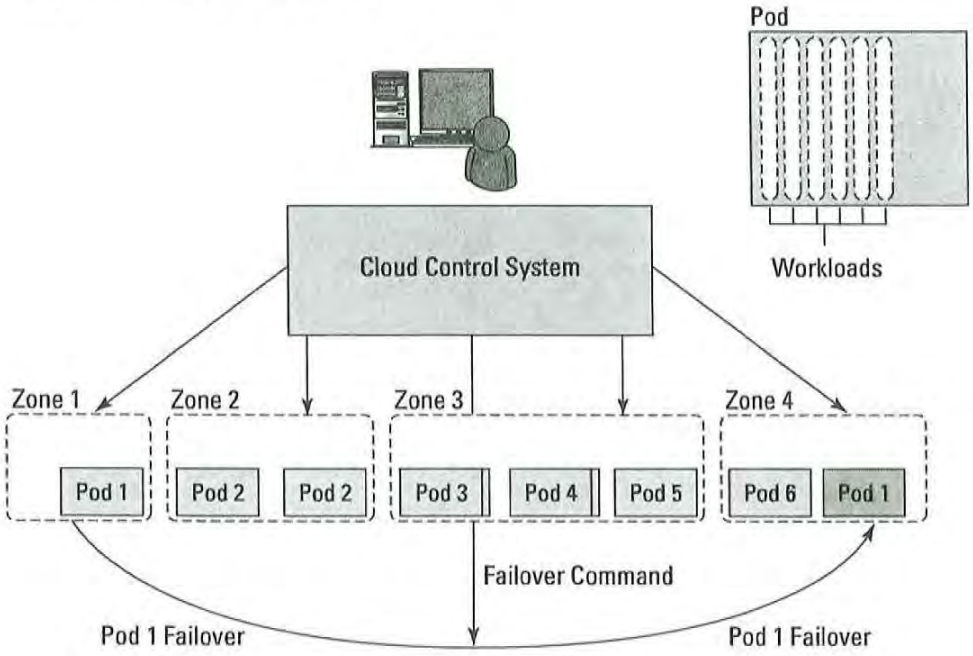
When a cloud computing infrastructure isolates user clouds from each other so the management system is incapable of interoperating with other private clouds, it creates an information silo, or simply a silo. Most often, the term *silo* is applied to PaaS offerings such as Force.com or QuickBase, but silos often are an expression of the manner in which a cloud computing infrastructure is architected. Silos are the cloud computing equivalent of compute islands: They are processing domains that are sealed off from the outside.

When you create a private virtual network within an IaaS framework, the chances are high that you are creating a silo. Silos impose restrictions on interoperability that runs counter to the open nature of build-componentized service-oriented applications. However, that is not always a bad thing. A silo can be its own ecosystem; it can be protected and secured in ways that an open system can't be. Silos just aren't as flexible as open systems and are subject to vendor lock-in.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 4.3

Pods, aggregation, and failover in IaaS



Defining Platform as a Service (PaaS)

The Platform as a Service model describes a software environment in which a developer can create customized solutions within the context of the development tools that the platform provides. Platforms can be based on specific types of development languages, application frameworks, or other constructs. A PaaS offering provides the tools and development environment to deploy applications on another vendor's application. Often a PaaS tool is a fully integrated development environment; that is, all the tools and services are part of the PaaS service. To be useful as a cloud computing offering, PaaS systems must offer a way to create user interfaces, and thus support standards such as HTLM, JavaScript, or other rich media technologies.

In a PaaS model, customers may interact with the software to enter and retrieve data, perform actions, get results, and to the degree that the vendor allows it, customize the platform involved. The customer takes no responsibility for maintaining the hardware, the software, or the development of the applications and is responsible only for his interaction with the platform. The vendor is responsible for all the operational aspects of the service, for maintenance, and for managing the product(s) lifecycle.

The one example that is most quoted as a PaaS offering is Google's App Engine platform, which is described in more detail in Chapter 8. Developers program against the App Engine using Google's published APIs. The tools for working within the development framework, as well as the structure of the file system and data stores, are defined by Google. Another example of a PaaS offering is

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Force.com, Salesforce.com's developer platform for its SaaS offerings, described in the next section. Force.com is an example of an add-on development environment.

A developer might write an application in a programming language like Python using the Google API. The vendor of the PaaS solution is in most cases the developer, who is offering a complete solution to the customer. Google itself also serves as a PaaS vendor within this system, because it offers many of its Web service applications to customers as part of this service model. You can think of Google Maps, Google Earth, Gmail, and the myriad of other PaaS offerings as conforming to the PaaS service model, although these applications themselves are offered to customers under what is more aptly described as the Software as a Service (SaaS) model that is described below.

The difficulty with PaaS is that it locks the developer (and the customer) into a solution that is dependent upon the platform vendor. An application written in Python against Google's API using the Google App Engine is likely to work only in that environment. There is considerable vendor lock-in associated with a PaaS solution.

Defining Software as a Service (SaaS)

The most complete cloud computing service model is one in which the computing hardware and software, as well as the solution itself, are provided by a vendor as a complete service offering. It is referred to as the Software as a Service (SaaS) model. SaaS provides the complete infrastructure, software, and solution stack as the service offering. A good way to think about SaaS is that it is the cloud-based equivalent of shrink-wrapped software.

Software as a Service (SaaS) may be succinctly described as software that is deployed on a hosted service and can be accessed globally over the Internet, most often in a browser. With the exception of the user interaction with the software, all other aspects of the service are abstracted away.

Every computer user is familiar with SaaS systems, which are either replacements or substitutes for locally installed software. Examples of SaaS software for end-users are Google Gmail and Calendar, QuickBooks online, Zoho Office Suite, and others that are equally well known. SaaS applications come in all shapes and sizes, and include custom software such as billing and invoicing systems, Customer Relationship Management (CRM) applications, Help Desk applications, Human Resource (HR) solutions, as well as myriad online versions of familiar applications.

Many people believe that SaaS software is not customizable, and in many SaaS applications this is indeed the case. For user-centric applications such as an office suite, that is mostly true; those suites allow you to set only options or preferences. However, many other SaaS solutions expose Application Programming Interfaces (API) to developers to allow them to create custom composite applications. These APIs may alter the security model used, the data schema, workflow characteristics, and other fundamental features of the service's expression as experienced by the user. Examples of an SaaS platformwith an exposed API are Salesforce.com and Quicken.com. So SaaS does not necessarily mean that the software is static or monolithic.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

SaaS characteristics

All Software as a Service (SaaS) applications share the following characteristics:

1. The software is available over the Internet globally through a browser on demand.
2. The typical license is subscription-based or usage-based and is billed on a recurring basis.

   In a small number of cases a flat fee may be changed, often coupled with a maintenance fee. Table 4.1 shows how different licensing models compare.
3. The software and the service are monitored and maintained by the vendor, regardless of where all the different software components are running.

   There may be executable client-side code, but the user isn't responsible for maintaining that code or its interaction with the service.
4. Reduced distribution and maintenance costs and minimal end-user system costs generally make SaaS applications cheaper to use than their shrink-wrapped versions.
5. Such applications feature automated upgrades, updates, and patch management and much faster rollout of changes.
6. SaaS applications often have a much lower barrier to entry than their locally installed competitors, a known recurring cost, and they scale on demand (a property of cloud computing in general).
7. All users have the same version of the software so each user's software is compatible with another's.
8. SaaS supports multiple users and provides a shared data model through a single-instance, multi-tenancy model.

   The alternative of software virtualization of individual instances also exists, but is less common.

TABLE 4.1

Shrink-Wrapped versus SaaS Licensing

	Shrink-Wrapped Software	Hybrid Model	SaaS
Licensing	Owned	Subscription (flat fee)	Metered subscription
Location	Locally installed	Available through an application	Cloud based
Management	Local IT staff	Application Service Provider (ASP)	Cloud vendor through a Service Level Agreement (SLA)

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Open SaaS and SOA

A considerable amount of SaaS software is based on open source software. When open source software is used in a SaaS, you may hear it referred to as *Open SaaS*. The advantages of using open source software are that systems are much cheaper to deploy because you don't have to purchase the operating system or software, there is less vendor lock-in, and applications are more portable. The popularity of open source software, from Linux to APACHE, MySQL, and Perl (the LAMP platform) on the Internet, and the number of people who are trained in open source software make Open SaaS an attractive proposition. The impact of Open SaaS will likely translate into better profitability for the companies that deploy open source software in the cloud, resulting in lower development costs and more robust solutions.

A mature SaaS implementation based on SOA is shown in Figure 4.4.

FIGURE 4.4

A modern implement of SaaS using an Enterprise Service Bus and architected with SOA components

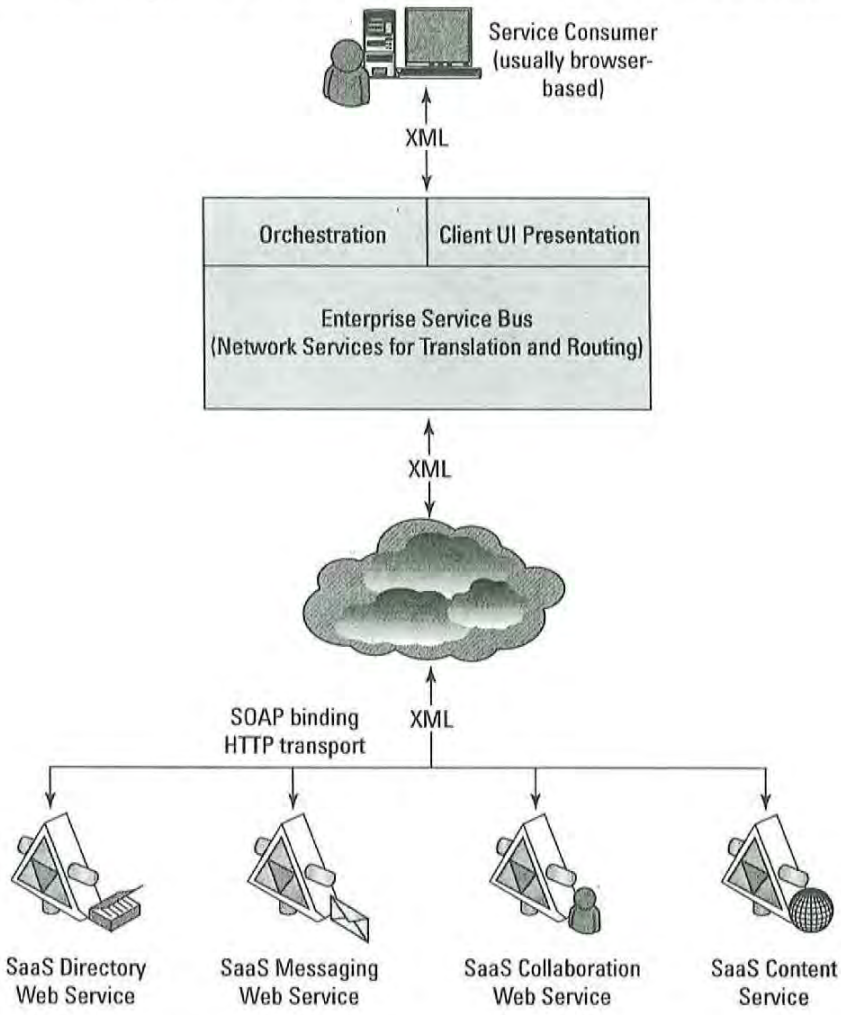


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The componentized nature of SaaS solutions enables many of these solutions to support a feature called mashups. A mashup is an application that can display a Web page that shows data and supports features from two or more sources. Annotating a map such as Google maps is an example of a mashup. Mashups are considered one of the premier examples of Web 2.0, and that is technology's ability to support social network systems.

A mashup requires three separate components:

- An interactive user interface, which is usually created with HTML/XHTML, Ajax, JavaScript, or CSS.
- Web services that can be accessed using an API, and whose data can be bound and transported by Web service protocols such as SOAP, REST, XML/HTTP, XML/RPC, and JSON/RPC.
- Data transfer in the form of XML, KML (Keyhole Markup Language), JSON (JavaScript Object Notation), or the like.

Mashups are an incredibly useful hybrid Web application, one that SaaS is a great enabler for. The Open Mashup Alliance (OMA; see `http://www.openmashup.org/`) is a non-profit industry group dedicated to supporting technologies that implement enterprise mashups. This group supports the developing standard, the Enterprise Mashup Markup Language (EMML), which is a Domain Specific Language (DSL). This group predicts that the use of mashups will grow by a factor of 10 within just a few years.

Gartner Group predicts that approximately 25 percent of all software sold by 2011 will use the SAAS model, offered either by vendors or an intermediary party, sometimes referred to as an *aggregator*. An aggregator bundles SaaS applications from different vendors and presents them as part of a unified platform or solution.

Note

There's a notion that SaaS will eventually replace all locally installed software. However, owning software has several advantages that greatly inhibit this trend: You aren't exposed to the risk of an SaaS company going out of business; you aren't dependent upon the vagaries of an Internet connection; you aren't subject to the often much slower processing speed of a distributed computing model (compared to your local system).

I believe the introduction of SaaS applications will eventually drive down the price of major applications such as Microsoft Office and Adobe Photoshop because the most important functionality contained in commercial packages becomes available at equal quality online for a much lower cost. ■

SaaS examples abound, and while many SaaS offerings are based on proprietary software, a cloud computing service is required to be highly interoperable with other services and therefore easily replaced by a newer or better version of that software's function. When you think about this difference, visualize the situation involved for large enterprise applications such as CRM and ERP, which involve large application suites containing multiple interacting data stores. SaaS software based on the Service Oriented Architecture described in Chapter 13 has the potential of decoupling this integration, with all the attendant benefits.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 5

Understanding Abstraction and Virtualization

In this chapter, I discuss different technologies that create shared pools of resources. The key to creating a pool is to provide an abstraction mechanism so that a logical address can be mapped to a physical resource. Computers use this technique for placing files on disk drives, and cloud computing networks use a set of techniques to create virtual servers, virtual storage, virtual networks, and perhaps one day virtual applications. Abstraction enables the key benefit of cloud computing: shared, ubiquitous access.

In this chapter, you learn about how load balancing can be used to create high performance cloud-based solutions. Google.com's network is an example of this approach. Google uses commodity servers to direct traffic appropriately.

Another technology involves creating virtual hardware systems. An example of this type of approach is hypervisors that create virtual machine technologies. Several important cloud computing approaches use a strictly hardware-based approach to abstraction. I describe VMware's vSphere infrastructure in some detail, along with some of the unique features and technologies that VMware has developed to support this type of cloud.

Finally, I describe some approaches to making applications portable. Application portability is a difficult proposition, and work to make applications portable is in its infancy. Two approaches are described, the Simple API and AppZero's Virtual Application Appliance (VAA). VAAs are containers that abstract an application from the operating system, and they offer the potential to make an application portable from one platform to another.

IN THIS CHAPTER

Understanding how abstraction makes cloud computing possible

Understanding how virtualization creates shared resource pools

Using load balancing to enable large cloud computing applications

Using hypervisors to make virtual machines possible

Discussing system imaging and application portability for the cloud

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Using Virtualization Technologies

The dictionary includes many definitions for the word "cloud." A cloud can be a mass of water droplets, gloom, an obscure area, or a mass of similar particles such as dust or smoke. When it comes to cloud computing, the definition that best fits the context is "a collection of objects that are grouped together." It is that act of grouping or creating a resource pool that is what succinctly differentiates cloud computing from all other types of networked systems.

Not all cloud computing applications combine their resources into pools that can be assigned on demand to users, but the vast majority of cloud-based systems do. The benefits of pooling resources to allocate them on demand are so compelling as to make the adoption of these technologies a priority. Without resource pooling, it is impossible to attain efficient utilization, provide reasonable costs to users, and proactively react to demand. In this chapter, you learn about the technologies that abstract physical resources such as processors, memory, disk, and network capacity into virtual resources.

When you use cloud computing, you are accessing pooled resources using a technique called virtualization. Virtualization assigns a logical name for a physical resource and then provides a pointer to that physical resource when a request is made. Virtualization provides a means to manage resources efficiently because the mapping of virtual resources to physical resources can be both dynamic and facile. Virtualization is dynamic in that the mapping can be assigned based on rapidly changing conditions, and it is facile because changes to a mapping assignment can be nearly instantaneous.

These are among the different types of virtualization that are characteristic of cloud computing:

- **Access:** A client can request access to a cloud service from any location.
- **Application:** A cloud has multiple application instances and directs requests to an instance based on conditions.
- **CPU:** Computers can be partitioned into a set of virtual machines with each machine being assigned a workload. Alternatively, systems can be virtualized through load-balancing technologies.
- **Storage:** Data is stored across storage devices and often replicated for redundancy.

To enable these characteristics, resources must be highly configurable and flexible. You can define the features in software and hardware that enable this flexibility as conforming to one or more of the following mobility patterns:

- **P2V:** Physical to Virtual
- **V2V:** Virtual to Virtual
- **V2P:** Virtual to Physical
- **P2P:** Physical to Physical
- **D2C:** Datacenter to Cloud

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **C2C:** Cloud to Cloud
- **C2D:** Cloud to Datacenter
- **D2D:** Datacenter to Datacenter

The techniques used to achieve these different types of virtualization are the subject of this chapter. According to Gartner ("Server Virtualization: One Path that Leads to Cloud Computing," by Thomas J. Bittman, 10/29/2009, Research Note G00171730), virtualization is a key enabler of the first four of five key attributes of cloud computing:

- **Service-based:** A service-based architecture is where clients are abstracted from service providers through service interfaces.
- **Scalable and elastic:** Services can be altered to affect capacity and performance on demand.
- **Shared services:** Resources are pooled in order to create greater efficiencies.
- **Metered usage:** Services are billed on a usage basis.
- **Internet delivery:** The services provided by cloud computing are based on Internet protocols and formats.

Load Balancing and Virtualization

One characteristic of cloud computing is virtualized network access to a service. No matter where you access the service, you are directed to the available resources. The technology used to distribute service requests to resources is referred to as *load balancing*. Load balancing can be implemented in hardware, as is the case with F5's BigIP servers, or in software, such as the Apache `mod_proxy_balancer` extension, the Pound load balancer and reverse proxy software, and the Squid proxy and cache daemon. Load balancing is an optimization technique; it can be used to increase utilization and throughput, lower latency, reduce response time, and avoid system overload.

The following network resources can be load balanced:

- Network interfaces and services such as DNS, FTP, and HTTP
- Connections through intelligent switches
- Processing through computer system assignment
- Storage resources
- Access to application instances

Without load balancing, cloud computing would very difficult to manage. Load balancing provides the necessary redundancy to make an intrinsically unreliable system reliable through managed redirection. It also provides fault tolerance when coupled with a failover mechanism. Load balancing is nearly always a feature of server farms and computer clusters and for high availability applications.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A load-balancing system can use different mechanisms to assign service direction. In the simplest load-balancing mechanisms, the load balancer listens to a network port for service requests. When a request from a client or service requester arrives, the load balancer uses a scheduling algorithm to assign where the request is sent. Typical scheduling algorithms in use today are round robin and weighted round robin, fastest response time, least connections and weighted least connections, and custom assignments based on other factors.

A session ticket is created by the load balancer so that subsequent related traffic from the client that is part of that session can be properly routed to the same resource. Without this session record or persistence, a load balancer would not be able to correctly failover a request from one resource to another. Persistence can be enforced using session data stored in a database and replicated across multiple load balancers. Other methods can use the client's browser to store a client-side cookie or through the use of a rewrite engine that modifies the URL. Of all these methods, a session cookie stored on the client has the least amount of overhead for a load balancer because it allows the load balancer an independent selection of resources.

The algorithm can be based on a simple round robin system where the next system in a list of systems gets the request. Round robin DNS is a common application, where IP addresses are assigned out of a pool of available IP addresses. Google uses round robin DNS, as described in the next section.

Advanced load balancing

The more sophisticated load balancers are workload managers. They determine the current utilization of the resources in their pool, the response time, the work queue length, connection latency and capacity, and other factors in order to assign tasks to each resource. Among the features you find in load balancers are polling resources for their health, the ability to bring standby servers online (priority activation), workload weighting based on a resource's capacity (asymmetric loading), HTTP traffic compression, TCP offload and buffering, security and authentication, and packet shaping using content filtering and priority queuing.

An Application Delivery Controller (ADC) is a combination load balancer and application server that is a server placed between a firewall or router and a server farm providing Web services. An Application Delivery Controller is assigned a virtual IP address (VIP) that it maps to a pool of servers based on application specific criteria. An ADC is a combination network and application layer device. You also may come across ADCs referred to as a content switch, multilayer switch, or Web switch.

These vendors, among others, sell ADC systems:

- A10 Networks (`http://www.a10networks.com/`)
- Barracuda Networks (`http://www.barracudanetworks.com/`)
- Brocade Communication Systems (`http://www.brocade.com/`)
- Cisco Systems (`http://www.cisco.com/`)
- Citrix Systems (`http://www.citrix.com/`)

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- F5 Networks (`http://www.f5.com/`)
- Nortel Networks (`http://www.nortel.com/`)
- Coyote Point Systems (`http://www.coyotepoint.com/`)
- Radware (`http://www.radware.com/`)

An ADC is considered to be an advanced version of a load balancer as it not only can provide the features described in the previous paragraph, but it conditions content in order to lower the workload of the Web servers. Services provided by an ADC include data compression, content caching, server health monitoring, security, SSL offload and advanced routing based on current conditions. An ADC is considered to be an application accelerator, and the current products in this area are usually focused on two areas of technology: network optimization, and an application or framework optimization. For example, you may find ADC's that are tuned to accelerate ASP.NET or AJAX applications.

An architectural layer containing ADCs is described as an Application Delivery Network (ADN), and is considered to provide WAN optimization services. Often an ADN is comprised of a pair of redundant ADCs. The purpose of an ADN is to distribute content to resources based on application specific criteria. ADN provide a caching mechanism to reduce traffic, traffic prioritization and optimization, and other techniques. ADN began to be deployed on Content Delivery Networks (CDN) in the late 1990s, where it added the ability to optimize applications (application fluency) to those networks. Most of the ADC vendors offer commercial ADN solutions.

In addition to the ADC vendors in the list above, these are additional ADN vendors, among others:

- Akamai Technologies (`http://www.akamai.com/`)
- Blue Coat Systems (`http://www.bluecoat.com/`)
- CDNetworks (`http://www.cdnetworks.com/`)
- Crescendo Networks (`http://www.crescendonetworks.com/`)
- Expand Networks (`http://www.expand.com/`)
- Juniper Networks (`http://www.juniper.net/`)

Google's cloud is a good example of the use of load balancing, so in the next section let's consider how Google handles the many requests that they get on a daily basis.

The Google cloud

According to the Web site tracking firm Alexa (`http://www.alexa.com/topsites`), Google is the single most heavily visited site on the Internet; that is, Google gets the most hits. The investment Google has made in infrastructure is enormous, and the Google cloud is one of the largest in use today. It is estimated that Google runs over a million servers worldwide, processes a billion search requests, and generates twenty petabytes of data per day.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Google is understandably reticent to disclose much about its network, because it believes that its infrastructure, system response, and low latency are key to the company's success. Google never gives datacenter tours to journalists, doesn't disclose where its datacenters are located, and obfuscates the locations of its datacenters by wrapping them in a corporate veil. Thus, the discretely named Tetra LLC (limited liability company) owns the land for the Council Bluffs, Iowa, site, and Lapis LLC owns the land for the Lenoir, North Carolina, site. This makes Google infrastructure watching something akin to a sport to many people.

So what follows is what we think we know about Google's infrastructure and the basic idea behind how Google distributes its traffic by pooling IP addresses and performing several layers of load balancing.

Google has many datacenters around the world. As of March 2008, Rich Miller of DataCenterKnowledge.com wrote that Google had at least 12 major installations in the United States and many more around the world. Google supports over 30 country specific versions of the Google index, and each localization is supported by one or more datacenters. For example, Paris, London, Moscow, Sao Paolo, Tokyo, Toronto, Hong Kong, Beijing and others support their countries' locale. Germany has three centers in Berlin, Frankfurt, and Munich; the Netherlands has two at Groningen and Eemshaven. The countries with multiple datacenters store index replicas and support network peering relationships. Network peering helps Google have low latency connections to large Internet hubs run by different network providers.

You can find a list of sites as of 2008 from Miller's FAQ at `http://www.datacenterknowledge.com/archives/2008/03/27/google-data-center-faq/`.

Based on current locations and the company's statements, Google's datacenters are sited based on the following factors (roughly in order of importance):

1. Availability of cheap and, if possible, renewable energy
2. The relative locations of other Google datacenters such that the site provides the lowest latency response between sites
3. Location of nearby Internet hubs and peering sites
4. A source of cooling water
5. The ability to purchase a large area of land surrounding the site

   Speculation on why Google purchases large parcels of land ranges from creating a buffer zone between the datacenter and surrounding roads and towns or possibly to allow for building wind farms when practical.
6. Tax concessions from municipalities that lower Google's overhead

Google maintains a pool of hundreds of IP addresses, all of which eventually resolve to its Mountain View, California, headquarters. When you initiate a Google search, your query is sent to a DNS server, which then queries Google's DNS servers. The Google DNS servers examine the pool of addresses to determine which addresses are geographically closest to the query origin and uses a round robin policy to assign an IP address to that request. The request usually goes to the nearest

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

datacenter, and that IP address is for a cluster of Google servers. This DNS assignment acts as a first level of IP virtualization, a pool of network addresses have been load balanced based on geography.

A Google cluster can contain thousands of servers. Google servers are racks of commodity (low cost) 1U or 2U servers containing 40 to 80 servers per rack with one switch per rack. Each switch is connected to a core gigabit switch. Google servers run a customized version of Linux with applications of several types.

When the query request arrives at its destination, a Google cluster is sent to a load balancer, which forwards that request to a Squid proxy server and Web cache dameon. This is the second level of IP distribution, based on a measure of the current system loading on proxy servers in the cluster. The Squid server checks its cache, and if it finds a match to the query, that match is returned and the query has been satisfied. If there is no match in the Squid cache, the query is sent to an individual Google Web Server based on current Web server utilizations, which is the third level of network load balancing, again based on utilization rates.

It is the Google Web Servers that perform the query against the Google index and then format the results into an HTML page that is returned to the requester. This procedure then performs two more levels of load balancing based on utilization rates.

Google's secret sauce is its in-memory inverted index and page rank algorithm. Google's GoogleBot (a spider or robot) crawls the Web and collects document information. Some details of the search and store algorithm are known. Google looks at the title and first few hundred words and builds a word index from the result. Indexes are stored on an index server.

Some documents are stored as snapshots (PDF, DOC, XLS, and so on), but lots of information is not addressed in the index. Each document is given a unique ID ("docid"), and the content of the document is disassembled into segments called shards, subjected to a data compression scheme and stored on a document server. The entire index is maintained in system memory partitioned over each instance of the index's replicas. A page rank is created based on the significant links to that page.

Queries are divided into word lists, and the Google algorithm examines the words and the relationships of one word to another. Those word relationships are mapped against the main index to create a list of documents, a feature called an inverted index. In an inverted index, words are mapped to documents, which can be done very quickly when the index is fully kept in memory.

The Web server takes the result of a query and composes the Web page from that result. Ads included on the page are from ad servers, which provide Google's AdSense and AdWords services. The query also is presented to a spelling server to provide suggestions for alternative spellings to include in the search result. Certain keywords, data input patterns, and other strings are recognized as having special operational significance. For example entering "2 plus 2" initiates Google's calculator program, and a ten-digit number returns a reverse phone lookup using the phonebook program. These programs are supported by special application servers.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Google doesn't use hardware virtualization; it performs server load balancing to distribute the processing load and to get high utilization rates. The workload management software transfers the workload from a failed server over to a redundant server, and the failed server is taken offline. Multiple instances of various Google applications are running on different hosts, and data is stored on redundant storage systems.

Understanding Hypervisors

Load balancing virtualizes systems and resources by mapping a logical address to a physical address. Another fundamental technology for abstraction creates virtual systems out of physical systems. If load balancing is like playing a game of hot potato, then virtual machine technologies is akin to playing slice and dice with the potato.

Given a computer system with a certain set of resources, you can set aside portions of those resources to create a virtual machine. From the standpoint of applications or users, a virtual machine has all the attributes and characteristics of a physical system but is strictly software that emulates a physical machine. A system virtual machine (or a hardware virtual machine) has its own address space in memory, its own processor resource allocation, and its own device I/O using its own virtual device drivers. Some virtual machines are designed to run only a single application or process and are referred to as process virtual machines.

A virtual machine is a computer that is walled off from the physical computer that the virtual machine is running on. This makes virtual machine technology very useful for running old versions of operating systems, testing applications in what amounts to a sandbox, or in the case of cloud computing, creating virtual machine instances that can be assigned a workload. Virtual machines provide the capability of running multiple machine instances, each with their own operating system.

From the standpoint of cloud computing, these features enable VMMs to manage application provisioning, provide for machine instance cloning and replication, allow for graceful system failover, and provide several other desirable features. The downside of virtual machine technologies is that having resources indirectly addressed means there is some level of overhead.

Virtual machine types

A low-level program is required to provide system resource access to virtual machines, and this program is referred to as the hypervisor or Virtual Machine Monitor (VMM). A hypervisor running on bare metal is a Type 1 VM or native VM. Examples of Type 1 Virtual Machine Monitors are LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogix VLX, VMware ESX and ESXi, and Wind River VxWorks, among others. The operating system loaded into a virtual machine is referred to as the guest operating system, and there is no constraint on running the same guest on multiple VMs on a physical system. Type 1 VMs have no host operating system because they are installed on a bare system.

An operating system running on a Type 1 VM is a full virtualization because it is a complete simulation of the hardware that it is running on.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Note

Not all CPUs support virtual machines, and many that do require that you enable this support in the BIOS. For example, AMD-V processors (code named Pacifica) and Intel VT-x (code named Vanderpool) were the first of these vendor's 64-bit offerings that added this type of support. ■

Some hypervisors are installed over an operating system and are referred to as Type 2 or hosted VM. Examples of Type 2 Virtual Machine Monitors are Containers, KVM, Microsoft Hyper V, Parallels Desktop for Mac, Wind River Simics, VMWare Fusion, Virtual Server 2005 R2, Xen, Windows Virtual PC, and VMware Workstation 6.0 and Server, among others. This is a very rich product category. Type 2 virtual machines are installed over a host operating system; for Microsoft Hyper-V, that operating system would be Windows Server. In the section that follows, the Xen hypervisor (which runs on top of a Linux host OS) is more fully described. Xen is used by Amazon Web Services to provide Amazon Machine Instances (AMIs).

Figure 5.1 shows a diagram of Type 1 and Type 2 hypervisors.

On a Type 2 VM, a software interface is created that emulates the devices with which a system would normally interact. This abstraction is meant to place many I/O operations outside the virtual environment, which makes it both programmatically easier and more efficient to execute device I/O than it would be inside a virtual environment. This type of virtualization is sometimes referred to as *paravirtualization*, and it is found in hypervisors such as Microsoft's Hyper-V and Xen. It is the host operating system that is performing the I/O through a para-API.

FIGURE 5.1

VMware's vSphere cloud computing infrastructure model

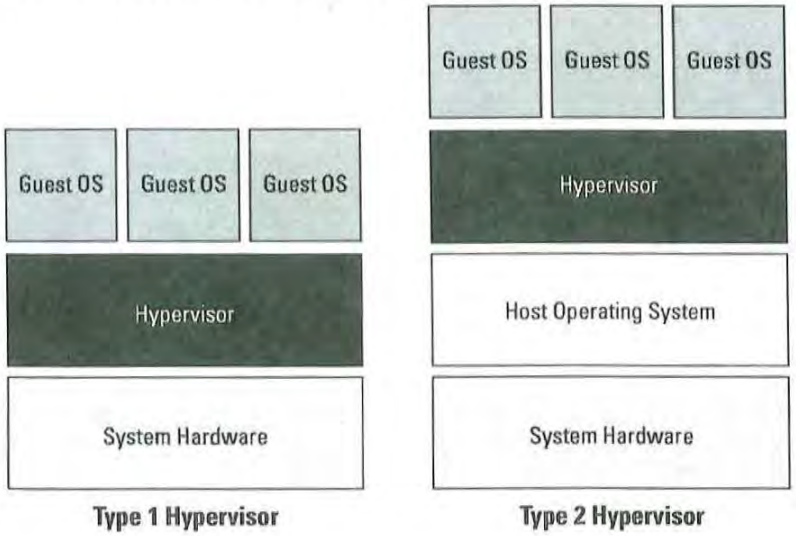


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Figure 5.2 shows the difference between emulation, paravirtualization, and full virtualization. In emulation, the virtual machine simulates hardware, so it can be independent of the underlying system hardware. A guest operating system using emulation does not need to be modified in any way. Paravirtualization requires that the host operating system provide a virtual machine interface for the guest operating system and that the guest access hardware through that host VM. An operating system running as a guest on a paravirtualization system must be ported to work with the host interface. Finally, in a full virtualization scheme, the VM is installed as a Type 1 Hypervisor directly onto the hardware. All operating systems in full virtualization communicate directly with the VM hypervisor, so guest operating systems do not require any modification. Guest operating systems in full virtualization systems are generally faster than other virtualization schemes.

The Virtual Machine Interface (VMI) open standard (`http://vmi.ncsa.uiuc.edu/`) that VMware has proposed is an example of a paravirtualization API. The latest version of VMI is 2.1, and it ships as a default installation with many versions of the Linux operating system.

Note

Wikipedia maintains a page called "Comparison of platform virtual machines" at `http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines`. The page contains a table of features of the most common Virtual Machine Managers. ■

You are probably familiar with process or application virtual machines. Most folks run the Java Virtual Machine or Microsoft's .NET Framework VM (called the Common Language Runtime or CLR) on their computers. A process virtual machine instantiates when a command begins a process, the VM is created by an interpreter, the VM then executes the process, and finally the VM exits the system and is destroyed. During the time the VM exists, it runs as a high-level abstraction.

FIGURE 5.2

Emulation, paravirtualization, and full virtualization types

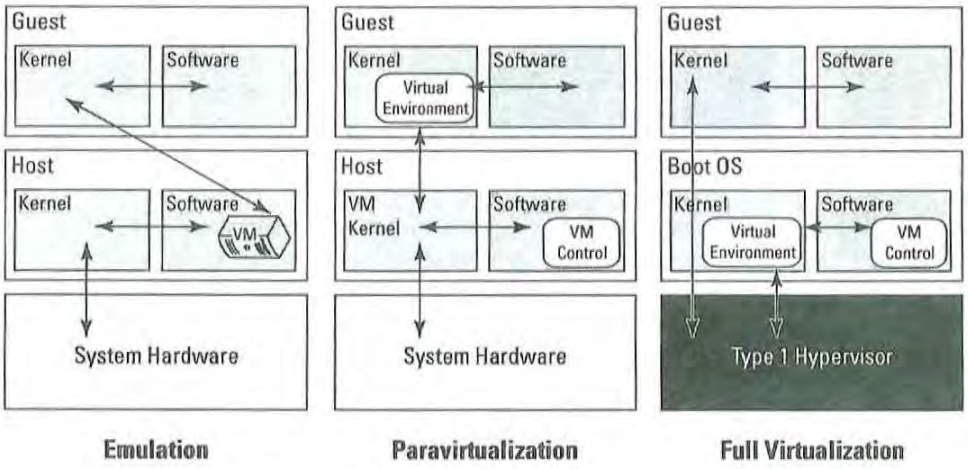


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Applications running inside an application virtual machine are generally slow, but these programs are very popular because they provide portability, offer rich programming languages, come with many advanced features, and allow platform independence for their programs. Although many cloud computing applications provide process virtual machine applications, this type of abstraction isn't really suitable for building a large or high-performing cloud network, with one exception.

The exception is the process VMs that enable a class of parallel cluster computing applications. These applications are high-performance systems where the virtual machine is operating one process per cluster node, and the system maintains the necessary intra-application communications over the network interconnect. Examples of this type of system are the Parallel Virtual Machine (PVM; see `http://www.csm.ornl.gov/pvm/pvm_home.html`) and the Message Passing Interface (MPI; see `http://www.mpi-forum.org/`). Some people do not consider these application VMs to be true virtual machines, noting that these applications can still access the host operating system services on the specific system on which they are running. The emphasis on using these process VMs is in creating a high-performance networked supercomputer often out of heterogeneous systems, rather than on creating a ubiquitous utility resource that characterizes a cloud network.

Some operating systems such as Sun Solaris and IBM AIX 6.1 support a feature known as *operating system virtualization*. This type of virtualization creates virtual servers at the operating system or kernel level. Each virtual server is running in its own virtual environment (VE) as a virtual private server (VPS). Different operating systems use different names to describe these machine instances, each of which can support its own guest OS. However, unlike true virtual machines, VPS must all be running the same OS and the same version of that OS. Sun Solaris 10 uses VPS to create what is called Solaris Zones. With IBM AIX, the VPS is called a System Workload Partition (WPAR). This type of virtualization allows for a dense collection of virtual machines with relatively low overhead. Operating system virtualization provides many of the benefits of virtualization previously noted in this section.

VMware vSphere

VMware vSphere is a management infrastructure framework that virtualizes system, storage, and networking hardware to create cloud computing infrastructures. vSphere is the branding for a set of management tools and a set of products previously labeled VMware Infrastructure. vSphere provides a set of services that applications can use to access cloud resources, including these:

- **VMware vCompute:** A service that aggregates servers into an assignable pool
- **VMware vStorage:** A service that aggregates storage resources into an assignable pool
- **VMware vNetwork:** A service that creates and manages virtual network interfaces
- **Application services:** Such as HA (High Availability) and Fault Tolerance
- **vCenter Server:** A provisioning, management, and monitoring console for VMware cloud infrastructures

Figure 5.3 shows an architectural diagram of a vSphere cloud infrastructure.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 5.3

VMware's vSphere cloud computing infrastructure model

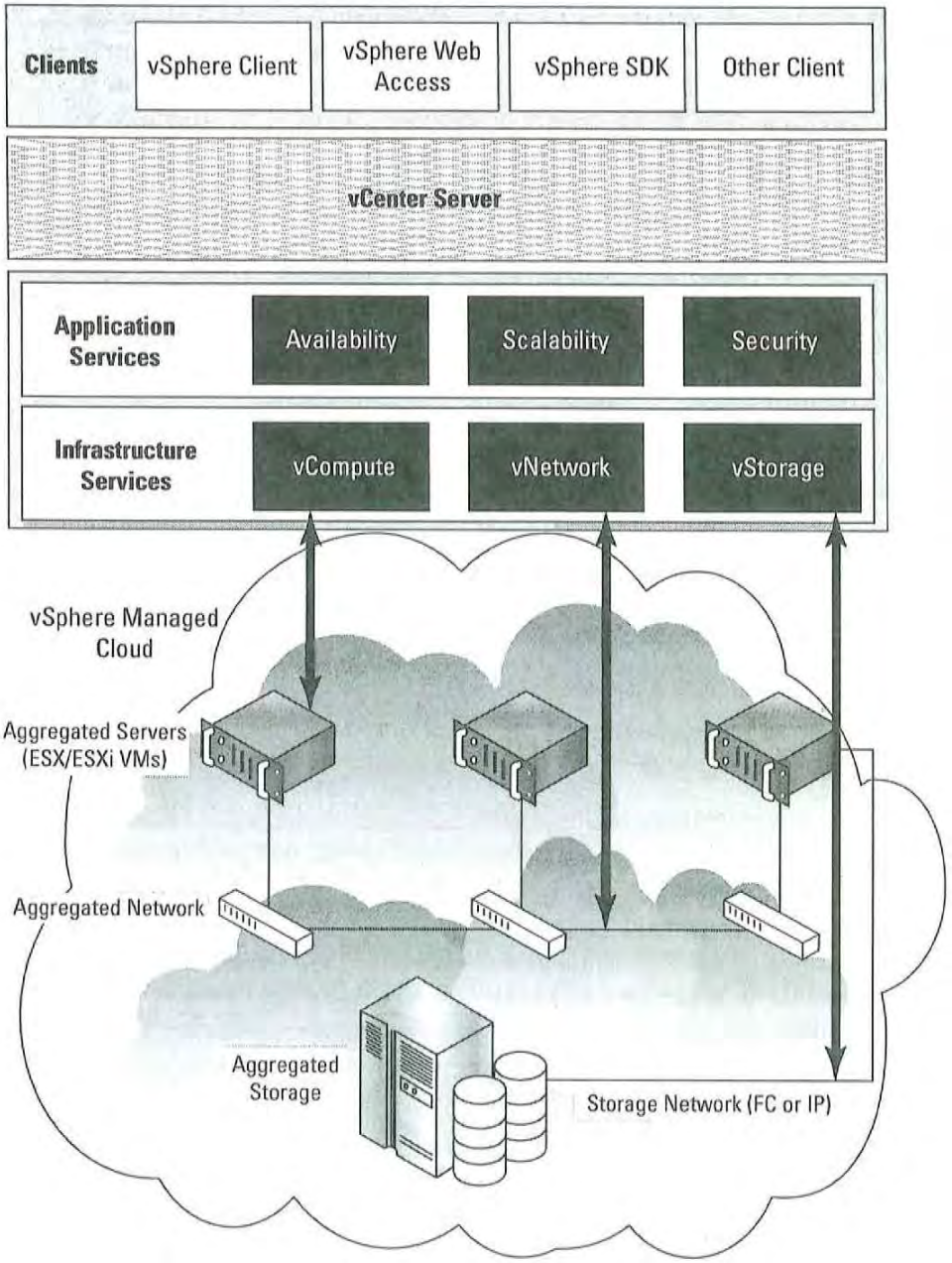


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A vSphere cloud is a pure infrastructure play. The virtualization layer that abstracts processing, memory, and storage uses the VMware ESX or ESXi virtualization server. ESX is a Type 1 hypervisor; it installs over bare metal (a clean system) using a Linux kernel to boot and installs the vmkernel hypervisor (virtualization kernel and support files). When the system is rebooted, the vmkernel loads first, and then the Linux kernel becomes the first guest operating system to run as a virtual machine on the system and contains the service console.

VMware is a very highly developed infrastructure and the current leader in this industry. A number of important add-on products are available for cloud computing applications. These are among the more notable products:

- **Virtual Machine File System (VMFS):** A high-performance cluster file system for an ESX/ESXi cluster.
- **VMotion:** A service that allows for the migration of a virtual machine from one physical server to another physical server while the virtual server runs continuously and without any interruption of ongoing transactions.

  The ability to live migrate virtual machines is considered to be a technological tour de force and a differentiator from other virtual machine system vendors.
- **Storage VMotion:** A product that can migrate files from one datastore to another datastore while the virtual machine that uses the datastore continues to run.
- **Virtual SMP:** A feature that allows a virtual machine to run on two or more physical processors at the same time.
- **Distributed Resource Scheduler (DRS):** A system for provisioning virtual machines and load balancing processing resources dynamically across the different physical systems that are in use. A part of the DRS called the distributed power management (DPM) module can manage the power consumption of systems.
- **vNetwork Distributed Switch (DVS):** A capability to maintain a network runtime state for virtual machines as they are migrated from one physical system to another. DVS also monitors network connections, provides firewall services, and enables the use of third-party switches such as the Cisco Nexus 1000V to manage virtual networks.

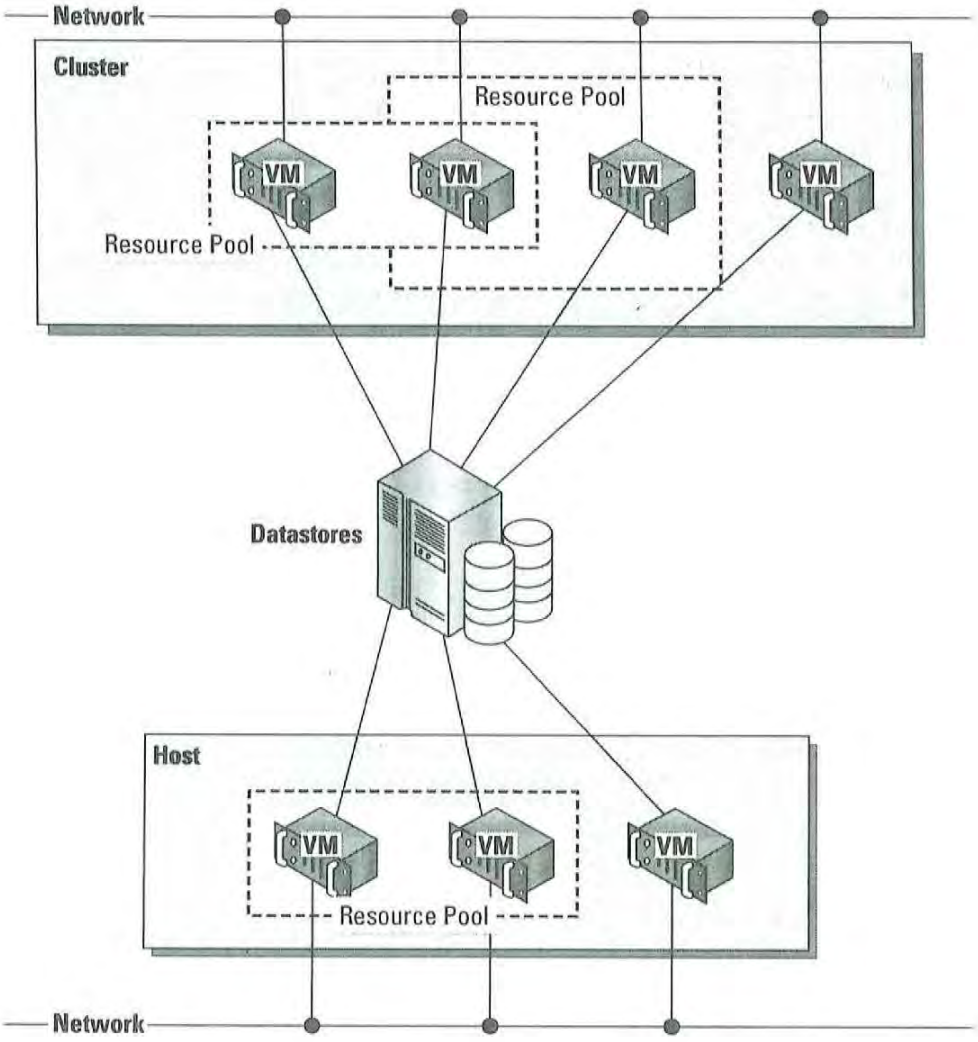
You can get a better sense of how the different resources are allocated by vSphere into a virtual set of components by examining Figure 5.4. Physical computers can be standalone hosts or a set of clustered systems. In either case, a set of virtual machines can be created that is part of a single physical system or spans two or more physical systems.

You can define a group of VMs as a Resource Pool (RP) and, by doing so, manage those virtual machines as a single object with a single policy. Resource Pools can be placed into a hierarchy or nested and can inherit properties of their parent RP. As more hosts or cluster nodes are added or removed, vSphere can dynamically adjust the provisioning of VMs to accommodate the policy in place. This fine tuning of pooled resources is required to accommodate the needs of cloud computing networks.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 5.4

Virtual infrastructure elements



The datastore shown at the center of Figure 5.4 is a shared storage resource. These storage resources can be either Direct Attached Storage (DAS) of a server using SCSI, SAS, or SATA connections, Fibre Channel disk arrays/SANs, iSCSI disk arrays/SANs, or Network Attached Storage (NAS) disk arrays. Although the lines drawn between the datastore and different VMs indicate a direct connection, with the exception of DAS, the other storage types are shared storage solutions.

Storage virtualization is most commonly achieved through a mapping mechanism where a logical storage address is translated into a physical storage address. Block-based storage such as those used

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

in SANs use a feature called a Logical Unit Identifier (LUN) with specific addresses stored in the form of an offset called the Logical Block Address (LBA). The address space mapping then maps the address of the logical or virtual disk (vdisk) to the logical unit on a storage controller. Storage virtualization may be done in software or in hardware, and it allows requests for virtualized storage to be redirected as needed.

Similarly, network virtualization abstracts networking hardware and software into a virtual network that can be managed. A virtual network can create virtual network interfaces (VNICs) or virtual LANs (VLANS) and can be managed by a hypervisor, operating system, or external management console. In a virtualized infrastructure such as the one presented in this section, internal network virtualization is occurring and the hypervisor interacts with networking hardware to create a pseudo-network interface. External network virtualization can be done using network switches and VLAN software.

The key feature that makes virtual infrastructure so appealing for organizations implementing a cloud computing solution is flexibility. Instantiating a virtual machine is a very fast process, typically only a few seconds in length. You can make machine images of systems in the configuration that you want to deploy or take snapshots of working virtual machines. These images can be brought on-line as needed.

Understanding Machine Imaging

In the preceding sections, you have seen how the abstractions that cloud computing needs can be achieved through redirection and virtualization. A third mechanism is commonly used to provide system portability, instantiate applications, and provision and deploy systems in the cloud. This third mechanism is through storing the state of a systems using a system image.

Cross-Ref

Backing up to the cloud often involves imaging or snapshot applications; this process is described in Chapter 15, "Working with Cloud-Based Storage." ■

A system image makes a copy or a clone of the entire computer system inside a single container such as a file. The system imaging program is used to make this image and can be used later to restore a system image. Some imaging programs can take snapshots of systems, and most allow you to view the files contained in the image and do partial restores.

Note

The one open standard for storing a system image is the Open Virtualization Format (OVF; see `http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf`) that is published by the Distributed Task Format (DMTF; `http://www.dmtf.org/`). Some notable virtualization vendors, such as VMware, Microsoft, Citrix, and Oracle (Sun), are supporting this effort. ■

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A prominent example of a system image and how it can be used in cloud computing architectures is the Amazon Machine Image (AMI) used by Amazon Web Services to store copies of a virtual machine. Because this is a key feature of Amazon's Elastic Compute Cloud and is discussed in detail in Chapter 9, I briefly mention it here. An AMI is a file system image that contains an operating system, all appropriate device drivers, and any applications and state information that the working virtual machine would have.

When you subscribe to AWS, you can choose to use one of its hundreds of canned AMIs or to create a custom system and capture that system's image to an AMI. An AMI can be for public use under a free distribution license, for pay-per-use with operating systems such as Windows, or shared by an EC2 user with other users who are given the privilege of access.

Cross-Ref

Refer to Chapter 9, "Using Amazon Web Services," for more information about AMIs and their uses in the EC2 service. ■

The AMI file system is not a standard bit-for-bit image of a system that is common to many disk imaging programs. AMI omits the kernel image and stores a pointer to a particular kernel that is part of the AWS kernel library. Among the choices are Red Hat Linux, Ubuntu, Microsoft Windows, Solaris, and others. Files in AMI are compressed and encrypted, and an XML file is written that describes the AMI archive. AMIs are typically stored in your Amazon S3 (Simple Storage System) buckets as a set of 10MB chunks.

Machine images are sometimes referred to as "virtual appliances"—systems that are meant to run on virtualization platforms. AWS EC2 runs on the Xen hypervisor, for example. The term *virtual appliance* is meant to differentiate the software image from an operating virtual machine. The system image contains the operating system and applications that create an environment. Most virtual appliances are used to run a single application and are configurable from a Web page. Virtual appliances are a relatively new paradigm for application deployment, and cloud computing is the major reason for the interest in them and for their adoption. This area of WAN application portability and deployment, and of WAN optimization of an application based on demand, is one with many new participants. Certeon (`http://www.certeon.com/`), Expand Networks (`http://www.expand.com/`), and Replify (`http://www.replify.com/`) are three vendors offering optimization appliances for VMware's infrastructure.

Porting Applications

Cloud computing applications have the ability to run on virtual systems and for these systems to be moved as needed to respond to demand. Systems (VMs running applications), storage, and network assets can all be virtualized and have sufficient flexibility to give acceptable distributed WAN application performance. Developers who write software to run in the cloud will undoubtedly want the ability to port their applications from one cloud vendor to another, but that is a much more difficult proposition. Cloud computing is a relatively new area of technology, and the major vendors have technologies that don't interoperate with one another.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The Simple Cloud API

If you build an application on a platform such as Microsoft Azure, porting that application to Amazon Web Services or GoogleApps may be difficult, if not impossible. In an effort to create an interoperability standard, Zend Technologies has started an open source initiative to create a common application program interface that will allow applications to be portable. The initiative is called the Simple API for Cloud Application Services (`http://www.simplecloud.org/`), and the effort has drawn interest from several major cloud computing companies. Among the founding supporters are IBM, Microsoft, Nivanix, Rackspace, and GoGrid.

Simple Cloud API has as its goal a set of common interfaces for:

- **File Storage Services:** Currently Amazon S3, Windows Azure Blob Storage, Nirvanix, and Local storage is supported by the Storage API. There are plans to extend this API to Rackspace Cloud Files and GoGrid Cloud Storage.
- **Document Storage Services:** Amazon SimpleDB and Windows Azure Table Storage are currently supported. Local document storage is planned.
- **Simple Queue Services:** Amazon SQS, Windows Azure Queue Storage, and Local queue services are supported.

Zend intends to add the interface to their open source PHP Framework (`http://www.framework.zend.com`) as the Zend_Cloud framework component. Vendors such as Microsoft and IBM are supplying adapters that will use part of the Simple Cloud API for their cloud application services.

AppZero Virtual Application Appliance

Applications that run in datacenters are captive to the operating systems and hardware platforms that they run on. Many datacenters are a veritable Noah's Ark of computing. So moving an application from one platform to another isn't nearly as simple as moving a machine image from one system to another.

The situation is further complicated by the fact that applications are tightly coupled with the operating systems on which they run. An application running on Windows, for example, isn't isolated from other applications. When the application loads, it often loads or uses different Dynamic Link Libraries (DLL), and it is through the sharing or modification of DLLs that Windows applications get themselves in trouble. Further modifications include modifying the registry during installation. These factors make it difficult to port applications from one platform to another without lots of careful work. If you are a Platform as a Service (PaaS) application developer, you are packaging a complete software stack that includes not only your application, but the operating system and application logic and rules as well. Vendor lock-in for you application is assured.

The ability to run an application from whatever platform you want is not one of the characteristics of cloud computing, but you can imagine that it is a very attractive proposition. While the Simple Cloud API is useful for applications written in PHP, other methods may be needed to make applications easily portable. One company working on this problem is AppZero (`http://www.appzero.com/`), and its solution is called the Virtual Application Appliance (VAA).

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The AppZero solution creates a virtual application appliance as an architectural layer between the Windows or the UNIX operating system and applications. The virtualization layer serves as the mediator for file I/O, memory I/O, and application calls and response to DLLs, which has the effect of sandboxing the application. The running application in AppZero changes none of the registry entries or any of the files on the Windows Server.

VAA creates a container that encapsulates the application and all the application's dependencies within a set of files; it is essentially an Application Image for a specific OS. Dependencies include DLL, service settings, necessary configuration files, registry entries, and machine and network settings. This container forms an installable server-side application stack that can be run after installation, but has no impact on the underlying operating system. VAAs are created using the AppZero Creator wizard, managed with the AppZero Admin tool, and may be installed using the AppZero Director, which creates a VAA runtime application. If desired, an application called AppZero Dissolve removes the VAA virtualization layer from the encapsulated application and installs that application directly into the operating system.

Note

Microsoft App-V (`http://www.microsoft.com/windows/enterprise/products/mdop/app-v.aspx`) and VMware ThinApp (`http://www.vmware.com/products/thinapp/`) are two application delivery platforms, but their main focus is on desktop installations and not on server deployment in the cloud. ■

Installations can be done over the network after the AppZero application appliance is installed. Therefore, with this system, you could run applications on the same Windows Server and eliminate one application from interfering with another; applications would be much more easily ported from one Windows system to another. AppZero's approach provides the necessary abstraction layer that frees an application from its platform dependence.

An interesting use of VAAs involves segmenting an application into several VAAs, some of which are read-only runtime components, while others can be modified. When backing up or replicating VAAs in a cloud, you would need to synchronize only those VAAs that are modified. In many instances, the portion of an application that changes is only a very small component of large applications, which means that this technique can greatly reduce the amount of data required to replicate a VM in the cloud.

AppZero envisages using VAAs to create what it calls a *stateless cloud.* In a stateless cloud, the application's state information is stored on a network share where it is available to run on different cloud systems as needed. This approach allows the cloud system to run with a VM containing a clean operating system (like AWS does now) and provisioned by the VAA. This approach should greatly reduce the number of complete system images that cloud vendors and cloud users should need to store to support their work; it also should make the running of applications on secure, well-performing VEs easier to achieve.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Summary

In this chapter, you learned about some of the more important characteristics of cloud computing networks and applications, including ubiquitousness and on-demand service. To enable a cloud service, you need to create a pool of resources you can call on. The key techniques for enabling this are abstraction and virtualization. Abstraction maps a logical identity or address to a physical identity or address. Changes to the underlying systems, therefore, do not affect the client requesting a service.

Several different methods for abstraction have been considered. A widely used technique is load balancing. With load balancing, system requests are directed to appropriate systems on demand. All large cloud networks use some form of load balancing. You learned about some of the details of Google's load balancing for queries.

Another technology virtualizes hardware. You learned about the different types of hypervisors—software that can serve as a virtualization layer for operating systems accessing the underlying hardware. As an example of hardware virtualization VMware's vSphere infrastructure was considered. vSphere can create virtual machines, virtual datastores, and virtual networks, and move these resources about while the system is active. vSphere is a potent cloud-building technology.

System imaging also can be useful in creating and instantiating machine instances. A brief explanation of Amazon Machine Instances was given.

Finally, the topic of application portability was considered. Applications are hard to move from platform to platform, because they are bound up with the operating system on which they run. Eventually, applications will be as portable as virtual machines. A cloud programming interface was described, as was an application delivery appliance.

In Chapter 6, "Capacity Planning," the idea of system workloads is described. Understanding this concept allows you to scale your systems correctly, choose the right type of infrastructure, and do availability planning. Some of the key performance metrics for cloud computing "right sizing" are described.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 10

Using Microsoft Cloud Services

Microsoft has a very extensive cloud computing portfolio under active development. Efforts to extend Microsoft products and third-party applications into the cloud are centered around adding more capabilities to existing Microsoft tools. Microsoft's approach is to view cloud applications as software plus service. In this model, the cloud is another platform and applications can run locally and access cloud services or run entirely in the cloud and be accessed by browsers using standard Service Oriented Architecture (SOA) protocols.

Microsoft calls their cloud operating system the Windows Azure Platform. You can think of Azure as a combination of virtualized infrastructure to which the .NET Framework has been added as a set of .NET Services. The Windows Azure service itself is a hosted environment of virtual machines enabled by a fabric called Windows Azure AppFabric. You can host your application on Azure and provision it with storage, growing it as you need it. Windows Azure service is an Infrastructure as a Service offering.

A number of services interoperate with Windows Azure, including SQL Azure (a version of SQL Server), SharePoint Services, Azure Dynamic CRM, and many of Windows Live Services comprising what is the Windows Azure Platform, which is a Platform as a Service cloud computing model. Eventually, many more services will be added, encompassing the whole range of Microsoft's offerings. This architecture positions Microsoft to either extend its product into the Web or to license its products, whichever way the cloud computing marketplace develops. From Microsoft's position and that of its developers, Windows Azure has lots of advantages.

Windows Live Services is a collection of applications and services that run on the Web. Some of these applications called Windows Live Essentials are add-ons to Windows and downloadable as applications. Other Windows

IN THIS CHAPTER

Understanding Microsoft's developing cloud computing strategy

Learning about the Windows Azure hosting service

Exploring application development using the Windows Azure Platform

Viewing the applications and services that are part of Windows Live

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Live Services are standalone Web applications viewable in a browser. An important subset of these Windows Live Services is available to Windows Azure applications through the Windows Live Messenger Connect API. A set of Windows Live for Mobile applications also exists. These applications and services are more fully described in this chapter.

Exploring Microsoft Cloud Services

Microsoft CEO Steve Balmer recently said at a University of Washington speech that Microsoft was "betting our company" on the cloud. Balmer also claimed that about 70 percent of Microsoft employees were currently working on cloud-related projects and that the number was expected to rise to about 90 percent within a year. Plans to integrate cloud-based applications and services into the Microsoft product portfolio dominates the thinking at Microsoft and is playing a central role in the company's ongoing product development. The starting place for Microsoft's cloud computing efforts may be found at `Microsoft.com/cloud`, shown in Figure 10.1.

Microsoft has a vast array of cloud computing products and initiatives, and a number of industry-leading Web applications. Although services like America Online Instant Messenger (AIM) may garner mindshare in the United States, surprisingly Microsoft Messenger is the market leader in many other countries. Product by product in any category you can name—calendars, event managers, photo galleries, image editors, movie making, and so on—Microsoft has a Web application for it. Some of these products are also-rans, some are good, some are category leaders, and a few of them are really unique. What is also true is that Web apps are under very active development. Microsoft sees its on-line application portfolio as a way of extending its desktop applications to make the company pervasive and to extend its products' lives well into the future.

Going forward, Microsoft sees its future as providing the best Web experience for any type of device, which means that it structures its development environment so the application alters its behavior depending upon the device. For a mobile device, that would mean adjusting the user interface to accommodate the small screen, while for a PC the Web application would take advantage of the PC hardware to accelerate the application and add richer graphics and other features. That means Microsoft is pushing cloud development in terms of applications serving as both a service and an application. This duality—like light, both a particle and a wave—manifests itself in the way Microsoft is currently structuring its Windows Live Web products. Eventually, the company intends to create a Microsoft app store to sell cloud applications to users.

Microsoft Live is only one part of the Microsoft cloud strategy. The second part of the strategy is the extension of the .NET Framework and related development tools to the cloud. To enable .NET developers to extend their applications into the cloud, or to build .NET style applications that run completely in the cloud, Microsoft has created a set of .NET services, which it now refers to as the Windows Azure Platform. .NET Services itself had as its origin the work Microsoft did to create its BizTalk products.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.1

Microsoft maintains a home page for cloud computing at `http://www.microsoft.com/cloud`.



Azure and its related services were built to allow developers to extend their applications into the cloud. Azure is a virtualized infrastructure to which a set of additional enterprise services has been layered on top, including:

- A virtualization service called Azure **AppFabric** that creates an application hosting environment. AppFabric (formerly .NET Services) is a cloud-enabled version of the .NET Framework.
- A high capacity non-relational storage facility called Storage.
- A set of virtual machine instances called Compute.
- A cloud-enabled version of SQL Server called SQL Azure Database.
- A database marketplace based on SQL Azure Database code-named "Dallas."
- An xRM (Anything Relations Management) service called Dynamics CRM based on Microsoft Dynamics.
- A document and collaboration service based on SharePoint called SharePoint Services.
- Windows Live Services, a collection of services that runs on Windows Live, which can be used in applications that run in the Azure cloud.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Eventually the entire Microsoft server portfolio will be available as a cloud-based application or service, including Exchange. So the Windows Azure Platform can be viewed in a sense as the next Microsoft operating system, the first one that is a cloud OS. The Microsoft vision for the Windows Azure Platform is shown in Figure 10.2, where the company sees applications developed in Visual Studio or through PHP and other languages deployed to the cloud, existing local (on-premises) applications interacting with Azure with standard SOA protocols (SOAP, REST, and XML), all running on the Windows Azure virtualized infrastructure.

The end result is pervasive computing available to users on the device of their choice. Just how Microsoft intends to integrate all these technologies into a unified offering is the story of this chapter.

FIGURE 10.2

The integrated vision for application development and deployment with Azure is illustrated in this overview page of the Azure platform (`http://www.microsoft.com/windowsazure/products/`).

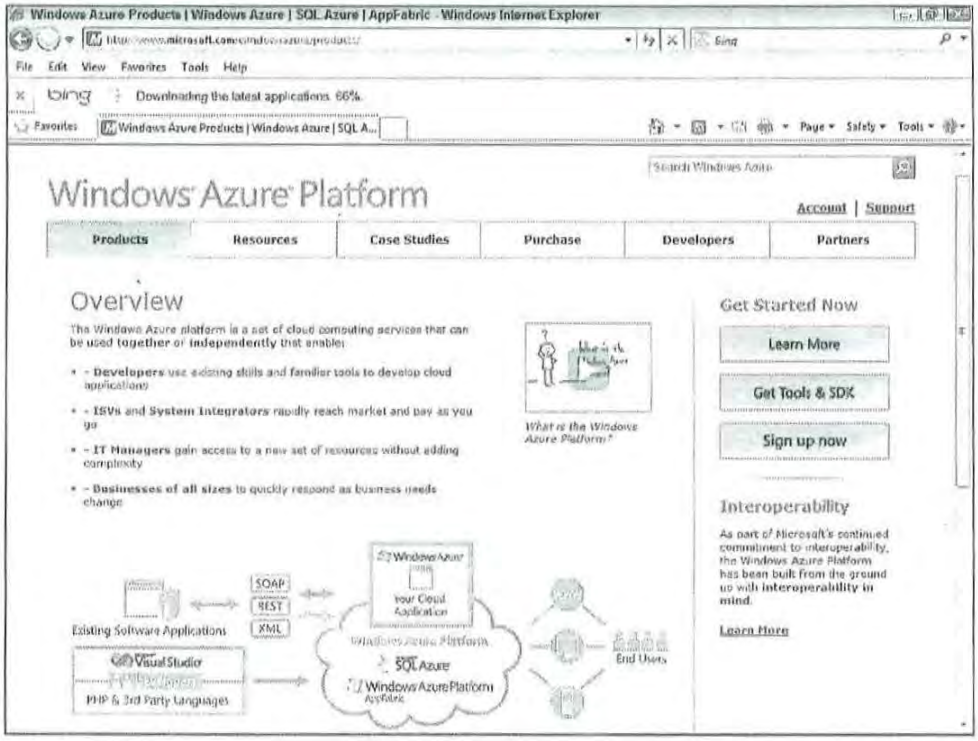


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Defining the Windows Azure Platform

Azure is Microsoft's Infrastructure as a Service (IaaS) Web hosting service. Azure is a deep blue color, the color of the clear sky onto which you can paint clouds. Taken together as a unit, Windows Azure Platform becomes a Platform as a Service (PaaS) offering. Hence, you may run into some people calling Azure an infrastructure service and others calling it a platform; in context, both are correct. Compared to Amazon's and Google's cloud services, Azure (the service) is a competitor to AWS. Windows Azure Platform is a competitor to Google's App Engine.

Figure 10.3 shows the home page of the Windows Azure Platform found at `http://www.microsoft.com/windowsazure`.

A developer creates an Azure application by first logging onto the Azure portal from the Sign up now button shown in Figure 10.3, supplying a Windows Live ID, creating a hosted account, and provisioning a storage account. The completed application can then be made available to users as a hosted application or service.

FIGURE 10.3

Window Azure Platform's home page may be found at `http://www.microsoft.com/windowsazure`, and is shown in this figure.

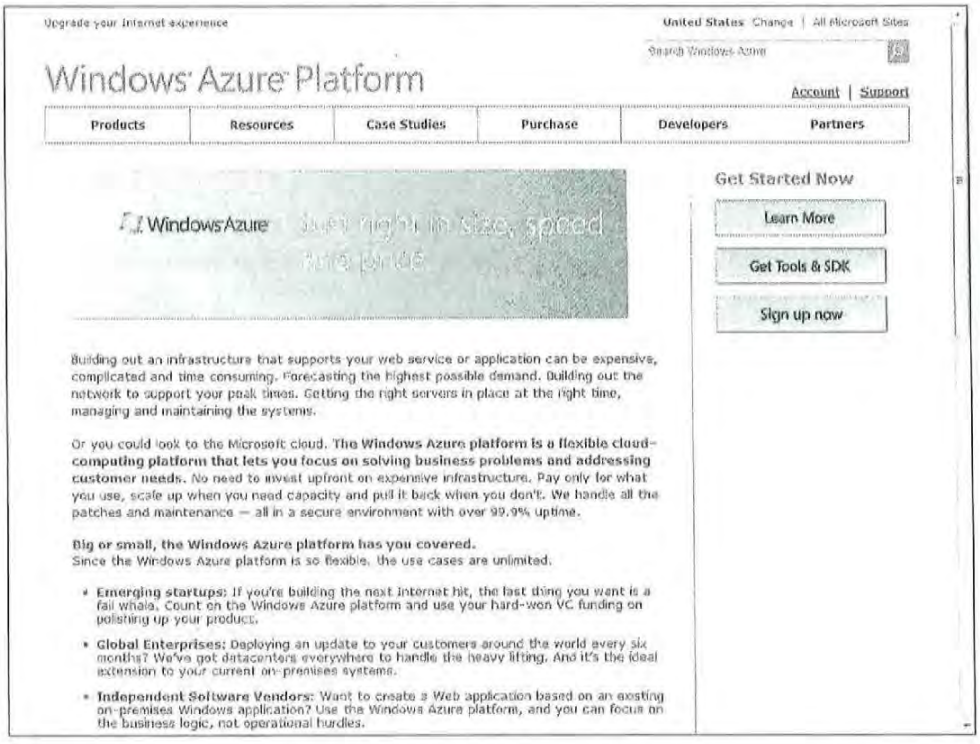


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The software plus services approach

Microsoft has a very different vision for cloud services than either Amazon or Google does. In Amazon's case, AWS is a pure infrastructure play. AWS essentially rents you a (virtual) computer on which to run your application. An Amazon Machine Image can be provisioned with an operating system, an enterprise application, or application stack, but that provisioning is not a prerequisite. An AMI is your machine, and you can configure it as you choose. AWS is a deployment enabler.

Google's approach with its Google App Engine (GAE) is to offer a cloud-based development platform on which you can add your program, provided that the program speaks the Google App Engine API and uses objects and properties from the App Engine framework. Google makes it possible to program in a number of languages, but you must write your applications to conform to Google's infrastructure. Google Apps lets you create a saleable cloud-based application, but that application can only work within the Google infrastructure, and the application is not easily ported to other environments.

Microsoft sees the cloud as being a complimentary platform to its other platforms. The company envisages a scenario where a Microsoft developer with an investment in an application wants to extend that application's availability to the cloud. Perhaps the application runs on a server, desktop, or mobile device running some form of Windows. Microsoft calls this approach *software plus services.*

The Windows Azure Platform allows a developer to modify his application so it can run in the cloud on virtual machines hosted in Microsoft datacenters. Windows Azure serves as a cloud operating system, and the suitably modified application can be hosted on Azure as a runtime application where it can make use of the various Azure Services. Additionally, local applications running on a server, desktop, or mobile device can access Windows Azure Services through the Windows Services Platform API.

Given that Microsoft owns the Office application market as well as the desktop OS market, this approach makes lots of sense. It is also quite possible that a hybrid application that can reside either locally or in the cloud will have lots of appeal not only to developers but to users who would prefer more control over their data and more security than the cloud might offer.

The Azure Platform

With Azure's architecture (shown in Figure 10.4), an application can run locally, run in the cloud, or some combination of both. Applications on Azure can be run as applications, as background processes or services, or as both. The Windows Azure service itself is shown as the oval in Figure 10.4 and is a cloud-based operating system with a fabric infrastructure of virtual machines hosted in Microsoft datacenters.

The Azure Windows Services Platform API uses the industry standard REST, HTTP, and XML protocols that are part of any Service Oriented Architecture cloud infrastructure to allow applications to talk to Azure. Developers can install a client-side managed class library that contains functions that can make calls to the Azure Windows Services Platform API as part of their applications. These

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

API functions have been added to Microsoft Visual Studio as part of Microsoft's Integrated Development Environment (IDE). There are plans to add IPsec connectivity to Azure in the near future. *IPsec* refers to the Internet Protocol Security protocol suite for creating a secure Internet connection between two endpoints. IPsec provides for authenticated communication using session-based negotiation and the exchange of cryptographic keys to enable encrypted communication to be sent and decrypted. IPsec is an IETF standard that is in wide use.

The Azure Service Platform hosts runtime versions of .NET Framework applications written in any of the languages in common use, such as Visual Basic, C++, C#, Java, and any application that has been compiled for .NET's Common Language Runtime (CLR). Azure also can deploy Web-based applications built with ASP.NET, the Windows Communication Foundation (WCF), and PHP, and it supports Microsoft's automated deployment technologies. Microsoft also has released SDKs for both Java and Ruby to allow applications written in those languages to place calls to the Azure Service Platform API to the AppFabric Service.

The Windows Azure service

Windows Azure is a virtualized Windows infrastructure run by Microsoft on a set of datacenters around the world. In Figure 10.4, the dashed oval encloses the portion of the Windows Azure Platform that is Azure itself—that is, the portion of the platform that is the IaaS part, which is shown in more detail in Figure 10.5.

Six main elements are part of Windows Azure:

- **Application:** This is the runtime of the application that is running in the cloud.
- **Compute:** This is the load-balanced Windows server computation and policy engine that allows you to create and manage virtual machines that serve either in a Web role and a Worker role.

  A Web role is a virtual machine instance running Microsoft IIS Web server that can accept and respond to HTTP or HTTPS requests. A Worker role can accept and respond to requests, but doesn't run IIS in that virtual machine. Worker roles can communicate with Azure Storage or through direct connections to clients.
- **Storage:** This is a non-relational storage system for large-scale storage.

  Azure Storage Service lets you create drives, manage queues, and store BLOBs (Binary Large Objects). You manipulate content in Azure Storage using the REST API, which is based on standard HTTP requests and is therefore platform-independent. Stored data can be read using `GET`s, written with `PUT`s, modified with `POST`s, and removed with `DELETE` requests.

  Azure Storage plays the same role in Azure that Amazon Simple Storage Service (S3) plays in Amazon Web Services. For relational database services, SQL Azure may be used.
- **Fabric:** This is the Windows Azure Hypervisor, which is a version of Hyper-V that runs on Windows Server 2008.
- **Config:** This is a management service.
- **Virtual machines:** These are instances of Windows that run the applications and services that are part of a particular deployment.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.4

The Windows Azure Platform extends applications running on other platforms to the cloud using Microsoft infrastructure and a set of enterprise services.

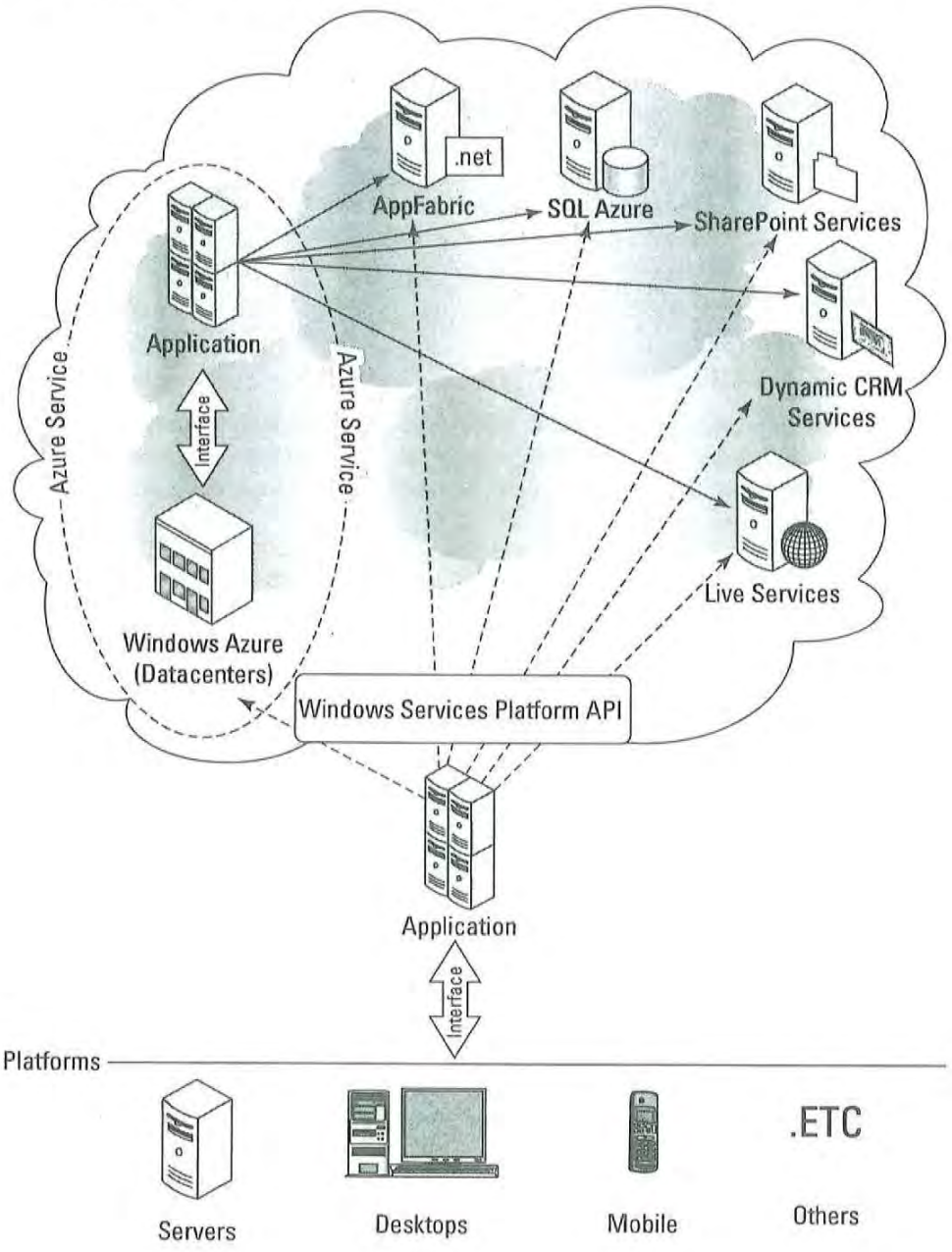


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.5

Windows Azure is a virtualized infrastructure that provides configurable virtual machines, independent storage, and a configuration interface.

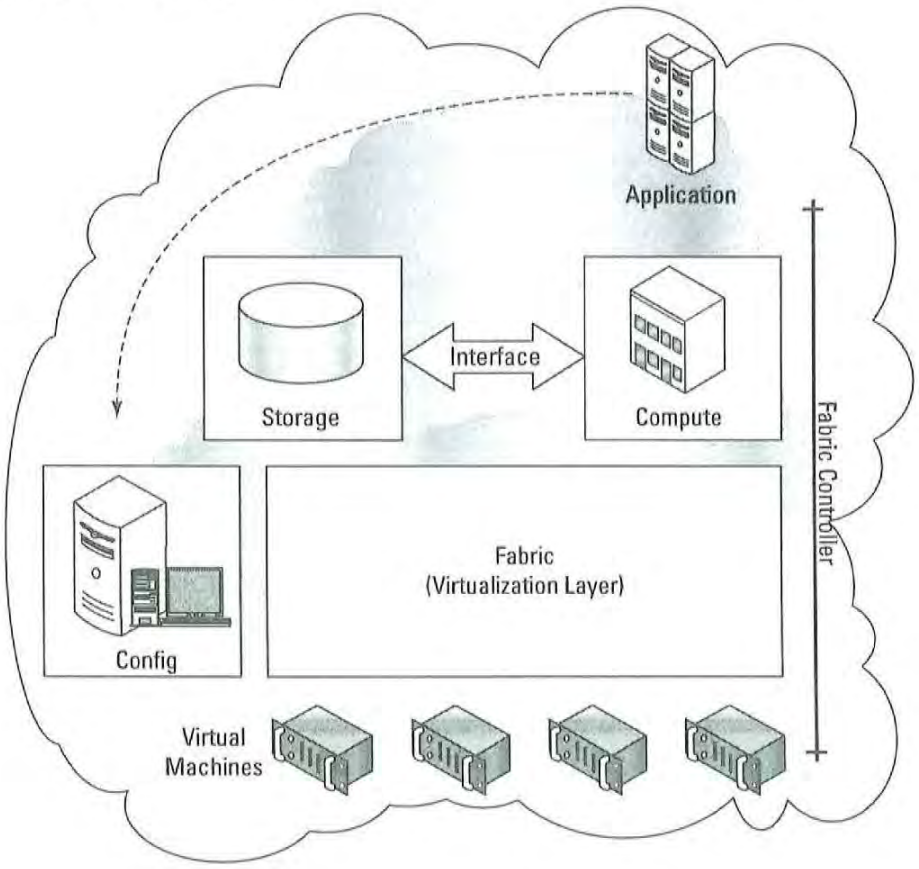


Table 10.1 shows the different Virtual Machine sizes available on Windows Azure.

TABLE 10.1

Windows Azure Virtual Machine Sizes

VM Size1	CPU Cores	Memory (GB)	Disk Space for Local Storage Resources (GB)
Small	1	1.7	250
Medium	2	3.5	500
Large	4	7	1000
ExtraLarge	8	14	2000

1. Microsoft has not released information that would allow you to match VM sizes to physical systems based on real CPUs.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The portion of the Azure environment that creates and manages a virtual resource pool is called the Fabric Controller. Applications that run on Azure are memory-managed, load-balanced, replicated, and backed up through snapshots automatically by the Fabric Controller.

Windows Azure AppFabric

Azure AppFabric (`http://msdn.microsoft.com/en-us/windowsazure/netservices.aspx`) is a Service Bus and Access Control facility based on .NET technology for client requests to Web services on Azure. Previously, these services were called Microsoft .NET Services. Azure AppFabric supports the standard Service Oriented Architecture (SOA) protocols such as REST and SOAP and the WS- protocols.

The function of a service bus in an SOA is to expose distributed services as an endpoint with a specific URI that clients can request services from, as shown in Figure 10.6. A particular set of endpoints and its associated Access Control rules for an application is referred to as the service namespace. Each namespace is assigned a management key that is part of the security mechanism. The Service Bus service registry makes endpoints discoverable, if so configured.

Azure AppFabric manages requests by locating the service, communicating the request, and making the necessary connection possible by performing network address translation, opening appropriate ports in any intervening firewalls. AppFabric manages the transaction to ensure that it is completed and that a response is sent to the client. A service bus also can serve to negotiate the exchange of information between a client and the service.

Azure AppFabric acts as an SOA service bus, as shown in Figure 10.6. AppFabric can provide a negotiated traversal of services through firewalls and NATs as a relay service using the Service Bus' rendezvous address. A rendezvous address not only includes the service URI, but also includes the namespace of the service bus. Alternatively, if both applications comply to .NET Services a direct connection between the applications can be used instead with the required NAT traversal information for the direct connection provided by the relay service of the Service Bus. NAT (Network Address Traversal) is a system for creating and maintaining Internet connections for TCP or UDP traffic where the connection point is hidden behind a router or a firewall and routing is performed by one of several possible mechanisms.

Cross-Ref

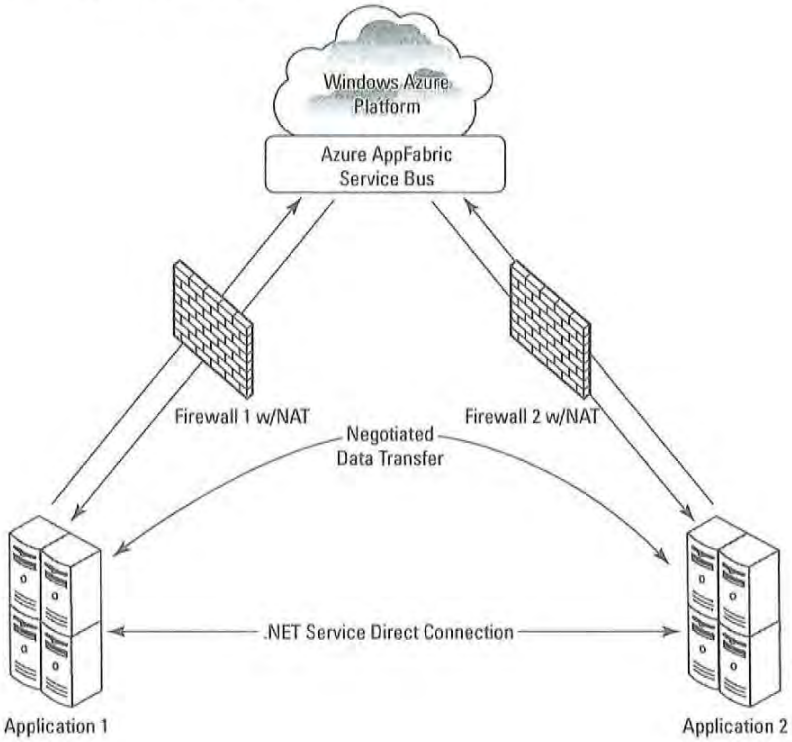
In Chapter 13, "Understanding Service Oriented Architecture," the role of a service bus in a Service Oriented Architecture is more fully explored. ■

The Access Control portion of Azure AppFabric is a claims access control system that provides a token-based trust mechanism for identity management. An application or user, as shown on the right of Figure 10.7, presents a claim for a service from an application on the left. The Access Control examines the request, and if it finds it to be valid, it grants a security token to the client.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.6

Azure AppFabric service pathways



These steps are associated with Access Control:

1. The client requests authentication from Access Control.
2. Access Control creates a token based on the stored rules for server application.
3. A token is signed and returned to the client application.
4. The client presents the token to the service application.
5. The server application verifies the signature and uses the token to decide what the client application is allowed to do.

Access Control allows one application to trust the identity of another application. This mechanism can federate with identity providers such as Active Directory Federation Services (ADFS v2) to create distributed systems based on SOA.

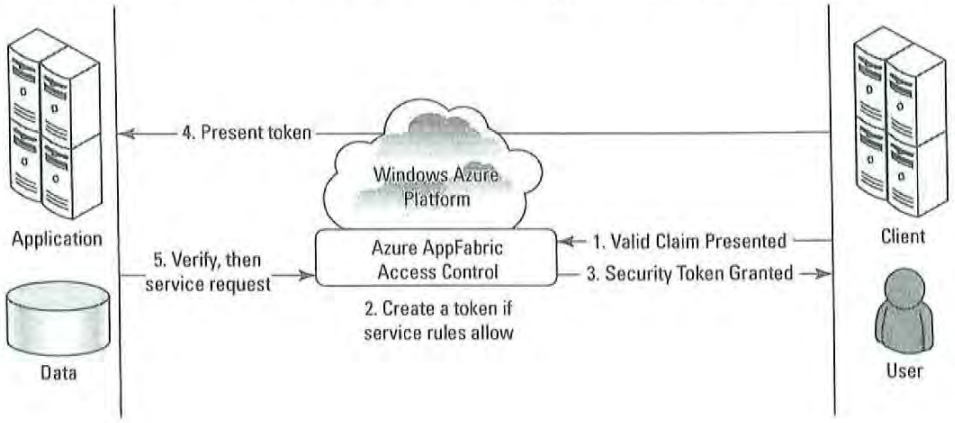
Note

"AppFabric" is also used by Microsoft for the name of its local server deployment technology called Windows Server AppFabric (`http://msdn.microsoft.com/en-us/windowsserver/ee695849.aspx`). Windows Server AppFabric enables Web data caching for application data and provides managed services using WindowsWorkflow Foundation and the Windows Communication Foundation. There are plans to integrate Windows Server AppFabric into the Azure platform AppFabic. ■

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.7

Azure AppFabric Access Control enables secure application requests through a token mechanism.



Microsoft likes to refer to the Azure AppFabric as an "Internet Service Bus" to differentiate it from the standard Enterprise Service Bus (ESB) that you find in SOA architectures. AppFabric has all the same components of an ESB, namely service orchestration, federated identity, access control, a namespace, service registry, and a messaging fabric, but it locates these components in the cloud. Often ESBs are located on LANs. According to Microsoft, this approach abstracts away from application developers the challenges related to NAT traversal, DDNS (Dynamic DNS), and UPnP. ESBs are described in Chapter 13; please refer to that chapter for further discussion on this topic.

Azure Content Delivery Network

The Windows Azure Content Delivery Network (CDN) is a worldwide content caching and delivery system for Windows Azure blob content. Currently, more than 18 Microsoft datacenters are hosting this service in Australia, Asia, Europe, South America, and the United States, referred to as endpoints. CDN is an edge network service that lowers latency and maximizes bandwidth by delivering content to users who are nearby.

Any storage account can be enabled for CDN. In order to share information stored in an Azure blob, you need to place the blob in a public blob container that is accessible to anyone using an anonymous sign-in. The Azure portal lists the domain name of the blob container in the form `http://<guid>.vo.msecnd.net/`. You also can register a custom domain name for a Windows Azure CDN endpoint.

For content in a public container named "Box" in the storage account named "MyAccount," a user would access the content with one of the following URLs:

- Windows Azure Blob services URL: `http://<MyAccount>.blob.core.windows.net/<Box>/`
- Windows Azure CDN URL: `http://<guid>.vo.msecnd.net/<Box>/`

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

When the Blob service URL is used, the request is redirected to the closest CDN endpoint to the client. The CDN service searches that location and serves the content; if the content isn't found, the CDN retrieves the Blob from the Blob service, caches the content, and then serves it to the user. Parameters can be set that determine how long content is cached (Time-To-Live, TTL), with the default being 72 hours.

SQL Azure

SQL Azure is a cloud-based relational database service that is based on Microsoft SQL Server. Initially, this service was called SQL Server Data Service. An application that uses SQL Azure Database can run locally on a server, PC, or mobile device, in a datacenter, or on Windows Azure. Data stored in an SQL Azure database is accessed using the Tabular Data Stream (TDS) protocol, the same protocol used for a local SQL Server database. SQL Azure Database supports Transact-SQL statements.

Azure data is replicated three times for data protection and writes are checked for consistency. SQL Azure eventually will support the Microsoft Sync Framework providing a facility for SQL Azure Databases to synchronize their data with local databases.

There is a current limit of 10GB for each SQL Azure Database. Queries against a single database are unified. However, if the storage size exceeds the limit, then data must be partitioned into logical sets and queries need to be structured to account for this partitioning. For example, names in a database might have to be partitioned A-K, L-R, and S-Z. SQL Azure Database is a shared database environment, and limitations are placed on how long a query can run or how many resources a query can use.

Note

Microsoft hopes to create a cloud-based global data marketplace using SQL Azure for stored information, a project that it has codenamed "Dallas." Applications will then be able to access both private and public domain data such as imagery, census data, statistical data, and other premium content using REST protocols. You can read about Microsoft's plans for Dallas at `http://www.microsoft.com/windowsazure/dallas/`. ■

From the standpoint of any application, an SQL Azure Database looks like and behaves like a local database with a few exceptions. The current exceptions are that the SQL Common Language Runtime (CLR) and support for spatial data were not included, although support will be added later for them. The biggest difference is that because SQL Azure is managed in the cloud, there are no administrative controls over the SQL engine. You can't shut the system down, nor can you directly interact with the SQL Servers.

Windows Azure pricing

Prices for working with the Windows Azure Platform are based either on a "consumption" (pay-as-you-go) model or through various contracts for levels of monthly service that Microsoft calls "commitments." When you exceed the subscription level of your commitment, the additional usage is charged on the consumption model.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Current pricing for Windows Azure is as follows:

- Compute: $0.12 / hour
- Storage: $0.15 / GB stored / month
- Storage transactions: $0.01 / 10K
- Data transfers (excluding CDN): $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)
- CDN data transfers: $0.15 GB for North America and Europe ($0.20 GB elsewhere)
- CDN transactions: $0.01 / 10K

  A transaction is an application request.

In the Windows Azure Service Level Agreement, Microsoft states that it guarantees an external connectivity between two or more role instances that are located in different Azure domains of at least 99.95 percent uptime. The connection between storage and Microsoft's Content Delivery Network (CDN, described below) is stated to be at least 99.9 percent uptime.

SQL Azure charges are based on two different programs:

- Web Editions: Up to 1GB database = $9.99 / month; up to 5GB database = $49.95 / month
- Business Edition: Up to 10GB database = $99.99 / month; up to 20GB database = $199.98 / month; up to 30GB database = $299.97 / month; up to 40GB database = $399.96 / month; up to 50GB database = $499.95 / month
- Data transfers: $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)

Note

The various consumption and subscription offers for the Windows Azure platform are summarized on the Pricing page at `http://www.microsoft.com/windowsazure/pricing/`. ■

These are the charges for Windows Azure Platform AppFabric:

- Access Control transactions of $1.99 / 100K transactions
- Service Bus connections: $3.99 per connection on a "pay-as-you-go" basis, $9.95 for a pack of 5 connections, $49.75 for a pack of 25 connections, $199 for a pack of 100 connections, and $995 for a pack of 500 connections
- Data transfers: $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)

Given that Windows Azure is a relatively new service and that IaaS likely will become a very competitive market, pricing is sure to change over time. You should definitely check the pricing page for current pricing if you are thinking of deploying on Azure.

Microsoft offers a TCO calculator for the Windows Azure Platform that you may find useful in determining your costs and savings. To access the calculator use the following link: `http://www.microsoft.com/windowsazure/economics/#tcoCompare-LB`. The calculator was described in brief in Chapter 2, "Computing the Total Cost of Ownership."

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Windows Live services

Windows Live is a collection of cloud-based applications and services, some of which can be used inside applications that run on Windows Azure Platform. Some Windows Live applications run as standalone applications and are available to users directly through a browser. Others are services that add capabilities to the Windows Azure Platform as part of Microsoft's software plus services strategy.

Microsoft has rolled out Windows Live in sets of releases they describe as four waves. The first wave was a rebranding of several Microsoft MSN applications and services in late 2005. More applications including Windows Mail, Windows Photo Gallery, and Windows Movie Maker were unbundled from Vista and rolled into a downloadable software suite called Windows Live Essentials. There has been continuous development, branding, marketing, and rebranding of the Windows Live portfolio that has had many people scratching their heads. Many Windows Live applications have been rolled into other services or discontinued entirely.

Here's what I believe the current situation is with Windows Live. If an application is bundled as part of an additional download for desktop users, it is part of the Windows Live Essentials package. Some applications that are part of Windows Live are standalone products, while others are extensions of existing Microsoft commercial software. An example of a standalone product would be Windows Live Calendar. An example of a cloud-based line extension is Windows Live Office, described more fully in Chapter 16.

Some parts of the Windows Live portfolio are shared applications and services that are accessible to developers, and those services are the Windows Live Services that are one component of the Windows Azure Platform. Developers access the services for Windows Live Services through a collection of APIs and controls called Windows Live Messenger Connect (previously called Live Services and Windows Live Dev). Using these APIs and controls, developers can add Windows Live Services capabilities and data to their application.

Note

To learn more about Windows Live Messenger Connect, visit the MSDN site's documentation found at `http://msdn.microsoft.com/en-us/library/ff749458.aspx.` ■

Messenger Connect was released as part of the Windows Live Wave 4 at the end of June 2010, and it unites APIs such as Windows Live ID, Windows Live Contacts, and Windows Live Messenger Web Toolkit into a single API. Messenger Connect works with ASP.NET, Windows Presentation Foundation (WPF), Java, Adobe Flash, PHP, and Microsoft's Silverlight graphics rendering technology through four different methods:

- Messenger Connect REST API Service
- Messenger Connect .NET and Silverlight Libraries
- Messenger Connection JavaScript Libraries and Controls
- Web activity feeds, either RSS 2.0 or ATOM

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Table 10.2 lists the current services that can be used by Windows Live Messenger Connect in applications and Web sites.

TABLE 10.2

Windows Live Services

Service Name	URL	Microsoft Description
Admin Center	Windows Live Admin Center SDK	A management utility for a domain using SOAP and RPC.
Alerts	Windows Live Alerts for RSS Feeds	Enables Windows Live Alerts from an RSS feed.
Alerts	Windows Live Alerts SDK	Allows developers to add Windows Live Alerts notification service to an application using SOAP.
Contacts	Windows Live Contacts API	Allows developers to use REST to query the Windows Live People Address Book, as well as to adjust permission to contact data based on the Windows Live ID Delegated Authentication protocol.
FeedSync	FeedSync	Synchronizes information obtained from RSS and ATOM sources.
Live Framework	Live Framework SDK	An API for building Live Mesh application based on Windows Live Services.
Live Framework	Live Framework Tools for Visual Studio	Includes the Live Mesh tools from Visual Studio 2008 and Visual Web Developer Express Edition 2008.
Messenger	Web Toolkit	UI controls for building Web applications using Windows Live Messenger.
Messenger	IM Control	A set of controls that can enable instant messaging in an application.
Messenger	Presence API	An API that can be used to indicate a Windows Live Messenger's presence and control instant messages to that person's browser using a set of HTTP commands.
Photo Gallery	Windows Live Photo Gallery SDK	Allows for the creation and editing of photos and videos using the Publishing Plug-in Platform of Windows Live Photo Gallery inside applications.
Spaces	Windows Live Spaces MetaWeblog API	An API that can use XML-RPC calls to get and send Weblog data.
Spaces	Windows Live Spaces API and Feeds	An API that integrates Windows Live Spaces, Windows Live Events, Windows Live Photos, and Windows Live Profile into applications.
Web Gadgets	Gadgets SDK	Lightweight, single-purpose applets that can run on Windows Live Personalized Experience and Windows Live Spaces.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Service Name	URL	Microsoft Description
Windows Live ID	Web Authentication	Used to integrate Windows Live ID authentication into a Web site.
Windows Live ID	Delegated Authentication	Allows an application to access data for an authenticated Windows Live ID user from Web services and sites that accept that authentication.
Windows Live ID	Client Authentication	An API for Windows Live ID sign-in from a desktop application.
Writer	Windows Live Writer SDK	Allows applications to incorporate the features of the Windows Live Writer in their application. Additional capabilities include features for creating and managing blogs, adding more content, and customizing the Windows Live Writer user interface.

Reference: Based on `http://en.wikipedia.org/wiki/Windows_Live_Messenger_Connect`. An API for Bing and the toolbar is also available as a service.

Using Windows Live

Windows Live includes several popular cloud-based services. The two best known and most widely used are Windows Live Hotmail and Windows Live Messenger, with more than 300 million users worldwide. Windows Live is based around five core services:

- E-mail
- Instant Messaging
- Photos
- Social Networking
- Online Storage

A user or application can consume Windows Live in a number of ways. Some Windows Live applications are entirely cloud-based Web services, so users can use these applications from within any browser. The Office Live applications described more fully in Chapter 16, "Microsoft Office Web Apps," is an example of this sort of service. Some of these services are aimed at mobile devices and are referred to as Windows Live for Mobile (described below), and they are consumed on conforming mobile devices. Some of these applications are client-side applications that you download from Windows Live for use on your desktop, of which Windows Live Essentials is the primary example.

You can access Windows Live services in one of the following ways:

- By navigating to the service from the command on the navigation bar at the top of Windows Live
- By directly entering the URL of the service
- By selecting the application from the Windows Live Essentials folder on the Start menu

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

If you haven't signed into Windows Live during your session, Windows Live requests that you do so before allowing you to proceed.

Table 10.3 lists the current offerings of Windows Live Services.

TABLE 10.3

Windows Live Services Offerings

Service Name	URL	Description
Windows Live Account	http://account.live.com/	Management service for Windows Live ID and relationships.
Windows Live Admin Center	http://admin.live.com/	E-mail hosting for Web site owners.
Windows Live Alerts	http://alerts.live.com/	Generates alerts sent to e-mail, mobile device, or Windows Messenger.
Windows Live Calendar	http://calendar.live.com/	Calendar service with appointments, meetings, and events; can be shared with others.
Windows Live Contacts	http://contacts.live.com/	Address book service with synchronization feature.
Windows Live Devices	http://devices.live.com/	Synchronization and remote access service for files stored on PCs and mobile devices.
Windows Live Essentials	http://essentials.live.com/	Downloadable applications that supplement Microsoft Windows.
Windows Live Family Safety	http://fss.live.com/	Allows you to manage and monitor your children's Internet activity so they can surf the Web more safely.
Windows Live FrameIt	http://frameit.live.com/	Adds an RSS feed to digital photo frame devices.
Windows Live Gallery	http://gallery.live.com/	A collection of developer add-ons for Windows Live products.
Windows Live Groups	http://groups.live.com/	A group discussion, collaboration, sharing, and coordination tool.
Windows Live Home	http://home.live.com/	A personalization Web page and tool for Windows Live with status information and navigation features.
Windows Live Hotmail	http://hotmail.com/	A Web-based free e-mail service with contacts and calendar.
Windows Live ID	http://login.live.com/	A sign-on service shared by Windows Live applications.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Service Name	URL	Description
Windows Live Mail	http://mail.live.com	Desktop e-mail client with RSS; replaces Outlook Express and Windows Mail. You can use Live Mail to manage Gmail or Yahoo! Plus Mail accounts, as well as your POP e-mail services.
Windows Live Messenger	http://messenger.live.com/	Allows you to chat instantly with friends and family on your desktop, on the Web, and on your mobile phone.
Windows Live Messenger Companion	http://essentials.live.com/	Windows Live Essentials add-on for Internet Explorer; shares link to a page on the site you're visiting. You can see the page and add a comment.
Windows Live Movie Maker	http://essentials.live.com/	Allows you to create beautiful, memorable movies and then publish to the Web in a few clicks.
Windows Live Office	http://office.live.com/	Contains document creation and editing tools based on Office, Excel, PowerPoint, and OneNote with Windows Live SkyDrive storage.
Windows Live OneCare Safety Scanner	http://safety.live.com/	Consists of a PC scanner for viruses, spyware, and other malware. Features include disk cleaner, defragmenter, port scanner, and registry cleaner.
Windows Live Photo Gallery	http://photogallery.live.com/	Allows you to edit, organize, tag, and share your photos.
Windows Live Photos	http://photos.live.com/	Photo storage and sharing service. You can use the service to publish photos to third-party photo services.
Windows Live Profile	http://profile.live.com/	Profile information management service for user information.
Windows Live SkyDrive	http://skydrive.live.com/	Online file storage system service.
Windows Live Spaces	http://spaces.live.com/	Social networking, blogging, and photo-sharing site.
Windows Live Sync	http://sync.live.com/	File synchronization and sharing site based on Live Mesh; originally called folder share.
Windows Live Writer	http://writer.live.com/	Allows you to compose a blog post, add your photos and links to your videos, and then publish to the Web. You can post the blogs from Writer to Blogger, WordPress, and other services.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The following Windows Live services have been discontinued or rebranded:

- Windows Live Agent
- Windows Live Barcode
- Windows Live Call (now part of Windows Live Messenger)
- Windows Live Events
- Windows Live Expo
- Windows Live Favorites (now Part of Windows Live SkyDrive)
- Windows Live Help Community
- Windows Live Hotspot Locator (now MSN WiFi Hotspots)
- Windows Live OneCare (now Microsoft Security Essentials)
- Windows Live Personalized Experience
- Windows Live QnA (now MSN QnA)
- Windows Live Search Center (now Windows Search 4)
- Windows Live Shopping (now Bing Shopping)
- Windows Live Toolbar
- Windows Live TV
- Windows Live Video Messages
- Windows Live Web Messenger (now part of Windows Live Web services)
- Windows Live WiFi Center

Windows Live Essentials

Windows Live Essentials applications are a collection of client-side applications that must be downloaded and installed on a desktop. Some of these applications were once part of Windows and have been unbundled from the operating system; others are entirely new. Live Essentials rely on cloud-based services for their data storage and retrieval, and in some cases for their processing.

Windows Live Essentials currently includes the following:

- Family Safety
- Windows Live Messenger
- Photo Gallery
- Mail
- Movie Maker

The download page for Windows Live Essentials (`http://essentials.live.com/`) is shown in Figure 10.8. All the Windows Essentials are downloaded as a single file. This page also has links

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

to download related software such as the Bing bar (which replaces the Windows Live Toolbar), Microsoft Office Outlook Connection, Office Live Add-in, and Microsoft Silverlight. When you install Windows Live Essentials, shortcuts for these programs are placed on the Windows Start menu.

Windows Live Essentials help alleviate a long-standing problem of Microsoft with Windows by allowing Microsoft to unbundle some of its add-on applications for the operating system so they don't compete with other vendors' products unfairly. Live Essentials moves these applications partially onto the cloud, while making them available easily as a download and a service. Shown in Figure 10.9 is Windows Live Family Safety, which is a Web filter and activity reporting tool for Windows accounts on a per-machine basis.

FIGURE 10.8

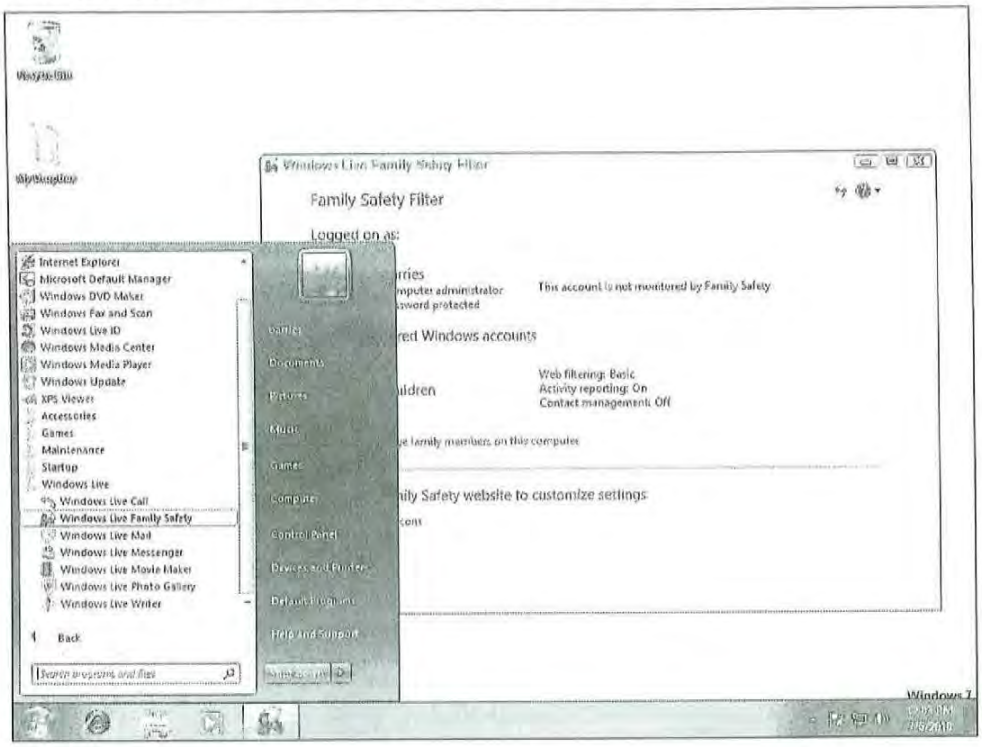
The Windows Live Essentials home page (`http://essentials.live.com/`) provides links to the downloads of Microsoft's cloud-based client-side applications, the application's own page, as well as links to download related software.



Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.9

Windows Live Essentials is available from the Start menu as a set of commands.



Windows Live Home

Windows Live Home is the central access page or portal for the Windows Live suite. An example is shown in Figure 10.10. The page provides navigation, lists activities, provides access to e-mail, shows your RSS feeds, and lists your account name and some related information. What you see on this page is customizable and depends on the services to which you are subscribed. The page can be themed, which changes the color, fonts, and look of the page.

These are the most commonly used features on Windows Live Home:

- Launching other Windows Live services
- Viewing e-mail headers from Hotmail and private messages from other users
- Viewing activity of people you follow
- Displaying weather information and RSS feed updates
- Managing calendars and events
- Viewing photos
- Modifying profile and relationships

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 10.10

Your personalized home page for Windows Live (`http://live.microsoft.com/home`) contains content and ads.



Windows Live for Mobile

Microsoft has a number of Windows Live services that are specifically meant to be run on mobile devices or cell phones that it calls Windows Live for Mobile (`http://mobile.live.com`). Some of these services run on the Windows Mobile platform, some are Web-based applications that conform to the lightweight Wireless Application Protocol (WAP) or on GPRS (General Packet Radio Service) browser, and some support SMS (Simple Message Service) systems. The current list of these services includes the following:

- Live Mesh Mobile
- Windows Live Calendar Mobile
- Windows Live Contacts Mobile
- Windows Live Groups Mobile
- Windows Live Home Mobile
- Windows Live Messenger Mobile

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- Windows Live Office Mobile
- Windows Live Profile Mobile
- Windows Live SkyDrive Mobile
- Windows Live Spaces Mobile

To download these client applications on supported devices, go to `http://gowindowslive.com/mobile` and select the download.

Summary

In this chapter, I described Microsoft cloud computing strategy. Microsoft seeks to extend its products into the cloud using a software plus service approach. In this model, the cloud is yet another platform, and applications can run locally and access cloud services, run in the cloud and be made available through SOA standard protocols, or some combination of both.

Microsoft's cloud operating system is called Windows Azure. Windows Azure is a hosted environment of virtualized systems tied together in a fabric using a service called AppFabric. This is offered to developers in the form of an Infrastructure as a Service model similar to Amazon Web Services. To Windows Azure is added a cloud-enabled version of the .NET Framework originally called .NET Services, which are now part of Azure AppFabric. This approach lets developers extend their applications into the cloud using development tools that they already possess with the minimum amount of reconfiguration. Microsoft has added a number of additional services and the entire offering is now a Platform as a Service cloud model that Microsoft calls the Windows Azure Platform.

The other major component of Microsoft's cloud computing strategy is a collection of user applications and related services called Windows Live. Some Windows Live applications are client-side applications, many others are Web-based applications, some are mobile apps, and an important subset of these services is available to developers through the Windows Live Messenger Connect APIs. The various offering in Windows Live were discussed in this chapter.

In Chapter 11, "Managing the Cloud," you learn about some of the management tools used to work with cloud applications and methods used for application deployment.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 11

Managing the Cloud

Cloud computing deployments must be monitored and managed in order to be optimized for best performance. To the problems associated with analyzing distributed network applications, the cloud adds the complexity of virtual infrastructure. This is one of the most active areas of product development in the entire cloud computing industry, and this chapter introduces you to the different products in this nascent area.

Cloud management software provides capabilities for managing faults, configuration, accounting, performance, and security; this is referred to as FCAPS. Many products address one or more of these areas, and through network frameworks, you can access all five areas. Framework products are being repositioned to work with cloud systems.

Your management responsibilities depend on the particular service model for your cloud deployment. Cloud management includes not only managing resources in the cloud, but managing resources on-premises. The management of resources in the cloud requires new technology, but management of resources on-premises allows vendors to use well-established network management technologies.

The lifecycle of a cloud application includes six defined parts, and each must be managed. In this chapter, the tasks associated with each stage are described.

Efforts are underway to develop cloud management interoperability standards. One effort you learn about in this chapter is the DMTF's (Distributed Management Task Force) Open Cloud Standards Incubator. The goal of these efforts is to develop management tools that work with any cloud type. Another group called the Cloud Commons is developing a technology called the Service Measurement Index (SMI). SMI aims to deploy methods for measuring various aspects of cloud performance in a standard way.

IN THIS CHAPTER

Learning about network management software

Viewing the essential monitoring features

Using lifecycle management techniques

Discovering emerging network management interoperability standards

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Administrating the Clouds

The explosive growth in cloud computing services has led many vendors to rename their products and reposition them to get in on the gold rush in the clouds. What was once a network management product is now a cloud management product. Nevertheless, this is one area of technology that is very actively funded, comes replete with interesting startups, has been the focus of several recent strategic acquisitions, and has resulted in some interesting product alliances. Let's join the party and see what all the fuss is about.

These fundamental features are offered by traditional network management systems:

- Administration of resources
- Configuring resources
- Enforcing security
- Monitoring operations
- Optimizing performance
- Policy management
- Performing maintenance
- Provisioning of resources

Network management systems are often described in terms of the acronym FCAPS, which stands for these features:

- Fault
- Configuration
- Accounting
- Performance
- Security

Most network management packages have one or more of these characteristics; no single package provides all five elements of FCAPS.

To get the complete set of all five of these management areas from a single vendor, you would need to adopt a network management framework. These large network management frameworks were industry leaders several years back: BMC PATROL, CA Unicenter, IBM Tivoli, HP OpenView, and Microsoft System Center. Network framework products have been sliced and diced in many different ways over the years, and they are rebranded from time to time. Today, for example, BMC PATROL is now part of BMC ProactiveNet Performance Management (`http://www.bmc.com/products/product-listing/ProactiveNet-Performance-Management.html`), HP OpenView has been split (`https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10^36657_4000_100`) into a set of HP Manager products.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The impact that cloud computing is having on network frameworks is profound. These five vendors have (or soon will have) products for cloud management. Computer Associates, for example, has completely repositioned its network management portfolio as an IT Management Software as a Service. Find the cloud products for these five large cloud vendors at the following URLs:

- BMC Cloud Computing (`http://www.bmc.com/solutions/esm-initiative/cloud-computing.html`)
- Computer Associates Cloud Solutions (`http://www.ca.com/us/cloud-computing.aspx`)
- HP Cloud Computing (`http://h20338.www2.hp.com/enterprise/w1/en/technologies/cloud-computing-overview.html`)
- IBM Cloud Computing (`http://www.ibm.com/ibm/cloud/`)
- Microsoft Cloud Services (`http://www.microsoft.com/cloud/`)

Figure 11.1 shows IBM Tivoli Service Automation Manager, a framework tool for managing cloud infrastructure.

FIGURE 11.1

Tivoli Service Automation Manager lets you create and stage cloud-based servers.

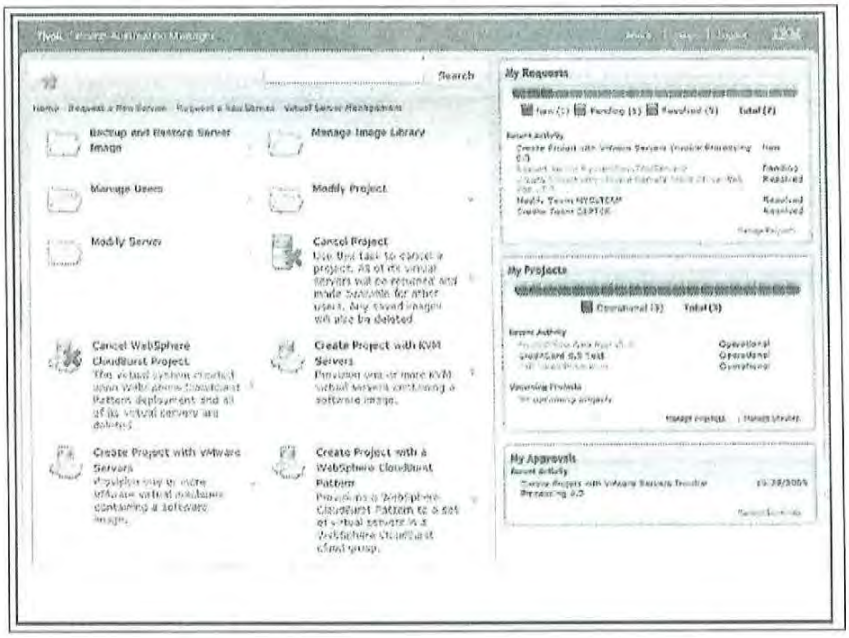


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Management responsibilities

What separates a network management package from a cloud computing management package is the "cloudly" characteristics that cloud management service must have:

- Billing is on a pay-as-you-go basis.
- The management service is extremely scalable.
- The management service is ubiquitous.
- Communication between the cloud and other systems uses cloud networking standards.

To monitor an entire cloud computing deployment stack, you monitor six different categories:

1. End-user services such as HTTP, TCP, POP3/SMTP, and others
2. Browser performance on the client
3. Application monitoring in the cloud, such as Apache, MySQL, and so on
4. Cloud infrastructure monitoring of services such as Amazon Web Services, GoGrid, Rackspace, and others
5. Machine instance monitoring where the service measures processor utilization, memory usage, disk consumption, queue lengths, and other important parameters
6. Network monitoring and discovery using standard protocols like the Simple Network Management Protocol (SNMP), Configuration Management Database (CMDB), Windows Management Instrumentation (WMI), and the like

It's important to note that there are really two aspects to cloud management:

Managing resources *in the cloud*

Using the cloud to manage resources *on-premises*

When you move to a cloud computing architecture from a traditional networked model like client/server or a three-tier architecture, many of the old management tasks for processes going on in the cloud become irrelevant or nearly impossible to manage because the tools to effectively manage resources of various kinds fall outside of your own purview. In the cloud, the particular service model you are using directly affects the type of monitoring you are responsible for.

Consider the case of an Infrastructure as a Service vendor such as Amazon Web Services or Rackspace. You can monitor your usage of resources either through their native monitoring tools like Amazon CloudWatch or Rackspace Control Panel or through the numerous third-party tools that work with these sites' APIs. In IaaS, you can alter aspects of your deployment, such as the number of machine instances you are running or the amount of storage you have, but you have very limited control over many important aspects of the operation. For example, your network bandwidth is locked into the type of instance you deploy. Even if you can provision more bandwidth, you likely have no control over how network traffic flows into and out of the system, whether there is packet prioritization, how routing is done, and other important characteristics.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The situation—as you move first to Platform as a Service (PaaS) like Windows Azure or Google App Engine and then onto Software as a Service (SaaS) for which Salesforce.com is a prime example—becomes even more restrictive. When you deploy an application on Google's PaaS App Engine cloud service, the Administration Console provides you with the following monitoring capabilities:

- Create a new application, and set it up in your domain.
- Invite other people to be part of developing your application.
- View data and error logs.
- Analyze your network traffic.
- Browse the application datastore, and manage its indexes.
- View the application's scheduled tasks.
- Test the application, and swap out versions.

However, you have almost no operational control. Essentially, Google App Engine lets you deploy the application and monitor it, and that's about it. All the management of devices, networks, and other aspects of the platform are managed by Google. You have even less control when you are selling software in the cloud, as you would with Salesforce.com.

Figure 11.2 graphically summarizes the management responsibilities by service model type.

The second aspect of cloud management is the role that cloud-based services can play in managing on-premises resources. From the standpoint of the client, a cloud service provider is no different than any other networked service. The full range of network management capabilities may be brought to bear to solve mobile, desktop, and local server issues, and the same sets of tools can be used for measurement.

Microsoft System Center is an example of how management products are being adapted for the cloud. System Center provides tools for managing Windows servers and desktops. The management services include an Operations Manager, the Windows Service Update Service (WSUS), a Configuration Manager for asset management, a Data Protection Manager, and a Virtual Machine Manager, among other components.

One of these service sets was called the System Center Online Desktop Manager (SCODM). Microsoft has taken SCODM and repositioned it as a cloud-based service for managing updates, monitoring PCs for license compliance and health, enforcing security policies, and using Forefront protect systems from malware, and the company has branded it as Windows Intune (`http://www.microsoft.com/windows/windowsintune/default.aspx`). From the client's standpoint, it makes little difference whether the service is in the cloud or on a set of servers in a datacenter. The benefit of a cloud management service accrues to the organization responsible for managing the desktops or mobile devices. Figure 11.3 shows an Overview screen from the beta version of Windows Intune. The product is due to be released in the first or second quarter of 2011.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 11.2

Management responsibilities by service model type

	Hosted	Managed services	Cloud (IaaS)	Cloud (PaaS)	SaaS
Example(s)	Hosted infrastructure	Network VoIP	Amazon AWS, Rackspace Cloud server	Google App Engine Microsoft Azure	Salesforce.com
IT primary responsibilities					
Provider primary responsibilities		*Varies by business agreement*			
Shared responsibilities					

Business service/ user satisfaction · Application · Database · Server · Operating system · Network

FIGURE 11.3

Intune is Microsoft's cloud-based management service for Windows systems.

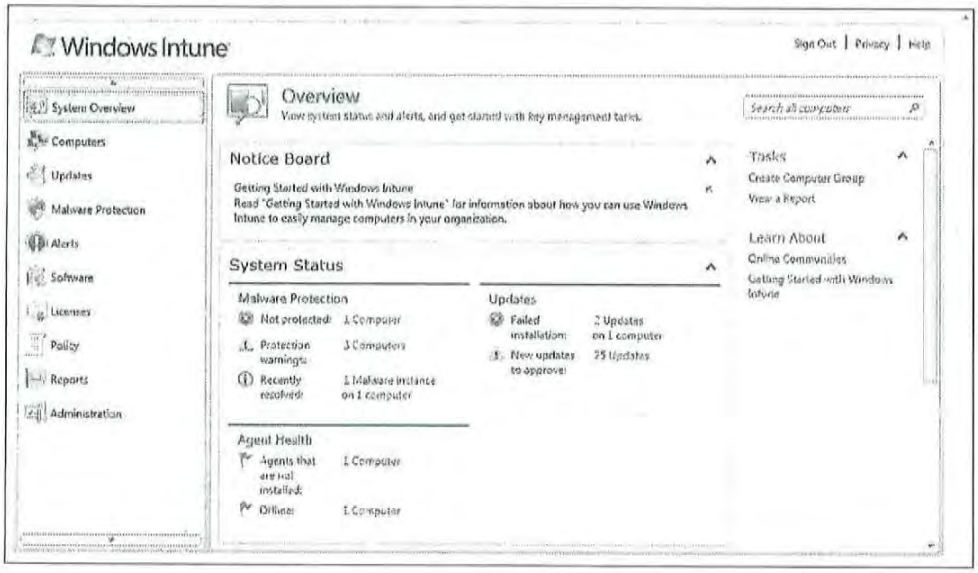


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Lifecycle management

Cloud services have a defined lifecycle, just like any other system deployment. A management program has to touch on each of the six different stages in that lifecycle:

1. The definition of the service as a template for creating instances

   Tasks performed in Phase 1 include the creation, updating, and deletion of service templates.

2. Client interactions with the service, usually through an SLA (Service Level Agreement) contract

   This phase manages client relationships and creates and manages service contracts.

3. The deployment of an instance to the cloud and the runtime management of instances

   Tasks performed in Phase 3 include the creation, updating, and deletion of service offerings.

4. The definition of the attributes of the service while in operation and performance of modifications of its properties

   The chief task during this management phase is to perform service optimization and customization.

5. Management of the operation of instances and routine maintenance

   During Phase 5, you must monitor resources, track and respond to events, and perform reporting and billing functions.

6. Retirement of the service

   End of life tasks include data protection and system migration, archiving, and service contract termination.

Cloud Management Products

Cloud management software and services is a very young industry, and as such, it has a very large number of companies, some with new products and others with older products competing in this area. Table 11.1 shows some of the current players in this market, along with the products they either are offering or are promising in the very near future. When considering products in cloud management, you should be aware that—as in all new areas of technology—there is considerable churn as companies grow, get acquired, or fail along the way. It is entirely possible that if you return to this list a year or two after this book is published, half of these products or services will no longer exist as listed; you should keep this in mind.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

TABLE 11.1

Cloud and Web Monitoring Solutions

Product	URL	Description
AbiCloud	http://www.abiquo.com/	Virtual machine conversion and management
Amazon CloudWatch	http://aws.amazon.com/cloudwatch/	AWS dashboard
BMC Cloud Computing Initiative	http://www.bmc.com/solutions/esm-initiative/cloud-computing.html	Cloud planning, lifecycle management, optimization, and guidance
CA Cloud Connected Management Suite	http://www.ca.com/us/cloud-solutions.aspx	CA Cloud Insight, CA Cloud Compose, CA Cloud Optimize, and CA Cloud Orchestrate are described below
Cacti	http://www.cacti.net/	Network performance graphing solution
CloudKick	https://www.cloudkick.com/	Cloud server monitoring
Dell Scalent	http://www.scalent.com/index.php	Virtualization provisioning system that will be rolled into Dell's Advanced Infrastructure Manager (AIM)
Elastra	http://www.elastra.com/	Federated hybrid cloud management software
Ganglia	http://ganglia.info/	Distributed network monitoring software
Gomez	http://www.gomez.com/	Web site monitoring and analytics
HP Cloud Computing	http://h20338.www2.hp.com/enterprise/w1/en/technologies/cloud-computing-overview.html	A variety of management products and services, both released and under development
Hyperic	http://www.hyperic.com/	Performance management for virtualized Java Apps with VMware integration
IBM Service Management and Cloud Computing	http://www-01.ibm.com/software/tivoli/solutions/cloudcomputing/	Various IBM Tivoli managers and monitors
Internetseer	http://www.internetseer.com/home/index.xtp	Web site monitoring service
Intune	http://www.microsoft.com/windows/windowsintune/default.aspx	Cloud-based Windows system management
Keynote	http://www.keynote.com/	Web, mobile, streaming, and customer test and measurement products

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Product	URL	Description
ManageEngine OpManager	http://www.manageengine.com/network-performance-management.html	Network and server monitoring, server desk, event and security management
ManageIQ	http://www.manageiq.com/	Enterprise Virtualization Management Suite (EVM) that provides monitoring, provisioning, and cloud integration services
Managed Methods JaxView	http://managedmethods.com/	SOA management tool
Monit	http://mmonit.com/monit/	Unix system monitoring and management
Montis	http://portal.monitis.com/index.php/home	Cloud-based monitoring service
Morph	http://mor.ph/	Infrastructure management, provisioning, deployment, and monitoring tools
Nagios	http://www.nagios.org/	Network monitoring system
NetIQ	http://www.netiq.com/	Network management, monitoring, deployment, and security software
New Relic RPM	http://www.newrelic.com/	Java and Ruby application monitor and troubleshooting
Nimsoft	http://www.ca.com/us/products/detail/CA-Nimsoft-Monitoring-Solution.aspx	Cloud monitoring software
OpenQRM	http://www.openqrm.com/	Data center management platform
Pareto Networks	http://www.paretonetworks.com/	Cloud provisioning and deployment
Pingdom	http://www.pingdom.com/	Web site and server uptime and performing monitoring
RightScale	http://www.rightscale.com/	Automated virtual server scaling
ScienceLogic	http://www.sciencelogic.com/	Datacenter and cloud management solutions and appliances
Scout	http://scoutapp.com/	Hosted server management service
ServiceUptime	http://www.serviceuptime.com/	Web site monitoring service
Site24X7	http://site24x7.com/	Web site monitoring service
Solarwinds	http://www.solarwinds.com/	Network monitoring and management software

continued

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

TABLE 11.1 *(continued)*

Product	URL	Description
Tapinsystems	`http://www.tapinsystems.com/home`	Provisioning and management service
Univa UD	`http://univaud.com/index.php`	Application and infrastructure management software for hybrid multi-clouds
VMware Hyperic	`http://www.springsource.com/`	Performance management for VMware deployed Java applications
Webmetrics	`http://www.webmetrics.com/`	Web performance management, load testing, and application monitor for cloud services
WebSitePulse	`http://www.websitepulse.com/`	Server, Web site, and application monitoring service
Whatsup Gold	`http://www.whatsupgold.com/`	Network monitoring and management software
Zenoss	`http://www.zenoss.com/`	IT operations monitoring
Zeus	`http://www.zeus.com/`	Web-based application traffic manager

The core management features offered by most cloud management service products include the following:

- Support of different cloud types
- Creation and provisioning of different types of cloud resources, such as machine instances, storage, or staged applications
- Performance reporting including availability and uptime, response time, resource quota usage, and other characteristics
- The creation of dashboards that can be customized for a particular client's needs

Automated deployment on IaaS systems represents one class of cloud management services. One of the more interesting and successful vendors in this area is Rightscale (`http://www.rightscale.com/`) whose software allows clients to stage and manage applications on AWS (Amazon Web Service), Eucalyptus, Rackspace, and the Chef Multicloud framework or a combination of these cloud types. Rightscale creates cloud-ready server templates and provides the automation and orchestration necessary to deploy them. Eucalyptus and Rackspace both use Amazon EC2 and S3 services, although Eucalyptus is open source and portable. RightScale server templates and the Rightscript technology are highly configurable and can be run under batch control. The RightScale user interface also provides real-time measurements of individual server instances.

Cloudkick (`https://www.cloudkick.com/`) is another infrastructure monitoring solution that is well regarded. Its service is noted for being agnostic and working with multiple vendor cloud platforms. The Cloudkick user interface is designed for rapid deployment assessment, and its at-a-glance-monitoring Insight module is particularly easy to use. Figure 11.4 shows the Insight

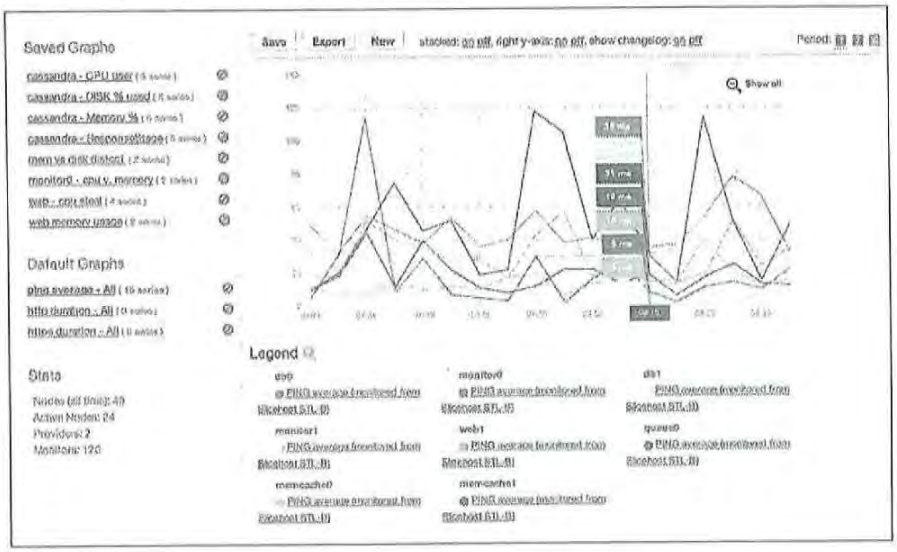
Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

module, and Figure 11.5 shows Cloudkick's real-time server visualization tool, which is one of the more interesting presentation tools we've seen. In Figure 11.5, the circles are servers with their location on the different axes based on observed metrics. Powerful servers are larger circles, and the colors indicate the current state of the server.

Users have commented on Cloudkick's instant launching being difficult, and both Cloudkick and RightScale are known to be easy to use with Linux virtual servers and less so with Windows instances.

FIGURE 11.4

Cloudkick's Insight module (`https://www.cloudkick.com/site_media/images/graphs2.png`) is powerful and particularly easy to use.



All of the service models support monitoring solutions, most often through interaction with the service API. Tapping into a service API allows management software to perform command actions that a user would normally perform. Some of these APIs are themselves scriptable, while in some cases, scripting is supported in the management software.

One key differentiator in monitoring and management software is whether the service needs to install an agent or it performs its service without an agent. The monitoring function normally can be performed through direct interaction with a cloud service or client using processes such as an HTTP GET or a network command like PING. For management functions, an agent is helpful in that it can provide needed hooks to manipulate a cloud resource. Agents also, as a general rule, are useful in helping to solve problems associated with firewall NAT traversal.

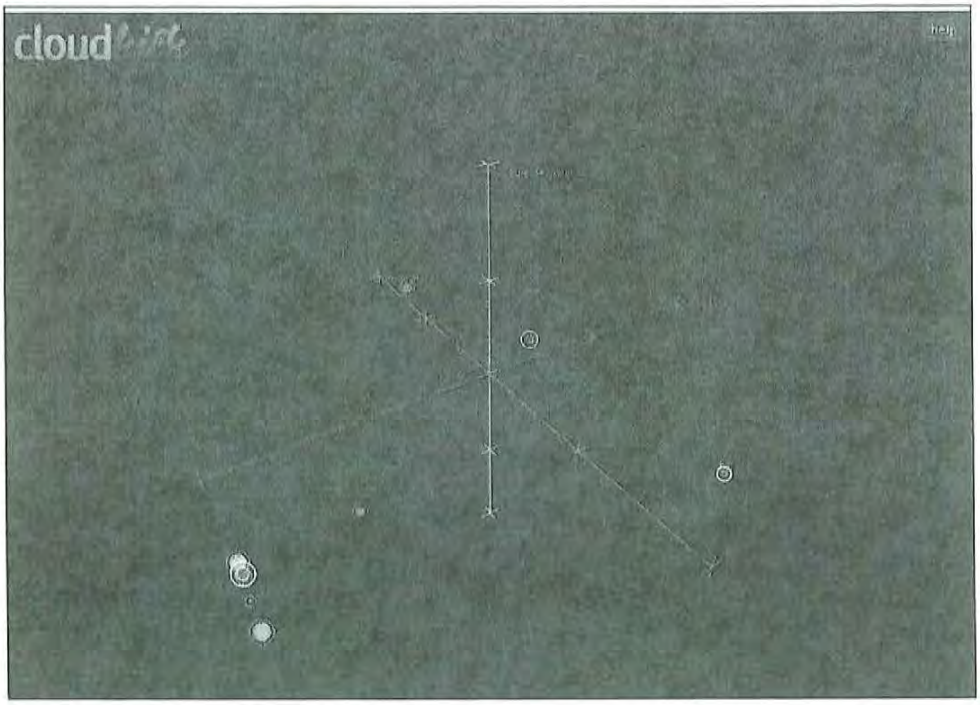
ManageIQ (`http://www.manageiq.com/`) and Service-now.com offer an integrated cloud stack that combines the ManageIQ Enterprise Virtualization Management Suite with Service-Now.com's ITSM SaaS service. The system has offers management, discovery, CMDB synchronization,

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

and automated provisioning services. You can integrate these services into your Web applications using an open API that these companies offer.

FIGURE 11.5

The Cloudkick visualization demo (`https://www.cloudkick.com/viz/demo/`) provides a real-time graphical illustration of the state of monitored servers.



Distributed network applications often benefit from the deployment of a management appliance. Because cloud services tend to distribute applications across multiple sites, physical appliances need to be deployed in different locations—something that only cloud service providers can do. However, there has been a tendency to create virtual appliances, and those can be deployed as server instances wherever an application is deployed. Pareto Networks (`http://www.paretonetworks.com/`) has a cloud computing service that can monitor and manage distributed network services using a physical or virtual appliance. The system can be used to control and provision network services. Pareto Networks plans to add an API to this service.

Emerging Cloud Management Standards

As it stands now, different cloud service providers use different technologies for creating and managing cloud resources. As the area matures, cloud providers are going to be under considerable

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

pressure from large cloud users like the federal government to conform to standards and make their systems interoperable with one another. No entity is likely to want to make a major investment in a service that is a silo or from which data is difficult to stage or to extract. To this end, a number of large industry players such as VMware, IBM, Microsoft, Citrix, and HP have gotten together to create standards that can be used to promote cloud interoperability. In the section that follows, you learn about the work of the DMTF in this area.

Another effort just getting underway has been started by CA (the company formerly known as Computer Associates) in association with Carnegie Mellon called the Cloud Commons. This effort is aimed at creating an industry community and working group, and promoting a set of monitoring standards that were part of CA's cloud technology portfolio but are now open sourced.

DMTF cloud management standards

The Distributed Management Task Force (DMTF; see `http://www.dmtf.org/`) is an industry organization that develops industry system management standards for platform interoperability. Its membership is a "who's who" in computing, and since its founding in 1992, the group has been responsible for several industry standards, most notably the Common Information Model (CIM). The DMTF organizes itself into a set of working groups that are tasked with specifying standards for different areas of technology.

A recent standard called the Virtualization Management Initiative (VMAN) was developed to extend CIM to virtual computer system management. VMAN has resulted in the creation of the Open Virtualization Format (OVF), which describes a standard method for creating, packaging, and provisioning virtual appliances. OVF is essentially a container and a file format that is open and both hypervisor- and processor-architecture-agnostic. Since OVF was announced in 2009, vendors such as VirtualBox, AbiCloud, IBM, Red Hat, and VMWare have announced or introduced products that use OVF.

It was, therefore, a natural extension of the work that DMTF does in virtualization to solve management issues in cloud computing. DMTF has created a working group called the Open Cloud Standards Incubator (OCSI) to help develop interoperability standards for managing interactions between and in public, private, and hybrid cloud systems. The group is focused on describing resource management and security protocols, packaging methods, and network management technologies. The Web site of the Cloud Management group (`http://dmtf.org/standards/cloud`) is shown in Figure 11.6.

DMTF's cloud management efforts are really in their initial stages, but the group has broad industry support. Part of the group's task is to provide industry education, so you can find a number of white papers and technology briefs published on this site. It's an effort that's worth checking back with over time. Although the OCSI's work has not yet been joined by Amazon or Salesforce.com, a set of open standards that extend the use of industry standard protocols—such as the Common Information Model (CIM), the Open Virtualization Format (OVF), and WBEM—to the cloud are going to be hard for vendors to resist.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 11.6

DMTF (`http://dmtf.org/standards/cloud`) has a large and important effort underway for developing cloud interoperability management standards.



Cloud Commons and SMI

CA Technologies (`http://www.ca.com`), the company once known as Computer Associates, has taken some of its technologies in measuring distributed network performance metrics and repositioned its products as the following:

- CA Cloud Insight, a cloud metrics measurement service
- CA Cloud Compose, a deployment service
- CA Cloud Optimize, a cloud optimization service
- CA Cloud Orchestrate, a workflow control and policy based automation service

Taken together, these products form the basis for CA's Cloud Connected Management Suite (`http://www.ca.com/us/cloud-solutions.aspx`).

CA has lots of experience in this area through its Unicenter management suite and the products that were spawned from it. The company also has invested in cloud vendors such as 3Tera,

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Oblicore, and Cassatt to create their cloud services. CA acquired Nimsoft in March 2010. Nimsoft has a monitoring and management package called Nimsoft United Monitoring that creates a monitoring portal with customizable dashboards. The system can gather information from up to 100 types of data points and can work with both Google and Rackspace cloud deployments. Among the data points that can be monitored are resource usage and UPS status.

At the heart of CA Cloud Insight is a method for measuring different cloud metrics that creates what CA calls a Service Measurement Index or SMI. The SMI measures things like SLA compliance, cost, and other values and rolls them up into a score. To help allow SMI to gain traction in the industry, CA has donated the core technology to the Software Engineering Institute at Carnegie Mellon as part of what is called the SMI Consortium. This same group is responsible for the Capability Maturity Model Integration (CMMI) process optimization technology and other efforts. The second CA initiative is the funding of an industry online community called the Cloud Commons (`http://www.cloudcommons.com/`), the home page of which is shown in Figure 11.7.

FIGURE 11.7

The Cloud Commons (`http://www.cloudcommons.com/web/guest`) is a new online community founded by CA to promote information exchange on cloud services and the SMI standard.

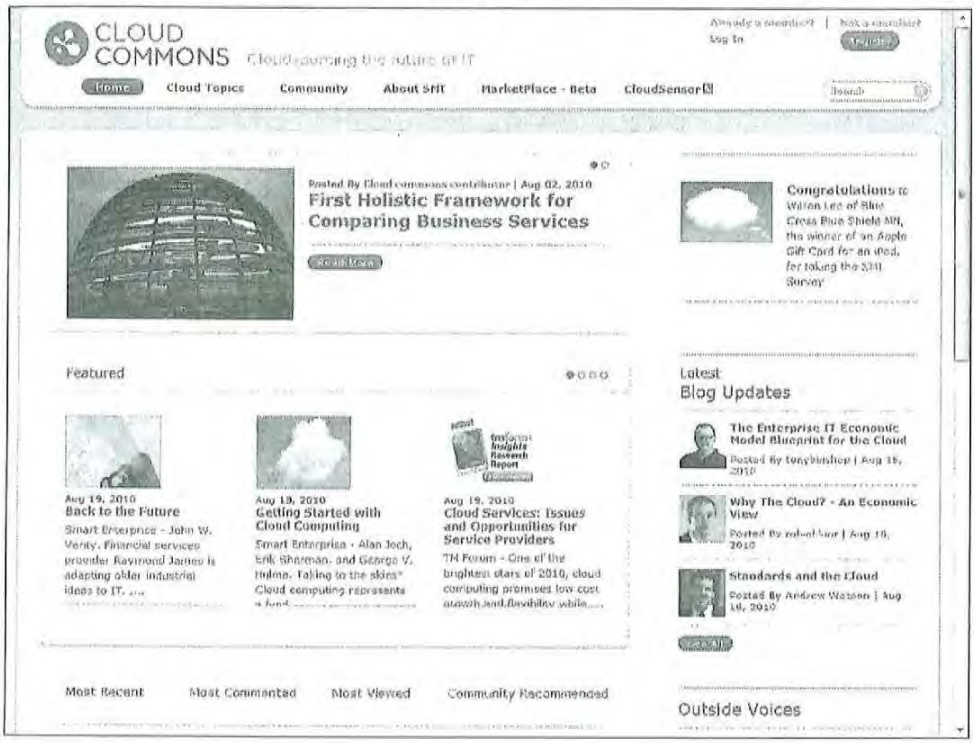


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Because the Cloud Commons is brand new, it is hard to tell whether this group will have impact in the cloud community, but it is an interesting effort. The hope is that not only will this site establish CA's performance metrics, but that community users will eventually provide detailed information and ratings on particular services.

To demonstrate the potential of cloud-based metrics, the Cloud Commons has built a dashboard called the CloudSensor that monitors the performance of the major cloud-based services in real time.

This tool measures the performance of the following:

- RackSpace file creation and deletion
- E-mail availability (system uptime) based on Google Gmail, Windows Live Hotmail, and Yahoo! Mail
- Amazon Web Services server creation/destruction times at four AWS sites
- Dashboard Response Times for the consoles of AWS.Amazon, Google App Status, RackSpace Cloud, and Saleforce
- Windows Azure storage benchmarks
- Windows Azure SQL benchmarks

It is meant to demonstrate the value of cloud performance measurements. These metrics are based on real-time data derived from real transactions. Each chart shows the last two hours of activity. Figure 11.8 shows the CloudSensor performance dashboard.

The Service Measurement Index (SMI) is based on a set of measurement technologies forming the SMI Framework that CA donated to the SMI Consortium. It measures cloud-based services in six areas:

- Agility
- Capability
- Cost
- Quality
- Risk
- Security

These form a set of Key Performance Indicators (KPI) that can be used to compare one service to another. Figure 11.9 shows the different characteristics that make up each of the KPIs of the Service Measurement Index.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 11.8

The CloudSensor (`http://dashboard.atgcloud.info:5001/cloudsensor/cloud-sensor.html`) dashboard displays real-time cloud service performance metrics.

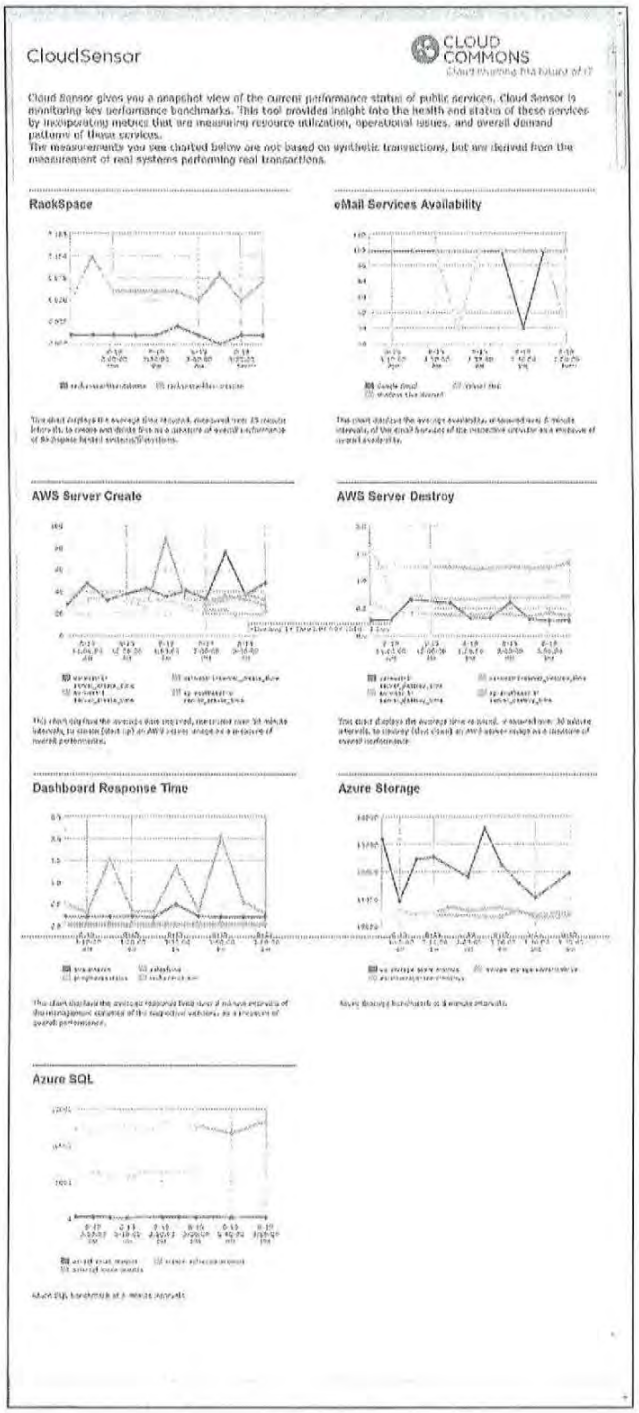
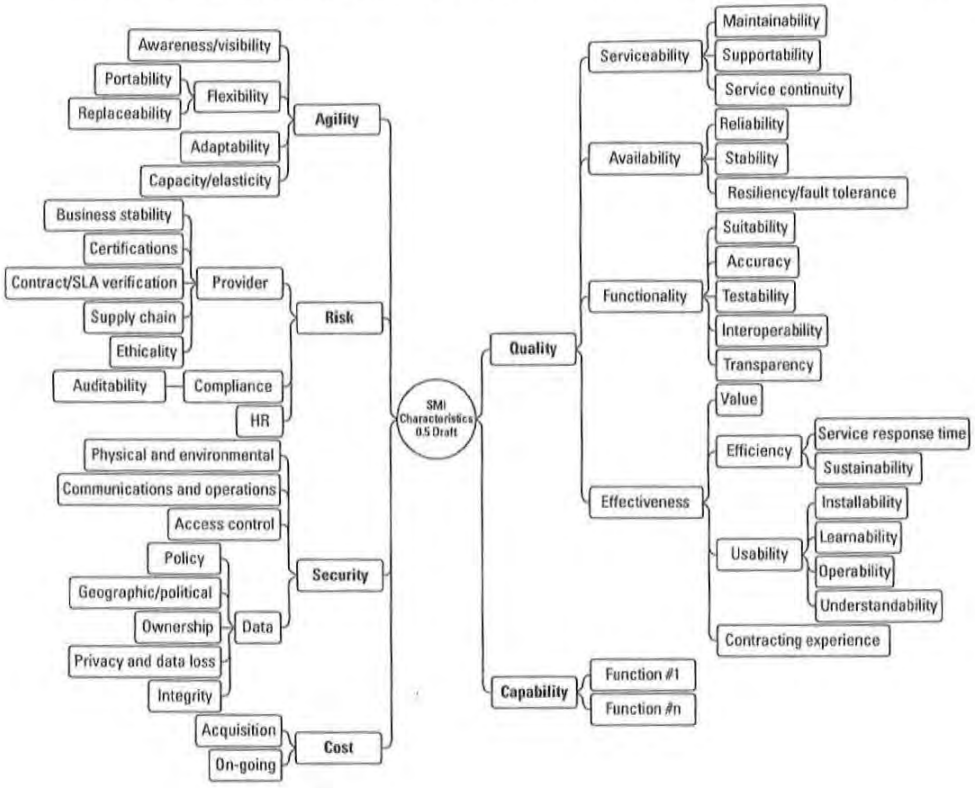


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 11.9

SMI defined characteristics (Source: "The Details behind the Service Measurement Index" by Keith Allen, 2010)



It's too early to determine whether SMI will gain traction, but the positioning of the technology as an open industry working group makes the project very interesting and worthy of note.

Summary

Cloud management is an important and growing area of technology. In this chapter, you learned about some of the products being offered or developed to address common management problems. Among the management tasks are deployment, monitoring, configuration, optimization, and often security. Nearly all network management software vendors are repositioning their products to work with cloud systems. Some network management is available from within the cloud service providers' platforms. Many of the software systems utilize the service provider's API to manage, monitor, and control resources. The use of virtualization has spawned many new products in this area.

In Chapter 12, you learn about security used in cloud computing. This is an extension of our discussion on network management, because security is another means of controlling access through network policies.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 12

Understanding Cloud Security

Cloud computing has lots of unique properties that make it very valuable. Unfortunately, many of those properties make security a singular concern. Many of the tools and techniques that you would use to protect your data, comply with regulations, and maintain the integrity of your systems are complicated by the fact that you are sharing your systems with others and many times outsourcing their operations as well. Cloud computing service providers are well aware of these concerns and have developed new technologies to address them.

Different types of cloud computing service models provide different levels of security services. You get the least amount of built in security with an Infrastructure as a Service provider, and the most with a Software as a Service provider. This chapter presents the concept of a security boundary separating the client's and vendor's responsibilities.

Adapting your on-premises systems to a cloud model requires that you determine what security mechanisms are required and mapping those to controls that exist in your chosen cloud service provider. When you identify missing security elements in the cloud, you can use that mapping to work to close the gap.

Storing data in the cloud is of particular concern. Data should be transferred and stored in an encrypted format. You can use proxy and brokerage services to separate clients from direct access to shared cloud storage.

Logging, auditing, and regulatory compliance are all features that require planning in cloud computing systems. They are among the services that need to be negotiated in Service Level Agreements.

Also in this chapter, you learn about identity and related protocols from a security standpoint. The concept of presence as it relates to identity is also introduced.

IN THIS CHAPTER

Reviewing cloud security concerns

Understanding how cloud data can be secured

Planning for security in your system

Learning how identity is used to allow secure cloud access

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Securing the Cloud

The Internet was designed primarily to be resilient; it was not designed to be secure. Any distributed application has a much greater attack surface than an application that is closely held on a Local Area Network. Cloud computing has all the vulnerabilities associated with Internet applications, and additional vulnerabilities arise from pooled, virtualized, and outsourced resources.

In the report "Assessing the Security Risks of Cloud Computing," Jay Heiser and Mark Nicolett of the Gartner Group (`http://www.gartner.com/DisplayDocument?id=685308`) highlighted the following areas of cloud computing that they felt were uniquely troublesome:

- Auditing
- Data integrity
- e-Discovery for legal compliance
- Privacy
- Recovery
- Regulatory compliance

Your risks in any cloud deployment are dependent upon the particular cloud service model chosen and the type of cloud on which you deploy your applications. In order to evaluate your risks, you need to perform the following analysis:

1. Determine which resources (data, services, or applications) you are planning to move to the cloud.
2. Determine the sensitivity of the resource to risk.

   Risks that need to be evaluated are loss of privacy, unauthorized access by others, loss of data, and interruptions in availability.
3. Determine the risk associated with the particular cloud type for a resource.

   Cloud types include public, private (both external and internal), hybrid, and shared community types. With each type, you need to consider where data and functionality will be maintained.
4. Take into account the particular cloud service model that you will be using.

   Different models such as IaaS, SaaS, and PaaS require their customers to be responsible for security at different levels of the service stack.
5. If you have selected a particular cloud service provider, you need to evaluate its system to understand how data is transferred, where it is stored, and how to move data both in and out of the cloud.

   You may want to consider building a flowchart that shows the overall mechanism of the system you are intending to use or are currently using.

One technique for maintaining security is to have "golden" system image references that you can return to when needed. The ability to take a system image off-line and analyze the image for

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

vulnerabilities or compromise is invaluable. The compromised image is a primary forensics tool. Many cloud providers offer a snapshot feature that can create a copy of the client's entire environment; this includes not only machine images, but applications and data, network interfaces, firewalls, and switch access. If you feel that a system has been compromised, you can replace that image with a known good version and contain the problem.

Many vendors maintain a security page where they list their various resources, certifications, and credentials. One of the more developed offerings is the AWS Security Center, shown in Figure 12.1, where you can download some backgrounders, white papers, and case studies related to the Amazon Web Service's security controls and mechanisms.

FIGURE 12.1

The AWS Security Center (`http://aws.amazon.com/security/`) is a good place to start learning about how Amazon Web Services protects users of its IaaS service.



The security boundary

In order to concisely discuss security in cloud computing, you need to define the particular model of cloud computing that applies. This nomenclature provides a framework for understanding what security is already built into the system, who has responsibility for a particular security mechanism, and where the boundary between the responsibility of the service provider is separate from the responsibility of the customer.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

All of Chapter 1 was concerned with defining what cloud computing is and defining the lexicon of cloud computing. There are many definitions and acronyms in the area of cloud computing that will probably not survive long. The most commonly used model based on U.S. National Institute of Standards and Technology (NIST; `http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html`) separates deployment models from service models and assigns those models a set of service attributes. Deployment models are cloud types: community, hybrid, private, and public clouds. Service models follow the SPI Model for three forms of service delivery: Software, Platform, and Infrastructure as a Service. In the NIST model, as you may recall, it was not required that a cloud use virtualization to pool resources, nor did that model require that a cloud support multi-tenancy. It is just these factors that make security such a complicated proposition in cloud computing.

Chapter 1 also presented the Cloud Security Alliance (CSA; `http://www.cloudsecurityalliance.org/`) cloud computing stack model, which shows how different functional units in a network stack relate to one another. As you may recall from Chapter 1, this model can be used to separate the different service models from one another. CSA is an industry working group that studies security issues in cloud computing and offers recommendations to its members. The work of the group is open and available, and you can download its guidance from its home page, shown in Figure 12.2.

FIGURE 12.2

The Cloud Security Alliance (CSA) home page at `http://www.cloudsecurityalliance.org/` offers a number of resources to anyone concerned with securing his cloud deployment.

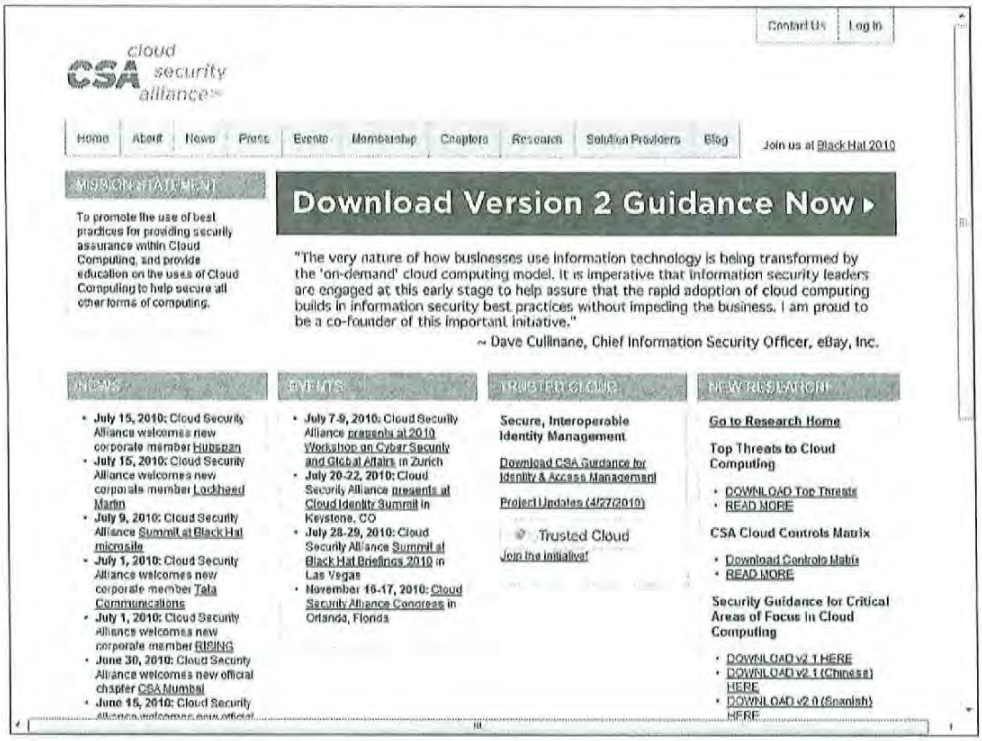


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The CSA partitions its guidance into a set of operational domains:

- Governance and enterprise risk management
- Legal and electronic discovery
- Compliance and audit
- Information lifecycle management
- Portability and interoperability
- Traditional security, business continuity, and disaster recovery
- Datacenter operations
- Incidence response, notification, and remediation
- Application security
- Encryption and key management
- Identity and access management
- Virtualization

You can download the group's current work in these areas from the different sections of its Web site.

One key difference between the NIST model and the CSA is that the CSA considers multi-tenancy to be an essential element in cloud computing. Multi-tenancy adds a number of additional security concerns to cloud computing that need to be accounted for. In multi-tenancy, different customers must be isolated, their data segmented, and their service accounted for. To provide these features, the cloud service provider must provide a policy-based environment that is capable of supporting different levels and quality of service, usually using different pricing models. Multi-tenancy expresses itself in different ways in the different cloud deployment models and imposes security concerns in different places.

Security service boundary

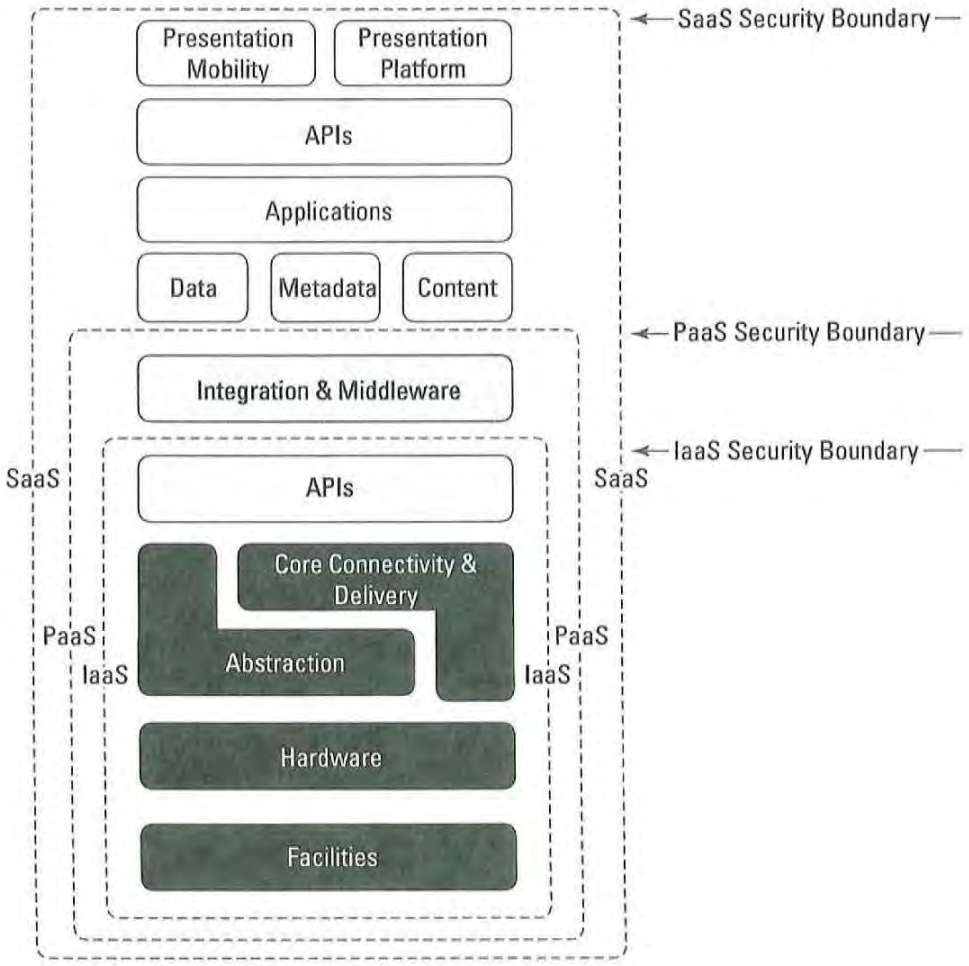
The CSA functional cloud computing hardware/software stack is the Cloud Reference Model. This model, which was discussed in Chapter 1, is reproduced in Figure 12.3. IaaS is the lowest level service, with PaaS and SaaS the next two services above. As you move upward in the stack, each service model inherits the capabilities of the model beneath it, as well as all the inherent security concerns and risk factors. IaaS supplies the infrastructure; PaaS adds application development frameworks, transactions, and control structures; and SaaS is an operating environment with applications, management, and the user interface. As you ascend the stack, IaaS has the least levels of integrated functionality and the lowest levels of integrated security, and SaaS has the most.

The most important lesson from this discussion of architecture is that each different type of cloud service delivery model creates a security boundary at which the cloud service provider's responsibilities end and the customer's responsibilities begin. Any security mechanism below the security boundary must be built into the system, and any security mechanism above must be maintained by the customer. As you move up the stack, it becomes more important to make sure that the type and level of security is part of your Service Level Agreement.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 12.3

The CSA Cloud Reference Model with security boundaries shown



In the SaaS model, the vendor provides security as part of the Service Level Agreement, with the compliance, governance, and liability levels stipulated under the contract for the entire stack. For the PaaS model, the security boundary may be defined for the vendor to include the software framework and middleware layer. In the PaaS model, the customer would be responsible for the security of the application and UI at the top of the stack. The model with the least built-in security is IaaS, where everything that involves software of any kind is the customer's problem. Numerous definitions of services tend to muddy this picture by adding or removing elements of the various functions from any particular offering, thus blurring which party has responsibility for which features, but the overall analysis is still useful.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

In thinking about the Cloud Security Reference Model in relationship to security needs, a fundamental distinction may be made between the nature of how services are provided versus where those services are located. A private cloud may be internal or external to an organization, and although a public cloud is most often external, there is no requirement that this mapping be made so. Cloud computing has a tendency to blur the location of the defined security perimeter in such a way that the previous notions of network firewalls and edge defenses often no longer apply.

This makes the location of trust boundaries in cloud computing rather ill defined, dynamic, and subject to change depending upon a number of factors. Establishing trust boundaries and creating a new perimeter defense that is consistent with your cloud computing network is an important consideration. The key to understanding where to place security mechanisms is to understand where physically in the cloud resources are deployed and consumed, what those resources are, who manages the resources, and what mechanisms are used to control them. Those factors help you gauge where systems are located and what areas of compliance you need to build into your system.

Table 12.1 lists some of the different service models and lists the parties responsible for security in the different instances.

TABLE 12.1

Security Responsibilities by Service Model

Model Type	Infrastructure Security Management	Infrastructure Owner	Infrastructure Location	Trust Condition
Hybrid	Both vendor and customer	Both vendor and customer	Both on- and off-premises	Both trusted and untrusted
Private/Community	Customer	Customer	On- or off-premises	Trusted
Private/Community	Customer	Vendor	Off- or on-premises	Trusted
Private/Community	Vendor	Customer	On- or off-premises	Trusted
Private/Community	Vendor	Vendor	Off- or on-premises	Trusted
Public	Vendor	Vendor	Off-premises	Untrusted

Security mapping

The cloud service model you choose determines where in the proposed deployment the variety of security features, compliance auditing, and other requirements must be placed. To determine the particular security mechanisms you need, you must perform a mapping of the particular cloud service model to the particular application you are deploying. These mechanisms must be supported by the various controls that are provided by your service provider, your organization, or a third party. It's unlikely that you will be able to duplicate security routines that are possible on-premises, but this analysis allows you to determine what coverage you need.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A security control model includes the security that you normally use for your applications, data, management, network, and physical hardware. You may also need to account for any compliance standards that are required for your industry. A compliance standard can be any government regulatory framework such as Payment Card Industry Data Security Standards (PCI-DSS), Health Insurance Portability and Accountability Act (HIPPA), Gramm–Leach–Bliley Act (GLBA), or the Sarbanes–Oxley Act (SOX) that requires you operate in a certain way and keep records.

Essentially, you are looking to identify the missing features that would be required for an on-premises deployment and seek to find their replacements in the cloud computing model. As you assign accountability for different aspects of security and contract away the operational responsibility to others, you want to make sure they remain accountable for the security you need.

Securing Data

Securing data sent to, received from, and stored in the cloud is the single largest security concern that most organizations should have with cloud computing. As with any WAN traffic, you must assume that any data can be intercepted and modified. That's why, as a matter of course, traffic to a cloud service provider and stored off-premises is encrypted. This is as true for general data as it is for any passwords or account IDs.

These are the key mechanisms for protecting data mechanisms:

- Access control
- Auditing
- Authentication
- Authorization

Whatever service model you choose should have mechanisms operating in all four areas that meet your security requirements, whether they are operating through the cloud service provider or your own local infrastructure.

Brokered cloud storage access

The problem with the data you store in the cloud is that it can be located anywhere in the cloud service provider's system: in another datacenter, another state or province, and in many cases even in another country. With other types of system architectures, such as client/server, you could count on a firewall to serve as your network's security perimeter; cloud computing has no physical system that serves this purpose. Therefore, to protect your cloud storage assets, you want to find a way to isolate data from direct client access.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

One approach to isolating storage in the cloud from direct client access is to create layered access to the data. In one scheme, two services are created: a broker with full access to storage but no access to the client, and a proxy with no access to storage but access to both the client and broker. The location of the proxy and the broker is not important (they can be local or in the cloud); what is important is that these two services are in the direct data path between the client and data stored in the cloud.

Under this system, when a client makes a request for data, here's what happens:

1. The request goes to the external service interface (or endpoint) of the proxy, which has only a partial trust.
2. The proxy, using its internal interface, forwards the request to the broker.
3. The broker requests the data from the cloud storage system.
4. The storage system returns the results to the broker.
5. The broker returns the results to the proxy.
6. The proxy completes the response by sending the data requested to the client.

Figure 12.4 shows this storage "proxy" system graphically.

Note

This discussion is based on a white paper called "Security Best Practices For Developing Windows Azure Applications," by Andrew Marshall, Michael Howard, Grant Bugher, and Brian Harden that you can find at `http://download.microsoft.com/download/7/3/E/73E4EE93-559F-4D0F-A6FC-7FEC5F1542D1/SecurityBestPracticesWindowsAzureApps.docx`**. In their presentation, the proxy service is called the Gatekeeper and assigned a Windows Server Web Role, and the broker is called the KeyMaster and assigned a Worker Role.** ■

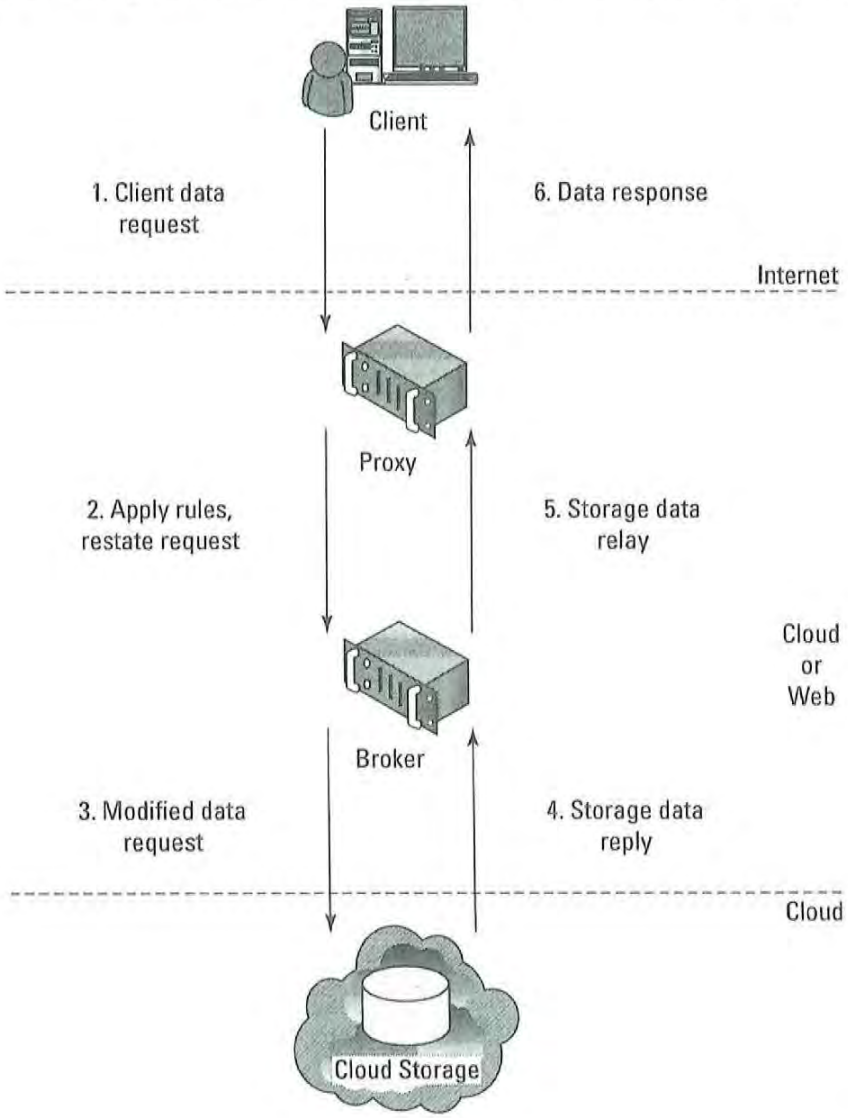
This design relies on the proxy service to impose some rules that allow it to safely request data that is appropriate to that particular client based on the client's identity and relay that request to the broker. The broker does not need full access to the cloud storage, but it may be configured to grant `READ` and `QUERY` operations, while not allowing `APPEND` or `DELETE`. The proxy has a limited trust role, while the broker can run with higher privileges or even as native code.

The use of multiple encryption keys can further separate the proxy service from the storage account. If you use two separate keys to create two different data zones—one for the untrusted communication between the proxy and broker services, and another a trusted zone between the broker and the cloud storage—you create a situation where there is further separation between the different service roles.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 12.4

In this design, direct access to cloud storage is eliminated in favor of a proxy/broker service.

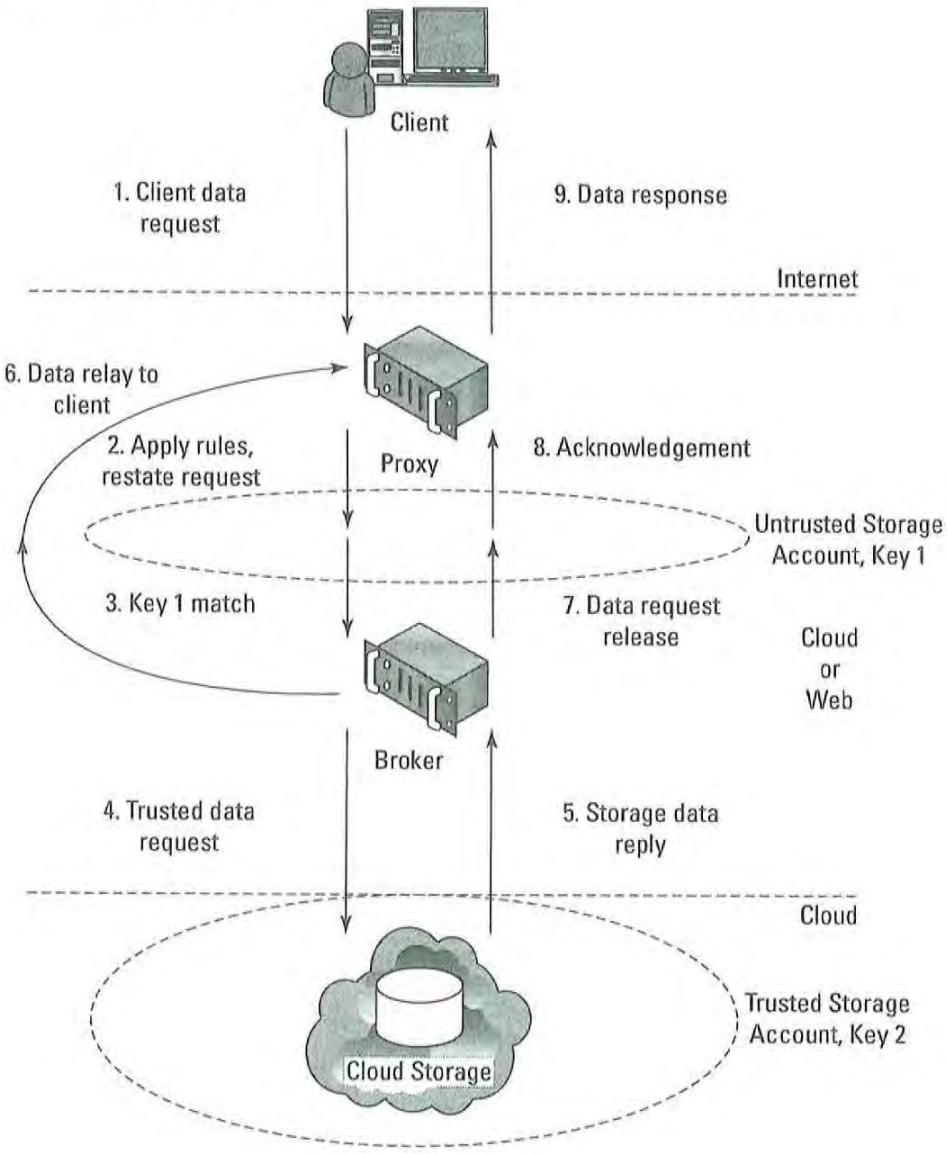


Even if the proxy service is compromised, that service does not have access to the trusted key necessary to access the cloud storage account. In the multi-key solution, shown in Figure 12.5, you have not only eliminated all internal service endpoints, but you also have eliminated the need to have the proxy service run at a reduced trust level.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 12.5

The creation of storage zones with associated encryption keys can further protect cloud storage from unauthorized access.



Storage location and tenancy

Some cloud service providers negotiate as part of their Service Level Agreements to contractually store and process data in locations that are predetermined by their contract. Not all do. If you can

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

get the commitment for specific data site storage, then you also should make sure the cloud vendor is under contract to conform to local privacy laws.

Because data stored in the cloud is usually stored from multiple tenants, each vendor has its own unique method for segregating one customer's data from another. It's important to have some understanding of how your specific service provider maintains data segregation.

Another question to ask a cloud storage provider is who is provided privileged access to storage. The more you know about how the vendor hires its IT staff and the security mechanism put into place to protect storage, the better.

Most cloud service providers store data in an encrypted form. While encryption is important and effective, it does present its own set of problems. When there is a problem with encrypted data, the result is that the data may not be recoverable. It is worth considering what type of encryption the cloud provider uses and to check that the system has been planned and tested by security experts.

Regardless of where your data is located, you should know what impact a disaster or interruption will have on your service and your data. Any cloud provider that doesn't offer the ability to replicate data and application infrastructure across multiple sites cannot recover your information in a timely manner. You should know how disaster recovery affects your data and how long it takes to do a complete restoration.

Encryption

Strong encryption technology is a core technology for protecting data in transit to and from the cloud as well as data stored in the cloud. It is or will be required by law. The goal of encrypted cloud storage is to create a virtual private storage system that maintains confidentiality and data integrity while maintaining the benefits of cloud storage: ubiquitous, reliable, shared data storage. Encryption should separate stored data (data at rest) from data in transit.

Depending upon the particular cloud provider, you can create multiple accounts with different keys as you saw in the example with Windows Azure Platform in the previous section. Microsoft allows up to five security accounts per client, and you can use these different accounts to create different zones. On Amazon Web Service, you can create multiple keys and rotate those keys during different sessions.

Although encryption protects your data from unauthorized access, it does nothing to prevent data loss. Indeed, a common means for losing encrypted data is to lose the keys that provide access to the data. Therefore, you need to approach key management seriously. Keys should have a defined lifecycle. Among the schemes used to protect keys are the creation of secure key stores that have restricted role-based access, automated key stores backup, and recovery techniques. It's a good idea to separate key management from the cloud provider that hosts your data.

One standard for interoperable cloud-based key management is the OASIS Key Management Interoperability Protocol (KMIP; `http://www.oasis-open.org/committees/kmip/`). IEEE 1619.3 (`https://siswg.net/index.php?option=com_docman`) also covers both storage encryption and key management for shared storage.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Auditing and compliance

Logging is the recording of events into a repository; auditing is the ability to monitor the events to understand performance. Logging and auditing is an important function because it is not only necessary for evaluation performance, but it is also used to investigate security and when illegal activity has been perpetrated. Logs should record system, application, and security events, at the very minimum.

Logging and auditing are unfortunately one of the weaker aspects of early cloud computing service offerings.

Cloud service providers often have proprietary log formats that you need to be aware of. Whatever monitoring and analysis tools you use need to be aware of these logs and able to work with them. Often, providers offer monitoring tools of their own, many in the form of a dashboard with the potential to customize the information you see through either the interface or programmatically using the vendor's API. You want to make full use of those built-in services.

Because cloud services are both multitenant and multisite operations, the logging activity and data for different clients may not only be co-located, they may also be moving across a landscape of different hosts and sites. You can't simply expect that an investigation will be provided with the necessary information at the time of discovery unless it is part of your Service Level Agreement. Even an SLA with the appropriate obligations contained in it may not be enough to guarantee you will get the information you need when the time comes. It is wise to determine whether the cloud service provider has been able to successfully support investigations in the past.

As it stands now, nearly all regulations were written without keeping cloud computing in mind. A regulator or auditor isn't likely to be familiar with the nature of running applications and storing data in the cloud. Even so, laws are written to ensure compliance, and the client is held responsible for compliance under the laws of the governing bodies that apply to the location where the processing or storage takes place.

Therefore, you must understand the following:

- Which regulations apply to your use of a particular cloud computing service
- Which regulations apply to the cloud service provider and where the demarcation line falls for responsibilities
- How your cloud service provider will support your need for information associated with regulation
- How to work with the regulator to provide the information necessary regardless of who had the responsibility to collect the data

Traditional service providers are much more likely to be the subject of security certifications and external audits of their facilities and procedures than cloud service providers. That makes the willingness for a cloud service provider to subject its service to regulatory compliance scrutiny an important factor in your selection of that provider over another. In the case of a cloud service provider who shows reluctance to or limits the scrutiny of its operations, it is probably wise to use the

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

service in ways that limit your exposure to risk. For example, although encrypting stored data is always a good policy, you also might want to consider not storing any sensitive information on that provider's system.

As it stands now, clients must guarantee their own regulatory compliance, even when their data is in the care of the service provider. You must ensure that your data is secure and that its integrity has not been compromised. When multiple regulatory entities are involved, as there surely are between site locations and different countries, then that burden to satisfy the laws of those governments is also your responsibility.

For any company with clients in multiple countries, the burden of regulatory compliance is onerous. While organizations such as the EEC (European Economic Community) or Common Market provide some relief for European regulation, countries such as the United States, Japan, China, and others each have their own sets of requirements. This makes regulatory compliance one of the most actively developing and important areas of cloud computing technology.

This situation is likely to change. On March 1, 2010, Massachusetts passed a law that requires companies that provide sensitive personal information on Massachusetts residents to encrypt data transmitted and stored on their systems. Businesses are required to limit the amount of personal data collected, monitor data usage, keep a data inventory, and be able to present a security plan on how they will keep the data safe. The steps require that companies verify that any third-party services they use conform to these requirements and that there be language in all SLAs that enforce these protections. The law takes full effect in March 2012.

Going forward, you want to ensure the following:

- You have contracts reviewed by your legal staff.
- You have a right-to-audit clause in your SLA.
- You review any third parties who are service providers and assess their impact on security and regulatory compliance.
- You understand the scope of the regulations that apply to your cloud computing applications and services.
- You consider what steps you must take to comply with the demands of regulations that apply.
- You consider adjusting your procedures to comply with regulations.
- You collect and maintain the evidence of your compliance with regulations.
- You determine whether your cloud service provider can provide an audit statement that is SAS 70 Type II-compliant.

The ISO/IEC 27001/27002 standard for information security management systems has a roadmap for mission-critical services that you may want to discuss with your cloud service provider. Amazon Web Services supports SAS70 Type II Audits.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Becoming a cloud service provider requires a large investment, but as we all know, even large companies can fail. When a cloud service provider fails, it may close or more likely be acquired by another company. You likely wouldn't use a service provider that you suspected of being in difficulty, but problems develop over years and cloud computing has a certain degree of vendor lock-in to it. That is, when you have created a cloud-based service, it can be difficult or often impossible to move it to another service provider. You should be aware of what happens to your data if the cloud service provider fails. At the very least, you would want to make sure your data could be obtained in a format that could be accessed by on-premise applications.

The various attributes of cloud computing make it difficult to respond to incidents, but that doesn't mean you should consider drawing up security incidence response policies. Although cloud computing creates shared responsibilities, it is often up to the client to initiate the inquiry that gets the ball rolling. You should be prepared to provide clear information to your cloud service provider about what you consider to be an incident or a breach in security and what are simply suspicious events.

Establishing Identity and Presence

Chapter 4 introduced the concept of identities, some of the protocols that support them, and some of the services that can work with them. Identities also are tied to the concept of accounts and can be used for contacts or "ID cards." Identities also are important from a security standpoint because they can be used to authenticate client requests for services in a distributed network system such as the Internet or, in this case, for cloud computing services.

Identity management is a primary mechanism for controlling access to data in the cloud, preventing unauthorized uses, maintaining user roles, and complying with regulations. The sections that follow describe some of the different security aspects of identity and the related concept of "presence." For this conversation, you can consider presence to be the mapping of an authenticated identity to a known location. Presence is important in cloud computing because it adds context that can modify services and service delivery.

Cloud computing requires the following:

- That you establish an identity
- That the identity be authenticated
- That the authentication be portable
- That authentication provide access to cloud resources

When applied to a number of users in a cloud computing system, these requirements describe systems that must provision identities, provide mechanisms that manage credentials and authentication, allow identities to be federated, and support a variety of user profiles and access policies. Automating these processes can be a major management task, just as they are for on-premises operations.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Identity protocol standards

The protocols that provide identity services have been and are under active development, and several form the basis for efforts to create interoperability among services.

OpenID 2.0 (`http://openid.net/`) is the standard associated with creating an identity and having a third-party service authenticate the use of that digital identity. It is the key to creating Single Sign-On (SSO) systems. Some cloud service providers have adopted OpenID as a service, and its use is growing.

In Chapter 4, you learned how OpenID is associated with contact cards such as vCards and InfoCards. In that chapter, I briefly discussed how OpenID provides access to important Web sites and how some Web sites allow you to use your logins based on OpenID from another site to gain access to their site.

OpenID doesn't specify the means for authentication of an identity, and it is up to the particular system how the authentication process is executed. Authentication can be by a Challenge and Response Protocol (CHAP), through a physical smart card, or using a flying finger or evil eye through a biometric measurement. In OpenIDL, the authentication procedure has the following steps:

1. The end-user uses a program like a browser that is called a user agent to enter an OpenID identifier, which is in the form of a URL or XRI.

   An OpenID might take the form of *name*`.openid.provider.org`.
2. The OpenID is presented to a service that provides access to the resource that is desired.
3. An entity called a relaying party queries the OpenID identity provider to authenticate the veracity of the OpenID credentials.
4. The authentication is sent back to the relaying party from the identity provider and access is either provided or denied.

According to a report by one of OpenID's directors called "OpenID 2009 Year in Review" by Brian Kissel (`http://openid.net/2009/12/16/openid-2009-year-in-review/`), there were over 1 billion OpenID accounts accepted by 9 million sites on the Internet.

The second protocol used to present identity-based claims in cloud computing is a set of authorization markup languages that create files in the form of being XACML and SAML. These protocols were described in Chapter 4 in detail, so I only mention them in passing here. SAML (Security Assertion Markup Language; `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`) is gaining growing acceptance among cloud service providers. It is a standard of OASIS and an XML standard for passing authentication and authorization between an identity provider and the service provider. SAML is a complimentary mechanism to OpenID and is used to create SSO systems.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Taken as a unit, OpenID and SAML are being positioned to be the standard authentication mechanism for clients accessing cloud services. It is particularly important for services such as mashups that draw information from two or more data services.

An open standard called OAuth (`http://oauth.net/`) provides a token service that can be used to present validated access to resources. OAuth is similar to OpenID, but provides a different mechanism for shared access. The use of OAuth tokens allows clients to present credentials that contain no account information (userID or password) to a cloud service. The token comes with a defined period after which it can no longer be used. Several important cloud service providers have begun to make OAuth APIs available based on the OAuth 2.0 standard, most notably Facebook's Graph API and the Google Data API.

The DataPortability Project (`http://dataportability.org/`) is an industry working group that promotes data interoperability between applications, and the group's work touches on a number of the emerging standards mentioned in this section. The group's Web site is shown in Figure 12.6.

FIGURE 12.6

The home page of the DataPortability Project, an industry working group that promotes open identity standards

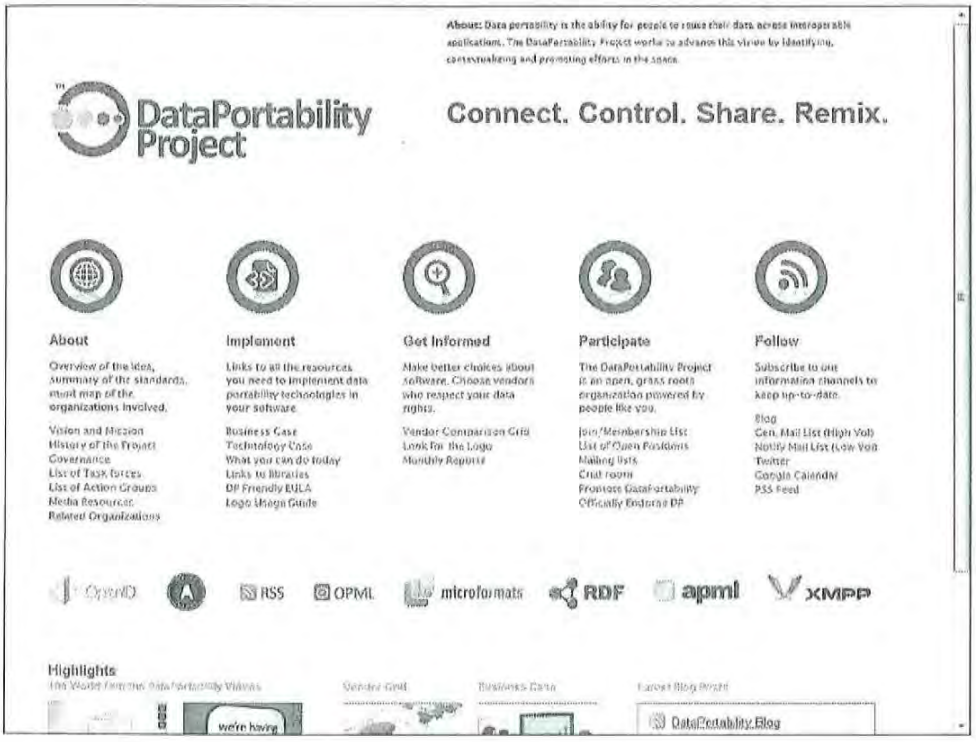


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A number of vendors have created server products, such as Identity and Access Managers (IAMs), to support these various standards.

Windows Azure identity standards

The Windows Azure Platform uses a claims-based identity based on open authentication and access protocols and is a good example of a service implementing the standards described in the previous section. These standards may be used without modification on a system that is running in the cloud or on-premises, in keeping with Microsoft's S+S (software plus services) approach to cloud computing.

Windows Azure security draws on the following three services:

- Active Directory Federation Services 2.0
- Windows Azure AppFabric Access Control Service
- Windows Identity Foundation (WIF)

The Windows Identity Foundation offers .NET developers Visual Studio integration of WS-Federation and WS-Trust open standards. ASP.NET Web applications created with WIF integrate the Windows Communication Foundation SOAP service (WCF-SOAP) into a unified object model. This allows WIF to have full access to the features of WS-Security and to work with tokens in the SAML format.

WIF relies on third-party authentication and accepts authentication requests from these services in the form of a set of claims. Claims are independent of where a user account or application is located, thus allowing claims to be used in single sign-on systems (SSO). Claims support both simple resource access and the Role Based Access Control (RBAC) policies that can be enforced by Windows group policies.

Active Directory Federation Services 2.0 (AD FS) is a Security Token Service (STS) that allows users to authenticate their access to applications both locally and in the cloud with a claims-based identity. Anyone who has an account in the local Windows directory can access an application; AD FS creates and retains trust relationships with federated systems. AD FS uses WS-Federation, WS-Trust, and SAML, which allows users to access a system based on IBM, Novell, SAP, and many other vendors.

The final piece of the Windows Azure Platform claims-based identity system is built directly into the AppFabric Access Control (AC) service. You may recall from Chapter 10 that AppFabric is a service bus for Azure components that supports REST Web services. Included in AppFabric are authentication and claims-based authorization access. These can be simple logons or more complex schemes supported by AD FS. AC allows authorization to be located anywhere and allows developers to separate identity from their application.

The claims-based identity in AC is based on the OAuth Web Resource Authorization Protocol (OAuth WRAP), which works with various REST APIs. The Oauth 2.0 protocol seems to be gaining acceptance in the cloud computing industry, because SAML tokens can be accepted by many vendors.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Presence

Presence is a fundamental concept in computer science. It is used on networks to indicate the status of available parties and their location. Commands like the WHO command in Linux that list users logged into the network go all the way back to the first network operating systems.

Presence provides not only identity, but status and, as part of status, location. The status is referred to as the presence state, the identity is the presentity, and the service that manages presence is called the presence service. Many presence services rely on agents called watchers, which are small programs that relay a client's ability to connect. Among the cloud computing services that rely on presence information are telephony systems such as VoIP, instant messaging services (IM), and geo-location-based systems such as GPS. Presence is playing an important role in cell phones, particularly smart phones.

When you access an application such as AroundMe on the Apple iPhone, which lists businesses, services, and restaurants in your vicinity, you are using an example of a presence service. Figure 12.7 shows the AroundMe app with some sample results. The presence service is provided by the GPS locator inside the phone, which provides a location through AT&T (the service provider) to the application. Presence is an essential and growing component of cloud-based services, and it adds a tremendous amount of value to the ubiquity that a cloud network offers.

FIGURE 12.7

The AroundMe iPhone app is an example of an application that makes use of a presence service.



Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

As cloud computing becomes more pervasive and vendors attempt to create federated systems, emerging presence services will become more important. Microsoft's Windows Identity Foundation (described in the previous section) created under the Geneva Framework project is one example of an attempt to create a claims-based presence system. WIF allows different systems to interoperate using a variety of authentication methods, including LDAP and Active Directory, OpenID, LiveID, Microsoft CardSpace, and Novell Digital Me.

The Internet Engineering Task Force (IETF) has developed a standard called the Extensible Messaging and Presence Protocol (XMPP) that can be used with a federation system called the Jabber Extensible Communications Platform (Jabber XCP) to provide presence information. Among the services that use Jabber XCP are the Defense Information Systems Agency (DISA), Google Talk, Earthlink, Facebook, the National Weather Service, Twitter, the U.S. Marine Corps, and the U.S. Joint Forces Command (USJFCOM).

Jabber XCP is popular because it is an extensible development platform that is platform-independent and supports several communications protocols, such as the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) and Instant Messaging and Presence Service (IMPS).

The notion of applying presence services over the standard Service Oriented Architecture (SOA) protocols such as SOAP/REST/HTTP is that in SOA all these protocols support unidirectional data exchange. You request a service/data, and a response is supplied. SOA architectures don't scale well and can't supply high-speed data transfers required by the collaboration services that are based on presence service technologies. SOA also has the problem of services that have trouble penetrating firewalls. It was these barriers that Jabber and XMPP were created to solve, and you will find that protocol incorporated into a number of cloud computing SaaS services. AOL, Apple, Google, IBM, and others are using this technology in some of their applications today.

Summary

In this chapter, you learned about many of the issues concerning security and cloud computing. This is a rapidly changing area of great importance to anyone considering deploying systems or storing data in the cloud. Security may, in fact, be the single most important area of cloud computing that you need to plan for. Depending upon the type of service you use, you may find many security services already built into your system.

An issue you need to consider is how to protect data in transit to and stored in the cloud. Data encryption, access restrictions, and data protection services were described. Multi-tenancy, system virtualization, and other factors make monitoring, regulatory compliance, and incidence response more challenging than on-premises systems are. A key concept for controlling access to cloud resources is identity management.

In Chapter 13, the standards used to create cloud systems are described. These standards create what is called the Service Oriented Architecture. You also learn how cloud systems can be built from many different interoperable modular parts.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Part IV

Understanding Services and Applications

IN THIS PART

Chapter 13
Understanding Service Oriented Architecture

Chapter 14
Moving Applications to the Cloud

Chapter 15
Working with Cloud-Based Storage

Chapter 16
Working with Productivity Software

Chapter 17
Using Webmail Services

Chapter 18
Communicating with the Cloud

Chapter 19
Using Media and Streaming

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 13

Understanding Service Oriented Architecture

Service Oriented Architecture (SOA) describes a standard method for requesting services from distributed components and managing the results. Because the clients requesting services, the components providing the services, the protocols used to deliver messages, and the responses can vary widely, SOA provides the translation and management layer in an architecture that removes the barrier for a client obtaining desired services. With SOA, clients and components can be written in different languages and can use multiple messaging protocols and networking protocols to communicate with one another. SOA provides the standards that transport the messages and makes the infrastructure to support it possible. SOA provides access to reusable Web services over a TCP/IP network, which makes this an important topic to cloud computing going forward.

You don't need SOA if you are creating a monolithic cloud application that performs a specific function such as backup, e-mail, Web page access, or instant messaging. Many of the large and familiar cloud computing applications are monolithic and were built with proprietary technologies—albeit often on top of open source software and hardware. However, as cloud computing applications expand their capability to provide additional and diverse services, SOA offers access to ready-made, modular, highly optimized, and widely shareable components that can minimize developer and infrastructure costs.

For over a decade now Service Oriented Architecture has been part of a collaborative effort on the part of both large and small vendors to come up with a common solution to architecting complex business software processes efficiently. As cloud computing matures and the applications offered become more capable, the key to being competitive and offering users the capability to customize their environments lays in the standardization that Service Oriented Architecture offers. The influence of SOA in cloud computing is therefore likely to grow.

IN THIS CHAPTER

- Using Service Oriented Architecture
- Defining message-based transactions
- Understanding processes and transactions
- Managing SOA distributed applications

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

This chapter provides the basis for understanding what SOA is, how SOA operates, what limitations and capabilities are part of the architecture, and the vocabulary that you need to know in this subject. SOA is an architecture, first and foremost, so in essence it is a blueprint for creating a system conforming to this standard. The environment it creates is a virtual message-passing system with a loose coupling between clients and services. The products that support SOA are a diverse lot. SOA components are meant to be modular and easily added to a business process, and this modularity makes them good candidates for Computer Aided Software Engineering (CASE) modeling tools. This chapter describes the software used to support SOA and some of the important ways in which the architecture is interpreted.

Cloud computing is not the next evolutionary step beyond SOA, as some think because of SOA's longer history. The two technologies are complementary. Whereas SOA can be used to construct large and complex applications that scale both horizontally and vertically, cloud computing applications tend to be scaled vertically. Horizontal scaling refers to applications with a large number of different business processes operating. Vertical scaling refers to large applications with a limited number of business processes operating. SOA techniques may be applied in both instances.

Introducing Service Oriented Architecture

Service Oriented Architecture (SOA) is a specification and a methodology for providing platform- and language-independent services for use in distributed applications. A service is a repeatable task within a business process, and a business task is a composition of services. SOA describes a message-passing taxonomy for a component-based architecture that provides services to clients upon demand. Clients access a component that complies with SOA by passing a message containing metadata to be acted upon in a standard format. The component acts on that message and returns a response that the client then uses for its own purpose. A common example of a message is an XML file transported over a network protocol such as SOAP.

Usually service providers and service consumers do not pass messages directly to each other. Implementations of SOA employ middleware software to play the role of transaction manager (or broker) and translator. That middleware can discover and list available services, as well as potential service consumers, often in the form of a registry, because SOA describes a distributed architecture security and trust services are built directly into many of these products to protect communication. Middleware products also can be where the logic of business processes reside; they can be general-purpose applications, industry-specific, private, or public services.

Middleware services manage lookup requests. The Universal Description Discovery and Integration (UDDI) protocol is the one most commonly used to broadcast and discover available Web services, often passing data in the form of an Electronic Business using eXtensible Markup Language (ebXML) documents. Service consumers find a Web service in a broker registry and bind their service requests to that specific service; if the broker supports several Web services, it can bind to any of the ones that are useful.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

This architecture does not contain executable links that require access to a specific API. The message presents data to the service, and the service responds. It is up to the client to determine if the service returned an appropriate result. An SOA is then seen as a method for creating an integrated process as a set of linked services. The component exposes itself as an "endpoint" (a term of art in SOA) to the client.

The most commonly used message-passing format is an Extensible Markup Language (XML) document using Simple Object Access Protocol (SOAP), but many more are possible, including Web Services Description Language (WSDL), Web Services Security (WSS), and Business Process Execution Language for Web Services (WS-BPEL). WSDL is commonly used to describe the service interface, how to bind information, and the nature of the component's service or endpoint. The Service Component Definition Language (SCDL) is used to define the service component that performs the service, providing the component service information that is not part of the Web service and that therefore wouldn't be part of WSDL.

Note

Whatever protocol is used to negotiate a transaction, the formal definition of the transaction is referred to as the "contract." Indeed, the notion of a contract implies a certain level of service that is available to clients and that may be part of any paid service in SOA. ■

Figure 13.1 shows a protocol stack for an SOA architecture and how those different protocols execute the functions required in the Service Oriented Architecture. In the figure, the box labeled Other Services could include Common Object Request Broker Architecture (CORBA), Representational State Transfer (REST), Remote Procedure Calls (RPC), Distributed Common Object Model (DCOM), Jini, Data Distribution Service (DDS), Windows Communication Foundation (WCF), and other technologies and protocols. It is this flexibility and neutrality that makes SOA so singularly useful in designing complex applications. These services and the manner in which they interact in regards to SOA have been codified by a number of standards organizations, and some of the more prominent efforts are described later in this chapter.

Tip

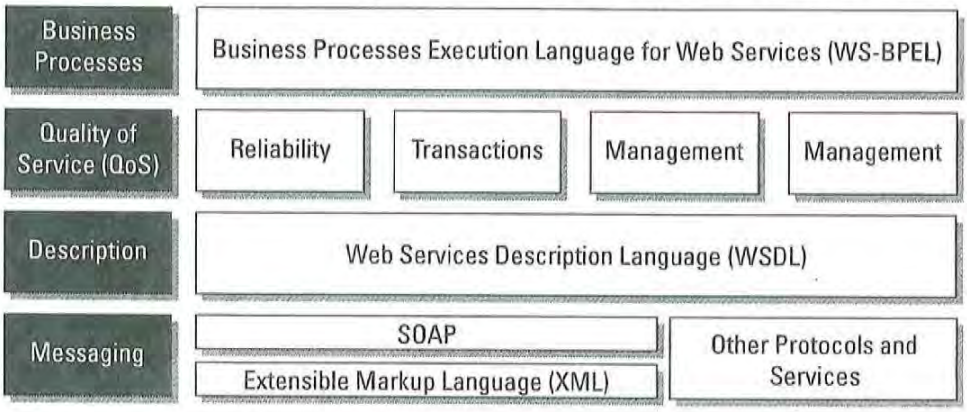
To read the IBMs SOA Foundation White Paper, see `http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf`. ■

SOA provides the framework needed to allow clients of any type to engage in a request-response mechanism with a service. The specification of the manner in which messages are passed in SOA, or in which events are handled, are referred to as their *contract*. The term is meant to imply that the client engages the service in a task that must be managed in a specified manner. In real systems, contracts may specifically be stated with a Quality of Service parameter in a real paper contract. Typically, SOA requires the use of an orchestrator or broker service to ensure that messages are correctly transacted. SOA makes no other demands on either the client (consumer) or the components (provider) of the service; it is concerned only with the interface or action boundary between the two. This is the earliest definition of SOA architecture.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.1

A protocol stack for SOA showing the relationship of each protocol to its function



Components are often written to comply with the Service Component Architecture (SCA), a language- and technology-agnostic design specification that has wide, but not universal, industry support. SCA can use the services of components that are written in the Business Process Execution Language (BPEL), Java, C#/.NET, XML, or Cobol, and can apply to C++ and Fortran, as well as to the dynamic languages Python, Ruby, PHP, and others. This allows components to be written in the easiest form that supports the business process that the component is meant to service. By wrapping data from legacy clients written in languages such as COBOL, SOA has greatly extended the life of many legacy applications.

Tip

To read David Chappel's white paper on SCA, go to `http://www.davidchappell.com/articles/Introducing_SCA.pdf`. ■

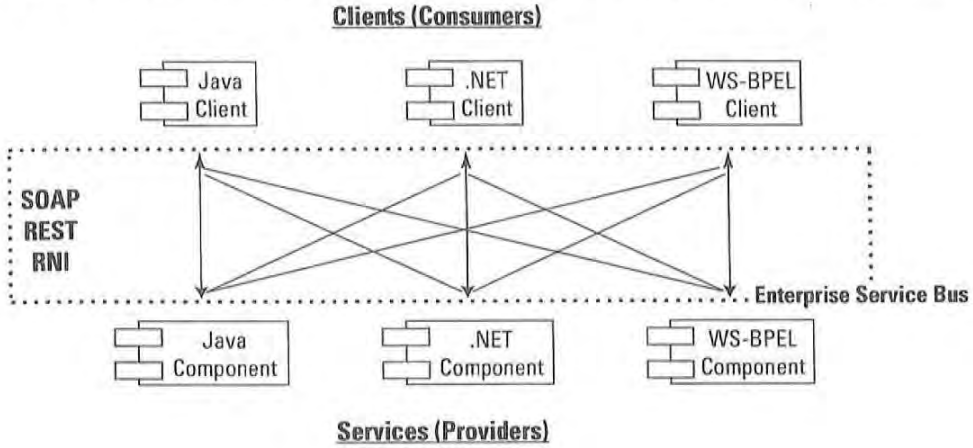
Components are coded with their service logic and their dependencies, QoS is established, and the service is instantiated. In the SCA model, data and messages are exchanged in a Service Data Object (SDO). This system of messaging using objects and services is sometimes referred to as a Data Access Service (DAS). Figure 13.2 shows how components of different types can communicate using different protocols as part of SOA.

When you combine Web services to create business processes, the integration must be managed. Two main methods are used to combine Web services: orchestration and choreography. In orchestration, a middleware service centrally coordinates all the different Web service operations, and all services send messages and receive messages from the orchestrator. The logic of the compound business process is found at the orchestrator alone. Figure 13.3 shows how orchestration is managed.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.2

SOA allows for different component and client construction, as well as access to each using different protocols.



By contrast, a compound business process that uses choreography has no central coordination function. In choreography, each Web service that is part of a business process is aware of when to process a message and with what client or component it needs to interact with. Choreography is a collaborative effort where the logic of the business process is pushed out to the members who are responsible for determining which operations to execute and when to execute them, the structure of the messages to be passed and their timing, and other factors. Figure 13.4 illustrates the nature of choreography.

What isn't clear from Figure 13.2, but is shown in Figure 13.3 (orchestration) and Figure 13.4 (choreography) is that business processes are conducted using a sequence, in parallel, or simply by being invoked (called to). An execution language like WS-BPEL provides commands for defining logic using conditional statements, loops, variables, fault handlers, and other constructs. Because a business process is a collection of activity graphs, complex processes are often shown as part of Unified Modeling Language (UML) diagrams. UML is the modeling language of the Object Management Group that provides a method for creating visual models for software in the form of 14 types of diagrams. Some of the diagram types are structure, behavior, class, component, object, interaction, state, and sequence.

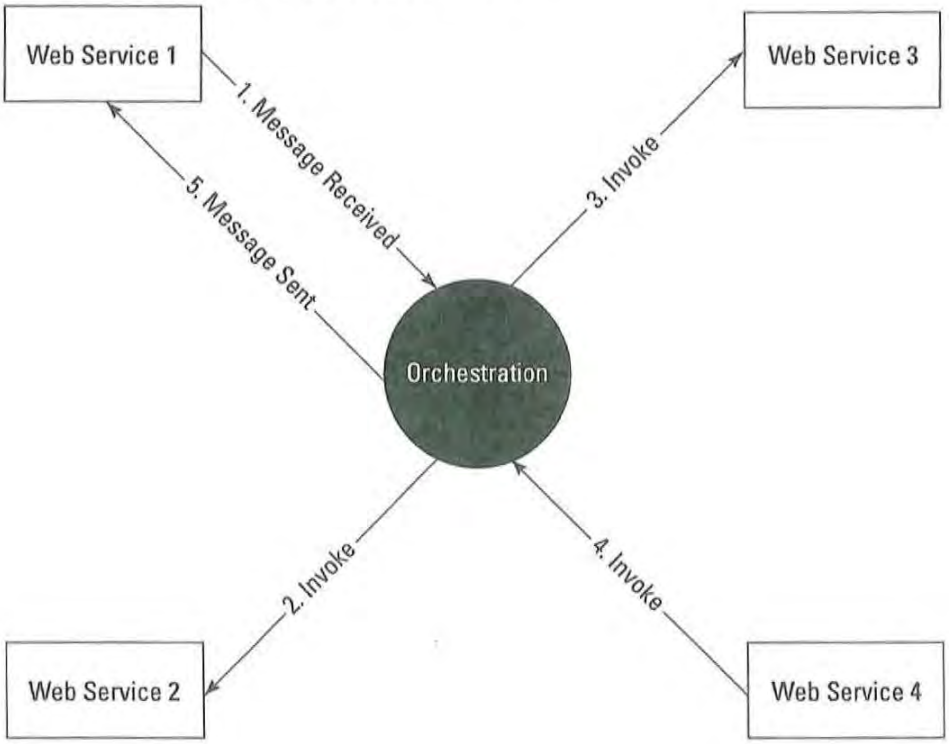
Tip

You can find a primer on BPEL by Matjaz Juric on Oracle's Web site at `http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html`. For information on SOA modeling, refer to *Service-Oriented Modeling: Analysis, Design, and Architecture,* by Michael Bell, Wiley, 2008, and Bell's later book *SOA Modeling Patterns for Service-Oriented Discovery and Analysis,* Wiley, 2010. ■

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.3

An orchestrated business process uses a central controlling service or element, referred to as the orchestrator, conductor, or less frequently, the coordinator.



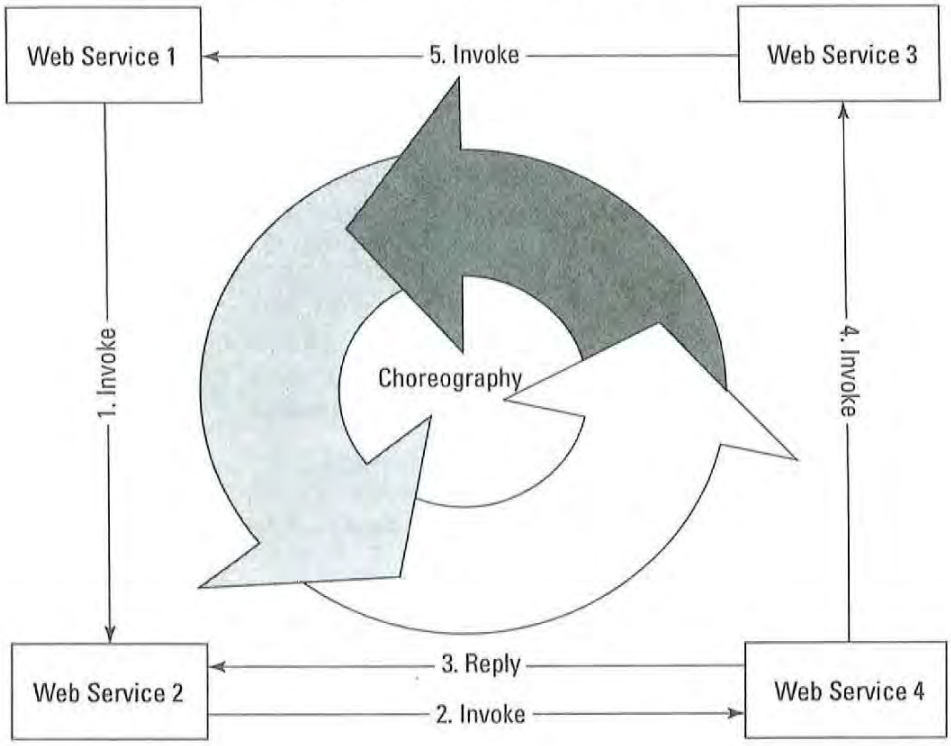
Most mature SOA implementations favor orchestration over choreography for a number of reasons. With orchestration a single central service manages the various processes, and changes to the business logic can be made in that one location. The integration of Web services into the architecture is easier than with choreography because these services don't need to know anything about the business process. Centralizing the business logic also makes it easier to put error handling mechanisms in place and to account for, manage, and analyze events that occur outside the business process that relate to a part of the process. Event handling is part of event-driven SOA or SOA 2.0, which extends Service Oriented Architecture to include both random and scheduled events that are triggered by a business process outside of a business process.

One way of performing orchestration is through the use of an Enterprise Service Bus or ESB. An ESB provides a middleware software layer for event management with a messaging infrastructure. ESBs are described later in the section called "The Enterprise Service Bus." An ESB isn't required by SOA, but it is often used to create a compliant and efficient service architecture.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.4

With choreography, business process execution is a cooperative affair.



Event-driven SOA or SOA 2.0

Event-driven SOA or SOA 2.0 is an extension of the Service Oriented Architecture to respond to events that occur as a result of business processes or perhaps cause and influence a business process. For example, in a business process, sales at a certain Web site are processed. If the business process recognizes the rate at which sales are occurring, it could perform an analysis to determine what events might influence the buying decision. This is the sort of analysis that event-driven SOA is meant to address. SOA 2.0 can allow low-level events to trigger a business process, correlate events with information contained in the SOA design, inhibit a business process if the appropriate events don't appear, or invoke a reaction or response based on a trigger.

To perform these tasks in SOA 2.0, a Causal Vector Engine (CVE) with some built-in artificial intelligence must be added to the SOA design. Events are analyzed in terms of event sequences, event relationships, and event timing to establish whether a certain condition has occurred. The CVE then determines how to react to the condition using a set of rules that are built into the system. Many CVE systems display events in a console in different contexts so that an observer can

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

analyze the display and take appropriate actions. A CVE application may include the ability to query event data in the same way that a stock ticker or trading application can query trading data. The CVE application provides the same kind of heartbeat and correlation functionality that a stock trading application does.

From the standpoint of the service requestor or consumer (client), the client simply needs to know the form required to initiate the action of the provider (service) and how to interpret the results returned from the service provider. The nature of the component's processing is unknown, the location where the processing is done is unknown, and the various operating systems and applications involved are unknown. The client is responsible for validating that the service returned the results that were expected. The SOA component is essentially a black box to the client. That is, SOA makes no demands of the component other than to conform to the rules of a standard endpoint. This level of abstraction offers operational advantages to Web service providers in that components can be continually upgraded, replaced, or moved to improve efficiencies without disrupting the clients that depend on those services, and the Quality of Service for that service can be accurately measured and delivered. In SOA, the service has been virtualized.

Cross-Ref

Communication protocols are discussed in detail in Chapter 3, "Understanding Cloud Architecture." ■

Any network service that spans different application types is a candidate for componentization. Consider a logon or authentication system. It would be wasteful to implement the same authentication functionality in several different applications when a single unified module could serve the same purpose. A single sign-on system for an accounting package, payroll module, or production database replaces three separate modules with attendant efficiencies in the amount of code that needs to be written and managed and the level of system overhead that is involved. SOA provides the rules so each application can access the authentication module in its own way, as required. What you gain with SOA is the ability to add significant capabilities with a fraction of the cost or effort and to federate applications if you desire. What you lose with SOA is the ability to perform fundamental customization of the service itself when that service is provided by a third party.

The Enterprise Service Bus

In Figure 13.5, those aforementioned hypothetical three different applications are shown interfaced with an authentication module through what has come to be called an Enterprise Service Bus (ESB). An ESB is not a physical bus in the sense of a network; rather, it is an architectural pattern comprised of a set of network services that manage transactions in a Service Oriented Architecture.

You may prefer to think of an ESB as a set of services that separate clients from components on a transactional basis and that the use of the word *bus* in the name indicates a high degree of connectivity or fabric quality to the system; that is, the system is *loosely coupled.* Messages flow from client to component through the ESB, which manages these transactions, even though the location of the services comprising the ESB may vary widely.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

An ESB is necessary but not essential to a Service Oriented Architecture because typical business processes can span a vast number of messages and events, and distributed processing is an inherently unreliable method of transport. An ESB therefore plays the role of a transaction broker in SOA, ensuring that messages get to where they where supposed to go and are acted upon properly. The service bus performs the function of mediation: message translation, registration, routing, logging, auditing, and managing transactional integrity. Transactional integrity is similar to ACID in a database system—atomicity, consistency, isolation, and durability, the essence of which is that transactions succeed or they fail and are rolled back.

FIGURE 13.5

An SOA application of a shared logon or Authentication module

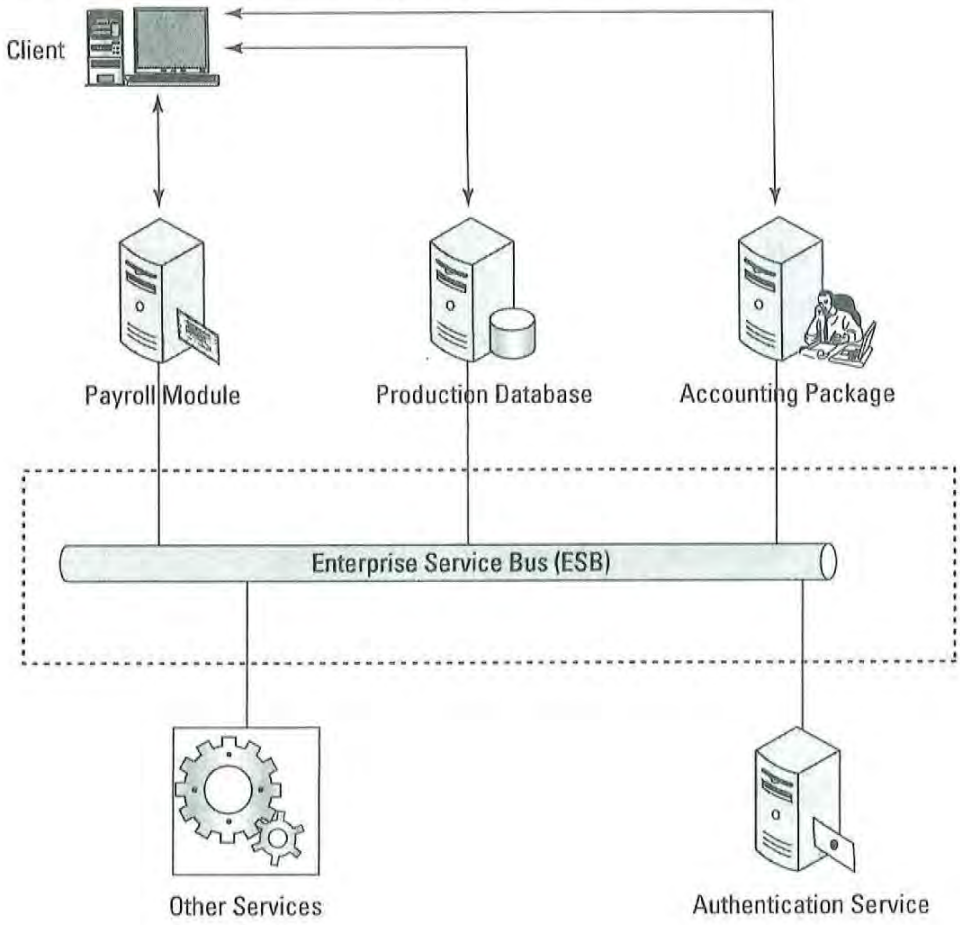


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

An ESB may be part of a network operating system or may be implemented using a set of middleware products. An ESB creates a virtual environment layered on top of an enterprise messaging system where services are advertised and accessed. Think of an ESB as a message transaction system. IBM's WebSphere ESB 7.0 is an ESB based on open standards such as Java EE, EJB, WS-Addressing, WS-Policy, and Kerberos security, and it runs on the WebSphere Application Server. It is interoperable with Open SCA. WebSphere ESB contains both a Service Federation Management tool and an integrated Registry and Repository function.

These typical features are found in ESBs, among others:

- **Monitoring services** aid in managing events.
- **Process management services** manage message transactions.
- **Data repositories or registries** store business logic and aid in governance of business processes.
- **Data services** pass messages between clients and services.
- **Data abstraction services** translate messages from one format to another, as required.
- **Governance** is a service that monitors compliance of your operations with governmental regulation, which can vary from state to state and from country to country.
- **Security services** validate clients and services and allow messages to pass from one to the other.

Figure 13.6 shows how these different services in an SOA relate to one another.

The difference between a repository and a registry in the context of a Service Oriented Architecture is subtle. Repositories and registries are both data stores, but a repository stores references to the components of the SOA, their source code, and linking information that are used to provide SOA services. An SOA registry contains references to rules, descriptions, and definitions of the services—that is, the metadata of the components.

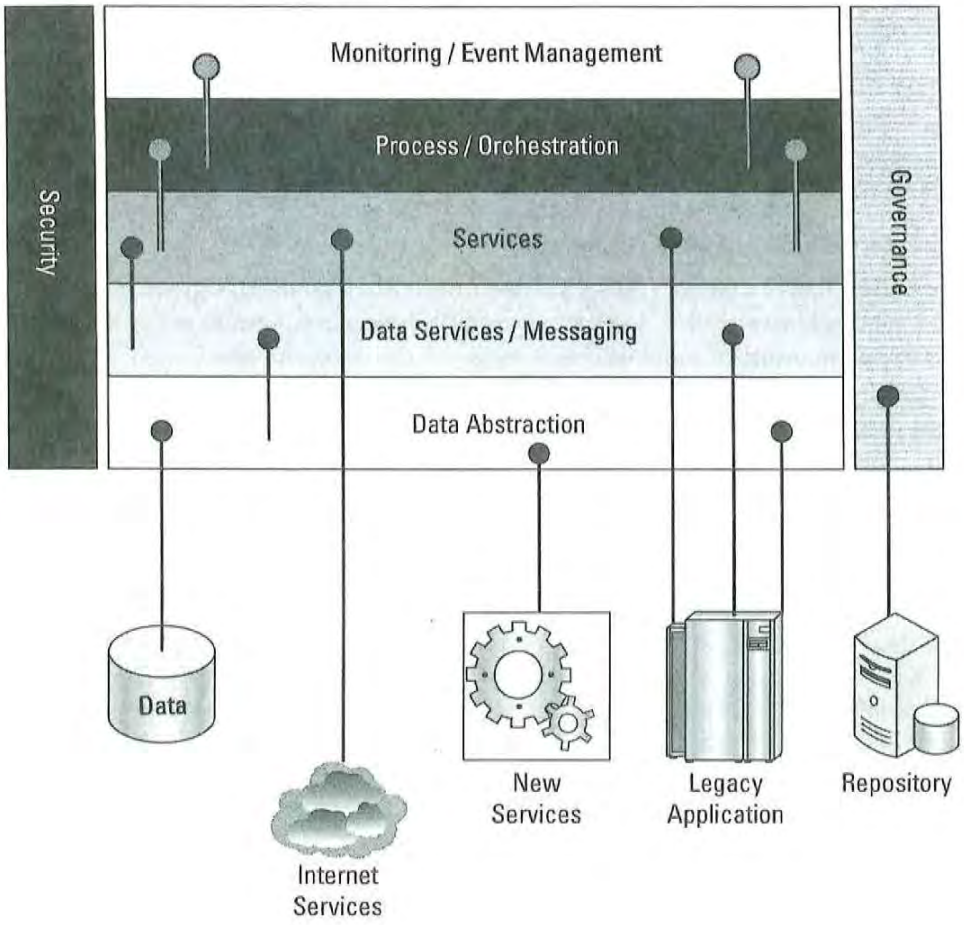
A repository serves the role that a name server does in a network operating system infrastructure, while the registry plays the role of a directory service (domain). The service broker uses the rules contained in the SOA registry to perform its function as translator and delivery agent. For developers, the registry serves as the central location to store component descriptions that allow composite applications to be created and the place in which services may be published for general use.

These services in an SOA also include the provider interfaces and standard sets of network protocols that were mentioned previously. Developers may also choose to create a Business Process Orchestration module to coordinate the access and transactional integrity of multiple business applications that integrate into a larger platform, described in the next section in more detail.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.6

This figure shows a network services model infrastructure for an SOA, which is based on the SOA meta-model of the Linthicum Group, 2007. A slightly different version of this diagram appears in *Networking Bible* by Barrie Sosinsky, Wiley, 2009.



Service catalogs

Finding any particular service and locating the service's requirement in a large SOA implementation can involve a large amount of network system overhead. To aid in locating services, SOA infrastructure often includes a catalog service. This service stores information on the following, among other things:

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- What services are available, both internal and external
- How to use a service
- Which applications are related to a particular service (dependencies)
- How services relate to one another
- Who owns the service and how a service is modified
- The event history of a service, including service levels, outages, and so on
- The nature of service contracts

Service catalogs are dynamic and under constant modification. Catalog servers have these features:

- They can be **standalone catalog servers** serving a single site.
- They serve the role of a **global catalog service** where two or more catalog servers are merged to include several sites. A global service usually requires some sort of synchronization or update to maintain a unified data store across the servers involved.
- They can be part of a **federated catalog service** where two or more global catalog servers have access to one another's information through a trusted query relationship.

Catalog services have an enormous impact on large system performance and eventually become essential as a SOA internetwork system grows. An internetwork is a network that is constructed through the consolidation of separate networks, in the same manner that the Internet has been built.

Note

The Information Technology Infrastructure Library (ITIL) developed by the United Kingdom's Office of Government Commerce (OGC) has developed a Service Catalog Management design specification as part of the ITIL v2 Service Design Package (SDP). Service Designs present best practice guidance for planned services. The ITIL Web site may be found at `http://www.itil-officialsite.com/home/home.asp`. ■

Defining SOA Communications

Message passing in SOA requires the use of two different protocol types: the data interchange format and the network protocol that carries the message. A client (or customer) connected to an ESB communicates over a network protocol such as HTTP, Representational State Transfer (REST), or Java Message Service (JMS) to a component (or service). Messages are most often in the form of the eXtensible Markup Language (XML) or in a variant such as the Simple Object Access Protocol (SOAP). SOAP is a messaging format used in Web services that use XML as the message format while relying on Application layer protocols such as HTTP and Remote Procedure Calls (RPC) for message negotiation and transmission.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The software used to write clients and components can be written in Java, .NET, Web Service Business Process Execution Language (WS-BPEL), or another form of executable code; the services that they message can be written in the same or another language. What is required is the ability to transport and translate a message into a form that both parties can understand.

An ESB may require a variety of combinations in order to support communications between a service consumer and a service provider. For example, in WebSphere ESB, you might see the following combinations:

- XML/JMS (Java Message Service)
- SOAP/JMS
- SOAP/HTTP
- Text/JMS
- Bytes/JMS

The Web Service Description Language (WSDL) is one of the most commonly used XML protocols for messaging in Web services, and it finds use in Service Oriented Architectures. Version 1.1 of WSDL is a W3C standard, but the current version WSDL 2.0 (formerly version 1.2) has yet to be ratified by the W3C. The significant difference between 1.1 and 2.0 is that version 2.0 has more support for RESTful (e.g. Web 2.0) application, but much less support in the current set of software development tools. The most common transport for WSDL is SOAP, and the WSDL file usually contains both XML data and an XML schema.

REST offers some very different capabilities than SOAP. With REST, each URL is an object that you can query and manipulate. You use HTML commands such as `GET`, `POST`, `PUT`, and `DELETE` to work with REST objects. SOAP uses a different approach to working with Web data, exposing Web objects through an API and transferring data using XML. The REST approach offers lightweight access using standard HTTP command, is easier to implement than SOAP, and comes with less overhead. SOAP is often more precise and provides a more error-free consumption model. SOAP often comes with more sophisticated development tools. All major Web services use REST, but many Web services, especially newer ones, combine REST with SOAP to derive the benefits that both offer.

Contained within WSDL are essential objects to support message transfer, including these:

- The **service** object, a container where the service resides.
- The **port or endpoint,** which is the unique address of the service.
- The **binding**, which is the description of the interface (e.g. RPC) and the transport (e.g. SOAP).
- The **portType**, or interface that defines the capabilities of the Web service, and what operations are to be performed, as well as the messages that must be sent to support the operation.
- The **operation** that is to be performed on the message.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- The **message** content, which is the data and metadata that the service operation is performed on. Each message may consist of one or more parts, and each part must include typing information.
- The **types** used to describe the data, usually as part of the XML schema that accompanies the WSDL.

Tip

A set of primers on WSDL may be found on the W3C Web site. The latest version, "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer" may be found at `http://www.w3.org/2002/ws/desc/wsdl20-primer`. ■

The code that follows is the WSDL 2.0 document that is created and analyzed in the W3C WSDL primer referenced in the accompanying tip.

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx= "http://www.w3.org/ns/wsdl-extensions">

  <documentation>
    This document describes the GreatH Web service. Additional
    application-level requirements for use of this service --
    beyond what WSDL 2.0 is able to describe -- are available
    at http://greath.example.com/2004/reservation-documentation.html
  </documentation>

  <types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/
   resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="checkAvailability"
   type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
```


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

```
      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>

    </xs:schema>
  </types>

  <interface name = "reservationInterface" >

    <fault name = "invalidDataFault"
           element = "ghns:invalidDataError"/>

    <operation name="opCheckAvailability"
           pattern="http://www.w3.org/ns/wsdl/in-out"
           style="http://www.w3.org/ns/wsdl/style/iri"
           wsdlx:safe = "true">
       <input messageLabel="In"
             element="ghns:checkAvailability" />
       <output messageLabel="Out"
             element="ghns:checkAvailabilityResponse" />
       <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>

  </interface>

  <binding name="reservationSOAPBinding"
    interface="tns:reservationInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

  </binding>

  <service name="reservationService"
       interface="tns:reservationInterface">

     <endpoint name="reservationEndpoint"
               binding="tns:reservationSOAPBinding"
               address ="http://greath.example.com/2004/
                    reservation"/>

  </service>

</description>
```


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Notice the major elements of the document. The XML document sets up the namespace, defines the interface, specifies the binding, names the service, provides the documentations for the service, and then supplies a schema that may be used to validate the document. In the message descriptions, the message types are declared. XML schema can be separate files, but what we see here is normal for WSDL, an inline schema that is part of the WSDL document. For a much more complete step-by-step description, refer to the W3C tutorial.

A WSDL file contains essential message data for a transaction, but it doesn't capture the full scope of a Service Oriented Architecture design. Additional requirements need to be specified. The functional requirements for message passing between client and service in SOA are embodied in the concept of a *service contract.* A service contract codifies the relationship between the data to be processed, the metadata that accompanies that data, the intended service, and the manner in which the service will act upon that message.

Messages therefore must have some of the following pieces of information contained inside them:

- **Header:** The header contains the name of the service, service version, owner of the service, and perhaps a responsibility assignment. This is often defined in terms of a RACI matrix where the various roles and responsibilities for processes are spelled out in terms of a set of tasks or deliverables. The acronym designates the **R**esponsible party or service, the **A**ccountable decision maker, the **C**onsulted party, and person(s) or service(s) that must be **I**nformed on the use of the service.
- **Service Type:** Examples of service types include data, business, integration, presentation, and process types.
- **Functional Specification:** This category includes the functional requirements, what service operations or actions and methods must be performed, and the manner in which a service is invoked or initiated. Invocation usually includes the URL and the nature of the service interface.
- **Transaction attributes:** A message may define a transaction that may need to be managed or tracked or be part of or include another transaction operated at a specific Quality of Service and under a specific Service Level Agreement (SLA). Security parameters also are part of a transaction's attributes, as are the role the message plays in a process and the terms or semantics used to describe the interaction of the message with a service's interface.

Depending upon the degree of formalization a service contract may require messages to carry a variable amount of information in order to be successfully transacted. You can see, therefore, how SOA expands the definition of a Web service transaction from WSDL.

Business Process Execution Language

If a message represents an atomic transaction in a Service Oriented Architecture, the next level of abstraction up is the grouping and managing of sets of transactions to form useful work and to execute a business process. An example of an execution language is the Business Process Execution Language (BPEL) or alternatively as the Web Service Business Process Execution Language

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

(WS-BPEL), a language standard for Web service interactions. The standard is maintained by the Organization for the Advancement of Structured Information Standards (OASIS) through their Web Services Business Process Execution Language Technical Committee (WSBPEL-TC; see `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`.)

BPEL is a meta-language comprised of two functions: executable commands for Web services and clients, and internal or abstract code for executing the internal business logic that processes require. A meta-language is any language whose statements refer to statements in another language referred to as the object language. BPEL is often used to compose, orchestrate, and coordinate business processes with Web services in the SOA model, and it has commands to manage asynchronous communications.

BPEL uses XML with specific support for messaging protocols such as SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, and WS-Transactions. BPEL also builds on IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG for data transport; the former is a system of directed graphs, while the latter is a block-structured language adding to additional verbs and nouns specific for business processes to BPEL, which were combined to form BPEL4WS and are being merged with BPEL. A version of BPEL to support human interaction is called BPEL4People, and it falls under the WS-HumanTask specifications of OASIS.

BPEL was designed to interact with WSDL and define business processes using an XML language. BPEL does not have a graphical component. A business process has an internal or executable view and an external or abstract view in BPEL. One process may interact with other processes, but the goal is to minimize the number of specific extensions added to BPEL to support any particular business process. Data functions in BPEL support process data and control flow, manage process instances, provide for logic and branching structures, and allow for process orchestration. Because transactions are long-lived and asynchronous, BPEL includes techniques for error handling and scopes transactions. As much as possible, BPEL uses Web services for standards and to assemble and decompose processes.

Business process modeling

SOA was created by the industry to solve a problem: how to make disparate, diverse, and distributed services talk to disparate and diverse clients. The final result of an SOA project isn't the access of services, per se; it is the creation of a business process. In a complex business project, the developers juggle many clients and many services, which can make visualization of the overall system difficult. To address this problem, various modeling tools have been developed to support SOA development and optimization, system and process management, change and life-cycle management.

Several methodologies have been developed to model SOAs. Working in a software package to model your business processes is similar in approach to designing and optimizing a relational database in entity-relationship, object-role modeling package, or another Computer Aided System Engineering (CASE) tool for data storage—and equally as valuable.

Commonly encountered system models include the following:

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **Unified Modeling Language (UML):** The UML standard is the work of the Object Management Group (`http://www.omg.org/`). UML creates graphical representations of software systems in the form of a set of diagram types. Elements in a UML architectural blueprint include actors, business processes, logic modules (components), program routines, database schemas, software components, and activities. UML diagrams are separated into seven structural types and four behavior types; structure types model the components of the system, while behavior types model states, actions, and events. UML is widely used in the industry for software system modeling. A developed system model can be reduced automatically to code.
- **XML Metadata Interchange (XMI):** XMI is another standard of the Object Management Group (OMG) and is used to exchange metadata using the Extensible Markup Language (XML). Metadata is structured into a metamodel that fits into the OMG's Meta-Object Facility. UML models often use XMI as their interchange format, although they can be used by other languages. XMI files are not generally interchangeable between the different modeling languages that can use them. XMI has been codified as an international standard by ISO, as ISO/IEC 19503:2005.
- **Systems Modeling Language (SysML)** is an open-source extension of the part of the UML system dealing with profiles. It is smaller, more focused, and easier to learn and work with than UML itself. SysML reuses 7 of UML 2.0's 13 diagrams. The effort to develop SysML was rolled into OMG in 2008, but remains open source. SysML can use XMI and is developing toward support of ISO 10303, which is the Standard for the Exchange of Product Model Data (STEP) AP-233. STEP aims to create the mechanisms for sharing information between the different software engineering tools described in this section (and others).
- **Business Process Modeling Notation (BPMN)** is a methodology for representing business processes as a set of connected visual objects that illustrate workflow in a Business Process Diagram (BPD). It is similar to UML. Originally developed in the Business Process Management Initiative (BPMI), it was incorporated into the Open Management Group in 2005. A BPD can be reduced to the OASIS standard WS-Business Process Execution Language, which is an executable language for information transfer between different Web services. However, this mapping is tool specific and standardized at this point.
- **Service-Oriented Modeling Framework (SOMF):** This framework was proposed by Michael Bell and combines a modeling language with a graphical display of the various SOA components so the system can be viewed as a map of objects and associated relationships. SOMF software allows developers to create an action plan to implement their business processes and can be valuable in system and architecture optimization, tracing message pathways, positioning software assets correctly, and providing a language for describing through abstraction and generalization how the processes operate. SOMF software not only allows you to determine what needs to be done, but also allows you to run "what-if" scenarios to see how changes will impact your SOA system.

Tip

The Enterprise Architect (EA) offered by Sparx Systems (`http://www.sparxsystems.com/`) is an example of a unified modeling language tool that supports software systems modeled in these various languages. ■

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A well-known SOMF modeling technology is IBM's Service-Oriented Modeling and Architecture (SOMA), introduced in 2004. SOMA reduces services to a set of service objects and breaks down relationships into three components: the services themselves, the service components that make use of those services, and the information flows required to interact between them. Flows consist both of processes as well as their internal composition. In SOMA, domains and functional groups are identified, variables that affect processes are analyzed, and component development and object oriented analysis are used to model specific cases. SOMA is meant to provide information on service and service boundaries, service granularity, and asset analysis.

In SOMF, the modeling is based on the elements of the service life cycle:

- **Conceptualization:** Defining the processes that will be supported
- **Discovery:** Determining how components are exposed
- **Analysis:** Identifying the relationships services have to clients
- **Integration:** Implementing the messaging infrastructure that connects clients to components
- **Logical design:** Building the system and process logic and determining how orchestration or choreography will be performed
- **Architecture design:** Creating the system component architecture and reducing design to software components with specified interfaces
- **System implementation:** Building and testing the system components

Figure 13.7 shows the various practices brought together in the SOMF modeling discipline.

Four modeling topologies for message passing are used in SOMF:

- **Circular topology:** A circular topology is one where message passing is carried out in a circular fashion. There is no orchestrator in this system, and each component providing a service is responsible for knowing which message to act on and where to send a message next. That is, the choreography of the system is maintained at the component.
- **Hierarchical topology:** In a hierarchical topology, services are arranged in a tree pattern with parent/child relationships. Messages from one service to another must traverse up the branch of the tree and down another branch from the root until the matching service is found. Hierarchical topologies offer the advantage of a well-defined set of relationships, a central location (the root) where logic may reside, and a clearly stated address space. The disadvantage of a hierarchical topology is that the overhead of passing a message from one service to another isn't optimized.
- **Network topology:** A network topology has a many-to-many relationship between services and their clients. The advantage of a network topology is that the overhead associated with message passing has been minimized (often there is a direct path), but there is considerable overhead built into the system in order to maintain the many links needed.
- **Star topology:** In a star topology, services are designed to connect through a central service. The star model is favored in orchestration processes and is useful for services that use broadcasting or multicasting services, publish and subscribe, and other related systems.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.7

SOMF 2.0 practices, principles, and design methodologies

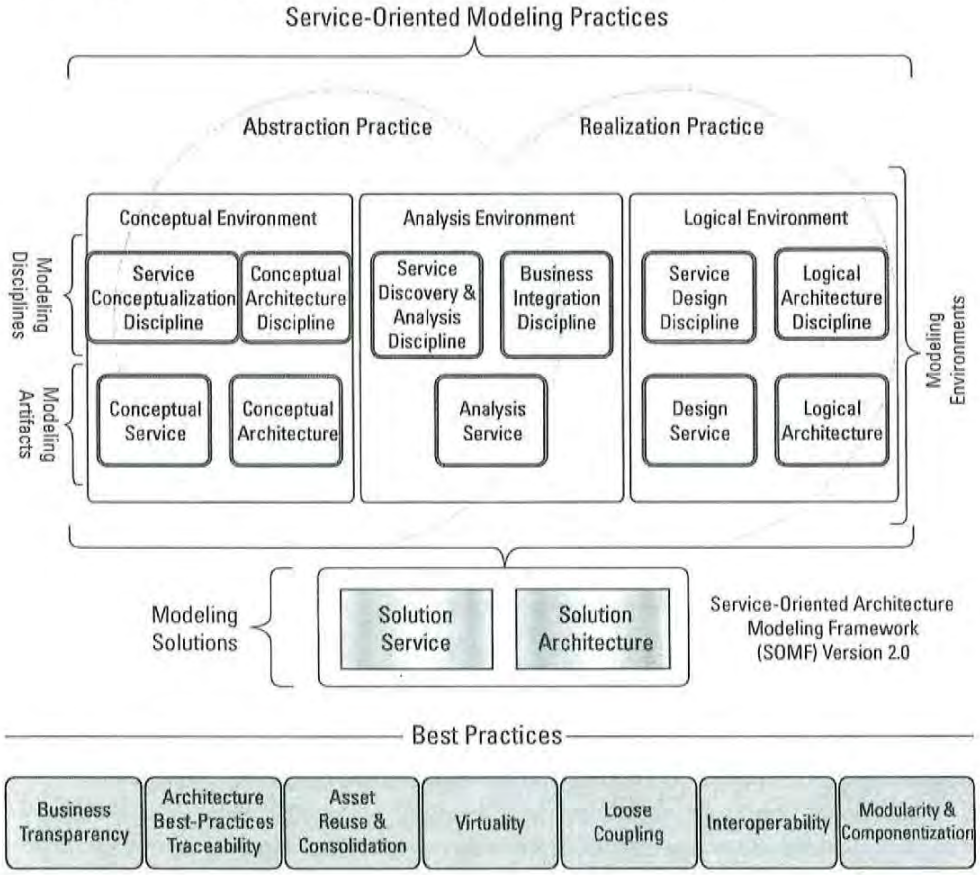


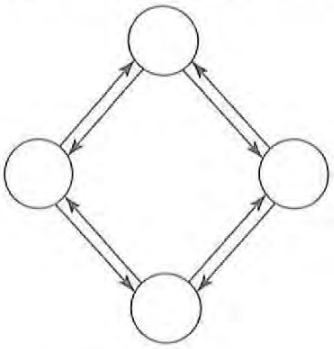
Figure 13.8 shows an interaction diagram of these four different topologies.

A service in SOMF that is granular and very narrow in scope is referred to as an atomic service. An atomic service cannot be decomposed into smaller services that provide a useful function. An example of an atomic service is a customer lookup that contains a customer ID and the customer's name and address. An address that doesn't contain the customer's name or lists a customer name without an ID wouldn't have much value, so this service stands alone. In Figure 13.8 an atomic service would be represented by one of the circles in the diagrams.

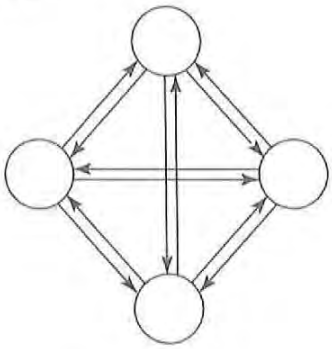
Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 13.8

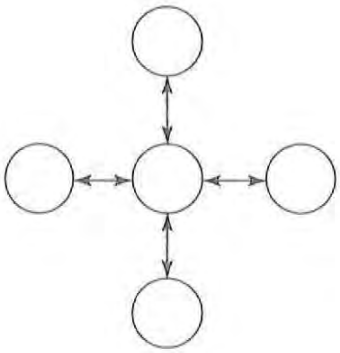
The four different message-passing topologies used in SOMF are shown above. The lines and arrows indicate message-passing pathways and relationships.



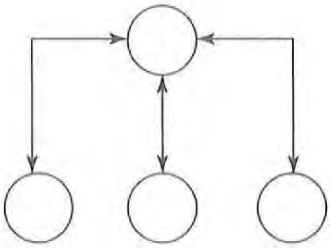
Circular topology



Network topology



Star topology



Hierarchical topology

Rarely do atomic services stand alone. A service that provides ID, name, and address might be part of a set of services that describe a bank account, customer purchase record, or any number of services. A collection of services that work together is referred to as a composite service. The Enterprise Service Bus described previously in this chapter is another example of a composite service; it contains functions for message routing, message interchange and translation, and process orchestration. A composite service is usually organized as a hierarchical topology and is multifunctional or a coarse-grained entity.

SOA describes a distributed collection of services performing business process functions. A collection of composite services that would form a process module is referred to as a *service cluster*. Service clusters may be composed of both atomic services and composite services. Large functions such as payroll modules would normally be composite SOA services.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Tip

To view a structured presentation on how to build an SOMF diagram, go to `http://www.modelingconcepts.com/pdf/SOMF_ANALYSIS_MODELING.pdf`**. The white paper "Enacting the Service Oriented Modeling Framework (SOMF) Using Enterprise Architect" by Frank Truyen (**`http://www.sparxsystems.com/downloads/whitepapers/EA-SOMF_Introduction.pdf`**) shows different SOMF diagram types.** ■

In an SOMF model, each of these three service types (atomic, composite, and clusters) appears as a specific shape, and connections are made between them that generalize, specify, expand, or contract the services they provide. Services are typed, granular services are identified, and then services may be aggregated, decomposed, unified, intersected, or subtracted from other services to suit the needs of the business process being modeled. The SOMF modeling notation has a symbol for each analysis that relates one service to another. As you build a business process, you add services to the model and connect them in ways that make sense for your workflow. When the model is complete and optimized, it is reduced to a conceptualized service that relates the business process to the specific implementation chosen. Some modeling technologies allow for the reduction of the model to executable code.

Managing and Monitoring SOA

Software for monitoring and managing an SOA infrastructure plays an important role in large SOA deployments. While SOA offers a logical design and reusable components, it does not make the task of network management any easier. If anything, SOA management requires proactive oversight because you can't wait for a particular application to fail before taking corrective action. Therefore, tools for managing SOAs tend to be multifaceted and run constantly.

SOA management tools

There are a number of network management frameworks products and suites, notably these:

- HP Software and Solutions OpenView SOA Manager (`https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10^36657_4000_100`)
- IBM Tivoli Framework Composite Application Manager for SOA (ITCAM; see `http://www-01.ibm.com/software/tivoli/solutions/`),
- Oracle BPEL Process Manager (`http://www.oracle.com/technology/bpel/index.html`)

These products have SOA tools for network management. IBM's product specializes in change management and SOA lifecycle development, and it integrates with a WebSphere and other Tivoli systems. HP SOA Manager provides dynamic mapping, monitoring, and optimization of SOA services such as Web services, software assets, and virtual services. These framework products create a central console with a variety of management views. Oracle's BPEL Process Manager and WebSphere are process managers for creating an Enterprise Service Bus.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The SOA management software technology is dynamic, with many small vendors' products some of which have been purchased and rolled into (or are being rolled into) larger systems. Oracle's recent acquisition of AmberPoint's SOA Management System is an example of this trend. BMC Software's AppSight (`http://www.bmc.com/products/product-listing/BMC-AppSight.html`) is an automated SOA problem-resolution package, as is Tidal Software's Intersperse package, which has root cause analysis services. The CA Wily SOA Solution (`http://www.ca.com/us/eitm/solution.aspx?id=8254`) is a monitoring and discovery service that can map SOA transactions and dependencies and discover components such as ESBs, Web portals, and various Web services. iTKO's LISA (`http://www.itko.com/products/index.jsp`). Enterprise SOA Testing platform specializes in testing Web service components that are used in SOA. Another example of an SOA transaction manager is OpTier's CoreFirst (`http://www.optier.com/corefirst_overview.aspx`).

Configuration and change management present a particular challenge in the area of SOA (and cloud computing in general). In addition to the fact that elements of an SOA infrastructure can be highly distributed and therefore require good discovery mechanisms, these environments also are highly virtualized. As workloads vary, solutions often provision virtual servers as needed and move these virtual servers' processing across physical servers. Virtualization will continue to challenge SOA management software well into the future.

SOA security

Any system that sends hundreds or thousands of messages across an internetwork as SOA does is subject to attack in all the traditional ways that network traffic is hijacked, spoofed, redirected, or blocked. Because SOA eliminates the use of application boundaries, the traditional methods where security is at the application level aren't likely to be effective.

Cisco has a family of products that enforce rules and policies for the transmission of XML messaging that they have named Application Oriented Networking (AON; `http://www.cisco.com/en/US/products/ps6480/`). A similar policy based XML security service may be found in Citrix's NetScaler 9.0 (`http://www.citrix.com/English/ps2/products/product.asp?contentID=21679`) Web application delivery appliance.

To address SOA security, a set of OASIS standards (`http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`) was created, which includes the following:

- **Security Assertion Markup Language (SAML)** is an XML standard that provides for data authentication and authorization between client and service. The SAML technology is used as part of Single Sign-on Systems (SSO) and allows a user logging into a system from a Web browser to have access to distributed SOA resources.
- **WS-Security (WSS)** is an extension of SOA that enforces security by applying tokens such as Kerberos, SAML, or X.509 to messages. Through the use of XML Signature and XML Encryption, WSS aims to offer client/service security.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

- **WS-SecureConversion** is a Web services protocol for creating and sharing security context. WS-SecureConversion is meant to operate in systems where WS-Security, WS-Trust, and WS-Policy are in use, and it attaches a security context token to communications such as SOAP used to transport messages in an SOA enterprise.
- **WS-SecurityPolicy** provides a set of network policies that extend WS-Security, WS-Trust, and WS-SecureConversion so messages complying to a policy must be signed and encrypted. The SecurityPolicy is part of a general WS-Policy framework.
- **WS-Trust** extends WS-Security to provide a mechanism to issue, renew, and validate security tokens. A Web service using WS-Trust can implement this system through the use of a Security Token Service (STS), a mechanism for attaching security tokens to messages and a set of mechanisms for key exchanges that are used to validate tokens and messages.

Another approach to enforcing security in SOA is to use an XML gateway that intercepts XML messages transported by SOAP or REST, identifies the source of the message, and verifies that the message was securely received. Providing XML Gateway SOA security requires a Public Key Infrastructure (PKI) so that encryption is enforced by digital signatures. Progress Software's Actional 8.0 (`http://web.progress.com/en/actional/index.html`) now includes Mindreef's SOAPscope Server and has an XML middleware service that performs diagnostic testing and Web services governance, adding that component to Actional's ability to monitor and map XML appliances and application servers.

The Open Cloud Consortium

The Open Cloud Consortium (OCC; see `http://opencloudconsortium.org/`) is an organization comprised of several universities and interested companies that supports the development of standards for cloud computing and for interoperating with the various frameworks.

OCC working groups perform these functions:

- They develop benchmarks for measuring cloud computing performance. Their benchmark and data generator for measuring large data clouds is called MalStone (`http://code.google.com/p/malgen/`).
- They provide testbeds that vendors can use to test their applications, including the Open Cloud Testbed and the Intercloud Testbed that are part of the work of the Open Cloud Testbed and Intercloud working groups.
- They support the development of open-source reference implementations for cloud computing. The Working Group on Standards and Interoperability For Large Data Clouds extends the architecture for data storage with a distributed file system, table services, and computing using MapReduce following the model that is part of Google's offering.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

MapReduce is Google's patented software framework that supports distributed large data sets organized by the Google File System (GFS) accessed by clusters of computers. The Apache Hadoop (`http://hadoop.apache.org/`) open-source system is based on MapReduce and GFS.

- They support the management of cloud computing infrastructure for scientific research as part of the Open Science Data Cloud (OSDCP) Working Group's initiative.

Relating SOA and Cloud Computing

Cloud computing is still in its infancy, and although Web services can implement a Service Oriented Architecture, it is not a requirement. Most of the large implementations of cloud computing described in this book are single-purpose applications that have been optimized on a grand scale: Carbonite's backup, Google's Gmail e-mail, and Twitter's Instant Messaging (IM) are several examples. Applications of those types have less of a need for the flexibility and loose coupling that SOA provides. As cloud applications become more diverse in scope, SOA offers an architectural blueprint for accessing diverse optimized services through a loosely coupled standardized method that provides an ability to evolve that is difficult to implement in any other way.

SOA is loosely coupled because the service is separated from the messaging. If a component doesn't provide the capabilities required, it is an easy task to switch to a different component, and switching requires almost no programming. Developers lose some of the ability to customize modules, but gain a significant advantage in simplifying their applications. Taken as a whole, applications that rely on SOA components can be very complex and appear to be tightly coupled, when in reality they are not.

SOA components are often best-of-breed service providers that can provide a measured service level and can play a role in Business Process Management (BPM) systems. The separation of services from their design allows for much easier system upgrades and maintenance.

Many Web 2.0 applications use SOA components, and SOA will become increasingly useful in larger applications that require many Web services. Web 2.0 is an acronym coined by Tim O'Reilly to describe Web services that allow for user input and modification. These applications often rely on REST and feature AJAX components in a user interface that supports Web syndication (think of the Google customizable user page), blogs, and wikis. Some people regard mashups as Web 2.0 applications as well. A *mashup* is the combination of data from two or more sources that creates a unique service. The layers added to Google maps are examples of mashups.

AJAX stands for Asynchronous JavaScipt and XM. AJAX is a set of development tools that allow for client input into Web applications; it is not a standard. Rather AJAX describes a group of technologies that leverage HTML and CSS for styling, Web objects in the Document Object Model (DOM)

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

for data, XML and XSLT for data interchange, the XMLHttpRequest for asynchronous communication, and JavaScript commands to request data from data sources.

The challenge SOA faces in designing systems to support Web 2.0 is the lack of standardization in how components in Web 2.0 are used. However, many people believe that SOA will play a role in creating what has been dubbed an "Internet of Services" where complex services will be available for use as a set of building blocks based on the convergence of SOA and Web 2.0. The Gartner Group refers to this trend as the development of "Advanced SOA," but features of SOA that are event-driven have been part of many vendors' middleware offerings for several years now.

Summary

This chapter described Service Oriented Architecture (SOA). SOA offers a design methodology for creating distributed applications using diverse clients and components. SOA defines a message-passing infrastructure from clients or consumers to and from service providers. Making SOA work correctly requires a certain set of middleware products in your infrastructure. These servers may aid in transaction management or brokering, message translation, or other services. Taken as a whole, these services are referred to as an Enterprise Service Bus (ESB).

Most message-passing protocols are based on a version structured XML, although that is not required in SOA. A variety of transport protocols are used, but SOAP and RPC are the most common ones. The nature of SOA messaging was explored. In a complex system of message passing and services, system management and security is an important consideration. Tools for setting up and running an SOA infrastructure were described.

Finally, this chapter described the relationship between SOA and cloud computing. You don't need to use SOA to build a massively scaled cloud computing application, but as cloud computing applications become more capable and user configurable, the logic and structure that SOA design imposes on infrastructure will prove to be invaluable to cloud applications. The two areas of technology benefit from their mutual convergence.

In Chapter 14, I consider the topic of transactional Web applications in cloud computing systems. That subject builds on what you have learned about SOA in this chapter, extending the discussion into the command Web APIs that are in use today.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

CHAPTER 14

Moving Applications to the Cloud

In this chapter, you learn about some of the important considerations involved in moving an application from a local or on-premises installation to one that is either fully or partly in the cloud. Some applications benefit from cloud deployment, and the cloud enhances some features.

The process for determining whether, what, and when to move your applications to the cloud involves an analysis of what critical features of the application need to be supported. After those critical features are understood, you can determine the features supported by your particular cloud service provider to see whether the cloud can support the application's critical features. Factors such as access to data, latencies, data security, and so on often limit what applications are good candidates for porting.

Two examples of application porting to the cloud are discussed in this chapter. In one application, physical hardware is eliminated by moving the entire application to the cloud. In the second example, a system is essentially cloned to the cloud to provide an overflow capability, an example of a hybrid application technique called cloud bursting.

When you move an application to the cloud, you must use the APIs of your particular cloud service provider. There are APIs for each of the types of cloud services: infrastructure, software services, and applications in the case of platform providers. These APIs are generally not interoperable. So although the situation may change in the future, an application developer must make an informed choice to select the vendor that both best suits his needs and allows him to have the greatest flexibility.

IN THIS CHAPTER

- **Learning about cloud transactions**
- **Determining the best features to move to the cloud**
- **Seeing a cloud burst solution**
- **Knowing the factors of cloud application development**

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Applications in the Clouds

When you deploy an application to the cloud, you start with the advantages and disadvantages of a distributed system that is the Internet and add to that mix the fundamental characteristics that clouds offer. In the cloud, your applications must account for system abstraction and redirection, scalability, a whole new set of application and system APIs, LAN/WAN latencies, and other factors that are specific to one cloud platform or another. In theory, any application can run either completely or partially in the cloud.

The question a developer needs to ask is whether his application's function is best served by cloud or local deployment. That answer depends upon the attributes of the application that the developer is trying to preserve or enhance, and how locating those services in the cloud impacts those attributes. This chapter takes a broad look at cloud computing from an application-specific viewpoint and attempts to highlight the factors that make cloud-based applications successful.

The location of an application or service plays a fundamental role in how the application must be written. An application or process that runs on a desktop or server is executed coherently, as a unit, under the control of an integrated program. An action triggers a program call, code executes, and a result is returned and may be acted upon.

Taken as a unit, "Request => Process => Response" is an atomic transaction. Because the transaction is executing locally within the purview of a monolithic application, the process is stateful and transaction is consistent. That is, the condition of the transaction is always known and the result is always accounted for. A coherent transaction either succeeds and is enacted, or fails and is rolled back. When rollback is not possible due to optimistic transaction commitment in a multiuser application, atomicity requires correcting the condition or performs some other compensating action at some later time.

The properties necessary to guarantee a reliable transaction in databases and other applications and the technologies necessary to achieve them have been called the ACID principle. The acronym stands for:

- **Atomicity:** The atomic property defines a transaction as something that cannot be subdivided and must be completed or abandoned as a unit.
- **Consistency:** The consistency property states that the system must go from one known state to another and that the system integrity must be maintained.
- **Isolation:** The isolation property states that the system cannot have other transactions operate on data that is currently being processed by a transaction.
- **Durability:** The durability property states that the system must have a mechanism to recover from committed transactions should that be necessary.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The ACID rules were developed by Jim Gray to apply to database technology in the late 1970s. The ACID principle is used today by any application that is reading and writing to a stored data set, which includes just about any application type you can think of.

An application that runs as a service on the Internet has a client portion that makes a request and a server portion that responds to that request. The request has been decoupled from the response because the transaction is executing in two or more places. In a distributed system, the transaction is stateless. In order to create a stateful system in a distributed architecture, a transaction manager or broker must be added so that the intermediary service can account for transactions and react accordingly when they succeed or fail.

When applications get moved to the cloud, they retain the features of a three-layered architecture, but now physical systems become virtualized systems. Virtual machines are not only stateless, but the place where program execution occurs is likely to be different every time the process runs. These fundamental properties must be accounted for in any cloud-based application.

Functionality mapping

Some applications can be successfully ported to the cloud, while others suffer from the translation. Understanding whether your particular application can benefit from cloud deployment requires that you deconstruct your application's functionality into its basic components and identify which functions are critical and can be supported by the cloud.

For example, any application that requires access to a data store quickly runs up against some of the limits that cloud computing imposes. Order transaction systems require that data in a database maintain the transactional integrity implied by the ACID model. For many non-relational cloud storage systems, such as the Amazon Simple Storage Service (S3), the newly announced Google Storage for Developers, and the Windows Azure Storage Service, the ability of the system to maintain transactional integrity through record locking isn't part of those systems. These types of storage systems are secure and store large amounts of data, but they have very slow access to that data and do not support query and retrieval well. These limitations are why all these vendors offer alternative relational cloud database systems such as SQL Azure.

In Figure 14.1, an attribute tree is constructed for an order transaction system where the functionality is decomposed into different functional areas. At the top are high-level attributes; some of these functions are essential to the operation of the application while others are not. Drilling down on the data management attribute, the second level explores data access and then access methods. A critical attribute for the application is the need to be able to access data when the client is both online and offline.

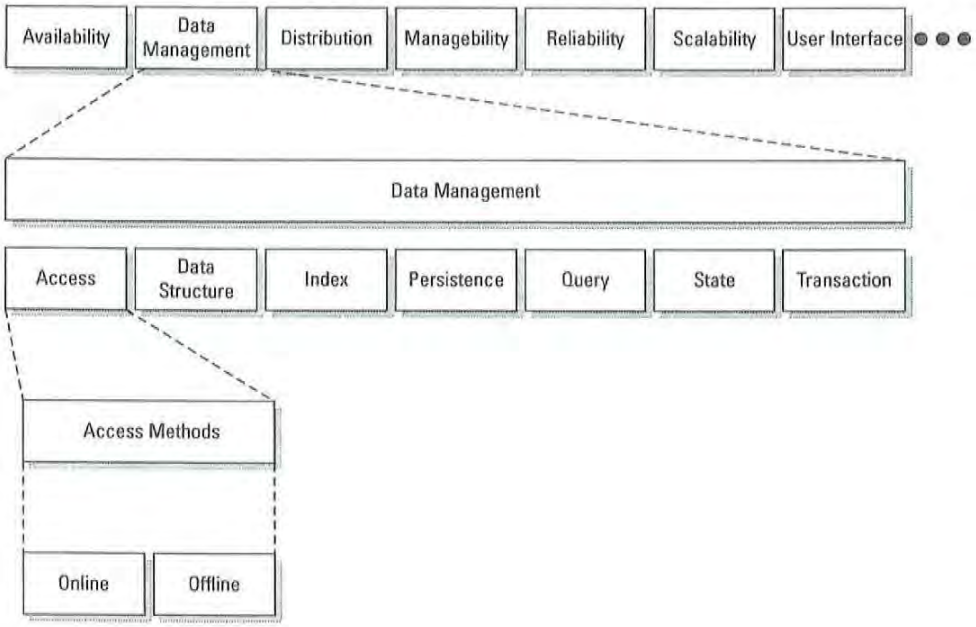
Note

This exercise isn't required for all attributes, just the critical ones, which should result in a manageable list. ■

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 14.1

An attribute map is created to expose critical functionality.



The choice to allow both online and offline data access determines the nature of your application's interaction with both cloud and local data stores. If the application needed to access data only when the client was online, then access to cloud-based storage would be the only data store your application would required. Perhaps the application could be entirely in the cloud and browser-based. The decision to allow both online and local data access means that you must create a hybrid application with a cloud component and a local component. Even if the access to data on the local system is a simple caching system, client-side support is needed. To support the application's data access, you may also be faced with building a synchronization or replication feature, which adds more overhead to the application.

This type of mapping exercise leads to some conclusions about the value of cloud computing to this particular application. You could safely conclude that an application that gets the most value from a cloud deployment is one that uses online storage without the need for offline storage. An application that needed offline storage alone might not benefit from a cloud deployment at all. In the case of a hybrid application, other factors such as scalability, costs, or ubiquitous access might offset the cost of offline access and make the cloud more attractive.

Application attributes

Table 14.1 lists some of the first- and second-level application attributes that you might want to consider in your analysis of an application's suitability to be ported to the cloud.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

TABLE 14.1

Application Attributes

First Level	Second Level	First Level	Second Level
Application	Abstraction		Implementation
	Architecture		Language/locale
	Configuration		Monitoring
	Interoperability		Operations
	Modularity		Staffing
	Object model		Startup/recovery
	Reusability		Tools
Availability	Caching	Scalability	Caching
	Fault management		Expertise
	Geographic location		Licensing
	Pooling		Lifecycle management
	Resource access		Load balancing
	Reliability		Replication
	Uptime		Scale up or out
Costs	Development		
	Resources		Staging
Data Management	Application needs		
	Data exchange	Security	Access
	Database needs		Auditing
	Index		Authentication
	Online/offline access		Authorization
	Portability		Cryptography
	Query		Encryption
	State		Identity
	Store type		Regulations
	Structure		Remote access
	Transactions		Security rules
			Trust relationships
Maintenance	APIs	User Interface	Ease of use
	Configuration		Interface features
	Deployment		User interaction

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Deconstructing an application's critical functionality is only half the process. Each cloud platform also has its own set of attributes that need to be mapped. In considering the needs of any feature, the key drivers for applications that benefit from deployment to the cloud are those that meet these criteria:

- Are not mission critical
- Are not core business functions
- Do not have sensitive data to protect
- Tolerate high network latencies or low network bandwidth
- Are legacy applications with no particular competitive advantage
- Are based on industry standard technologies
- Do not need to be customized
- Are mature enough and understood well enough to be successfully ported to the cloud

Cloud service attributes

You then want to match up application attributes to these key cloud service attributes:

- Applications
- Core services
- Infrastructure
- Platform features
- Storage

At the current stage in its development, it is impracticable to match the needs of an application to a set of cloud service providers. Each provider has a unique solution, uses its own APIs, and provides unique services. Therefore, each cloud provider needs separate developer skills, and integration between clouds would be a major chore. Perhaps someday this situation may change as more standards are developed, but at the moment application developers need to match their application to the single best vendor.

Table 14.2 lists some of the first- and second-level cloud features that you might want to consider in your analysis. You should map your critical application features to the features of a particular cloud platform to get the best match.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

TABLE 14.2

Cloud Service Attributes

First Level	Second Level	First Level	Second Level
Application	Accounting		Operating system support (platform)
	Database		Resource pooling
	Event management		Scale up or out
	Messaging		Site location
	Location service		Redundancy and replication
	Relation management		Virtual machine types
	Web server		API
Core Services	Accounting	Platform Features	Application support
	Application support		Deployment technology
	Auditing		Development environment
	Data access		Language and locale
	Identity		Programming language support
	Index		Testing
	Query		API and commands
	Transaction management	Storage	Query
	Workflow		Non-relational
Infrastructure	Application support		Relational
	Availability		Reliability
	I/O (network) characteristics		Replication
	Load balancing		SQL support

System abstraction

The cloud turns physical systems into virtual systems. Organizations choose to deploy systems to the cloud entirely when they can recreate the essential part of their process and eliminate infrastructure. As an example, consider a service that does medical imaging. In the past, this service created patient scans and then rendered the image on a local computer. After the image was rendered, it was posted to the hospital LAN and made available to the people who read the scans. When the people reading the scans were outside the hospital, across the country, or around the world, those people would have to log into the hospital server via VPN to download the file.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

The scanning service decided to eliminate infrastructure and streamline the process. The service began its redeployment by first moving the stored images off the hospital's LAN and onto shared storage in the cloud. This feature eliminated the need to maintain a great deal of managed storage locally. As the service began to outsource the reading of scans to other countries, it enabled a content delivery network feature that the cloud service provider had. CDN (Content Delivery Network) placed copies of recently used and created scans in locations that were closer to the readers and made the system faster.

The second stage in the redeployment was to eliminate the local processing associated with the scanning machines themselves. Most of the time the scanning machine was operating, it was collecting data, and an economic analysis revealed that it was significantly cheaper to process the files in the cloud.

In the new system, shown in Figure 14.2, the files are created locally and transmitted to the cloud. Virtual machines are provisioned to process the scans. The system leverages a message queuing server to create a steady stream of execution for the application server to process. At times of peak load, the system creates new machine instances to handle the load. As the application server completes the scan processing, it notifies the message queue, records the result in a database, and displays it on a Web page on a Web server, all of which are in the cloud.

This new system results in greater system efficiencies because the system is always processing at its optimum load. The rendered scans are available from anywhere viewed inside a browser. Also, because the system is scalable, the scanning service can expand to other sites and bring on new capacity to handle additional load. As the service loses sites, it can also release resources as well. When it is decided that the scans need to be converted into a different format, this can be done in a central location and doesn't need to be rolled out to the computers attached to individual scan systems.

Infrastructure, storage, and the queuing system all come together to eliminate a great deal of cost and operational complexity. This is a pure cloud play.

Cloud bursting

Many cloud deployments are hybrid applications: Part of the application is on a local system, and part is in the cloud. Often, this is the first stop on the path for many organizations migrating their applications to the cloud. There are many reasons why this is desirable, but one of the most common reasons is that the cloud can serve as excess capacity at times of high volume. This type of hybrid has been called *cloud bursting*. Examples of systems where there is high volume over short periods of time are transaction processing systems such as reservations systems.

In a reservation system, there is a certain low level of background transactions occurring at any time. At certain times, events trigger high demand. If the system builds infrastructure to accommodate peak demand, then that infrastructure is wasted.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

FIGURE 14.2

An application deployed entirely to the cloud

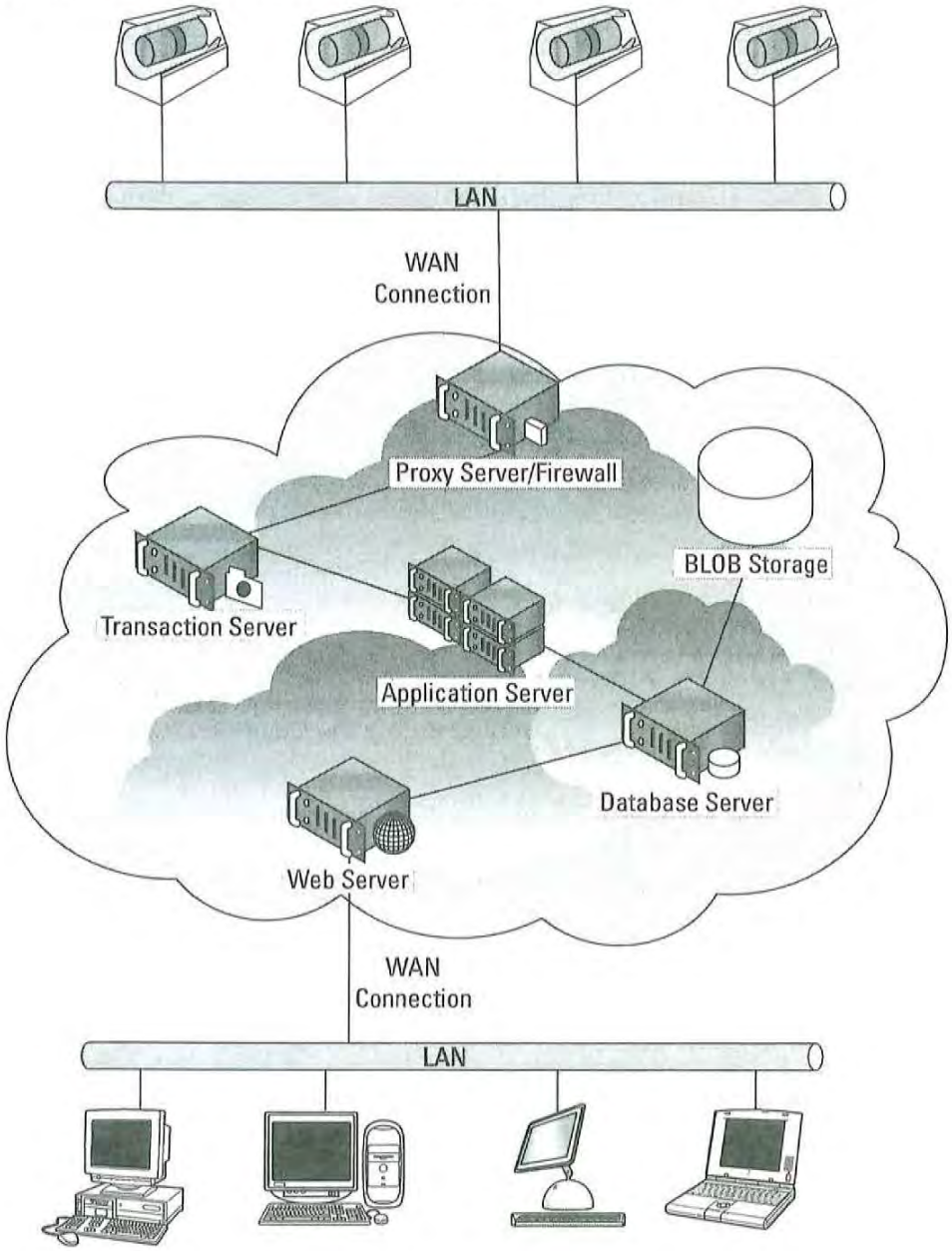
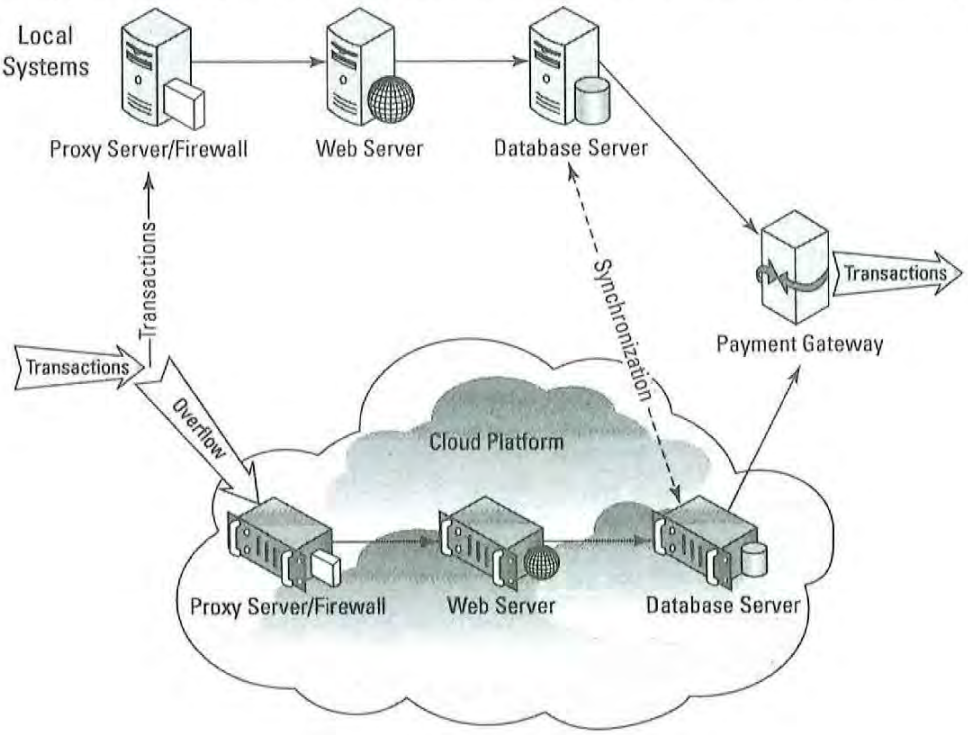


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Most systems built to perform cloud bursting have a simple underlying design: clone the local system in the cloud. Often, there may be little activity in the cloud portion of the system, but when the activity grows, the copy of the system in the cloud picks up the extra activity and, when necessary, provisions extra resources. Figure 14.3 shows a simple reservations system set up for cloud bursting.

FIGURE 14.3

An application that provides for transaction overflow in a reservation system is an example of cloud bursting.



Reservation systems often require that transactions not only are atomic, but that when there is a pool of items being reserved, the system is consistent. When a transaction enters the local branch in Figure 14.3 and another transaction enters the cloud platform branch, they can't both reserve the same item. So there must be a transaction manager in this system to manage the pool. This is shown as a dotted line between the two database servers, labeled "Synchronization." The underlying mechanism is to perform record locking on a set of database records and when the transaction or a batch of transactions completes, the system performs a commit operation.

In most reservation systems, the actual transaction commitment is a small part of the traffic and processing. Most of the traffic is generated on the Web site as users browse the content. So it makes sense in this scenario to recreate the company Web site and create additional load-balanced Web server instances as needed. You also can optimize that Web site for faster transactions with less

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

customization. If your Web site relies on dynamic data driven content, you can speed up its operation by switching more of the content over to static content. You'll need to synchronize changes between your on-premises and cloud-based Web servers in order to keep the information current.

The other step in a reservation system that is often a bottleneck is the payments gateway to credit card companies and financial institutions. It may make sense to move the payment portion completely to the cloud so that the processing of payments doesn't affect the other parts of the system. Because the commitment of the payment either is effective or not, this portion of the process does not need to be tracked. The fact that a virtual server is executing the payments and that the process is stateless has no impact in this point.

Eventually, developers will want to create composite applications that are built from the best-of-breed cloud services on multiple platforms. This offers the benefit of redundant suppliers, access to additional services and features, more data sources, and a whole host of other advantages. Cloud architectures offer enough advantages that over time large organizations will want to adopt them as a core architectural design. For example, in Figure 14.4, an internal cloud provides high-speed transactional services on the LAN, an external cloud services other needs of users, and the company cloud is replicated to multiple sites. Where services get located then depends upon factors such as cost, latency, and convenience.

FIGURE 14.4

For the users in large organizations, it is literally clouds everywhere in their future.

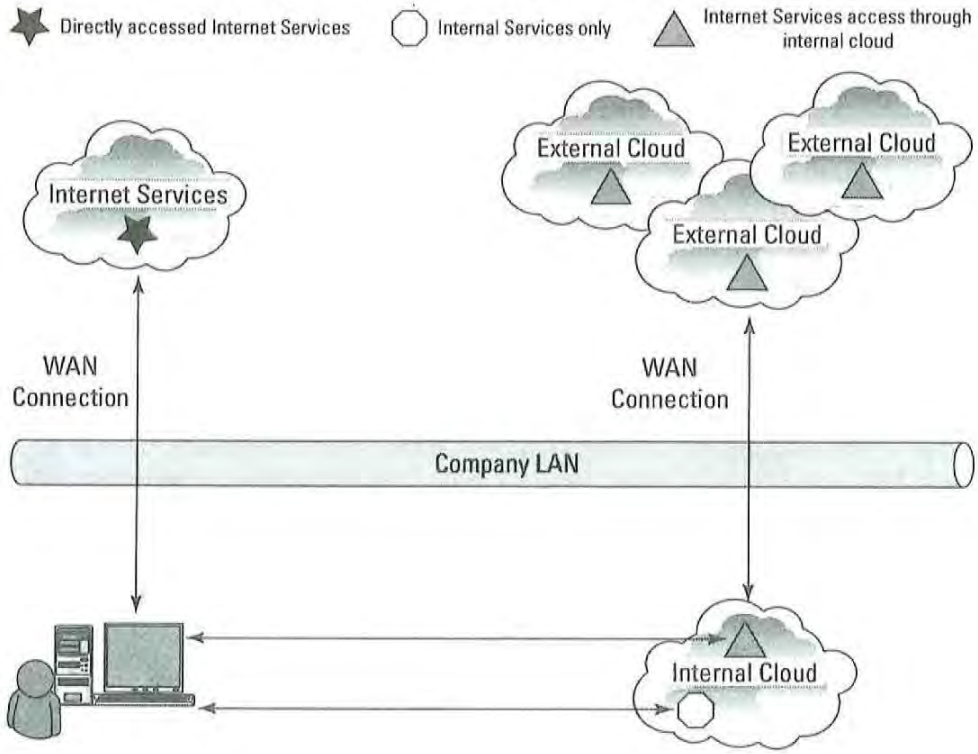


Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Applications and Cloud APIs

The nature of a cloud provider's Cloud API will impact your ability to move an application to the cloud and affect the way many of your application features operate. Cloud APIs are the Application Programming Interface to functions that exchange information in and with the cloud, request supported operations, and provide management and monitoring functions for applications running in the cloud.

At this stage in cloud computing's development, Amazon Web Service's Cloud API dominates the conversation, but that is probably subject to change. Each cloud vendor has its own specific API; most are exposed as REST, a few are exposed as SOAP, or some are both. Each API provides specific calls required by that vendor's infrastructure and service.

Most importantly, the cloud API contains the authentication and authorization mechanisms needed to access cloud services. When a vendor like Google allows other cloud application providers to access its ID mechanism (as is the case now), there is the same flexibility in using those services on other platforms.

Although efforts are underway to create more standardized cloud APIs, the situation limits the portability of any application developed for the cloud. The two cross-platform API initiatives are the Simple Cloud API (`http://www.simplecloud.org/`) and the work arising out of the Cloud Computing Interoperability Forum (`http://www.cloudforum.org/`). Several cross-platform cloud API projects are underway, including the Design Cloud, Deltacloud, jclouds, and libcloud APIs. These cross-platform APIs are based on generalizing the major cloud vendors APIs.

Each layer of a cloud application has its own specific API as well. So at the infrastructure level in addition to AWS EC2, you have the following:

- Windows Azure (`http://www.microsoft.com/windowsazure/windowsazure/`)
- VMWare vCloud (`https://www.vmware.com/products/vcloud/`)
- Racksapce Cloud Servers (`http://www.rackspacecloud.com/cloud_hosting_products/servers/api`)
- RimHosting (`http://rimuhosting.com/`)

All these present the developer with their own APIs. Individual services such as Windows Azure SQL, Flickr, and Google Maps present a service cloud API. If your application is developed in a platform such as Facebook, LinkedIn, or the Salesforce Force APIs, each platform has its own specific application API.

The point is that the decision to move an application to the cloud rapidly funnels you into a specific solution that provides a measure of vendor lock-in that, depending upon the nature of your application, can be anywhere from very easy to nearly impossible to port to any other cloud technology. This may not always be so, but it currently is true and it should give a developer some moments to pause.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

Summary

In this chapter, you learned about some of the factors involved in deciding to move an application to the cloud. Cloud computing supports some application features better than others. To determine whether your application will port successfully, you should perform a functionality mapping exercise. This process involves determining the critical application features and then matching them to the cloud provider's offering to see if those features can be supported.

Examples of applications that were ported were presented. One application virtualized the entire application in the cloud and presented users with a browser-based service. The second scenario, called cloud bursting, is an overflow solution that clones the application to the cloud and directs traffic to the cloud during times of high traffic.

The role of a cloud vendor's specific API and the impact that it has on porting an application was also considered. This aspect of a migration isn't given enough thought beforehand and can cause problems later on should you wish to move to other solutions.

In Chapter 15, "Working with Cloud-Based Storage," various storage, backup, and disaster recovery solutions offered by cloud vendors are discussed.

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC

A thorough, down-to-earth look at cloud computing

The chance to lower IT costs makes cloud computing a hot topic, and it's getting hotter all the time. If you want a terra firma take on everything you should know about the cloud, this book is it. Starting with a clear definition of what cloud computing is, *why* it is, and its pros and cons, *Cloud Computing Bible* is a wide-ranging and complete reference. You'll get thoroughly up to speed on cloud platforms, infrastructure, services and applications, security, and much more.

- Learn what cloud computing is and what it is not
- Assess the value of cloud computing, including licensing models, ROI, and more
- Understand abstraction, partitioning, virtualization, capacity planning, and various programming solutions
- See how to use Google®, Amazon®, and Microsoft® Web services effectively
- Explore cloud communication methods — IM, Twitter®, Google Buzz™, Facebook®, and others
- Discover how cloud services are changing mobile phones — and vice versa

Barrie Sosinsky

is a veteran computer book writer specializing in network systems, databases, design, development, and testing. Among his 35 technical books have been Wiley's *Networking Bible* and many others on operating systems, Web topics, storage, and application software. He has written nearly 500 articles for computer magazines and Web sites.

Shelving Category:
COMPUTERS / Enterprise Applications / General
Reader Level:
Beginning to Advanced

ISBN 978-0-470-90356-8
90000
9 780470 903568

www.wiley.com/compbooks

WILEY

Exhibit 1052
Microsoft Corporation v. Edge Networking Systems, LLC