

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re <i>Ex Parte</i> Reexamination of:	)	
U.S. Patent No. 7,640,332	)	
Issued: December 29, 2009	)	Group Art Unit: To Be Assigned
Named Inventor: Sanjay Dahiya	)	Examiner: To Be Assigned
Control Number: To Be Assigned	)	
Filed: December 20, 2007	)	
Title: SYSTEM AND METHOD FOR HOT	)	
DEPLOYMENT/REDEPLOYMENT IN	)	
GRID COMPUTING ENVIRONMENT	)	
	)	

Mail Stop *Ex Parte* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

REQUEST FOR *EX PARTE* REEXAMINATION

Requester NetApp, Inc. respectfully requests *ex parte* reexamination under 35 U.S.C. § 302 and 37 C.F.R. § 1.510 of U.S. Patent No. 7,640,332, assigned to Patent Owner Valtrus Innovations Ltd., per the assignments at reel/frame: 056157/0492. Prior art not cited or applied during prosecution renders issued claims 1-18 unpatentable under 35 U.S.C. § 103.

As this Request demonstrates, prior art references, including at least Smith, Fanshier, and i-Dalfo, raise substantial new questions of patentability (“SNQs”) with respect to claims 1-18. In particular, this prior art demonstrates claims 1-18 are unpatentable under 35 U.S.C. § 103.

NetApp respectfully requests that the Office grant this Request for *ex parte* reexamination of the ’332 patent and cancel claims 1-18.

In accordance with 37 C.F.R. § 1.510(b), this Request includes the following:

- (1) A statement pointing out each substantial new question of patentability based on prior patents and printed publications.
- (2) An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited prior art to every claim for which reexamination is requested. For each statement of the patent owner and accompanying information submitted pursuant to [37 C.F.R.] § 1.501(a)(2) which is relied upon in the detailed explanation, the request must explain how that statement is being used to determine the proper meaning of a patent claim in connection with the prior art applied to that claim and how each relevant claim is being interpreted. If appropriate, the party requesting reexamination may also point out how claims distinguish over cited prior art.
- (3) A copy of every patent or printed publication relied upon or referred to in paragraph (b) (1) and (2) of this section accompanied by an English language translation of all the necessary and pertinent parts of any non-English language patent or printed publication.
- (4) A copy of the entire patent including the front face, drawings, and specification/claims (in double column format) for which reexamination is requested, and a copy of any disclaimer, certificate of correction, or reexamination certificate issued in the patent. All copies must have each page plainly written on only one side of a sheet of paper.
- (5) A certification that a copy of the request filed by a person other than the patent owner has been served in its entirety on the patent owner at the address as provided for in [37 C.F.R.] § 1.33(c). The name and address of the party served must be indicated. If service was not possible, a duplicate copy must be supplied to the Office.
- (6) A certification by the third party requester that the statutory estoppel provisions of 35 U.S.C. [§] 315(e)(1) or 35 U.S.C. [§] 325(e)(1) do not prohibit the requester from filing the *ex parte* reexamination request.

Attachments

- (1) Certificate of Service on Patent Owner
- (2) Information Disclosure Statement Form PTO/SB/08

Exhibits

- Ex. A:** U.S. Patent No. 7,640,332 to Sanjay Dahiya ("the '332 patent")
Ex. B: Patent Prosecution History of U.S. Patent No. 7,640,332
Ex. C: Declaration of Jon Weissman, Ph.D.
Ex. D: Declaration of Gordon MacPherson
Ex. E: Curriculum Vitae of Jon Weissman, Ph.D.

Prior Art Exhibits

- PA-1:** Matthew Smith, Towards a Service-Oriented Ad Hoc Grid, Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (Jul. 2004)
- PA-2:** U.S. Patent Publication No. 2003/0225851 to Fanshier et al.
- PA-3:** U.S. Patent Publication No. 2007/0143379 to Roma i Dalfo et al.
- PA-4:** U.S. Patent No. 8,225,310 to Robertson et al.
- PA-5:** U.S. Patent Publication No. 2004/0177358 to Narayanaswamy et al.
- PA-6:** Sotomayor, Borja, The Globus Toolkit 3 Programmer's Tutorial (2004)
- PA-7:** U.S. Patent Publication No. 2003/0135536 to Lyons et al.
- PA-8:** Li Gong, Project JXTA: A Technology Overview, Sun Microsystems, Inc. (Apr. 25, 2001).
- PA-9:** Li Gong, JXTA: A Network Programming Environment, IEEE Internet Computing, Vol. 5, Issue 3 (June 2001).
- PA-10:** K. Admin et al., Open Collaborative Grid Service Architecture (OCGSA), EuroWeb 2002 Conference (Dec. 2002).
- PA-11:** Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations, International Journal of High Performance Computing Applications, 15(3), 200-222 (2001)
- PA-12:** F. Dougliis, The Grid Grows Up, IEEE Internet Computing, Volume 7, Issue 4 (Aug 2003)
- PA-13:** Ramos, T., An Extensible Resource Discovery Mechanism for Grid Computing Environments, Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06) (May 2006).
- PA-14:** Brebner, P., Deployment of Infrastructure and Services in the Open Grid Services Architecture (OGSA), Lecture Notes in Computer Science, Vol. 3798 (Nov. 2005).
- PA-15:** Laszewski, G., A File Transfer Component for Grids, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'03 (June 2003).
- PA-16:** Ferreira, L., Globus Toolkit 3.0 Quick Start, IBM Redpaper (1st Ed. Sept. 2003)

- PA-17:** Data Management using GridFTP, The 7th Global Grid Forum Meeting, Tokyo, Japan (Mar. 2003).
- PA-18:** Kurniawan, B., Java for the Web with Servlets, JSP, and EJB (1st Ed. Apr. 2002).
- PA-19:** Friese, T., *Service-Oriented Ad Hoc Grid Computing*, Dissertation Philipps-Universität Marburg (Oct. 2006)
- PA-20:** U.S. Patent Publication No. 2005/0257215 to Denby et al.
- PA-21:** Globus Toolkit® v2.2 (GT2) Tutorial – Data Management (2002)

Litigation Document Exhibits

- L-1:** Complaint for Patent Infringement, *Valtrus Innovations Ltd. v. NetApp, Inc.*, 2:25-cv-00517, Dkt. 1 (E.D. Tex. May 9, 2026)
- L-2:** Consolidation Order, *Valtrus Innovations Ltd. et al. v. NetApp, Inc.*, No. 2:25-cv-00517, Dkt. 11 (E.D. Tex. July 26, 2025)
- L-3:** Complaint for Patent Infringement, *Valtrus Innovations Ltd. v. The Home Depot, Inc. et al.*, 2-25-cv-00081, Dkt. 1 (E.D. Tex. Jan. 27, 2025)
- L-4:** Complaint for Patent Infringement, *Valtrus Innovations Ltd. v. Lenovo Grp. Ltd., et al.*, 2-25-cv-00080, Dkt. 1 (E.D. Tex. Jan. 27, 2025)
- L-5:** Complaint for Patent Infringement, *Valtrus Innovations Ltd. v. FedEx Corp., et al.*, 2-25-cv-00516, Dkt. 1 (E.D. Tex. May 9, 2025)
- L-6:** Declaratory Judgment Complaint, *NetApp, Inc. v. Valtrus Innovations Ltd., et al.*, No. 1:25-cv-6138, Dkt. 1 (S.D.N.Y. July 25, 2025)

TABLE OF CONTENTS

I.	RELATED PROCEEDINGS.....	1
A.	Certification Regarding Statutory Estoppel.....	1
B.	Other Proceedings.....	1
C.	Request for Expedited Reexamination.....	1
II.	IDENTIFICATION OF CLAIMS THAT SHOULD BE REEXAMINED.....	1
III.	BACKGROUND OF THE TECHNOLOGY.....	1
IV.	OVERVIEW OF THE '332 PATENT AND ITS PROSECUTION HISTORY.....	3
A.	'332 Patent Specification.....	3
B.	The Claims of the '332 Patent.....	5
C.	'332 Patent Prosecution History.....	9
D.	Level of Ordinary Skill in the Art.....	9
V.	CLAIM CONSTRUCTION.....	10
VI.	IDENTIFICATION OF PRIOR ART AND SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY.....	10
A.	Identification of Prior Art.....	10
B.	Substantial New Questions of Patentability.....	10
1.	The Combined Teachings of Smith and Fanshier Raise a Substantial New Question of Patentability with Respect to Independent Claims 1, 9, and 14.....	12
2.	The Combined Teachings of Smith and Fanshier Raise a Substantial New Question of Patentability with Respect to Dependent Claims 2- 8, 10-13, and 15-18.....	15
3.	The Combined Teachings of Smith and i-Dalfo Raise a Substantial New Question of Patentability with Respect to Independent Claims 1, 9, and 14.....	18
4.	The Combined Teachings of Smith and i-Dalfo Raise a Substantial New Question of Patentability with Respect to Dependent Claims 2- 8, 10-13, and 15-18.....	19

5.	The Combined Teachings of Smith, i-Dalfo, and Fanshier Raise a Substantial New Question of Patentability with Respect to Claims 1-18	20
VII.	PROPOSED REJECTIONS OF THE '332 PATENT	20
VIII.	DETAILED EXPLANATION OF THE PROPOSED REJECTIONS	21
A.	Smith and Fanshier Renders Obvious (SNQ 1) Claims 1-18	21
1.	Overview of Smith	21
2.	Smith is a Printed Publication	22
3.	Overview of Fanshier	23
4.	The Smith-Fanshier Combination	24
5.	Claim 1	30
a.	[1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”	30
b.	[1a]: “adding a new version of an application release bundle in a repository server;”	33
c.	[1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	40
d.	[1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	47
e.	[1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	56
6.	Independent Claim 9	65

a.	[9pre]: “A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising”	65
a.	[9a]: “adding a new version of an application release bundle in a repository server;”	67
b.	[9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	67
c.	[9c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	68
d.	[9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	68
7.	Independent Claim 14	69
a.	[14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”	69
a.	[14a]: “adding a new version of an application release bundle in a repository server;”	70
b.	[14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	70
c.	[14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	71

d.	[14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	71
8.	Claims 2, 10, and 15	72
9.	Claims 3, 11, and 16	73
10.	Claims 4 and 18	74
11.	Claim 5: “wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application.”	74
12.	Claims 6, 12, and 17	75
13.	Claims 7 and 13	75
14.	Claim 8: “The method of claim 1, wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.	76
B.	Smith and i-Dalfo Renders Obvious (SNQ 2) Claims 1-18.....	77
1.	Overview of i-Dalfo.....	77
2.	The Smith-i-Dalfo Combination.....	78
3.	Claim 1.....	82
a.	[1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”	82
b.	[1a]: “adding a new version of an application release bundle in a repository server;”	83
c.	[1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	84

d.	[1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	85
e.	[1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	88
4.	Independent Claim 9	89
a.	[9pre]: “A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising”	89
a.	[9a]: “adding a new version of an application release bundle in a repository server;”	91
b.	[9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	91
c.	[9c]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	91
d.	[9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	92
5.	Independent Claim 14	93
a.	[14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”	93

a.	[14a]: “a method for hot deployment/redeployment in a grid computing environment comprising:”	94
b.	[14a]: “adding a new version of an application release bundle in a repository server;”	94
c.	[14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	94
d.	[14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	95
e.	[14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	95
6.	Claims 2, 10, and 15: “selecting and invoking an appropriate hot deployment plug-in . . . and . . . hot deploying/redeploying . . . using the appropriate hot deployment plug-in”	96
7.	Claims 3, 11, and 16	96
8.	Claims 4 and 18	98
9.	Claim 5: “wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application.”	98
10.	Claims 6, 12, and 17	99
11.	Claims 7 and 13	99
12.	Claim 8: “wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.	100
C.	Smith, i-Dalfo, and Fanshier Renders Obvious (SNQ 3) Claims 1-18	100
1.	Claim 1	100

a.	[1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”.....	100
b.	[1a]: “adding a new version of an application release bundle in a repository server;”.....	101
c.	[1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”.....	101
d.	[1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”.....	101
e.	[1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”.....	103
2.	Independent Claim 9.....	104
a.	[9pre]: “A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising”.....	104
b.	[9a]: “adding a new version of an application release bundle in a repository server;”.....	104
c.	[9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”.....	105
d.	[9c]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”.....	105

e.	[9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	106
3.	Independent Claim 14	106
a.	[14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”	106
b.	[14a]: “a method for hot deployment/redeployment in a grid computing environment comprising:”	106
c.	[14a]: “adding a new version of an application release bundle in a repository server;”	107
d.	[14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	107
e.	[14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	107
f.	[14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	108
4.	Claims 2, 10, and 15: “selecting and invoking an appropriate hot deployment plug-in . . . and . . . hot deploying/redeploying . . . using the appropriate hot deployment plug-in”	108
5.	Claims 3, 11, and 16	109
6.	Claims 4 and 18	110

7.	Claim 5: “wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application.”	111
8.	Claims 6, 12, and 17	111
9.	Claims 7 and 13	112
10.	Claim 8: “wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.	112
IX.	CONCLUSION.....	113

I. RELATED PROCEEDINGS

A. Certification Regarding Statutory Estoppel

As required by 37 C.F.R. § 1.510(b)(6), NetApp certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) and 35 U.S.C. § 325(e)(1) do not prohibit NetApp from filing the instant request for *ex parte* reexamination.

B. Other Proceedings

Valtrus and Key Patent Innovations Ltd. asserted the '332 patent in *Valtrus Innovations Ltd. v. NetApp Inc.*, Case No. 2-25-cv-00517 (E.D. Tex.). *See* L-1. Valtrus also asserted the '332 patent against Home Depot, Lenovo, and FedEx in 2025. *See* L-3; L-4; L-5. On June 27, 2025, the action against NetApp was consolidated with two actions against FedEx. *See* L-2. And on July 25, 2025, NetApp filed a declaratory judgment action against Valtrus in *NetApp, Inc. v. Valtrus Innovations Ltd. et al.*, 1:25-cv-6138 (S.D.N.Y.). *See* L-6.

C. Request for Expedited Reexamination

Due to the ongoing nature of the above-identified lawsuits, Requester respectfully urges that, pursuant to 35 U.S.C. § 305, this Request be granted and reexamination be conducted not only with “special dispatch,” but also with “priority over all other cases” in accordance with M.P.E.P. § 2261.

II. IDENTIFICATION OF CLAIMS THAT SHOULD BE REEXAMINED

Under 37 C.F.R. § 1.510(b)(2), NetApp respectfully requests *ex parte* reexamination of claims 1-18 of the '332 patent and the issuance of a reexamination certificate cancelling these claims.

III. BACKGROUND OF THE TECHNOLOGY

Distributed computing in the early 2000s enabled high-performance scientific research and enterprise applications. PA-12, 1-2; PA-11, 1-3; Ex. C ¶33. Grid computing frameworks

allowed organizations to share and coordinate computational resources across geographically dispersed locations. PA-12, 1-2; PA-11, 1-3; Ex. C ¶33. These resources could be utilized to perform parallel processing tasks, making large-scale computation more accessible outside of traditional supercomputing centers. PA-12, 1-2; PA-11, 1-3; Ex. C ¶33.

At the time, service-oriented architectures (SOA) began to influence the design of distributed systems, emphasizing modularity and interoperability. PA-10, 1-4; PA-12, 1-2; PA-11, 6-10; Ex. C ¶34. One such SOA was first developed in the late 1990s by the Globus Alliance—the Globus Toolkit. PA-10, 1-4; PA-12, 1-2; PA-11, 6-10; Ex. C ¶34. The Globus Toolkit was open-source software designed to support the development of grid computing environments. PA-10, 1-4; PA-12, 1-2; PA-11, 6-10; Ex. C ¶34. The Globus Toolkit provided services for resource management, data transfer, security, and information discovery across distributed computing resources, and played a central role in allowing institutions to build and use computational grids. PA-10, 1-4; PA-12, 1-2; PA-11, 6-10; Ex. C ¶34.

The Globus Toolkit supported emerging standards like the Open Grid Services Architecture (OGSA). Ex. C ¶35; PA-10, 1-4; PA-11, 1-2; PA-13, 1-2. Key components of the toolkit included GRAM (Grid Resource Allocation Manager) for job execution, GridFTP for high-performance data transfer, and MDS (Monitoring and Discovery Service) for resource discovery and status reporting. Ex. C ¶35; PA-11, 10-11. The OGSA's modular architecture integrated diverse systems and provided a reference implementation for grid middleware, influencing academic projects and commercial grid solutions. Ex. C ¶35; PA-12, 1-2; PA-11, 10-11; PA-10, 1-4.

IV. OVERVIEW OF THE '332 PATENT AND ITS PROSECUTION HISTORY

A. '332 Patent Specification

The '332 patent discloses methods for hot deploying applications in a grid computing environment. Ex. A, Abstract, 1:10-61. Grid computing, as was well known in the art, is a “type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains.” *Id.*, 1:12-27. In a grid environment, “many containers running several applications and different servers can be running different sets of applications, i.e., the applications can run in multi-vendor and multi-version application server environments.” *Id.*, 1:46-61. This includes “multiple grid nodes[,] and each node can have hundreds of servers running on heterogeneous platforms.” *Id.*, 2:45-48. To “hot deploy” applications, according to the '332 patent, includes doing so “without having to shutdown/restart an application container.” *Id.*, 1:28-61.

Independent claim 1 recites a method for hot deployment and/or redeployment in a grid computing environment that includes “adding a new version of an application release bundle in a repository server”; “determining . . . which of the one or more grid nodes are running an application associated with the added new version of the application release bundle”; “notifying a client application manager associated [with a node] about adding the new version . . . along with a type of data transfer protocol to use”; and “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by . . . the client application manager[.]” *Id.*, 7:32-51.

This method, and an exemplary hot deployment system for performing this method, are depicted in Figures 1 and 2:

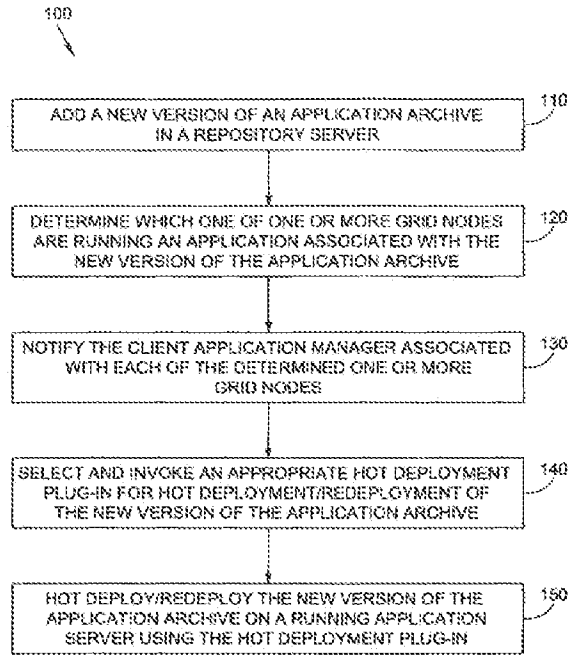


FIG. 1

Id., Fig. 1.

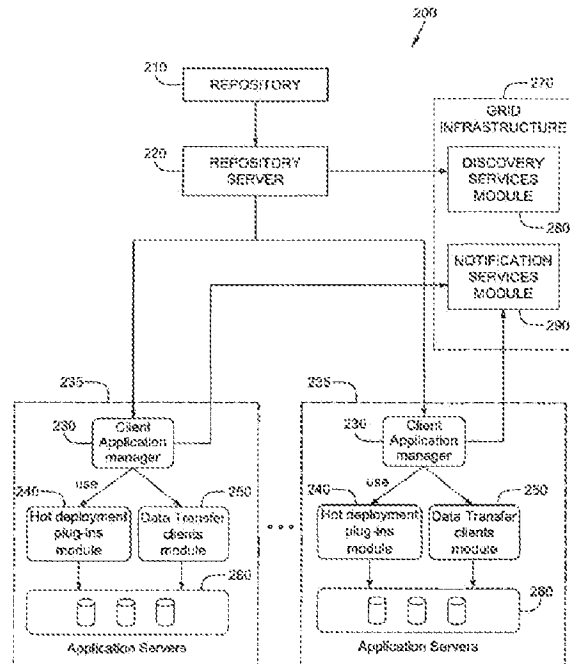


FIG. 2

Id., Fig. 2.

B. The Claims of the '332 Patent

This Request concerns claims 1-18 of the '332 patent, including independent claims 1, 9, and 14, and their dependent claims. These claims are reproduced below:

1. A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:

adding a new version of an application release bundle in a repository server;

determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;

notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.
2. The method of claim 1, wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:

selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

3. The method of claim 1, wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:
4. The method of claim 1, wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.
5. The method of claim 1, wherein the repository server stores one or more new versions of the application release bundle associated with the application.
6. The method of claim 1, further comprising:
 - registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.
7. The method of claim 1, wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.
8. The method of claim 1, wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.
9. A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform
 - a method for hot deployment/redeployment in a grid computing environment comprising:
 - adding a new version of an application release bundle in a repository server;
 - determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;
 - notifying a client application manager associated with one or more of the determined grid nodes about adding the new

version of the application release bundle along with a type of data transfer protocol to use; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.

10. The computer-readable medium of claim 9, wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:

selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

11. The computer-readable medium of claim 9, wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.

12. The computer-readable medium of claim 9, wherein the method further comprises:

registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.

13. The computer-readable medium of claim 9, wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.
14. A computer system comprising:
 - a processing unit; and
 - a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:
 - adding a new version of an application release bundle in a repository server;
 - determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;
 - notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and
 - hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.
15. The system of claim 14, wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:
 - selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and
 - hot deploying/redeploying the new version of the application release bundle on running one or more application servers

using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

16. The system of claim 14, wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.

17. The system of claim 14, wherein the method further comprises:

registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.

18. The system of claim 14, wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.

Id., 7:32-10:38.

C. '332 Patent Prosecution History

The application leading to the '332 patent (U.S. Application No. 12/003,182) was allowed without the Office issuing an office action and following the entry of an Examiner's Amendment in the Notice of Allowance. *See* Ex. B, 60-71. The Examiner's Amendment added the "adding," "determining," and "notifying" clauses in the issued independent claims. *Id.*, 64-70. The '332 patent issued on December 29, 2009. Ex. A, Cover.

D. Level of Ordinary Skill in the Art

A person of ordinary skill in the art ("POSITA") would have had at least a bachelor's degree in computer science, electrical engineering, or similar field and at least two years of experience working with distributed computing systems and/or grid computing environments.

Ex. C ¶¶43-45. Additional relevant work experience can compensate for less education, and vice versa. *Id.*

V. CLAIM CONSTRUCTION

The claims in this *ex parte* reexamination should be given their broadest reasonable interpretation. NetApp reserves the right to argue in an appropriate forum that certain limitations in the challenged claims are indefinite, lack written description support, or are invalid for other reasons not at issue here.

VI. IDENTIFICATION OF PRIOR ART AND SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. Identification of Prior Art

The earliest possible priority date for claims 1-18 of the '332 patent is December 27, 2006. Ex. A, Cover (30). Reexamination is requested in view of the following prior art:

- PA-1: Matthew Smith, *Towards a Service-Oriented Ad Hoc Grid*, Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (Jul. 2004)
- PA-2: U.S. Patent Publication No. 2003/0225851 A1 to Fanshler et al.
- PA-3: U.S. Patent Publication No. 2007/0143379 to Roma i-Dalfo et al.

These prior art documents are listed on a form PTO/SB/08 filed concurrently herewith. Copies of all the listed references are attached.

B. Substantial New Questions of Patentability

For an SNQ to be present, "it is only necessary that: (A) an item of information raises a[n] SNQ regarding at least one claim, i.e., the teaching of the item of information is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable; and (B) the same question of patentability . . . has not been decided by the Office in an earlier concluded examination or review of the patent, decided in a final holding of invalidity (after all appeals) by a federal court in a decision on the merits involving the claim, or

raised to or by the Office in a pending reexamination or supplemental examination of the patent.” M.P.E.P. § 2816.02(I); *see also In re Vivint, Inc.*, 14 F.4th 1342, 1349 (Fed. Cir. 2021). “For example, the same question of patentability may already have been decided if the Office has previously considered, in an earlier concluded examination or review of the patent, the same question of patentability based on the same prior art.” *See* M.P.E.P. § 2816.02(I) (citing *In re Recreative Techs. Corp.*, 83 F.3d 1394 (Fed. Cir. 1996)).

Each proposed rejection in this Request presents a distinct SNQ of patentability. First, a reasonable Examiner would find the teachings of the prior art submitted in this Request (as also discussed in detail below) important in deciding whether the claims are patentable. Second, the same question of patentability has not been decided by the Office in an earlier concluded review or examination of the ’332 patent. During prosecution, the Examiner did not consider any of the prior art relied upon in this Request. *See* Section VIB.1-VIB.5. Third, the same question of patentability has not been decided in a final holding of invalidity by a federal court. Though the ’332 patent is currently involved in a co-pending district court litigation, that litigation has not resolved any question of validity regarding the ’332 claims. *See* Section IB. And finally, the same question of patentability has not been raised to or by the Office in a pending reexamination or supplemental examination of the patent. To NetApp’s best knowledge, there has been no previous Office proceedings, *inter partes* review or *ex parte* reexamination, involving the claims of the ’332 patent, much less any involving the specific combination of prior art that NetApp has raised below.

1. The Combined Teachings of Smith and Fanshier Raise a Substantial New Question of Patentability with Respect to Independent Claims 1, 9, and 14

Combining Smith (PA-1) and Fanshier (PA-2) raises a substantial new question of patentability (“SNQ”) for independent claims 1, 9, and 14 because Smith and Fanshier were not before the Examiner during prosecution, and their combination renders these claims obvious.

Smith and Fanshier both disclose methods of deploying applications in a distributed computing environment: Smith discloses deployments in an ad hoc grid and Fanshier discloses deploying applications “across a network.” PA-1, Abstract, 1; PA-2, Abstract, ¶¶7, 12. Smith discloses methods of grid node discovery, which use “property scouts” to determine grid node properties, and deploy services on grid nodes using a “hot deployment service (HDS).” *See, e.g.*, PA-1, Abstract. Fanshier similarly discloses deploying applications in a distributed system using a slave deployer 116 on a network server, which is notified of a deployment and provided the deployment via an “application source 124.” PA-2, ¶¶7-8, 12-16.

Independent claim 1 recites “[a] method for hot deployment and/or redeployment in grid computing environment” made up of “one or more grid nodes.” Ex. A, 7:32-34. A “new version of an application release bundle” is added “in a repository server.” *Id.*, 7:35-36. A discovery services module determines “which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.” *Id.*, 7:37-41. A “client application manager associated with one or more of the determined grid nodes” is notified “about adding the new version of the application release bundle along with a type of data transfer protocol to use.” *Id.*, 7:42-45. The new version of the application release bundle is then “hot deploy[ed]” or “redeploy[ed] . . . on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer

protocol by a respective one of the client application managers.” *Id.*, 7:46-51. Independent claim 9 repeats the limitations of claim 1 and adds a “computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform” the method recited in claim 1. *Id.*, 8:29-51. Independent claim 14 similarly repeats claim 1’s method while adding a “computer system,” “processing unit,” and “memory having stored therein code” that executes the method of claim 1. *Id.*, 7:21-10:3.

Smith in view of Fanshier raises an SNQ for independent claims 1, 9, and 14 because their combination renders these claims obvious. Smith discloses peer-to-peer node discovery in a grid computing environment with a plurality of “nodes.” PA-1, Abstract, 1-2, 3-4. Smith utilizes “property scouts” to determine whether “the service implementation is supported by the grid platform of the node,” including whether the node has an appropriate operating system and the Java Runtime Environment. *Id.*, 2, 4-5. Smith then deploys a new or updated Java service via a disposable ClassLoader using a “hot deployment service (HDS),” which deploys the service on the grid node “non-disruptively, i.e. without requiring a restart of the OGSA platform after deployment.” *Id.*, 5-6. Smith renders obvious a “type of data transfer protocol to use” by its disclosure of JXTA and the Globus Toolkit 3.0 (GT3), where JXTA contemplates UDP or HTTP and GT3 includes GridFTP as a file transfer protocol. *Id.*, 1, 3-4. Smith’s deployment via HDS and its “disposable ClassLoader” also discloses hot deploying an application using a “hot deployment plug-in.” *Id.*, 5-6.

Fanshier adds an “application source” from which to deploy new applications, which teaches a “repository server.” PA-2 ¶¶13, 19. Fanshier also discloses notifying a slave deployer on a target server to receive a new deployment and teaches and suggests communicating with target servers via a “file distribution servlet,” which utilizes HTTP. *Id.* ¶¶13-15. Fanshier further

discloses a MyNotificationListener service class that can be used by a target server, which discloses a hot deployment plug-in. *See id.* ¶¶39-55.

As explained in Section VIII, a POSITA would have understood that Smith and Fanshier disclose similar deployments and would have combined their teachings. For example, both disclose deploying Java components for use in the Java Runtime Environment. PA-1, 4 (“Java Runtime Environment”), 6 (updating Java classes using disposable ClassLoader); PA-2, ¶16 (“Application deployment can cover all standard J2EE deployable modules, such as enterprise applications, Enterprise JavaBeans (EJBs), web applications, and connectors,” where “[a] deployable unit can be utilized according to the J2EE specification.”). Both similarly disclose doing so via Java archive files (e.g., .jar files). *See* PA-1, Fig. 8 (deploying “service.jar”); PA-2, ¶16 (“[a]n archived EJB can be identified by a jar extension”), Table 1. Smith and Fanshier therefore teach complementary and compatible systems. Ex. C ¶¶51-65. Indeed, a POSITA considering Smith would have looked to Fanshier because both are in the same field of endeavor as to each other and as to the ’332 patent: e.g., methods of deploying applications in a distributed computing environment. *Id.*

The teachings of Smith and Fanshier were not previously considered or addressed during the prosecution of the application resulting in the ’332 patent. Specifically, Smith and Fanshier were not cited on any information disclosure statement or identified by the Examiner. Ex. A, Cover Page. Because the Examiner did not consider Smith and Fanshier and their combination renders claims 1, 9, and 14 obvious, Smith and Fanshier raise an SNQ with respect to these claims.

2. The Combined Teachings of Smith and Fanshier Raise a Substantial New Question of Patentability with Respect to Dependent Claims 2-8, 10-13, and 15-18

Combining Smith and Fanshier raises an SNQ for dependent claims 2-8, 10-13, and 15-18 because neither was before the Examiner during prosecution and their combination renders the claims obvious.

Dependent claims 2, 10, and 15 depend from independent claims 1, 9, and 14, respectively, and further recite “selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use” and “hot deploying/redeploying . . . using the appropriate hot deployment plug-in.” Ex. A at 7:52-67, 8:52-67, 10:4-19. As discussed above in Section VI.B.1, Smith discloses a disposable ClassLoader for hot deploying new services and describes the use of JXTA and GT3 to facilitate communication with grid nodes. PA-1, 1, 3-4. Fanshier alternatively discloses a MyNotificationListener class definition that relies on HTTP. PA-2 ¶¶39-55. It would have been obvious to combine Smith and Fanshier to select and invoke Smith’s disposable ClassLoader via Smith’s HDS, or Fanshier’s MyNotificationListener by its Application MBean or slave deployer, which teaches selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use. *See* Section VIII. The combination of Smith and Fanshier therefore render claims dependent claims 2, 10, and 15 obvious.

Dependent claims 3, 11, and 16 depend from independent claims 1, 9, and 14, respectively, and further recite “hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.” As discussed above in Section VI.B.1, Smith in view of Fanshier raise an SNQ for

claims 3, 11, and 16 because their combination renders obvious a “data transfer client” that is a client-side component capable of communicating over JXTA, GridFTP, or HTTP to receive and deploy Smith’s new classes.

Dependent claims 4 and 18 depend from independent claims 1 and 14, respectively, and further recite “wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.” Ex. A at 8:11-13, 10:36-38. Smith in view of Fanshier raises an SNQ for claims 4 and 18 because both disclose “a set of components” (e.g., Java components) that are hosted by one or more application servers. PA-1, 5-7, Fig. 9; PA-2 ¶¶16, 57 (“Java 2 Platform, Enterprise Edition (J2EE)” deployments), ¶16 (“Application deployment can cover all standard J2EE deployable modules, such as enterprise applications, Enterprise JavaBeans (EJBs), web applications, and connectors.”).

Dependent claim 5 depends from independent claim 1 and further recites “wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application.” Ex. A at 8:14-16. Fanshier discloses deploying applications from an “application source” that may comprise a server, which is a repository server storing at least one application for deployment. PA-2 ¶¶13-15, 19. Smith also discloses deploying new versions of existing services. PA-1, 6. Smith in view of Fanshier therefore raises an SNQ for claim 5.

Dependent claims 6, 12, and 17 depend from independent claims 1, 9, and 14, respectively, and further recite “registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.” Ex. A at 8:16-21, 9:11-16, 10:30-35. Fanshier discloses “[a]pplications,” including the Application MBean, that “can listen for

these events by adding themselves as a notification listener on the application MBean.” PA-2

¶¶39-46 Smith in view of Fanshier therefore raises an SNQ for claims 6, 12, and 17.

Dependent claims 7 and 13 depend from independent claims 1 and 9, respectively, and further recite “wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.” Ex. A at 8:22-24, 9:17-19. Smith discloses using the Globus Toolkit 3.0 (GT3), which utilizes GridFTP. PA-1, 1, 5-6. Fanshier also discloses a “file distribution servlet” that utilizes HTTP. PA-2 ¶¶13-15. Smith in view of Fanshier therefore raises an SNQ for claims 7 and 13.

Dependent claims 8 depends from independent claim 1 and further recites “wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.” Ex. A at 8:25-29. Smith discloses deploying services that include Java classes, which run in containers within the Java Runtime Environment. PA-1, 5-7. Fanshier also discloses deploying EJBs that are deployable units within Java containers. PA-2 ¶¶16. Smith in view of Fanshier therefore raises an SNQ for claim 8.

As explained above, the teachings of Smith and Fanshier were not previously considered or addressed during the prosecution of the application resulting in the ’332 patent. Specifically, Smith and Fanshier were not cited on any information disclosure statement or identified by the Examiner. Ex. A, Cover Page. Because the Examiner did not consider Smith and Fanshier and their combination renders claims 2-8, 10-13, and 15-18 obvious, Smith and Fanshier raise an SNQ with respect to these claims.

3. The Combined Teachings of Smith and i-Dalfo Raise a Substantial New Question of Patentability with Respect to Independent Claims 1, 9, and 14

Combining Smith (PA-1) and i-Dalfo (PA-3) raises an SNQ for independent claims 1, 9, and 14 because Smith and i-Dalfo were not before the Examiner during prosecution and their combination renders these claims obvious.

Like Smith and Fanshier, i-Dalfo relates to deploying applications in a distributed system and thus is in the same field of endeavor as Smith, Fanshier, and the '332 patent. *See* PA-3 ¶¶2-4, 10-15, Fig. 1. To the extent Smith and Fanshier do not expressly disclose or render obvious “adding a new version of an application release bundle in a repository server” and “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use,” the combination of Smith and i-Dalfo does. *See* Section VIII.

For example, i-Dalfo discloses a “deployment repository 104 [that] includes metadata information that allows the deployment engine 110 to correctly deploy or update an application (e.g. application 116 on the computing device 102.” PA-3 ¶11. The deployment repository of i-Dalfo is “typically located on a server.” *Id.* ¶¶4, 10, 10-13, 15, 24, Fig. 1. i-Dalfo also discloses a “deployment engine 110” on a computing device that is a client application manager. *Id.* ¶¶12-13, 15, Figs. 1, 2. The deployment engine 110 “processes the metadata to determine whether there are additional resources needed to correctly install the application(s).” *Id.* ¶15. i-Dalfo’s deployment engine downloads metadata from the repository server, which i-Dalfo discloses includes “location information (e.g., URLs) of resources” for the deployment. *Id.* The deployment engine 110 can then “download the resource(s) using this location information.” *Id.* i-Dalfo’s location information URL provides a data transfer protocol to use to retrieve the deployment. *See* Section VIII.

The teachings of i-Dalfo and its combination with Smith were not previously considered or addressed during the prosecution of the application resulting in the '332 patent. Nor was i-Dalfo cited on any information disclosure statement or raised by the Examiner during prosecution. Ex. A, Cover Page. Because the Examiner did not consider Smith and i-Dalfo and their combination renders obvious independent claims 1, 9, and 14, the combination of Smith and i-Dalfo raises an SNQ with respect to these claims.

4. The Combined Teachings of Smith and i-Dalfo Raise a Substantial New Question of Patentability with Respect to Dependent Claims 2-8, 10-13, and 15-18

Combining Smith and i-Dalfo raises an SNQ for dependent claims 2-8, 10-13, and 15-18 because, as explained above, neither of these references were before the Examiner during prosecution and their combination renders dependent claims 2-8, 10-13, and 15-18 obvious.

Dependent claims 3, 11, and 16 depend from independent claims 1, 9, and 14, respectively, and further recite “hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.” i-Dalfo raises an SNQ for claims 3, 11, and 16 because it discloses a deployment engine 110 that is a data transfer client communicating over, for example, HTTP. PA-3 ¶¶12-13, 15, Figs. 1, 2. For example, i-Dalfo discloses downloading the necessary resources for the deployment via a URL and discloses that one or more modules may implement this feature. *See id.* ¶¶9, 23, 33, 36, 39, 45. Combined with Smith, Smith and i-Dalfo raises an SNQ for dependent claims 3, 11, and 16.

Dependent claims 7 and 13 depend from independent claims 1 and 9, respectively, and further recite “wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.” Ex. A at 8:22-24, 9:17-19. i-Dalfo’s use of a URL

teaches or at least renders obvious use of HTTP, which uses TCP/IP. PA-3 ¶¶12-13, 15, Figs. 1, 2. The combination of Smith and i-Dalfo therefore raises an SNQ for claims 7 and 13.

As discussed above in Section VI.B.2, Smith teaches dependent claims 2, 4-6, 8, 10, 12, 15, 17, and 18. The combination of Smith and i-Dalfo therefore raises an SNQ for these claims for the same reasons.

5. The Combined Teachings of Smith, i-Dalfo, and Fanshier Raise a Substantial New Question of Patentability with Respect to Claims 1-18

To the extent i-Dalfo does not expressly disclose or render obvious “adding a new version of an application release bundle in a repository server” and/or “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use” (it does), the combination of Smith, i-Dalfo, and Fanshier renders these features obvious as explained in above Sections VI.B.1 and VI.B.2. The combination of Smith, Fanshier, and i-Dalfo therefore raises an SNQ for claims 1-18.

VII. PROPOSED REJECTIONS OF THE '332 PATENT

The following rejections are proposed in this Request for claims 1-18 of the '332 patent:

- Claims 1-18 are unpatentable under 35 U.S.C. § 103 as being obvious over Smith in view of Fanshier.
- Claims 1-18 are unpatentable under 35 U.S.C. § 103 as being obvious over Smith in view of i-Dalfo.
- Claims 1-18 are unpatentable under 35 U.S.C. § 103 as being obvious over Smith, Fanshier, and i-Dalfo.

VIII. DETAILED EXPLANATION OF THE PROPOSED REJECTIONS

A. Smith and Fanshler Renders Obvious (SNQ 1) Claims 1-18

As explained below, the combination of Smith and Fanshler renders obvious claims 1-18.

1. Overview of Smith

Like the '332 patent, Smith discloses methods for hot deploying services in a grid computing environment. PA-1, Abstract. Using the Globus Toolkit 3.0 (GT3) on the Open Grid Services Architecture (OGSA), Smith discloses leveraging resources from “workstations within organizations[,] as well as scattered personal computers, similar to the basic idea of many distributed computing projects.” *Id.*, 1, 2.

To manage the heterogeneity and dynamic network topology of a grid environment, Smith discloses methods of “peer-to-peer node discovery, automatic node property assessment using property scouts to check how well a node meets a certain set of requirements, [and] hot deployment of services into a running system without disrupting other services already running there.” *Id.*, 1; *see also id.*, Abstract, 1-2, 2-6. In addition to deploying new services, Smith discloses updating existing services in long-running grid environments: “it must be possible to refresh the implementation of [service] classes,” as “it is likely that grid services will need to be updated at some point.” *Id.*, 6. According to Smith, “[t]he ability to do this in a running system is just as important as the ability to deploy services in the first place.” *Id.*

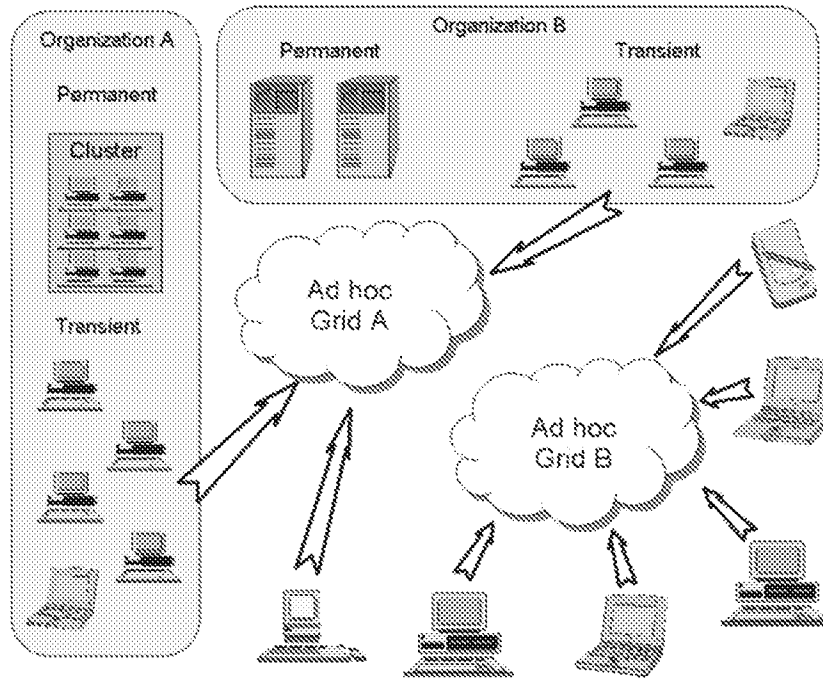


Fig. 1. Ad hoc grid architecture overview

Id., Fig. 1.

Smith discloses that its hot deployment technique provides a non-disruptive reloading of service classes within the Java Runtime Environment. *Id.*, 4 (requiring node to have a “Java Runtime Environment”), 5-6 (describing service deployment). Smith utilizes a “hot deployment service,” or HDS, and a “hot deployment algorithm” to deploy the service on a grid node: “[e]ach grid service is loaded into its own disposable ClassLoader and if the grid service needs to be redeployed, the old ClassLoader is disengaged from the system and a new ClassLoader is put in its place.” *Id.*, 5-6. These new definitions are integrated into the running system. *Id.*

2. Smith is a Printed Publication

Smith is an IEEE article added to the IEEE digital library Xplore on December 27, 2004, and was publicly available for download from IEEE Xplore by December 27, 2004. Ex. D ¶¶1-12. IEEE is a well-known, professional organization that publishes technical articles and

standards through its digital library, IEEE Xplore. *See id.* Smith was disseminated and widely available to interested persons and POSITAs relevant to this proceeding because it was publicly available on IEEE Xplore at least as of December 27, 2004. Smith therefore qualifies as a prior art, printed publication to the '332 patent.

3. Overview of Fanshier

Like Smith, Fanshier discloses deploying applications in a distributed computing environment. PA-2, Abstract, ¶¶7-8, 12-14. A deployment tool 100 notifies a master deployer 104 on an administrative server 102 of a new deployment, which notifies a slave deployer 108, 116 on a remote server 114, 122 to prepare for the deployment. *Id.* Abstract, ¶¶12-14, Figs. 1, 2. The slave deployer then “pull[s]” or “download[s]” the application from an application source 124 for the deployment. *See, e.g., id.*, Abstract, ¶¶8, 14, Fig. 2.

Fanshier’s system is depicted in Figure 1, reproduced below:

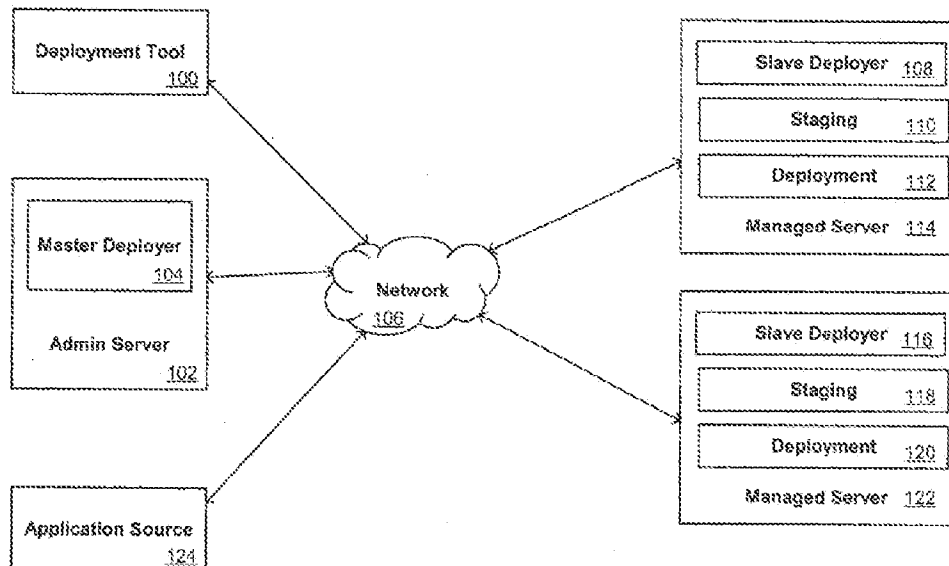


Figure 1

Id., Fig. 1.

Fanshier's slave deployer pulls or downloads the application into a staging area 204 on the target server. *Id.* ¶14. In one specific embodiment, Fanshier's application deployment includes components for deployment in the Java 2.0 Runtime Environment (J2EE), just like Smith. *Id.* ¶16. In this embodiment, Fanshier discloses utilizing Java managed bean objects (MBeans) to execute, monitor, and report status of the deployment on the server deploying the components. *Id.* ¶¶16, 35-53.

4. The Smith-Fanshier Combination

Smith and Fanshier are analogous art: Smith and Fanshier both disclose methods of deployment in distributed computing environments and, thus, are both in the same field of endeavor as each other and the '332 patent. *See* PA-1, Abstract, 1; PA-2, Abstract; Ex. C ¶¶51-52. They are likewise reasonably pertinent to the problems identified in the '332 patent because they both disclose methods that improve deploying applications in a distributed environment, where Smith focuses on grid environments and Fanshier focuses more generally on distributed servers. *See* PA-1, Abstract, 1. PA-2, Abstract, ¶¶ 6-7; Ex. C ¶¶51-52. Smith and Fanshier are therefore analogous art and compatible with one another. Ex. C ¶¶51-52.

A POSITA would have combined Smith and Fanshier: A POSITA would have found it obvious to combine Smith and Fanshier to hot deploy new versions of application components (e.g., Java classes) in a grid computing environment. Ex. C ¶53. Smith discloses a service-oriented grid computing environment, which includes grid node and application discovery, hot deployment, and use of a hot deployment "plug-in" via its disposable ClassLoader. A POSITA would have been motivated to combine Fanshier's use of an "application source"—which teaches an application repository—and Fanshier's deployment notifications, with Smith's environment. *Id.* In doing so, a POSITA would have arrived at the claimed features of the '332 patent. *Id.*

First, a POSITA would have combined Fanshier's application source with Smith's architecture to hot deploy applications in a grid environment. Ex. C ¶54. Specifically, the combination would have employed Smith's methods of node discovery, node property assessment, and methods of hot deployment in a grid environment, to deploy updated versions of applications from Fanshier's "application source," which teaches a repository server. *See* PA-1, 3-6; PA-2 Abstract, ¶¶8, 13, 19; Ex. C ¶54.

A POSITA would have been motivated to combine Fanshier's application source with Smith to maintain a repository server with updated versions of applications for deployment. Ex. C ¶55. This would have been nothing more than combining known features in the prior art according to understood principles with predictable results, which would have provided Smith with a well-known source of deployment. *Id.* Indeed, repositories were "typically located on a server that is accessible to [a deployment engine of another] computing device." PA-3, ¶4; *see also id.* PA-4, Abstract, 3:9-35, 7:10-26, 8:38-9:8, 9:39-10:1; PA-5 ¶¶17-18. It was also well known for repository servers to "install and deploy new/updated components developed by the content management system/application vendor[s]." PA-4, 1:16-34.

A POSITA would have also been motivated to combine Smith with Fanshier to deploy new application release bundles from a repository server because it would have provided a centralized storage location for versions of software components in the combined system. Ex. C ¶56. This would have improved Smith's deployment of services by maintaining a current list of deployed applications and would have tracked any updated versions added by developers for deployment. *See* PA-4, 2:40-3:8 (disclosing that updated applications "are registered at the repository, in some embodiments a change record in the repository is updated"); PA-3, ¶13 (updating applications from a single source using metadata). This would benefit Smith by

offering the ability to maintain multiple applications, and new and older versions of applications and their components, in the combined system, for example, in the event rollback or version control is desired. *See* PA-4, 2:40-3:8 (“new and/or updated classes are ‘hot deployed’ [from the repository server] . . . while retaining the superseded class as long as it is still necessary.”); *see also* PA-1, 6 (contemplating running “different versions of the same grid service on the same grid node, simply by using different ClassLoaders for each version.”).

Using one or more repositories from which to hot deploy services is compatible with Smith. Indeed, while Smith contemplates an ad hoc grid environment, Smith also discloses that its “discovery mechanism is a loosely coupled grid service itself” and contemplates the ability for its system to “use a central registry when needed.” PA-1, 4; Ex. C ¶57. A POSITA would have had a reasonable expectation of success in making the combinations because repository servers for deploying application components, including Java components, were well-known and understood in the art. Ex. C ¶57; PA-4, 3:9-35, 8:38-9:8; PA-3 ¶¶4, 10. Applying this type of storage to Smith in view of Fanshier would have been well within the knowledge and capabilities of a POSITA. Ex. C ¶57.

Second, a POSITA would have combined Fanshier’s deployment notifications with Smith’s architecture to notify target grid nodes of a deployment. Ex. C ¶58. In deploying an updated version of a service in the combined system, it would have been obvious to notify a grid node of the deployment by indicating that the deployment can be retrieved or downloaded from an application source, as taught by Fanshier. *Id.* While Smith does not expressly disclose the means by which grid nodes are notified of a deployment, a POSITA would have understood the need to do so and would have looked to Fanshier for these implementation details. *Id.* After node discovery and property assessment in Smith, it would have been obvious to notify the grid

node of a deployment. *Id.* Fanshier discloses this by its master deployer “contact[ing]” or “notify[ing]” a slave deployer of the deployment. *E.g.*, PA-2, Abstract, ¶¶7, 13, 17, 39. This includes “send[ing] [a] path” to the slave deployers 108, 116” for the slave deployer to “pull the file[s] into a staging directory 110, 118.” *Id.* ¶13.

A POSITA would have been motivated to combine Smith with Fanshier because it would have been an obvious and expected step to first notify the grid node of the deployment before transferring any necessary components for the deployment to the grid node before executing it, as contemplated by Fanshier. Ex. C ¶59. Providing Fanshier’s deployment notification in Smith would have been nothing more than combining known features in the prior art according to understood principles. *Id.* For example, a POSITA would have understood that notifying a grid node of deployments was well-known and ubiquitous in distributed computing. *See, e.g.*, PA-7 ¶26 (“If deployer 110 configuration is so set, an update may initiate at 206 when deployer 110 sends a message to application server 120 to request an update for the modified application.”); PA-2 ¶7; Ex. C ¶59. In the combined system, the notification would have informed a target grid node of the deployment and enabled it to, for example, accept the deployment and allocate resources for the deployment (e.g., for staging components and files, and for executing Smith’s hot deployment service). *See* PA-2 ¶17 (“All members of the target can be notified of the deployment and can take the appropriate action.”); Ex. C ¶59. The combination would have been well within a POSITA’s capabilities and a POSITA would have had a reasonable expectation of success in the combination. Ex. C ¶59.

Third, the Smith-Fanshier combination also renders obvious providing a notification with a “protocol to use” for the deployment, as claimed. Ex. C ¶60. A POSITA would have understood that, for a grid node to deploy a service, it must first have the files to do so and it

would have been obvious to identify where and how to retrieve these files in the combined system. *Id.* A POSITA would have understood from the use of JXTA for grid node discovery that Smith supports various protocols for nodes to communicate via JXTA, including “TCP/IP, HTTP, . . . and many other protocols.” PA-9, 95. Smith’s use of the Globus Toolkit 3.0 (GT3), and its known GridFTP capability for transferring files between grid nodes, also renders obvious such a “protocol” for deployments. *Id.* Fanshier similarly discloses using a “file distribution servlet”—a distribution mechanism that relies on the HTTP protocol—to transfer deployment files in the combined system. *See* PA-2, ¶15; PA-18, 6-15; Ex. C ¶60. By “send[ing] [a] path” for a node to retrieve or download necessary files, Fanshier also renders obvious such a claim feature. PA-2, ¶13; Ex. C ¶60.

In the Smith-Fanshier combination, it would have been obvious to a POSITA to apply one or more of these teachings to provide a “path” that utilizes a JXTA, TCP/IP, HTTP, or GridFTP file transfer protocol for grid nodes to use to retrieve the necessary components for the deployment. Ex. C ¶61. These protocols were well known and commonly used in distributed computing environments for communicating data and transferring files. *Id.* In the combination, a grid node would have been notified of a deployment that includes a path to retrieve the deployment, including one that relies on JXTA, TCP/IP, HTTP, or GridFTP. *Id.*

Implementing the notifications in Smith-Fanshier to include JXTA, TCP/IP, HTTP, or GridFTP would have been nothing more than utilizing known and available methods of transferring files between grid nodes and would have involved nothing more than combining known elements in the prior art according to understood principles. Ex. C ¶62. A POSITA would have been motivated to modify Smith to utilize its JXTA or GT3’s GridFTP protocol, or alternatively Fanshier’s TCP/IP or HTTP, because these were commonly accepted methods of

transferring files at the time. *Id.* Because these protocols were known and ubiquitous in distributed computing environments, a POSITA would have had an expectation of success in the combination. *Id.*; *see also* Section VIII.A.5.d (providing additional reasons in Element [1c] for the combination).

Fourth, it would have been obvious in Smith-Fanshier to implement a “client application manager” on a grid node to manage the deployment. Ex. C ¶63. To the extent Smith does not expressly disclose this via its disclosure of “the OGSA platform” or “OGSA application” that implements its hot deployment service (HDS), PA-1, 5-6, it would have been obvious to implement this functionality as a “slave deployer,” as expressly disclosed in Fanshier. PA-1, 5-6 (describing the features of HDS and exemplary pseudocode for loading new Java classes); PA-2, ¶¶7-8, 12-16. Fanshier’s slave deployer “deploy[s] the application” on a target server and “can pull the application from an application source, such as an archive file or application directory as specified by [a] deployment tool” before staging the application before deployment. PA-2 ¶¶7-8, 13-14. In one embodiment, the slave deployer may be an ApplicationMBean on the target server (e.g., grid node) that receives notifications and supports the deployment. PA-2 ¶¶39-53; Ex. C ¶63.

A POSITA would have been motivated to modify Smith to implement Fanshier’s slave deployer to improve coordination and implementation of deployments in Smith. Ex. C ¶64. For example, Fanshier discloses that the slave deployers can be notified by a master deployer, which directs the slave deployer to “pull” or retrieve the components of the deployment and stage them on the target grid node for deployment. PA-2 ¶¶7-8, 13-14; Ex. C ¶64. This notification and “preparation” phase allows the target grid node to “load-and-validate” the deployment. PA-2 ¶13; Ex. C ¶64. A slave deployer in Smith would have also enabled grid nodes to receive

instructions from a master deployer to deploy the updated version of the application and “inform the master deployer 104 about the success of the load” of the deployment. PA-2 ¶13; Ex. C ¶64. This also would have enabled a grid node to accept a deployment and prepare the target grid node for the deployment, including retrieving the deployment components, preparing resources on the grid node to receive the deployment (e.g., for staging components and files), and to coordinate executing of the deployment (e.g., via Smith’s hot deployment service). PA-2 ¶¶7-8, 13-14; Ex. C ¶64. A POSITA would have had a reasonable expectation of success in the combination because Smith contemplates software running on its grid nodes to execute its hot deployment service (HDS), and Fanshier discloses one known client application manager that deploys applications as contemplated by Smith in the field of distributed computing. Ex. C ¶64

Nothing in Smith’s ad hoc grid deployment and Fanshier’s deployment on “managed servers” teaches against their combination. Ex. C ¶65. Instead, a POSITA would have had an expectation of success because both Smith and Fanshier relate to deploying Java components in a distributed system, and a POSITA would have understood their teachings are interrelated and compatible, as discussed. *Id.* It was also known that grid computing environments, including ad hoc grids, may include application registration and management. *See, e.g., id.*; PA-10, 4 (disclosing a grid framework that “requires that every collaborative service be associated with a registration service,” where “[a] single registration service can register instances of different collaborative services.”).

5. Claim 1

- a. [1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”**

Smith and Fanshier discloses “*a method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more*

grid nodes.” Ex. C ¶¶66-71. Smith, for example, discloses methods of hot deploying and redeploying in a grid computing environment that includes grid nodes.

A grid computing environment that includes one or more grid nodes: Smith discloses this feature by disclosing a “a spontaneously formed, service-oriented ad hoc *grid* to harness the unused resources of idle networked workstations and high-performance computing nodes on demand.” PA-1, 1 (emphasis added). Smith’s ad hoc grid forms “cooperating heterogenous *computing nodes* into a logical community without a preconfigured fixed infrastructure and with only minimal administrative requirements.” *Id.*, 2 (emphasis added). In Figure 1, Smith depicts “how two separate ad hoc grids are composed,” one spanning two organizations, and one “created from *scattered nodes* on the Internet.” *Id.* (emphasis added).

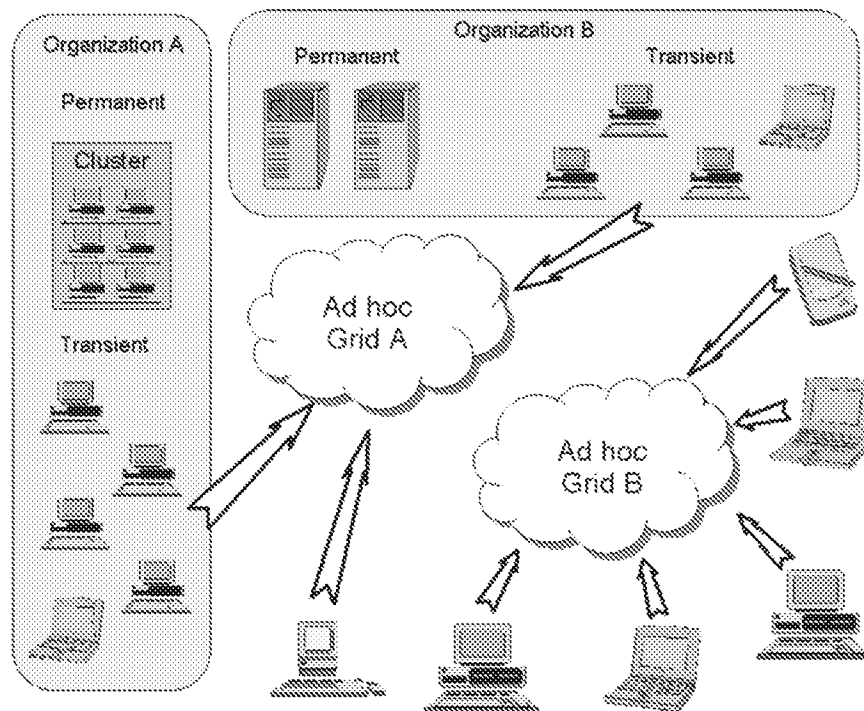


Fig. 1. Ad hoc grid architecture overview

Id., Fig. 1.

Hot deployment and/or redeployment in a grid computing environment: Smith

discloses this feature by disclosing, for example, deploying “services” on its grid nodes. PA-1, 2-3 (“Service Deployment”); *see also id.* (“In a dynamically changing environment, deployment is even more critical as there is no single deployment cycle that reaches all machines. Instead, services need to be deployed and instantiated on demand on machines as they become available.”); Ex. C ¶68. Smith similarly discloses redeploying services: “if the grid service needs to be redeployed, the old ClassLoader is disengaged from the system and a new ClassLoader is put in its place.” PA-1, 6.

When a node becomes available that meets the requirements for the deployment of a service, Smith discloses using a “*hot deployment service*” that can “autonomously carry out the deployment and use the newly available machine for its application flow.” *Id.*, 3 (emphasis added). Smith’s hot deployment service or HDS “allow[s] remote deployment of grid services without requiring the developer to have an administrative account on the system.” *Id.*, 5-6; *see also id.* (“Automatic and non-disruptive service deployment is one of the most important requirements for an ad hoc grid. With increased adoption of the grid paradigm, the number of developers wanting to use the grid will increase, thus giving each developer administrative rights for all grid nodes is not feasible, especially in an ad hoc grid environment where personal computers are also members of the grid.”); Ex. C ¶69. Smith’s hot deployment corresponds with the written description of the ’332 patent, which describes hot deployment as deploying “without having to shutdown/restart an application container.” Ex. A, 1:39-41.

Smith discloses that its HDS deploys new *and* updated versions of existing services: “[i]n long running grid environments, it is likely that grid services will need to be updated at some point.” PA-1, 6. “The ability to do this in a running system is just as important as the

ability to deploy services in the first place.” *Id.* Smith explains that, prior to its HDS, it was “impossible to *update* a class definition as the Java ClassLoader does [not] support reloading of classes, which “effectively makes it impossible to *update* a previously installed service unless all contained classes are renamed or the GT3 environment is restarted.” *Id.* (emphases added). By non-disruptively reloading service classes via its disposable ClassLoader technique, Smith’s HDS supported deployment and redeployment of new and updated services. *See id.*

Smith therefore discloses [1pre]. Ex. C ¶¶66-71.

b. [1a]: “adding a new version of an application release bundle in a repository server;”

The combination of Smith and Fanshier discloses “*adding a new version of an application release bundle in a repository server.*” Ex. C ¶¶72-84. For example, Smith discloses “a new version of an application release bundle,” and Fanshier renders obvious adding the new version of an application release bundle and deploying it from “a repository server.”

New version of an application release bundle: as discussed above, Smith discloses deploying new versions of grid services on grid nodes. Ex. C ¶¶66-71. For example:

It must be possible to refresh the implementation of these classes. In long running grid environments, it is likely that grid services will need to be updated at some point. The ability to do this in a running system is just as important as the ability to deploy services in the first place.

PA-1, 6. To deploy new services and update existing services, Smith discloses “enabl[ing] a non-disruptive reloading of service classes,” where “[e]ach grid service is loaded into its own disposable ClassLoader[,] and if the grid service needs to be redeployed, the old ClassLoader is disengaged from the system and a new ClassLoader is put in its place.” *Id.*

Smith utilizes its hot deployment service or HDS to effectuate this deployment. *Id.* Smith discloses “remote clients call [a] deployment service where the HotFactoryCallBackImpl

creates a DisposableClassLoader for each service creation request received,” and “[t]he GridServiceHandle is then returned to the clients.” *Id.* “The new ClassLoader is then capable of loading the new class definitions and integrating them into the running system.” *Id.* Smith shows an example of this deployment in Figure 9 and GridService C: “[a]fter the instantiation of the third service ‘C’, its implementation was updated and reinstantiated as ‘C*’.” *Id.*, 6.

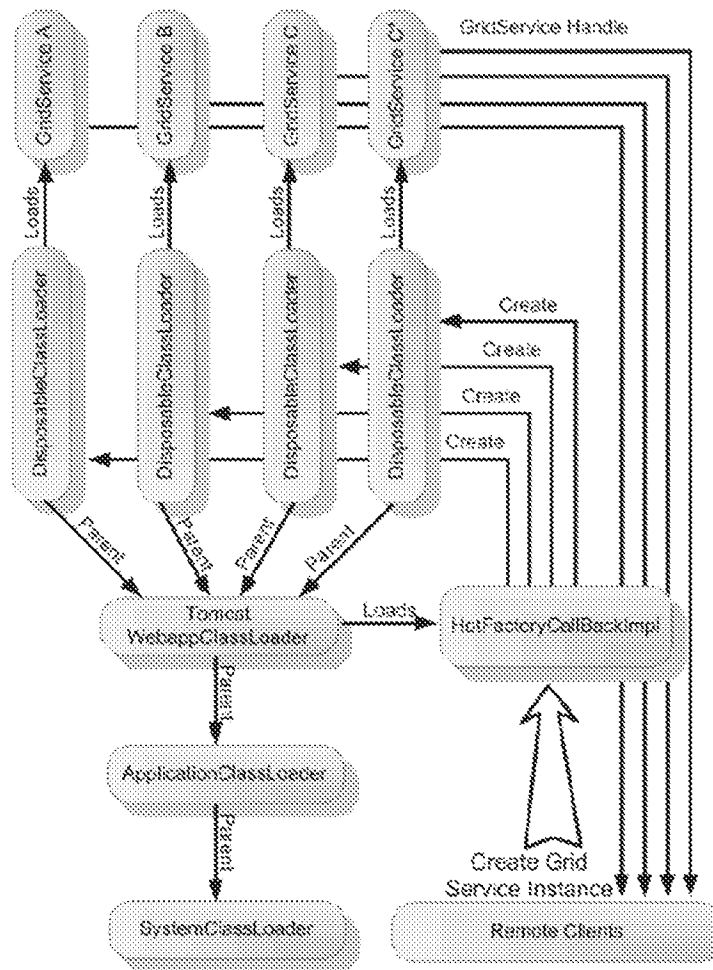


Fig. 9. ClassLoader hierarchy

Id., Fig. 9. A POSITA would have understood that Smith’s disclosure of HDS to deploy and redeploy services in a grid node, along with disclosing the need to update existing services in

“long running grid environments,” discloses or at least renders obvious deploying an updated version of a service in Smith’s grid system. Ex. C ¶¶73-74.

Smith’s grid “service” likewise discloses or at least renders obvious an “application release bundle.” Ex. C ¶75. Smith discloses deploying Java classes for implementing services in the Java Runtime Environment. PA-1, 4 (requiring nodes to have the “Java Runtime Environment”); *id.* (“Client C2 wants to deploy grid service B which requires the Java runtime environment 1.4”); *id.*, 6 (disclosing HDS to deploy update class definitions “as the Java ClassLoader” using a “disposable ClassLoader” that is “capable of loading the new class definitions and integrating them into the running system.”). Smith’s deployment therefore includes Java components for the Java Runtime Environment, which is consistent with the ’332’s “application release bundle,” which includes “a set of *components* that are hosted by one or more application servers 260 in the one or more grid nodes 235.” Ex. A, 1:28-45 (emphasis added); *see also id.* (disclosing that “J2EE [(e.g., Java 2 Platform, Enterprise Edition or J2EE)] based applications are built as a set of *components*”) (emphasis added), 3:17-19 (“application includes a set of *components* that are hosted by one or more application servers in the one or more grid nodes”) (emphasis added), 4:54-56 (“The application release bundle includes, for example, a *set of components* that are hosted by one or more application servers 260 in the one or more grid nodes 235.”) (emphasis added).

Smith’s exemplary “deployment algorithm” and disclosed pseudocode for implementing Smith’s HDS is also consistent with the ’332 patent’s deployment of an “application release bundle.” Ex. C ¶76. As shown below, the HDS algorithm (1) registers a service description with the AXIS/OGSA request handlers; (2) makes schema files available to the OGSA environment; and (3) makes the service class files available to the class loader. PA-1, 6. Smith’s “service.gar”

file, or grid archive file, as shown in the exemplary pseudocode below, packages the service (including the Web Service Deployment Descriptor (WSDD), service classes, and schemas). The final step copies compiled service code (as a “jar” or Java archive file) to the container’s classpath or library location so the service can be instantiated and run:

```
with service.gar do {  
  with wsdd_document do {  
    root = registry.getRootNode()  
    service_descriptor =  
      wsdd_document.getServiceEntry()  
    foreach property of service_descriptor  
      do { serviceOptions.put(property) }  
    serviceEntry =  
      createServiceEntry(serviceOptions)  
    root.bind(service_descriptor.name,  
      serviceEntry)  
    axisEngine.deploy(wsdd_document) }  
  copy schema to schema_location  
  copy jar to jar_location.service }
```

Fig. 8. Deployment algorithm

Id., Fig. 8.

Smith’s deployment of components via a Jar file is consistent with the ’332 patent, which discloses deploying an application release bundle that may include “*application archive code*, such as zip, *jar*, tar, tgz, exe and the like.” Ex. A, 2:53-55 (emphasis added). For example, Smith’s “service.gar” file satisfies this feature because, just like the patent’s jar and tar files, Smith’s gar file is a “grid archive” file that contains components to be deployed on a grid node. Ex. C ¶77. It also includes a “jar” (e.g., Java Archive) file that is copied to a location in the service container on the grid node. PA-1, 6 (“the OGSA application needs to be enabled to find and load the new classes that make up the service (copy jar to jar location.service).”). Smith’s deployment of grid “services” therefore discloses deploying an “application release bundle”

(e.g., as Java classes within the Java Runtime Environment), consistent with the '332 patent.

Ex. C ¶77. A POSITA would have further understood deploying a “service” via Java components is synonymous with deploying an “application release bundle” in network computing, or at the very least renders this feature obvious. *Id.*

Adding . . . in a repository server: software release repositories were well known in the art as of the priority date of the '332 patent, including adding new versions of Java classes to the same. Ex. C ¶¶78-84; PA-4, Abstract (“A new or updated software object class is received at a repository”), 6:31-34 (business object class “added to and/or updated in the repository”), 8:22-30 (same); 9:39-10:11 (updating “business object classes deployed from a repository”); PA-5 ¶¶17-18 (deploying enterprise Java beans (EJBs) and obtaining information about a server from a repository); PA-3 ¶24 (storing metadata “by the deployment repository [that] includes information (e.g., a timestamp or version indication)”).

Given this knowledge, a POSITA would have found it obvious to modify Smith to add new versions of its services and Java classes for deployment “in a repository server,” as taught or at least suggested by Fanshier. Ex. C ¶¶78-84.

Fanshier discloses one known method of adding a new version of an application for deployment “in a repository server.” Like Smith, Fanshier discloses a system for deploying an application in a distributed computing environment. PA-2, Abstract (disclosing “deploy[ing] an application across a network”), ¶7 (providing for “the deployment of applications or application modules, such as across a network or to a server cluster.”). A deployment tool 100, such as a standalone application or an administration console, “can contact a master deployer 104 in an administration server 102 and point the master deployer to the *source 124* of an application archive file or directory.” *Id.* ¶13 (emphasis added). Fanshier’s application source 124 stores an

application for deployment, where the deployment may be “applications or application modules,” including Java components. *Id.* ¶¶8, 16, 57. The master deployer 104 then sends the path of the deployment on the application source 124 to slave deployers 108, 116 on target servers across network 106, and slave deployers 108, 116 “pull” or “retrieve” the application from the application source. *Id.* ¶¶8, 13. This includes downloading an “archive file or application directory.” *Id.* ¶8. Fanshier discloses that the source locations “can include . . . a file on an administration *server*.” *Id.* ¶19 (emphasis added).

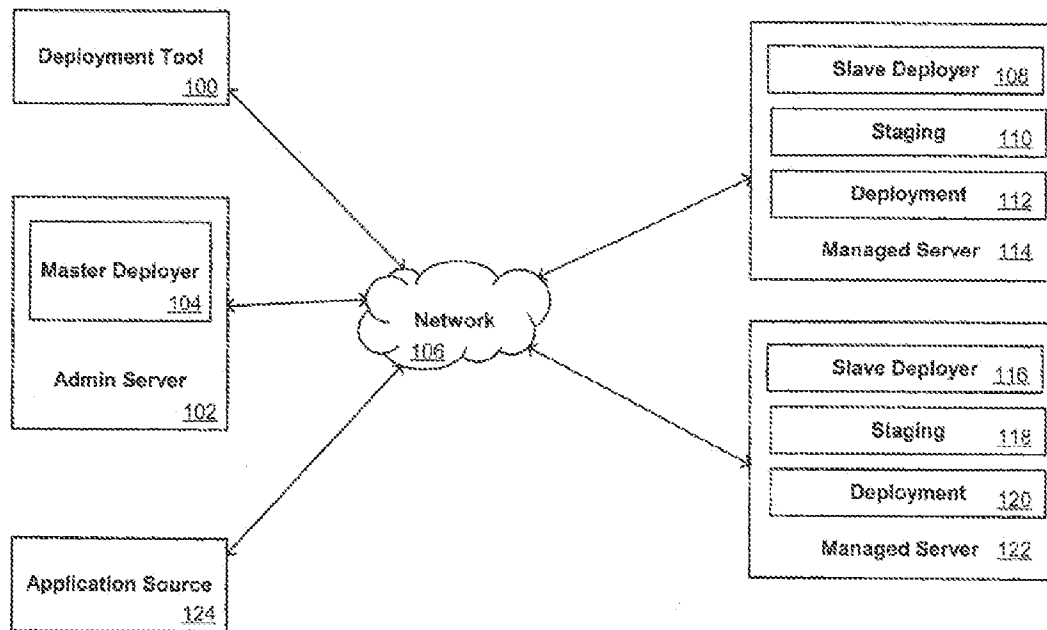


Figure 1

Id., Fig. 1.

Fanshier’s deployment tool 100 contacting a master deployer to deploy application components from an “application source,” which Fanshier discloses may be an administration “server,” discloses or at least renders obvious adding an application release bundle for deployment to a repository server.

A POSITA would have understood that, by notifying a master deployer of the path to an application for deployment on the application source, Fanshier teaches or at least suggests “adding” the application. Ex. C ¶¶82-84. For the application to be in a file or directory on the application source, it must be added to the application source so the master deployer can access it and deploy it. *Id.* Indeed, “adding” a new version of an application to a repository like Fanshier’s was not new, and a POSITA would have understood Fanshier teaches or renders this known feature obvious. *See, e.g.*, PA-4, 6:31-34 (“In some embodiments, the global change record comprises a timestamp, version stamp, flag, and/or other data indicating that at least one business object class has been *added* to and/or updated in the repository.”) (emphasis added).

Fanshier’s application source is a repository “server”: “[s]ource locations for an application can include a local directory or a file *on an administration server*.” PA-2, ¶19 (emphasis added). Regardless, it would have been obvious to implement Fanshier’s application source as a server based on Fanshier’s disclosure that its application source is a server, and based on a POSITA’s general knowledge that such repositories were commonly hosted on servers. Ex. C ¶83; PA-4, 3:9-35 (“content system 110 . . . store[s] and/or manage[s] one or more logical repositories” and may comprise “a single content server.”), 8:38-9:8; PA-3, Fig. 1 (deployment repository 104 on server 108), ¶10.

As explained above in above in Section VIII.A.4, a POSITA would have been motivated to combine Smith and Fanshier with a reasonable expectation of success to achieve this claim feature. *See* Section VIII.A.4. Smith and Fanshier therefore render obvious [1a]. Ex. C ¶¶72-84.

- c. [1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

The combination of Smith and Fanshier discloses “*determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.*” Ex. C ¶¶85-100. For example, Smith discloses a discovery services module that can evaluate grid nodes for an application associated with the added new version of the application release bundle. *Id.* It also would have been obvious to employ Smith’s discovery services module to determine which grid nodes have an associated application when a new version is added to a repository server, as taught by Fanshier. *Id.*

Determining by a discovery services module which grid node is running an application associated with the added version of the application release bundle: Smith discloses a discovery services module that determines nodes available to run a service on a grid. PA-1, 3-5. For example, Smith discloses that the network topology of its grid is dynamic and “thus a node detection solution geared towards frequent node arrivals and departures is required.” *Id.*, 2. To accomplish node detection, Smith discloses an “automatic discovery mechanism” that discovers nodes in the grid. *Id.*, 3 (disclosing understanding “fluctuating topology and large number of participants.”); *see also id.*, Abstract (employing “peer-to-peer based node discovery”), 1 (“[o]ur solutions to these problems are based on peer-to-peer node discovery”)

To determine nodes in its grid, Smith discloses the use of the JXTA open source, peer-to-peer protocol. *Id.*, 3. As was known at the time, JXTA formed as an overlay network using

UDP, TCP, and HTTP as its transport protocols. Ex. C ¶¶86-87; PA-8, 10-11 (“JXTA technology is designed to be also independent of transport protocols. It can be implemented on top of TCP/IP, HTTP, Bluetooth, HomePNA, and many other protocols.”); PA-9, 88-95 (same). JXTA as a discovery mechanism enabled “[p]eers within th[e] system [to] form peer groups of cooperating nodes, allowing node discovery scoped by group membership.” *Id.*, 4.

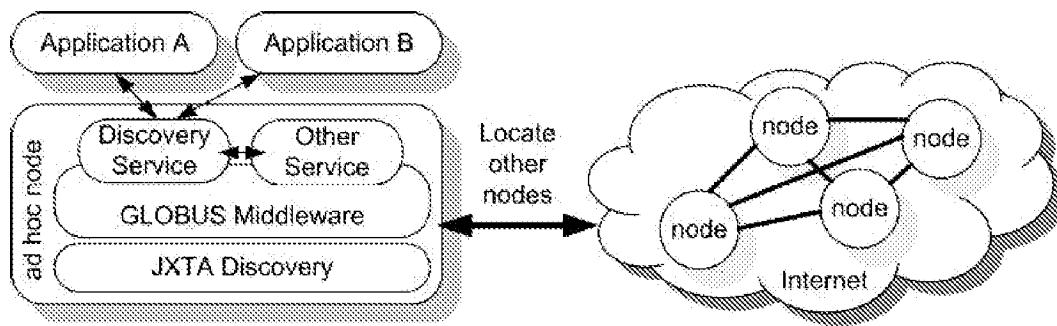


Fig. 2. JXTA-based discovery components.

PA-1, Fig. 2.

Once a node is discovered, Smith discloses determining the properties of the node: “[i]n an ad hoc grid environment, where nodes enter and leave the grid on a regular basis, . . . it is no longer given that all nodes are capable or well suited to run a certain service.” *Id.*, 4. Using a “property scout,” Smith discloses analyzing the “properties of the nodes within the grid . . . [to] be matched to the requirements of the grid service.” *Id.* The property scouts may include “custom analyzers” created by the developers of a grid service to “perform some specific operations and check the system performance [of a node] using actual operational code.” *Id.*

Smith’s property scouts allowed the system to understand “[i]nformation about the underlying hardware and operating system” of each node, which “is particularly important when deploying legacy code, because tight integration of system resources is a common occurrence there.” *Id.*, 2. Smith also discloses determining “operating system type, available hardware

resources and availability of required libraries.” *Id.*; *see also id.*, 4-5 (determining “[o]perating system type, language support, processor power, available memory and up-time are properties of a node which can be taken into account when a service is to be deployed.”).

```
<check>
  <OS name=windows version=XP
    modifier=none/>
  <Language name=JRE version=1.2
    modifier=orHigher/>
  <Library name=custom.dll/>
</check>
```

Fig. 4. Set of requirements

Id., Fig. 4 (depicting an exemplary node requirements checked by a property scout).

In addition to these properties, Smith contemplates determining additional node properties by its property scouts: “[c]learly, it is not possible to anticipate all requirement types and evaluation metrics since each grid service can have unique needs specific to its task.” *Id.*, 4. “To be able to incorporate such application specific application requirements,” Smith’s property scouts are implemented as a “plug-in architecture . . . to give the property scouts the necessary flexibility to cope with an ad hoc environment and unanticipated requirements.” *Id.*, 4-5. To analyze additional properties, “[c]ustom analyzers conforming to the plug-in interface can be created by the developers of a grid service specifically tailored to their needs.” *Id.*, 5. A “requirements description file” includes necessary requirements to be analyzed in the node by the custom analyzers. *Id.*

In one embodiment, Smith discloses determining whether a grid node includes the Java Runtime Environment in which to deploy a service. *Id.*, 4 (“The grid service using this requirements block needs . . . a Java Runtime Environment version 1.2 or higher”); *id.* (“Client

C1 wants to deploy grid service A which requires the Java runtime environment version 1.3. Client C2 wants to deploy grid service B which requires the Java runtime environment 1.4”). By discovering nodes in Smith’s grid environment—which may go offline and reappear dynamically—and analyzing the grid node properties to determine if a grid node includes Java Runtime Environment, Smith discloses or at least renders obvious determining by a discovery services module which of the one or more grid nodes in the grid computing environment are running an application associated with a new version of an application release bundle, as claimed. Ex. C ¶¶87-91. The Java Runtime Environment is an “application” that is “associated with” the Java components/Classes (e.g., the “application release bundle”) that are deployed to the grid node in Smith. *Id.*

To the extent claim 1 requires determining whether preexisting versions of the deployed service are present on a grid node, Smith also renders this feature obvious. Indeed, doing this was well known in grid environments. *See, e.g.*, Ex. C ¶92; PA-19, 42 (“Every Grid service container must maintain a list of services it provides. This list of services must again be exposed to the underlying P2P infrastructure”). For example, it would have been obvious in Smith to perform this check when a new ClassLoader is ready for deployment. Ex. C ¶92. This would have been particularly useful for Smith’s redeployments, where a node may come online and require a redeployment, and thus Smith would have a need to rediscover the node and determine that it includes an existing server before hot deploying the redeployment on that node. *See* PA-1, 6 (“Each grid service is loaded into its own disposable ClassLoader and if the grid service needs to be redeployed, the old ClassLoader is disengaged from the system and a new ClassLoader is put in its place.”); Ex. C ¶92. “The *new* ClassLoader is then capable of loading the new class definitions and integrating them into the running system.” *Id.* (emphasis added).

Indeed, Smith contemplates having multiple versions of ClassLoaders on a node: “[i]t is also possible to run different versions of the same grid service on the same grid node, simply by using different ClassLoaders *for each version*.” *Id.* (emphasis added). Smith thus contemplates versioning its classes and redeploying “new ClassLoader[s]” to its grid nodes and would have benefited from determining whether an existing version of an application is running on a node during its discovery. Ex. C ¶92.

A POSITA would have been motivated to modify Smith in this way, if required, to enable the system to determine whether a particular node includes the latest version of classes for a given service upon discovery so that the correct versions are deployed across the grid. Ex. C ¶93. For example, if bugs or security issues are addressed in newer versions, a POSITA would have been motivated to deploy the most recent version and “disengage[]” prior versions, as Smith contemplates. *Id.* Smith suggests gathering such information, as was known in the art to be obtained through metadata, by disclosing, “[t]o enable autonomous deployment, *meta-information* from the grid participant must be available to the deployment service.” PA-1, 2 (emphasis added).

A POSITA would have also had a reasonable expectation of success in making this modification because checking which versions of classes are currently deployed on a computer in a distributed computing environment was well-known and understood in the art. Ex. C ¶94; PA-19, 42 (“Every Grid service container must maintain a list of services it provides.”); PA-10, 4 (using metadata in conjunction with Open Grid Services Architecture to enable “services [to] be discovered not only by service names but by a variety of other metadata information.”); PA-11, 8 (disclosing “enquiry mechanisms that permit discovery of their structure, state, and capabilities,” including querying “[c]ode repositories” on a node to “for managing versioned

source and object code.”); PA-13, 4-5 (using an “extensible resource discovery mechanism” and a “configuration file” on the grid node to search for nodes running a particular service); PA-5 ¶18 (“retrieved information [from the application server] may include applications 110, for example, that are already deployed”); PA-7 ¶25 (“[d]eployer 110 may be configured to automatically notice that an update is available . . . [by] periodically poll[ing] the applications in mirror 112 or storage devices 130a, 103n, to determine if file modification times have changed.”); *see also* PA-4, 8:22-37 (“check[ing] a corresponding data in the repository, e.g., a class or collection of class specific change record and/or date value or other indicator”); PA-3 ¶¶ 12, 15 (“the resources may also have time stamp or version information (107) that can be inspected before downloading the resource(s) so that only new or updated resource(s) are downloaded”). Given this knowledge, a POSITA would have had a reasonable expectation of success, for example, by employing Smith’s property scouts to also determine whether existing classes are deployed in grid nodes. Ex. C ¶94. Indeed, Smith discloses the property scouts are customizable and extensible to identify additional node properties. PA-1, 4-5.

Upon adding to the repository server: Smith discloses that, when deploying a service, its system checks grid nodes for an application (e.g., the Java Runtime Environment) associated with components/classes to be deployed (e.g., the application release bundle). PA-1, 2-6.

In the Smith-Fanshier combination, it would have been obvious to perform Smith’s node discovery and property check when the new classes (e.g., the application release bundle) are added to the repository server (as taught by Fanshier). Ex. C ¶¶95-100. This corresponds with the sequence of steps in Smith, including: “peer-to-peer based node discovery, automatic node property assessment, [and] hot deployment of services into a running system.” PA-1, Abstract. This also corresponds with the sequence in Fanshier: “[a] deployment tool 100 . . . contact[s]

... and point[s] [a] master deployer to the source 124 of an application archive file or directory” before deployment. PA-2, ¶13. A POSITA thus would have understood and expected the steps of node discovery, property assessment, and hot deployment to be preceded by the application version being made available on a repository server in the combined system. Ex. C ¶96.

While the claims and the written description of the ’332 patent do not define “upon” or place a temporal restriction on when the deployment must initiate following it being “added” to the repository, Fanshier renders this obvious. *Id.* ¶97. For example, Fanshier discloses that its deployment tool 100 contacts a master deployer 104 to point the master deployer to the application source 124, which triggers a new deployment. PA-2, ¶13; *see also id.* ¶7 (“A master deployer on an administration server can be contacted, such as by a deployment tool, indicating that an application is to be deployed.”). A POSITA would have found it obvious to utilize Fanshier’s deployment tool to initiate Smith’s discovery process when a new version of an application release bundle (e.g., application component or new class) is added and marked for deployment in the repository, and for this to initiate the node discovery process. Ex. C ¶97.

Initiating discovery in this way would have been nothing more than combining known elements in the prior art to achieve predictable results. Ex. C ¶98. A POSITA would have had a reasonable expectation of success in the combination given the similarities between Smith and Fanshier and the ubiquity of deployment tools for initiating deployments. *See, e.g.*, PA-7 ¶21 (“deployer 110 checks for updates”). Indeed, a POSITA would have known automatically pushing updates to available computers in a network was known and understood in the art. For example, other prior art discloses that “[a]utomatic upgrade utilities are known in the prior art” and discloses methods of doing so “from [a] master software upgrade server.” *See, e.g.*, PA-20, ¶¶37-42 (“The Product Supplier or Software Vendor publishes a software update, i.e., patch,

upgrade or new release,” which are “either be pushed down to the software upgrade server or they can be pulled from the published location” so that it can be distributed “from [a] master software upgrade server.”).

It would have been obvious to modify Smith in this way to discover new nodes in the grid for a service when a new version of the service is made available, thereby improving dissemination of the service. Ex. C ¶¶99. Likewise, it would have been obvious to determine which nodes already have the service or application for the service when an update is made available to identify those nodes available to receive the update. *Id.* Doing so would have been nothing more than combining known features in the art according to understood principles to achieve predictable results. *Id.* And a POSITA would have utilized Fanshier’s deployment tool, for example, to trigger discovery in order to control the release of new software versions. *Id.*

As explained above in Section VIII.A.4, a POSITA would have been motivated to combine Smith and Fanshier with a reasonable expectation of success to achieve this claim feature. *See* Section VIII.A.4. Smith and Fanshier therefore render obvious [1b]. Ex. C ¶¶85-100.

- d. [1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

The combination of Smith and Fanshier discloses “*notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use.*”

Ex. C ¶¶101-118. For example, Smith’s OGSA application on a grid node is a “client application manager.” *Id.* Fanshier similarly discloses a slave deployer that is a “client application manager” and notifying the slave deployer of a deployment. *Id.* In the combined

system, it would have been obvious to provide Smith's grid nodes with a notification that includes "a type of data transfer protocol to use" based on Smith's JXTA or GT3, or alternatively based on Fanshier's disclosure of distributing applications from a file distribution servlet. *Id.*

Notifying a client application manager associated with a grid node about adding the new version of the application release bundle: Smith discloses deploying a service using a "hot deployment service (HDS)" using an "OGSA application" on a grid node. PA-1, 5-6. To deploy the service, Smith discloses pseudocode in Figure 8 associated with its HDS:

```
with service.gar do {  
  with wsdd_document do {  
    root = registry.getRootNode()  
    service_descriptor =  
      wsdd_document.getServiceEntry()  
    foreach property of service_descriptor  
      do { serviceOptions.put(property) }  
    serviceEntry =  
      createServiceEntry(serviceOptions)  
    root.bind(service_descriptor.name,  
      serviceEntry)  
    axisEngine.deploy(wsdd_document) }  
  copy schema to schema_location  
  copy jar to jar_location.service }
```

Fig. 8. Deployment algorithm

Id., Fig. 8.

Smith's accompanying disclosure explains that the HDS "undertakes the following steps to deploy a grid service into a running GT3 environment": it (1) "[r]egister[s] the service description with the AXIS/OGSA request handlers"; (2) "[m]ake[s] the schema files available to the OGSA environment"; and (3) "[m]ake[s] the service class files available to the class loader."

Id., 6. "To register the newly deployed service with *the OGSA application*, the HDS needs to dynamically update the in-memory service registry of *the OGSA application* (the code within

with wsdd document do)." *Id.* (emphases added). Schema files are copied and then "*the OSGA application* needs to be enabled to find and load the new classes that make up the service." *Id.* (emphasis added).

A POSITA would have understood that Smith's "OSGA application" on each grid node discloses a "client application manager associated with one or more of the determined grid nodes." Ex. C ¶¶102-104. A POSITA also would have understood that, to deploy Smith's service on a grid node via the OSGA application, the OSGA application would be notified of the deployment via the HDS, which includes a WSDD (Web Service Deployment Descriptor) that provides details about the service to be deployed, such as the class implementing the service, and instructions to deploy and register the service. PA-1, Fig. 8; Ex. C ¶¶102-104. Smith's particular implementation of the OSGA application may include the Globus Toolkit 3.0 (GT3) deployed as an Apache Axis engine within Tomcat. PA-1, 3.

While Smith does not expressly disclose the details of its grid nodes being "notif[ied]" of a deployment or how the components of the deployment are transferred to the grid nodes, this feature would have been obvious in view of Fanshier. Ex. C ¶¶105.

For example, Fanshier discloses that a master deployer can "*contact* a slave deployer" on a server "that is to deploy the application, indicating that the slave deployer should prepare to deploy the application." PA-2 ¶7 (emphasis add). The master deployer provides the source of the deployment (e.g., the repository server from which to retrieve the components to be deployed) and "*send[s]* this path to slave deployers 108, 116." *Id.* ¶13 (emphasis added); *see also id.*, ¶14 (disclosing that its master deployer "inform[s] the target deployers, which can download the application to a staging area"), ¶17 ("All members of the target can be *notified* of the deployment and can take the appropriate action.") (emphasis added), ¶39 ("A deployment

notification can be broadcast for certain events, which can be defined in a deployment notification class.”) (emphasis added); claim 16 (“*contacting* a slave deployer . . . by the administration server to indicate that an application is to be deployed on that managed server”) (emphasis added). The slave deployer then retrieves the application components to be deployed from an application source. *Id.* ¶8 (slave deployer then “pull[ing] the application”), ¶13 (“pull[ing] the file”) ¶14 (“download[ing] the application”), ¶23 (same).

Fanshier discloses a specific embodiment for its slave deployer in an “Application MBean,” which registers to listen for deployment tasks from an administrator server. *Id.* ¶¶39-46 (disclosing, for example, “MyNotificationListener class” to “implement a notification listener interface and perform any processing required when the notification is received”).

Fanshier’s slave deployer is a “client application manager,” consistent with the ’332 patent. The ’332 patent states that “[t]he term client refers to nodes running application servers on each grid node,” Ex. A, 2.65-66, and that the client application manager can “register[] with the repository server and can accept notifications from a server using grid notification mechanisms.” *Id.* This is exactly what Fanshier discloses with respect to its slave deployer—a service that runs on a target server that receives notifications regarding a deployment and retrieves the necessary components for the deployment. Ex. C ¶¶105-108. A POSITA therefore would have understood Fanshier’s slave deployer and/or Application MBean is a client application manager that is “notif[ied]” of a new deployment. *Id.* In the Smith combinations, the system would have included notified grid nodes of a “new version of the application release bundle,” as discussed in Sections [1a] and [1b]. A POSITA would have combined Fanshier’s notifications to its slave deployers—implemented as Smith’s OGSA application—for the reasons explained in Section VIII.A.4.

Along with a type of data transfer protocol to use: The Smith-Fanshler combination renders this feature obvious in four alternative ways. Ex. C ¶¶109-118.

First, Smith renders this feature obvious by disclosing the use of JXTA to communicate with grid nodes for discovery and formation of peer groups. PA-1, 3-4; Ex. C ¶110. Smith's JXTA peer group, as was known in the art, is a "virtual entity that speaks the set of peer group protocols" and "provid[es] a common set of services." PA-9, 91. JXTA supports "pipes" between peers—"communication channels for sending and receiving messages." *Id.* This may include "[a]ny number of unicast and multicast protocols and algorithms, and their combinations," where "JXTA runs on top of [existing] transports." *Id.*, 92. JXTA itself is a set of protocols "defined by one or more messages exchanged among participants in the protocol; each message has a predefined format, and may include various data fields." *Id.*, 90 ("In this regard, it is akin to TCP/IP. TCP/IP links Internet nodes together, while JXTA technology connects peer nodes with each other. TCP/IP is platform-independent by virtue of being a set of protocols. So is JXTA. Moreover, JXTA technology is transport-independent and can utilize TCP/IP as well as other transport standards."). JXTA supports "TCP/IP, HTTP, Bluetooth, Home-PNA, and many other protocols." *Id.*, 95.

It would have been obvious and understood that, during the discovery process and Smith's creation of peer groups, pipes between peers would be defined based on a JXTA communication protocol available between peers. Ex. C ¶111. After a peer is discovered and a deployment is initiated in Smith, it would have been obvious for the peer to be notified of the deployment using the same pipe—whether using a JXTA protocol or one or more of "TCP/IP, HTTP, . . . [or] other protocols" supported by JXTA. PA-9, 95; Ex. C ¶111. For example, the location from which the client application manager can pull, download, or otherwise retrieve the

components of the deployment in the Smith-Fanshier combination would be provided to the client application manager via a notification (as discussed above in Fanshier), via JXTA or TCP/IP, with or without HTTP. Ex. C ¶111; PA-9, 90, 95. This is particularly so in the “heterogeneous” grid system of Smith, where different nodes may be different systems and thus require particular transport protocols suitable between peers. Ex. C ¶111; PA-1, 2 (discussing challenges in a heterogeneous environment); PA-9, 92 (“For example, when two peers communicate with each other and both have TCP/IP support, then an implementation can easily create bidirectional pipes.”).

Second, it alternatively would have been obvious to notify a grid node of a deployment and transfer necessary components for the deployment using the tools available in GT3, which implements the teachings of Smith. Ex. C ¶112. Smith discloses that its system is built using OGSA and OGSI, which “define virtualization of available resources at the system-independent level of *resource access* to allow uniform access to a heterogeneous system.” PA-1, 2 (emphasis added). Using this, Smith discloses that “[t]he important parts of a proof-of-concept implementation [of its system is] based on the Globus Toolkit 3.0 (GT3).” *Id.* GT3 is “the most widely used implementation of the Open Grid Services Architecture.” *Id.*, 3; *see also id.*, 1. GT3 is an open-source software framework designed to support the development of grid computing environments. PA-13, 1-2 (“Globus Toolkit [5] emerged in 1997 as an open source project and quickly became a *de facto* standard for grid computing infrastructure.”); PA-11, 10-11 (describing the Globus Toolkit and its protocols); PA-12, 24-25 (“The Grid’s success to date owes much to the relatively early emergence of clean architectural principles, *de facto* standard software (in particular, the Globus Toolkit[])”); PA-16, 5-7 (“GT3 is an implementation of the Open Grid Services Infrastructure (OGSI) Version 1.0”); PA-6, 14-16

(“The Globus Toolkit is a software toolkit, developed by The Globus Alliance . . . which we can use to program grid-based applications.”). GT3 provides core services such as security, resource management, and data transfer, and allows for dynamic discovery and invocation of distributed services, making it easier to build and manage interoperable grid applications across heterogeneous systems. *See, e.g., id.* Using OGSA and GT3, Smith disclose that “the underlying system of each node is guaranteed to support the grid service interface.” PA-1, 3.

GT3 implemented “GridFTP” as a file transfer protocol. PA-11, 10 (Globus Toolkit including “[a]n extended version of the File Transfer Protocol, GridFTP”); PA-13, 5 (“After creation, the GAR file and the *PersonalizedServices.xml* file are sent to all slave machines by the GRIDFTP mechanism in a safe channel.”); PA-14, 2 (“GT3 supports end-user deployment of *applications* using mechanisms including: . . . data transfer services (Grid File Transfer Protocol – GridFTP”); PA-15, 1-2 (“GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks that is based on FTP and developed as part of the Globus Project.”), 4 (“as obvious GridFTP is just one way of performing file transfers in Grids”); PA-16, 20-21 (GridFTP “provides support to transfer files among machines in the Grid and for the management of these transfers.”), *see also* PA-19, 67 (using GridFTP to transfer a deployment in GT4), 142-143 (“RFT as the Grid service-oriented control interface for other underlying data transport protocols such as GridFTP”). GridFTP is a high-performance, secure, and reliable extension of the standard file transfer protocol (FTP). *See, e.g., id.* GridFTP was used to transfer, among other things, files supporting the deployment of services across a grid. *See, e.g., id.*; PA-13, 4-5 (transferring GAR file that is “sent to all slave machines by the GRIDFTP mechanism in a safe channel”), Fig. 7. This includes transferring the necessary service files (e.g., WSDL, deployment descriptors, JARs) to a target node or system. *Id.*

It would have been obvious in view of Smith's use of GT3 in the Smith-Fanshier to notify a node of a deployment and direct the node to retrieve the components of the deployment using GT3's GridFTP. Ex. C ¶¶112-114. GridFTP was optimized for large data transfers in grid environment and utilized parallel streams to transfer data. *Id.* Application bundles— particularly those with accompanying input data files—would likely be large and would therefore benefit by the use of GridFTP. *Id.* A POSITA therefore would have been motivated to utilize GridFTP as the transfer protocol to use, for example, for deployments involving large files. *Id.* A POSITA also would have been motivated to utilize functionality built into GT3, which Smith expressly discloses is used to “implement” its system. *See* PA-1, 1-2.

Third, Fanshier's slave deployer “retriev[ing],” “download[ing],” or “pul[ling]” the application from a source alternatively renders this feature obvious. For example, Fanshier discloses providing “[a] file or directory path can be used to identify an application” for deployment, and “[w]hen using a deployment tool, the contents of that file or directory can be made available to an administrator by copying or transferring, such as by using a *file distribution servlet*.” PA-2 ¶15 (emphasis added). While Fanshier does not expressly disclose that its notification to a slave deployer includes use of its distribution servlet, Fanshier's disclosure of doing so between its deployment tool and master deployer, and its disclosure of its slave deployer being notified and retrieving the deployment, renders this obvious. Ex. C ¶¶115-118.

As was known in the art, a file distribution servlet is designed to deliver files to clients over a network that typically involves receiving requests from a client, identifying a desired file, and sending it back as part of the HTTP response. PA-18, 6-15. A POSITA would have implemented Smith-Fanshier to apply Fanshier's file distribution servlet communication to

notify grid nodes of a deployment and to utilize the same protocol. Ex. C ¶¶115-118. For a slave deployer on a server to “retrieve,” “download,” or “pull” an application, it must know the source location and how to retrieve it. *Id.* Fanshier’s file distribution servlet is one such way, which receives requests from clients, including other servers, via HTTP. *Id.* To the extent Fanshier does not teach providing this specific protocol with its notification to the slave deployer, it would have been obvious to do so to enable the slave deployer to understand how to retrieve or download the necessary components for the deployment. *Id.* This would have been accomplished, for example, using a URL that includes an HTTP protocol that indicates where the slave deployer can access the files associated with the deployment. *Id.*

This would have involved nothing more than the simple substitution of one known element for another to obtain predictable results. *Id.* ¶¶117-118. A POSITA would have been motivated to apply Fanshier’s file distribution servlet in this manner to utilize a known and available distribution mechanism across the system in the Smith combinations (e.g., between an administrator and master deployer, and between a master deployer and slave deployer). *Id.* A POSITA would have had an expectation of success in the combination because protocols such as HTTP were well-known and utilized to transfer files in distributed computing environments, including between grid nodes, and Fanshier suggests using it to transfer deployments. *Id.*

As explained above in Section VIII.A.4, a POSITA would have been motivated to combine Smith and Fanshier with a reasonable expectation of success to achieve this claim feature. *See* Section VIII.A.4. Smith and Fanshier therefore render obvious [1c]. Ex. C ¶¶101-118.

- e. [1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

The combination of Smith and Fanshier discloses “*hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.*” Ex. C ¶¶119-144. For example, Smith’s hot deployment service (HDS) discloses this limitation and Fanshier’s MBeans for deployment alternatively disclose “an appropriate hot deployment plug-in.” *Id.* Fanshier also alternatively discloses a “one or more application servers in an associated grid node.” *Id.*

Hot deploying/redeploying the new version of the application release bundle . . . in an associated grid node . . . by a respective one of the client application managers: As discussed above, Smith discloses “hot deployment of services into a running system.” PA-1, Abstract. This enables “hot deployment of services into a running system without disrupting other services already running there.” *Id.*, 1. Smith’s hot deployment service (HDS) “allow[s] remote deployment of grid services without requiring the developer to have an administrative account on the [grid node].” *Id.*, 5-6. This means the deployment does not require a “restart of the OGSA platform after deployment.” *Id.* Smith discloses a particular implementation that enables hot deployment, including using a “disposable” Java ClassLoader in which the “new ClassLoader is [] capable of loading the new class definitions and integrating them into the running system.” *Id.*, 6. A “FactoryCallback implementation” within the Globus Toolkit 3.0 loads the service classes and enables hot-swapping class definitions on the node. *Id.*

Smith's hot deployment via its HDS discloses hot deploying/redeploying a new version of an application release bundle in an associated grid node. Ex. C ¶¶120-122. Smith discloses deploying its service using its HDS using the OSGA application on the grid node, which Smith discloses may be implemented via GT3 as an Axis service in Tomcat. PA-1, 3, 5-6, Fig. 8. As discussed in Limitation [1c], this teaches a "client application manager." Smith's hot deployment of "services" also discloses or at least renders obvious an "application release bundle." *See* Limitation [1b]. So does Fanshier. *See* Limitation [1b].

Smith therefore discloses hot deploying/redeploying the new version of the application release bundle in an associated grid node by a respective one of the client application managers."

On running one or more application servers: Smith discloses or at least renders obvious hot deployment on "one or more application servers in an associated grid node," consistent with the '332 patent. Ex. C ¶¶123-131.

For example, the '332 patent states that "[t]he term[] 'application server' and 'application container' are used interchangeably throughout the document." Ex. A, 2:25-26. The hot deployed components are hosted in a "container or a server," for example in the Java 2 Platform, Enterprise Edition (J2EE). *See id.*, 1:27-45; *see also id.*, 2:66-3:1 ("The grid nodes include one or more application servers that include components, such as Web containers, Web service containers, and EJB container."), 6:53-57 ("allow[ing] the distributed applications with diverse application containers in a grid environment to be hot deployed and/or redeployed without having to temporarily suspend and/or reinitialize to reduce overhead on the grid system."). This description is consistent with a POSITA's understanding of an application server, which is a software framework providing a runtime environment for executing

applications and managing their resources. Ex. C ¶124; PA-4, 5:60-6:6 (“An application server, such as application server 108, downloads needed business object classes from the repository, and instantiates a required business object using the locally stored class definition down loaded from the repository.”); PA-7 ¶15 (“Application server 120 may comprise any suitable server or other mechanism for supplying files or services. For example, application server 120 may comprise a Java™ to platform enterprise edition (J2EE) web server, an IBM WebSphere™, BEA iPlanet™, IBM Domino™, or other suitable server.”).

Like the '332 patent, Smith discloses hot deploying new classes within a Java Runtime Environment, which a POSITA would have understood are hosted within a Java container that is or is hosted on an “application server.” Ex. C ¶125; PA-1, 4 (requiring grid nodes to have “a Java Runtime Environment version 1.2 or higher); *id.* (“Client C1 wants to deploy grid service A which requires the Java runtime environment version 1.3.”); *id.*, Fig. 5 (listing requirements including “JRE 1.4”).

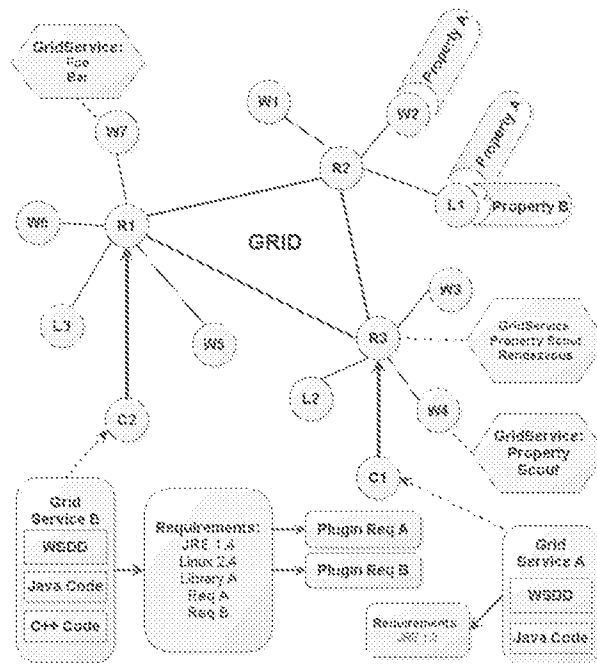


Fig. 5. Peer-to-peer based ad hoc grid with property scouts

Id., Fig. 5.

To the extent claim 1 requires a separate “server” within a grid node, Smith discloses or at least renders this obvious. Ex. C ¶126. For example, Smith disclosing applying its hot deployment methods to web servers, stating that it would otherwise be “undesirable . . . to restart an entire web server every time a web page is added, but currently this is common practice in service-oriented grid environments.” *Id.*, 6. Smith also discloses that its system is built on OGSA using GT3, which deploys services inside a web application container. *Id.*, 7 (describing its solution as curing the problem of deployment of a grid sever that would otherwise require “restarting the OGSA web application inside the web application container”).

Smith also contemplates multiple machines or servers in each grid node, satisfying “one or more application servers”:

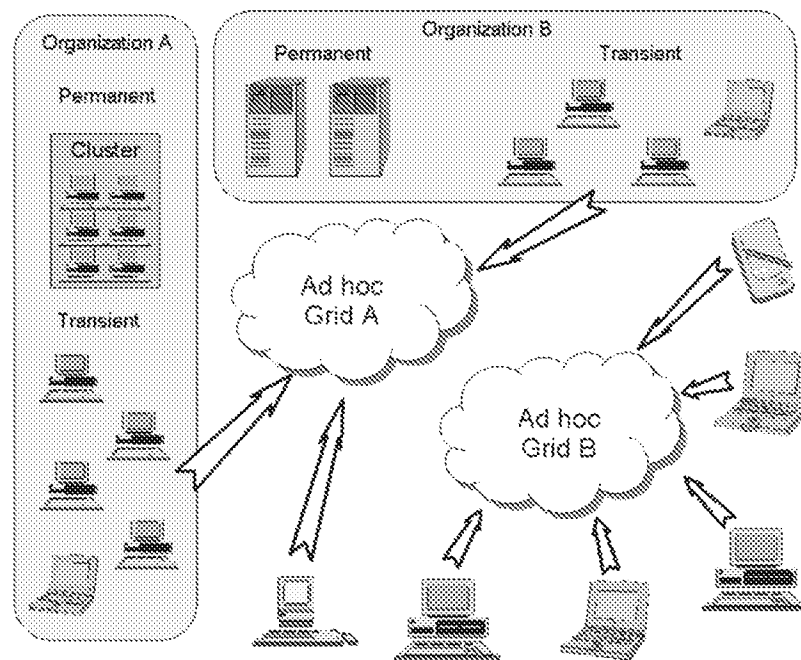


Fig. 1. Ad hoc grid architecture overview

Id., Fig. 1 (depicting Organization A as having a server cluster and Organization B as having “Permanent” servers); Ex. C ¶127.

To the extent Smith does not satisfy an “application server” alone (it does), Smith in view of Fanshier renders it obvious. Ex. C ¶¶128-131.

As discussed, Fanshier discloses deploying applications to target servers. PA-2 ¶14. The “master deployer can contact a slave deployer on each managed server that is to deploy the application, indicating that the slave deployer should prepare to deploy the application.” *Id.* ¶7. Like the ’332 patent, Fanshier discloses deploying applications that “can cover all standard J2EE deployable modules, such as enterprise applications, Enterprise JavaBeans (EJBs), web applications, and connectors.” *Id.* ¶16; *see also id.* ¶57. “A deployable unit can be utilized according to the J2EE Specification.” *Id.* ¶16. In one embodiment, MBeans can be utilized to manage the deployment on the target server, including an Application MBean that can cause “all relevant MBeans being initialized and the application being downloaded to the managed server,” where “[t]he relevant *containers* can also be notified of the redeployment.” *Id.* ¶¶48-51 (emphasis added).

A POSITA would have understood Fanshier’s J2EE application deployment to include target servers with containers, and thus Fanshier discloses deploying applications to an application “server.” Ex. C ¶¶128-130. Fanshier’s target server is an application server because it includes provides a runtime environment for executing applications and managing their resources, including application “containers” for running those applications. *Id.* This is also consistent with the ’332 patent, which states that an “application server” and “container” are interchangeable. Ex. A, 2:25-26.

It would have been obvious in the Smith combinations to hot deploy or redeploy a new version of an application release bundle in an application server in the manner disclosed by Smith, for example by updating classes within a container using disposable ClassLoaders to

deploy the new service without restarting the machine or the container. Ex. C ¶131. It would have been obvious to implement this within a server that runs an application and that has an application container using the Java Runtime Environment, which both Smith and Fanshier contemplate for applications and services. *Id.* The new classes would be loaded using Smith's GT3 factories or Fanshier's slave deployers. *Id.*

Hot deploying/redeploying . . . using an appropriate hot deployment plug-in based on the data transfer protocol: The '332 patent discloses that the "hot deployment plug-in" and "data transfer clients" are separate modules or components on each grid node. *See* Ex. A, 3:49-51 ("[A]s shown in FIG. 2, each grid node 235 includes a client application manager 230, hot deployment plug-ins 240 *and* data transfer clients 250 . . .") (emphasis added); 4:57-6:3 ("The client application manager 230 associated with each grid node 235 hot deploys/redeploys the new version of the application release bundle in the one or more application servers 260 using the hot deployment plug-ins module 240 *and* the data transfer clients module 250.") (emphasis added).

The '332 patent also discloses hot deploying the application release bundle without specifying a transfer protocol: "[i]n some embodiments, the client application manager 230 associated with each grid node 235 selects an appropriate hot deployment plug-in needed to hot deploy/redeploy the new version of the application release bundle in the associated one of the one or more application servers 260 using the grid infrastructure 270. In these embodiments, the client application manager 230 *selects an appropriate hot deployment plug-in* using the discovery services module 280 and hot deploys/redeploys the new version of the application release bundle *via the hot deployment plug-ins module 240* on one or more running application servers 260 in each determined grid node 235." *Id.*, 4:42-53 (emphases added).

In other embodiments, “[t]he hot deployment/redeployment plug-ins can *include* protocols, such as ftp, http, gridftp, tcp, and udp,” where “the hot deployment/redeployment plug-ins are used for data transfer associated with the added new version of the application release bundle.” *Id.*, 5:4-20 (emphasis added); *see also id.* (“[I]n these embodiments, the client application manager 230 uses the data transfer clients module 250 via the hot deployment plug-ins module 240 to transfer data associated with added new version of the application release bundle.”).

Claim 1 does not recite the hot deployment plug-in “including” any protocols, as described in the specification. Instead, claim 1 recites more broadly that its “hot deploying/redeploying” is accomplished “using an appropriate hot deployment plug-in” and “based on the data transfer protocol by a respective one of the client application managers.” *Id.*, cl. 1. Claim 1 therefore encompasses hot deploying/redeploying a new version of an application release bundle using an appropriate hot deployment plug-in and using a data transfer protocol specified by a data transfer client. Ex. C ¶¶132-135. This is supported by dependent claim 3, which recites using a separate hot deployment plug-in” and “data transfer client” to hot deploy the application release bundle. *See id.*, cl. 3.

Smith’s disposable ClassLoader discloses “an appropriate hot deployment plug-in” for hot deploying or deploying a service. Ex. C ¶136. Smith discloses that, to hot deploy the service, “[e]ach grid service is loaded into its own disposable ClassLoader[,] and if the grid service needs to be redeployed, the old ClassLoader is *disengaged* from the system and a *new* ClassLoader is put in its place.” PA-1, 6 (emphases added). A “class loading mechanism” that can be “disengaged” and replaced to load new class definitions discloses a hot deployment “plug-in.” Ex. C ¶136.

While Smith does not specify how its components are transferred to a grid node, it would have been obvious to transfer the components of the deployment, including the HDS and new class definitions with the disposable ClassLoader in Smith, via one or more of JXTA (including its support for TCP/IP and HTTP), GridFTP, or HTTP via Fanshier's file distribution servlet. *See* Limitation [1c] (describing Smith and Fanshier's "data transfer protocol to use"). Each of these disclose a data transfer client, as contemplated by the '332 patent. Ex. C ¶137. The new class definitions would then be hot deployed via Smith's disposable ClassLoader, which is a "plug-in" that can be engaged or disengaged from the OGSA application. *Id.* This satisfies or at least renders obvious using an appropriate hot deployment plug-in for the deployment "based on the data transfer protocol." *Id.*

To the extent claim 1 requires a protocol-specific plug-in, Smith's GridFTP alternatively renders this limitation obvious. As discussed above, Smith discloses that its system is implemented using the Globus Toolkit 3.0 (GT3). PA-1, 1-2. As discussed in Limitation [1c], GT3 was known in the art to include GridFTP—an extension of the file transfer protocol (FTP) used for grid computing. Ex. C ¶138. It was known that GridFTP utilized a "globus_ftp_client" for managing the transfer of files to a client. *See* PA-17, 21-36; PA-21, 28-55. As explained in Limitation [1c], it would have been obvious to implement Smith-Fanshier to utilize GridFTP. *See* Limitation [1c]. Using GridFTP to transfer the deployment to the grid node via a GridFTP client on the grid node satisfies or at least renders obvious using an appropriate hot deployment plug-in (e.g., the GridFTP client) based on the data transfer protocol. Ex. C ¶138.

A POSITA would have understood, for example, that Globus's GridFTP client extends the functionality of applications running on the grid node and can be tailored by administrators to perform specific functions. *See* PA-17, 21-36 (describing plug-in architecture of GridFTP);

PA-21, 28-55 (same); *see also* PA-15, 6-7 (implementing a GridFTP user interface component “has the ability to download and install files in his user space on the client computer,” which “[i]n essence it is not different than installing a browser plug-in such as Acrobat Reader, a Microsoft program, or an MP3 player.”). The application on the grid node (e.g., a Java application) on the grid node also operates independently of the GridFTP client, which can be added or updated without changing the application, which teaches or at least suggests a “plug-in.” Ex. C ¶139.

At a minimum, it would have been obvious to implement GridFTP as a “plug-in” to achieve these benefits: to enable the management of file transfers for deployment to a grid node with an extensible plug-in, such as the GridFTP client, to decouple the applications running on the node with the components for transferring files for updating those applications. Ex. C ¶140. This would have enabled the GridFTP client to be managed, deployed, and updated separately from other applications on the node, giving administrators more flexibility and allowing them to manage the file transfer clients (and potentially install other protocol clients) without disturbing other applications on the node. *Id.*

Fanshier’s notification listener class alternatively renders this feature obvious, including a “plug-in” for managing deployment via a protocol. Ex. C ¶¶141-144.

Fanshier discloses that “[a] deployment notification can be broadcast for certain events, which can be defined in a deployment notification class.” PA-2 ¶¶39-46. “Applications can listen for these events by adding themselves as a notification listener on the application MBean.” *Id.* “The MyNotificationListener class . . . can implement a notification listener interface and perform any processing required when the notification is received.” *Id.*

Fanshier's MyNotificationListener service class can be added and removed for each target server, extending the functionality of the Application MBean, and therefore discloses a "plug-in." Ex. C ¶¶141-144. The MyNotificationListener class is a hot deployment plug-in because it manages the deployment by receiving notifications of a deployment, which triggers transferring the deployment to the target node. PA-2 ¶¶39-46. As discussed in Limitation [1c], it would have been obvious to employ an HTTP protocol for this transfer to the slave deployer for the deployment. Ex. C ¶¶141-144. It also would have been obvious to utilize the same protocol for the MyNotificationListener class to receive notifications. *Id.* Fanshier's MyNotificationListener class therefore satisfies using an appropriate hot deployment plug-in based on the data transfer protocol, as claimed.

It would have been obvious to implement the MyNotificationListener class as the hot deployment plug-in in the Smith combinations for the reasons provided in Limitation [1c] for notifying an application MBean as the client application manager in the Smith-Fanshier system. See Limitation [1c]; Ex. C ¶¶101-144.

6. Independent Claim 9

- a. **[9pre]: "A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising"**

Independent claim 9 recites a "computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform" the method of claim 1. Smith and Fanshier disclose computer systems that include computer-readable medium with program code to cause executing of their disclosed features. For example, Smith discloses the use of the Open Grid Services Architecture (OGSA) and Globus Toolkit 3.0, which are

implemented on servers and computer systems with memory and software instructions. PA-1, 1-2; Ex. C ¶¶145-148.

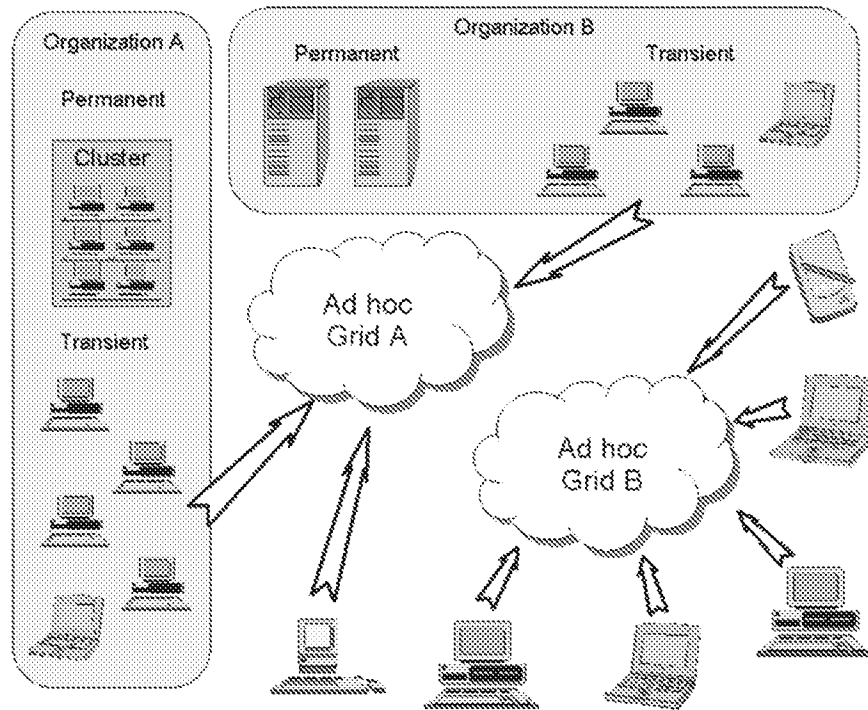


Fig. 1. Ad hoc grid architecture overview

PA-1, Fig. 1.

Fanshier similarly discloses a “computer-readable medium,” “[a] computer program product for execution by a server computer for deploying an application module,” and a computer system with a “processor” and “object code executed by said processor” for carrying out its methods. PA-2, cl. 32-34. The Smith-Fanshier combination thus discloses Limitation [9pre]. Ex. C ¶¶145-148.

The method of Limitation [9pre] is otherwise substantively identical to Limitation [1pre]:

<u>Limitation [9pre]</u>	<u>Limitation [1pre]</u>
“a method for hot deployment/redeployment in a grid computing environment comprising.”	“A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising.”

Thus, Limitation [9pre] would have been obvious for the same reasons discussed above in Limitation [1pre]. Ex. C ¶¶145-148.

- a. **[9a]: “adding a new version of an application release bundle in a repository server;”**

Limitation [9a] is substantively identical to Limitation [1a]:

<u>Limitation [9a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server.”	“adding a new version of an application release bundle in a repository server.”

Thus, Limitation [9a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶149-150.

- b. **[9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”**

Limitation [9b] is substantively identical to Limitation [1b]:

<u>Limitation [9b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.”	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.”

Thus, Limitation [9b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶151-152.

- c. [9c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Limitation [9c] is substantively identical to Limitation [1c]:

<u>Limitation [9c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [9c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶153-154.

- d. [9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Limitation [9d] is substantively identical to Limitation [1d]:

<u>Limitation [9d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [9d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶155-156.

7. **Independent Claim 14**

- a. [14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”

Independent claim 14 recites a computer system with a processing unit and memory for executing the method of claim 1. As discussed in claim 9, Smith and Fanshier disclose computer systems that include computer-readable medium with program code to cause execution of their disclosed features. For example, Smith discloses the use of the Open Grid Services Architecture (OGSA) and Globus ToolKit 3.0, which are implemented on servers and computer systems with memory and software instructions. PA-1, 1-2; Ex. C ¶¶157-159.

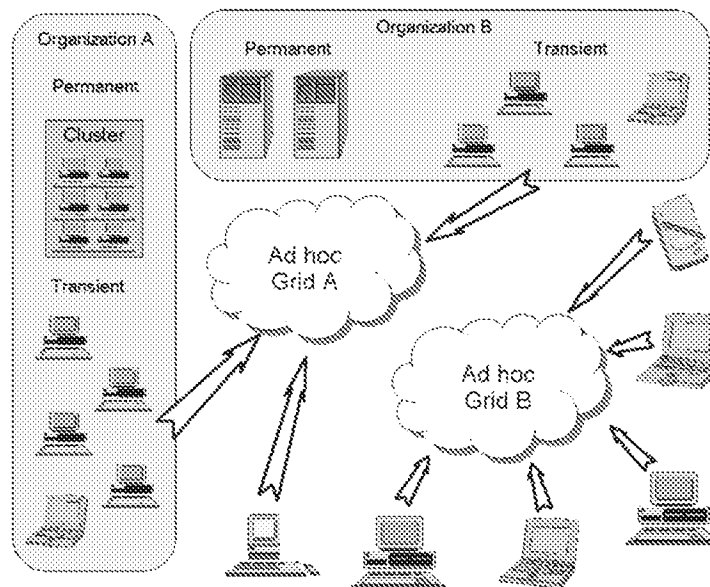


Fig. 1. Ad hoc grid architecture overview

PA-1, Fig. 1.

Fanshier similarly discloses a “computer system,” with a “processor,” “object code executed by said processor” for carrying out its methods. PA-2, cl. 32-34; *see also id.* ¶7 (describing “[s]ystems and methods in accordance with one embodiment of the present invention”).

The Smith-Fanshier combination therefore discloses [14pre]. Ex. C ¶¶157-159.

- a. **[14a]: “adding a new version of an application release bundle in a repository server;”**

Limitation [14a] is substantively identical to Limitation [1a]:

<u>Limitation [14a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server;”	“adding a new version of an application release bundle in a repository server;”

Thus, Limitation [14a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶160-161.

- b. **[14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”**

Limitation [14b] is substantively identical to Limitation [1b]:

<u>Limitation [9b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Thus, Limitation [14b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶162-163.

- c. [14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Limitation [14c] is substantively identical to Limitation [1c]:

<u>Limitation [14c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [14c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶164-165.

- d. [14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Limitation [14d] is substantively identical to Limitation [1d]:

<u>Limitation [14d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [14d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶166-167.

8. Claims 2, 10, and 15

Claims 2, 10, and 15 modify their respective independent claims by reciting:

wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:

selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

Ex. A, cl. 2, 10, 15.

For the reasons explained in Limitation [1d], the Smith-Fanshier combination renders this feature obvious, including “selecting and invoking” an “appropriate hot deployment plug-in . . . based on the data transfer protocol to use.” Ex. C ¶¶168-171

As explained, Smith’s disposable ClassLoader, use of JXTA and GT3 with GridFTP, and Fanshier’s MyNotificationListener class definition, renders obvious selecting and invoking an “appropriate hot deployment plug-in.” *Id.* In the Smith-Fanshier combination, it would have been obvious and understood that, in order to receive and deploy the new class definitions based on Smith’s HDS and the particular protocol to receive the deployment, that the application server on the grid node would “select” and “invok[e]” Smith’s HDS and its disposable ClassLoader, which is a client-side component capable of communicating over JXTA, GridFTP, or HTTP to receive and deploy Smith’s new classes. *Id.* A POSITA would have understood that “using” one of these, as discussed in Limitation [1d], includes “selecting” and “invoking” them to deploy the service classes. *Id.*

To the extent claims 2, 10, and 15 require selecting and invoking a protocol-specific plug-in, Smith renders this obvious by disclosing use of GT3 and its well-known use of GridFTP, which satisfies a protocol-specific plug-in for hot deploying Smith's service. *See* Limitation [1d]. *Id.*

9. Claims 3, 11, and 16

Claims 3, 11, and 16 modify their respective independent claims by reciting:

wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.

Ex. A, cl. 3, 11, 16.

A POSITA would have understood that a client-side component communicating over JXTA, GridFTP, or HTTP in the Smith combinations (e.g., to receive Smith's new classes) renders obvious "a data transfer client associated with a respective one of the client application managers." Ex. C ¶¶172-174.

For example, it would have been obvious in the Smith combinations that, to enable file transfers for deployment, components of the transfer protocol are available on the grid node to coordinate the data transfer via the protocol. *Id.* This is fundamental to network computing and distributed systems. *Id.* Therefore, it would have been obvious in Smith combinations to implement a "data transfer client" that is a client-side component capable of communicating over JXTA, GridFTP, or HTTP to receive and deploy Smith's new classes. *Id.* Smith's OGSA application on its grid nodes renders this feature obvious. PA-1, 5-6, Ex. C ¶¶172-174. So does Fanshier's slave deployer, which may be implemented as an Application MBean. Ex. PA-2 ¶¶13-15; Ex. C ¶¶172-174.

10. Claims 4 and 18

Claims 4 and 18 modify their respective independent claims by reciting:

wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.

Ex. A, cl. 4, 18.

To the extent “the application” in claims 4 and 18 modifies the “application release bundle” of claims 1 and 15, as discussed in Limitations [1a] and [1d], both Smith and Fanshier disclose “a set of components” that are hosted by “one or more application servers” in the one or more grid nodes. *See* Limitations [1a], [1d]; Ex. C ¶¶175-176. For example, Smith discloses deploying classes via a disposable ClassLoader. PA-1, 5-7, Fig. 9. So does Fanshier. PA-2 ¶57 (“Java 2 Platform, Enterprise Edition (J2EE)” deployments), ¶16 (“Application deployment can cover all standard J2EE deployable modules, such as enterprise applications, Enterprise JavaBeans (EJBs), web applications, and connectors.”); Ex. C ¶¶175-176. The Smith-Fanshier combination therefore renders this feature obvious.

11. Claim 5: “wherein the repository server stores one or more new versions of the application release bundle associated with the application.”

To the extent the claimed “repository server” of claim 5 is understood to reference the “repository server” of claim 1, claim 5 only requires the repository server to store one new version of the application release bundle (e.g., one *or* more). The Smith-Fanshier combination renders this feature obvious for the same reason it renders obvious a “adding a new version of an application release bundle in a repository server” in claim 1. *See* Limitation [1a]. To “add” the new version of an application release bundle is to “store” it on the repository server. Ex. C ¶177.

12. Claims 6, 12, and 17

Claims 6, 12, and 17 modify their respective independent claims by reciting:

registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.

Ex. A, cl. 6, 12, and 17.

As discussed in Limitation [1c], Fanshier discloses a client application manager associated with one or more of the determined grid nodes in its Application MBean. *See* Limitation [1c]; PA-2 ¶¶35-53. Fanshier discloses that its Application MBean “can listen for these events by *adding themselves as a notification listener* on the application MBean.” *Id.* ¶¶39-46. “The MyNotificationListener class . . . can implement a notification listener interface and perform any processing required when the notification is received.” *Id.* This discloses “registering” a client application manager to “accept[] notification of receipt of the new version of the application release bundle,” as claimed. A POSITA would have implemented Fanshier’s Application MBean as the client application manager, or at least its functionality, in Smith’s grid nodes as discussed in Limitation [1c]. Ex. C ¶¶178-179. A POSITA further would have implemented Fanshier’s teachings of “adding themselves as a notification listener” on Smith’s grid nodes because, after discovery, it would have enabled grid nodes to listen for updated applications for existing services and be notified of new versions or updates as they are released. *Id.* This would have been nothing more than applying well-known concepts in the prior art, as taught by Fanshier, to Smith, with predictable results. *Id.*

13. Claims 7 and 13

Claims 7 and 13 modify their respective independent claims by reciting:

wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.

Ex. A, cl. 7, 13.

To the extent the claimed “the data transfer plug-in” of these claims is understood to be the “data transfer protocol” of claims 1 and 9, Smith and Fanshier render this feature obvious for the reasons explained with respect to “protocol” in Limitation [1c]. Ex. C ¶¶180-181. For example, Smith renders obvious the use of GridFTP, while Fanshier renders obvious the use of HTTP. *See* Limitation [1c]. HTTP uses TCP/IP and Fanshier therefore renders this protocol obvious for the same reasons. *See* Limitation [1c]; Ex. C ¶¶180-181.

14. Claim 8: “The method of claim 1, wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.”

Claim 8 recites components “selected from the group consisting of Web containers, Web service containers, and EJB containers” and is satisfied by a component that is a Web container, a Web service container, or a EJB container.

Smith and Fanshier disclose this feature. For example, Smith discloses deploying services that include Java class definitions. PA-1, 5-6. A POSITA would have understood these classes to run in containers within the Java Runtime Environment because Smith discloses that its grid nodes must include a Java Runtime Environment. *See id.*, 4; Ex. C ¶¶182-184. Smith also discloses that its methods improve existing OGSA web applications “inside the web application container” by enabling hot deployment of services within those applications. *Id.*, 7. Smith therefore discloses at least web containers and web service containers. Ex. C ¶¶182-184.

Fanshier alternatively discloses deploying EJBs that are deployable units within Java containers. PA-2 ¶¶16; Ex. C ¶¶182-184. As discussed in Limitation [1a], it would have been obvious in the Smith-Fanshier combination to deploy either Smith or Fanshier’s components as the “application release bundle.” *See* Limitation [1a].

B. Smith and i-Dalfo Renders Obvious (SNQ 2) Claims 1-18

As explained below, the combination of Smith and i-Dalfo renders obvious claims 1-18.

1. Overview of i-Dalfo

Like Smith, i-Dalfo discloses deploying applications in a distributed computing environment. PA-3 (“i-Dalfo”), Abstract. i-Dalfo discloses a deployment tool and a deployment repository that includes metadata related to the deployment. *Id.*, Abstract, ¶¶10-11. “[W]hen an application is to be updated (or have resources updated), an authorized user, such as a system administrator, updates the metadata on the deployment repository 104 rather than directly updating the application (or resource) on the computing device 102.” *Id.* ¶13. The deployment tool queries the repository and receives a response with the metadata, which may “include references or links to resources to be included in the deployment of the application,” including “location information (e.g., URLs) of resources.” *Id.*, Abstract, ¶¶12, 15, 21. The deployment engine 110 “can then download the resource(s) using this location information.” *Id.* ¶15.

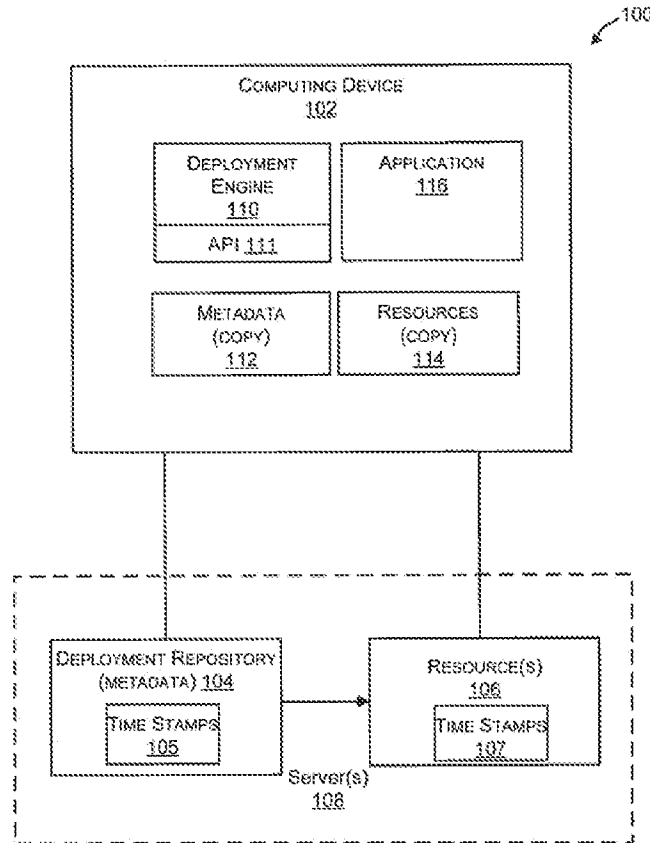


Fig. 1

Id., Fig. 1.

2. The Smith-i-Dalfo Combination

Smith and i-Dalfo are analogous art: Smith and i-Dalfo both disclose methods of deployment in distributed computing environments and, thus both, are in the same field of endeavor as each other and the '332 patent. PA-1, Abstract; PA-3, Abstract, ¶¶ 2-3; Ex. C ¶185. They are reasonably pertinent to the problems identified in the '332 patent because they both disclose methods that improve deploying applications in a distributed environment, where Smith focuses on grid environments and i-Dalfo (like Fanshier) focuses more generally on distributed servers. *See* PA-1, Abstract, 1. PA-3, Abstract, ¶¶ 2-3; Ex. C ¶185. Smith and i-Dalfo are therefore analogous art and their teachings are compatible. Ex. C ¶185.

A POSITA would have combined Smith and iDalfo: A POSITA would have found it obvious to modify Smith to incorporate i-Dalfo's use of a repository with metadata that indicates a new version of an application deployment with "location information (e.g., URLs) of resources" to be downloaded for the deployment. PA-3, Abstract, ¶¶10-15, 45; Ex. C ¶186. In the combined system, a notice of deployment to a grid node would include a server receiving i-Dalfo's metadata, including "location information (e.g., URLs) of resources" and "time stamp or version information [] that can be inspected before downloading the resource(s) so that only new or updated resource(s) are downloaded." PA-3, ¶15. A client application manager on the node, implemented in the combined system as one or more of Smith's HDS or i-Dalfo's deployment engine—would then "correctly install the application(s) using the metadata 112 and resources 114 that ha[ve] been downloaded to computing device 102." *Id.* ¶16; Ex. C ¶186.

A POSITA would have combined Smith's node discovery and register the discovered node for listening and/or "polling" i-Dalfo's repository for deployments. PA-1, 2-3, 3-4; PA-3 ¶¶12, 24; Ex. C ¶187. A POSITA would have been motivated to implement i-Dalfo's use of metadata into Smith for the reasons i-Dalfo explains: "[t]his feature can be advantageous when the system 100 includes multiple computing devices with the same application and/or resource to be updated or initially installed." PA-3, ¶13. Using metadata "provides a simple process (i.e., the updating of just the deployment repository 104) to deploy an application or update on multiple computing devices" that can avoid other, manual processes. *Id.* For example, an administrator does not have to "directly [deploy updated versions] on the computing device 102" and instead can notice updates via metadata in the shared repository. *Id.*; Ex. C ¶187.

A POSITA would have additionally been motivated to combine i-Dalfo's repository server with Smith because it would have provided a centralized storage location for versions of

software components in the combined system. Ex. C ¶188. This would have improved Smith's deployment of services by maintaining a current list of deployed applications and would have tracked any updated versions added by developers for deployment. *See* PA-4, 2:40-3:8 (disclosing that updated applications "are registered at the repository, in some embodiments a change record in the repository is updated"); PA-3, ¶13 (updating applications from a single source using metadata). Doing so would have benefitted Smith by offering the ability to maintain multiple applications, and new and older versions of applications and their components, in the combined system, for example, in the event rollback or version control is desired. *See* PA-4, 2:40-3:8 ("new and/or updated classes are 'hot deployed' [from the repository server] . . . while retaining the superseded class as long as it is still necessary."); *see also* PA-1, 6 (contemplating running "different versions of the same grid service on the same grid node, simply by using different ClassLoaders for each version.").

A POSITA also would have been motivated to combine Smith with i-Dalfo's repository metadata to provide nodes with a path or means by which to obtain the deployment. Ex. C ¶189. In remote deployments, this typically would have been required for the grid node to receive the files, components, and classes with which to execute the deployment. *Id.* A POSITA would have used i-Dalfo's metadata as it is low overhead that can provide necessary information about a deployment. *Id.* For example, it is information that likely requires little storage to implement and thus is easily transmitted to grid nodes and stored in a repository, and it is a convenient way to indicate new versions of deployments, as i-Dalfo discloses. *See* PA-3 ¶13.

Using i-Dalfo's repository from which to hot deploy services is compatible with Smith. Ex. C ¶190. Indeed, while Smith contemplates an ad hoc grid environment, Smith also discloses its "discovery mechanism is a loosely coupled grid service itself" and contemplates the ability

for its system to “use a central registry when needed.” PA-1, 4; Ex. C ¶190. A POSITA would have had a reasonable expectation of success in combining Smith and i-Dalfo because both relate to deploying applications in a distributed computing environment and because repository servers like i-Dalfo’s for deploying application components, including Java components, were well-known and understood in the art. Ex. C ¶190; PA-4, 3:9-35, 8:38-9:8; PA-3 ¶¶4, 10. Indeed, it was well known for repository servers to “install and deploy new/updated components developed by the content management system/application vendor[s].” PA-4, 1:16-34. Applying this type of storage to Smith would have been well within the knowledge and capabilities of a POSITA. Ex. C ¶190.

A POSITA also would have found it obvious to utilize i-Dalfo’s deployment engine 110 or implement its features as a “client application manager” in the combined system, to the extent Smith does not expressly disclose this (it does). Ex. C ¶191. i-Dalfo’s deployment engine 110 is notified of a deployment via metadata, which indicates a new version of an application is available on the repository for i-Dalfo’s server to download and deploy. PA-3, ¶¶12-13, 15, Figs. 1, 2. The deployment engine 110 “processes the metadata to determine whether there are additional resources needed to correctly install the application(s).” *Id.* In the combined system, Smith’s grid node would be notified of a deployment with resource information for the deployment, including location information (e.g., URLs) of resources, as taught by i-Dalfo. Ex. C ¶191.

A POSITA would have been motivated to modify Smith in this way for the reasons i-Dalfo explains: to provide a means for grid nodes to determine whether new versions of deployments are available, to “determine whether there are additional resources needed to correctly install the application(s),” and to actually download and deploy the application. PA-3

¶¶15-18; Ex. C ¶192. This would have involved nothing more than combining known features in the art with predictable results. Ex. C ¶192. Implementing Smith in this manner would have been well within the capabilities of a POSITA and led to a reasonable chance of success given that both Smith and i-Dalfo relate to deployments in a distributed computing environment. *Id.*

A POSITA additionally would have combined i-Dalfo with Smith to provide a means for its grid node to retrieve a deployment—which is an expected and obvious step in the deployment process. Ex. C ¶193. i-Dalfo discloses that the metadata indicating the new deployment additionally includes location information (*e.g., URLs*) of resources.” PA-3 ¶15 (emphasis added). The deployment engine 110 can then “download the resource(s) *using this location information.*” *Id.* (emphasis added); *see also id.* ¶¶21, 28. A POSITA would have been motivated to notify grid nodes of a deployment using i-Dalfo’s metadata and had an expectation of success in doing so because it would have entailed nothing more than providing data in a notification for the node to determine how to download the resources for the deployment. Ex. C ¶193. This would have been nothing more than combining known elements in the prior art with expected results. *Id.*

3. Claim 1

- a. [1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”

The combination of Smith and i-Dalfo discloses “*a method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes.*” Ex. C ¶¶194-198 As discussed in SNQ 1, Smith discloses these features. *See* Section VIII.A.5.a.

b. [1a]: “adding a new version of an application release bundle in a repository server;”

The combination of Smith and i-Dalfo discloses “*adding a new version of an application release bundle in a repository server.*” Ex. C ¶¶199-203. As discussed in SNQ 1, Smith discloses “a new version of an application release bundle.” *See* Section VIII.A.5.b. i-Dalfo further discloses a “adding a new version . . . in a repository server.” Ex. C ¶¶199-203.

New version of an application release bundle: as discussed above, Smith discloses deploying new versions of an application release bundle on its grid nodes. *See* Section VIII.A.5.b; Ex. C ¶¶199-203.

To the extent Smith does not disclose or render obvious this claim feature (it does), i-Dalfo also discloses a new version of an application release bundle by disclosing that “when an application is to be updated (or have resources updated), an authorized user, such as a system administrator, updates the metadata on the deployment repository 104.” PA-3, ¶13; *see also id.* ¶¶11-12, Fig. 2. A POSITA would have combined Smith and i-Dalfo for the reasons discussed above. *See* Section VIII.B.2; Ex. C ¶¶199-203.

Adding . . . in a repository server: As discussed in Section VIII.A.5.b, software release repositories were well known in the art as of the earliest claimed priority date of the ’332 patent, including adding new versions of Java classes to the same. Ex. C ¶¶199-203; PA-4, Abstract (“A new or updated software object class is received at a repository”), 6:31-34 (business object class “added to and/or updated in the repository”), 8:22-30 (same); 9:39-10:11 (updating “business object classes deployed from a repository”); PA-5 ¶¶17-18 (deploying enterprise Java beans (EJBs) and obtaining information about a server from a repository); PA-3 ¶24 (storing metadata “by the deployment repository [that] includes information (e.g., a timestamp or version indication)”).

Given this knowledge, a POSITA would have found it obvious to modify Smith to add new versions of its services and Java classes for deployment “in a repository server,” as taught by i-Dalfo. *See* PA-3, ¶11 (“deployment *repository 104* includes metadata information that allows the deployment engine 110 to correctly deploy or update an application (e.g. application 116 on the computing device 102)”) (emphasis added); *see also id.*, ¶4 (disclosing that repositories are “typically located on a server”), ¶10 (“ the deployment repository 104 and the resources 106 are accessible to the computing device 102 via one or more *servers 108*” (emphasis added)); *see also id.* ¶¶10-13, 15, 24, Fig. 1 (depicting deployment repository 104 as “Server(s) 108”); Ex. C ¶¶199-203. A POSITA would have been motivated to combine i-Dalfo’s adding a new version of an application to a repository as discussed in Section VIII.B.2.

- c. [1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

The combination of Smith and i-Dalfo discloses “*determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.*” Ex. C ¶¶204-206. As discussed in SNQ 1, Smith discloses a discovery services module that can evaluate grid nodes for an application associated with the added new version of the application release bundle. *See* Section VIII.A.5.c. It would have been obvious to employ Smith’s discovery services module to determine which grid nodes have an associated application upon adding the new version of an application release bundle a repository server, as taught by i-Dalfo. Ex. C ¶¶204-206.

Upon adding to the repository server: In the Smith-i-Dalfo combination using a repository server, it would have been obvious to perform this node discovery and property

check when the new classes (e.g., the application release bundle) are added to the repository server (as taught by i-Dalfo). Ex. C ¶¶205-206. This corresponds with the sequence of steps in Smith, including: “peer-to-peer based node discovery, automatic node property assessment, [and] hot deployment of services into a running system.” PA-1, Abstract. A POSITA thus would have understood and expected the steps of node discovery, property assessment, and hot deployment to be preceded by the application version being made available on a repository server in the combined system. Ex. C ¶¶205-206.

Initiating discovery in this way would have been nothing more than combining known elements in the prior art to achieve predictable results. Ex. C ¶¶205-206. A POSITA would have had a reasonable expectation of success in the combination given the similarities between Smith and i-Dalfo and the ubiquity of deployment tools for initiating deployments. *See, e.g.*, PA-7 ¶21 (“deployer 110 checks for updates”). Modifying Smith to discover new nodes when a new version of the service is made available would have improved dissemination of the service. Ex. C ¶¶205-206. Likewise, it would have been obvious to determine which nodes already have the service or application for the service when an update is made available to identify those nodes available to receive the update. *Id.* Doing so would have been nothing more than combining known features in the art according to understood principles to achieve predictable results. *Id.*

- d. [1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

The combination of Smith and i-Dalfo discloses “*notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use.*”

Ex. C ¶¶207-216. As discussed in SNQ 1, Smith's OGSA application on a grid node is a "client application manager." *Id.* i-Dalfo similarly discloses a deployment engine 110 that can receive metadata from a deployment repository, download the deployment, and install it. PA-3 ¶¶13-16. In the combined system, it would have been obvious to provide Smith's grid nodes with a notification that includes "a type of data transfer protocol to use" based on i-Dalfo's metadata that "includes location information (e.g., URLs) of resources." *Id.* ¶15; Ex. C ¶¶205-206.

Notifying a client application manager associated with a grid node about adding the new version of the application release bundle: As discussed in SNQ 1, a POSITA would have understood that Smith's OGSA application on each grid node discloses a "client application manager associated with one or more of the determined grid nodes." *See* Section VIII.A.5.d; Ex. C ¶¶208-210.

While Smith does not expressly disclose the details of its grid nodes receiving notifications of deployment or how the components of the deployment are transferred, this feature would have been obvious in view of i-Dalfo. Ex. C ¶¶208-210. For example, i-Dalfo discloses a deployment engine 110 on a computing device that is notified of a deployment via metadata. i-Dalfo., ¶¶12-13, 15, Figs. 1, 2. The deployment engine 110 "processes the metadata to determine whether there are additional resources needed to correctly install the application(s)." *Id.*

It would have been obvious to combine i-Dalfo's metadata notification with Smith to deploy services. Ex. C ¶¶208-210. As discussed in SNQ 1, it would have been an obvious and expected step to first notify a grid node of a deployment and transfer any necessary components for the deployment to the grid node before executing it. *Id.* Combining Smith and i-Dalfo to provide such a notification would have been nothing more than combining known features in

the prior art according to understood principles. *Id.* For example, a POSITA would have understood that notifying a grid node of deployments was well-known and ubiquitous in distributed computing. *See, e.g.*, PA-7 ¶26 (“If deployer 110 configuration is so set, an update may initiate at 206 when deployer 110 sends a message to application server 120 to request an update for the modified application.”); Ex. C ¶¶208-210. A POSITA would have been motivated to combine i-Dalfo’s notifications with Smith to inform a target grid node of the deployment, particularly if resources must be allocated on the grid node for the deployment (e.g., for staging components and files, and for executing Smith’s hot deployment service). Ex. C ¶¶208-210.

Along with a type of data transfer protocol to use: The Smith-i-Dalfo combination renders this feature obvious in three alternative ways. Ex. C ¶¶211-216.

First, Smith renders this feature obvious by disclosing the use of JXTA, as discussed in SNQ 1. *See* Section VIII.A.5.d.; Ex. C ¶212.

Second, it alternatively would have been obvious to notify a grid node of a deployment and transfer necessary components for the deployment using GridFTP in Smith’s architecture. *See* Section VIII.A.5.d.; Ex. C ¶213.

Third, it alternatively would have been obvious to provide a notification with a protocol to use in Smith in view of i-Dalfo. i-Dalfo’s deployment engine downloads the metadata and “processes the metadata to determine whether there are additional resources needed to correctly install the application(s).” PA-3 ¶¶12-13, 15, Figs. 1, 2. The resources may include “dynamic linked libraries, files, assemblies, etc.,” and “[i]n some embodiments, the metadata includes *location information (e.g., URLs)* of resources.” *Id.* ¶15 (emphasis added). The deployment

engine 110 can then “download the resource(s) *using this location information.*” *Id.* (emphasis added); *see also id.* ¶¶21, 28; Ex. C ¶¶214-216.

As was known, a URL typically includes a protocol, such as HTTP, HTTPS, or FTP, and specify how to access resources over the Internet. Ex. C ¶¶214-216. These URLs rely on TCP/IP for underlying data transport. *Id.* A POSITA thus would have understood that i-Dalfo’s providing “location information (e.g., URLs)” for resources related to a deployment discloses notifying of a deployment “along with a type of data transfer protocol to use,” as claimed. For example, the URL may be http://, https://, ftp://, or gridftp://, which would indicate these associated protocols to use for retrieving necessary files and components for a deployment. *Id.*

i-Dalfo’s deployment tool querying the repository does not teach away from the deployment tool be “notified” of a new deployment because i-Dalfo discloses that its deployment engine 110 on the computing device 102 “accesses or queries the deployment repository 103 periodically,” and in response receives a query response and “downloads” metadata from the repository indicating the new deployment. Ex. C ¶¶214-216. i-Dalfo’s query response is a from the deployment repository is a “notification” of a deployment, as claimed. *Id.*

- e. **[1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”**

The combination of Smith and i-Dalfo discloses “*hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.*” Ex. C ¶217. For the reasons discussed in SNQ 1, Smith discloses these features. *See* Section VIII.A.5.e. For example, it would have been obvious to transfer the components of the deployment, including Smith’s HDS

and new class definitions with the disposable ClassLoader in Smith, via one or more of Smith's JXTA (including its support for TCP/IP and HTTP), GridFTP, or HTTP via i-Dalfo's URL. *See* Limitation [1c]. These disclose various data transfer clients, as contemplated by the '332 patent. Ex. C ¶217. The new class definitions would then be hot deployed via Smith's disposable ClassLoader, which is a plug-in. *Id.* This satisfies or at least renders obvious using an appropriate hot deployment plug-in for the deployment "based on the data transfer protocol." *Id.*

4. Independent Claim 9

- a. [9pre]: "A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising"

Independent claim 9 recites a "computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform" the method of claim 1. Smith and i-Dalfo disclose computer systems that include computer-readable medium with program code to cause executing of their disclosed feature. For example, Smith discloses the use of the Open Grid Services Architecture (OGSA) and Globus ToolKit 3.0, which are implemented on servers and computer systems with memory and software instructions. PA-1, 1-2; Ex. C ¶¶218-220.

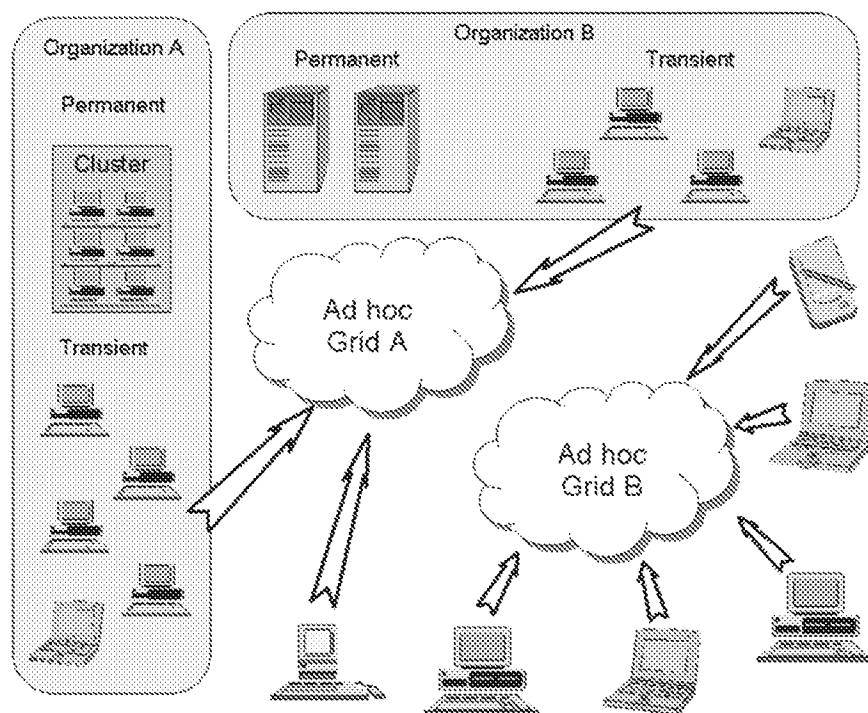


Fig. 1. Ad hoc grid architecture overview

PA-1, Fig. 1.

i-Dalfo similarly discloses a “computer-readable medium,” “computer-executable instructions, such as program modules,” and a “computer system.” i-Dalfo, ¶¶9, 23, 33-39, 45-49, cl. 10-20. The Smith-i-Dalfo combination therefore discloses these features. Ex. C ¶219.

The method of Limitation [9pre] is otherwise substantively identical to Limitation [1pre]:

<u>Limitation [9pre]</u>	<u>Limitation [1pre]</u>
“a method for hot deployment/redeployment in a grid computing environment comprising.”	“A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising;”

Thus, Limitation [9pre] would have been obvious for the same reasons discussed above in Limitation [1pre]. Ex. C ¶¶218-220.

- a. [9a]: “adding a new version of an application release bundle in a repository server;”

Limitation [9a] is substantively identical to Limitation [1a]:

<u>Limitation [9a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server;”	“adding a new version of an application release bundle in a repository server;”

Thus, Limitation [9a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶221-222.

- b. [9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Limitation [9b] is substantively identical to Limitation [1b]:

<u>Limitation [9b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Thus, Limitation [9b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶223-224.

- c. [9c]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Limitation [9c] is substantively identical to Limitation [1c]:

<u>Limitation [9c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [9c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶225-226.

- d. [9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Limitation [9d] is substantively identical to Limitation [1d]:

<u>Limitation [9d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [9d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶227-228.

5. Independent Claim 14

- a. [14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”

Independent claim 14 recites a computer system with a processing unit and memory for executing the method of claim 1. As discussed in claim 9, Smith and Fanshier disclose computer systems that include computer-readable medium with program code to cause execution of their disclosed features. Ex. C. ¶¶229-231. For example, Smith discloses the use of the Open Grid Services Architecture (OGSA) and Globus ToolKit 3.0, which are implemented on servers and computer systems with memory and software instructions. PA-1, 1-2; Ex. C ¶¶232.

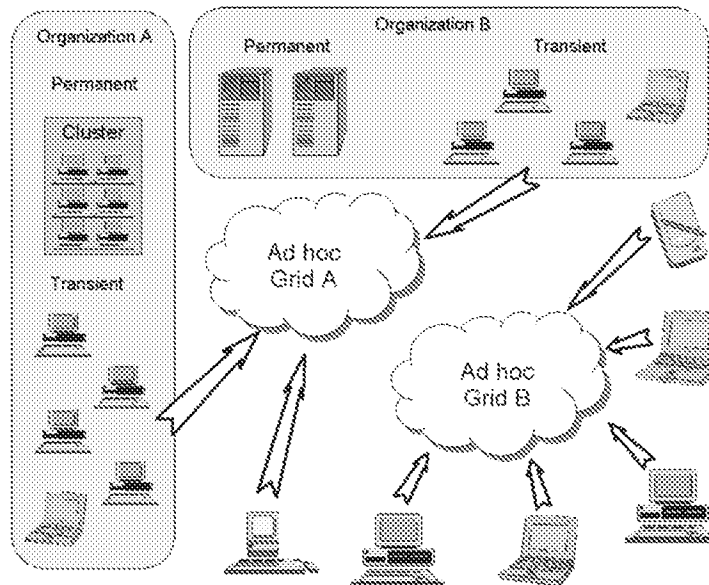


Fig. 1. Ad hoc grid architecture overview

PA-1, Fig. 1.

i-Dalfo similarly discloses a “computer system,” “one or more processors or processing units 304,” and “system memory 306.” i-Dalfo, ¶¶9, 23, 33-39, 45-49, cl. 10-20. The Smith-i-Dalfo combination therefore discloses these features. Ex. C ¶¶229-231.

The Smith-i-Dalfo combination therefore discloses [14pre]. *Id.*

- a. [14a]: “a method for hot deployment/redeployment in a grid computing environment comprising:”

Limitation [14a] is verbatim of Limitation [1p]. The Smith combinations satisfy this limitation for the same reasons. *See* Limitation [1p]. Ex. C ¶232.

- b. [14a]: “adding a new version of an application release bundle in a repository server;”

Limitation [14a] is substantively identical to Limitation [1a]:

<u>Limitation [14a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server;”	“adding a new version of an application release bundle in a repository server;”

Thus, Limitation [14a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶233-234.

- c. [14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Limitation [14b] is substantively identical to Limitation [1b]:

<u>Limitation [9b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the

application release bundle in the repository server;”	application release bundle in the repository server;”
---	---

Thus, Limitation [14b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶235-236.

- d. **[14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”**

Limitation [14c] is substantively identical to Limitation [1c]:

<u>Limitation [14c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [14c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶237-238.

- e. **[14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”**

Limitation [14d] is substantively identical to Limitation [1d]:

<u>Limitation [14d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [14d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶239-240.

6. Claims 2, 10, and 15: “selecting and invoking an appropriate hot deployment plug-in . . . and . . . hot deploying/redeploying . . . using the appropriate hot deployment plug-in”

Claims 2, 10, and 15 modify their respective independent claims by reciting:

wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:

selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

Ex. A, cl. 2, 10, 15.

For the reasons explained in Limitation [1d] and SNQ 1, the Smith-i-Dalfo combination renders this feature obvious, including “selecting and invoking” an “appropriate hot deployment plug-in . . . based on the data transfer protocol to use.” *See* Section VIII.A.8; Ex. C ¶¶241-242.

7. Claims 3, 11, and 16

Claims 3, 11, and 16 modify their respective independent claims by reciting:

wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:

hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.

Ex. A, cl. 3, 11, 16.

For the reasons explained in SNQ 1, Smith discloses this feature. *See* Section VIII.A.9.

It also would have been obvious in Smith that, to enable file transfers for deployment,

components of the transfer protocol are available on the grid node to coordinate the data transfer via the protocol. Ex. C ¶¶243-245. This is fundamental to network computing and distributed systems. *Id.* Therefore, it would have been obvious in Smith-i-Dalfo combinations to implement a “data transfer client” that is a client-side component capable of communicating over JXTA, GridFTP, or HTTP to receive and deploy Smith’s new classes. *Id.* For example, i-Dalfo discloses downloading the necessary resources for the deployment via a URL, and discloses that one or more modules may implement this feature. *See* PA-3 ¶¶9, 23 (“it should be appreciated that the logical operations of various embodiments are implemented [] as a sequence of computer implemented acts or program modules running on a computing system”), 33, 36, 39, 45.

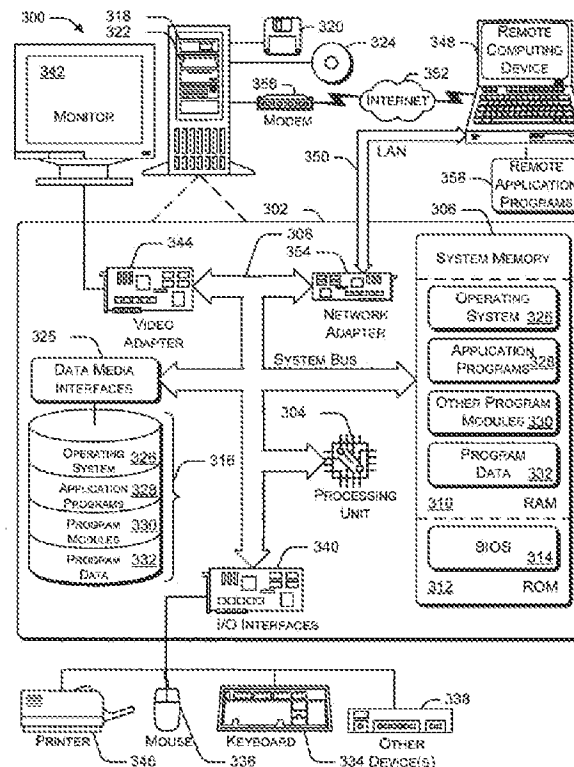


Fig. 3

Id., Fig. 3.

Given i-Dalfo's disclosure of communicating over a network using a URL (e.g., HTTP protocol), it would have been obvious and understood to utilize a "data transfer client associated with a respective one of the client application managers," or similar software module, for accomplishing this function. Ex. C ¶¶243-245. The Smith-i-Dalfo combination therefore renders this feature obvious. *Id.*

8. Claims 4 and 18

Claims 4 and 18 modify their respective independent claims by reciting:

wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.

Ex. A, cl. 4, 18.

To the extent "the application" in claims 4 and 18 modifies the "application release bundle" of claims 1 and 15, as discussed in Limitations [1a] and [1d], Smith discloses "a set of components" that are hosted by "one or more application servers" in the one or more grid nodes. *See* Limitations [1a], [1d]. For example, Smith discloses deploying classes via a disposable ClassLoader. PA-1, 5-7, Fig. 9. The Smith-i-Dalfo combination therefore renders this feature obvious. Ex. C ¶¶246-247.

9. Claim 5: "wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application."

To the extent the claimed "repository sewer" of claim 5 is understood to reference the "repository server" of claim 1, claim 5 only requires the repository server to store one new version of the application release bundle (e.g., one *or* more). The Smith-i-Dalfo combination renders this feature obvious for the same reason it renders obvious a "adding a new version of an application release bundle in a repository server" in claim 1. *See* Limitation [1a]. To "add"

the new version of an application release bundle is to “store” it on the repository server. Ex. C ¶248.

10. Claims 6, 12, and 17

Claims 6, 12, and 17 modify their respective independent claims by reciting:

registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.

Ex. A, cl. 6, 12, and 17.

As discussed in Limitation [1c], i-Dalfo’s the deployment engine 110 “periodically poll[ing] the deployment repository 104 for metadata corresponding to new or updated application(s) for the computing device 102” renders this feature obvious. PA-3 ¶12. By polling and receiving metadata indicating a new version of a deployment is available, i-Dalfo discloses “registering . . . for accepting notification of receipt of the new version of the application release bundle.” Ex. C ¶¶249-250.

11. Claims 7 and 13

Claims 7 and 13 modify their respective independent claims by reciting:

wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.

Ex. A, cl. 7, 13.

To the extent the claimed “the data transfer plug-in” of these claims is understood to be the “data transfer protocol” of claims 1 and 9, Smith and i-Dalfo render this feature obvious for the reasons explained with respect to “protocol” in Limitation [1c]. For example, Smith renders obvious the use of GridFTP, while i-Dalfo renders obvious the use of HTTP. *See* Limitation [1c]. HTTP uses TCP/IP and i-Dalfo via its “URL” renders this protocol obvious for the same reasons. *See* Limitation [1c]; Ex. C ¶¶251-252.

12. Claim 8: “wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers.”

Claim 8 recites components “selected from the group consisting of Web containers, Web service containers, and EJB containers” and is satisfied by a component that is a Web container, a Web service container, or a EJB container.

Smith discloses this feature. For example, Smith discloses deploying services that include Java class definitions. PA-1, 5-6. A POSITA would have understood these classes to run in containers within the Java Runtime Environment because Smith discloses that its grid nodes must include a Java Runtime Environment. *See id.*, 4; Ex. C ¶¶253-254. Smith also discloses that its methods improve existing OGSA web applications “inside the web application container” by enabling hot deployment of services within those applications. PA-1, 7. Smith therefore discloses at least web containers and web service containers. Ex. C ¶¶253-254.

C. Smith, i-Dalfo, and Fanshier Renders Obvious (SNQ 3) Claims 1-18

The combination of Smith, i-Dalfo, and Fanshier renders obvious claims 1-18. To the extent Smith-i-Dalfo fail to render obvious “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle,” Smith and i-Dalfo, in view of Fanshier does. Ex. C ¶255.

1. Claim 1

- a. [1pre]: “A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”**

As discussed in SNQ 1, Smith discloses these features. *See* Section VIII.A.5.a; Ex. C ¶256.

- b. [1a]: “adding a new version of an application release bundle in a repository server;”

As discussed in SNQ 2, the combination of Smith and i-Dalfo discloses these features.

See Section VIII.B.3.b; Ex. C ¶257.

- c. [1b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

As discussed in SNQ 2, the combination of Smith and i-Dalfo discloses these features.

See Section VIII.B.3.c; Ex. C ¶258.

- d. [1c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

The combination of Smith, i-Dalfo, and Fanshier discloses “*notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use.*” Ex. C ¶¶259-262.

As discussed in SNQ 2, Smith’s OGSA application on a grid node is a “client application manager” and i-Dalfo similarly discloses a deployment engine 110 that can receive metadata from a deployment repository, download the deployment, and install it. See Section VIII.B.3.d (Element [1c]). To the extent i-Dalfo’s “polling” the repository for metadata and associated application updates does not render the claimed notification obvious (it does), this feature would have alternatively been obvious in view of Fanshier’s notifications. Ex. C ¶¶259-262.

As discuss in SNQ 1, Fanshier discloses that a master deployer can “*contact* a slave deployer” on a server “that is to deploy the application, indicating that the slave deployer should

prepare to deploy the application.” PA-2 ¶7 (emphasis add). The master deployer provides the source of the deployment (e.g., the repository server from which to retrieve the components to be deployed) and “*send[s]* this path to slave deployers 108, 116.” *Id.* ¶13 (emphasis added); *see also id.*, ¶14 (disclosing that its master deployer “inform[s] the target deployers, which can download the application to a staging area”), ¶17 (“All members of the target can be *notified* of the deployment and can take the appropriate action.”) (emphasis added), ¶39 (“A deployment *notification* can be broadcast for certain events, which can be defined in a deployment notification class.”) (emphasis added); claim 16 (“*contacting* a slave deployer . . . by the administration server to indicate that an application is to be deployed on that managed server”) (emphasis added). The slave deployer then retrieves the application components to be deployed from an application source. *Id.* ¶8 (slave deployer then “pull[ing] the application”), ¶13 (“pull[ing] the file”) ¶14 (“download[ing] the application”), ¶23 (same). Fanshier discloses a specific embodiment for its slave deployer in an “Application MBean,” which registers to listen for deployment tasks from an administrator server. *Id.* ¶¶39-46 (disclosing, for example, “MyNotificationListener class” to “implement a notification listener interface and perform any processing required when the notification is received”).

It would have been obvious to modify Smith-i-Dalfo in the combined system with Fanshier’s application notifications when a new application version is added to i-Dalfo’s repository. Ex. C ¶¶259-262. This would have been nothing more than combining known methods of indicating new deployments in a distributed computing environment, with predictable results, where i-Dalfo discloses polling a repository and Fanshier discloses sending affirmative notifications. *Id.* A POSITA would have had a reasonable expectation of success because there were known methods of noticing deployments in distributed computing. *Id.* A

POSITA would have been motivated to modify Smith-i-Dalfo in this way to notify grid nodes of a deployment as soon as the new version is available instead of relying on grid nodes to first poll the repository, which may involve a delay period. *Id.* This would facilitate deployment across multiple nodes while minimizing delay, particularly for updates of particular importance, such as those involving security patches or high priority applications. *Id.* The notification in the combined system would provide a “data transfer protocol to use” as explained in SNQ 2. *See* Section VIII.B.3.d (Element [1c]).

- e. **[1d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”**

The combination of Smith and i-Dalfo discloses “*hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.*” Ex. C ¶263. For the reasons discussed in SNQ 1, Smith discloses these features. *See* Section VIII.A.5.e. For example, it would have been obvious to transfer the components of the deployment, including Smith’s HDS and new class definitions with the disposable ClassLoader in Smith, via one or more of Smith’s JXTA (including its support for TCP/IP and HTTP), GridFTP, or HTTP via i-Dalfo’s URL. *See* Limitation [1c]. These disclose various data transfer clients, as contemplated by the ’332 patent. Ex. C ¶263. The new class definitions would then be hot deployed via Smith’s disposable ClassLoader, which is a plug-in. *Id.* This satisfies or at least renders obvious using an appropriate hot deployment plug-in for the deployment “based on the data transfer protocol.” *Id.*

2. Independent Claim 9

- a. [9pre]: “A computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform a method for hot deployment/redeployment in a grid computing environment comprising”

Independent claim 9 recites a “computer-readable medium with program code embodied therein for causing when executed, a computer system, to perform” the method of claim 1.

Smith and i-Dalfo disclose these features as explained in SNQ 2. *See* Section VIII.B.4.a.

The method of Limitation [9pre] is otherwise substantively identical to Limitation [1pre]:

<u>Limitation [9pre]</u>	<u>Limitation [1pre]</u>
“a method for hot deployment/redeployment in a grid computing environment comprising.”	“A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising;”

Thus, Limitation [9pre] would have been obvious for the same reasons discussed above in Limitation [1pre]. Ex. C ¶¶264-266.

- b. [9a]: “adding a new version of an application release bundle in a repository server;”

Limitation [9a] is substantively identical to Limitation [1a]:

<u>Limitation [9a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server;”	“adding a new version of an application release bundle in a repository server;”

Thus, Limitation [9a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶267-268.

- c. [9b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Limitation [9b] is substantively identical to Limitation [1b]:

<u>Limitation [9b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Thus, Limitation [9b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶269-270.

- d. [9c]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Limitation [9c] is substantively identical to Limitation [1c]:

<u>Limitation [9c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [9c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶271-272.

- e. [9d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Limitation [9d] is substantively identical to Limitation [1d]:

<u>Limitation [9d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [9d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶273-274.

3. Independent Claim 14

- a. [14pre]: “A computer system comprising: a processing unit; and a memory coupled to the processing unit, the memory having stored therein code for causing, when executed, the processing unit to perform a method comprising:”

Independent claim 14 recites a computer system with a processing unit and memory for executing the method of claim 1. Smith and i-Dalfo disclose these features as explained in SNQ 2. *See* Section VIII.B.5.a (Element [14pre]); Ex. C ¶275.

- b. [14a]: “a method for hot deployment/redeployment in a grid computing environment comprising:”

Limitation [14a] is verbatim of Limitation [1p]. The Smith combinations satisfy this limitation for the same reasons. *See* Limitation [1p]; Ex. C ¶276.

- c. **[14a]: “adding a new version of an application release bundle in a repository server;”**

Limitation [14a] is substantively identical to Limitation [1a]:

<u>Limitation [14a]</u>	<u>Limitation [1a]</u>
“adding a new version of an application release bundle in a repository server;”	“adding a new version of an application release bundle in a repository server;”

Thus, Limitation [14a] would have been obvious for the same reasons discussed above in Limitation [1a]. Ex. C ¶¶277-278.

- d. **[14b]: “determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”**

Limitation [14b] is substantively identical to Limitation [1b]:

<u>Limitation [14b]</u>	<u>Limitation [1b]</u>
“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”	“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

Thus, Limitation [14b] would have been obvious for the same reasons discussed above in Limitation [1b]. Ex. C ¶¶279-280.

- e. **[14c]: “notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”**

Limitation [14c] is substantively identical to Limitation [1c]:

<u>Limitation [14c]</u>	<u>Limitation [1c]</u>
“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”	“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use; and”

Thus, Limitation [14c] would have been obvious for the same reasons discussed above in Limitation [1c]. Ex. C ¶¶281-282.

- f. [14d]: “hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Limitation [14d] is substantively identical to Limitation [1d]:

<u>Limitation [14d]</u>	<u>Limitation [1d]</u>
“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”	“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

Thus, Limitation [14d] would have been obvious for the same reasons discussed above in Limitation [1d]. Ex. C ¶¶283-284.

4. Claims 2, 10, and 15: “selecting and invoking an appropriate hot deployment plug-in . . . and . . . hot deploying/redeploying . . . using the appropriate hot deployment plug-in”

Claims 2, 10, and 15 modify their respective independent claims by reciting:

wherein hot deploying/redeploying the new version of the application release bundle on the running one or more application servers in the associated grid

using the appropriate hot deployment plug-in based on the data transfer protocol by the respective client application manager, comprises:

- selecting and invoking an appropriate hot deployment plug-in by a respective one of the client application managers based on the data transfer protocol to use for hot deploying/redeploying the new version of the application release bundle in an associated grid node; and

- hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in in the associated grid node by the respective client application manager.

Ex. A, cl. 2, 10, 15.

For the reasons explained in SNQ 2, the Smith-i-Dalfo-Fanshier combination renders this feature obvious, including “selecting and invoking” an “appropriate hot deployment plug-in . . . based on the data transfer protocol to use.” *See* Section VIII.B.6; Ex. C ¶¶285-286.

5. Claims 3, 11, and 16

Claims 3, 11, and 16 modify their respective independent claims by reciting:

- wherein hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by the respective client application manager comprises:

- hot deploying/redeploying the new version of the application release bundle on running one or more application servers using the appropriate hot deployment plug-in by a data transfer client associated with a respective one of the client application managers.

Ex. A, cl. 3, 11, 16.

For the reasons explained in SNQ 1, Smith discloses this feature. *See* Section VIII.A.9. It also would have been obvious in Smith that, to enable file transfers for deployment, components of the transfer protocol are available on the grid node to coordinate the data transfer via the protocol. Ex. C ¶¶287-289. This is fundamental to network computing and distributed systems. *Id.* Therefore, it would have been obvious in Smith combinations to implement a “data transfer client” that is a client-side component capable of communicating over JXTA, GridFTP, or HTTP to receive and deploy Smith’s new classes. *Id.* For example, i-Dalfo discloses downloading the necessary resources for the deployment via a URL, and discloses that one or

more modules may implement this feature. See PA-3 ¶¶9, 23 (“it should be appreciated that the logical operations of various embodiments are implemented [] as a sequence of computer implemented acts or program modules running on a computing system”), 33, 36, 39, 45.

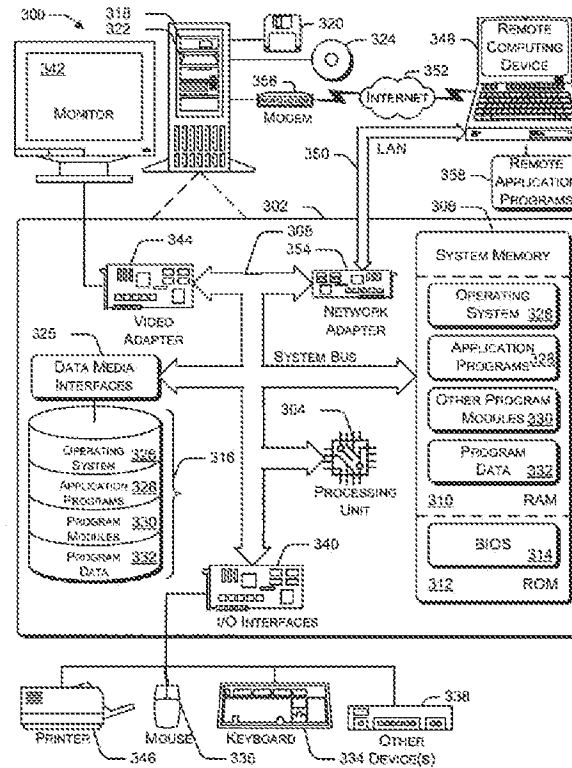


Fig. 3

Id., Fig. 3.

Given i-Dalfo's disclosure of communicating over a network using a URL (e.g., HTTP protocol), it would have been obvious and understood to utilize a "data transfer client associated with a respective one of the client application managers," or similar software module, for accomplishing this function. Ex. C ¶¶287-289. The Smith-i-Dalfo-Fanshier combination therefore renders this feature obvious.

6. Claims 4 and 18

Claims 4 and 18 modify their respective independent claims by reciting:

wherein the application comprises a set of components that are hosted by one or more application servers in the one or more grid nodes.

Ex. A, cl. 4, 18.

To the extent “the application” in claims 4 and 18 modifies the “application release bundle” of claims 1 and 15, as discussed in Limitations [1a] and [1d], Smith discloses “a set of components” that are hosted by “one or more application servers” in the one or more grid nodes. *See* Limitations [1a], [1d]. For example, Smith discloses deploying classes via a disposable ClassLoader. PA-1, 5-7, Fig. 9. The Smith-i-Dalfo-Fanshier combination therefore renders this feature obvious. Ex. C ¶¶290-291.

7. Claim 5: “wherein the repository se[rv]er stores one or more new versions of the application release bundle associated with the application.”

To the extent the claimed “repository sewer” of claim 5 is understood to reference the “repository server” of claim 1, claim 5 only requires the repository server to store one new version of the application release bundle (e.g., one *or* more). The Smith-i-Dalfo combination renders this feature obvious for the same reason it renders obvious a “adding a new version of an application release bundle in a repository server” in claim 1. *See* Limitation [1a]. To “add” the new version of an application release bundle is to “store” it on the repository server. Ex. C ¶292.

8. Claims 6, 12, and 17

Claims 6, 12, and 17 modify their respective independent claims by reciting:

registering the client application manager associated with one or more of the determined grid nodes with the repository server for accepting notification of receipt of the new version of the application release bundle.

Ex. A, cl. 6, 12, and 17.

As discussed in SNQ 1, Fanshier's Application MBean discloses this feature and it would have been obvious to implement the client application manager in the combined system to include this functionality. *See* Section VIII.B.10; *see also* Section VIII.A.5.d (discussing the "client application manager" in the Smith-Fanshier combination); Ex. C ¶¶293-294.

9. Claims 7 and 13

Claims 7 and 13 modify their respective independent claims by reciting:

wherein the data transfer plug-in includes protocols selected from the group consisting of ftp, http, gridftp, tcp, and udp.

Ex. A, cl. 7, 13.

To the extent the claimed "the data transfer plug-in" of these claims is understood to be the "data transfer protocol" of claims 1 and 9, Smith and i-Dalfo render this feature obvious for the reasons explained with respect to "protocol" in Limitation [1c]. For example, Smith renders obvious the use of GridFTP, while i-Dalfo renders obvious the use of HTTP. *See* SNQ 2. HTTP uses TCP/IP and i-Dalfo via its "URL" renders this protocol obvious for the same reasons. *See* SNQ 2; Ex. C ¶¶295-296.

10. Claim 8: "wherein each grid node comprises one or more application servers that include components selected from the group consisting of Web containers, Web service containers, and EJB containers."

Claim 8 recites components "selected from the group consisting of Web containers, Web service containers, and EJB containers" and is satisfied by a component that is a Web container, a Web service container, or a EJB container.

Smith discloses this feature. For example, Smith discloses deploying services that include Java class definitions. PA-1, 5-6. A POSITA would have understood these classes to run in containers within the Java Runtime Environment because Smith discloses that its grid nodes must include a Java Runtime Environment. *See id.*, 4; Ex. C ¶¶297-298. Smith also

discloses that its methods improve existing OGSA web applications “inside the web application container” by enabling hot deployment of services within those applications. PA-1, 7. Smith therefore discloses at least web containers and web service containers. Ex. C ¶¶297-298..

IX. CONCLUSION

For the foregoing reasons, NetApp requests reexamination and cancellation of claims 1-18 of the '332 patent.

NetApp is willing to provide appropriate assistance to permit the Examiner to address and decide the issues presented by this Request. As the M.P.E.P. explains, the Examiner may, when appropriate, cut and paste claim charts or other material within the Request to incorporate them within the body of an Office Action. M.P.E.P. § 2262. NetApp is, thus, through the undersigned counsel, available to provide the Examiner with a digital copy of this Request, or any portion of it, in response to a request by email or phone. NetApp also understands that the Examiner may, in appropriate circumstances, set forth specific rejections in an Office Action and incorporate by reference NetApp's reasons for the proposed rejections, if the Examiner agrees with the proposed rejections and reasons supporting them.

If there is any additional fee due in connection with this filing that is not already submitted herewith, please charge the fee to Deposit Account No. 06-0916.

Respectfully submitted,
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP

Dated: September 3, 2025

By: /Cory C. Bell/
Cory C. Bell
Reg. No. 75,096

Attachments: (1) Certificate of Service to Patent Owner
(2) Information Disclosure Statement Form PTO/SB/08
(3) Exhibits listed on page iii of this Request