US 20150178113A1

(19) United States

(12) Patent Application Publication

Dake

(10) Pub. No.: US 2015/0178113 A1

(43) Pub. Date: Jun. 25, 2015

(54) **LOADING RUNTIME CONFIGURATION FILES INTO VIRTUAL MACHINE INSTANCES**

(71) Applicant: **Red Hat, Inc.**, Raleigh, NC (US)

(72) Inventor: **Steven Dake**, Scottsdale, AZ (US)

(73) Assignee: **Red Hat, Inc.**, Raleigh, NC (US)

(21) Appl. No.: **14/140,230**

(22) Filed: **Dec. 24, 2013**

Publication Classification

(51) **Int. Cl.**

G06F 9/455 (2006.01)

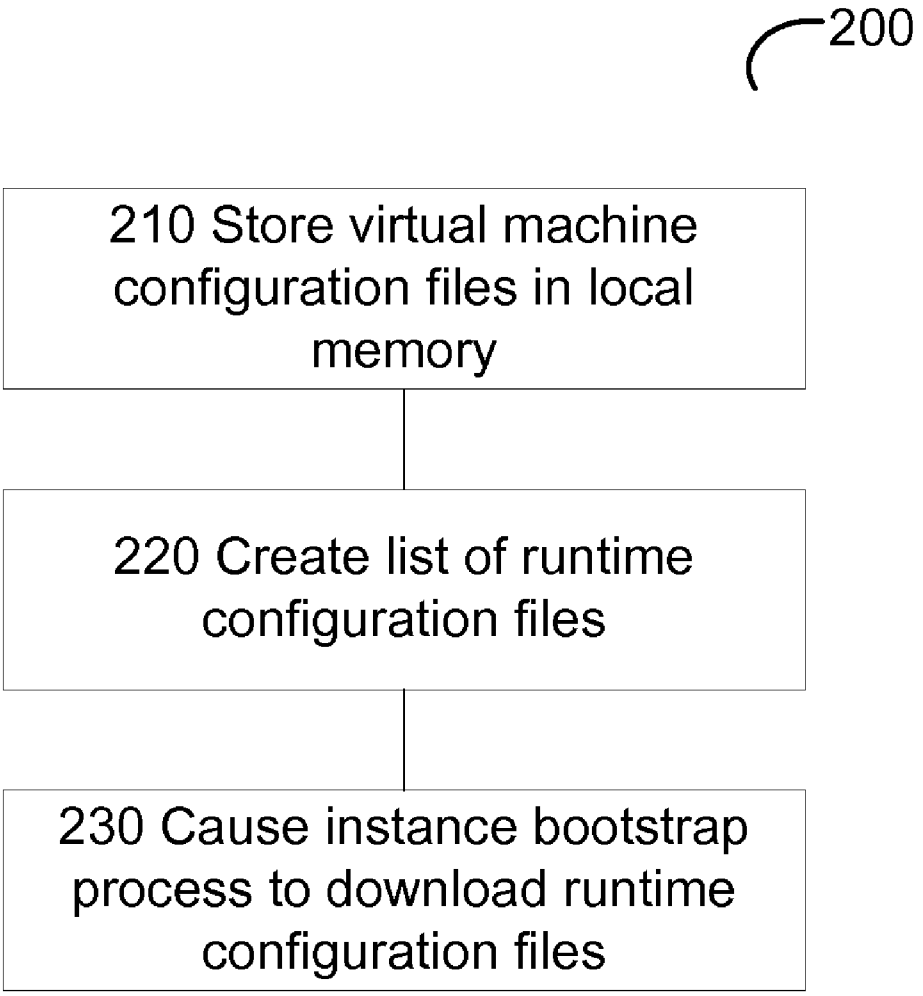
G06F 9/445 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 9/45533*** (2013.01); ***G06F 9/44505*** (2013.01)

(57) **ABSTRACT**

Systems and methods for loading runtime configuration files into virtual machine instances. An example method may comprise: storing, by a processing device, a plurality of virtual machine configuration files in a storage memory accessible by a virtual machine instance; creating a file list referencing a virtual machine configuration file of the plurality of virtual machine configuration files, the file list further specifying a target location of the virtual machine configuration file in the virtual machine instance; and causing a boot process of the virtual machine instance to download the virtual machine configuration file specified by the file list from the storage memory into the target location specified by the file list.



MICROSOFT CORP.
EXHIBIT 1006

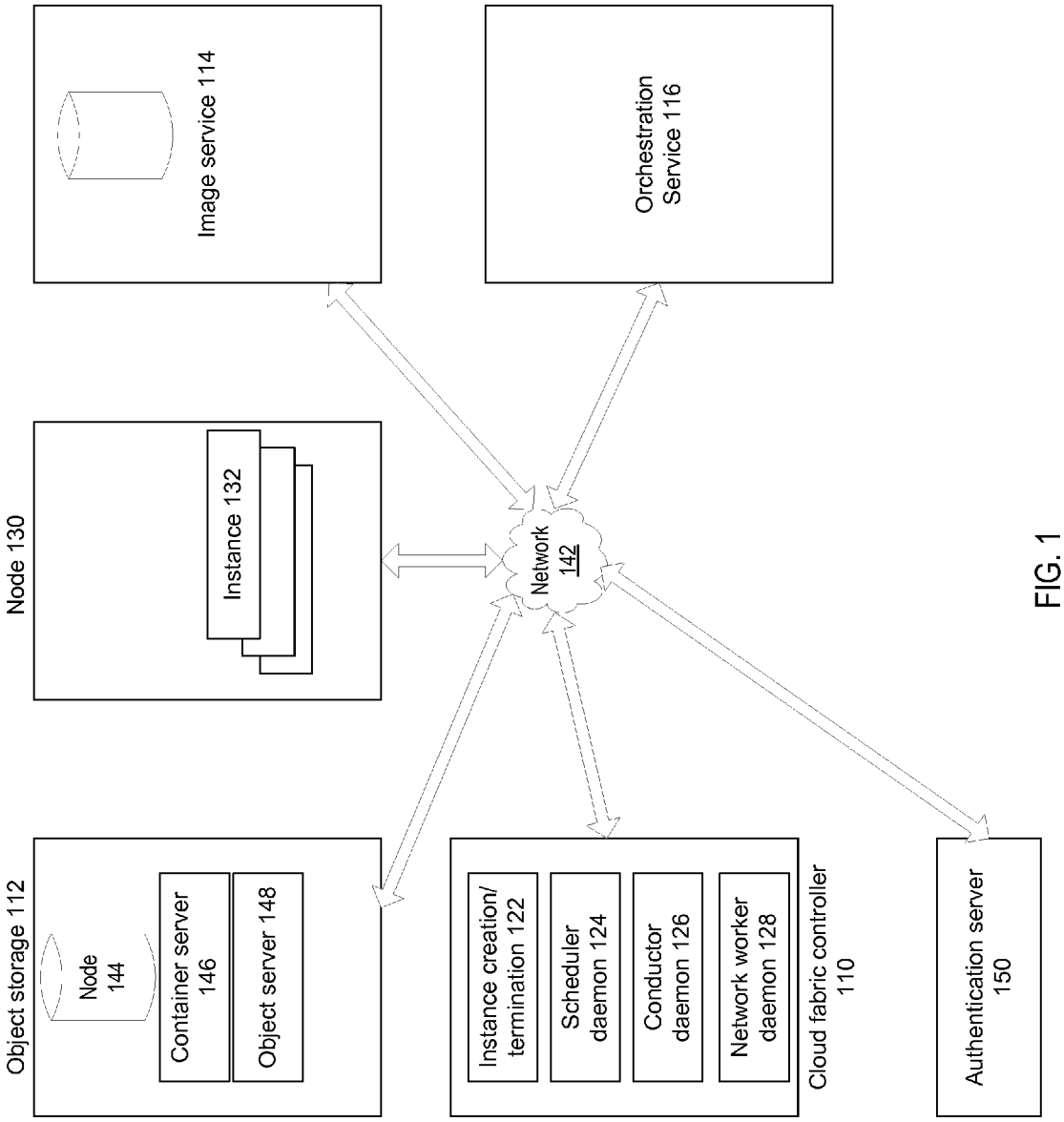


Image service 114
Orchestration Service 116
Node 130
Instance 132
Network 142
Object storage 112
Node 144
Container server 146
Object server 148
Instance creation/ termination 122
Scheduler daemon 124
Conductor daemon 126
Network worker daemon 128
Cloud fabric controller 110
Authentication server 150

FIG. 1

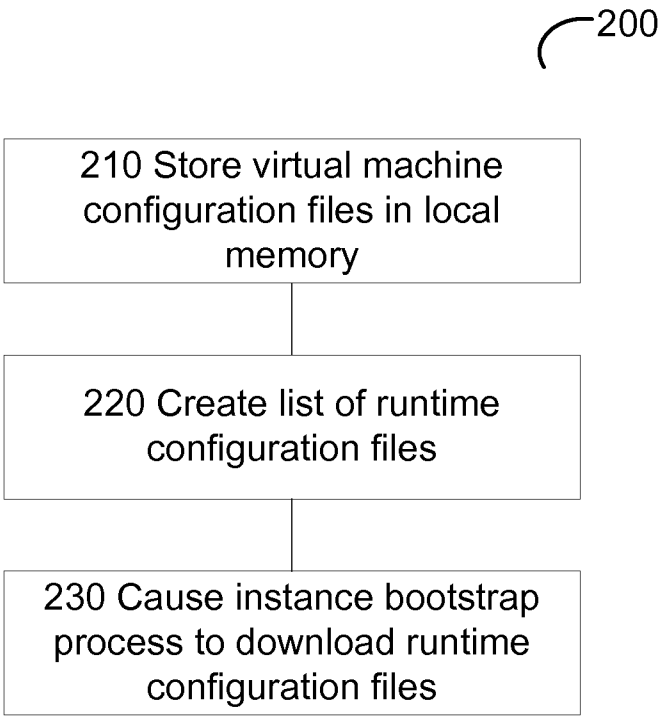

200
210 Store virtual machine configuration files in local memory
220 Create list of runtime configuration files
230 Cause instance bootstrap process to download runtime configuration files

FIG. 2

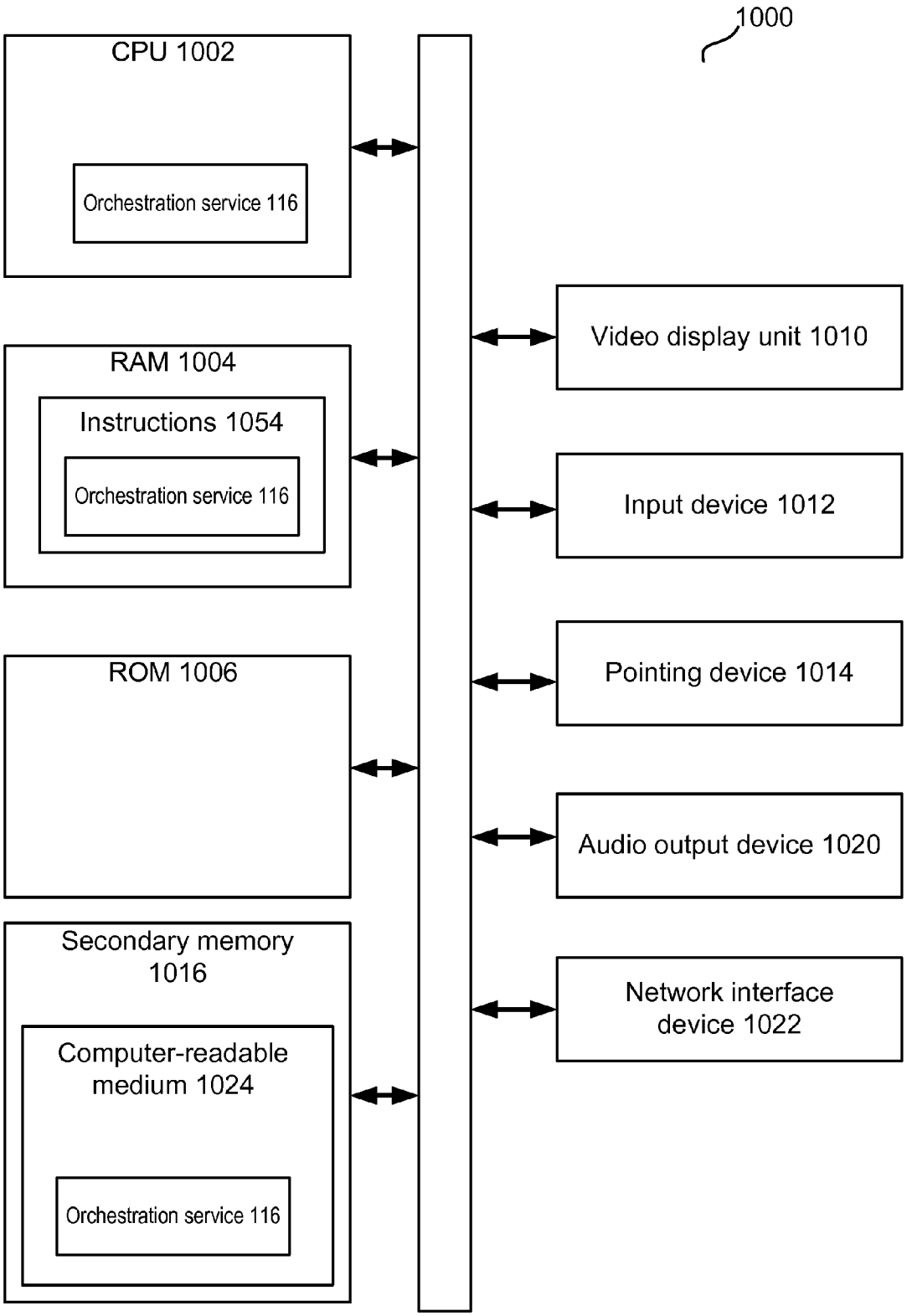

1000
CPU 1002
Orchestration service 116
RAM 1004
Instructions 1054
Orchestration service 116
ROM 1006
Secondary memory 1016
Computer-readable medium 1024
Orchestration service 116
Video display unit 1010
Input device 1012
Pointing device 1014
Audio output device 1020
Network interface device 1022

FIG. 3

LOADING RUNTIME CONFIGURATION FILES INTO VIRTUAL MACHINE INSTANCES

TECHNICAL FIELD

[0001] The present disclosure is generally related to cloud computing, and is more specifically related to loading runtime configuration files into virtual machine instances.

BACKGROUND

[0002] Cloud computing defines a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Essential characteristics of the cloud computing model include on demand self-service, broad network access, resource pooling, rapid elasticity and measured service. The cloud computing model comprises several service models, including Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The cloud computing model may be implemented according to one of the several deployment models, including private cloud, community cloud, public cloud and hybrid cloud.

[0003] Cloud infrastructure is a collection of hardware and software that implements the cloud computing model. Cloud infrastructure may be viewed as comprising a physical layer and an abstraction layer. The physical layer may comprise hardware resources designed to support the cloud services being provided, and typically includes server, storage and network components. The abstraction layer may comprise the software deployed across the physical layer, which manifests the essential cloud characteristics. Conceptually the abstraction layer resides above the physical layer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The present disclosure is illustrated by way of examples, and not by way of limitation, and may be more fully understood with references to the following detailed description when considered in connection with the figures, in which:

[0005] FIG. **1** schematically depicts component diagrams of an example cloud computing platform, in accordance with one or more aspects of the present disclosure;

[0006] FIG. **2** depicts a flow diagram of an illustrative example of a method for loading runtime configuration files into virtual machine instances, in accordance with one or more aspects of the present disclosure; and

[0007] FIG. **3** depicts an illustrative computer system operating in accordance with one or more aspects of the present disclosure.

DETAILED DESCRIPTION

[0008] Described herein are methods and systems for loading configuration files into virtual machine instances. In certain implementations, a configuration file may be provided by a runtime agent, e.g., a script file executable by a virtual machine instance to configure one or more virtual machine instance parameters. Alternatively, a configuration file may be provided by a data file comprising one or more values to be assigned to virtual machine instance parameters.

[0009] Using vendor-supplied distribution images (also referred to as "gold" images") of operating systems for building virtual machines in a cloud environment is not always practical, as the virtual machine images may need customization (e.g., custom software configuration). Uploading customized images to an image distribution service where those images may be accessed by the virtual machine instance building processes may not always be practical, e.g., due to the fact that a wide variety of such images may be required for a given cloud environment, thus resulting in a potentially very large number of customized images that would needed to be stored by the image distribution service.

[0010] To address the above noted and other deficiencies, the orchestration server may implement the method of loading runtime configuration files into images of virtual machine instances described herein. In accordance with one or more aspects of the present disclosure, runtime configuration files (e.g., in the form of executable script files or other binary applications executable by the virtual machine instance bootstrap process) may be stored in a storage memory which is locally accessible by virtual machine instances. For every instance to be built, a list of runtime configuration files may be compiled by the orchestration process. For every runtime configuration file, the list may specify its target location within the virtual machine instance. The list may be communicated to the virtual machine instance, causing the instance boot process to download the runtime configuration files into the target locations within the virtual machine instance.

[0011] Various aspects of the above referenced methods and systems are described in details herein below by way of examples, rather than by way of limitation.

[0012] FIG. **1** schematically depicts a high-level component diagram of an example cloud computing platform represented by a distributed computer system **1000**, in accordance with one or more aspects of the present disclosure.

[0013] In certain implementations, cloud computing platform **1000** may implement OpenStack platform. Cloud computing platform **1000** may comprise a cloud fabric controller service **110**, an object storage service **112**, an image service **114**, and an orchestration service **116**. While FIG. **1** illustrates each of the above listed services running on a dedicated hardware server, in certain implementations, two or more of those services may be collocated on a single hardware server. Servers **100** executing the above listed services may be interconnected by one or more networks **142**, including one or more local area networks, one or more wide area networks, or any combination thereof.

[0014] Cloud fabric controller service **110** may be employed for hosting and managing cloud computing systems. In the OpenStack platform, the cloud fabric controller functionality may be provided by Compute Service. The cloud fabric controller service may comprise several daemon processes, including a worker daemon **122** that creates and terminates virtual machine instances through hypervisor APIs, a scheduler daemon **124** that retrieves a virtual machine instance requests from a queue and assigns each request to a host computer, a conductor daemon **126** that manages interactions between worker daemon **122** and a cloud database, and a network worker daemon **128** that retrieves and performs networking tasks from a queue.

[0015] Object storage service **112** may provide a multi-tenant object storage system for large amounts of unstructured data. In certain implementations, object storage service **112** may implement a REST-compliant application program-

ming interface (API). In an illustrative example, HTTP protocol may be used as the transport layer for the API. In an illustrative example, object storage service **112** may comprise one or more storage nodes **144**, one or more container servers **146** to manage mappings of object containers, one or more object servers **148** to manage objects (such as files) on the storage nodes, and one or more authentication servers **150** to manage accounts defined within the object storage service.

[0016] Image service **114** may be employed to discover, register, and retrieve virtual machine images. The service includes a REST-compliant API that allows users to query virtual machine image metadata and retrieve images via HTTP requests. Virtual machine images made available through the image service may be stored in a variety of locations, including the object storage service, various file systems, Amazon S3 file system, HTTP access to remote Internet locations, and/or block devices.

[0017] Orchestration service **116** may be employed for combining virtual machines, networking elements, and other cloud resources into a running stack. Cloud computing platform **1000** may comprise one or more nodes **130** on which a plurality of virtual machines instances **132** may be instantiated. Cloud computing platform **1000** may provide command line-based, HTTP-based and/or API-based instance management.

[0018] In an illustrative example, orchestration service **116** may implement the method of loading runtime configuration files into images of virtual machine instances described herein. Orchestration service **116** may store a plurality of runtime configuration files (e.g., in the form of executable script files or other binary applications executable by the virtual machine instance bootstrap process) in a storage memory which is locally accessible by virtual machine instances. In certain implementations, the storage memory may be provided by object storage service **112** locally accessible by virtual machine instances.

[0019] For every instance to be built, orchestration service **116** may compile a list of runtime configuration files to be injected into the instance. For every runtime configuration file, the list may specify its target location (e.g., the full file system path and the filename) within the virtual machine instance, so that the virtual machine instance bootstrap process would be able to reference the runtime configuration file after the configuration file has been successfully injected into the virtual machine instances. In an illustrative example, the list may comprise a plurality of entries, each entry referencing a location from which the file may be retrieved and the target location of the file within the file system of the virtual machine instance. In an illustrative example, the file retrieval location may be specified, e.g., by an object identifier in object storage service **112**. In another illustrative example, the file retrieval location may be specified by a network file system (NFS) path and file name, assuming that the referenced NFS volume would be mounted by the virtual machine instance bootstrap process before accessing the NFS location.

[0020] Upon compiling the configuration file list, orchestration service **116** may communicate the list to the virtual machine instance, thus causing the instance bootstrap process to download the runtime configuration files into the target locations within the virtual machine instance. In an illustrative example, the instance bootstrap process may be configured to retrieve the list via the User Data interface.

[0021] User Data is a mechanism allowing a virtual machine instance to retrieve one or more locally stored files at the instance launch time. The instance can retrieve the files by issuing an HTTP GET request to a pre-defined URI, for example: http://169.254.169.254/user-data.

[0022] Upon retrieving the list, the virtual machine instance bootstrap process may process each list element by downloading the specified runtime file and placing the file into the location specified by the list. After the configuration files have been successfully downloaded and placed into the specified locations, the virtual machine instance bootstrap process may use the downloaded files, e.g., by executing the runtime agent scripts and/or retrieving configuration data from the downloaded files.

[0023] In certain implementations, the orchestration process may employ the User Data mechanism for transferring the runtime configuration files to the virtual machine instances. Instead of placing the configuration files into the local object storage, the orchestration service may put the files into a local file system employed by the User Data service, so that the virtual machine instance would be able to retrieve the files at the instance launch time, as described herein above.

[0024] FIG. **2** depicts a flow diagram of an example method **200** performed by an orchestration service for loading runtime configuration files into virtual machine instances. Method **200** and/or each of its individual functions, routines, subroutines, or operations may be performed by one or more processors of the computer system (e.g., orchestration server **116** of FIG. **1**) executing the method. In certain implementations, method **200** may be performed by a single processing thread. Alternatively, method **200** may be performed by two or more processing threads, each thread executing one or more individual functions, routines, subroutines, or operations of the method. In an illustrative example, the processing threads implementing method **200** may be synchronized (e.g., using semaphores, critical sections, and/or other thread synchronization mechanisms). Alternatively, the processing threads implementing method **200** may be executed asynchronously with respect to each other.

[0025] At block **210**, the orchestration service may store a plurality of virtual machine configuration files in a storage memory accessible by a virtual machine instance. In certain implementations, a configuration file may be provided by a runtime agent, e.g., a script file or other binary application executable by a virtual machine instance. In an illustrative example, the storage memory may be provided by an object storage, as described in more details herein above.

[0026] In certain implementations, the orchestration process may employ the User Data mechanism for transferring the runtime configuration files to the virtual machine instances, as described in more details herein above.

[0027] At block **220**, the orchestration service may create a file list referencing one or more configuration files of the plurality of configuration files stored in the local storage memory, to be injected into a virtual machine instance. For every virtual machine configuration file, the list may specify its target location (e.g., the full file system path and the filename) within the virtual machine instance, so that the virtual machine instance bootstrap process would be able to reference the runtime configuration file after the configuration file has been successfully injected into the virtual machine instances, as described in more details herein above.

[0028] At block **230**, the orchestration service may cause virtual machine instance bootstrap process to download the virtual machine configuration files specified by the file list

from the storage memory into the target locations specified by the file list, as described in more details herein above.

[0029] Upon completing the operations referenced by block **230**, the method may terminate.

[0030] FIG. **3** depicts an illustrative computer system operating in accordance with examples of the present disclosure. In illustrative examples, computer system **1000** may correspond to orchestration server **116** of FIG. **1**.

[0031] In one example, computer system **1000** may be connected (e.g., via a network, such as a Local Area Network (LAN), an intranet, an extranet, or the Internet) to other computer systems (e.g., other nodes). Computer system **1000** may operate in the capacity of a server or a client computer in a client-server environment, or as a peer computer in a peer-to-peer or distributed network environment. Computer system **1000** may be provided by a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, switch or bridge, or any device capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that device. Further, the term "computer" shall include any collection of computers that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methods described herein.

[0032] In a further aspect, computer system **1000** may include a processor **1002**, a volatile memory **1004** (e.g., random access memory (RAM)), a non-volatile memory **1006** (e.g., read-only memory (ROM) or electrically-erasable programmable ROM (EEPROM)), and a storage memory **1016** (e.g., a data storage device), which may communicate with each other via a bus **1008**.

[0033] Processor **1002** may be provided by one or more processors such as a general purpose processor (such as, for example, a complex instruction set computing (CISC) microprocessor, a reduced instruction set computing (RISC) microprocessor, a very long instruction word (VLIW) microprocessor, a microprocessor implementing other types of instruction sets, or a microprocessor implementing a combination of types of instruction sets) or a specialized processor (such as, for example, an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), or a network processor).

[0034] Computer system **1000** may further include a network interface device **1022**. Computer system **1000** also may include a video display unit **1010** (e.g., an LCD), an alphanumeric input device **1012** (e.g., a keyboard), a pointing device **1014** (e.g., a mouse), and an audio output device **1020** (e.g., a speaker).

[0035] In an illustrative example, secondary memory **1016** may include a tangible computer-readable storage medium **1024** on which may be stored instructions **1054** encoding orchestration service **116** of FIG. **1** implementing method **200** for loading runtime configuration files into virtual machine instances in accordance with one or more aspect of the present disclosure. Instructions **1054** may also reside, completely or partially, within main memory **1004** and/or within processor **1002** during execution thereof by computer system **1000**, hence, main memory **1004** and processor **1002** may also constitute machine-readable storage media.

[0036] While computer-readable storage medium **1024** is shown in the illustrative examples as a single medium, the term "computer-readable storage medium" shall include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of executable instructions. The term "computer-readable storage medium" shall also include any tangible medium that is capable of storing or encoding a set of instructions for execution by a computer that cause the computer to perform any one or more of the methods described herein. The term "computer-readable storage medium" shall include, but not be limited to, solid-state memories, optical media, and magnetic media.

[0037] The methods, components, and features described herein may be implemented by discrete hardware components or may be integrated in the functionality of other hardware components such as ASICS, FPGAs, DSPs or similar devices. In addition, the methods, components, and features may be implemented by firmware modules or functional circuitry within hardware devices. Further, the methods, components, and features may be implemented in any combination of hardware devices and software components, or only in software.

[0038] Unless specifically stated otherwise, terms such as "updating", "identifying", "determining", "sending", "assigning", or the like, refer to actions and processes performed or implemented by computer systems that manipulates and transforms data represented as physical (electronic) quantities within the computer system registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0039] Examples described herein also relate to an apparatus for performing the methods described herein. This apparatus may be specially constructed for performing the methods described herein, or it may comprise a general purpose computer system selectively programmed by a computer program stored in the computer system. Such a computer program may be stored in a computer-readable tangible storage medium.

[0040] The methods and illustrative examples described herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used in accordance with the teachings described herein, or it may prove convenient to construct more specialized apparatus to perform method **200** and/or each of its individual functions, routines, subroutines, or operations. Examples of the structure for a variety of these systems are set forth in the description above.

[0041] The above description is intended to be illustrative, and not restrictive. Although the present disclosure has been described with references to specific illustrative examples and implementations, it will be recognized that the present disclosure is not limited to the examples and implementations described. The scope of the disclosure should be determined with reference to the following claims, along with the full scope of equivalents to which the claims are entitled.

What is claimed is:

1. A method, comprising:

storing, by a processing device, a plurality of virtual machine configuration files in a storage memory accessible by a virtual machine instance;

creating a file list referencing a virtual machine configuration file of the plurality of virtual machine configuration files, the file list further specifying a target location of the virtual machine configuration file in the virtual machine instance; and

causing a boot process of the virtual machine instance to download the virtual machine configuration file specified by the file list from the storage memory into the target location specified by the file list.

2. The method of claim **1**, wherein the virtual machine configuration file is provided by one of: an executable file configured to modify a parameter of the virtual machine instance or a data file comprising a parameter of the virtual machine instance.

3. The method of claim **1**, wherein the storage memory is accessible by the virtual machine instance via a local area network.

4. The method of claim **1**, wherein the storage memory is provided by a file system.

5. The method of claim **1**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to receive the virtual machine configuration file via a local area network connection to a pre-defined network address.

6. The method of claim **1**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to download the virtual machine configuration file from a pre-defined file system location.

7. The method of claim **1**, wherein the storage memory comprises a plurality of object containers.

8. The method of claim **1**, wherein the storage memory is accessible by the boot process via a local area network connection to a pre-defined network address.

9. The method of claim **1**, further comprising: creating the virtual machine configuration file before storing the plurality of virtual machine configuration files.

10. A system, comprising:

a memory; and

one or more processors, coupled to the memory, to:

store a plurality of virtual machine configuration files in a storage memory accessible by a virtual machine instance;

creating a file list referencing a virtual machine configuration file of the plurality of virtual machine configuration files, the file list further specifying a target location of the virtual machine configuration file in the virtual machine instance; and

causing a boot process of the virtual machine instance to download the virtual machine configuration file specified by the file list from the storage memory into the target location specified by the file list.

11. The system of claim **10**, wherein the virtual machine configuration file is provided by one of: an executable file configured to modify a parameter of the virtual machine instance or a data file comprising a parameter of the virtual machine instance.

12. The system of claim **10**, wherein the storage memory is accessible by the virtual machine instance via a local area network.

13. The system of claim **10**, wherein the storage memory is provided by a file system.

14. The system of claim **10**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to receive the virtual machine configuration file via a local area network connection to a pre-defined network address.

15. The system of claim **10**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to download the virtual machine configuration file from a pre-defined file system location.

16. The system of claim **10**, wherein the storage memory comprises a plurality of object containers.

17. The system of claim **10**, wherein the storage memory is accessible by the boot process via a local area network connection to a pre-defined network address.

18. A non-transitory computer-readable storage medium comprising executable instructions that, when executed by a processing device of a file system server, cause the processing device to perform operations, comprising:

storing a plurality of virtual machine configuration files in a storage memory accessible by a virtual machine instance;

creating a file list referencing a virtual machine configuration file of the plurality of virtual machine configuration files, the file list further specifying a target location of the virtual machine configuration file in the virtual machine instance; and

causing a boot process of the virtual machine instance to download the virtual machine configuration file specified by the file list from the storage memory into the target location specified by the file list.

19. The non-transitory computer-readable storage medium of claim **18**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to receive the virtual machine configuration file via a local area network connection to a pre-defined network address.

20. The non-transitory computer-readable storage medium of claim **18**, wherein causing the boot process to download the virtual machine configuration file comprises causing the boot process to download the virtual machine configuration file from a predefined file system location.

* * * * *