



(51) International Patent Classification:

G06F 11/36 (2006.01) G06F 11/00 (2006.01)
G06F 21/50 (2013.01) G06F 21/00 (2013.01)

(21) International Application Number:

PCT/US2022/052868

(22) International Filing Date:

14 December 2022 (14.12.2022)

(25) Filing Language:

English

(26) Publication Language:

English

(71) Applicants: **ROBIN SYSTEMS, INC** [US/US]; 2001 Gateway Place Suite 340E, San Jose, CA 95110 (US). **ROBIN SOFTWARE DEVELOPMENT CENTER INDIA PRIVATE LIMITED** [US/US]; Wework, Embassy Tech Village Block L, Devarabisanahalli Outer Ring Road Bellandur, Bangalore 560103 (IN).

(72) Inventors: **MISHRA, Ashok, Kumar**; 2001 Gateway Place Suite 340e, San Jose, CA 95110 (US). **SINGH, Ramendra, Pratap**; Wework, Embassy Tech Village Block L, Devarabisanahalli Outer Ring Road Bellandur, Bangalore 560103 (IN).

(74) Agent: **STEVENS, David, R.**; Stevens Law Group, 1754 Technology Drive, Suite 226, San Jose, CA 95110 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) Title: LOCAL PROTECT IMAGE FOR CRITICAL APPLICATIONS

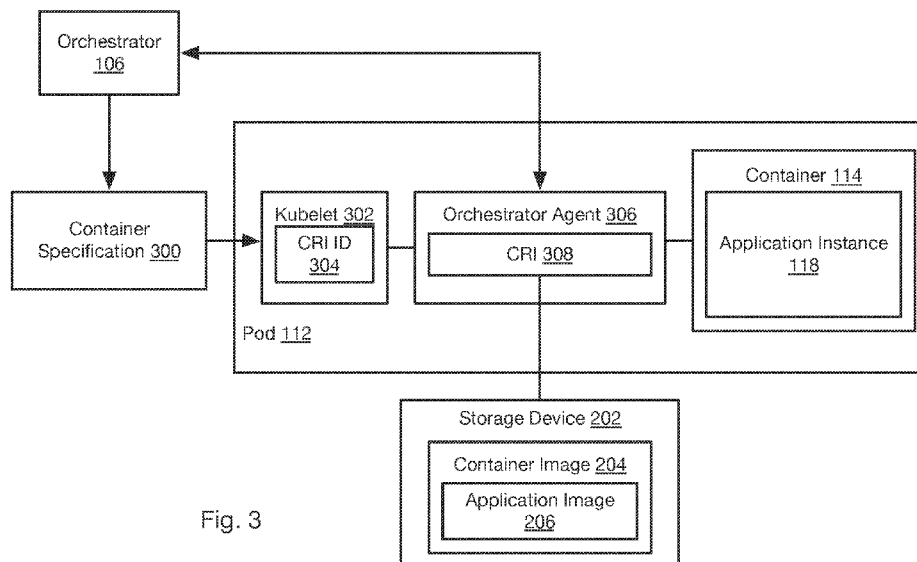


Fig. 3

(57) Abstract: A critical executable image is used to instantiate an instance on a computing device. The executable image is further saved on the storage device. In response to a failure of the instance, the instance may be restored from the locally stored executable image. The executable image may be of a container and corresponding application image. The computing device may receive a specification, e.g., container specification, to instantiate the instance, the specification including an annotation instructing the computing device to locally store the executable image after instantiation.

Published:

— *with international search report (Art. 21(3))*

Title: LOCAL PROTECT IMAGE FOR CRITICAL APPLICATIONS

BACKGROUND

FIELD OF THE INVENTION

[001] This invention relates to providing a local protect image for critical applications.

BACKGROUND OF THE INVENTION

[002] Many computing installations are extremely complex and require many components executing on many different host computing devices to operate and interoperate correctly. Tools may be used to automatically manage an installation and perform tasks such as monitoring, scaling up, scaling down, and handling failures. Some components of a computing installation are particularly critical such that constant availability should be provided.

[003] It would be an advancement in the art to facilitate the availability of critical components of a computing installation.

SUMMARY OF THE INVENTION

[004] An apparatus includes a computing device including one or more processing devices and one or more memory devices operably coupled to the one or more processing devices. The one or more memory devices store executable code that, when executed by the one or more processing devices, causes the one or more processing devices to receive an instruction to instantiate an instance from an executable image, the instruction including an annotation. The instance is instantiated from the executable image. In response to the annotation, the executable image is stored on a storage device coupled to

the one or more processing devices after instantiating the instance from the executable image.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be described and explained with additional specificity and detail through use of the accompanying drawings, in which:

[006] Fig. 1 is a schematic block diagram of a network environment in which a complex application may be deployed in accordance with an embodiment;

[007] Fig. 2 is a schematic block diagram showing components for providing a local protect image in accordance with an embodiment;

[008] Fig. 3 is process flow diagram of a components for instantiating from a local protect image in accordance with an embodiment;

[009] Fig. 4A is a process flow diagram of a method for creating and using a local protect image in accordance with an embodiment;

[0010] Fig. 4B is a process flow diagram of an alternative method for creating and using a local protect image in accordance with an embodiment; and

[0011] Fig. 5 is a schematic block diagram of an example computing device suitable for implementing methods in accordance with embodiments of the invention.

DETAILED DESCRIPTION

[0012] Fig. 1 illustrates an example network environment 100 in which the systems and methods disclosed herein may be used. The components of the network environment 100 may be connected to one another by a network such as a local area network (LAN), wide area network (WAN), the Internet, a backplane of a chassis, or other type of network. The components of the network environment 100 may be connected by wired or wireless network connections. The network environment 100 includes a plurality of servers 102. Each of the servers 102 may include one or more computing devices, such as a computing device having some or all of the attributes of the computing device 500 of Fig. 5.

[0013] Computing resources may also be allocated and utilized within a cloud computing platform 104, such as amazon web services (AWS), GOOGLE CLOUD, AZURE, or other cloud computing platform. Cloud computing resources may include purchased physical storage, processor time, memory, and/or networking bandwidth in units designated by the provider by the cloud computing platform.

[0014] In some embodiments, some or all of the servers 102 may function as edge servers in a telecommunication network. For example, some or all of the servers 102 may be coupled to baseband units (BBU) 102a that provide translation between radio frequency signals output and received by antennas 102b and digital data transmitted and received by the servers 102. For example, each BBU 102a may perform this translation according to a cellular wireless data protocol (e.g., 4G, 5G, etc.). Servers 102 that function as edge servers may have limited computational resources or may be heavily loaded.

[0015] An orchestrator 106 provisions computing resources to application instances 118 of one or more different application executables, such as according to a

manifest that defines requirements of computing resources for each application instance. The manifest may define dynamic requirements defining the scaling up or scaling down of a number of application instances 118 and corresponding computing resources in response to usage. The orchestrator 106 may include or cooperate with a utility such as KUBERNETES to perform dynamic scaling up and scaling down the number of application instances 118.

[0016] An orchestrator 106 may execute on a computer system that is distinct from the servers 102 and is connected to the servers 102 by a network that requires the use of a destination address for communication, such as using a networking including ethernet protocol, internet protocol (IP), Fibre Channel, or other protocol, including any higher-level protocols built on the previously-mentioned protocols, such as user datagram protocol (UDP), transport control protocol (TCP), or the like.

[0017] The orchestrator 106 may cooperate with the servers 102 to initialize and configure the servers 102. For example, each server 102 may cooperate with the orchestrator 106 to obtain a gateway address to use for outbound communication and a source address assigned to the server 102 for use in inbound communication. The server 102 may cooperate with the orchestrator 106 to install an operating system on the server 102. For example, the gateway address and source address may be provided and the operating system installed using the approach described in U.S. Application Serial No. 16/903,266, filed June 16, 2020 and entitled AUTOMATED INITIALIZATION OF SERVERS, which is hereby incorporated herein by reference in its entirety.

[0018] The orchestrator 106 may be accessible by way of an orchestrator dashboard 108. The orchestrator dashboard 108 may be implemented as a web server or other server-

side application that is accessible by way of a browser or client application executing on a user computing device 110, such as a desktop computer, laptop computer, mobile phone, tablet computer, or other computing device.

[0019] The orchestrator 106 may cooperate with the servers 102 in order to provision computing resources of the servers 102 and instantiate components of a distributed computing system on the servers 102 and/or on the cloud computing platform 104. For example, the orchestrator 106 may ingest a manifest defining the provisioning of computing resources to, and the instantiation of, components such as a cluster 111, pod 112 (e.g., KUBERNETES pod), container 114 (e.g., DOCKER container), storage volume 116, and an application instance 118. The orchestrator may then allocate computing resources and instantiate the components according to the manifest.

[0020] The manifest may define requirements such as network latency requirements, affinity requirements (same node, same chassis, same rack, same data center, same cloud region, etc.), anti-affinity requirements (different node, different chassis, different rack, different data center, different cloud region, etc.), as well as minimum provisioning requirements (number of cores, amount of memory, etc.), performance or quality of service (QoS) requirements, or other constraints. The orchestrator 106 may therefore provision computing resources in order to satisfy or approximately satisfy the requirements of the manifest.

[0021] The instantiation of components and the management of the components may be implemented by means of workflows. A workflow is a series of tasks, executables, configuration, parameters, and other computing functions that are predefined and stored in a workflow repository 120. A workflow may be defined to instantiate each type of

component (cluster 111, pod 112, container 114, storage volume 116, application instance, etc.), monitor the performance of each type of component, repair each type of component, upgrade each type of component, replace each type of component, copy (snapshot, backup, etc.) and restore from a copy each type of component, and other tasks. Some or all of the tasks performed by a workflow may be implemented using KUBERNETES or other utility for performing some or all of the tasks.

[0022] The orchestrator 106 may instruct a workflow orchestrator 122 to perform a task with respect to a component. In response, the workflow orchestrator 122 retrieves the workflow from the workflow repository 120 corresponding to the task (e.g., the type of task (instantiate, monitor, upgrade, replace, copy, restore, etc.) and the type of component. The workflow orchestrator 122 then selects a worker 124 from a worker pool and instructs the worker 124 to implement the workflow with respect to a server 102 or the cloud computing platform 104. The instruction from the orchestrator 106 may specify a particular server 102, cloud region or cloud provider, or other location for performing the workflow. The worker 124, which may be a container, then implements the functions of the workflow with respect to the location instructed by the orchestrator 106. In some implementations, the worker 124 may also perform the tasks of retrieving a workflow from the workflow repository 120 as instructed by the workflow orchestrator 122. The workflow orchestrator 122 and/or the workers 124 may retrieve executable images for instantiating components from an image store 126.

[0023] Referring to Fig. 2, a host 200 may be a server 102, a unit of computing resources on the cloud computing platform 104, a virtual machine, or other computing device. A host 200 may have one or more storage devices 202 mounted thereto. Each

storage device 202 may be implemented as a hard disc drive (HDD), solid state drive (SSD), or other type of persistent storage device. A storage device 202 may be physically mounted to a host 200 or may be coupled to the host 200 by a network.

[0024] A container 114 and corresponding application instance 118 may be instantiated on a host 200, such as by a pod 112 for the host 200. The container 114 and application instance 118 may be instantiated from a container image 204 including an application image 206. The container image 204 may be retrieved from the image store 126. The storage device 202 may store a storage volume 116 that is mounted to the container 114 and is accessed by the application instance 118.

[0025] For critical applications instances 118, the container image 204, including the application image 206, may also be written to the storage device 202 to provide a local protect image for rapid recovery. In this manner, the container 114 and application instance 118 may be restored from the storage device 202 without the delay caused by requesting and transmitting the container image 204 and application image 206. Likewise, the container 114 and application instance 118 may be restored in the absence of a network connection to the image store 126.

[0026] A critical application instance 118 may be identified in various ways. For example, an administrator may input an explicit designation of the application instance 118 as critical. An application instance 118 may be designated as critical where the application instance 118 lacks redundancy. For example, where the application instance 118 has no other application instances 118 that provide the same function, such as in a load balancing arrangement or where the number of other application instances 118 in the load balancing arrangement is below a threshold number.

[0027] An application instance 118 may be designated as critical where many other components of the cluster are dependent thereon. For example, an application instance 118 may be deemed critical if the application instance 118 has the greatest number of dependencies for the cluster 111, has a number of dependencies exceeding a threshold amount, or is one of the N application instances 118 of the cluster 111 with the highest number of dependencies, where N is a predefined integer.

[0028] Dependencies may be in the form of application sessions to the application instance 118, network connections to the application instance 118, environmental variables of components referring to the application instance 118, or other dependencies. Dependencies may include indirect dependencies. A component that is dependent on a component that is dependent on the application instance 118 may be deemed dependent on the application instance 118.

[0029] Dependencies may be determined by examining observability data such as log files, metrics, events, alerts, inventory, and other data that may record a dependency of a component on the application instance 118. Dependencies may also be defined in the manifest ingested by the orchestrator 106 when instantiating cluster and may therefore be derived from the manifest.

[0030] Fig. 3 illustrates an approach for storing a local copy of the container image 204 and application image 206 in the storage device of a host 200 executing the container 114 and application instance 118 instantiated from the container image 204 and application image 206.

[0031] The orchestrator 106 may provide a container specification 300 to a pod 112, e.g., to a KUBERNETES Kubelet 302 implementing the pod 112. The Kubelet 302

may function as a logical host for one or more containers 114 of the pod 112. The pod 112 may include a set of namespaces, a file system (e.g., built on a storage volume 116), or other data structures that are shared by containers 114 belonging to the pod 112.

[0032] The container specification 300 may include the container image 204 and application image 206 or may include an identifier of one or both of the container image 204 and application image 206. The container specification 300 may include an annotation indicating that the container image 204 and the application image 206 are to be locally stored in the storage device 202.

[0033] In response to receiving the container specification, the Kubelet 302 may call a container runtime interface (CRI). A CRI is an executable corresponding to a particular type of container 114 and configured to perform tasks for that type of container 114, such as instantiating the container 114, suspending the container 114, pausing the container 114, de-instantiating the container, loading the application instance 118 into the container 114, or performing other tasks with respect to the container 114.

[0034] For example, the Kubelet 302 may be configured with a CRI identifier 304, e.g., a pointer, path name, or other reference, that will be used by the Kubelet 302 to invoke CRI functions. In some embodiments, the CRI identifier 304 may refer to an orchestrator agent 306 that implements a CRI 308 as well as being configured to coordinate with the orchestrator 106 to perform actions beyond those performed by a conventional CRI.

[0035] For example, the Kubelet 302 may instruct the orchestrator agent 306 to instantiate the container 114 and application instance 118 in response to receiving the container specification 300. The Kubelet 302 may pass the annotation to the orchestrator agent 306. The orchestrator agent 306 may then instantiate the container 114 and

application instance 118 by either using the container image 204 and application image 206 included in the container specification or retrieving the container image 204 and application image 206 from the image store 126 using one or more identifiers included in the container specification 300.

[0036] In response to the annotation, the orchestrator agent 306 further stores a copy of the container image 204 and application image 206 in the storage device 202 of the host 200 executing the pod 112.

[0037] In one approach, the container image 204 and application image 206 are stored in the storage device 202 before the container specification 300 is received. For example, the orchestrator 106 may communicate with the orchestrator agent 306 in bypass of the Kubelet 302 and instruct the orchestrator agent 306 to retrieve the container image 204 and application image 206 and store them in the storage device 202. The orchestrator 106 may then transmit the container specification 300 to the Kubelet 302, the container specification directing the Kubelet 302 to instantiate the container 114 and application instance 118 from the copy of the container image 204 and application image 206 on the storage device 202.

[0038] In some embodiments, the orchestrator agent 306 may be omitted: the orchestrator 106 invokes transfer of the container image 204 and application image 206 to the storage device 202 directly, such as using a remote command line interface (e.g., secure shell (SSH)) or other remote procedure call. The orchestrator 106 may then transmit the container specification 300 to the Kubelet 302, the container specification directing the Kubelet 302 to instantiate the container 114 and application instance 118 from the copy of the container image 204 and application image 206 on the storage device 202. The Kubelet

302 may then call a CRI to instantiate the container 114 and application instance 118 from the copy of the container image 204 and application image 206 on the storage device 202 without using an orchestrator agent 306.

[0039] If the container 114 fails or the host 200 is restarted, the Kubelet 302 may again read the specification 300 and instruct the orchestrator agent 306 to instantiate the container 114 and application instance 118 from the copy of the container image 204 and application image 206 on the storage device 202. The round-trip time required to request and retrieve the container image 204 and application image 206 is therefore avoided. Likewise, the container 114 and application instance 118 may be restored even when there is no network connection between the host 200 and the image store 126 or the image store 126 is otherwise unavailable.

[0040] Fig. 4A illustrates a method 400a for creating a local protect image for a container image 204 and application image 206. The method 400a may be preceded by a determination that the container 114 and application instance 118 to be instantiated are critical as described above.

[0041] The method 400a may include the orchestrator 402 invoking container instantiation in a pod 112, such as by transmitting a container specification 300 to the Kubelet 302 of the pod 112. The container image 204 (which includes the application image 206) referenced by the container specification is retrieved 404 from the image store 126 and transmitted 406 by the image store 126 to the host 200 of the pod 112 either directly or by way of some other entity, such as the orchestrator 106 or some other component. The container image 204 may be retrieved by the orchestrator 106 and transmitted with the container specification 300, requested by the Kubelet 302 from the image store 126 in

response to the container specification 300, or requested by the orchestrator agent 306 from the image store 126 in response to the annotation to the container specification 300.

[0042] A component executing on the host 200 may then store 408 the container image 204 on a storage device 202 of the host 200. The orchestrator agent 306 may store the container image 204 on the storage device as may the Kubelet 302 itself or the orchestrator 106.

[0043] A component executing on the host 200 may then instantiate 410 the container 114, including the application instance 118 hosted by the container 114, using the container image 204. The container 114 and application instance 118 may be instantiated 410 by the orchestrator agent 306, a CRI, or some other component. The container 114 and application instance 118 may then commence operating to perform tasks the application instance 118 is programmed to perform.

[0044] The method 400a may further include detecting 412 failure of any of the container 114, the application instance 118, the pod 112, or the host 200 itself. In the case of failure of the host 200, failure may be detected by the cluster 111 or the orchestrator 106, such as using a monitoring workflow from the workflow repository 120. Failure of the host 200 may be detected by a component on the host 200, e.g., the Kubelet 302 or orchestrator agent 306, detecting restarting of the host 200.

[0045] In response to detecting 412 failure, a component executing on the host 200 may then retrieve 414 the container image 204 from the storage device 202 and instantiate 416 the container 114 and corresponding application instance 118 using the container image 204. Instantiating 416 may be performed by the orchestrator agent 306 independently or upon being invoked as the CRI of the Kubelet 302. Instantiating 416 may

also be invoked by the orchestrator 106 in response to detecting 412 failure.

[0046] Fig. 4B is an alternative method 400b that may be used for creating a local protect image for a container image 204 and application image 206. The method 400b may be performed without the use of an orchestrator agent 306.

[0047] In the method 400b, the orchestrator invokes 418 transfer of the container image 204, by the image store 126, to a storage device 202 of a host 200. The image store 126 then transmits 420 the container image 204 to the host 200 and causes the host 200 to store 422 the container image 204 on the storage device 202. For example, a workflow from the workflow repository 120 may be used to achieve storage of the container image 204 on the storage device 202. For example, the workflow may use a remote command line interface (CLI), such as SSH, to transfer and store the container image 204 on the storage device 202.

[0048] The orchestrator 424 may then invoke 424 a component on the host 200, such as the Kubelet 302, to instantiate the container 114 and application instance 118 from the previously-stored container image 204. The component then instantiates 426 the container 114 and application instance 118 from the container image 204 stored on the storage device 202. For example, the Kubelet 302 may call a CRI to instantiate the container 114 and application instance 118.

[0049] In the event a failure is detected 428, re-instantiating the container 114 and application instance 118 may be performed by repeating steps 424-426. For example, the orchestrator 106 may invoke 426 container instantiation in response to detecting failure and restarting of the host 200. In some scenarios, the Kubelet 302 may invoke the CRI without an instruction from the orchestrator 106, such as in view of a previously received

container specification 300 instructing instantiation of the container 114 and application instance from the container image 204 stored on the storage device 202. For example, the Kubelet 302 may do so in response to failure of the container image 204 or upon starting up after restarting of the host 200 following failure or other event.

[0050] In yet another alternative to the methods 400a, 400b, the local protect image stored on the storage device 202 may be stored at some point following instantiation of the container 114 and application instance 118. For example, rather than deleting the container image 204 following instantiation, the orchestrator agent 306 or some other component may store the container image 204 on the storage device 202 and configure the Kubelet 302 to use the container image 204 to recover from failure of the host 200, container 114, and/or application instance 118.

[0051] Fig. 5 is a block diagram illustrating an example computing device 500. Computing device 500 may be used to perform various procedures, such as those discussed herein. The servers 102, orchestrator 106, workflow orchestrator 122, and cloud computing platform 104 may each be implemented using one or more computing devices 500. The orchestrator 106 and workflow orchestrator 122 may be implemented on different computing devices 500 or a single computing device 500 may execute both of the orchestrator 106 and workflow orchestrator 122.

[0052] Computing device 500 includes one or more processor(s) 502, one or more memory device(s) 504, one or more interface(s) 506, one or more mass storage device(s) 508, one or more Input/output (I/O) device(s) 510, and a display device 530 all of which are coupled to a bus 512. Processor(s) 502 include one or more processors or controllers that execute instructions stored in memory device(s) 504 and/or mass storage device(s)

508. Processor(s) 502 may also include various types of computer-readable media, such as cache memory.

[0053] Memory device(s) 504 include various computer-readable media, such as volatile memory (e.g., random access memory (RAM) 514) and/or nonvolatile memory (e.g., read-only memory (ROM) 516). Memory device(s) 504 may also include rewritable ROM, such as Flash memory.

[0054] Mass storage device(s) 508 include various computer readable media, such as magnetic tapes, magnetic disks, optical disks, solid-state memory (e.g., Flash memory), and so forth. As shown in Fig. 5, a particular mass storage device is a hard disk drive 524. Various drives may also be included in mass storage device(s) 508 to enable reading from and/or writing to the various computer readable media. Mass storage device(s) 508 include removable media 526 and/or non-removable media.

[0055] I/O device(s) 510 include various devices that allow data and/or other information to be input to or retrieved from computing device 500. Example I/O device(s) 510 include cursor control devices, keyboards, keypads, microphones, monitors or other display devices, speakers, printers, network interface cards, modems, lenses, CCDs or other image capture devices, and the like.

[0056] Display device 530 includes any type of device capable of displaying information to one or more users of computing device 500. Examples of display device 530 include a monitor, display terminal, video projection device, and the like.

[0057] Interface(s) 506 include various interfaces that allow computing device 500 to interact with other systems, devices, or computing environments. Example interface(s) 506 include any number of different network interfaces 520, such as interfaces to local area

networks (LANs), wide area networks (WANs), wireless networks, and the Internet. Other interface(s) include user interface 518 and peripheral device interface 522. The interface(s) 506 may also include one or more peripheral interfaces such as interfaces for printers, pointing devices (mice, track pad, etc.), keyboards, and the like.

[0058] Bus 512 allows processor(s) 502, memory device(s) 504, interface(s) 506, mass storage device(s) 508, I/O device(s) 510, and display device 530 to communicate with one another, as well as other devices or components coupled to bus 512. Bus 512 represents one or more of several types of bus structures, such as a system bus, PCI bus, IEEE 1394 bus, USB bus, and so forth.

[0059] For purposes of illustration, programs and other executable program components are shown herein as discrete blocks, although it is understood that such programs and components may reside at various times in different storage components of computing device 500, and are executed by processor(s) 502. Alternatively, the systems and procedures described herein can be implemented in hardware, or a combination of hardware, software, and/or firmware. For example, one or more application specific integrated circuits (ASICs) can be programmed to carry out one or more of the systems and procedures described herein.

[0060] In the above disclosure, reference has been made to the accompanying drawings, which form a part hereof, and in which is shown by way of illustration specific implementations in which the disclosure may be practiced. It is understood that other implementations may be utilized and structural changes may be made without departing from the scope of the present disclosure. References in the specification to “one embodiment,” “an embodiment,” “an example embodiment,” etc., indicate that the

embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to affect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0061] Implementations of the systems, devices, and methods disclosed herein may comprise or utilize a special purpose or general-purpose computer including computer hardware, such as, for example, one or more processors and system memory, as discussed herein. Implementations within the scope of the present disclosure may also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are computer storage media (devices). Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, implementations of the disclosure can comprise at least two distinctly different kinds of computer-readable media: computer storage media (devices) and transmission media.

[0062] Computer storage media (devices) includes RAM, ROM, EEPROM, CD-ROM, solid state drives (“SSDs”) (e.g., based on RAM), Flash memory, phase-change memory (“PCM”), other types of memory, other optical disk storage, magnetic disk storage

or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0063] An implementation of the devices, systems, and methods disclosed herein may communicate over a computer network. A “network” is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links, which can be used to carry desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0064] Computer-executable instructions comprise, for example, instructions and data which, when executed at a processor, cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described

above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0065] Those skilled in the art will appreciate that the disclosure may be practiced in network computing environments with many types of computer system configurations, including, an in-dash vehicle computer, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, tablets, pagers, routers, switches, various storage devices, and the like. The disclosure may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0066] Further, where appropriate, functions described herein can be performed in one or more of: hardware, software, firmware, digital components, or analog components. For example, one or more application specific integrated circuits (ASICs) can be programmed to carry out one or more of the systems and procedures described herein. Certain terms are used throughout the description and claims to refer to particular system components. As one skilled in the art will appreciate, components may be referred to by different names. This document does not intend to distinguish between components that differ in name, but not function.

[0067] It should be noted that the sensor embodiments discussed above may comprise computer hardware, software, firmware, or any combination thereof to perform at least a portion of their functions. For example, a sensor may include computer code configured to be executed in one or more processors, and may include hardware logic/electrical circuitry controlled by the computer code. These example devices are provided herein purposes of illustration, and are not intended to be limiting. Embodiments of the present disclosure may be implemented in further types of devices, as would be known to persons skilled in the relevant art(s).

[0068] At least some embodiments of the disclosure have been directed to computer program products comprising such logic (e.g., in the form of software) stored on any computer useable medium. Such software, when executed in one or more data processing devices, causes a device to operate as described herein.

[0069] While various embodiments of the present disclosure have been described above, it should be understood that they have been presented by way of example only, and not limitation. It will be apparent to persons skilled in the relevant art that various changes in form and detail can be made therein without departing from the spirit and scope of the disclosure. Thus, the breadth and scope of the present disclosure should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents. The foregoing description has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the disclosure to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. Further, it should be noted that any

or all of the aforementioned alternate implementations may be used in any combination desired to form additional hybrid implementations of the disclosure.

CLAIMS:

1. An apparatus comprising:

a computing device including one or more processing devices and one or more memory devices operably coupled to the one or more processing devices, the one or more memory devices storing executable code that, when executed by the one or more processing devices, causes the one or more processing devices to:

receive an instruction to instantiate an instance from an executable image, the instruction including an annotation;

instantiate the instance from the executable image; and

in response to the annotation, store the executable image on a storage device coupled to the one or more processing devices after instantiating the instance from the executable image.
2. The apparatus of claim 1, wherein the executable code, when executed by the one or more processing devices, causes the one or more processing devices to:

in response to a detected failure, re-instantiate the instance from the executable image on the storage device.
3. The apparatus of claim 2, wherein the detected failure is failure of the instance.
4. The apparatus of claim 2, wherein the detected failure is failure of the computing device.

5. The apparatus of claim 1, wherein the instance is a container executing an application instance.

6. The apparatus of claim 5, wherein the instruction is a container specification.

7. The apparatus of claim 6, wherein the executable code, when executed by the one or more processing devices, causes the one or more processing devices to:
 instantiate the instance from the executable image by calling a container runtime interface (CRI).

8. The apparatus of claim 7, wherein the CRI is programmed to store the executable image on the storage device in response to the annotation.

9. The apparatus of claim 8, wherein the CRI is programmed to retrieve the executable image from an image store.

10. The apparatus of claim 1, wherein the computing device is part of a cloud computing platform.

11. A method comprising:

receiving, by a computer system, an instruction to instantiate an instance from an executable image, the instruction including an annotation;

instantiating, by the computer system, the instance from the executable image;

and

in response to the annotation, storing, by the computer system, the executable image on a storage device coupled to the computer system after instantiating the instance from the executable image.

12. The method of claim 11, further comprising:

in response to a detected failure, re-instantiating, by the computer system, the instance from the executable image on the storage device.

13. The method of claim 12, wherein the detected failure is failure of the instance.

14. The method of claim 12, wherein the detected failure is failure of the computer system.

15. The method of claim 11, wherein the instance is a container executing an application instance.

16. The method of claim 15, wherein the instruction is a container

specification.

17. The method of claim 16, further comprising:
instantiating, by the computer system, the instance from the executable image by
calling a container runtime interface (CRI).

18. The method of claim 17, further comprising storing, by the CRI, the
executable image on the storage device in response to the annotation.

19. The method of claim 18, further comprising retrieving, by the CRI, the
executable image from an image store.

20. The method of claim 11, wherein the computer system is part of a cloud
computing platform.

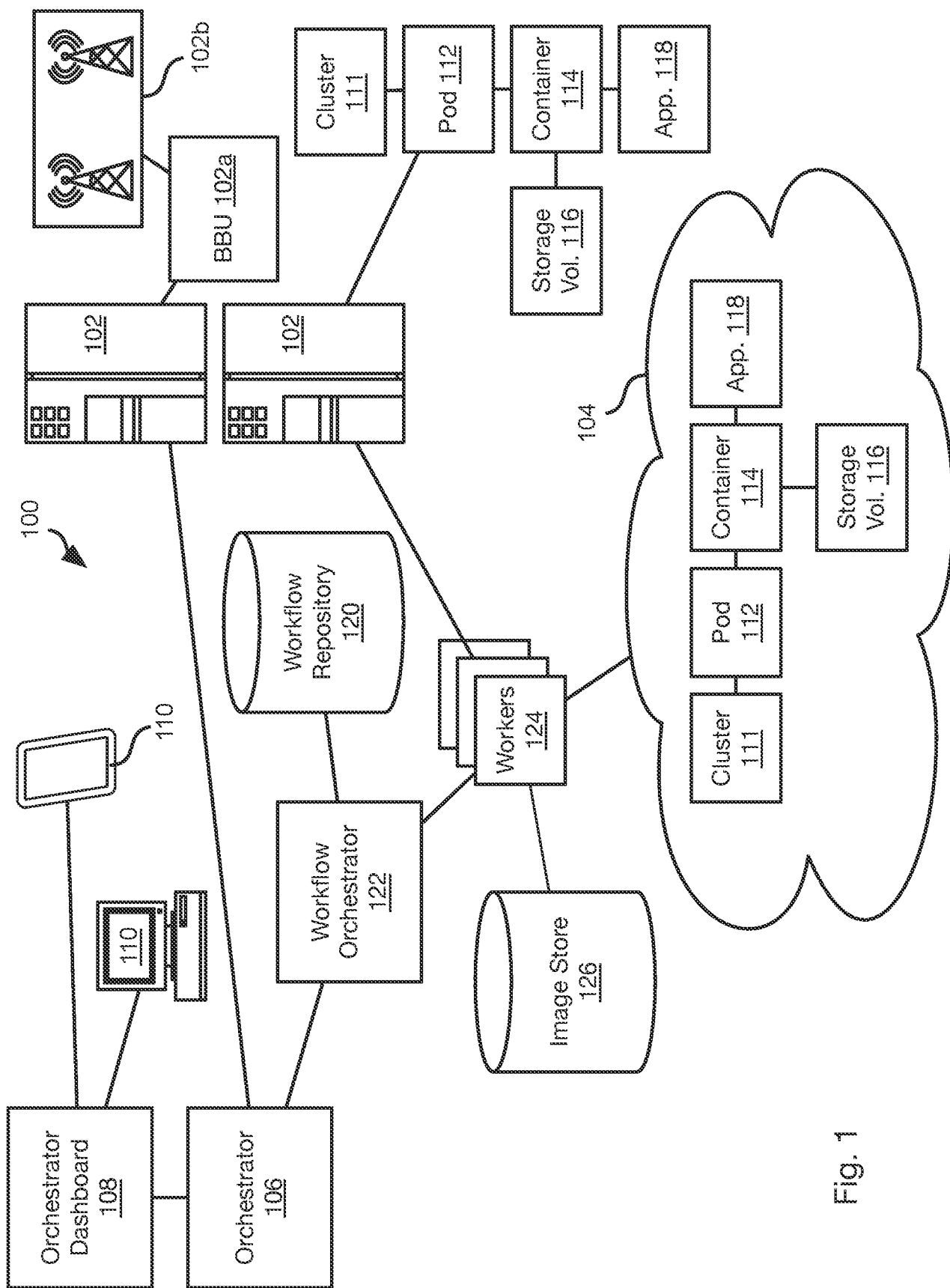


Fig. 1

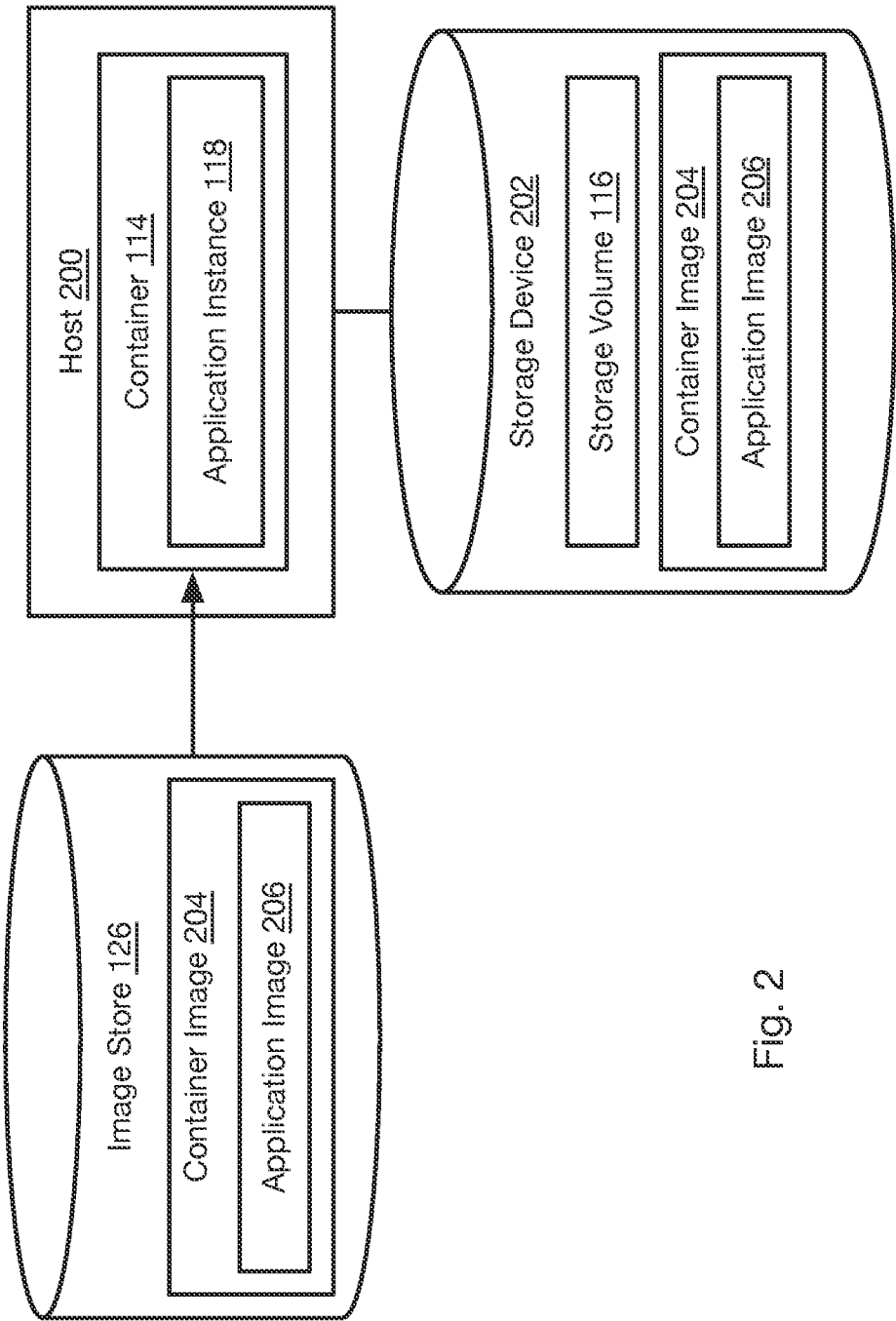


Fig. 2

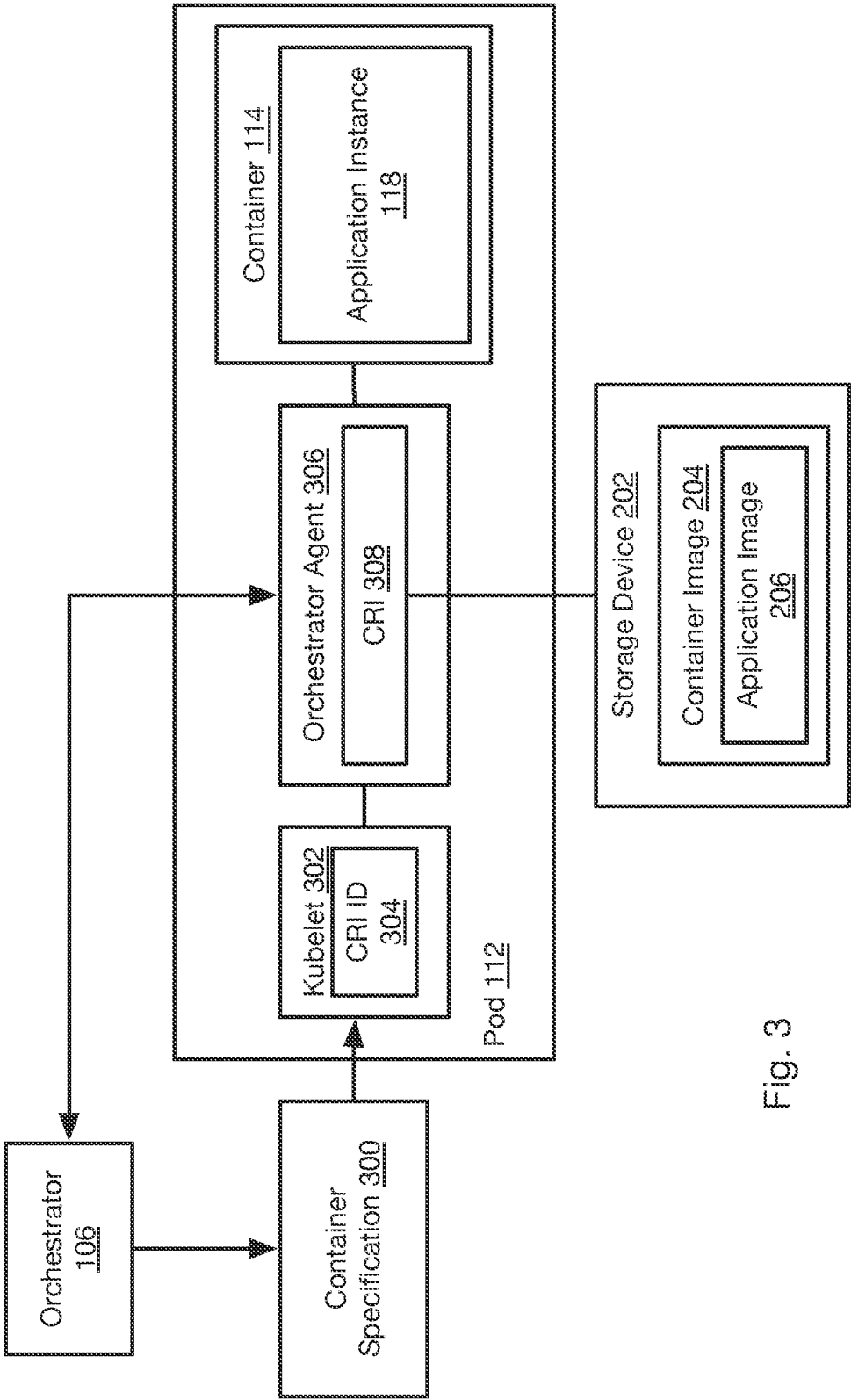


Fig. 3

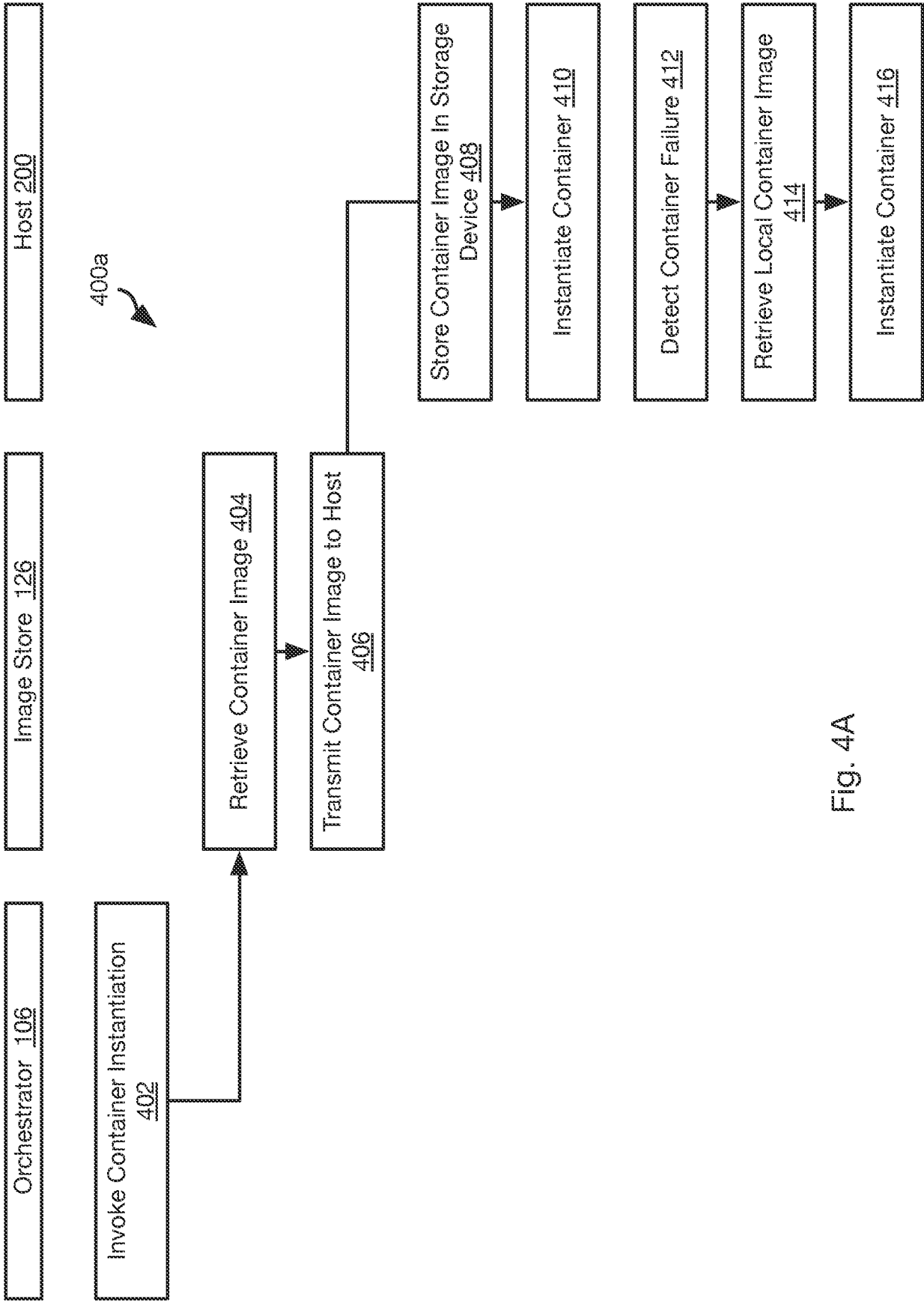


Fig. 4A

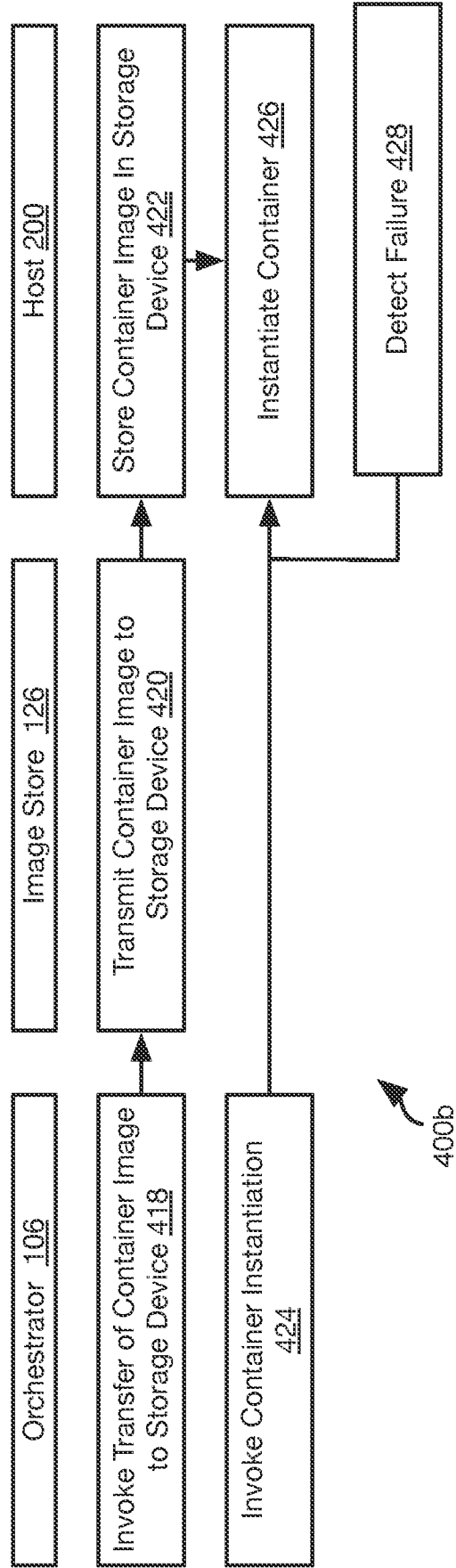


Fig. 4B

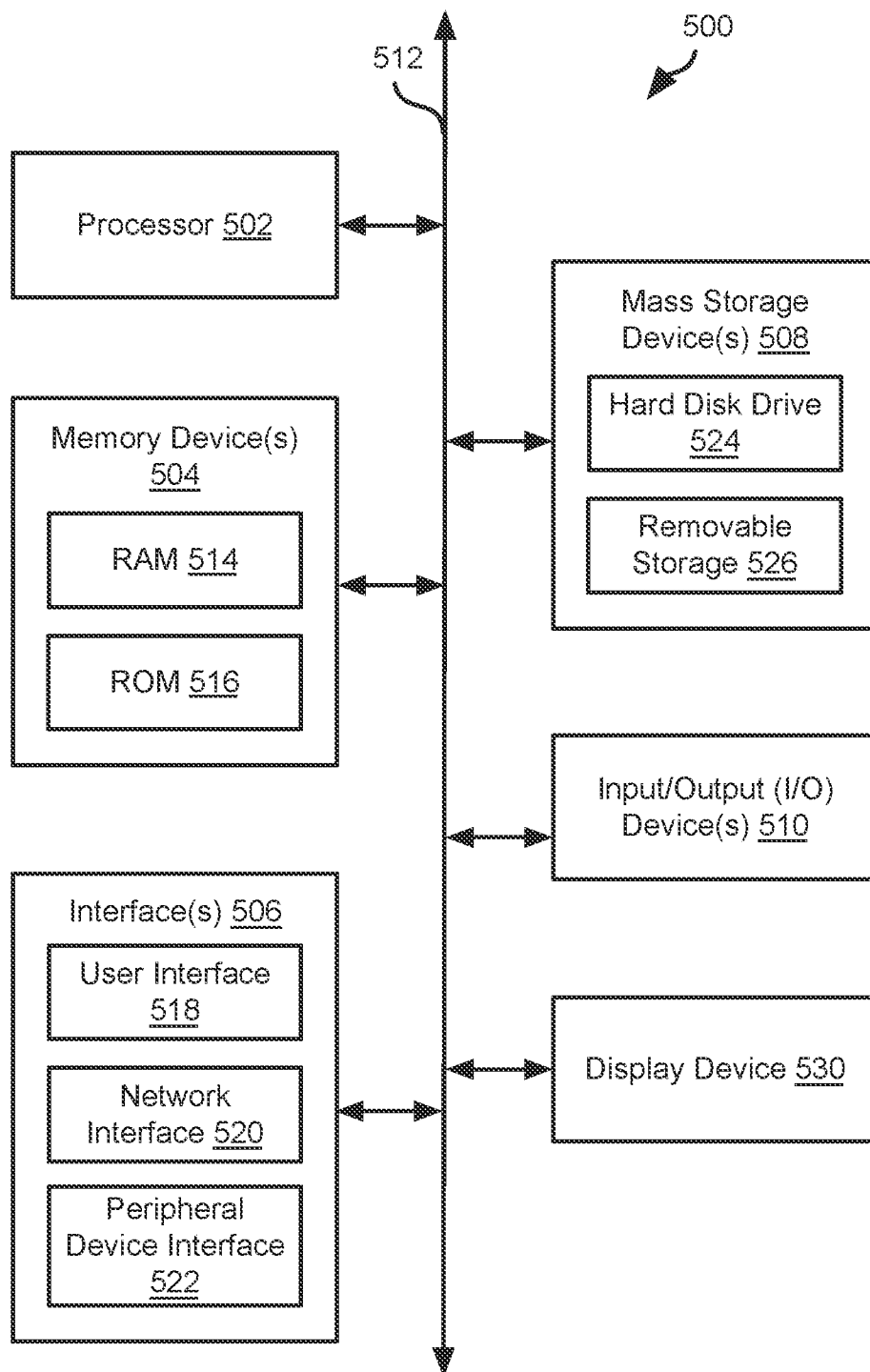


Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 22/52868

A. CLASSIFICATION OF SUBJECT MATTER

IPC - INV. G06F 11/36, G06F 21/50 (2023.01)

ADD. G06F 11/00, G06F 21/00 (2023.01)

CPC - INV. G06F 11/0703, G06F 11/0793, G06F 21/50

ADD. G06F 11/00, G06F 21/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2020/0082095 A1 (CA, Inc.) 12 March 2020 (12.03.2020), entire document, especially Figures 1-3, 6; para: [0043], [0048]-[0050], [0055], [0078], [0091], [0127], [0130]	1-20
A	US 2018/0054314 A1 (Hewlett Packard Enterprise Development LP) 22 February 2018 (22.02.2018), entire document	1-20
A	US 2020/0272488 A1 (Red Hat, Inc.) 27 August 2020 (27.08.2020), entire document	1-20
A	US 2022/0229804 A1 (International Business Machines Corporation) 21 July 2022 (21.07.2022), entire document	1-20
A	US 11,409,588 B1 (Kyndryl, Inc.) 09 August 2022 (09.08.2022), entire document	1-20

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

09 March 2023 (09.03.2023)

Date of mailing of the international search report

APR 04 2023

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents

P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Kari Rodriguez

Telephone No. PCT Helpdesk: 571-272-4300