(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 September 2012 (27.09.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/129219 A2

(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/US2012/029787

(22) International Filing Date:
20 March 2012 (20.03.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/070,812 24 March 2011 (24.03.2011) US

(71) Applicant *(for all designated States except US)*: **AMAZON TECHNOLOGIES, INC.** [US/US]; P.O.Box 8102, Reno, NV 89507 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MARSHALL, Bradley, E.** [US/US]; 410 Terry Avenue North, Seattle, WA 98109-5210 (US). **SIVASUBRAMANIAN, Swaminathan** [IN/US]; 410 Terry Avenue North, Seattle, WA 98109-5210 (US). **CERTAIN, Tate, Andrew** [US/US]; 410 Terry Avenue North, Seattle, WA 98109-5210 (US). **MANISCALCO, Nicholas, J.** [US/US]; 410 Terry Avenue North, Seattle, WA 98109-5210 (US).

(74) Agent: **D'AURELIO, Michael, J.**; Thomas Kayden, Horstemeyer & Risley, LLP, 400 Interstate North Parkway, Suite 1500, Atlanta, GA 30339 (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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

WO 2012/129219 A2

(54) **Title**: REPLICATION OF MACHINE INSTANCES IN A COMPUTING ENVIRONMENT

(57) **Abstract**: Disclosed are various embodiments for replication of machine instances in a computing environment. A clone machine instance is instantiated from a machine image associated with an original machine instance. A stored execution state of the original machine instance is applied to the clone machine instance. At least a portion of a series of stored events received by the original machine instance is applied to the clone machine instance.

REPLICATION OF MACHINE INSTANCES IN A COMPUTING ENVIRONMENT

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to, and the benefit of, U.S. Patent Application entitled, "REPLICATION OF MACHINE INSTANCES IN A COMPUTING ENVIRONMENT" having Application No. 13/070,812, filed March 24, 2011 which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] A system crash (i.e. "crash") in computing is a condition in which a computer or a program, either an application or part of the operating system, ceases to function properly, often resulting in the system being unusable. When computing systems undergo crashes, the user of the machine may have the option of rebooting the computer and/or restarting the applications that were being executed. However, in doing so, the user may lose information about events that led up to the crash, thus hampering the ability to determine what caused the crash. The user may have another option, which is keeping the virtual machine offline or otherwise out of service such that events that led up to the crash can be examined. However, keeping the computing system out of service can have a negative impact on the user's productivity.

[0003] Various forms of shared computing resources have been implemented. As one example, a shared computing resource may include multiple networked computing devices executing instances of virtual machines, where each of these virtual machines are associated with a particular user. These users may be, for example, customers of a utility computing service. Each virtual machine may execute one or more applications, such as a web

server, video encoder, load balancer, or database, among many other possibilities. The users may be capable of terminating machine instances or starting execution of new machine instances whenever they desire, sometimes incurring costs only for the time that they actually use the machine instance. This model thereby provides an elastic computing resource.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Many aspects of the present disclosure can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the disclosure. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

[0005] FIG. 1 is a drawing of a networked environment according to various embodiments of the present disclosure.

[0006] FIG. 2 is a flowchart illustrating one example of functionality implemented as portions of a replicator module executed in a computing device in the networked environment of FIG. 1 according to various embodiments of the present disclosure.

[0007] FIG. 3 is a schematic block diagram that provides one example illustration of a computing device employed in the networked environment of FIG. 1 according to various embodiments of the present disclosure.

DETAILED DESCRIPTION

[0008] The present disclosure relates to replicating machine instances in a shared computing environment. The systems and methods can be used, among

other reasons, to facilitate troubleshooting of computing system failures or computing anomalies. According to some embodiments, a clone machine instance is created from an original machine instance. The machine image of the original machine instance is used to create this clone, resulting in a copy of the original machine instance as it existed at boot time. The state of the clone machine instance is then moved forward in time by first applying a snapshot of the execution environment of the original machine instance as it existed at a particular point in time after boot time. The state of the clone machine instance is then moved forward again in time by applying to the clone machine instance the same inputs seen by the original machine instance during a particular period of time. These inputs may include transactions with a virtual block device (*e.g.*, read, write, *etc.*), network traffic, and possibly other types of inputs.

[0009] In some embodiments an interface is provided to allow the user to control execution of the clone machine instance. This interface may allow the user to select different snapshots managed by the user and different series of inputs. This interface may allow the user to choose specific types of inputs to be applied. In the following discussion, a general description of the system and its components is provided, followed by a discussion of the operation of the same.

[0010] With reference to FIG. 1, shown is a networked environment 100 according to various embodiments. The networked environment 100 includes one or more computing devices 103, one or more utility computing resources 106, and one or more clients 109 in data communication by way of a network 112. The network 112 includes, for example, the Internet, intranets, extranets, wide area networks (WANs), local area networks (LANs), wired networks,

wireless networks, or other suitable networks, *etc.*, or any combination of two or more such networks.

[0011] The utility computing resource 106 includes a plurality of computing devices 115a ... 115n, at least one virtual block device 118, a virtual private router 121, and a data store 124. Such components of the utility computing resource 106 may be in data communication with each other and/or the computing device 103 by way of a network such as network 112. Such components of the utility computing resource 106 may be located in a single location (e.g. a building or cluster of buildings) or may be dispersed among many different geographical locations. Computing devices 115a ... 115n may correspond to differing hardware platforms in various embodiments. Accordingly, computing devices 115a ... 115n may have differing hardware configurations, for example, of central processing units (CPUs), system memory, data storage, network bandwidth, and/or other hardware characteristics.

[0012] Each computing device 115a ... 115n may execute one or more machine instances (MI) 127, where a machine instance 127 corresponds to an actual physical machine or to a virtual machine. In the example of FIG. 1, an original machine instance 127-O is executed on computing device 115a while a clone machine instance 127-C is executed on computing device 115n. However, in other embodiments, original machine instance 127-O and clone machine instance 127-C execute as virtual machines on the same computing device 115.

[0013] A virtual machine instance is a virtualized computer system, or a software implementation of a computer system layered on top of a physical computing system, such as through a hypervisor. Virtual machines may provide for multiple and/or different operating system environments to run concurrently

on a single system having a processor circuit and a memory. As a non-limiting example, multiple instances of a Linux® operating system environment may execute concurrently with multiple instances of a Microsoft® Windows® operating system environment on a single computer system. Each machine instance may be controlled by different users, who may have administrative access only to their own instance(s) and no access to the instances of other users. Multiple machine instances may, in fact, execute concurrently on a computer system including parallel processors, although multiple instances may appear to execute concurrently on a multithreaded computer system with fewer processors than instances. In some cases, different machine instances run on a particular physical computing device 115 are controlled by two or more different users, while in other cases all of the machine instances are controlled by a single user.

[0014] Various applications and/or other functionality may be executed in the computing devices 115a ... 115n according to various embodiments. Also, various data is stored in the data store 124 that is accessible to the computing devices 115a ... 115n. The data store 124 may be representative of a plurality of data stores as can be appreciated. The data stored in the data store 124, for example, is associated with the operation of the various applications and/or functional entities described herein. The data stored in the data store 124 may include, for example, one or more machine images 130 used to start the execution of a machine instance 127. A machine image 130 includes one or more disk images of an operating system environment and may include additional software such as web servers, databases, load balancers, caches, among other possible application software possibilities.

[0015] In addition to the storage provided by the data store 124, one or more of the machine instances 127 may utilize a virtual block device 118 for storage. A virtual block device 118 appears to the machine instance 127 as a local block level device (*e.g.*, an attached disk drive) but that can be attached to and unattached from the instance through a network. The virtual block device 118 may be implemented, for example, by one or more server computers with a disk drive or other block level data storage device, by a storage area network (SAN), or by any other system which provides storage capability and the appropriate virtual interface.

[0016] As discussed previously, the computing devices 115a ... 115n which make up the utility computing resource 106 are attached to the network 112, which allows communication among computing devices 103, clients 109, and utility computing resources 106. In addition, network 112 can facilitate communications among individual components within these entities, such as among computing devices 115a-n. Although network 112 may include a number of physical networking devices (e.g. routers, switches, aggregators, etc.), a virtual private router 121 may further provide a virtual network overlay on top of the network 112. This virtual network overlay may be user-specific. With a user-specific virtual network overlay in place, machine instances 127 inside the utility computing resource 106 that are associated with a particular user appear to be in the same IP network as computing entities outside the utility computing resource 106 that are associated with the same user. The virtual private router 121 may be implemented, for example, by a router, a switch, a server computer with a network interface, or by any other computing system with a network interface and appropriate network overlay functionality. The virtual network

overlay allows manipulation of network traffic so that two machines, or machine instances, can appear to use the same IP address. The virtual network overlay distinguishes between the two in order to deliver packets to the correct machine instance, using packet wrapping, tunneling techniques, or other approaches.

[0017] Turning now from the utility computing resource 106 to the computing device 103, the computing device 103 may comprise, for example, a server computer or any other system providing computing capability. Alternatively, a plurality of computing devices 103 may be employed that are arranged, for example, in one or more server banks or computer banks or other arrangements. For example, a plurality of computing devices 103 together may comprise, for example, a cloud computing resource, a grid computing resource, and/or any other utility computing arrangement. Such computing devices 103 may be located in a single installation or may be distributed among many different geographical locations. For purposes of convenience, the computing device 103 is referred to herein in the singular. Even though the computing device 103 is referred to in the singular, it is understood that a plurality of computing devices 103 may be employed in the various arrangements as described above.

[0018] Various applications and/or other functionality may be executed in the computing device 103 according to various embodiments. The components executed on the computing device 103, for example, include a replicator module 133, and may also include other applications, services, processes, systems, engines, or functionality not discussed in detail herein. The replicator module 133 is executed to clone or replicate machine instances. In doing so, the replicator module 133 reproduces the state of an original machine instance 127-O as it existed at a particular point in time on a clone machine instance 127-C.

The replicator module 133 is further executed to take inputs which were received at the original machine instance 127-O and to apply those inputs to the clone machine instance 127-C such that the execution of the clone machine instance 127-C moves forward in time from the snapshot of the particular point in time.

[0019] In the embodiment illustrated in FIG. 1, the computing device 103 which executes the replicator module 133 is separate from, but in communication with, the utility computing resource 106. In other embodiments (not shown), the computing device 103 resides within the utility computing resource 106.

[0020] Various data is stored in a data store 136 that is accessible to the computing device 103. The data stored in the data store 136 includes, for example, execution state 139, one or more event capture buffers 142, and potentially other data which is associated with the operation of the replicator module 133. The data store 136 may be representative of a plurality of data stores as can be appreciated.

[0021] Turning now from the computing device 103 to the client 109, the client 109 is representative of a plurality of client devices that may be coupled to the network 112. The client 109 may comprise, for example, a processor-based system such as a computer system. Such a computer system may be embodied in the form of a desktop computer, a laptop computer, a personal digital assistant, a cellular telephone, a set-top box, a music player, a web pad, a tablet computer system, a game console, or other devices with like capability.

[0022] The client 109 may utilize web services to interact with the utility computing resource 106. These web services may be implemented via a variety of middleware frameworks, such as remote procedure calls, service-oriented

architecture (SOA), representational state transfer (REST), and other frameworks. The client 109 may be configured to execute various applications such as a browser 145 and/or other applications. The browser 145 may be executed in a client 109, for example, to access and render network pages, such as web pages, or other network content served by the computing device 103 and/or other servers. The client 109 may be configured to execute applications beyond browser 145 such as, for example, email applications, instant message applications, and/or other applications.

[0023] Next, a general description of the operation of the various components of the networked environment 100 is provided. To begin, a user at a client 109 configures one or more machine instances (MI) 127 for execution within the utility computing resource 106. The initiation of the execution of the machine instances 127 may be performed manually (e.g. through a browser or other user interface) by the user or programmatically. In configuring a machine instance 127, the user may specify a particular machine image 130 which can include a desired operating system and, potentially, one or more applications). Further, the configuring can include specifying a desired hardware platform (e.g. having a particular chipset, processor speed, memory size, *etc.*). In executing, the machine instances 127 thereby use computing resources of the utility computing resource 106.

[0024] The user initiates replication, or cloning, of a machine instance 127 by interacting (manually or programmatically) with the replicator module 133. As will be further described below, this replication performed by the replicator module 133 allows a user to investigate any sort anomalous behavior by "replaying," on the clone instance, activity leading up to the anomalous behavior

of the original instance. Such anomalous behavior may include, for example, an unresponsive system, a system that returns unexpected data, a crash, *etc.* The replication also enhances testing by allowing the user to execute the original machine instance, and then perform a series of replications, advancing the clone instance to a different execution state each time.

[0025] The replicator module 133 associates each replica or clone with a replay buffer size and/or snapshot period. For example, a clone may be configured to take a snapshot of the original instance's execution state every five minutes, and then to buffer for five minutes after the snapshot. The replay buffer may include any events which originate from outside the original instance and interact with the original instance, for example, local and remote storage device transactions, network traffic, and environmental information. When the event data includes environmental information, environmental sensors associated with the original instance can be used to record the environmental information such as, but not limited to, temperature, humidity, vibration, voltage and/or current supplied to various computing components hosting the original image. The events that trigger the recording of this information can be, for example, periodic samples or samples triggered by alarms. Thus, if the original instance crashes, the replicator module 133 can replay activity before the crash, up to five minutes. In some embodiments multiple clones can be used with staggered snapshot periods, for example, a first clone is created using a snapshot taken at time t, a second clone is created using a snapshot at t+5 minutes, a third clone is created using a snapshot at t+10 minutes, *etc.* Multiple clones also be created with different replay sizes, for example, a first clone is created from a snapshot at

time t using a 5 minute replay buffer, and a second clone is also created at time t but using a 10 minute replay buffer.

[0026] In some embodiments, the clone machine instance 127-C utilizes the same IP address as the original machine instance 127-O. As noted above, the virtual network overlay takes care of distinguishing between the two machine instances and ensuring delivery to the correct machine instance. This capability, coupled with the replay buffer, allows the original machine instance 127-O to continue executing and to continue to send/receive network traffic while the clone machine instance 127-C simultaneously receives virtual network traffic destined for the same address from the capture buffer.

[0027] The replicator module 133 may provide an event control interface that allows the user to start and stop replay of events into the clone by specifying particular timestamps or ranges of timestamps. In some embodiments, this event control interface allows the user to select a different snapshot from which to start execution by specifying a particular time at which the snapshot was taken. This interface may be programmatic or may be a graphical user interface.

[0028] Referring next to FIG. 2, shown is a flowchart that provides one example of the operation of a portion of the replicator module 133 according to various embodiments. It is understood that the flowchart of FIG. 2 provides merely an example of the many different types of functional arrangements that may be employed to implement the operation of the portion of the replicator module 133 as described herein. As an alternative, the flowchart of FIG. 2 may be viewed as depicting an example of steps of a method implemented in the computing device 103 (FIG. 1) according to one or more embodiments.

[0029] The process 200 of FIG. 2 begins with block 203, the replicator module 133 instantiates a clone machine instance 127-C from the machine image 130 used to instantiate a corresponding original machine instance 127-O. As mentioned above, a machine image is a stored image of an operating system, sometimes also including a particular set of software applications. In some embodiments, the instantiation in block 203 creates a virtual machine and boots the virtual machine from the machine image 130. In other embodiments, the instantiation in block 203 reserves or allocates a physical machine and boots the physical machine from the machine image 130. In either case, the result is the particular machine instance 127-C, which is a clone of the original machine instance 127-O as it existed at boot time.

[0030] In block 206, the replicator module 133 examines configuration settings and attaches a debugger to the clone machine instance 127-C, if the settings so indicate. These configuration settings may be provided by the user through a client 109 (*e.g.*, via a web services call, a network page, *etc.*), or may be default settings.

[0031] Next, in block 209, the replicator module 133 accesses execution state 139 in data store 136, which represents a snapshot of the execution state of the original machine instance 127-O at a particular point in time. The replicator module 133 applies the execution state 139 to the clone machine instance 127-C, thus setting the execution state of the clone machine instance 127-C to the time the snapshot was taken. In some embodiments, this may be the most recent snapshot of the execution state. In other embodiments, if multiple snapshots are available, one snapshot is selected through the event controller interface described above.

[0032] The execution state 139 includes at least the contents of the memory of the original machine instance 127-O and the register contents of the processor of the original machine instance 127-O at the time of the last snapshot. The execution state 139 may also include the contents of block devices locally attached to original machine instance 127-O. The execution state 139 may also include the storage state of one or more virtual block devices that are remotely accessible to the original machine instance. The execution state 139 may have been stored by replicator module 133 or by another module.

[0033] In block 212, the replicator module 133 accesses an event capture buffer 142 in data store 136, which contains a series of inputs or events received by the original machine instance 127-O after the snapshot. The event capture buffer 142 may have been stored by replicator module 133 or by another module. The replicator module 133 begins applying, or "playing back," some or all of the stored events to the clone machine instance 127-C. The individual events include timestamps, and the replicator module 133 may use these timestamps during playback to maintain the same timing relationship between events that was present at the time the events were captured. The replicator module 133 may also use these timestamps during playback to alter the timing, such that playback occurs slower than, or faster than, the original events. The playback in block 212 may occur in response to programmatic or user input through the event controller interface described above, or may occur automatically after the application of the execution state in block 209.

[0034] The number of events applied to the clone machine instance 127-C may be determined based on criteria supplied through the event controller interface. As one example, a criterion may specify that five minutes of the event

capture buffer 142, or other time period, are played back. As another example, a criterion may specify that 33% of the event capture buffer 142, or other percentage, is played back. In some embodiments, the user may also specify which portion of the event capture buffer 142 is to be applied. The portion of the event capture buffer 142 may be, for example, the first five minutes of a buffer, or the middle third of a buffer, or some other portion. In some embodiments multiple capture buffers 142 are stored according to different configuration data. For example, the events may be included or excluded based on where the events originate from (e.g. a device, an IP address, etc.), a time they are received, the type of event, or a user from which the events are received, among other examples.

[0035] Since the original machine instance 127-O executes in the environment of the utility computing resource 106, the events applied can include virtual block device transactions between the original machine instance 127-O and the virtual block device 118, and/or virtual network traffic between the original machine instance 127-O and the virtual private router 121. In some embodiments, the input events are restricted and do not include keystrokes, mouse movements, or other forms of user input, since the environment of the utility computing resource 106 is sometimes restricted in this manner. Some embodiments of the replicator module 133 allow specified types of events to be selectively included or excluded from playback to the clone machine instance 127-C. In one example, only virtual block device transactions are applied to the clone machine instance 127-C. In another example, virtual block device transactions are excluded from playback. In yet another example, only those virtual block device transactions which involve a specific device are applied. In

yet another example, only virtual network traffic involving a particular source or destination address is applied. In addition to excluding some events, in some embodiments, entropic events can be injected into the events in order to observe how the cloned instance behaves. For example, various events (e.g. network traffic, hard drive failure events, etc.) may be applied at random times in conjunction with the original event data.

[0036] In some embodiments, the event capture buffer 142 used for playback may be the buffer with the most recent events. In other embodiments, if multiple capture buffers are available, one buffer can be selected based on criteria selected by the user associated with the original machine instance 127-O. In this way, a user can select from among various buffers that contain desired event data.

[0037] Once playback from the event capture buffer 142 has begun, the replicator module 133 waits at block 215 for a periodic timer or receipt of a debugger command. Although an event-driven model will now be described for buffer playback and debugger commands, a polling mechanism is also possible, as should be appreciated. On expiration of a periodic timer, the replicator module 133 moves to block 218. In block 218, the replicator module 133 plays back the next portion of the event capture buffer 142. If that portion is not the last, the replicator module 133 moves to block 215 to await another command or timer expiration. However, if this is the last portion of the buffer (playback is finished), the replicator module 133 moves to block 221.

[0038] When a debugger command is received, the replicator module 133 handles the command in block 221. Such handling may involve traditional debugger functions such as starting, stopping, or pausing execution of the clone

machine instance 127-C. In addition, the replicator module 133 may provide more functions than those of a traditional debugger. For example, the replicator module 133 may allow the user to select a different snapshot from which to restart execution, to select a different portion of the capture buffer for playback, or to select a different capture buffer altogether. If the command handling involves starting from a different snapshot, the replicator module 133 continues processing at block 209. If the command handling involves starting from a different snapshot, the replicator module 133 continues processing at block 212. Otherwise, the replicator module 133 moves to block 218, where the next portion of the capture buffer is played back.

[0039] The embodiment described in connection with FIG. 2 utilizes a capture buffer and replays events from this capture buffer. However, other embodiments create a clone machine instance and then apply a snapshot of the execution state 139 to the clone machine instance 127-C, but do not store events to, or replay events from, a capture buffer. Even without a capture buffer, the cloning of a machine instance allows forensic analysis to be performed while the original machine instance continues execution or is rebooted/restarted.

[0040] With reference to FIG. 3, shown is a schematic block diagram of the computing device 103 according to an embodiment of the present disclosure. The computing device 103 includes at least one processor circuit, for example, having a processor 303 and a memory 306, both of which are coupled to a local interface 309. To this end, the computing device 103 may comprise, for example, at least one server computer or like device. The local interface 309 may comprise, for example, a data bus with an accompanying address/control bus or other bus structure as can be appreciated.

[0041] Stored in the memory 306 are both data and several components that are executable by the processor 303. In particular, stored in the memory 306 and executable by the processor 303 are a replicator module 133 and potentially other applications. Also stored in the memory 306 may be a data store 136 and other data. In addition, an operating system may be stored in the memory 306 and executable by the processor 303.

[0042] It is understood that there may be other applications that are stored in the memory 306 and are executable by the processors 303 as can be appreciated. Where any component discussed herein is implemented in the form of software, any one of a number of programming languages may be employed such as, for example, C, C++, C#, Objective C, Java, Javascript, Perl, PHP, Visual Basic, Python, Ruby, Delphi, Flash, or other programming languages.

[0043] A number of software components are stored in the memory 306 and are executable by the processor 303. In this respect, the term "executable" means a program file that is in a form that can ultimately be run by the processor 303. Examples of executable programs may be, for example, a compiled program that can be translated into machine code in a format that can be loaded into a random access portion of the memory 306 and run by the processor 303, source code that may be expressed in proper format such as object code that is capable of being loaded into a random access portion of the memory 306 and executed by the processor 303, or source code that may be interpreted by another executable program to generate instructions in a random access portion of the memory 306 to be executed by the processor 303, *etc.* An executable program may be stored in any portion or component of the memory 306

including, for example, random access memory (RAM), read-only memory (ROM), hard drive, solid-state drive, USB flash drive, memory card, optical disc such as compact disc (CD) or digital versatile disc (DVD), floppy disk, magnetic tape, or other memory components.

[0044] The memory 306 is defined herein as including both volatile and nonvolatile memory and data storage components. Volatile components are those that do not retain data values upon loss of power. Nonvolatile components are those that retain data upon a loss of power. Thus, the memory 306 may comprise, for example, random access memory (RAM), read-only memory (ROM), hard disk drives, solid-state drives, USB flash drives, memory cards accessed via a memory card reader, floppy disks accessed via an associated floppy disk drive, optical discs accessed via an optical disc drive, magnetic tapes accessed via an appropriate tape drive, and/or other memory components, or a combination of any two or more of these memory components. In addition, the RAM may comprise, for example, static random access memory (SRAM), dynamic random access memory (DRAM), or magnetic random access memory (MRAM) and other such devices. The ROM may comprise, for example, a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other like memory device.

[0045] Also, the processor 303 may represent multiple processors 303 and the memory 306 may represent multiple memories 306 that operate in parallel processing circuits, respectively. In such a case, the local interface 309 may be an appropriate network 112 (FIG. 1) that facilitates communication between any two of the multiple processors 303, between any processor 303 and any of the

memories 306, or between any two of the memories 306, *etc.* The local interface 309 may comprise additional systems designed to coordinate this communication, including, for example, performing load balancing. The processor 303 may be of electrical or of some other available construction.

[0046] Although replicator module 133 and other various systems described herein may be embodied in software or code executed by general purpose hardware as discussed above, as an alternative the same may also be embodied in dedicated hardware or a combination of software/general purpose hardware and dedicated hardware. If embodied in dedicated hardware, each can be implemented as a circuit or state machine that employs any one of or a combination of a number of technologies. These technologies may include, but are not limited to, discrete logic circuits having logic gates for implementing various logic functions upon an application of one or more data signals, application specific integrated circuits having appropriate logic gates, or other components, *etc.* Such technologies are generally well known by those skilled in the art and, consequently, are not described in detail herein.

[0047] The flowchart of FIG. 2 shows the functionality and operation of an implementation of portions of environment 100. If embodied in software, each block may represent a module, segment, or portion of code that comprises program instructions to implement the specified logical function(s). The program instructions may be embodied in the form of source code that comprises human-readable statements written in a programming language or machine code that comprises numerical instructions recognizable by a suitable execution system such as a processor 303 in a computer system or other system. The machine code may be converted from the source code, *etc.* If embodied in hardware,

each block may represent a circuit or a number of interconnected circuits to implement the specified logical function(s).

[0048] Although the flowchart of FIG. 2 shows a specific order of execution, it is understood that the order of execution may differ from that which is depicted. For example, the order of execution of two or more blocks may be scrambled relative to the order shown. Also, two or more blocks shown in succession in FIG. 2 may be executed concurrently or with partial concurrence. Further, in some embodiments, one or more of the blocks shown in FIG. 2 may be skipped or omitted. In addition, any number of counters, state variables, warning semaphores, or messages might be added to the logical flow described herein, for purposes of enhanced utility, accounting, performance measurement, or providing troubleshooting aids, *etc.* It is understood that all such variations are within the scope of the present disclosure.

[0049] Also, any logic or application described herein, including replicator module 133, that comprises software or code can be embodied in any non-transitory computer-readable medium for use by or in connection with an instruction execution system such as, for example, a processor 303 in a computer system or other system. In this sense, the logic may comprise, for example, statements including instructions and declarations that can be fetched from the computer-readable medium and executed by the instruction execution system. In the context of the present disclosure, a "computer-readable medium" can be any medium that can contain, store, or maintain the logic or application described herein for use by or in connection with the instruction execution system. The computer-readable medium can comprise any one of many physical media such as, for example, magnetic, optical, or semiconductor media.

More specific examples of a suitable computer-readable medium would include, but are not limited to, magnetic tapes, magnetic floppy diskettes, magnetic hard drives, memory cards, solid-state drives, USB flash drives, or optical discs. Also, the computer-readable medium may be a random access memory (RAM) including, for example, static random access memory (SRAM) and dynamic random access memory (DRAM), or magnetic random access memory (MRAM). In addition, the computer-readable medium may be a read-only memory (ROM), a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other type of memory device.

Various embodiments of the disclosure can be described by the following clauses:

Clause 1. A non-transitory computer-readable medium embodying a program executable in a computing device, the program comprising:

code that stores an execution state of an original virtual machine instance, the execution state corresponding to a specific point in time;

code that stores a series of timestamped events received by the original virtual machine instance;

code that instantiates a clone virtual machine instance from a machine image from which the original virtual machine instance was instantiated;

code that applies the stored execution state to the clone virtual machine instance; and

code that applies, to the clone virtual machine instance, at least a portion of the series of stored timestamped events.

Clause 2. The computer-readable medium of Clause 1, further comprising code that selects the portion of the series of timestamped events based at least in part on a specified window of time.

Clause 3. The computer-readable medium of Clause 1, wherein the stored timestamp events are stored in a capture buffer, and further comprising code that selects the portion of the series of timestamped events relative to the size the capture buffer.

Clause 4. The computer-readable medium of Clause 1, further comprising code that instantiates another clone virtual machine instance.

Clause 5. A system, comprising:

at least one computing device; and

a replicator executable in the at least one computing device, the replicator comprising:

logic that creates a clone machine instance; and

logic that applies an execution state of an original machine instance to the clone machine instance.

Clause 6. The system of Clause 5, wherein the logic that creates the clone machine instance comprises logic that instantiates the clone machine instance from a machine image associated with the original machine instance.

Clause 7. The system of Clause 5, further comprising logic that applies, to the clone machine instance, a series of stored events received by the original machine instance.

Clause 8. The system of Clause 7 wherein the series of stored events comprises network traffic and block device transactions.

Clause 9. The system of Clause 8, wherein the network traffic is traffic to or from the original machine instance.

Clause 10. The system of Clause 7 wherein the series of stored events comprises environment information.

Clause 11. The system of Clause 5, wherein the original machine instance comprises a virtual machine.

Clause 12. The system of Clause 5, wherein the stored execution state includes a memory state of the original machine instance and a register state of a processor of the original machine instance.

Clause 13. The system of Clause 5, wherein the stored execution state includes a storage state of block device accessible to the original machine instance.

Clause 14. The system of Clause 5, wherein each of the stored events includes a timestamp and the logic that applies the series of stored events applies each stored event in accordance with the corresponding timestamp.

Clause 15. A computer-implemented method, comprising the steps of:

instantiating, via a computer, a clone machine instance from a machine image associated with an original machine instance;

applying, via the computer, a stored execution state of the original machine instance to the clone machine instance; and

applying, to the clone machine instance via the computer, at least a portion of a series of stored events received by the original machine instance.

Clause 16. The method of Clause 15, wherein the series of stored events includes at least one of block device transactions, network traffic, and environment information.

Clause 17. The method of Clause 15, further comprising:

injecting entropic events into the stored events; and

applying the entropic events in addition to applying the at least a portion of a series of stored events received by the original machine instance.

Clause 18. The method of Clause 15, further comprising the step of pausing the application of the series of stored events.

Clause 19. The method of Clause 15, wherein applying the at least a portion of a series of stored events received by the original machine instance comprises selectively excluding some of the stored events from being applied.

Clause 20. The method of Clause 15, further comprising:

instantiating, via a computer, a second clone machine instance from the machine image associated with an original machine instance;

applying, via the computer, a second stored execution state of the original machine instance to the second clone machine instance; and

applying, to the second clone machine instance, at least a portion of the series of stored events received by the original machine instance.

Clause 21. The method of Clause 15, further comprising the step of selecting the stored execution state from a plurality of stored execution states.

Clause 22. The method of Clause 15, further comprising the step of selecting the portion of the series of stored events from the entire series of stored events.

[0050] It should be emphasized that the above-described embodiments of the present disclosure are merely possible examples of implementations set forth for a clear understanding of the principles of the disclosure. Many variations and modifications may be made to the above-described embodiment(s) without departing substantially from the spirit and principles of

the disclosure. All such modifications and variations are intended to be included herein within the scope of this disclosure and protected by the following claims.

CLAIMS

Therefore, the following is claimed:

1. A system, comprising:

at least one computing device; and

a replicator executable in the at least one computing device, the replicator comprising:

logic that creates a clone machine instance; and

logic that applies an execution state of an original machine instance to the clone machine instance.

2. The system of claim 1, wherein the logic that creates the clone machine instance comprises logic that instantiates the clone machine instance from a machine image associated with the original machine instance.

3. The system of claim 1, further comprising logic that applies, to the clone machine instance, a series of stored events received by the original machine instance.

4. The system of claim 1, wherein the original machine instance comprises a virtual machine.

5. The system of claim 1, wherein the stored execution state includes a memory state of the original machine instance and a register state of a processor of the original machine instance.

6. The system of claim 1, wherein the stored execution state includes a storage state of block device accessible to the original machine instance.

7. The system of claim 1, wherein each of the stored events includes a timestamp and the logic that applies the series of stored events applies each stored event in accordance with the corresponding timestamp.

8. A computer-implemented method, comprising the steps of:

instantiating, via a computer, a clone machine instance from a machine image associated with an original machine instance;

applying, via the computer, a stored execution state of the original machine instance to the clone machine instance; and

applying, to the clone machine instance via the computer, at least a portion of a series of stored events received by the original machine instance.

9. The computer-implemented method of claim 8, wherein the series of stored events includes at least one of block device transactions, network traffic, and environment information.

10. The computer-implemented method of claim 8, further comprising:

injecting entropic events into the stored events; and

applying the entropic events in addition to applying the at least a portion of a series of stored events received by the original machine instance.

11. The computer-implemented method of claim 8, further comprising the step of pausing the application of the series of stored events.

12. The computer-implemented method of claim 8, wherein applying the at least a portion of a series of stored events received by the original machine instance comprises selectively excluding some of the stored events from being applied.

13. The computer-implemented method of claim 8, further comprising:

instantiating, via a computer, a second clone machine instance from the machine image associated with an original machine instance;

applying, via the computer, a second stored execution state of the original machine instance to the second clone machine instance; and

applying, to the second clone machine instance, at least a portion of the series of stored events received by the original machine instance.

14. The computer-implemented method of claim 8, further comprising the step of selecting the stored execution state from a plurality of stored execution states.

15. The computer-implemented method of claim 8, further comprising the step of selecting the portion of the series of stored events from the entire series of stored events.

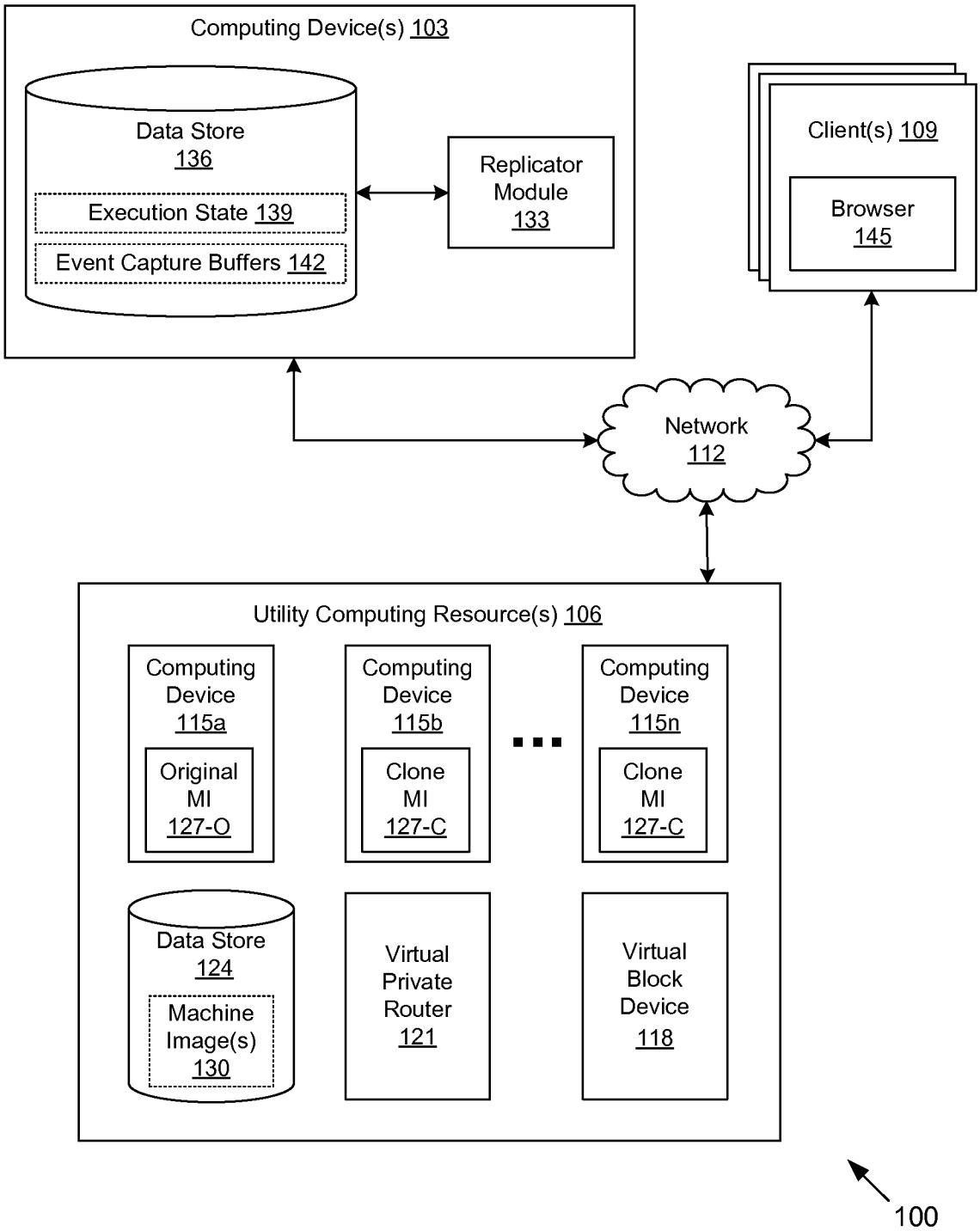
Computing Device(s) 103
Data Store
136
Execution State 139
Event Capture Buffers 142
Replicator
Module
133
Client(s) 109
Browser
145
Network
112
Utility Computing Resource(s) 106
Computing
Device
115a
Original
MI
127-O
Computing
Device
115b
Clone
MI
127-C
Computing
Device
115n
Clone
MI
127-C
Data Store
124
Machine
Image(s)
130
Virtual
Private
Router
121
Virtual
Block
Device
118
100

FIG. 1

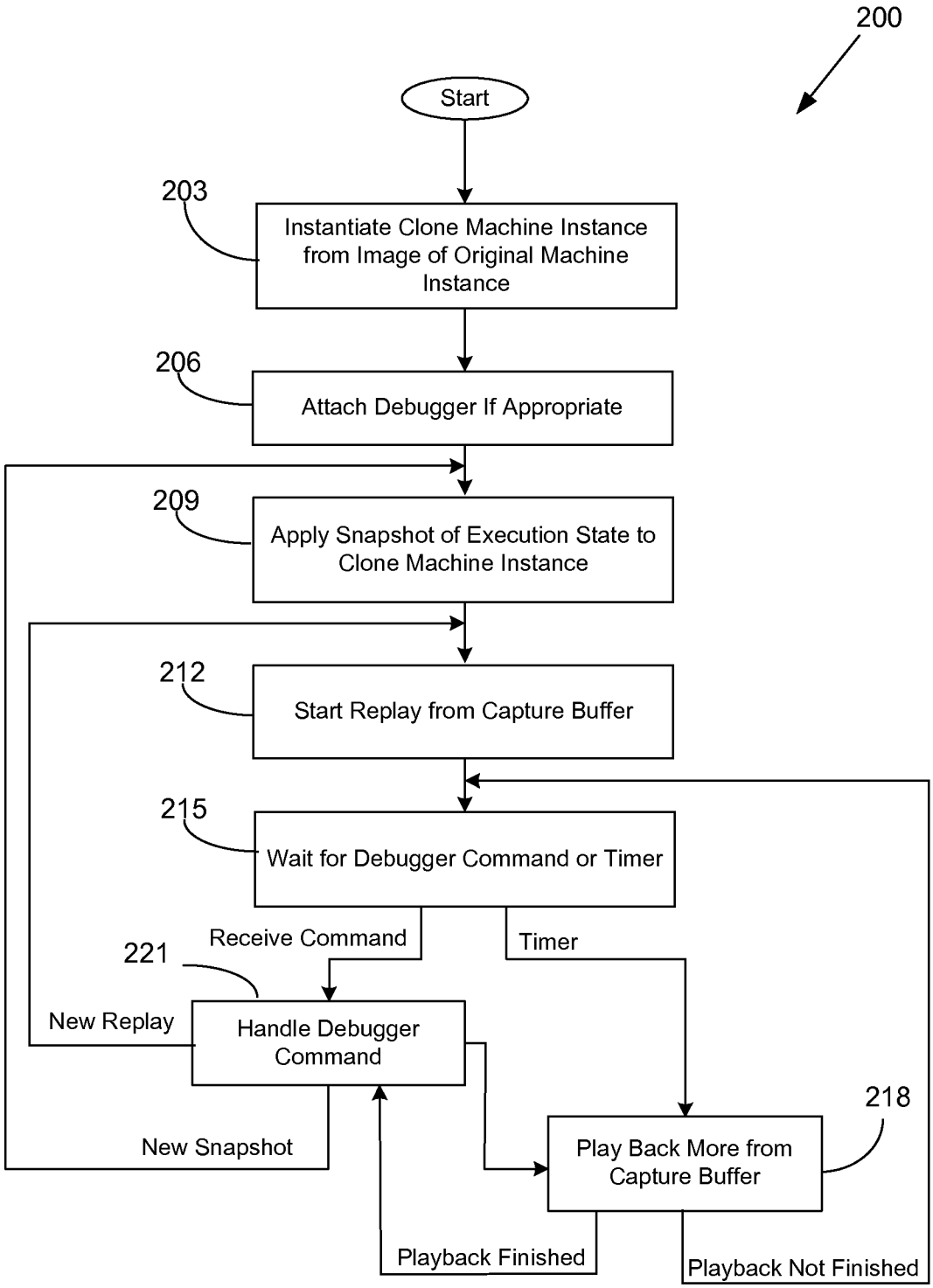
200
Start
203
Instantiate Clone Machine Instance from Image of Original Machine Instance
206
Attach Debugger If Appropriate
209
Apply Snapshot of Execution State to Clone Machine Instance
212
Start Replay from Capture Buffer
215
Wait for Debugger Command or Timer
Receive Command
Timer
221
New Replay
Handle Debugger Command
218
New Snapshot
Play Back More from Capture Buffer
Playback Finished
Playback Not Finished

FIG. 2

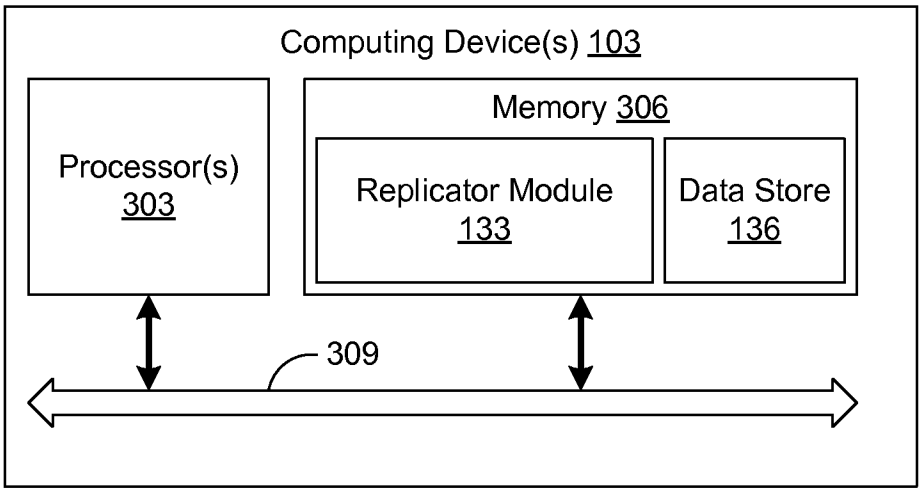
Computing Device(s) 103
Processor(s) 303
Memory 306
Replicator Module 133
Data Store 136
309

FIG. 3