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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 June 2011 (16.06.2011)

PCT

(10) International Publication Number
WO 2011/071624 A2

(51) International Patent Classification:
H04L 12/26 (2006.01)

(21) International Application Number:
PCT/US2010/055739

(22) International Filing Date:
5 November 2010 (05.11.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/636,712 12 December 2009 (12.12.2009) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **WHEELER, Bradley**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **GRIFFIN, Bryan**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (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, RO, RS, RU, 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, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: CLOUD COMPUTING MONITORING AND MANAGEMENT SYSTEM

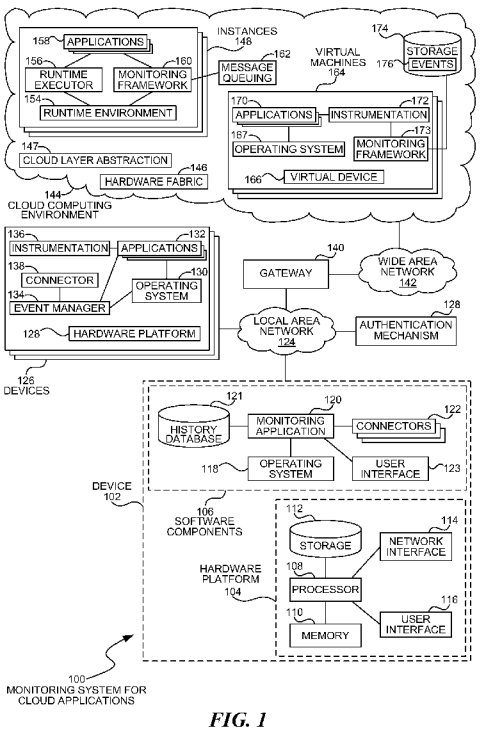

FIG. 1

(57) Abstract: A cloud computing monitoring system has an alert capturing system and a message transfer system that provides performance tracking and alert management to a local monitoring system. The alert capturing system may operate as part of a managed code framework and may capture and route alerts that may be transmitted to an operating system, as well as application exceptions and debugging information. A message queuing system may transmit the alerts to a local monitoring system, which may have a connector that subscribes to the cloud system's message queuing system.

WO 2011/071624 A2

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

CLOUD COMPUTING MONITORING AND MANAGEMENT SYSTEM

Background

[0001] Cloud computing is a computing paradigm that abstracts many aspects of a conventional computer. In a cloud environment, the hardware components may be abstracted into a hardware fabric. The hardware fabric may be many server computers located in one or more datacenters, and the datacenters may be geographically dispersed.

[0002] In many cloud environments, the conventional notion of an operating system may also be abstracted so that applications may operate in a runtime environment, but with limited access to operating system functions.

[0003] The cloud environment may execute applications in a manner that is highly scalable. A developer may provide an application to execute, and a management system may determine how much computing resources to allocate, the geographic location of those resources, and may determine which hardware platforms on which to execute the application. In some cases, an administrator may be able to determine some upper and lower limits to the computing resources, but the cloud management system may handle allocating the specific resources and managing the execution of the application.

[0004] Cloud environments may allow applications to scale up and down with load, as the cloud management system may allocate resources during high load periods and free up resources during low loads.

Summary

[0005] A cloud computing monitoring system has an event capturing system and a message transfer system that provides performance tracking and alert management to a local monitoring system. The event capturing system may operate as part of a managed code framework and may capture and route alerts that may be transmitted to a monitoring system, as well as application exceptions and debugging information. A message queuing system may transmit the events to a local monitoring system, which may have a connector that subscribes to the cloud system's message queuing system. The monitoring system may be a framework or executable code library that can be linked to and called by the application.

[0006] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

Brief Description of the Drawings

[0007] In the drawings,

[0008] FIGURE 1 is a diagram illustration of an embodiment showing a network environment in which a monitoring system may operate with a cloud computing environment.

[0009] FIGURE 2 is a timeline illustration of an embodiment showing a method of capturing, transmitting, and using monitored events.

[0010] FIGURE 3 is a diagram illustration of an embodiment showing a cloud computing environment with a monitoring framework.

Detailed Description

[0011] A cloud computing runtime environment may have a monitoring framework that includes executable routines for capturing and reporting errors, debugging information, performance information, status, and other information that may be transferred to a centralized monitoring application. The centralized monitoring application may collect information from multiple executing applications to provide alerting and management functions to administrators. In some embodiments, the monitoring application may be used in a network operations center for real time network and applications monitoring.

[0012] The monitoring framework may include functions for capturing information from an application and communicating that information to a monitoring application. The monitoring framework may receive the information to transmit, prepare messages from the information, and transmit those messages to the monitoring system in a format that the monitoring system may consume. In some embodiments, a message queuing system may be used to transmit the messages to a monitoring system that has a connector or other mechanism by which the monitoring system may subscribe to the message queue.

[0013] The monitoring framework may be included in a runtime environment for a cloud computing environment in some embodiments. The runtime environment may be a managed code environment that may include real time linking, garbage collection, and other services. In some embodiments, the monitoring framework may be included in a software development kit or other predefined set of executables against which an application may be developed, tested, and deployed in a cloud environment.

[0014] Throughout this specification, like reference numbers signify the same elements throughout the description of the figures.

[0015] When elements are referred to as being "connected" or "coupled," the elements can be directly connected or coupled together or one or more intervening elements may also be

present. In contrast, when elements are referred to as being "directly connected" or "directly coupled," there are no intervening elements present.

[0016] The subject matter may be embodied as devices, systems, methods, and/or computer program products. Accordingly, some or all of the subject matter may be embodied in hardware and/or in software (including firmware, resident software, micro-code, state machines, gate arrays, etc.) Furthermore, the subject matter may take the form of a computer program product on a computer-usable or computer-readable storage medium having computer-usable or computer-readable program code embodied in the medium for use by or in connection with an instruction execution system. In the context of this document, a computer-usable or computer-readable medium may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

[0017] The computer-usable or computer-readable medium may be for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media.

[0018] Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and may be accessed by an instruction execution system. Note that the computer-usable or computer-readable medium can be paper or other suitable medium upon which the program is printed, as the program can be electronically captured via, for instance, optical scanning of the paper or other suitable medium, then compiled, interpreted, of otherwise processed in a suitable manner, if necessary, and then stored in a computer memory.

[0019] Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" can be defined as a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By

way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above-mentioned should also be included within the scope of computer-readable media.

[0020] When the subject matter is embodied in the general context of computer-executable instructions, the embodiment may comprise program modules, executed by one or more systems, computers, or other devices. Generally, program modules include routines, programs, objects, components, data structures, and the like, that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0021] Figure 1 is a diagram of an embodiment 100, showing a cloud environment with a monitoring system. Embodiment 100 is a simplified example of a device with a monitoring system that may work in conjunction with applications in a cloud computing environment, where the cloud applications may use a monitoring framework to capture and transmit messages consumed by the monitoring application.

[0022] The diagram of Figure 1 illustrates functional components of a system. In some cases, the component may be a hardware component, a software component, or a combination of hardware and software. Some of the components may be application level software, while other components may be operating system level components. In some cases, the connection of one component to another may be a close connection where two or more components are operating on a single hardware platform. In other cases, the connections may be made over network connections spanning long distances. Each embodiment may use different hardware, software, and interconnection architectures to achieve the described functions.

[0023] Embodiment 100 is an example of a monitoring system that may monitor cloud applications. The monitoring system may monitor many different devices, applications, and other components, then consolidate the status of the components into a user interface. The monitoring system may be used to monitor the health and status of an enterprise's information technology infrastructure in a centralized manner.

[0024] The information infrastructure may include many different servers, services, applications, and other components. Some of the components may be local or on premise components, such as applications executing on the monitoring device as well as applications executing on devices within a local area network. Other components may be remote components, such as cloud applications.

[0025] Cloud computing environments have different variations. In one type of cloud computing environment, a virtual machine may be created and executed on a remote hardware fabric. The virtual machine may have an operating system or other function that may be configured and instrumented to interface with a monitoring system.

[0026] In another type of cloud computing environment, a hardware fabric may have multiple server devices that each have an operating system, and each may also have a cloud layer operating on top of the operating systems. The cloud layer may abstract the operating system from the application, and provide many automated management functions. The cloud layer may provide load balancing, redundancy, replication in different geographic areas, resource management, and other functions. In many implementations, the cloud layer may automatically manage the resources used to execute a cloud application.

[0027] Many cloud computing environments may implement a monitoring framework. A monitoring framework may be a set of functions that can be called by an application to capture information and transmit the information to a monitoring system. The monitoring framework may create messages that encapsulate the information to be transmitted, where the messages are compatible with a monitoring system both in format and delivery.

[0028] A device 102 may be used to monitor various components, including applications executed in a cloud environment. The device 102 may have a hardware platform 104 and various software components 106. The device 102 is illustrated as a standalone device on which a monitoring application may operate.

[0029] The hardware platform 104 may by a typical computing platform, such as a server or desktop computer. The hardware platform 104 may include a processor 108, random access memory 110, and nonvolatile storage 112. The hardware platform 104 may also include a network interface 114 as well as a user interface 116.

[0030] In many embodiments, the hardware platform 104 may be a server computer, but in other embodiments, the hardware platform 104 may be any type of computing device. For example, the hardware platform 104 may be a server computer, desktop computer, laptop computer, netbook computer, or other device. In some cases, the hardware platform 104 may be a mobile device, such as a personal digital assistant, portable computer, mobile telephone, or other mobile device.

[0031] The software components 106 may include an operating system 118 on which a monitoring application 120 may execute. The monitoring application 120 may have

several connectors 122 that may gather information from specific devices, applications, or other monitored components.

[0032] The monitoring application 120 may gather information from other components and aggregate the information into a user interface. In one use scenario, the monitoring application 120 may be used to monitor the performance, configuration, and use of hardware and software components within a company or other enterprise. The monitoring application 120 may alert an administrator when problems may come up on any of the monitored components, as well as give a status of the components. In many embodiments, the monitoring application 120 may provide real time or near-real time updates and status of the monitored components.

[0033] In some embodiments, the monitoring application 120 may have a user interface 123 in which the status of the monitored components and any alerts may be presented. The monitoring application 120 may use a messaging system to transmit alerts or other messages to various recipients using email, voicemail, or other mechanisms. The alerts may be emergency messages, for example, that may be transmitted when an urgent event may be detected.

[0034] The monitoring application 120 may have several connectors 122. The connectors 122 may connect with various information sources to collect and gather information about monitored components. The information sources may be any application, function, device, or other component that may generate alerts, events, performance data, or other information that may be consumed by a monitoring application 120. The connectors 122 may be active functions that may request information from a source as well as passive functions that may receive information on a periodic basis or when the information is available.

[0035] An active connector 122 may periodically initiate data collection. In some embodiments, a connector 122 may transmit a request for information to a service that may return the information requested. In another embodiment, a connector 122 may contact a remote data storage device and download or retrieve data stored on the storage device.

[0036] A passive connector 122 may receive a transmission that may be initiated by another device or service. The other device or service may transmit messages or other forms of information to the connector 122 on a periodic basis or when there is information to transmit.

[0037] Some connectors 122 may subscribe to a message queue to receive information updates. A connector 122 may initiate a subscription by contacting a queuing system or other messaging system using a Uniform Resource Identifier (URI) or other identifier or address. In some embodiments, such a connector 122 may present credentials to the messaging system to be able to receive messages.

[0038] Some messaging systems may create several subscriptions to which one or more monitoring systems may subscribe. For example, a cloud application may have several subscriptions available, such as one for high priority alerts, another one for operational status, and another one for debugging information. In some cases, the monitoring application 120 may subscribe to one or more of the available subscriptions.

[0039] The monitoring application 120 may gather many different types of information from a monitored component. The information may include general status information, performance information, alerts and emergency status, debugging information, and other information. Some embodiments may have different mechanisms for handling different types of information, including different manners for gathering, storing, processing, and presenting the information.

[0040] Status information may indicate the current operational state of a component. The status information may be updated when a component may be started, paused, stopped, or have other state changes. In many cases, the status information may be provided on a real time or near-real time basis. The status information may be used to present high level overview of the various monitored components. For example, a dashboard view of a set of monitored applications may be presented with green icons for operational status and red icons for stopped or paused status. A single user interface may provide a current status for many different monitored components in an easy to scan user interface.

[0041] Performance information may include various summary or detailed statistics about the operation of a component. For a monitored hardware component, performance information may include processor usage, disk capacity, network activity, memory usage, and other information. For a monitored software application, performance information may include number of requests processed, amount of data transmitted, or other performance metrics. The type of performance information may differ with different types of applications. Some performance information may be real time information that may be transmitted and displayed in a time sensitive manner. Other performance information may be historical information that may be transmitted to the monitoring application 120 on a delayed basis.

[0042] The performance information may be summary statistics as well as detailed information used to generate the summary statistics. Some embodiments may transmit summary statistics in different manner than the detailed information. For example, the summary statistics may be updated frequently using a message notification system that pushes the summary statistics to the monitoring application 120 while the detailed information may be stored in a remote database and pulled by the monitoring application 120 when the information may be requested.

[0043] In some embodiments, the summary statistics may be generated by a monitoring framework from the detailed performance information. In other embodiments, the detailed performance information may be transmitted to the monitoring application 120 and the summary statistics may be generated by the monitoring application 120.

[0044] Alert and emergency information may include high priority messages that may be used to identify actions that may be taken. For example, an alert may be generated when the available storage space has decreased to a very low limit, or when a terminal error has occurred with an application. In some embodiments, alerts may be processed by the monitoring application 120 and transmitted to an administrator using email, voicemail, or other mechanism.

[0045] Debugging information may be information that may be used by a developer to track down and solve problems. Debugging information may be very detailed and very voluminous. For example, debugging information may include indicators when a function is called along with parameters that are passed into or out of the function.

[0046] In some embodiments, the information transmitted to the monitoring system 120 may be defined by both policies and configuration settings. A policy may be a high level definition of the information that may be collected and transmitted. Some embodiments may have hierarchical policies where child policies inherit properties from parent policies. Configuration settings may include specific parameters, algorithms, or conditions that may be adjusted to determine which information to collect and transmit. Some embodiments may have other mechanisms for defining the information that may be collected and transmitted.

[0047] The monitoring application 120 may store the collected information in a history database 121. The history database 121 may contain information that was collected over a period of time. In some embodiments, the history database 121 may be used to generate summary statistics that may be displayed on the user interface 123, as well as detailed data that a user may display by drilling down into the information.

[0048] The monitoring application 120 may collect information from many different sources. The device 102 may be connected to a local area network 124 and to several devices 126 connected to the local area network 124. The devices 126 may have sources for information collected, managed, and displayed by the monitoring application 120.

[0049] The devices 126 may have a hardware platform 128, as well as an operating system 130 and various applications 132. The hardware platform 128 may be similar to the hardware platform 104 of device 102, and the devices 126 may be server computers, desktop computers, laptop computer, personal digital assistants, cellular telephones, network appliance, or any other computing device.

[0050] The applications 132 may have instrumentation 136 that may identify information to collect and cause the information to be stored in an event manager 134. The event manager 134 may be an application or operating system level function that collects various events that occur in the operating system 130 or applications 132 for administrative or other uses. A connector 138 may communicate with the connectors 122 to communicate information collected by the event manager 134 and transmit the information to the monitoring application 120.

[0051] The instrumentation 136 may be functions that are added to or called by the applications 132 to collect information. The instrumentation 136 may be routines that gather debugging information, performance information, status information, or other information. In some cases, the instrumentation 136 may monitor an application or operating system function to gather status, performance, and other information without being called directly by the application or operating system function.

[0052] A cloud computing environment 144 may also be monitored by the monitoring application. The device 102 may be connected to a local area network 124 to a gateway 140 to a wide area network 142. The wide area network 142 may be the Internet, for example. The cloud computing environment 144 may be connected to the Internet or wide area network 142.

[0053] The cloud computing environment 144 may have a hardware fabric 146 that is abstracted from a user, developer, or administrator. In many cases, a hardware fabric 146 may be a large datacenter or a group of large datacenters that may contain many hundreds, thousands, or even hundreds of thousands of computing devices. Many datacenters have redundant power sources, redundant network connections, and many failover mechanisms so that very high availability and very high uptimes are achieved.

[0054] The cloud computing environment 144 may include internal management systems that perform load balancing, clustering, and other functions that allow capacity to be allocated or de-allocated to certain processes or functions. In many cases, a cloud computing environment 144 may move certain processes or functions to specific geographical regions, transfer processes from one data center to another, move processes from one hardware platform to another, or perform other allocation processes without interaction with end users, developers, or administrators.

[0055] Some cloud computing environments may be shared environments, where a datacenter operator may offer cloud computing infrastructure for executing applications from many different customers. Each customer may have an application that is executed for that customer and is managed by the customer. Even though the application is managed by the customer, the underlying datacenter operations may be managed by the datacenter operator. The customer may use a monitoring framework to gather performance, status, debugging, and other information about their application separately from the monitoring and management operations that may be performed by the datacenter operator.

[0056] The cloud computing environment 144 may have a cloud layer abstraction 147 that may abstract instances 148 having a runtime environment 154. The cloud layer abstraction 147 may be a software layer that joins multiple hardware devices into a system where applications may be executed without the conventional notion of an operating system.

[0057] The runtime environment 154 may execute applications 158 using a runtime executor 156. The runtime executor 156 may perform linking and execution control of the applications 158, and the runtime environment 154 may provide additional management functions such as garbage collection, compilation, and other functions.

[0058] A monitoring framework 160 may be linked into and called by the applications 158. The monitoring framework 160 may be a set of functions that collect, process, and transmit information from the applications to a monitoring application 120.

[0059] In some embodiments, the monitoring framework 160 may use a message queuing system 162 to transmit information to the monitoring application 120. A message queuing system 162 may collect messages from various sources and make the messages available to a subscriber. In some aspects, the message queuing system 162 may operate like an email or other message system where the messages may be gathered together in a queue that can be accessed when the recipient is ready to receive the messages.

[0060] The message queuing system 162 may have a subscription service by which a recipient may receive messages. An intended recipient may contact the message queuing system 162 to receive messages, and the messages may be pushed to the recipient or pulled by the recipient. In the case of the monitoring application 120, a connector 122 may be configured to subscribe to a message queue and communicate with the message queuing system 162 to receive information.

[0061] Some message queuing systems may allow one and only one subscriber to a particular message queue. Other message queuing systems may permit multiple subscribers to a single queue.

[0062] In many embodiments, a cloud computing environment may have multiple instances 148 operating on many different physical machines and sometimes in many different datacenters that may be geographically dispersed around the globe. In such embodiments, a message queuing system 162 may act as a central repository for any messages created by the various instances and allow a monitoring application 120 to monitor all of the various instances of the application 158 as a single group or unit.

[0063] In some embodiments, a subscriber may present credentials or may otherwise be authenticated to the message queuing system 162. The authentication may be performed in many different manners. In some cases, a connector 122 may obtain an authenticated token from an authentication mechanism 178 and present the authenticated token to the message queuing system 162 to subscribe.

[0064] Some cloud computing environments 144 may use a virtual machine paradigm. Virtual machines 164 may have a virtual device 166 that may execute an operating system 167. Various applications 170 may execute within the operating system 167.

[0065] The virtual machine paradigm illustrated by the virtual machines 164 is different from the cloud layer abstraction of the instances 148 in that the operating system 167 may be exposed to, selectable by, and managed by the developer or administrator of the various applications. In the case of the instances 148, the operating system may not be accessed by the developer or administrator of the applications 158, but in the case of the virtual machines 164, the operating system 167 may be accessed by the developers or administrators of the applications 170.

[0066] The applications 170 may have instrumentation 172 and may access the monitoring framework 173. In some embodiments, the monitoring framework 173 may be the same as monitoring framework 160.

[0067] The monitoring framework 173 is illustrated as outputting information to a storage 174 in which various events 176 may be stored. The storage 174 may store information that may be pulled by a connector 122. The connector 122 may access the storage 174 to download the events 176 that may contain the information generated by the monitoring framework 173.

[0068] The storage 174 may illustrate a different transmission mechanism than the message queuing system 162. The message queuing system 162 may illustrate a mechanism by which information may be transmitted to a connector 122 using messages. In many such systems, a message queuing system 162 may have many features that may facilitate communication and security. These may include authentication, encryption, message storage, and other features. The transmission mechanisms of the message queuing system 162 may be a push type transmission where the message queuing system 162 may transmit messages to the connector 122 when the messages are available. Some message queuing systems may allow the connector 122 to request messages and may act as a pull type transmission. The storage 174 may be pull type transmission where a connector 122 may contact the storage 174 to retrieve the events 176.

[0069] In some embodiments, a monitoring framework may use both a message queuing system and a storage mechanism for transmitting information. In some such embodiments, certain classifications or types of data may be transmitted using one mechanism while other types of data may be transmitted using the other. For example, a monitoring framework may transmit alerts and emergency messages using a message queuing system but may transmit debugging information using a storage mechanism.

[0070] Figure 2 is a timeline illustration of an embodiment 200 showing a method for generating, transmitting, and using information produced by an application. Embodiment 200 illustrates the operations that may be performed by an application 202 in the left hand column, a monitoring framework 204 in the center column, and a monitoring application 206 in the right hand column. Embodiment 200 may illustrate some of the functions performed by applications 158 or 170, the monitoring frameworks 160 or 173, and the monitoring application 120 of embodiment 100.

[0071] Other embodiments may use different sequencing, additional or fewer steps, and different nomenclature or terminology to accomplish similar functions. In some embodiments, various operations or set of operations may be performed in parallel with other operations, either in a synchronous or asynchronous manner. The steps selected here were chosen to illustrate some principles of operations in a simplified form.

[0072] The application 202 may generate information in many different manners. For example, the application 202 may throw an exception in block 208, start or stop a process in block 210, or generate performance data in block 212. Other examples may include generating debugging information, capturing a data value, or encountering a predefined condition. The information generated by an application 202 may be created within the application itself or as part of an instrumentation framework that may be linked to and called from the application executable.

[0073] Any of the information generated in blocks 208-212 may be used to generate an event in block 214. The event of block 214 may be the information that may be transmitted by the application 202 and may be consumed by the monitoring application 206. The event may be transmitted by the application 202 and received by the monitoring framework 204 in block 216.

[0074] The monitoring framework 204 may perform some processing to the event prior to transmitting the event to the monitoring application 206. The processing may include filtering the event in block 218 as well as aggregating the event.

[0075] The filtering in block 218 may classify the event and determine how the event may be handled based on the classification. For example, some events may be identified as high priority events and may be expedited to the monitoring application 206, while other events may be ignored based on a policy or configuration setting and not transferred at all.

[0076] If the event is not to be transmitted in block 220, the event may be stored in block 222. In some embodiments, the event may be stored in a data storage system so that the monitoring application 206 may pull the information from the data storage system at a later time. In some situations, the event may be discarded.

[0077] If the event is to be transmitted in block 220 and not aggregated in block 224, a message may be formatted in block 227 and the event may be transmitted in block 228.

[0078] If the vent is to be aggregated in block 224, the event may be stored with other events for an aggregated transmission in block 226. An aggregated transmission may be performed in several manners. In one case, a set of events may be consolidated into a single message for transmission. Such events may be repeated instances of the same event or may be a group of similar or even unrelated events. In another case, an event that recurs multiple times may be consolidated into a single event that includes a count of the number of times the even occurred. For example, a known error event may be aggregated so that a single message may be transmitted after 100 occurrences of the event have been received.

[0079] The monitoring application 206 may receive the event in block 230. Various mechanisms may be used to transmit the event in block 228 and receive the event in block 230. Such mechanisms include mechanisms that push events from the monitoring framework 204 to the monitoring application 206, as well as mechanisms that pull events from the monitoring framework 204 to the monitoring application 206. The mechanisms may include message queuing systems, data storage systems, and other communication mechanisms.

[0080] After receiving the event in block 230, the event may be stored in block 232. The event may be classified in block 234 and if the event is a high importance event in block 236, an alert may be sent in block 238. If the event is not a high importance event in block 236, the event may be displayed in block 240.

[0081] Figure 3 is a diagram of an embodiment 300, showing a cloud environment with a monitoring system. Embodiment 300 is a simplified example of a runtime instance that may execute applications with a monitoring framework.

[0082] The diagram of Figure 3 illustrates functional components of a system. In some cases, the component may be a hardware component, a software component, or a combination of hardware and software. Some of the components may be application level software, while other components may be operating system level components. In some cases, the connection of one component to another may be a close connection where two or more components are operating on a single hardware platform. In other cases, the connections may be made over network connections spanning long distances. Each embodiment may use different hardware, software, and interconnection architectures to achieve the described functions.

[0083] Embodiment 300 is an example of a system that may operate as a cloud computing environment. A hardware fabric 302 may operate a software fabric 304 that provides an abstracted cloud computing layer. The cloud computing layer may include a runtime environment 306 that may have multiple instances, such as the instances 148 illustrated in embodiment 100.

[0084] The runtime environment 306 may include an execution engine 310 that may execute the application 312 as managed code. In some embodiments, a compiler 314 may compile the application 312 from source code or intermediate code into executable code. A linker 316 may link various frameworks, dynamic linked libraries, or other code elements to the application 312. Some embodiments may have the application 312 defined in an interpreted language.

[0085] The runtime environment 306 may include various managed code capabilities, such as dynamic linking, garbage collection 318, memory management, resource management, error capturing, and other features.

[0086] The application 312 may include instrumentation 320 that may identify and capture certain conditions, data, errors, performance metrics, or other events or information. The instrumentation 320 may call a monitoring framework 322 that may contain several functions. The monitoring framework 322 may process the information received from the application 312 and prepare the information to be transmitted to a monitoring system.

[0087] The monitoring framework 322 may have a receiving function 324 that may receive information from the application 312. The receiving function 324 may perform initial processing of the information, such as placing the received information into a format that may be used by the monitoring framework 322 for other functions as well as a format that may be used by a monitoring application. In some embodiments, the receiving function 324 may perform some handshaking with the application 312.

[0088] The receiving function 324 may gather other information in addition to the information received from the application 312. For example, the receiving function 324 may receive an error condition from the application 312. The receiving function 324 may gather other data, such as a timestamp, the values of some configuration settings, values of certain variables, or other information. The receiving function 324 may aggregate and organize the information into a format that may be used by other functions in the monitoring framework 322.

[0089] In some embodiments, the configuration settings 340 may indicate that the application 312 is operated in a debugging mode. A debugging mode may define a high level of debugging information that may be captured by the monitoring framework 322. In some cases, the debugging mode may be a setting used by the monitoring framework 322 to capture a higher level of detail. In other cases, the application 312 may be executed in a debugging mode so that the application 312 generates a larger quantity of events or with a higher level of detail than a normal operation.

[0090] A classification function 326 may be part of the monitoring framework 322 and may operate to classify the event. The classification may be used in conjunction with policies 338 and configuration settings 340 to determine how the event may be handled. Some events may be transmitted with high priority, while other events may be aggregated or even discarded.

[0091] A message generation function 328 may format the event and other information into a message that may be transported using a message queuing system 334 or stored in a data storage system 336.

[0092] A message aggregation function 330 may consolidate messages together into a single message. In some cases, multiple instances of a single message may be consolidated into a single message. In other cases, different messages may be grouped together into a single message.

[0093] A message transport function 332 may cause the message to be transported from the monitoring framework 322 to a monitoring application. The message transport function 332 may use the message queuing system 334, data storage system 336, or other mechanism to transport the message.

[0094] In some embodiments, the monitoring framework 322 may be the same or similar to a monitoring framework that may be used for application development. When used for application development, the monitoring framework 322 may be incorporated into a local application development platform and compiled and linked with the application. The application may be executed on a local device in debug or development mode so that a developer may test and refine the application. Once the application is ready to be deployed on the cloud, the application may be uploaded to the cloud and compiled and linked with the monitoring framework in the cloud.

[0095] The foregoing description of the subject matter has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the subject matter to the precise form disclosed, and other modifications and variations may be possible in light of the above teachings. The embodiment was chosen and described in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and various modifications as are suited to the particular use contemplated. It is intended that the appended claims be construed to include other alternative embodiments except insofar as limited by the prior art.

Claims

1. A cloud computing environment comprising:

a hardware fabric;

a runtime management system configured to execute an application;

a monitoring framework being a linkable library comprising:

a message configuration function configured to receive information from said application and create a message in a predefined format consumable by a monitoring application; and

a message transmission function configured to transmit said message to said monitoring application.

2. The cloud computing environment of claim 1 further comprising:

a message queuing system being called by said message transmission function, said message queuing system configured to receive messages from said monitoring framework and transmit said messages to said monitoring application.

3. The cloud computing environment of claim 2, said message queuing system comprising a message queue configured to store said messages until transmitting said messages to said monitoring application.

4. The cloud computing environment of claim 3, said message queuing system having a subscription mechanism for a plurality of message queues.

5. The cloud computing environment of claim 4, said monitoring application requesting a subscription to said message queue.

6. The cloud computing environment of claim 5, said message queuing system authenticating said monitoring application for said subscription.

7. The cloud computing environment of claim 1, said runtime management system comprising a managed code environment.

8. The cloud computing environment of claim 7, said monitoring framework being dynamically linked to said application.

9. The cloud computing environment of claim 8, said monitoring framework comprising a runtime version and a development version, said development version being executable in a conventional computing environment.

10. The cloud computing environment of claim 1, said information comprising exceptions thrown by said application and captured by said application.

11. The cloud computing environment of claim 1, said information comprising performance information generated by said application.

12. The cloud computing environment of claim 1, said information comprising debugging information generated by said application.

13. The cloud computing environment of claim 12, said debugging information being generated when said application is operated in a debugging mode.

14. A method performed by a cloud based runtime environment, said method comprising:

linking to a cloud application executing in a cloud environment;

receiving information to transmit to a monitoring application, said monitoring application being located on a remote device;

evaluating said information to determine an information type;

creating a message comprising at least a portion of said information, said message having a predefined format; and

transmitting said message to said monitoring application.

15. The method of claim 14, said cloud based runtime environment not having an operating system directly accessible by said cloud application.

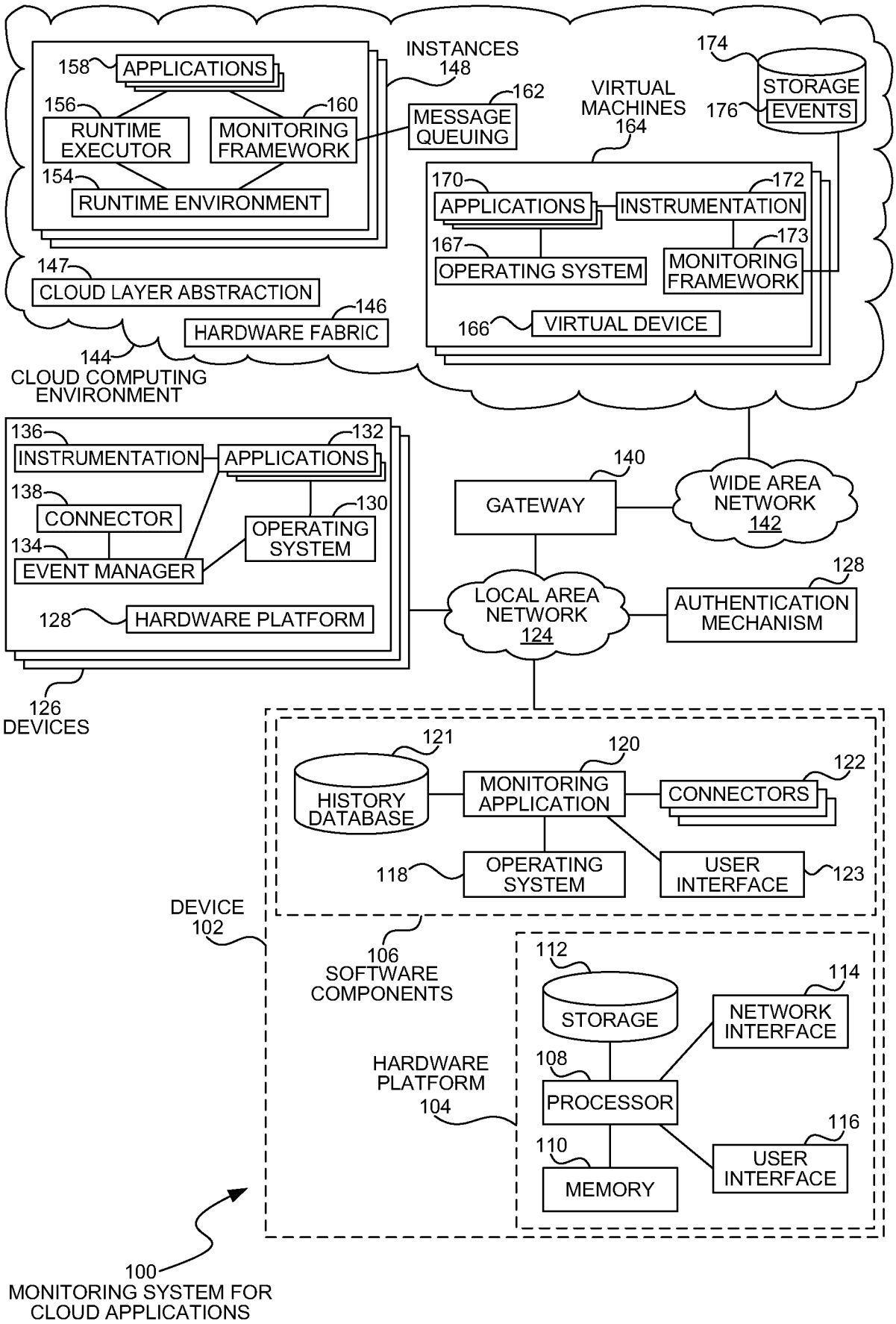
158
APPLICATIONS
156
RUNTIME EXECUTOR
160
MONITORING FRAMEWORK
154
RUNTIME ENVIRONMENT
INSTANCES 148
162
MESSAGE QUEUING
VIRTUAL MACHINES 164
174
STORAGE
176
EVENTS
170
APPLICATIONS
172
INSTRUMENTATION
167
OPERATING SYSTEM
173
MONITORING FRAMEWORK
166
VIRTUAL DEVICE
147
CLOUD LAYER ABSTRACTION
146
HARDWARE FABRIC
144
CLOUD COMPUTING ENVIRONMENT
136
INSTRUMENTATION
132
APPLICATIONS
138
CONNECTOR
130
OPERATING SYSTEM
134
EVENT MANAGER
128
HARDWARE PLATFORM
126
DEVICES
140
GATEWAY
WIDE AREA NETWORK 142
LOCAL AREA NETWORK 124
128
AUTHENTICATION MECHANISM
121
HISTORY DATABASE
120
MONITORING APPLICATION
122
CONNECTORS
118
OPERATING SYSTEM
USER INTERFACE
123
DEVICE 102
106
SOFTWARE COMPONENTS
112
STORAGE
114
NETWORK INTERFACE
HARDWARE PLATFORM 104
108
PROCESSOR
116
USER INTERFACE
110
MEMORY
100
MONITORING SYSTEM FOR CLOUD APPLICATIONS

FIG. 1

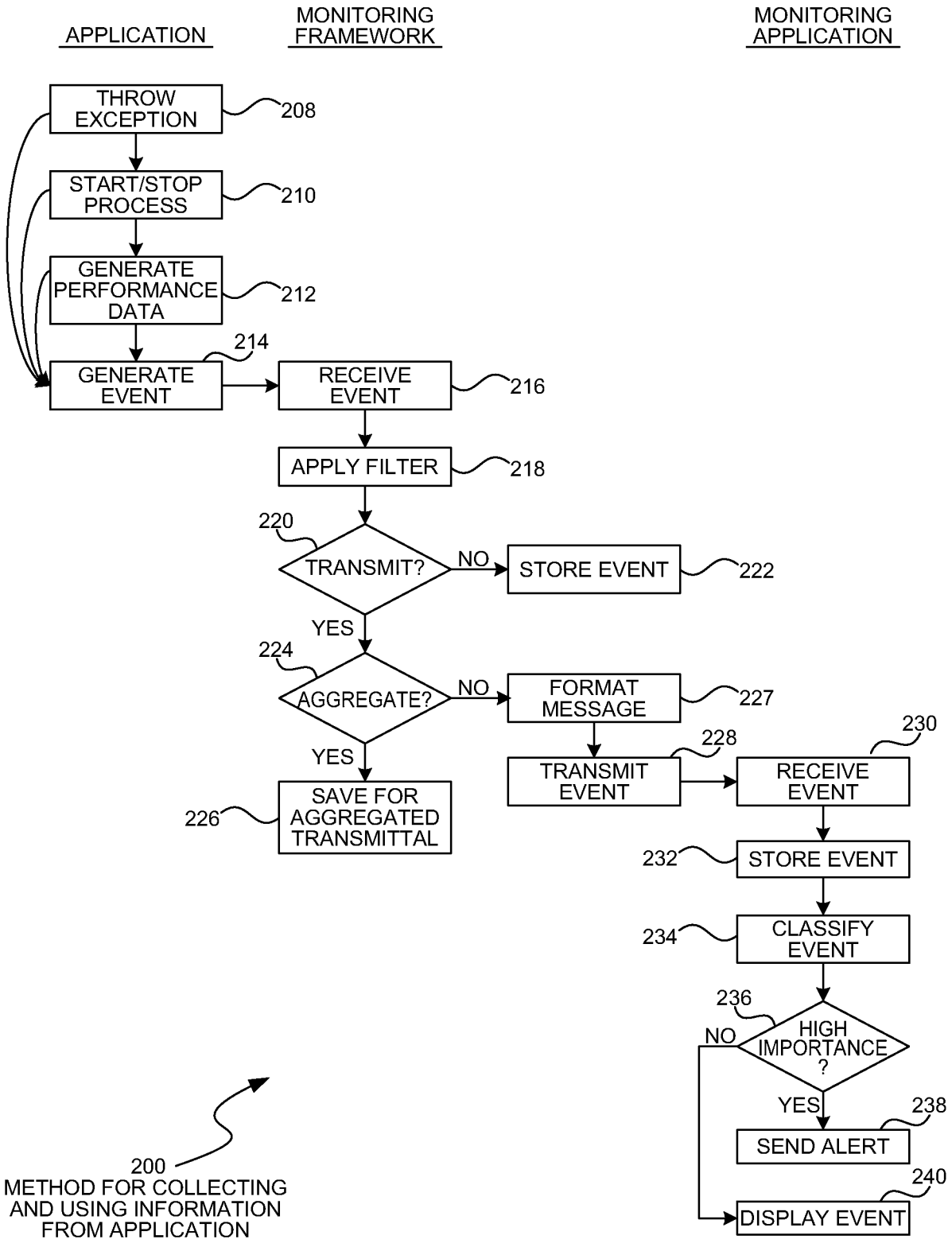
APPLICATION
MONITORING FRAMEWORK
MONITORING APPLICATION
THROW EXCEPTION
208
START/STOP PROCESS
210
GENERATE PERFORMANCE DATA
212
214
GENERATE EVENT
RECEIVE EVENT
216
APPLY FILTER
218
220
TRANSMIT?
NO
STORE EVENT
222
YES
224
AGGREGATE?
NO
FORMAT MESSAGE
227
YES
228
230
TRANSMIT EVENT
RECEIVE EVENT
226
SAVE FOR AGGREGATED TRANSMITTAL
232
STORE EVENT
234
CLASSIFY EVENT
236
NO
HIGH IMPORTANCE ?
YES
238
SEND ALERT
240
DISPLAY EVENT
200
METHOD FOR COLLECTING AND USING INFORMATION FROM APPLICATION

FIG. 2

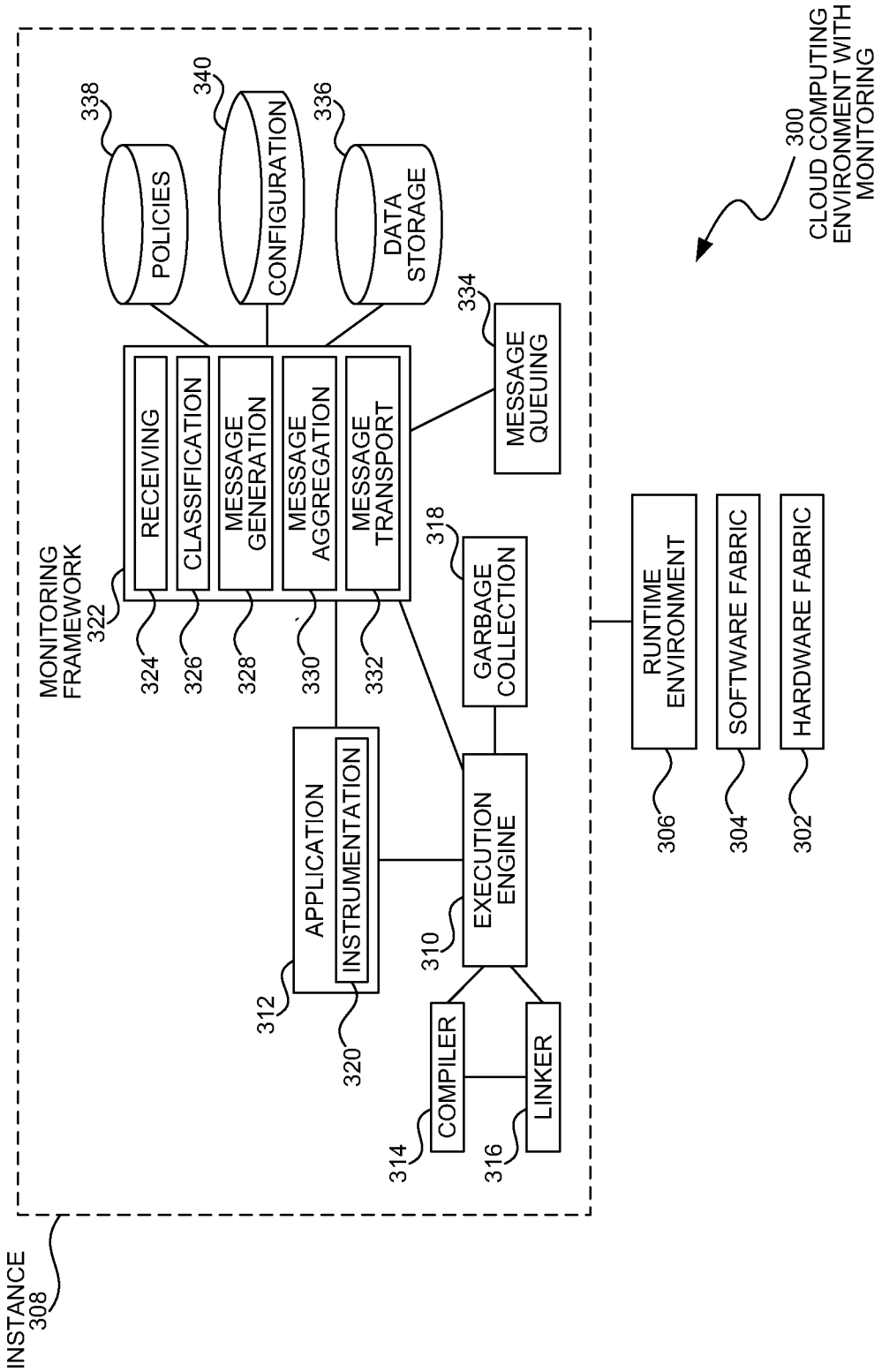
300
CLOUD COMPUTING ENVIRONMENT WITH MONITORING
INSTANCE
308
MONITORING FRAMEWORK
322
RECEIVING
CLASSIFICATION
MESSAGE GENERATION
MESSAGE AGGREGATION
MESSAGE TRANSPORT
324
326
328
330
332
POLICIES
CONFIGURATION
DATA STORAGE
338
340
336
MESSAGE QUEUING
334
GARBAGE COLLECTION
318
APPLICATION
INSTRUMENTATION
312
320
EXECUTION ENGINE
310
COMPILER
LINKER
314
316
RUNTIME ENVIRONMENT
SOFTWARE FABRIC
HARDWARE FABRIC
306
304
302

FIG. 3