(19) DANMARK

(10) **DK/EP 2619664 T3**

Patent- og Varemærkestyrelsen

(12) Oversættelse af europæisk patentskrift

(51) Int.Cl.: ***G 06 F 9/455 (2018.01)*** ***G 06 F 11/00 (2006.01)*** ***G 06 F 11/30 (2006.01)***

(45) Oversættelsen bekendtgjort den: **2022-03-07**

(80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2021-12-22**

(86) Europæisk ansøgning nr.: **11827381.2**

(86) Europæisk indleveringsdag: **2011-09-20**

(87) Den europæiske ansøgnings publiceringsdag: **2013-07-31**

(86) International ansøgning nr.: **US2011052413**

(87) Internationalt publikationsnr.: **WO2012040241**

(30) Prioritet: **2010-09-20 US 886138**

(84) Designerede stater: **AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**

(73) Patenthaver: **Netflix, Inc., 100 Winchester Circle, Los Gatos, California 95032, USA**

(72) Opfinder: **ORZELL, Gregory S., 45 Seward St. 2, San Francisco, California 94114, USA**
IZRAILEVSKY, Yury, 13991 Saratoga Ave., Saratoga, California 95070, USA

(74) Fuldmægtig i Danmark: **RWS Group, Europa House, Chiltern Park, Chiltern Hill, Chalfont St Peter, Bucks SL9 9FG, Storbritannien**

(54) Benævnelse: **VALIDERING AF TOLERANCEN AF APPLIKATIONER I NETVÆRK**

(56) Fremdragne publikationer:

US-A1- 2008 052 719
US-A1- 2010 223 378
US-B1- 6 694 346
William Hoarau ET AL: "Easy fault injection and stress testing with FAIL-FCI", , 1 January 2005 (2005-01-01), XP055265742, Retrieved from the Internet: URL:https://www.lri.fr/~bibli/Rapports-int ernes/2005/RR1421.pdf [retrieved on 2016-04-15]
HOARAU ET AL: "FAIL-FCI: Versatile fault injection", FUTURE GENERATIONS COMPUTER SYSTEMS, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 23, no. 7, 17 May 2007 (2007-05-17), pages 913-919, XP022081808, ISSN: 0167-739X, DOI: 10.1016/J.FUTURE.2007.01.005
BRENTN CHUN ED - TERUO HIGASHINO: "DART: Distributed Automated Regression Testing for Large-Scale Network Applications", 18 August 2005 (2005-08-18), PRINCIPLES OF DISTRIBUTED SYSTEMS; [LECTURE NOTES IN COMPUTER SCIENCE;;LNCS], SPRINGER-VERLAG, BERLIN/HEIDELBERG, PAGE(S) 20 - 36, XP019012325, ISBN: 978-3-540-27324-0 * page 235, paragraph 1 - page 237, paragraph 2 * * page 239, paragraph 3 - page 240, paragraph 1 *
Anonymous: "Log file - Wikipedia", , 7 December 2009 (2009-12-07), XP055754572, Retrieved from the Internet: URL:https://en.wikipedia.org/w/index.php?t itle=Log_file&oldid=330225477 [retrieved on 2020-11-27]

Fortsættes ...

DESCRIPTION

BACKGROUND OF THE INVENTION

Field of Invention

[0001] Embodiments of the present invention generally relate to techniques for assessing the resiliency of a distributed computing service provided by a collection of interacting servers.

Description of Related Art

[0002] A broad variety of computing applications have been made available to users over computer networks. Frequently, a networked application may be provided using multiple interacting computing servers. For example, a web site may be provided using a web server (running on one computing system) configured to receive requests from users for web pages. The requests can be passed to an application server (running on another computing system), which in turn processes the requests and generate responses passed back to the web server, and ultimately to the users.

[0003] Another example includes a content distribution system used to provide access to media titles over a network. Typically, a content distribution system may include access servers, content servers, *etc.,* which clients connect to using a content player, such as a gaming console, computing system, computing tablet, mobile telephone, network-aware DVD players, *etc.* The content server stores files (or "streams") available for download from the content server to the content player. Each stream may provide a digital version of a movie, a television program, a sporting event, user generated content, a staged or live event captured by recorded video, *etc.* Users access the service by connecting to a web server, where a list of content is available. Once a request for a particular title is received, it may be streamed to the client system over a connection to an available content server.

[0004] The software applications running on systems such as these are often updated as ongoing development results in patches to fix vulnerabilities or errors as well upgrades to make new features available. At the same time, the servers in a networked application may depend on one another in unforeseen or unintended ways and changes to one system may result in an unintended dependency on another. When this happens, if a server fails, then access to the networked application can be disrupted.

[0005] "East fault injection and stress testing with FAIL-FCI" by Hoarau et al discloses fault-loading existing distributed application and injecting specific faults at very specific moments in the program code of the application under test. "FAIL-FCI: Versatile fault injection" by Hoarau et al also discloses fault-loading existing distributed application and injecting specific faults at very specific moments in the program code of the application under test. "DART: Distributed Automated Regression Testing for Large-Scale Network Applications" by Chun, B N discloses a framework for distributed automated regression testing of large-scale network applications that provides a programming environment, scripted execution of multinode commands, fault injection, and performance anomaly injection

SUMMARY OF THE INVENTION

[0006] One embodiment of the invention disclosed herein provides a computer-implemented method for validating the resiliency of a networked application. The method may generally include identifying a plurality of active application components within a network used to provide the networked application and selecting, based on one or more selection criteria, at least one of the identified application components. This method may also include terminating the selected active application component and, following the termination of the selected active application component, monitoring one or more remaining active application components within the network.

[0007] Other embodiments include, without limitation, a computer-readable medium that includes instructions that enable a processing unit to implement one or more aspects of the disclosed methods as well as a system configured to implement one or more aspects of the disclosed methods.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] So that the manner in which the above recited features of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

Figure 1 illustrates a computing infrastructure configured to implement one or more aspects of the present invention.

Figure 2 illustrates a plurality of interacting server instances in a cloud computing environment, according to one embodiment of the present invention.

Figure 3 is a view of a computing system which includes a resiliency monitoring application, according to one embodiment of the invention.

Figure 4 further a method for validating the resiliency of networked applications, according to one embodiment of the present invention.

DETAILED DESCRIPTION

[0009] Embodiments of the invention provide techniques for validating the resiliency of a networked application made available using a collection of interacting servers. For example, a network monitoring application may be configured to terminate an instance of a running application to determine whether systems that depend on the failed one can still function correctly (or degrade gracefully) following a random, unanticipated failure. Thus, the monitoring application may observe the impact of a server failure on other systems in the networked application in a controlled manner. This approach may be useful in cloud based deployments where any server can disappear at any time.

[0010] In one embodiment, the network monitoring application observes each running server (or application) at unspecified intervals, picks one and terminates it. In the case of a cloud based deployment, this may include terminating a virtual machine instance, terminating a process running on the server, *etc.* For physical servers in a data center, it could involve shutting off a server, terminating a process running on the server, closing a network connection on the server, *etc.* However performed, the participation of the selected server in the network application ends, cutting off the server (or application) from the rest of the network application.

[0011] By observing the effects of the failed server on the rest of the network application, a provider can ensure that each component can tolerate any single instance disappearing without warning. In one embodiment, the network monitoring application may be used in a test environment prior to deploying an update or patch to servers (or applications) in a production environment. Doing so allows the effects of the update or patch to be evaluated without being deployed to the production environment. Further, certain applications (or hosts or systems) can be excluded (or included) from possible termination using an exclusion/inclusion list. Similarly, in a cloud based deployment, the network monitoring application can be configured to terminate server instances that are members of an auto scaling group. Doing so allows the functioning of the auto scaling processes to be evaluated against, randomly occurring server failures. Thus, in various embodiments, the network monitoring application helps enforce requirements for fault tolerance, which might otherwise be lost over time as production systems are upgraded, patched, or otherwise changed in manners that create unintended or unwanted dependencies. More generally, any logical group of systems may be defined and tested by the network monitoring application described herein.

[0012] In the following description, numerous specific details are set forth to provide a more thorough understanding of the present invention. However, it will be apparent to one of skill in the art that the present invention may be practiced without one or more of these specific details. In other instances, well-known features have not been described in order to avoid obscuring the present invention.

[0013] Further, particular embodiments of the invention are described using an example of a networked application used to stream movies, music, television programming, user generated content *etc.,* over a data communications network to end-user client devices. However, it should be understood that embodiments of the invention may be adapted to validate the resiliency to individual system failure for a broad variety of networked applications or services. Accordingly, references to a streaming media service are merely illustrative and not limiting.

[0014] Figure 1 illustrates a computing infrastructure 100 configured to implement one or more aspects of the present invention. In this example, the computing infrastructure 100 represents an infrastructure used to provide a networked application or service to client systems. As shown, server systems 137 at a data center 130 and virtual machine instances 145 running at cloud provider 140 are used to provide a network application or service to clients $105_{1\text{-}4}$ over a network 120. Additionally, the data center 130 includes an availability/reliability monitoring system 135.

[0015] As noted, embodiments of the invention may used to validate the resiliency of virtual machine instances 145 deployed using a cloud computing infrastructure made available by cloud provider 140. Cloud computing generally refers to the provision of scalable computing resources as a service over a network. More formally, cloud computing may be defined as a computing capability that provides an abstraction between the computing resource and its underlying technical architecture (*e.g.*, servers, storage, networks), enabling convenient, on-demand network access to a pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Thus, cloud computing allows a user to access virtual computing resources (*e.g.,* storage, data, applications, and even complete virtualized computing systems) in "the cloud," without regard for the underlying physical systems (or locations of those systems) used to provide the computing resources.

[0016] Typically, cloud computing resources are provided to a user on a pay-peruse basis, where users are charged only for the computing resources actually used (*e.g.,* an amount of storage space consumed by a user or a number of virtual machine instances spawned by the user). A user can access any of the resources that reside in the cloud at any time and from anywhere. Once provisioned, a virtual machine instance 145 provides an abstraction of a computing server, and a user can generally install and execute applications on the virtual machine instance 145 in the same manner as thought they controlled the physical computing server represented by the virtual machine instance 145.

[0017] In context of the present invention, a service provider may deploy servers or applications on virtual machine instance 145 and allow clients 105 to connect to and access the applications (*e.g.*, a streaming media service) in same manner as accessing physical server systems 137 in data center 130. At the same time, as the service provider can rapidly scale the service simply by spawning additional virtual machine instances 145. This allows the provider to respond to peak demand periods without having to build and maintain a large computing infrastructure at the data center 130. The cloud provider 140 may provide an auto scaling feature used to automatically scale up or down the number of virtual machine instances 145 allocated to a given application (or application component) based on its needs at any given time.

[0018] However, as service provider does not control the underlying computing hardware, the servers or applications executing on the virtual machine instances 145 should be configured to tolerate any single virtual instance (or service provided by the virtual machine instance) disappearing without warning. Accordingly, as described in greater detail below, monitoring system 135 may include an application program configured to periodically terminate an instance of a running application on the server systems 137 or virtual machine instances 145 and observe the impact on the service overall (in either production or test environments). And may also ensure that, following the termination of a virtual machine instance, instances associated with an auto scaling group are properly scaled. In cases where unknown or unwanted dependencies are identified, the applications may be refactored as appropriate.

[0019] Figure 2 illustrates a plurality of interacting server instances in a computing cloud 200, according to one embodiment of the invention. Illustratively, computing cloud 200 provides an example of a computing infrastructure used to provide a streaming media service to client systems. Of course, as noted above, computing cloud 200 could be used to provide a broad variety of computing services to clients 105.

[0020] As shown, the computing cloud 200 includes virtual machine instances (220, 225, 230 and 235) allocated among four different auto scaling groups $205_{1\text{-}4}$. Each auto scaling group may be associated with a minimum and/or maximum number of instances that should be provisioned, based on demand for services. In this example, a first auto scaling group 205_1 includes web server instances 220 used to receive initial requests from client systems 105. Once received, the request is passed to one of the application server instances 235 in a second auto scaling group 205_4, which may generate content for a web page passed back to the web server instance 220, where it is served to a requesting client system 105. For example, an initial web page may include a form allowing a user submit credentials in order to access streaming media content. In such a case, the credentials are passed back to the web server instance 220 and to the application server instance 235. And in turn, the application server 235 may validate the credentials by communicating with one of the database server instances 230 in a third auto scaling group 205_3. For example, the database server instance 230 may retrieve information from a database 210 indicating a subscription status for a user, determined using the credentials. Once authenticated, the application server instance 235 generates web pages showing the media titles available for streaming passed to the client 105 or the web server instance 220.

[0021] Thereafter, when a client requests to stream a title, one of the content streaming instances 225 (in a fourth auto scaling group 205_2) retrieves a streaming media data from a content database 215 and transmits it to the requesting client system 105. In a case where the streaming media service is hosted from a provider's data center, the virtual machine instances 220, 225, 230 and 235 generally correspond to physical computing systems in the data center.

[0022] In one embodiment, the monitoring system 135 may be configured to evaluate the resiliency of the streaming media service provided by the computing cloud 200 (or the systems/applications in a provider's data center). For example, the monitoring system 135 may select to terminate one of the instances 220, 225, 230 or 235 (or an instance selected from a specified one of the auto scaling groups $205_{1\text{-}4}$). The selection may be done at random intervals or may occur on a scheduled basis. Terminating an instance allows the provider to evaluate whether systems that depend on the terminated one continue to function correctly (or degrade gracefully) following a random, unanticipated failure. For example, if one of the application server instances 235 is terminated, the ability of content streaming instances 225 to continue to stream content to clients 105 may be observed. Thus, the monitoring system 135 allows users to observe the impact of a server failure on other systems in the networked application in a controlled manner

[0023] In one embodiment, some of the instances 220, 225, 230 and 235 may be excluded from being eligible to be terminated by the monitoring system 135. Individual instances may be excluded using an exclusion list. Such a list may exempt individual instances from being eligible for termination using an instance ID. For example, an instance used as an authentication server may be excluded from eligibility for termination. Similarly, if the failure of one instance (or an application provided by that instance) is known to be disruptive to others, it may be excluded from eligibility for termination.

[0024] Additionally, instances may be exempted based on group membership. For example, the monitoring system 135 may be configured to exclude all instances in a specified auto scaling group. Another example would be to exclude all instances which belong to a specific security group. Note, in this context, a security group is a group of systems to which a group of firewall-like access rules applied to any instance which is a member of the group. For example, the database server instances 230 could be a member of a "database" group that allows access to the application server instances 235, but blocks access to the web server instances 220. Similarly, the web server instances 220 could belong to a "web group," and be granted access to the public internet on a specified port (e.g., port 80 for HTTP traffic). Of course, other logical groups of systems may be defined and tested by the network monitoring application apart from the auto scaling and security group examples discussed above

[0025] Note, that while shown outside of the computing cloud 200, the monitoring system 135 may itself be running a virtual machine instance spawned in the computing cloud 200.

[0026] Figure 3 is a view of the monitoring system 135 which includes a resiliency monitoring application, according to one embodiment of the invention. As shown, the monitoring system 135 includes, without limitation, a central processing unit (CPU) 205, a network interface 315, an interconnect 320, a memory 325, and storage 330. The monitoring system 135 may also include an I/O device interface 310 connecting I/O devices 212 (e.g., keyboard, display and mouse devices) to the monitoring system 135.

[0027] In general, the CPU 305 retrieves and executes programming instructions stored in the memory 325. Similarly, the CPU 305 stores and retrieves application data residing in the memory 325. The interconnect 320 facilitates transmission of programming instructions and application data between the CPU 305, I/O devices interface 310, storage 330, network interface 315, and memory 325. CPU 305 is included to be representative of a single CPU, multiple CPUs, a single CPU having multiple processing cores, and the like. And the memory 325 is generally included to be representative of a random access memory. The storage 330 may be a disk drive storage device. Although shown as a single unit, the storage 330 may be a combination of fixed and/or removable storage devices, such as fixed disc drives, floppy disc drives, tape drives, removable memory cards, optical storage, network attached storage (NAS), or a storage area-network (SAN).

[0028] Illustratively, the memory 325 contains a monitoring application 321 and storage 330 includes monitoring logs 335. As shown, the monitoring application 321 includes a termination component 323, a recovery monitor 327, and instance monitoring parameters 329. As noted above, the monitoring application 321 may provide a software application configured to periodically select and terminate a running virtual machine instance, server, application or other component used in a networked application (e.g., one of the virtual machine instances 220, 225, 230 or 230 in cloud computing cloud 200 or a application running on a server in a provider's data center). For convenience, reference will be made terminating "instances," but, it will be understood that terminating any other component is contemplated.

[0029] In one embodiment, the termination component 323 selects which instance to terminate (as well as when to terminate an instance) according to the monitoring parameters 329. The selection parameters may specify criteria such as excluded instances, or groups of instances, times of day, *etc.,* which the termination component 323 may use to make a termination selection. For example, the monitoring parameters 329 may specify to select an instance at random (or select from a group at random) at any time or during a specified time interval.

[0030] Once an instance is selected (and terminated) the recovery monitor 327 may observe the actions of the remaining instances of a networked application, and generate corresponding information which recovery monitor 327 then stores the information in logs 335. The content of logs 335 may include information specified by the monitoring parameters 329 as well as include the logging data created natively by the instances (or applications running on an instance). That is, the applications running on an instance may generate logs depending on the applications running thereon (e.g., an access history log for a web-server).

[0031] Figure 4 illustrates a method 400 for validating the resiliency of networked applications, according to one embodiment of the present invention. As shown, the method 400 begins at step 405 where the monitoring application is initialized according to the configuration specified by the monitoring parameters. For example, the configuration parameters may specify criteria for determining when to select an instance to terminate, as well as for selecting which instance, application, or server, to terminate. At step 410, the monitoring application waits until reaching the time to terminate a running instance. Once reached, at step 415, the monitoring application identifies a plurality of active application components (e.g., active virtual machine instances, applications, or processes) being used to provide a network application.

[0032] At step 420, the monitoring application selects a virtual machine instance (or process or application) to terminate. Once selected, the monitoring application transmits a message to kill the selected instance. For example, in the case of a virtual machine instance, the monitoring application may transmit a terminate instance message to the cloud network. Alternatively, the monitoring application may shutdown a server program (*e.g.,* an HTTP web server) on a virtual machine instance (or on a data center server) or use mechanisms provided an operating system to kill a process.

[0033] At step 425, the monitoring application waits for the selected instance to terminate (or otherwise shutdown or cease executing). Following the termination of the selected instance, at step 430, the monitoring application observes the behavior of the remaining instances (or applications) and records log data to capture how the disappearance of the terminated instance impacts the rest of the network application. By observing the effects of the terminated on the rest of the network application, a provider can ensure that each component can tolerate any single instance disappearing without warning.

[0034] Advantageously, embodiments of the invention provide techniques for validating the resiliency of a networked application made available using a collection of interacting servers. In one embodiment, a network monitoring application observes each running server (or application) and at unspecified intervals, picks one and terminates it. In the case of a cloud based deployment, this may include terminating a virtual machine instance or terminating a process running on the server. Doing so may test the effectiveness of an auto-scaling (or other logical group of systems) made available by a cloud service provider. Additionally, some systems (or groups of systems) may be excluded from being eligible for termination by the network monitoring application (e.g., systems belonging to a security group).

[0035] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof. For example, aspects of the present invention may be implemented in hardware or software or in a combination of hardware and software. One embodiment of the invention may be implemented as a program product for use with a computer system. The program(s) of the program product define functions of the embodiments (including the methods described herein) and can be contained on a variety of computer-readable storage media. Illustrative computer-readable storage media include, but are not limited to: (i) non-writable storage media (e.g., read-only memory devices within a computer such as CD-ROM disks readable by a CD-ROM drive, flash memory, ROM chips or any type of solid-state non-volatile semiconductor memory) on which information is permanently stored; and (ii) writable storage media (e.g., floppy disks within a diskette drive or hard-disk drive or any type of solid-state random-access semiconductor memory) on which alterable information is stored. Such computer-readable storage media, when carrying computer-readable instructions that direct the functions of the present invention, are embodiments of the present invention.

[0036] Therefore, the scope of the present invention is determined by the claims that follow.

REFERENCES CITED IN THE DESCRIPTION

Cited references

Non-patent literature cited in the description

- **HOARAU**East fault injection and stress testing with FAIL-FCI, [0005]
- **HOARAU**FAIL-FCI: Versatile fault injection, [0005]
- **CHUN, B N**DART: Distributed Automated Regression Testing for Large-Scale Network Applications, [0005]

Patentkrav

1. Computerimplementeret fremgangsmåde (400) til at validere tolerancen i en netværksapplikation, idet fremgangsmåden omfatter:
identificering (415) af en flerhed af aktive applikationskomponenter i et netværk, der bruges til at tilvejebringe netværksapplikationen, hvor hver af flerheden af aktive applikationskomponenter omfatter en virtuel maskinforekomst, der udfører i en computersky, hvor autoskalering anvendes i computerskyen til automatisk op- eller nedskalering af et antal virtuelle maskinforekomster, der er allokeret til en given applikationskomponent på basis af behov ved et givent tidspunkt,
udvælgelse (420) af mindst én af de identificerede applikationskomponenter på basis af:
en tilsvarende virtuel maskinforekomst, der er medlem af en første autoskaleringsgruppe, der er indbefattet af en flerhed af forskellige autoskaleringsgrupper, og
et udvælgelseskriterie, der udelukker (1) virtuelle maskinforekomster, der er medlem af en sikkerhedsgruppe, som specificerer én eller flere netværksadgangsregler, der er anvendt på medlemmerne af sikkerhedsgruppen, og (2) virtuelle maskinforekomster, som når de fejler, vides at forstyrre andre virtuelle maskinforekomster,
afslutning (425) af den valgte applikationskomponent, og
efter afslutningen af den valgte applikationskomponent, overvågning (430) af handlingerne af én eller flere resterende aktive applikationskomponenter i netværket for at evaluere funktionen af autoskaleringen i forhold til tilfældigt forekommende serverfejl, og
oprettelse af én eller flere logposter, der registrerer data, som svarer til handlingerne af de resterende aktive applikationskomponenter, som følger af afslutningen af den valgte aktive applikationskomponent.

2. Den computerimplementerede fremgangsmåde ifølge 1, hvor hver autoskaleringsgruppe i flerheden af forskellige

autoskaleringsgrupper angiver mindst én af et minimums- eller maksimumsantal af virtuelle maskinforekomster, der opsplittes i autoskaleringsgruppen.

3. Den computerimplementerede fremgangsmåde ifølge et hvilket som helst af kravene 1 eller 2, hvor afslutning af den valgte aktive applikationskomponent omfatter afslutning af udførelsen af den valgte virtuelle maskinforekomst i computerskyen.

4. Den computerimplementerede fremgangsmåde ifølge et hvilket som helst af de foregående krav, hvor den valgte aktive applikationskomponent omfatter en angivet proces, der udføres på en virtuel maskinforekomst, og hvor afslutning af den valgte applikationskomponent omfatter standsning af udførelsen af den angivne proces.

5. Computerprogramprodukt, der omfatter instruktioner, som, når de udføres af en
processor, får processoren til at udføre en operation til validering af tolerancen af en netværksapplikation, hvor operationen omfatter den computerimplementerede fremgangsmåde ifølge et hvilket som helst af de foregående krav.

6. Computerprogramproduktet ifølge krav 5, der omfatter et computerlæsbart medie, der indbefatter instruktionerne.

7. System (315), der omfatter:
en processor (305), og
en hukommelse (325), der indeholder et program (321), der, når de udføres af processoren, udfører en operation til at validere tolerancen for en netværksapplikation, idet operationen omfatter den computerimplementerede fremgangsmåde ifølge et hvilket som helst af kravene 1 til 4.

DRAWINGS

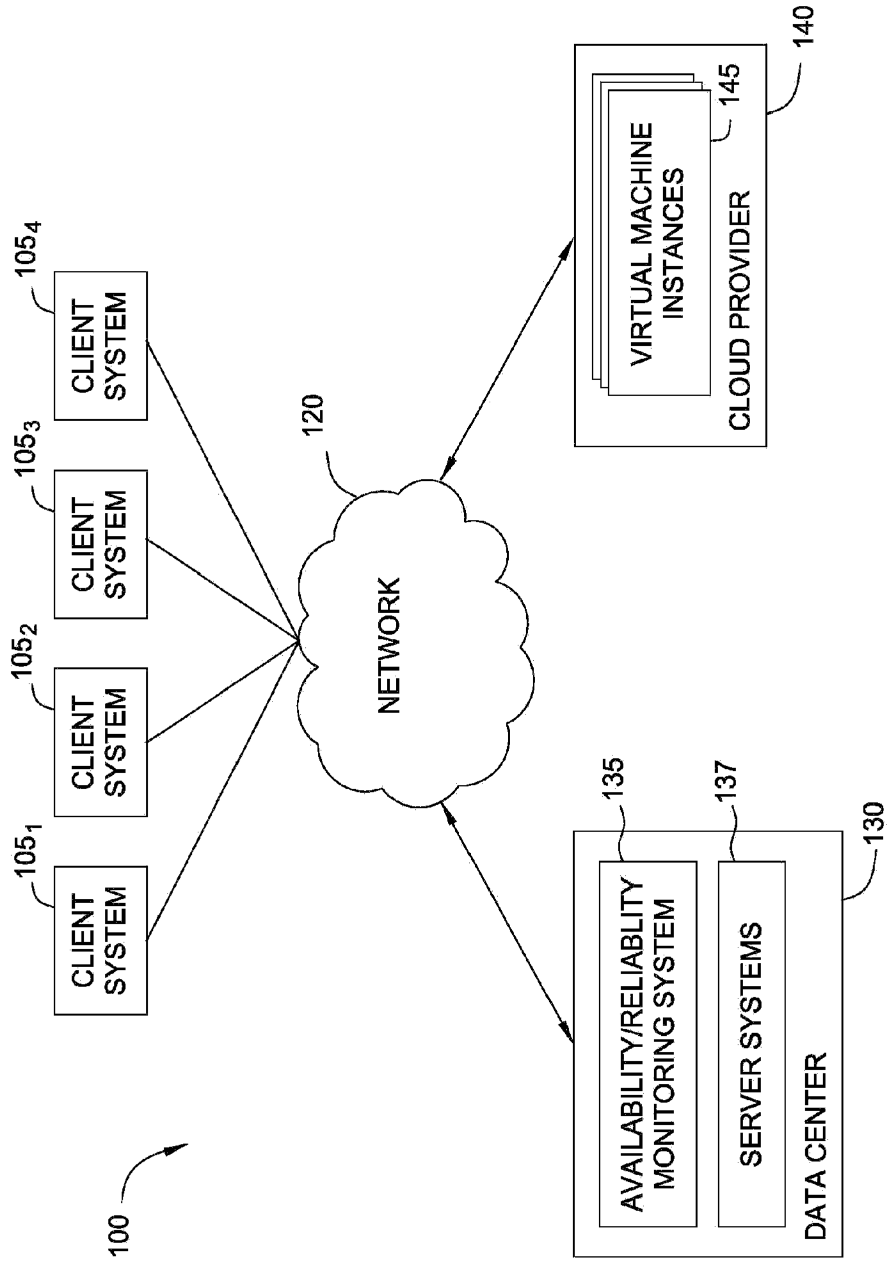

FIG. 1

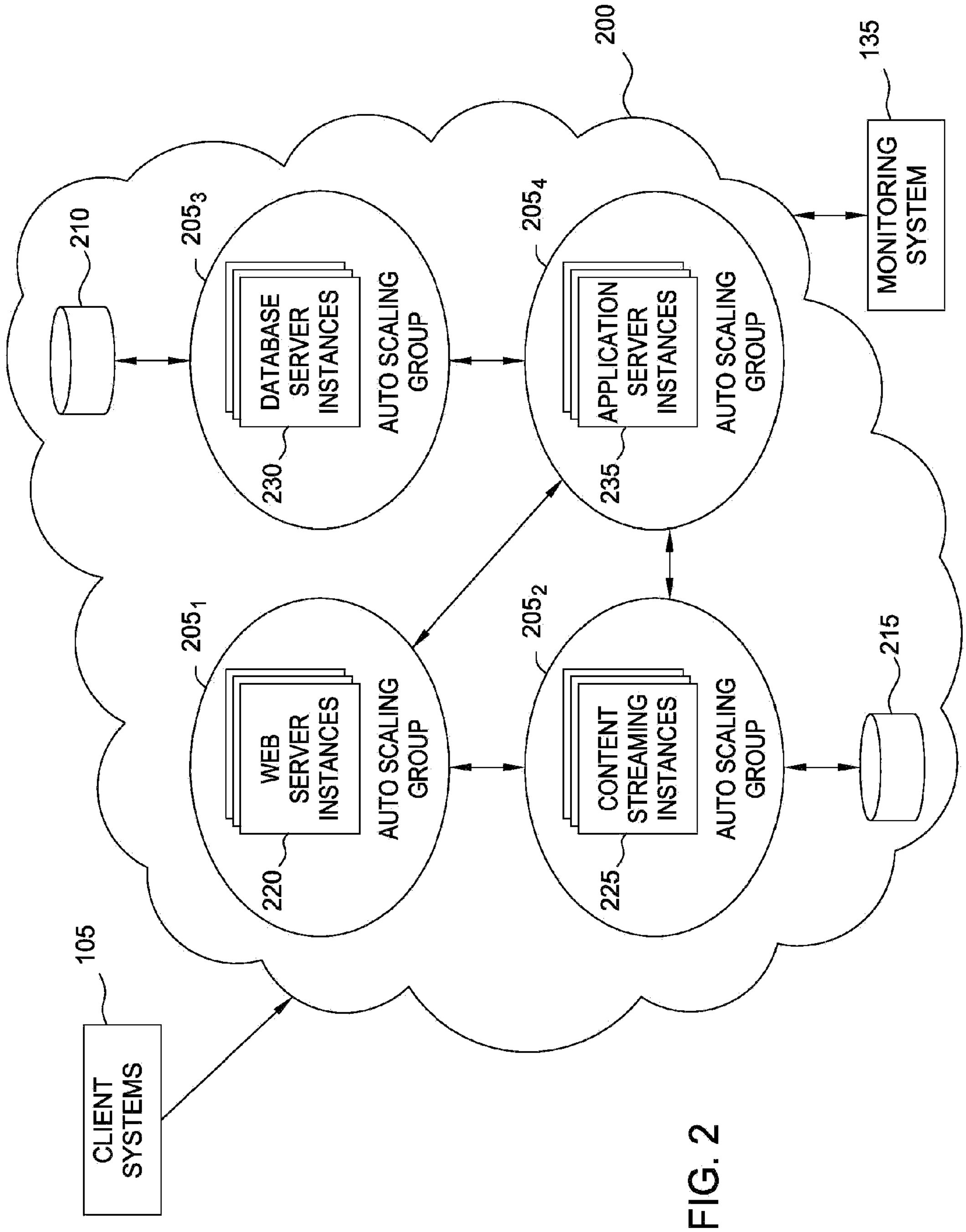
200
210
205₃
DATABASE SERVER INSTANCES
230
AUTO SCALING GROUP
205₄
APPLICATION SERVER INSTANCES
235
AUTO SCALING GROUP
135
MONITORING SYSTEM
205₁
WEB SERVER INSTANCES
220
AUTO SCALING GROUP
205₂
CONTENT STREAMING INSTANCES
225
AUTO SCALING GROUP
215
105
CLIENT SYSTEMS

FIG. 2

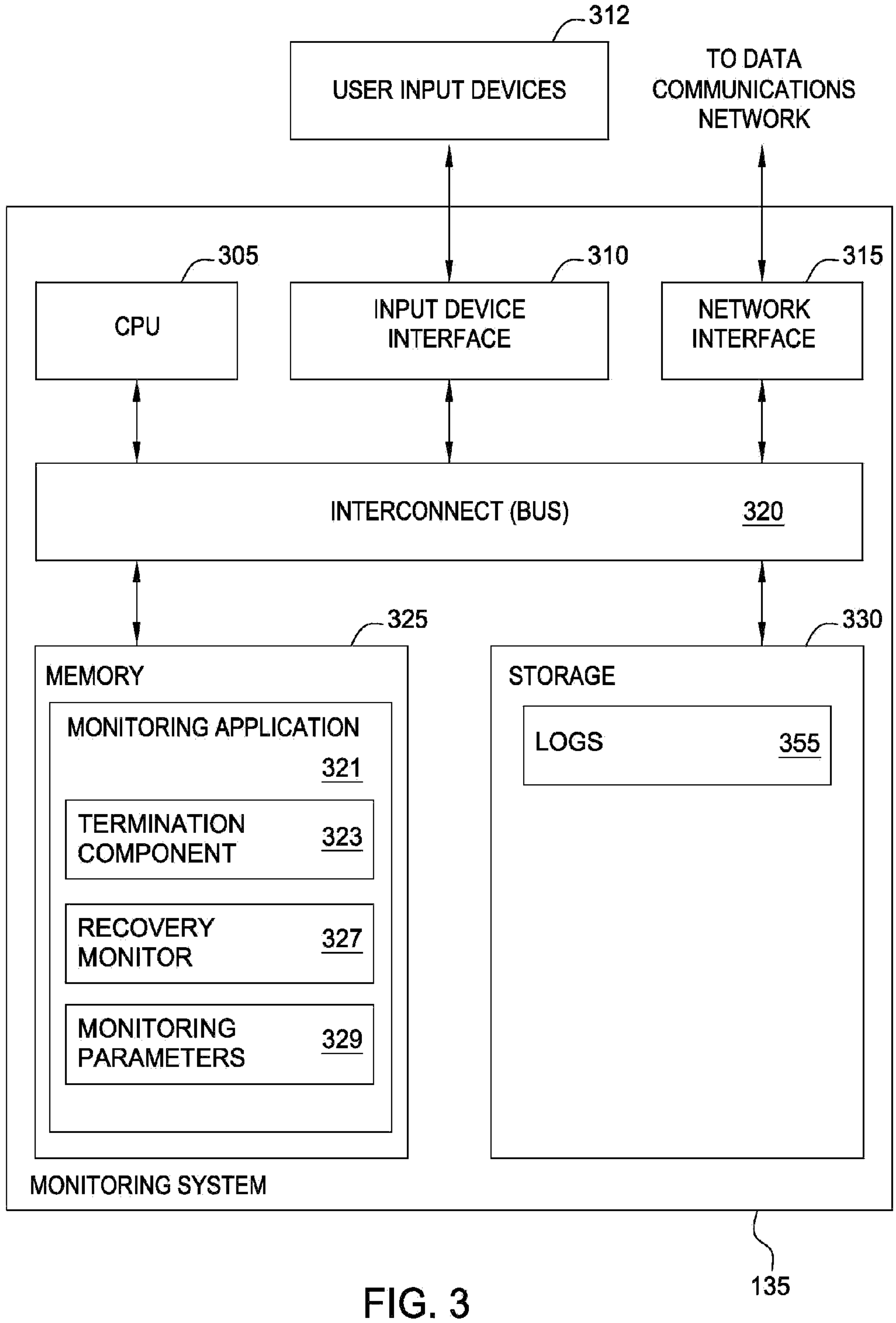
312
USER INPUT DEVICES
TO DATA COMMUNICATIONS NETWORK
305
CPU
310
INPUT DEVICE INTERFACE
315
NETWORK INTERFACE
INTERCONNECT (BUS)
320
325
MEMORY
MONITORING APPLICATION
321
TERMINATION COMPONENT
323
RECOVERY MONITOR
327
MONITORING PARAMETERS
329
330
STORAGE
LOGS
355
MONITORING SYSTEM
135

FIG. 3

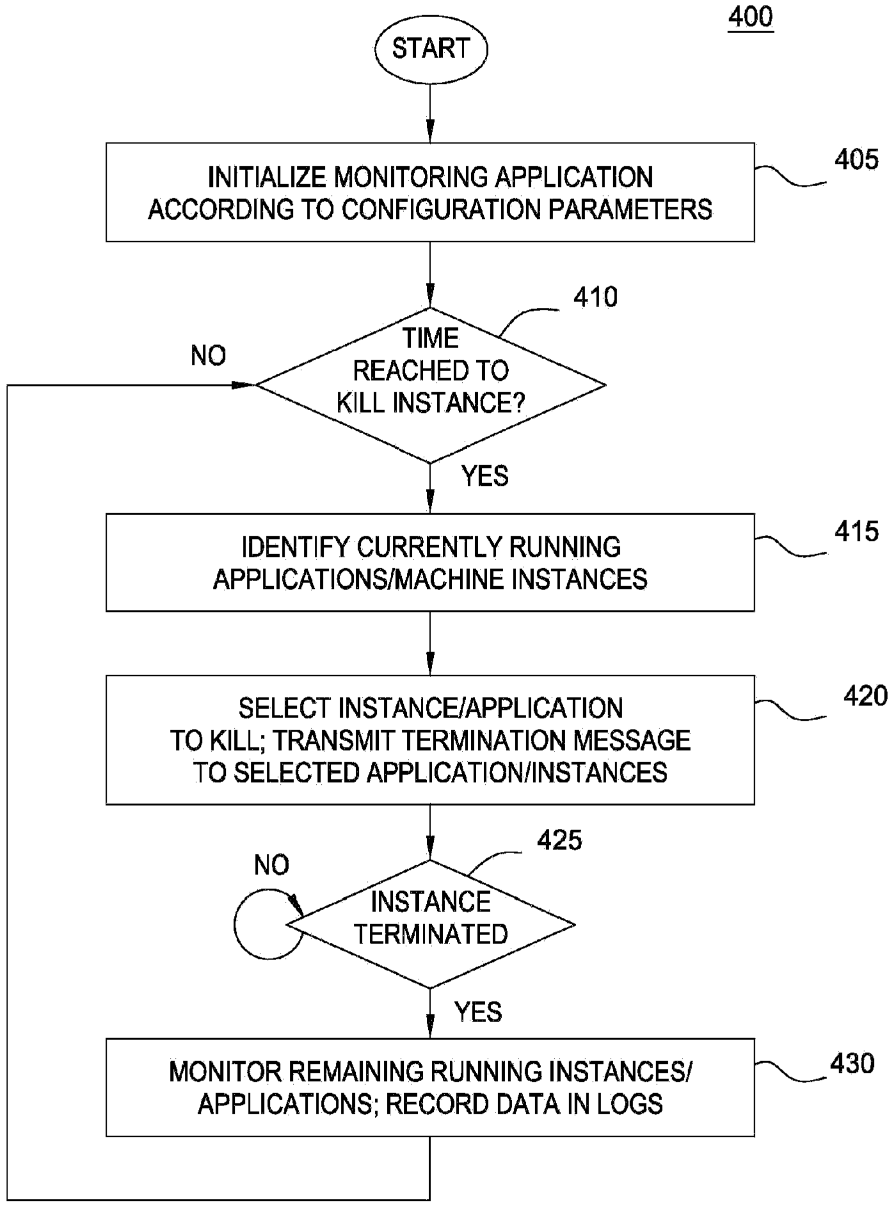
400
START
INITIALIZE MONITORING APPLICATION ACCORDING TO CONFIGURATION PARAMETERS
405
410
NO
TIME REACHED TO KILL INSTANCE?
YES
IDENTIFY CURRENTLY RUNNING APPLICATIONS/MACHINE INSTANCES
415
SELECT INSTANCE/APPLICATION TO KILL; TRANSMIT TERMINATION MESSAGE TO SELECTED APPLICATION/INSTANCES
420
425
NO
INSTANCE TERMINATED
YES
MONITOR REMAINING RUNNING INSTANCES/ APPLICATIONS; RECORD DATA IN LOGS
430

FIG. 4