(19) DANMARK

(10) **DK/EP 2863306 T3**

Patent- og Varemærkestyrelsen

(12) Oversættelse af europæisk patentskrift

(51) Int.Cl.: ***G 06 F 9/50 (2006.01)*** ***G 06 N 5/04 (2006.01)*** ***H 04 L 12/911 (2013.01)***

(45) Oversættelsen bekendtgjort den: **2021-07-26**

(80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2021-05-12**

(86) Europæisk ansøgning nr.: **14187822.3**

(86) Europæisk indleveringsdag: **2014-10-06**

(87) Den europæiske ansøgnings publiceringsdag: **2015-04-22**

(30) Prioritet: **2013-10-18 US 201314057898**

(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: **Jacobson, Daniel Isaac, 305 Langton Avenue, Los Gatos, CA 94022, USA**
Joshi, Neeraj, 496 King George Avenue, San Jose, CA 95136, USA
Oberai, Puneet, 800 Bauer Drive, San Carlos, CA 94070, USA
Yuan, Yong, 297 San Moreno Pl., Fremont, CA 94070, USA
Tuffs, Philip Simon, 381 Junipero Avenue, Pacific Grove, CA 93950, USA

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

(54) Benævnelse: **Forudsigende autoskaleringsprogram**

(56) Fremdragne publikationer:
WO-A1-2013/006708
US-A1- 2011 258 621
US-A1- 2011 295 999
US-A1- 2012 259 976
US-A1- 2013 174 146

DESCRIPTION

BACKGROUND

Technical Field

[0001] Embodiments generally relate to workload management, and more specifically to predictively scaling a number of instances of an application executing within a cloud computing environment in anticipation of the application's future workload.

Description of Related Art

[0002] Efficient resource allocation is a constant challenge for modern cloud computing environments. For example, a particular application may require 100 application instances in order to process its peak workload, but could require only 40 application instances for processing its average workload. In this example, the application could be configured to operate using 40 application instances, but such a configuration would fail to accommodate the application's peak workload. Likewise, the application could be configured to operate using 100 application instances all of the time, but such a configuration would lead to an inefficient use of resources, as application instances may sit idle or are underutilized during times of non-peak workload. As such, cloud solutions may attempt to reactively scale the number of application instances in order to meet the fluctuating workload. That is, logic could determine when the application resources are sitting idle or when the application is unable to keep up with its current workload, and could scale the number of application instances down or up accordingly. However, as the application's workload is almost certainly dynamic in practice, it can be challenging to scale the number of application instances in order to accommodate the increasing or decreasing workload.

[0003] As an additional challenge, the start-up process of many applications may require a substantial amount of time. Thus, if the additional application instances are not created until the system detects the application is underperforming relative to the current workload, and the application instances take a substantial amount of time to initialize, the application may fall further behind relative to its current workload while waiting on the application instances to initialize. Such a scenario can lead to further underperformance by the application, and in some circumstances can lead to the system reactively spawning even more additional application instances in order to address the underperformance, resulting in an excessive number of application instances. Moreover, in the event such reactive scaling techniques are not fast enough to meet the increasing workload, the system may fail to catch up altogether, potentially resulting in interruptions, errors or even a complete system failure. WO 2013/006708 discloses a method and apparatus for automatic provisioning steps using a

physical computing block-based infrastructure platform and a virtualized environment to provide automatic elasticity. US 2011/258621 discloses autonomic scaling of virtual machines in a cloud computing environment. US 2013/174146 discloses dynamically scaling multi-tier applications in a cloud environment. US 2011/0295999 discloses a decision system and methods for providing customized deployment architectures that take into account the importance of various cloud resources. US 2012/0259976 discloses a framework to improve the predictability of performance problems in an enterprise application.

SUMMARY

[0004] One embodiment disclosed herein provides a method according to claim 1.

[0005] Other embodiments include, without limitation, a computer program product according to claim 10 as well as a system according to claim 9.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The invention is defined by the independent claims. Further aspects of the invention are outlined in the dependent claims. Embodiments which do not fall within the scope of the claims do not describe part of the present invention.

Figure 1 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment described herein.

Figure 2 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment described herein which is not covered by the claims.

Figure 3 is a flow diagram illustrating a method for determining performance patterns of a cloud application, according to one embodiment described herein.

Figure 4 is a flow diagram illustrating a method for predictively scaling instances of a cloud application, according to one embodiment described herein.

Figure 5 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment described herein.

DETAILED DESCRIPTION

[0007] Generally speaking, many large-scale applications comprise a number of different application instances. The application's workload can be distributed throughout these

application instances through the use of a load balancing algorithm(s). Generally, there are a number of advantages to designing applications in this fashion. For instance, application instances can be added or removed in order to dynamically scale the size of the application, allowing for the application to handle an increased workload *(i.e.,* by adding instances) or to free-up idle or underutilized computing resources *(i.e.,* by removing instance). Moreover, if one application instance fails, the remaining application instances can take on the failing instance's workload, providing redundancy for the application.

[0008] One challenge in managing a large-scale application (*e.g.*, a video streaming service and its supporting applications) running in a cloud environment is that the workload of such an application can vary dramatically over time. For example, such a video streaming service may experience its peak workload in the evenings during peak television viewing hours, yet the same streaming service could experience a much lighter workload in the early morning hours when viewer demand is not as great. Moreover, as the video streaming service adds new viewers, the peak workload on a given night of the week could increase substantially on a week-over-week basis, in accordance with the growth in viewers. This may be particularly pronounced during times of rapid growth in viewership, e.g., during a promotional period for the video streaming service.

[0009] Currently, cloud computing environments may be configured to reactively scale the number of application instances, based on the application's current workload at a given point in time. For example, the cloud computing environment could detect that the currently instantiated application instances are unable to keep up with the number of incoming requests, and thus could determine that additional application instances should be instantiated in order to help meet the application's current workload.

[0010] While such reactive scaling techniques can perform adequately in certain situations (e.g., when the start-up time of the application is relatively short, when the workload changes at a relatively slow rate, *etc.),* these techniques are ill-suited for use with some applications. For example, the application instances for a particular application may take over an hour to fully initialize. With such an application, if the additional application instances are not created until the current number of application instances are unable to keep up with the current workload *(i.e.,* a reactive scaling solution), a backlog of incoming requests may continue to accumulate for over an hour until the additional application instances are brought online. As such, users may experience delays or service interruptions during this start-up time. Moreover, as the workload may have continued to increase during this lengthy start-up time, the number of additional application instances may no longer be sufficient by the time the additional application instances are fully initialized. This problem is compounded, as the backlog of requests built up during the additional application instance's start-up time must also be processed.

[0011] As such, embodiments provide techniques for predictively scaling a number of application instances for an application in a cloud computing environment. Embodiments monitor performance of an application within a cloud computing environment over a first

window of time to collect historical performance data. For example, embodiments could collect performance information such as how many requests per unit of time an application instance can process and an average start-up time for creating a new application instance. Additionally, embodiments monitor a workload of the application over a second window of time to collect historical workload data. Here, embodiments collect information as to how the application's workload varies throughout the day, as well as how the workload varies on individual days of the week.

[0012] Embodiments could then analyze both the historical performance data and the historical workload data to determine one or more scaling patterns for the application. For example, embodiments perform a statistical analysis of the collected data in order to determine one or more patterns that are indicative of an increase or decrease in the application's workload. Additionally, embodiments determine that a present state of the application matches one of the one or more scaling patterns, and in response, determine a plan for predictively scaling the application. For example, embodiments could determine a number of application instances that are likely to be needed at a future point in time, and could establish a timeline for scaling the current number of application instances in order to achieve the determined number of application instances by the future point in time. Embodiments then predictively scale the plurality of application instances, based on the determined plan. Advantageously, doing so provides a predictive scaling solution that can anticipate fluctuations in the application's workload and can scale the number of application instance's accordingly.

[0013] Figure 1 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment described herein. As shown, the system 100 includes a plurality of client devices 110 and a cloud computing environment, connected via a network 115. 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 shared 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. Typically, cloud computing resources are provided to a user on a pay-per-use 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 virtualized systems instantiated by the user). A user can access the resources that reside in the cloud at any time, and from anywhere across the Internet.

[0014] Here, the cloud computing environment 120 includes a predictive scaling component 125, a plurality of compute nodes 130 and historical performance data 140. Generally, the compute nodes 130 reflect computing systems (*e.g.*, virtual machines) within the cloud computing environment 120. Here, each of the compute nodes 130 includes an application

instance 135 of a distributed application. For purposes of this example, assume that the application instances 135 hosted on the compute nodes 130 all correspond to a single distributed application (*e.g.*, a video streaming service). Of note, while a single application instance 135 is shown within each of the compute nodes 130, such a depiction is for illustrative purposes only, and other embodiments may instantiate 2 or more application instances 135 on a compute node 130. More generally, embodiments described herein may be configured for use with any configuration for hosting the application instances 135 within the cloud computing environment 120.

[0015] Here, the predictive scaling component 125 monitors the execution of the application instances 135 over a period of time to collect the historical performance data 140. Generally, the historical performance data 140 comprises data relating to the performance of the application instances 135, such as how many incoming requests a particular application instance processes on average during a unit of time (*e.g.*, processed requests per minute) as well as the latencies of those requests (*e.g.*, how long, on average, a single request takes to process). Additionally, the predictive scaling component 125 monitors the workload of the distributed application over a period of time. For example, the predictive scaling component 125 could monitor a number of incoming requests from the client devices 110 to collect historical workload data for the distributed application. The predictive scaling component 125 then performs a statistical analysis of the workload data (and potentially the historical performance data 140 as well) in order to determine one or more scaling patterns.

[0016] Generally, the scaling patterns represent a pattern or trend in the incoming workload of the application that is indicative of a future workload of the application. Such a statistical analysis can also consider factors such as the time of day, the day of the week, a related application's workload, and so on in determining the scaling patterns. For example, the predictive scaling component 125 could determine that the rate of increase of the incoming workload during the hours leading up to the hours of peak workload is indicative of the amount of peak workload for a given day. For example, the predictive scaling component 125 could determine that a relatively large rate of increase in the application's workload during the hours leading up to the time of historically peak workload (e.g., evening hours when the demand for video streaming services are greatest), indicates that the peak workload for that particular day will be relatively high. As another example, if the predictive scaling component 125 determines that the rate of increase in the application's workload is relatively low, the predictive scaling component 125 could determine that the peak workload will also be relatively low for that particular day. Of course, these examples of scaling patterns are provided without limitation and for illustrative purposes only, and more generally any pattern determined from statistically analyzing the collected workload data can be used, consistent with the functionality described herein.

[0017] Upon determining the scaling patterns, the predictive scaling component 125 monitors the application's current workload to determine when the current workload matches one of the determined scaling patterns. Continuing the above example, the predictive scaling component 125 could determine that, for a given day, the application's rate of increase in workload is

relatively high during the hours leading up to the time period of historically peak workload. The predictive scaling component 125 then determines an anticipated future workload of the application, based on the scaling pattern and the application's current workload. In one embodiment, the predictive scaling component 125 is configured to determine the application's future workload, at a future point in time based on the application's start-up time. For example, if the application's start-up time is approximately 1 hour, the predictive scaling component 125 could determine the application's future workload 1 hour from the current time. In one embodiment not encompassed by the claims, the future point in time is determine based on the application's average start-up time plus some predefined period of time (*e.g.*, 10 minutes), to help ensure the application instances will be started in advance of the future workload.

[0018] Once the application's future workload is estimated, the predictive scaling component 125 then determines a number of application instances needed to satisfy the estimated future workload. Generally, if the future workload is greater than the application's current workload, a number of additional applications may need to be instantiated in order to accommodate the increased workload. On the other hand, if the estimated future workload is less than the current workload, some number of application instances could be shutdown in order to avoid idle or underutilized computing resources.

[0019] The predictive scaling component 125 then determines a plan for predictively scaling the application, based on the determined number of application instances needed to satisfy the estimated future workload and an average start-up time for the application instances. For example, the predictive scaling component 125 could determine that 10 additional application instances will be needed to accommodate the estimated future workload 1 hour from now, and could further determine that 20 additional application instances will be needed to accommodate the estimated future workload 2 hours from now. If the predictive scaling component 125 then determines that the application instances have an average start-up time of 45 minutes, the predictive scaling component 125 could determine a plan to instantiate 10 additional application instances 15 minutes from now *(i.e.,* to satisfy the estimated workload 1 hour into the future), and to instantiate 10 additional application instances 75 minutes from now *(i.e.,* to satisfy the estimated workload 2 hours into the future). Advantageously, doing so allows the predictive scaling component 125 to predictively scale the distributed cloud application based on the application's estimated future workload, allowing the application to more smoothly handle fluctuations in workload.

[0020] On the other hand, if the predictive scaling component 125 determines that the future workload is less than the current workload and that 10 fewer application instances will be needed to process the future workload, the predictive scaling component 125 then determine a plan for shutting down particular application instances over a window of time. In doing so, the predictive scaling component 125 could consider an average shutdown time of an application instance. By reducing the number of application instances in response to a predicted decrease in future workload, the predictive scaling component 125 helps to avoid underutilized application instances, thereby reducing the cost of running the application. Moreover, doing so also frees up the computing resources associated with the shutdown application instances,

thereby freeing these resources up for other purposes as needed.

[0021] Generally, the predictive scaling component 125 may be combined with other scaling and load management techniques as well, in order to ensure the system health, availability and optimization of the application. As an example, in one embodiment, the predictive scaling component 125 is configured to operate in conjunction with a reactive scaling component that is configured to add or remove application instances in response to changes in workload (*e.g.*, incoming request volume). For instance, the predictive scaling component 125 could be configured to provide a minimum number of active application instances based on an estimated future workload of the application, while the reactive scaling component could instantiate additional application instances in response to an unanticipated workload. Here, while traditional reactive scaling techniques could reduce the number of active application instances in response to a temporary drop in request volume, doing so could lead to a failure of the service when the normal request volume for the application resumes. In such a scenario, the predictive scaling component 125 could prevent the number of application instances from falling below a threshold number of application instances, based on the application's predicted workload at a future point in time, thereby avoiding the failure of the service when the normal request volume resumes. Advantageously, combining the predictive scaling component 125 with other scaling techniques, such as a reactive scaling component, may improve the performance of the system, allowing the reactive scaling component to respond to fluctuations in workload while ensuring a minimum number of application instances remain instantiated based on a predicted future workload of the application.

[0022] Of note, while the above example involves the predictive scaling component 125 working in conjunction with a reactive scaling component, such an example is provided without limitation and for illustrative purposes only. More generally, it is broadly contemplated that the predictive scaling component 125 may work in conjunction with a variety of other scaling and more general workload management techniques, in order to improve the performance and efficiency of the system. For example, the predictive scaling component 125 could work in conjunction with a hardware scaling component configured to scale the resources *(e.g.,* memory, processors, *etc.)* executing the application instances, based on the application's workload. As another sample, the predictive scaling component 125 could work in conjunction with a component configured to scale a downstream resource (*e.g.*, a number of database instances, hardware resources on which the database instances are executing, *etc.),* in response to the application's current workload and/or predictive future workload.

[0023] Figure 2 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment not encompassed by the claims described herein. As shown, the cloud computing environment 200 includes the predictive scaling component 125, unallocated compute resources 210, and a plurality of compute nodes 215. Here, each of the compute nodes 215 is configured with one or more application instances 135 and one or more monitoring components 225. The monitoring components 225 could monitor their respective application instance 135 in order to collect performance and workload data. The monitoring components 225 could then transmit the collected data to the predictive scaling

component 125, and the predictive scaling component 125 could use such data to predictively scale the number of application instances 135 and the number of compute nodes 215.

[0024] For example, the predictive scaling component 125 could determine that the present workload of the application instances 135 matches a particular scaling pattern, indicating that the application's workload will substantially increase within some period of time. In response to such a determination, the predictive scaling component 125 could allocate a number of additional compute nodes 215 using the unallocated compute resources 210, and could initialize a number of new application instances 135 on these additional compute nodes. Generally, it is preferable for the predictive scaling component 125 to determine the application's future workload at some point in the future that is at least as long as the application instances' average start-up time, in order to assure that the application instances can be fully operational and ready to process the future workload at the future point in time. For example, if the application instances take an average of 60 minutes to fully initialize, the predictive scaling component 125 may be configured to predict the future workload 75 minutes into the future and to create a number of additional application instances sufficient to process this estimated future workload, so that the application instances will be initialized and operational in advance of the future workload.

[0025] In one embodiment not encompassed by the claims, the predictive scaling component 125 is configured to monitor the growth in application traffic over time and to adjust its application scaling accordingly. For exmple, a video streaming service could continue to increase its subscriber base over time, and the number of subscribers using the video streaming service may increase month-over-month. As a result, the number of application instances needed to process the incoming workload at a given point in time during the week may continue to increase proportionately to the change in the subscription base. For example, if the subscriber base increases 20% within a one year period, the incoming workload on an average Tuesday evening at 5:00pm may be approximately 20% higher today than it was a year ago. As such, the predictive scaling component 125 could be configured to take the growth (or decrease) in users into account when generating a plan for scaling the application instances. Of course, such an example is provided without limitation and for illustrative purposes only.

[0026] Moreover, while the predictive scaling component 125 may alter the scaling plan based on the growth in users, such growth may not always relate in a proportionate manner. For example, while the growth in subscriptions in a one year period may increase slightly, the average workload at a particular day and time may increase substantially over the same one year period due to other factors. For instance, as more content is added to the video streaming service over the course of the year, the viewing patterns of the subscribers may also change as a result. As a result, the predictive scaling component 125 may consider the growth in subscribers as one of many factors in predicting the future workload of the application and in determining the plan for scaling the number of application instances accordingly.

[0027] Additionally, the predictive scaling component 125 may consider known events in

predicting the application's future workload. Examples of such events include daylight savings time, the availability of particular content (e.g., a live stream of a particular sporting event), holidays, and so on. Such events may be specified to the predictive scaling component 125 programmatically (e.g., events that happen each year such as holidays, daylight savings time, etc.) or could be manually specified by a user (e.g., a live stream of a particular event). More generally, however, the predictive scaling component 125 may consider any event having any correlation to the workload of the application. Moreover, different events may have a different correlation to the application, depending on the type of the application. For example, if the application is a video streaming service, a holiday could indicate an increase in the application's workload as users are home from work. On the other hand, if the application is a personnel application for use by a business, the holiday could indicate a decrease in the application's workload as the users are out of the office on the specified day. Advantageously, by considering such events, the predictive scaling component 125 can more accurately and reliability predict the future workload of the application and scale the application instances accordingly.

[0028] Figure 3 is a flow diagram illustrating a method for determining performance patterns of a cloud application, according to one embodiment described herein. As shown, the method 300 begins at block 310, where the predictive scaling component 125 monitors the performance and workload of an application instance(s) in order to collect historical performance data. Such data may include, without limitation, a number of incoming requests per unit of time, a number of requests processed by a given application instance per unit of time, a start-up time of a new application instance, a shutdown time of an application instance, and so on. Additionally, the predictive scaling component 125 monitors the performance of the cloud computing environment with respect to the application (block 315). For example, in addition to the start-up time of the application instances, there may be an additional start-up time of configuring and initializing a new cloud compute node on which to run the new application instance.

[0029] The predictive scaling component 125 then analyzes the collected performance data in order to determine scaling patterns for the application (block 320), and the method 300 ends. For example, the predictive scaling component 125 could determine that the historical workload data indicates that the application's workload increases dramatically leading up to 5:00pm each day, and that the amount of peak workload is proportional to the rate of increase around this time. The predictive scaling component 125 could then use such a scaling pattern in order to predictively scale a number of application instances, in order to accommodate the predicted future workload.

[0030] In one embodiment not encompassed by the claims, the predictive scaling component 125 is configured to operate in a feedback loop, where data is continually fed back to the predictive scaling component 125 in order to improve future scaling operations. Such data may include, for example, data describing a workload of the application, data describing one or more scaling patterns for the application, data describing an impact of scaling operations on the system as a whole, and so on. Advantageously, by feeding such data back into the

predictive scaling component 125, embodiments not encompassed by the claims can more accurately predict when future scaling operations are needed and can optimize such future scaling operations as well. For example, such data could be used to determine that a particular scaling plan in which a substantial number of additional application instances were provisioned caused unintended side effects, e.g., placing sudden pressure on one or more downstream components. In response to such a determination, the predictive scaling component 125 could learn from the historical scaling operations and related data, and the predictive scaling component 125 could optimize future scaling operations (*e.g.*, by restricting the number of additional application instances instantiated, by evenly distributing optimal sized scaling operations over a period of time, *etc.)* to minimize unintended side effects in the future.

[0031] Figure 4 is a flow diagram illustrating a method for predictively scaling instances of a cloud application, according to one embodiment described herein. As shown, the method 400 begins at block 410, where the predictive scaling component 125 provides scaling pattern data for an application hosted in a cloud computing environment. Such pattern data could be determined, for example, using the method 300 shown in Figure 3 and discussed above. The predictive scaling component 125 then monitors the application executing in the cloud environment (block 414), in order to detect when one of the scaling patterns is satisfied (block 420). Here, if the application's current performance and workload do not match any of the scaling patterns, the method 400 returns to block 414, where the predictive scaling component 125 continues monitoring the application.

[0032] If instead the predictive scaling component 125 detects that the application's current performance and workload matches one of the scaling patterns, the method 400 proceeds to block 425, where the predictive scaling component 125 determines a predicted future workload of the application based on the scaling pattern. For instance, continuing the example given above where the historical workload data indicates that the application's workload increases dramatically leading up to 5:00pm each day, and that the amount of peak workload is proportional to the rate of increase around this time, the predictive scaling component 125 could determine an estimated future workload of the application, based on the application's current rate of increase in workload leading up to 5:00pm. As discussed above, the predictive scaling component 125 may be configured to estimate the future workload at some point in the future at least as long as the application instance's average start-up time, so as to ensure that the application instances can be fully initialized in advance of the future workload.

[0033] Once the predictive scaling component 125 has determined the estimated future workload, the predictive scaling component 125 determines a number of additional application instances to create based on the estimated future workload (block 430). Of note, this example assumes that the estimated future workload is greater than the application's current workload. However, in other examples where the future workload is less than the application's current workload, the predictive scaling component 125 could determine a number of application instances that are no longer needed to satisfy the estimated future workload.

[0034] Returning to the present example, the predictive scaling component 125 also

determines a timeline for creating the additional application instances (block 435). Here, the predictive scaling component 125 may be configured to consider the average start-up of an application instance in creating the timeline, in order to ensure that the additional application instances will be fully initialized in anticipation of the increased future workload. The predictive scaling component 125 then instantiates the additional application instances based on the determined timeline (block 440), and the method 400 ends.

[0035] Figure 5 illustrates a computing infrastructure configured with a predictive scaling component, according to one embodiment described herein. As shown, the environment 700 includes a predictive scaling system 705 which includes, without limitation, a central processing unit (CPU) 702, a network interface 708, an interconnect 710, and a system memory 712. The CPU 702 retrieves and executes programming instructions stored in the system memory 712. Similarly, the CPU 702 stores and retrieves application data residing in the system memory 712. The interconnect 710 facilitates transmission, such as of programming instructions and application data, between the CPU 702, input/output (I/O) devices interface 706, storage 704, network interface 708, and system memory 712. The I/O devices interface 706 is configured to receive input data from user I/O devices 722. Examples of user I/O devices 722 may include one of more buttons, a keyboard, and a mouse or other pointing device. The I/O device interface 706 may also include an audio output unit configured to generate an electrical audio output signal, and user I/O devices may further includes a speaker configured to generate an acoustic output in response to the electrical audio output signal. Another example of an I/O device is a display device that generally represents any technically feasible means for generating an image for display. For example, the display device may be a liquid crystal display (LCD) display, CRT display, or DLP display. The display device may be a TV that includes a broadcast or cable tuner for receiving digital or analog television signals.

[0036] CPU 702 is included to be representative of a single CPU, multiple CPUs, a single CPU having multiple processing cores, and the like. And the system memory 712 is generally included to be representative of a random access memory. The storage 704 may be a disk drive storage device. Although shown as a single unit, the storage 704 may be a combination of fixed and/or removable storage devices, such as fixed disc drives, floppy disc drives, tape drives, removable memory cards, or optical storage, network attached storage (NAS), or a storage area-network (SAN). The network interface 708 is configured to transmit data via the communications network, *e.g.*, to transmit context tokens and localized digital assets from client devices as well as to return a assembled variation of digital content generated by the dynamic context-based assembler to the client devices.

[0037] The system memory 712 stores a predictive scaling component 125 that is configured to monitor an application's performance and workload over time, for use in predictively scaling a number of instances of the application in anticipation of future workloads. As discussed above, the predictive scaling component 125 monitors performance of an application within a cloud computing environment over a first window of time to collect historical performance data, where the application comprises a plurality of application instances. Additionally, the predictive scaling component 125 monitors a workload of the application over a second window of time to

collect historical workload data. The predictive scaling component 125 could then analyze both the historical performance data and the historical workload data to determine one or more scaling patterns for the application. Upon determining a present state of the application matches one of the one or more scaling patterns, the predictive scaling component 125 determines a plan for predictively scaling the application, and uses the plan to predictively scale the plurality of application instances.

[0038] In the preceding, reference is made to various embodiments. However, it should be understood that the present disclosure is not limited to specific described embodiments. Instead, any combination of the preceding features and elements, whether related to different embodiments or not, is contemplated to implement and practice the functionality described herein. Furthermore, although particular embodiments may achieve advantages over other possible solutions and/or over the prior art, whether or not a particular advantage is achieved by a given embodiment is not limiting of the present disclosure. Thus, the preceding aspects, features, embodiments and advantages are merely illustrative and are not considered elements or limitations of the appended claims except where explicitly recited in a claim(s). Likewise, any reference to "the invention" shall not be construed as a generalization of any inventive subject matter disclosed herein and shall not be considered to be an element or limitation of the appended claims except where explicitly recited in a claim(s).

[0039] While the foregoing is directed to various embodiments, other and further embodiments may be devised without departing from the basic scope thereof. For example, embodiments may be implemented in hardware or software or in a combination of hardware and software. One embodiment 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 described herein, are embodiments of the present disclosure.

[0040] Therefore, the scope of the present disclosure is determined by the claims that follow.

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all

liability in this regard.

Patent documents cited in the description

- WO2013006708A [0003]
- US2011258621A [0003]
- US2013174146A [0003]
- US20110295999A [0003]
- US20120259976A [0003]

Patentkrav

1. Fremgangsmåde, der udføres af en forudsigende skaleringskomponent (125), der omfatter:

overvågning af ydeevnen af et program (135) i et skybaseret computermiljø (120) i løbet af et første tidsrum for at indsamle historiske data om ydeevne (140), hvor programmet (135) omfatter en flerhed af programforekomster, og hvor historiske ydeevnedata (140) indbefatter starttidspunktsdata og sluttidspunktsdata for flerheden af programforekomster,

overvågning af en arbejdsbyrde for programmet (135) i løbet af et andet tidsrum for at indsamle historiske arbejdsbyrdedata, hvor de historiske arbejdsbyrdedata indbefatter mindst et vist antal indkommende anmodninger pr. tidsenhed,

analyse af de historiske arbejdsbyrdedata for at bestemme ét eller flere skaleringsmønstre for programmet (135), hvor skaleringsmønstrene repræsenterer en tendens i en indkommende arbejdsbyrde for programmet (135), som er tegn på en fremtidig arbejdsbyrde for programmet (135),

efter bestemmelse af, at en nuværende tilstand af programmet (135) passer til ét eller flere skaleringsmønstre, hvor den nuværende tilstand af programmet (135) indbefatter en nuværende arbejdsbyrde for programmet (135), bestemmelse af en anslået fremtidig arbejdsbyrde for programmet (135) på et eller andet fremtidigt tidspunkt baseret på det skaleringsmønster, som passer til den nuværende tilstand, hvor tidspunktet i fremtiden bestemmes på basis af starttidspunktsdataene,

bestemmelse af en plan for forudsigende skalering af programmet (135) på basis af den anslåede fremtidig arbejdsbyrde, og

forudsigende skalering af flerheden af programforekomster på basis af den fastlagte plan og på, om:

starttidspunktsdataene, når det bestemmes, at der er brug for yderligere programforekomster, eller

sluttidspunktsdataene, når det bestemmes, at programforekomster skal lukkes ned.

2. Fremgangsmåden ifølge krav 1, der yderligere omfatter:
når det bestemmes, at der er brug for yderligere programforekomster:
bestemmelse af en første tidslinje til til instantiering af et antal yderligere programforekomster på basis af starttidspunktsdataene,
hvor forudsigende skalering af flerheden af programforekomster yderligere er baseret på den fastlagte første tidslinje, og yderligere omfatter:
instantiering af antallet af yderligere programforekomster i henhold til den fastlagt første tidslinje.

3. Fremgangsmåden ifølge krav 1 eller krav 2, der yderligere omfatter:
når det bestemmes, at programforekomster skal lukkes ned:
bestemmelse af en anden tidslinje til til nedlukning af et antal programforekomster på basis af sluttidspunktsdataene,
hvor forudsigende skalering af flerheden af programforekomster yderligere er baseret på den fastlagte anden tidslinje, og yderligere omfatter:
nedlukning af antallet af programforekomster i henhold til den fastlagte anden tidslinje.

4. Fremgangsmåden ifølge et hvilket som helst af de foregående krav, hvor de historiske ydeevnedata (140) yderligere indbefatter mindst et vist antal anmodninger, der er behandlet af en programforekomst pr. en anden tidsenhed.

5. Fremgangsmåden ifølge et hvilket som helst af de foregående krav, hvor den eller flere skaleringsmønstre også bestemmes på basis af mindst én af en dage i ugen og et tidspunkt på dagen.

6. Fremgangsmåden ifølge et hvilket som helst af de foregående krav, hvor den nuværende tilstand af programmet (135) yderligere omfatter mindst én af en nuværende dag i ugen, en nuværende tidsværdi, en nuværende ændringshastighed i den nuværende arbejdsbyrde programmet (135) og et nuværende

antal programforekomster i flerheden af programforekomster.

7. Fremgangsmåden ifølge et hvilket som helst af de foregående krav, hvor forudsigende skalering af flerheden af programforekomster samarbejder med mindst én af en anden programskaleringsteknik og en algoritme til udligning af arbejdsbyrden.

8. Fremgangsmåden ifølge krav 7, hvor forudsigende skalering af flerheden af programforekomster på basis af den fastlagte plan yderligere omfatter:
bestemmelse af en grænseværdi for antallet af programforekomster, der skal vedligeholdes, på basis af den fastlagte plan, og
reaktiv skalering af flerheden af programforekomster som reaktion på en nuværende arbejdsbyrde for programmet (135) uden skalering af flerheden af programforekomster under grænseværdien for antallet af programforekomster.

9. System (705), der omfatter:
én eller flere processorer (702), og
en hukommelse (712), der indeholder én eller flere programmer, der, når de udføres af én eller flere processorer, får den ene eller flere processorer til at udføre fremgangsmåden ifølge et hvilket som helst af de foregående krav.

10. Computerprogramprodukt, der omfatter instruktioner, som, når de udføres på et processor, får processoren til at udføre fremgangsmåden ifølge et hvilket som helst af kravene 1 til 8.

DRAWINGS

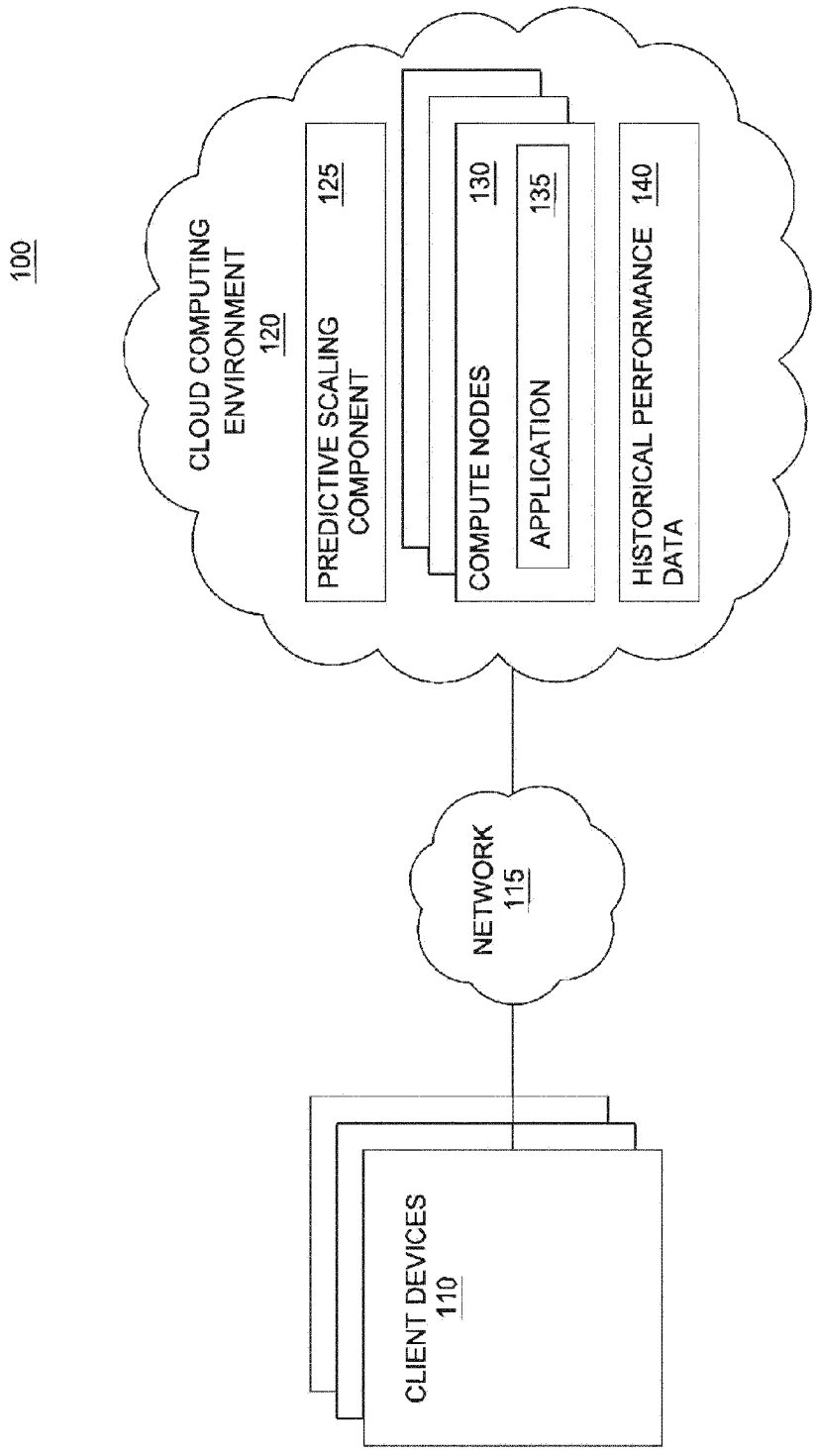


FIG. 1

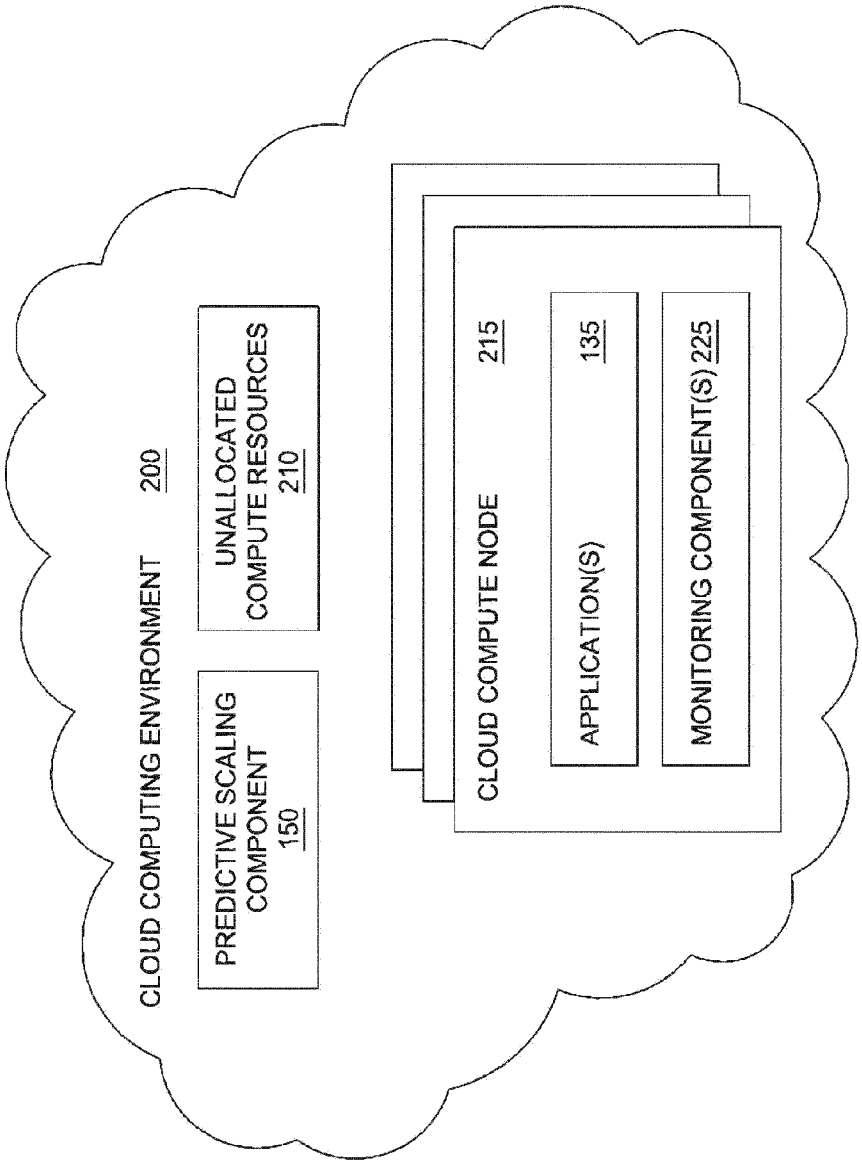

CLOUD COMPUTING ENVIRONMENT 200
PREDICTIVE SCALING COMPONENT 150
UNALLOCATED COMPUTE RESOURCES 210
CLOUD COMPUTE NODE 215
APPLICATION(S) 135
MONITORING COMPONENT(S) 225

FIG. 2

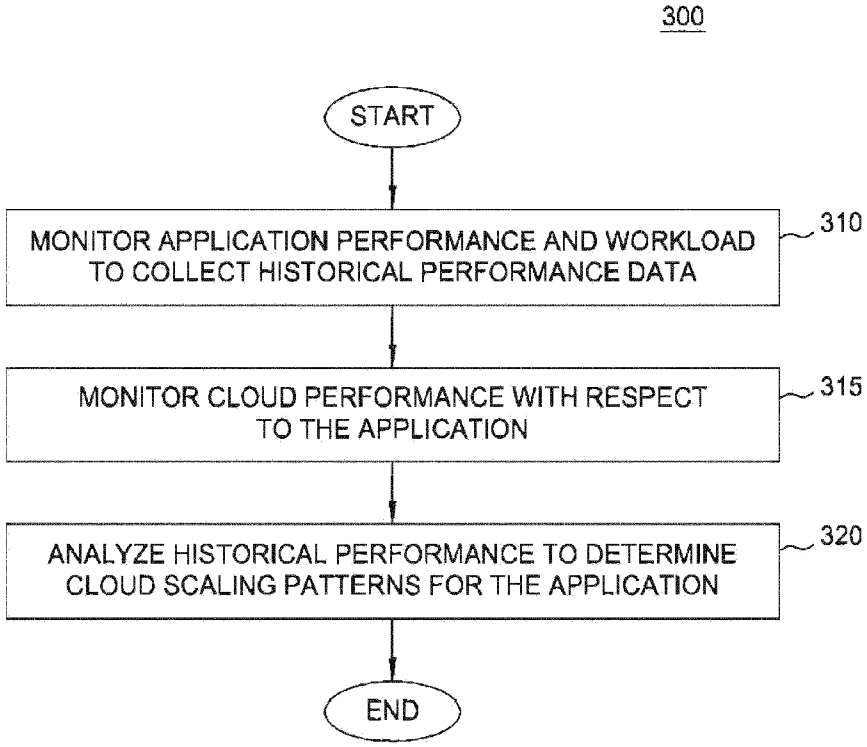

300
START
MONITOR APPLICATION PERFORMANCE AND WORKLOAD TO COLLECT HISTORICAL PERFORMANCE DATA
310
MONITOR CLOUD PERFORMANCE WITH RESPECT TO THE APPLICATION
315
ANALYZE HISTORICAL PERFORMANCE TO DETERMINE CLOUD SCALING PATTERNS FOR THE APPLICATION
320
END

FIG. 3

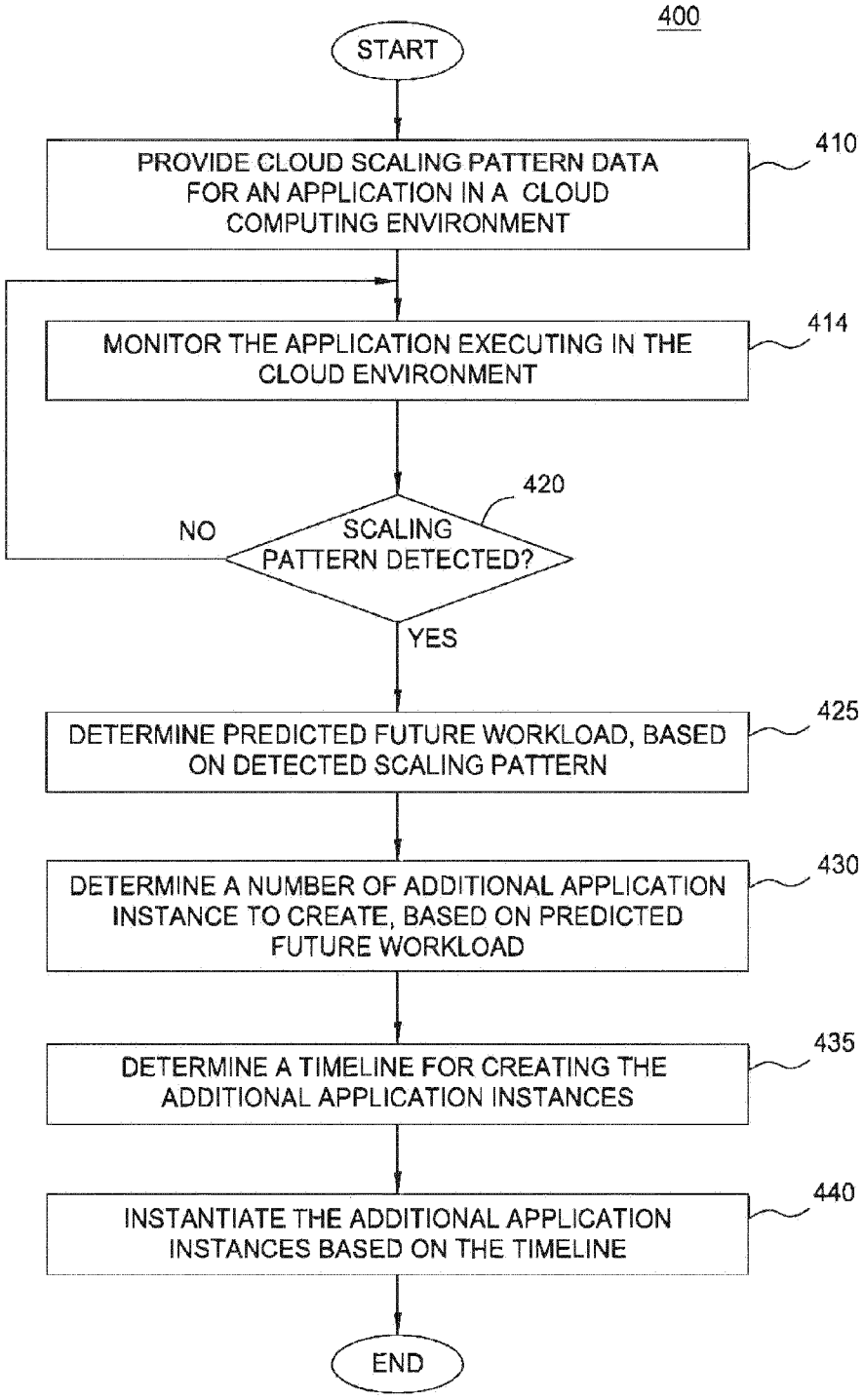

400
START
PROVIDE CLOUD SCALING PATTERN DATA FOR AN APPLICATION IN A CLOUD COMPUTING ENVIRONMENT
410
MONITOR THE APPLICATION EXECUTING IN THE CLOUD ENVIRONMENT
414
420
SCALING PATTERN DETECTED?
NO
YES
DETERMINE PREDICTED FUTURE WORKLOAD, BASED ON DETECTED SCALING PATTERN
425
DETERMINE A NUMBER OF ADDITIONAL APPLICATION INSTANCE TO CREATE, BASED ON PREDICTED FUTURE WORKLOAD
430
DETERMINE A TIMELINE FOR CREATING THE ADDITIONAL APPLICATION INSTANCES
435
INSTANTIATE THE ADDITIONAL APPLICATION INSTANCES BASED ON THE TIMELINE
440
END

FIG. 4

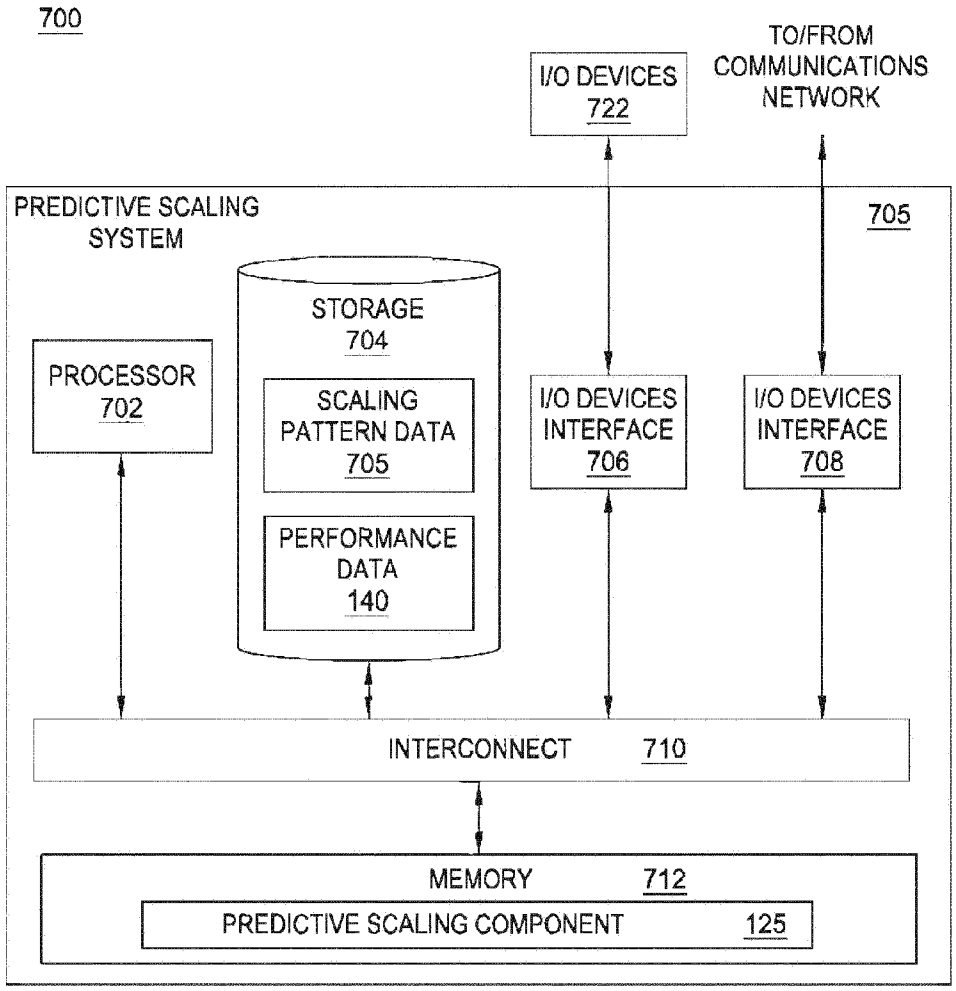

700
I/O DEVICES
722
TO/FROM
COMMUNICATIONS
NETWORK
PREDICTIVE SCALING
SYSTEM
705
STORAGE
704
PROCESSOR
702
SCALING
PATTERN DATA
705
PERFORMANCE
DATA
140
I/O DEVICES
INTERFACE
706
I/O DEVICES
INTERFACE
708
INTERCONNECT
710
MEMORY
712
PREDICTIVE SCALING COMPONENT
125

FIG. 5