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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 June 2014 (26.06.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/098790 A1

(51) International Patent Classification:
G06F 9/46 (2006.01)

(21) International Application Number:
PCT/US2012/070063

(22) International Filing Date:
17 December 2012 (17.12.2012)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **EMPIRE TECHNOLOGY DEVELOPMENT LLC** [US/US]; 2711 Centerville Road, Suite 400, Wilmington, DE 19808 (US).

(72) Inventor: **KURABAYASHI, Shuichi**; B-103, 2-3-5, Engyo, Fujisawa-shi, Kanagawa, 252-0805 (JP).

(74) Agent: **LEE, David, S.**; Brundidge & Stanger, P.C., 2318 Mill Road, Suite 1020, Alexandria, VA 22314 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

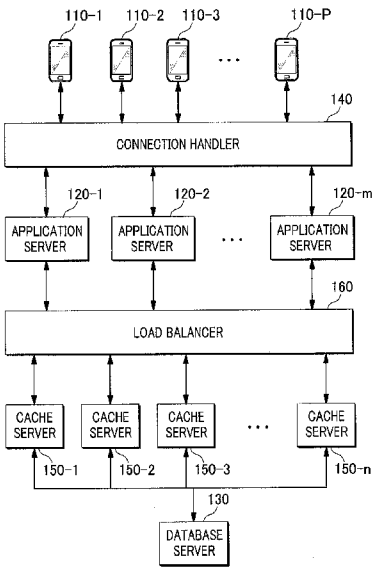
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU,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

Published:

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

(54) Title: LOAD BALANCING SCHEME

FIG. 1



(57) Abstract: Technologies are generally described for load balancing scheme in a cloud computing environment hosting a mobile device. In some examples, a load balancer may include multiple request processing units, each of the multiple request processing units comprising a network socket that is connected to at least one application server and at least one cache server and a programmable processor configured to process a cache request from one of the at least one application server, a performance checking unit configured to measure processing loads of the programmable processors, and a processor managing unit configured to adjust the processing loads by writing or deleting a load balancing program in at least one of the programmable processors.

WO 2014/098790 A1

LOAD BALANCING SCHEME

BACKGROUND

[0001] Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. As applications by which a large number of users share data of a relatively large size on a cloud datacenter increase, a load balancer may be required to efficiently process data traffic between computation nodes and storage nodes.

SUMMARY

[0002] In an example, a load balancer may include multiple request processing units, each comprising a network socket that is connected to at least one application server and at least one cache server, and a programmable processor configured to process a cache request from one of the at least one application server. The load balancer further comprises a performance checking unit configured to measure processing loads of the programmable processors and a processor managing unit configured to adjust the processing loads by writing or deleting a load balancing program in at least one of the programmable processors.

[0003] In another example, a method performed under control of a load balancer including multiple programmable processors may include receiving a cache request from one of at least one application server; writing, into one of the plurality of programmable processors, a load balancing program that handles the cache request by designating one of at least one cache server; and transmitting, to the one of the at least one application server, an address of the one of the at least one cache server corresponding to the cache request.

[0004] In yet another example, a computer-readable storage medium may store thereon computer-executable instructions that, in response to execution, cause a load balancer including multiple programmable processors to perform operations including receiving a cache request from one of at least one application server; writing, into one of the plurality of programmable processors, a load

balancing program that handles the cache request by designating one of at least one cache server; and transmitting, to the one of the at least one application, server an address of the one of the at least one cache server corresponding to the cache request.

[0005] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE FIGURES

[0006] The foregoing and other features of this disclosure will become more apparent from the following description and appended claims, taken in conjunction with the accompanying drawings. Understanding that these drawings depict only several embodiments in accordance with the disclosure and are, therefore, not to be considered limiting of its scope, the disclosure will be described with additional specificity and detail through use of the accompanying drawings, in which:

[0007] **Fig. 1** schematically shows an illustrative example of an environment in which a load balancer provides a load balancing scheme between multiple application servers and multiple cache servers, arranged in accordance with at least some embodiments described herein;

[0008] **Fig. 2** shows a schematic block diagram of an illustrative example of a load balancer, arranged in accordance with at least some embodiments described herein;

[0009] **Fig. 3** shows an example flow diagram of a process for providing a load balancing scheme, arranged in accordance with at least some embodiments described herein;

[0010] **Fig. 4** shows an example flow diagram of a process for writing a load balancing program into a programmable processor, arranged in accordance with at least some embodiments described herein;

[0011] **Fig. 5** illustrates computer program products that may be utilized to provide a load balancing scheme, arranged in accordance with at least some embodiments described herein; and

[0012] **Fig. 6** is a block diagram illustrating an example computing device that may be utilized to provide a load balancing scheme, arranged in accordance with at least some embodiments described herein.

DETAILED DESCRIPTION

[0013] In the following detailed description, reference is made to the accompanying drawings, which form a part hereof. In the drawings, similar symbols typically identify similar components, unless context dictates otherwise. The illustrative embodiments described in the detailed description, drawings, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing from the spirit or scope of the subject matter presented herein. It will be readily understood that the aspects of the present disclosure, as generally described herein, and illustrated in the drawings, can be arranged, substituted, combined, separated, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

[0014] This disclosure is generally drawn, *inter alia*, to methods, apparatuses, systems, devices, and computer program products related to a load balancing scheme. Further, technologies are generally described for a load balancer configured to provide load balancing between multiple application servers and multiple cache servers.

[0015] In some embodiments, a cloud-based network system may include at least three server layers including, for example, multiple application servers configured to handle requests from user devices, multiple cache servers configured to handle data caching for the multiple application servers, and a cloud database server configured to store and provide data for use by the multiple application servers. In such cases, a load balancer may be configured to balance loads between the multiple application servers and the multiple cache servers to resolve a communication bottleneck between the multiple application servers and the multiple cache servers.

[0016] The load balancer may employ thereon multiple programmable processors, which are connected to respective network sockets. The load balancer may dynamically write a load balancing program for performing load balancing into one of the programmable processors. In some embodiments, an application

may be executed through one of the multiple application servers, and the one of the multiple application servers may request of the load balancer data regarding the running of the application. In such cases, the load balancer may write into one of the programmable processors a load balancing program, which is an application-specific program according to one or more characteristics of the application. The one or more characteristics of the application may be decided by the type of the application (e.g., whether the application is an e-mail application, a game application, an SNS service providing application, an e-book application, a map application, a video playing application, etc.). By way of example, but not limitation, video playing applications or game applications may be characterized that they require more data transmission than e-mail applications and SNS service providing applications. The load balancing program may be required to handle the application considering such characteristics.

[0017] By way of example, but not limitation, when the user device runs an application for playing a video that is stored in the cloud database server, a corresponding application server may request of the load balancer the video that is stored in the cloud database server. Then, the load balancer may write into one of the multiple programmable processors a load balancing program of a type adapted to cache the video in one of the multiple cache servers thus making it possible for the user device to play the video smoothly. In some other examples, when the user device run an application for a social networking service (or "SNS") via a corresponding application server, the load balancer may proactively write into one of the multiple programmable processors a load balancing program of a type that is adapted to preferentially store information data viewed by a greater number of users in one the multiple cache servers. By way of such customized load balancing, it may be possible to efficiently manage data caching according to one or more characteristics of the application.

[0018] In some embodiments, the load balancer may measure a processing load of each of the multiple programmable processors. When the processing load is concentrated on a specific programmable processor, the load balancer may increase the number of programmable processors for the processing load by writing a corresponding load balancing program into one or more of the multiple programmable processors. Alternatively, when the programmable processor for the processing load exhibits low load balancing cost, that is, when a surplus

calculation power of the programmable processor is great, the load balancer may reduce the number of programmable processors for the processing load. In this way, the number of programmable processors allocated to a specific processing load may be automatically adjusted.

[0019] **Fig. 1** schematically shows an illustrative example of an environment in which a load balancer provides a load balancing scheme between multiple application servers and multiple cache servers, arranged in accordance with at least some embodiments described herein. As depicted in Fig. 1, one of user devices 110-1, 110-2, 110-3, ..., 110-p may send to one of application servers 120-1, 120-2, ..., 120-m a request for application-related data, which may be stored in a database server 130, via a connection handler 140. Further, one of cache servers 150-1, 150-2, ..., 150-n may cache the application-related data from database server 130, and a load balancer 160 may process the request for application-related data. By way of example, but not limitation, the request may include an HTTP (Hypertext Transfer Protocol) request such as, for example, an HTTP GET request or an HTTP POST request.

[0020] In some embodiments, connection handler 140 may allocate or assign the request from the one of user devices 110-1, 110-2, 110-3, ..., 110-p to the appropriate one of application servers 120-1, 120-2, ..., 120-m. In some embodiments, connection handler 140 may allocate or assign the request based at least in part on processing loads of application servers 120-1, 120-2, ..., 120-m.

[0021] In some embodiments, database server 130 may store data for various applications, and cache servers 150-1, 150-2, ..., 150-n may cache at least some of the data stored in database server 130. By way of example, but not limitation, database server 130 may be a cloud datacenter, and cache servers 150-1, 150-2, ..., 150-n may be in-memory cache servers.

[0022] In some embodiments, a load balancer 160 may employ thereon multiple programmable processors (not shown in Fig. 1), which may be respectively connected to network sockets (not shown in Fig. 1). Load balancer 160 may dynamically write a load balancing program for performing load balancing into one of the multiple programmable processors. In such cases, the load balancing program may be an application-specific program corresponding to one or more characteristics of the application. The load balancing program may be a program that determines how to store data in cache servers 150-1, 150-2, ...,

150-n. Since a single cache server store data beyond the memory storage capacity, multiple cache servers 150-1, 150-2, ..., 150-n are employed. In order to improve responsiveness to the application, the data likely to be used for cache servers 150-1, 150-2, ...150-n may be selectively maintained in the cache memory. However, because it may be difficult to predict how long certain data will go unused by an application, load balancer 160 may perform the load balancing using the load balancing program that reflects the one or more characteristics of the application. By way of example, but not limitation, the one or more characteristics may include data access tendency of the application, which may refer to how often the application requests data stored in database server 130, or what kind of data (e.g., a document, an audio, a video, an image, etc.) the application requests from database server 130.

[0023] In some embodiments, load balancer 160 may automatically measure a load of each of the multiple programmable processors. When a load of a specific programmable processor (which is processing a request from an application) exceeds a threshold value (e.g., when the load balancing cost exceeds I/O cost and/or application cost), the number of programmable processors assigned to the request from the application may be increased. But when the programmable processor processing the request from the application shows a low load balancing cost (i.e., the surplus calculation power of the programmable processors is great), the number of programmable processors allocated to the load balancing process of the application may be decreased. In this way, the number of programmable processors allocated to each application can be adjusted automatically.

[0024] **Fig. 2** shows a schematic block diagram of an illustrative example of load balancer 160, arranged in accordance with at least some embodiments described herein. As depicted, load balancer 160 may include multiple request processing units 210-1, 210-2, ..., 210-r, a memory 220, a database 230, a processor managing unit 240, a performance checking unit 250 and a compiler 260. Although illustrated as discrete components, various components may be divided into additional components, combined into fewer components, or eliminated altogether while being contemplated within the scope of the disclosed subject matter.

[0025] Each of request processing units 210-1, 210-2, ..., 210-r may include a programmable processor 270-1, 270-2, ..., or 270-r and a network socket 280-1, 280-2, ..., or 280-r. Each of programmable processors 270-1, 270-2, ..., 270-r may be configured to process a cache request from one of application servers 120-1, 120-2, ..., 120-m. In some embodiments, each of programmable processors 270-1, 270-2, ..., 270-4 may analyze the cache request as to its characteristics and, as will be described below in more details, provide the analysis results to processor managing unit 240 so that processor managing unit 240 may select a load balancing program that is adapted to process the cache request according to the characteristics. By way of example, but not limitation, programmable processors 270-1, 270-2, ..., 270-r may be an FPGA (Field Programmable Gate Array). Each of network sockets 280-1, 280-2, ..., 280-r may be connected to one or more of application servers 120-1, 120-2, ..., 120-m and one or more of cache servers 150-1, 150-2, ..., 150-n.

[0026] Memory 220 may store multiple load balancing programs for processing a cache request from one or more of application servers 120-1, 120-2, ..., 120-m. By way of example, but not limitation, the multiple load balancing programs may be application-specific programs according to one or more characteristics of applications. Each of the load balancing programs determines how to store data in one or more of cache servers 150-1, 150-2, ..., 150-n.

[0027] In some embodiments, each of network sockets 280-1, 280-2, ..., 280-r may receive a cache request from one or more of application servers 120-1, 120-2, ..., 120-m, and each of programmable processors 270-1, 270-2, ..., 270-r may analyze the cache request. By way of example, but not limitation, programmable processors 270-1, 270-2, ..., 270-r may analyze the cache request and check whether there exists a cache server (among cache servers 150-1, 150-2, ..., 150-n) in which data related to the cache request is stored. If there is a cache server that stores the data related to the cache request, an IP address of the corresponding cache server (among cache servers 150-1, 150-2, ..., 150-n) is returned to the one or more of application servers 120-1, 120-2, ..., 120-m. On the other hand, if there is no cache server that stores the data corresponding to the cache request, a value of NULL is returned to the one or more of application servers 120-1, 120-2, ..., 120-m.

[0028] Database 230 may be configured to record information regarding a relationship between the cache request and the cache server (among cache servers 150-1, 150-2, ..., 150-n) that stores the data corresponding to the cache request. By way of example, but not limitation, when cache server 150-1 stores data related to a cache request from application server 120-1, a data packet of relationship information including the cache request, an IP address of application server 120-1 and an IP address of cache server 150-1 may be recorded in database 230.

[0029] Processor managing unit 240 may adjust the processing loads by writing or deleting a load balancing program in one or more of programmable processors 270-1, 270-2, ..., 270-r. In some embodiments, when the value NULL is returned because there is no cache server corresponding to the cache request, processor managing unit 240 may write into one of programmable processors 270-1, 270-2, ..., 270-r a load balancing program that handles the cache request by designating one or more of cache servers 150-1, 150-2, ..., 150-n. Thereafter, the corresponding programmable processor (among programmable processors 270-1, 270-2, ..., 270-r) may send one or more IP addresses of the one or more of cache servers 150-1, 150-2, ..., 150-n to the one of application servers 120-1, 120-2, ..., 120-m that transmitted the cache request.

[0030] In some embodiments, the load balancing program written into the one of programmable processors 270-1, 270-2, ..., 270-r by processor managing unit 240 may be an application-specific program determined based at least in part on an analysis of the cache request by the one of programmable processors 270-1, 270-2, ..., 270-r. By way of non-limiting example, the one of programmable processors 270-1, 270-2, ..., 270-r may analyze the cache request and determine that the cache request relates to a video transmission service, and processor managing unit 240 may proactively write into the one of programmable processors 270-1, 270-2, ..., 270-r a load balancing program (among the multiple load balancing programs stored in memory 220) of a type adapted to store the moving picture in a cache server.

[0031] Performance checking unit 250 may be configured to measure processing loads of programmable processors 270-1, 270-2, ..., 270-r. In some embodiments, performance checking unit 250 may compare a processing load of each of programmable processors 270-1, 270-2, ..., 270-r with a predetermined

value. By way of example, but not limitation, the predetermined value may be either an I/O cost or an application cost, whichever is smaller, or may be the sum of the I/O cost and the application cost. Processing managing unit 240 may then adjust the load of each programmable processors 270-1, 270-2, ..., 270-r based at least part on the comparison. That is, performance checking unit 250 may measure a throughput of each of programmable processors 270-1, 270-2, ..., 270-r, and if a delay of a certain amount of time is detected, processing managing unit 240 may change the number of programmable processors 270-1, 270-2, ..., 270-r assigned to the corresponding application.

[0032] Compiler 260 may translate the load balancing program into a bitstream representing configuration information of programmable processors 270-1, 270-2, ..., 270-r. In some embodiments, in response to a request from performance checking unit 250, processor managing unit 240 may acquire, from memory 220, a load balancing program for an application and write the load balancing program in one or more of programmable processor 270-1, 270-2, ..., 270-r. By way of example, but not limitation, complier 260 may translate the load balancing program written by HDL into a bitstream representing configuration information of the one or more of programmable processors 270-1, 270-2, ..., 270-r.

[0033] **Fig. 3** shows an example flow diagram of a process for providing a load balancing scheme, arranged in accordance with at least some embodiments described herein. The method in Fig. 3 may be implemented in or by load balancer 160, which may include multiple request processing units 210-1, 210-2, ..., 210-r, a memory 220, a database 230, a processor managing unit 240, a performance checking unit 250 and a compiler 260 discussed above. An example process may include one or more operations, actions, or functions as illustrated by one or more blocks S300, S310, S320, S330, S340, S350, S360, S370 and/or S380. Although illustrated as discrete blocks, various blocks may be divided into additional blocks, combined into fewer blocks, or eliminated, depending on the desired implementation. In the below description with regard to Fig. 3, the process will be described mainly using request processing unit 210-1 (including programmable processor 270-1 and network socket 280-1), application server 120-1 and cache server 150-1 for the convenience of explanation. However, one

skilled in the art would appreciate that the process can also be implemented by other components. Processing may begin at block S300.

[0034] At block S300 (Receiving Cache Request), request processing unit 210-1 may receive a cache request from application server 120-1 through network socket 280-1. By way of example, but not limitation, the cache request may include an HTTP (Hypertext Transfer Protocol) request such as, for example, an HTTP GET request or an HTTP POST request. Processing may continue from block S300 to block S310.

[0035] At decision block S310 (Determine Whether Data Regarding Cache Request Exists), request processing unit 210-1 may determine whether the data corresponding to the cache request exists in cache servers 150-1, 150-2, ..., 150-n. Processing may continue from block S310 to block S320, upon a negative determination, or to block S330, upon a positive determination.

[0036] At block S320 (Write Load Balancing Program), when it is determined that data regarding the cache request does not exist in cache servers 150-1, 150-2, ..., 150-n, processor managing unit 240 may write a load balancing program that may handle the cache request into programmable processor 270-1 by designating cache server 150-1, which is available for handling the cache request. Processing may continue from block S320 to block S330.

[0037] At block S330 (Transmit Address of Cache Server), programmable processor 270-1 may transmit, to application server 120-1, an address of cache server 150-1 designated by programmable processor 270-1. In some embodiments, if the data regarding the cache request exists in one of cache servers 150-1, 150-2, ..., 150-n, programmable processor 270-1 may transmit, to application server 120-1, an address of the one of cache servers 150-1, 150-2, ..., 150-n that stores the data regarding the cache request. Processing may continue from block S330 to block S340.

[0038] At block S340 (Measure Processing Load), performance checking unit 250 may measure a processing load of programmable processor 270-1. Processing may continue from block S340 to block S350.

[0039] At block S350 (Compare Processing Load), performance checking unit 250 may compare the processing load with a predetermined value. Processing may continue from block S350 to block S360.

[0040] At block S360 (Determine Whether Processing Load is Larger Than Predetermined Value), performance checking unit 250 may determine whether the processing load is larger than the predetermined value. By way of example, but not limitation, the predetermined value may be either an I/O cost or an application cost, whichever is smaller, or may be the sum of the I/O cost and the application cost. Processing may continue from block S360 to block S370 or S380.

[0041] At block S370 (Increase Number of Programmable Processors), if the processing load is larger than the predetermined value, processor managing unit 240 may increase the number of programmable processors assigned to the cache request from application server 120-1. In some embodiments, processor managing unit 240 may increase the number of programmable processors for the cache request by simply copying the loading balancing program from programmable processor 270-1 and writing it into another programmable processor. By way of example, but not limitation, complier 260 may translate the load balancing program written by HDL into a bitstream representing configuration information of the other programmable processor.

[0042] At block S380 (Decrease Number of Programmable Processors), if the processing load is not larger than the predetermined value, processor managing unit 240 may decrease the number of programmable processors assigned to the cache request from application server 120-1. In some embodiments, processor managing unit may decrease the number of programmable processors for the cache request by simply deleting the load balancing program in programmable processor 270-1.

[0043] **Fig. 4** shows an example flow diagram of a process for writing a load balancing program into a programmable processor, arranged in accordance with at least some embodiments described herein. The method in Fig. 4 may be implemented in or by load balancer 160, which may include multiple request processing units 210-1, 210-2, ..., 210-r, a memory 220, a database 230, a processor managing unit 240, a performance checking unit 250 and a compiler 260 discussed above. An example process may include one or more operations, actions, or functions as illustrated by one or more blocks S400, S410, S420 and/or S430. Those blocks are sub-blocks of block S320 of Fig. 3, and thus, the below description may be related with and/or based on the above description with regard

to Fig. 3. Although illustrated as discrete blocks, various blocks may be divided into additional blocks, combined into fewer blocks, or eliminated, depending on the desired implementation. Processing may begin at block S400.

[0044] At block S400 (Analyze Cache Request), programmable processor 270-1 may analyze the cache request from application server 120-1. Processing may continue from block S400 to block S410.

[0045] At block S410 (Determine Load Balancing Program), processor managing unit 240 may select a load balancing program from the multiple load balancing programs stored in memory 220 based at least in part on the analysis of programmable processor 270-1. By way of example, but not limitation, when the cache request is related with an SNS service (as a result of the analysis), processor managing unit 240 may choose a load balancing program of a type adapted to preferentially cache information used and/or subscribed by a greater number of users in a cache server. Processing may continue from block S410 to block S420.

[0046] At block S420 (Translate Load Balancing Program), compiler 260 may translate the load balancing program selected by processor managing unit 240 into a bitstream representing configuration information of programmable processors 270-1. Processing may continue from block S420 to block S430.

[0047] At block S430 (Write Load Balancing Program), processor managing unit 240 may write into programmable processor 270-1 the load balancing program determined by processor managing unit 240 and translated by compiler 260.

[0048] One skilled in the art will appreciate that, for this and other processes and methods disclosed herein, the functions performed in the processes and methods may be implemented in differing order. Furthermore, the outlined steps and operations are only provided as examples, and some of the steps and operations may be optional, combined into fewer steps and operations, or expanded into additional steps and operations without detracting from the essence of the disclosed embodiments.

[0049] **Fig. 5** illustrates computer program products that may be utilized to provide a load balancing scheme, arranged in accordance with at least some embodiments described herein. Program product 500 may include a signal bearing medium 502. Signal bearing medium 502 may include one or more instructions 504 that, when executed by, for example, a processor, may provide

the functionality described above with respect to Figs. 1-4. By way of example, instructions 504 may include: one or more instructions for receiving a cache request from one of at least one application server, writing, into one of the plurality of programmable processors, a load balancing program that handles the cache request by designating one of at least one cache server, and transmitting, to the one of the at least one application server, an address of the one of the at least one cache server that stores the data corresponding to the cache request. Thus, for example, referring to Fig. 2, load balancer 160 may undertake one or more of the blocks shown in Fig. 3 in response to instructions 504.

[0050] In some implementations, signal bearing medium 502 may encompass a computer-readable medium 506, including, but not limited to, a hard disk drive, a CD, a DVD, a digital tape, memory, etc. In some implementations, signal bearing medium 502 may encompass a recordable medium 508, including, but not limited to, memory, read/write (R/W) CDs, R/W DVDs, etc. In some implementations, signal bearing medium 502 may encompass a communications medium 510, including, but not limited to, a digital and/or an analog communication medium (e.g., a fiber optic cable, a waveguide, a wired communications link, a wireless communication link, etc.). Thus, for example, program product 500 may be conveyed to one or more modules of load balancer 160 by an RF signal bearing medium 502, where the signal bearing medium 502 is conveyed by a wireless communications medium 510 (e.g., a wireless communications medium conforming with the IEEE 702.11 standard).

[0051] **Fig. 6** is a block diagram illustrating an example computing device that may be utilized to provide a load balancing scheme, arranged in accordance with at least some embodiments described herein. In a very basic configuration 602, computing device 600 typically includes one or more processors 604 and a system memory 606. A memory bus 608 may be used for communicating between processor 604 and system memory 606.

[0052] Depending on the desired configuration, processor 604 may be of any type including but not limited to a microprocessor (μP), a microcontroller (μC), a digital signal processor (DSP), or any combination thereof. Processor 604 may include one more levels of caching, such as a level one cache 610 and a level two cache 612, a processor core 614, and registers 616. An example processor core 614 may include an arithmetic logic unit (ALU), a floating point unit (FPU),

a digital signal processing core (DSP Core), or any combination thereof. An example memory controller 618 may also be used with processor 604, or in some implementations memory controller 618 may be an internal part of processor 604.

[0053] Depending on the desired configuration, system memory 606 may be of any type including but not limited to volatile memory (such as RAM), non-volatile memory (such as ROM, flash memory, etc.) or any combination thereof. System memory 606 may include an operating system 620, one or more applications 622, and program data 624. Application 622 may include instructions 626 that may be arranged to perform the functions as described herein including the actions described with respect to the load balancer 160 architecture as shown in Fig. 3 or including the actions described with respect to the flow charts shown in Figs. 3 and 4. In some examples, application 622 may be arranged to operate with program data 624 on an operating system 620 such that implementations for instructions for an electronic device as described herein.

[0054] Computing device 600 may have additional features or functionality, and additional interfaces to facilitate communications between basic configuration 602 and any required devices and interfaces. For example, a bus/interface controller 630 may be used to facilitate communications between basic configuration 602 and one or more data storage devices 632 via a storage interface bus 634. Data storage devices 632 may be removable storage devices 636, non-removable storage devices 638, or a combination thereof. Examples of removable storage and non-removable storage devices include magnetic disk devices such as flexible disk drives and hard-disk drives (HDD), optical disk drives such as compact disk (CD) drives or digital versatile disk (DVD) drives, solid state drives (SSD), and tape drives to name a few. Example computer storage media may include 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.

[0055] System memory 606, removable storage devices 636 and non-removable storage devices 638 are examples of computer storage media. 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 may be used to store the desired information and which may be accessed by computing device 600. Any such computer storage media may be part of computing device 600.

[0056] Computing device 600 may also include an interface bus 640 for facilitating communication from various interface devices (e.g., output devices 642, peripheral interfaces 644, and communication devices 646) to basic configuration 602 via bus/interface controller 630. Example output devices 642 include a graphics processing unit 648 and an audio processing unit 650, which may be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 652. Example peripheral interfaces 644 include a serial interface controller 654 or a parallel interface controller 656, which may be configured to communicate with external devices such as input devices (e.g., keyboard, mouse, pen, voice input device, touch input device, etc.) or other peripheral devices (e.g., printer, scanner, etc.) via one or more I/O ports 658. An example communication device 646 includes a network controller 660, which may be arranged to facilitate communications with one or more other computing devices 662 over a network communication link via one or more communication ports 664.

[0057] The network communication link may be one example of a communication media. Communication media may typically be embodied by 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 may include any information delivery media. A "modulated data signal" may be 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 may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), microwave, infrared (IR) and other wireless media. The term computer readable media as used herein may include both storage media and communication media.

[0058] Computing device 600 may be implemented as a portion of a small-form factor portable (or mobile) electronic device such as a cell phone, a personal data assistant (PDA), a personal media player device, a wireless web-

watch device, a personal headset device, an application specific device, or a hybrid device that include any of the above functions. Computing device 600 may also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0059] The present disclosure is not to be limited in terms of the particular embodiments described in this application, which are intended as illustrations of various aspects. Many modifications and variations can be made without departing from its spirit and scope, as will be apparent to those skilled in the art. Functionally equivalent methods and apparatuses within the scope of the disclosure, in addition to those enumerated herein, will be apparent to those skilled in the art from the foregoing descriptions. Such modifications and variations are intended to fall within the scope of the appended claims. The present disclosure is to be limited only by the terms of the appended claims, along with the full scope of equivalents to which such claims are entitled. It is to be understood that this disclosure is not limited to particular methods, reagents, compounds, compositions or biological systems, which can, of course, vary. It is also to be understood that the terminology used herein is for the purpose of describing particular embodiments only, and is not intended to be limiting.

[0060] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

[0061] It will be understood by those within the art that, in general, terms used herein, and especially in the appended claims (*e.g.*, bodies of the appended claims) are generally intended as "open" terms (*e.g.*, the term "including" should be interpreted as "including but not limited to," the term "having" should be interpreted as "having at least," the term "includes" should be interpreted as "includes but is not limited to," etc.). It will be further understood by those within the art that if a specific number of an introduced claim recitation is intended, such an intent will be explicitly recited in the claim, and in the absence of such recitation no such intent is present. For example, as an aid to understanding, the following appended claims may contain usage of the introductory phrases "at least one" and "one or more" to introduce claim recitations. However, the use of such

phrases should not be construed to imply that the introduction of a claim recitation by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim recitation to embodiments containing only one such recitation, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an" (*e.g.*, "a" and/or "an" should be interpreted to mean "at least one" or "one or more"); the same holds true for the use of definite articles used to introduce claim recitations. In addition, even if a specific number of an introduced claim recitation is explicitly recited, those skilled in the art will recognize that such recitation should be interpreted to mean at least the recited number (*e.g.*, the bare recitation of "two recitations," without other modifiers, means at least two recitations, or two or more recitations). Furthermore, in those instances where a convention analogous to "at least one of A, B, and C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (*e.g.*, "a system having at least one of A, B, and C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). In those instances where a convention analogous to "at least one of A, B, or C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (*e.g.*, "a system having at least one of A, B, or C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). It will be further understood by those within the art that virtually any disjunctive word and/or phrase presenting two or more alternative terms, whether in the description, claims, or drawings, should be understood to contemplate the possibilities of including one of the terms, either of the terms, or both terms. For example, the phrase "A or B" will be understood to include the possibilities of "A" or "B" or "A and B."

[0062] In addition, where features or aspects of the disclosure are described in terms of Markush groups, those skilled in the art will recognize that the disclosure is also thereby described in terms of any individual member or subgroup of members of the Markush group.

[0063] As will be understood by one skilled in the art, for any and all purposes, such as in terms of providing a written description, all ranges disclosed

herein also encompass any and all possible subranges and combinations of subranges thereof. Any listed range can be easily recognized as sufficiently describing and enabling the same range being broken down into at least equal halves, thirds, quarters, fifths, tenths, etc. As a non-limiting example, each range discussed herein can be readily broken down into a lower third, middle third and upper third, etc. As will also be understood by one skilled in the art all language such as "up to," "at least," and the like include the number recited and refer to ranges which can be subsequently broken down into subranges as discussed above. Finally, as will be understood by one skilled in the art, a range includes each individual member. Thus, for example, a group having 1-3 cells refers to groups having 1, 2, or 3 cells. Similarly, a group having 1-5 cells refers to groups having 1, 2, 3, 4, or 5 cells, and so forth.

[0064] From the foregoing, it will be appreciated that various embodiments of the present disclosure have been described herein for purposes of illustration, and that various modifications may be made without departing from the scope and spirit of the present disclosure. Accordingly, the various embodiments disclosed herein are not intended to be limiting, with the true scope and spirit being indicated by the following claims.

CLAIMS

What is claimed is:

1. A load balancer, comprising:

a plurality of request processing units, each of the plurality of request processing units comprising:

a network socket that is connected to at least one application server and at least one cache server, and

a programmable processor configured to process a cache request from one of the at least one application server;

a performance checking unit configured to measure processing loads of the programmable processors; and

a processor managing unit configured to adjust the processing loads by writing or deleting a load balancing program in at least one of the programmable processors.

2. The load balancer of Claim 1, wherein the programmable processor is an FPGA (Field Programmable Gate Array).

3. The load balancer of Claim 1, further comprising:

a memory configured to store a plurality of load balancing programs,

wherein the network sockets are each configured to receive the cache request and the programmable processors are further configured to analyze the cache request, and

wherein the processor managing unit is further configured to select the load balancing program from the plurality of load balancing programs based at least in part on the analysis of the cache request.

4. The load balancer of Claim 1, wherein the programmable processors are further configured to check whether data corresponding to the cache request exists in the at least one cache server.

5. The load balancer of Claim 4, wherein the programmable processors are further configured, if the data corresponding to the cache request exists in the at least one cache server, to transmit, to the one of the at least one application server, an address of the cache server that stores the data corresponding to the cache request.

6. The load balancer of Claim 5, further comprising:

a database configured to record relationship information of the cache request and the cache server that stores the data corresponding to the cache request.

7. The load balancer of Claim 4, wherein the processor managing unit is further configured, if the data corresponding to the cache request does not exist in the at least one cache server, to write in one of the programmable processors the load balancing program that handles the cache request by designating one of the at least one cache server, and

wherein the one of the programmable processors is configured to transmit, to the one of the at least one application server, an address of the one of the at least one cache server designated by the load balancing program.

8. The load balancer of Claim 1, wherein the performance checking unit is further configured to compare processing load of each of the programmable processors with a predetermined value, and

wherein the processing managing unit is further configured to adjust the load of each of the programmable processors based at least part on the comparison.

9. The load balancer of Claim 1, wherein the at least one cache server is connected to a cloud datacenter.

10. The load balancer of Claim 1, further comprising:

a compiler configured to translate the load balancing program into a bitstream representing configuration information of the at least one of the programmable processors, and

wherein the processor managing unit is further configured to write the bitstream in the at least one of the programmable processors.

11. A method performed under control of a load balancer including a plurality of programmable processors, comprising:

receiving a cache request from one of at least one application server;

writing, into one of the plurality of programmable processors, a load balancing program that handles the cache request by designating one of at least one cache server; and

transmitting, to the one of the at least one application server, an address of the one of the at least one cache server corresponding to the cache request.

12. The method of Claim 11, further comprising:

determining whether the data corresponding to the cache request exists in the at least one cache server.

13. The method of Claim 11, further comprising:

measuring processing load of each of the plurality of programmable processors;

comparing the processing load with a predetermined value; and

adjusting the processing load based at least in part on the comparing.

14. The method of Claim 13, wherein the adjusting comprises writing the load balancing program into at least one of the plurality of programmable processors.

15. The method of Claim 11, wherein the programmable processor is an FPGA (Field Programmable Gate Array).

16. The method of Claim 11, wherein the at least one cache server is connected to a cloud datacenter.

17. The method of Claim 11, further comprising:

recording, in a database of the load balancer, relationship information of the cache request and the one of the at least one cache server.

18. The method of Claim 11, further comprising:

analyzing the cache request; and

determining the load balancing program from a plurality of load balancing programs based at least in part on the analyzing.

19. The method of Claim 11, further comprising:

translating the load balancing program into a bitstream representing configuration information of the one of the plurality of programmable processors,

wherein the writing includes writing the bitstream into the one of the plurality of programmable processors.

20. A computer-readable storage medium having stored thereon computer-executable instructions that, in response to execution, cause a load balancer including a plurality of programmable processors to perform operations, comprising:

receiving a cache request from one of at least one application server;

writing, into one of the plurality of programmable processors, a load balancing program that handles the cache request by designating one of at least one cache server; and

transmitting, to the one of the at least one application, server an address of the one of the at least one cache server corresponding to the cache request.

21. The computer-readable storage medium of Claim 20, wherein the programmable processor is an FPGA (Field Programmable Gate Array).

22. The computer-readable storage medium of Claim 20, wherein the operations further comprise:

measuring processing load of each of the plurality of programmable processors;

comparing the processing load with a predetermined value; and

adjusting the processing load based at least in part on the comparing.

23. The computer-readable storage medium of Claim 22, wherein the adjusting comprises writing the load balancing program into at least one of the plurality of programmable processors.

24. The computer-readable storage medium of Claim 20, wherein the operations further comprise:

analyzing the cache request; and

determining the load balancing program from a plurality of load balancing programs based at least in part on the analyzing.

FIG. 1

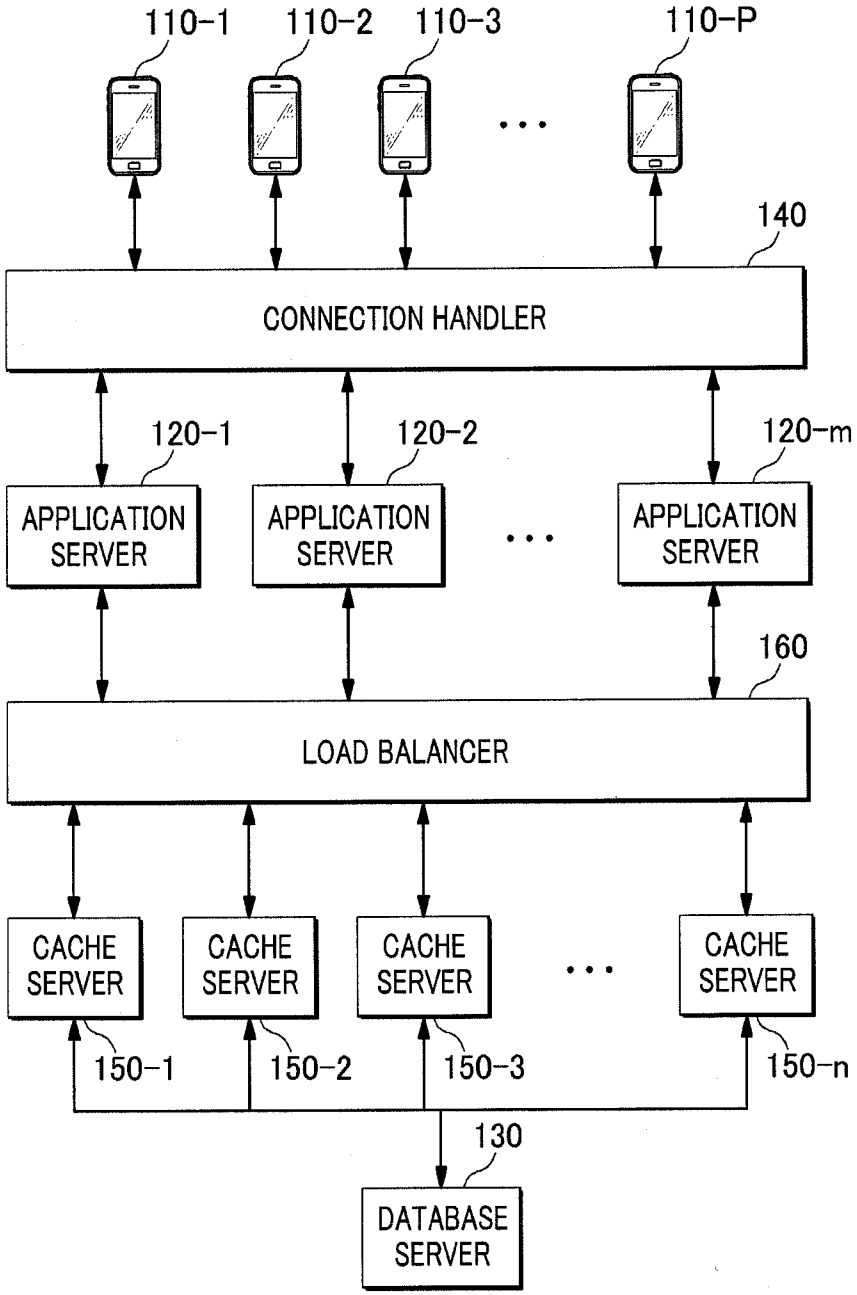

110-1
110-2
110-3
110-P
140
CONNECTION HANDLER
120-1
120-2
120-m
APPLICATION SERVER
APPLICATION SERVER
APPLICATION SERVER
160
LOAD BALANCER
CACHE SERVER
CACHE SERVER
CACHE SERVER
CACHE SERVER
150-1
150-2
150-3
150-n
130
DATABASE SERVER

FIG. 2

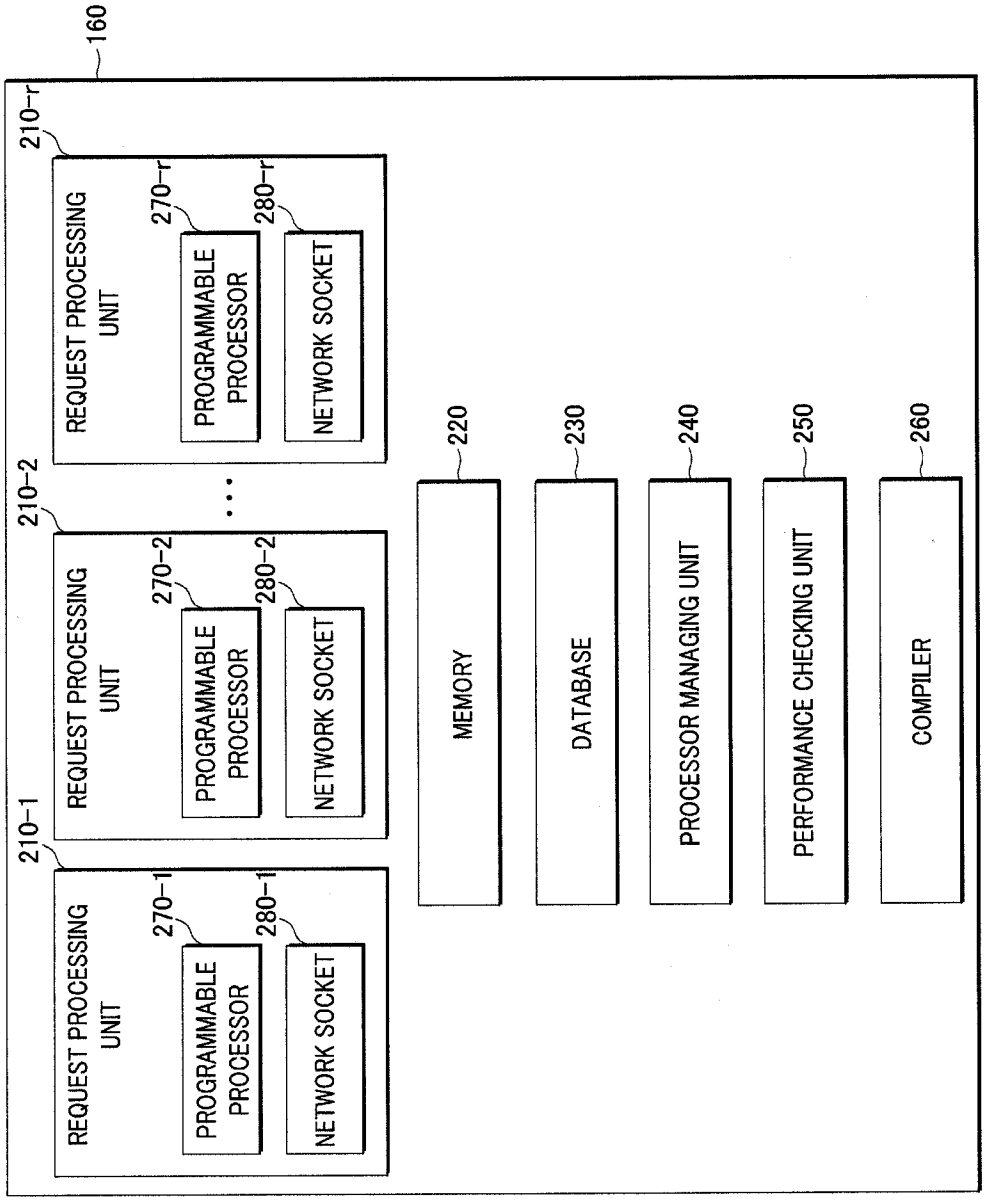

160
210-1
REQUEST PROCESSING UNIT
270-1
PROGRAMMABLE PROCESSOR
280-1
NETWORK SOCKET
210-2
REQUEST PROCESSING UNIT
270-2
PROGRAMMABLE PROCESSOR
280-2
NETWORK SOCKET
210-r
REQUEST PROCESSING UNIT
270-r
PROGRAMMABLE PROCESSOR
280-r
NETWORK SOCKET
220
MEMORY
230
DATABASE
240
PROCESSOR MANAGING UNIT
250
PERFORMANCE CHECKING UNIT
260
COMPILER

FIG. 3

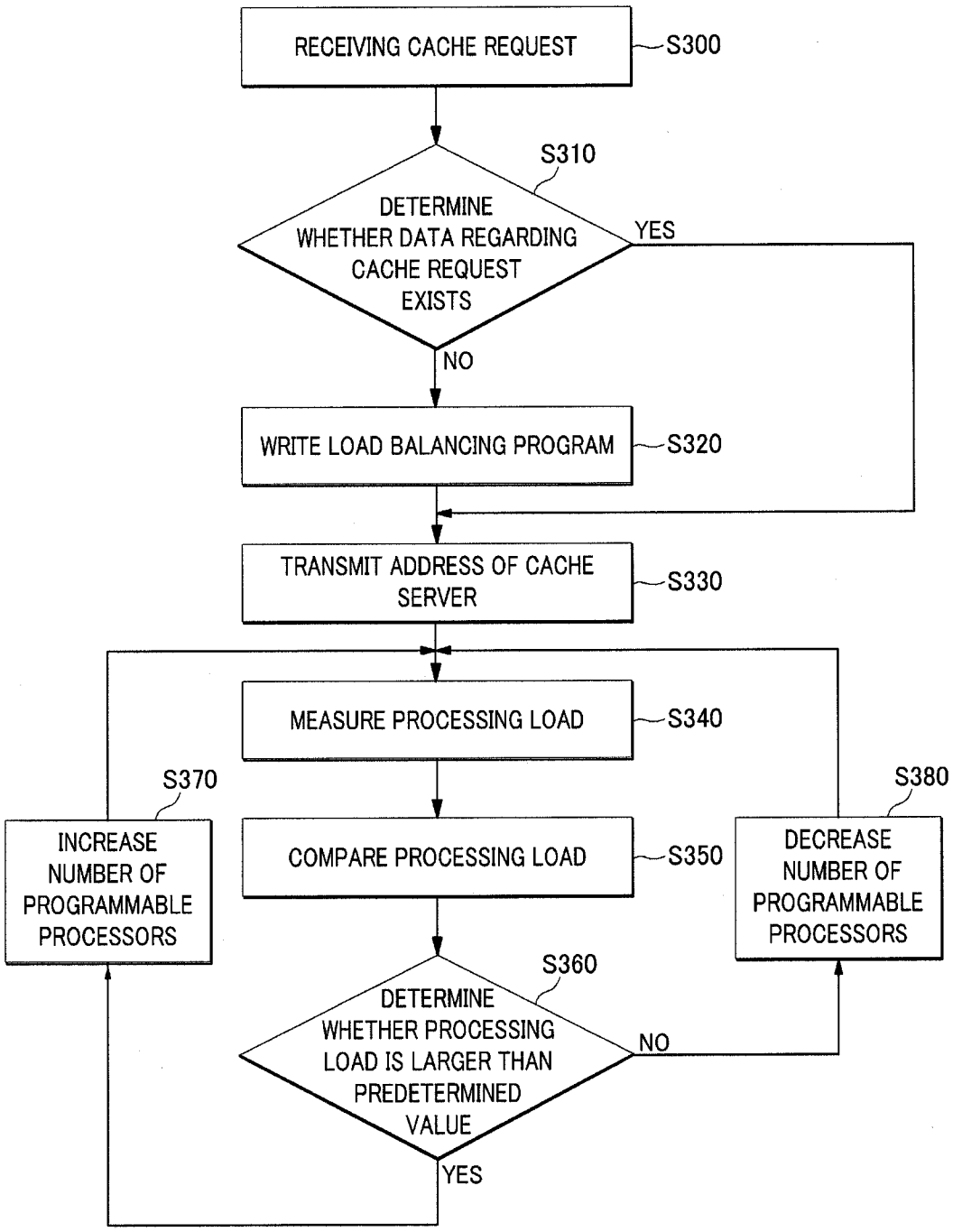

RECEIVING CACHE REQUEST
S300
S310
DETERMINE WHETHER DATA REGARDING CACHE REQUEST EXISTS
YES
NO
WRITE LOAD BALANCING PROGRAM
S320
TRANSMIT ADDRESS OF CACHE SERVER
S330
MEASURE PROCESSING LOAD
S340
S370
INCREASE NUMBER OF PROGRAMMABLE PROCESSORS
COMPARE PROCESSING LOAD
S350
S380
DECREASE NUMBER OF PROGRAMMABLE PROCESSORS
S360
DETERMINE WHETHER PROCESSING LOAD IS LARGER THAN PREDETERMINED VALUE
NO
YES

FIG. 4

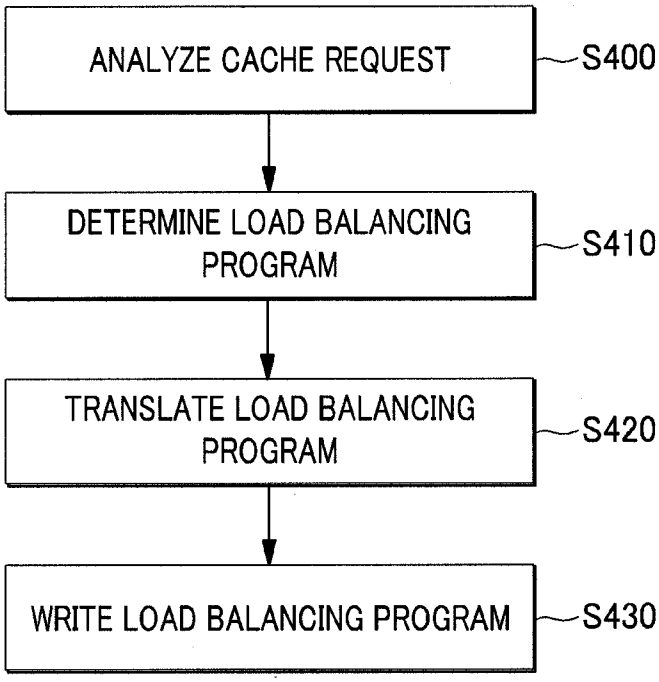

ANALYZE CACHE REQUEST
S400
DETERMINE LOAD BALANCING PROGRAM
S410
TRANSLATE LOAD BALANCING PROGRAM
S420
WRITE LOAD BALANCING PROGRAM
S430

FIG. 5

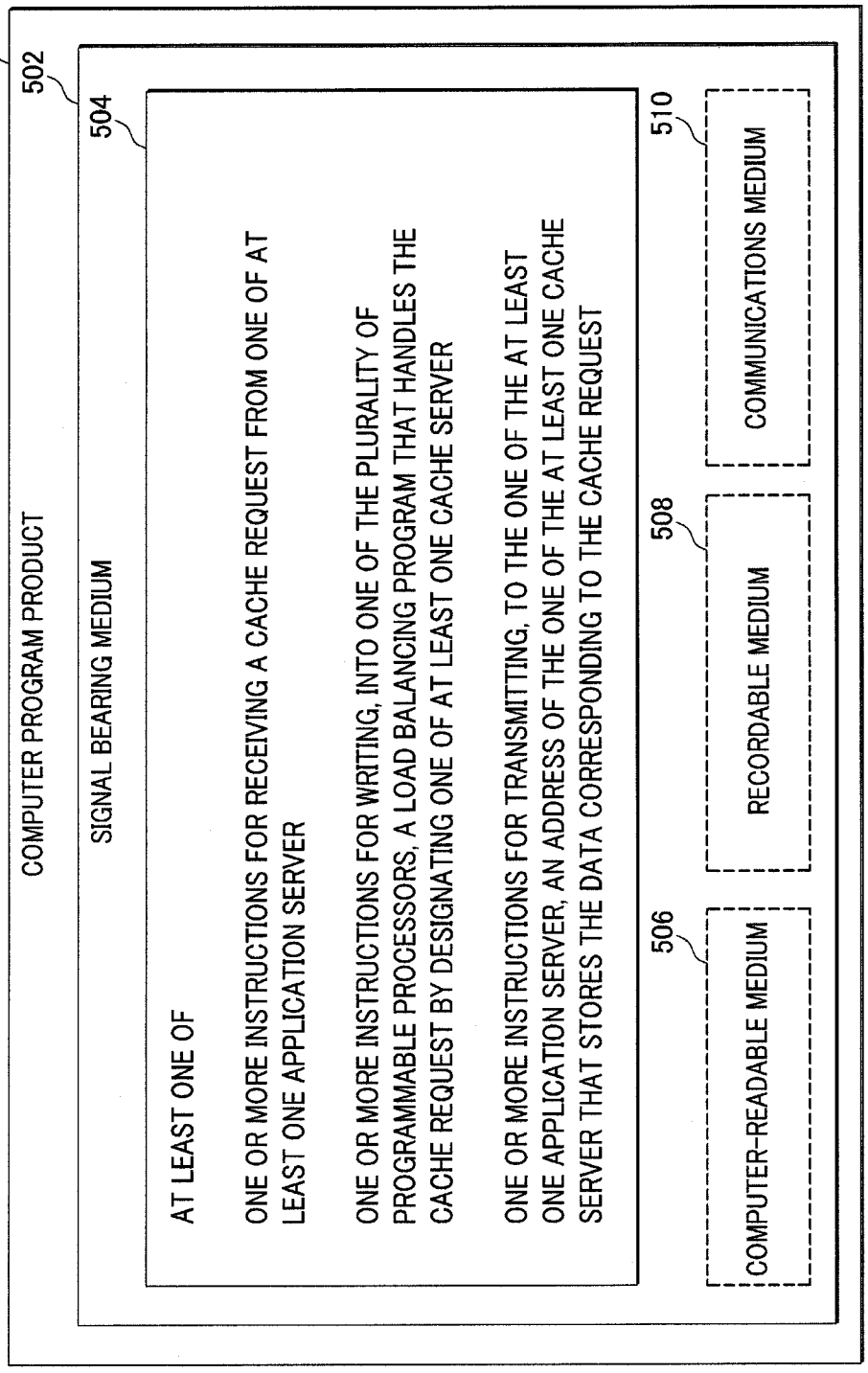
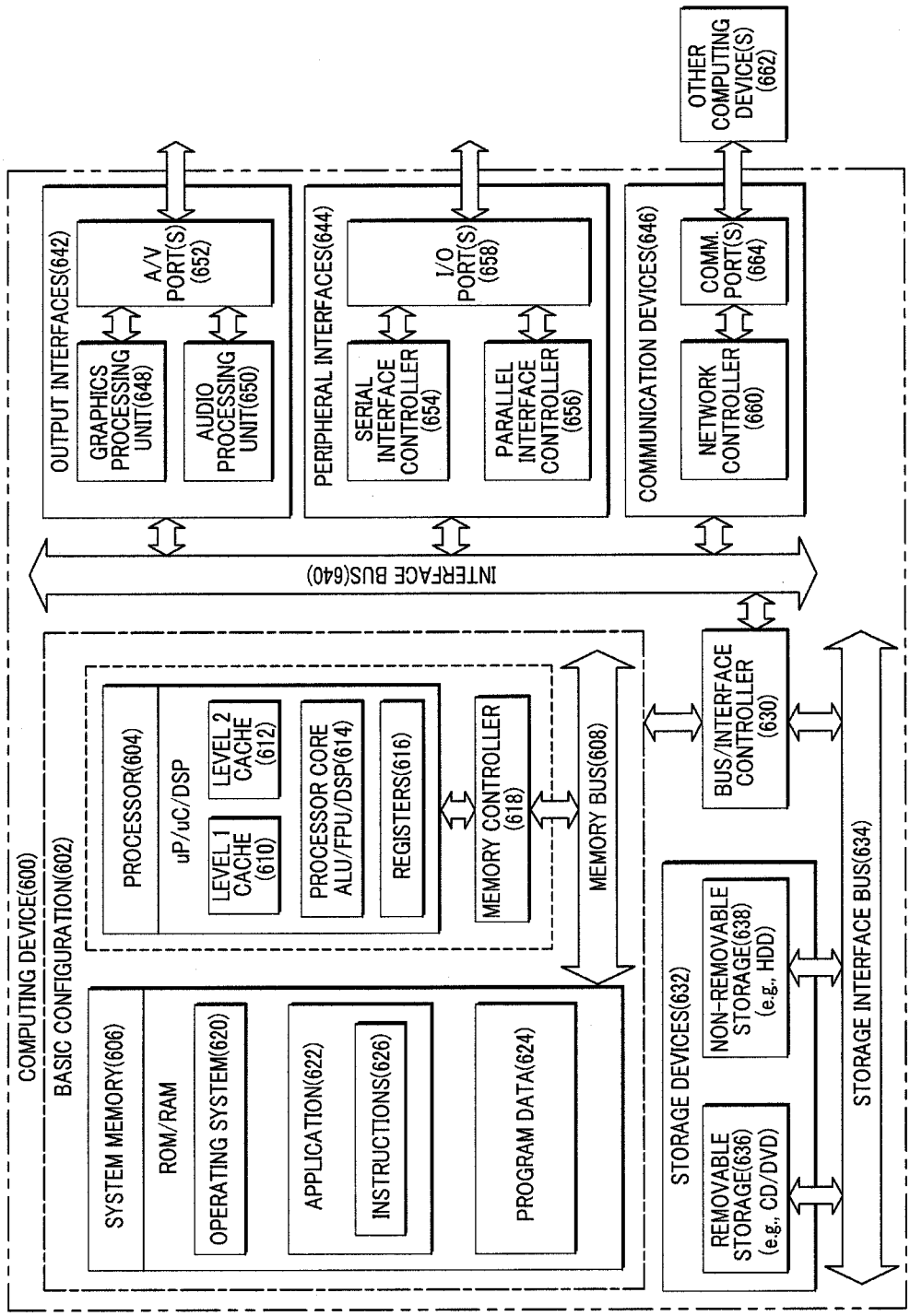

500
COMPUTER PROGRAM PRODUCT
502
SIGNAL BEARING MEDIUM
504
AT LEAST ONE OF
ONE OR MORE INSTRUCTIONS FOR RECEIVING A CACHE REQUEST FROM ONE OF AT LEAST ONE APPLICATION SERVER
ONE OR MORE INSTRUCTIONS FOR WRITING, INTO ONE OF THE PLURALITY OF PROGRAMMABLE PROCESSORS, A LOAD BALANCING PROGRAM THAT HANDLES THE CACHE REQUEST BY DESIGNATING ONE OF AT LEAST ONE CACHE SERVER
ONE OR MORE INSTRUCTIONS FOR TRANSMITTING, TO THE ONE OF THE AT LEAST ONE APPLICATION SERVER, AN ADDRESS OF THE ONE OF THE AT LEAST ONE CACHE SERVER THAT STORES THE DATA CORRESPONDING TO THE CACHE REQUEST
506
COMPUTER-READABLE MEDIUM
508
RECORDABLE MEDIUM
510
COMMUNICATIONS MEDIUM

FIG. 6


COMPUTING DEVICE(600)
BASIC CONFIGURATION(602)
PROCESSOR(604)
uP/uC/DSP
LEVEL 1 CACHE (610)
LEVEL 2 CACHE (612)
PROCESSOR CORE ALU/FPU/DSP(614)
REGISTERS(616)
MEMORY CONTROLLER (618)
MEMORY BUS(608)
SYSTEM MEMORY(606)
ROM/RAM
OPERATING SYSTEM(620)
APPLICATION(622)
INSTRUCTIONS(626)
PROGRAM DATA(624)
INTERFACE BUS(640)
OUTPUT INTERFACES(642)
GRAPHICS PROCESSING UNIT(648)
AUDIO PROCESSING UNIT(650)
A/V PORT(S) (652)
PERIPHERAL INTERFACES(644)
SERIAL INTERFACE CONTROLLER (654)
PARALLEL INTERFACE CONTROLLER (656)
I/O PORT(S) (658)
COMMUNICATION DEVICES(646)
NETWORK CONTROLLER (660)
COMM. PORT(S) (664)
OTHER COMPUTING DEVICE(S) (662)
BUS/INTERFACE CONTROLLER (630)
STORAGE DEVICES(632)
REMOVABLE STORAGE(636) (e.g., CD/DVD)
NON-REMOVABLE STORAGE(638) (e.g., HDD)
STORAGE INTERFACE BUS(634)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US12/70063

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/46 (2013.01)
USPC - 718/105
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8): G06F 12/00; 13/00, 17/30, 9/46, 15/173, 15/16, 17/56 (2013.01)
USPC: 709/217, 226, 224, 246, 235; 707/E17.12; 711/122, 118; 718/105; 712/15

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
MicroPatent (US-G, US-A, EP-A, EP-B, WO, JP-bib, DE-C,B, DE-A, DE-T, DE-U, GB-A, FR-A); DialogPRO; IEEE/IEEEXplore; Google/Google Scholar; IP.com; Search Terms: application, server, request, cache, buffer, write, install, delete, reprogram, rewrite, instruction, api, fpga, workload, load, balance

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 2008/0091845 A1 (MILLS, B., et al.) April 17, 2008, figures 1B, 2A, 3, paragraphs [015], [0025], [0058], [0059], [0061], [0063], [0064], [0066], [0068], [0071], [0080], [0095], [0096]	1-8, 11-15, 17, 18, 20-24 ---------- 9, 10, 16, 19
Y	US 2012/0271964 A1 (PORTER, K.) October 25, 2012, abstract, figure 3, paragraphs [0012], [0031], [0037]	9, 16
Y	US 2011/0238948 A1 (VORBACH, M., et al.) September 29, 2011, abstract, paragraphs [0908], [1005], [1039]	10, 19

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
8 February 2013 (08.02.2013)	26 FEB 2013
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Shane Thomas PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)