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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 May 2009 (14.05.2009)

PCT

(10) International Publication Number
WO 2009/061432 A1

(51) International Patent Classification:
G06F 15/173 (2006.01)

(21) International Application Number:
PCT/US2008/012514

(22) International Filing Date:
6 November 2008 (06.11.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/985,915 6 November 2007 (06.11.2007) US

(71) Applicant *(for all designated States except US)*: **CREDIT SUISSE SECURITIES (USA) LLC** [US/US]; 11 Madison Avenue, New York, New York 10010 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **BUBBA, Head** [US/US]; Suite 2H, 9921 4th Avenue, Brooklyn, New York 11209 (US).

(74) Agents: **KING & SPALDING** et al.; Intellectual Property Dept. - Patents, 34th Floor, 1180 Peachtree Street, N.E., Atlanta, Georgia 30309-3521 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) **Title:** PREDICTING AND MANAGING RESOURCE ALLOCATION ACCORDING TO SERVICE LEVEL AGREEMENTS

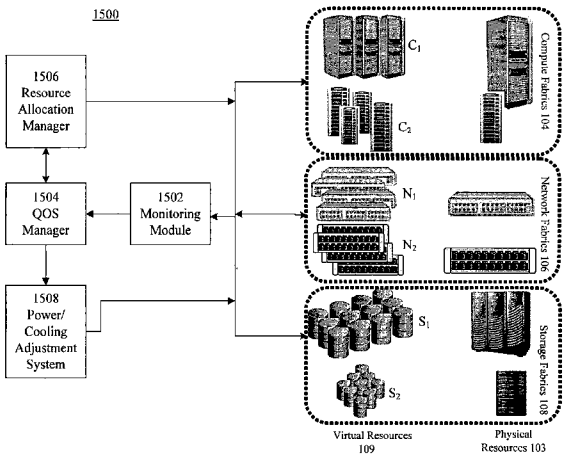


Figure 15

(57) **Abstract:** Allocating computing resources comprises allocating an amount of a resource to an application program based on an established service level requirement for utilization of the resource by the application program, determining whether the application program's utilization of the resource exceeds a utilization threshold, and changing the allocated amount of the resource in response to a determination that the application program's utilization of the resource exceeds the utilization threshold. The utilization threshold is based on the established service level requirement and is different than the established service level requirement. Changing the allocation of the resource based on the utilization threshold allows allocating sufficient resources to the application program prior to a breach of a service level agreement for the application program.

WO 2009/061432 A1

PREDICTING AND MANAGING RESOURCE ALLOCATION ACCORDING TO SERVICE LEVEL AGREEMENTS

RELATED APPLICATIONS

This patent application claims priority under 35 U.S.C. § 119 to United States Provisional Patent Application No. 60/985,915 entitled "Quality of Service, Service Level Agreement Manager" filed November 6, 2007. This patent application also is related to United States Patent Application No. 11/900,424 entitled "Economic Allocation and Management of Resources Via a Virtual Resource Market" filed September 11, 2007, which claims priority under 35 U.S.C. § 119 to United States Provisional Patent Application No. 60/908,350 entitled "Economic Allocation and Management of Resources Via a Virtual Resource Market" filed March 27, 2007. The complete disclosure of each of the above-identified priority and related applications is hereby fully incorporated herein by reference.

TECHNICAL FIELD

The invention relates to allocating and managing distributed computing resources. More particularly, the invention relates to dynamic monitoring and allocation of available resources and managing the available resources to modify the allocation of resources prior to a breach of a service level agreement for an application program.

BACKGROUND

In an enterprise organization, distributed computing resources, such as compute resources, network resources, and storage resources, are provided to operate multiple application programs. The distributed computing resources may be provided in one general location, in which case the resources communicate via one or more local area networks. Alternatively, the distributed computing resources may be provided in several locations, in which case the resources communicate via one or more local area networks and one or more wide area networks, such as the Internet.

The distributed computing resources can be allocated among application programs to accommodate the operations of those programs. For example, compute resources, network resources, and storage resources can be allocated to each application program, and each

application program can utilize its allocated resources for its operation. However, conventional allocation methods allocate a particular resource to an application program based on whether the particular resource currently is underutilized and can accommodate the operations of an additional application program. In this manner, the operations of additional application programs are added to a resource until the resource is utilized to its maximum capacity or beyond.

Such conventional allocation of resources does not account for business or operational priorities, or operational requirements, or quality of service of the application programs. Accordingly, limited and valuable resources may be allocated to lower-priority application programs instead of to higher-priority applications simply because the lower-priority application programs were addressed first, before the higher-priority application programs. Additionally, such conventional allocation of resources does not result in an economically efficient distribution of resources based on the value accorded to specific resources by an application program or by the user of an application program. Thus, resources allocated via conventional methods may be under-utilized because those resources are consumed by application programs that cannot fully utilize the specific performance characteristics of the allocated resources. Alternatively, resources allocated via conventional methods may be under-performing because those resources are consumed by application programs that require more sophisticated performance characteristics than those of the allocated resources.

Furthermore, in conventional resource allocation methods, resources are not dynamically reallocated to maintain an efficicient and/or economical distribution of the resources or to anticipate and prevent a breach of a service level agreement. Once resources are initially allocated to an application program, the application program operates on those resources indefinitely. When the resource becomes over-utilized as more and more application program operations are added to the resource, the conventional solution is to buy more resources. That conventional solution does not consider reallocating existing resources to locate unused capacity of the existing resources. That conventional solution also does not consider the need to dynamically migrate a program's operations to more suitable resources. Additionally, conventional solutions do not reallocate resources until those resources are unable to meet the service level requirements of a service level

agreement associated with an application program utilizing those resources. Accordingly, additional or alternative resources are not allocated to the application program until the service level agreement has already been breached.

Accordingly, a need exists in the art for efficiently allocating resources among application programs. A further need exists for monitoring the utilization of resources allocated among application programs to allow a new allocation of more appropriate resources prior to a breach of service level agreements associated with the application programs.

SUMMARY

The invention includes methods and systems for allocating and managing distributed computing resources to maintain appropriate resource allocation without breaching a service level agreement associated with an application program utilizing the allocated resources. The invention can anticipate a potential future breach of a service level agreement and can allocate additional or alternative resources to the application program to avoid the breach and to meet the requirements of the service level agreement. In one aspect, resources allocated to a lower priority application program can be shifted to a higher priority application program to prevent breaching a service level agreement associated with the higher priority application program. In another aspect, a particular application program, or a family of application programs including the particular application program, can be migrated to alternative resources to prevent breaching a service level agreement associated with the particular application program.

In yet another aspect, resources are monitored to ensure compliance with the service level requirements set forth in the service level agreement associated with an application program. The resources are monitored to determine when utilization of a particular resource by the application program exceeds a predetermined threshold that is lower than an amount of the particular resource that is currently allocated to the application program. When the utilization exceeds the threshold, additional or alternative resources are allocated to the application program, thereby allocating sufficient resources prior to the utilization exceeding the allocated amount, which is based on the service level requirement and thus prior to a breach of the service level agreement associated with the application program.

In yet another aspect, power and/or cooling requirements for resources can be adjusted after resource reallocation. If an allocated amount of a particular resource is increased or decreased, then the amount of power and/or cooling provided to that resource can be adjusted accordingly. If an application program is migrated from an original resource to an alternate resource, then the power and/or cooling can be decreased for the original resource and increased for the alternate resource. Accordingly, increased energy efficiency and cost savings can be achieved.

Additionally, the invention includes methods and systems for allocating and managing distributed computing resources via a market exchange model. Utilizing the market exchange model can result in an efficient and economic allocation of the resources for use by application programs. The resources can be allocated based on market prices for units of each resource. Accordingly, offers to provide resources for use by application programs can be created, where each offer specifies at least one performance characteristic and a value associated with a corresponding unit of resource. Bids to obtain the resources for use by the application programs also are created. Each bid specifies at least one service level required for operation of a corresponding application program and a value corresponding to a perceived value associated with operating the corresponding application program or to a perceived value of obtaining a resource that meets or exceeds the service level requirements of the corresponding application program. Bids are matched to offers via the market exchange model by matching the service level requirement and value of each bid to the performance characteristic and value of one of the offers. Then, resources associated with each offer are allocated to the application program associated with a matching bid, and the application program's operations are migrated to the allocated resources. Resources are monitored to ensure compliance with the service level requirement of each bid, and non-complying resources are replaced via the market exchange model.

Performance monitoring can be performed for each application program that has been allocated resources in the distributed computing environment. In this manner, the performance of the allocated resources is continually or periodically monitored to ensure that the resources meet the service level requirements of the application program. If the resources are not providing the required level of service, then new bids can be created and submitted to the market exchange to obtain resources that will meet the service level

requirements. Additionally, in this manner, the performance of the allocated resources is continually or periodically monitored to ensure that the performance of the resources does not exceed the service level requirements by an amount that would indicate an application program user is paying for unused resources (in other words, excess capacity of the resource). If the resources are not being sufficiently utilized by the application program, then new bids can be created and submitted to the market exchange to obtain resources that are more suitable and/or economical with regard to the service level requirements of the application program.

Thus, performance of allocated resources can be monitored to determine whether those resources are under-utilized or under-performing. If so, new resources can be identified and allocated to the application program, and the application program's operations can be migrated to the new resources, thereby providing the properly performing resources for the application program.

After the resources have been allocated as described previously over a period of time, information related to the allocated resources can be used to manage the resources. Profit and loss information for each of the resources is generated by subtracting actual or idealized costs and expenses for each resource from actual or idealized revenues generated by the resource during the relevant time period. Profit and loss information is compared to determine in which resources the owner should invest, in which resources the owner should maintain its current position, and which resources the owner should divest. The owner can invest in resources that are making a profit and can divest resources that are generating a loss.

Accordingly, an owner or user can evaluate physical resources to determine which resources are the most economical, including both price and performance. For example, compute resources can comprise different platforms that each has different price and performance characteristics. Comparison of the profit and loss statements of each resource for the time period will show which platforms are generating more revenue per unit of performance. Based on that information, the owner can determine which platforms to maintain or in which to increase the owner's position and which platforms to divest. Because the resources have been allocated as described previously, the evaluation provides

an indication of which resources are more desirable, based on price and performance characteristics, for use by application programs.

These and other aspects, objects, and features of the invention will become apparent from the following detailed description of the exemplary embodiments, read in conjunction with, and reference to, the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram depicting a system for allocating and managing distributed computing resources according to an exemplary embodiment of the invention.

Figure 2 is a flow chart depicting a method for allocating and managing distributed computing resources according to an exemplary embodiment of the invention.

Figure 3 is a flow chart depicting a method for creating offers to provide available virtual resources at a specified price per unit of performance for each virtual resource according to an exemplary embodiment of the invention.

Figure 4 is a flow chart depicting a method for identifying service level requirements required for operation of an application program according to an exemplary embodiment of the invention.

Figure 5 is a flow chart depicting a method for creating bids to obtain units of resources required to meet the service level requirements of an application program according to an exemplary embodiment of the invention.

Figure 6 is a flow chart depicting a method for matching resource offers to requirements bids to obtain virtual resources for use by an application program according to an exemplary embodiment of the invention.

Figure 7 is a flow chart depicting a method for completing a transaction to purchase resources required for operation of an application program based on matching resource offers and requirements bids according to an exemplary embodiment of the invention.

Figure 8 is a flow chart depicting a method for migrating an application program's operations to purchased resources according to an exemplary embodiment of the invention.

Figure 9 is a flow chart depicting a method for monitoring the performance of allocated resources and the service level requirements of an application program according to an exemplary embodiment of the invention.

Figure 10 is a block diagram depicting reallocation of distributed computing resources for an application program according to an exemplary embodiment of the invention.

Figure 11 is a flow chart depicting a method for managing the maintenance, acquisition, and/or divestment of resources based on resource allocation over a period of time according to an exemplary embodiment of the invention.

Figure 12 is a flowchart depicting a method for managing resource requirements based on a service level agreement for an application program according to an exemplary embodiment of the invention.

Figure 13 is a flowchart depicting a method for modifying the allocation of a selected resource for an application program prior to breach of a service level agreement associated with the application program according to an exemplary embodiment of the invention.

Figure 14 is a flow chart depicting a method for adjusting the cooling and power requirements of original and alternate resources based on a change in allocated resources according to an exemplary embodiment of the invention.

Figure 15 is a block diagram depicting a system for managing resource requirements based on a service level agreement for an application program according to an exemplary embodiment of the invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Referring to the drawings, in which like numerals represent like elements, aspects of the exemplary embodiments will be described.

Figure 1 is a block diagram depicting a system 100 for allocating and managing distributed computing resources according to an exemplary embodiment of the invention. The system 100 will be discussed in more detail with reference to the methods illustrated in Figures 2-9 and 11.

Figure 2 is a flow chart depicting a method 200 for allocating and managing distributed computing resources according to an exemplary embodiment of the invention. The method 200 will be described with reference to Figures 1 and 2.

According to one exemplary embodiment, the distributed computing resources can be resources of an enterprise organization. In that case, users of the resources are members of the organization. Alternatively, the distributed computing resources can be resources of multiple organizations coupled to a central system for allocating those resources.

As illustrated in step 205 of Figure 2, a resource broker 102 monitors distributed computing resources for providing computing services to identify resource capacity that is available to operate one or more application programs. In an exemplary embodiment, the distributed computing resources can comprise physical computing resources 103, such as compute fabrics 104, network fabrics 106, and storage fabrics 108. Each of the compute fabrics 104, network fabrics 106, and storage fabrics 108 can comprise one or more virtual resources 109 configured to provide services. For example, as illustrated in Figure 1, the compute fabrics 104 comprise virtual compute fabrics C1 and C2, the network fabrics 106 comprise virtual network fabrics N1 and N2, and the storage fabrics 108 comprise virtual storage fabrics S1 and S2.

In an exemplary embodiment, the resource broker 102 can comprise a software module operating in the distributed computing environment and functioning as an interface between virtual resources 109 and a market exchange 112.

In step 210, the resource broker 102 creates offers to provide available virtual resources 109 at a specified price per unit of performance for each resource. Such offers are referred to herein as "resource offers 110" 110. One or more resource offers 110 can be created for each of the fabrics 104, 106, 108. For example, as illustrated in Figure 1, the resource offers 110 comprise three offers of the compute fabric C_1, two offers of the compute fabric C_2, three offers of the network fabric N_1, two offers of the network fabric N_2, three offers of the storage fabric S_1, and two offers of the storage fabric S_2. Step 210 will be described in further detail hereinafter with reference to Figure 3.

Then, in step 215, the resource broker 102 communicates the resource offers 110 to the market exchange 112. In an exemplary embodiment, the market exchange 112 can comprise a software module operating in the distributed computing environment and functioning as an interface between the resource broker 102 and a requirements broker 114.

In step 220, the requirements broker 114 identifies service level resource requirements required for operation of an application program. In an exemplary

embodiment, the required resources can comprise computing, network, and storage resources, such as one or more of the compute fabrics 104, network fabrics 106, and storage fabrics 108 illustrated in Figure 1. Application programs can conform to well-established architectures, such as canonical application architectures 116. The exemplary canonical application architectures illustrated in Figure 1 comprise a message hub illustrated as canonical architecture A, an n-tier application illustrated as canonical application architecture B, and a compute farm illustrated as canonical application architecture C. Other application architectures are within the scope of the invention. Step 220 will be described in further detail hereinafter with reference to Figure 4.

In an exemplary embodiment, the requirements broker 114 can comprise a software module operating in the distributed computing environment and functioning as an interface between application programs and the market exchange 112.

In step 225, bids to purchase units of resources required to meet the service level requirements of an application program are created. Such bids are referred to herein as "requirements bids 118." One or more requirements bids 118 can be created for each application program. For example, as illustrated in Figure 1, the requirements bids 118 comprise a bid for compute fabric, a bid for network fabric, and a bid for storage fabric for each canonical architecture A, B, C. Step 225 will be described in further detail hereinafter with reference to Figure 5.

In step 230, the requirements broker 114 communicates the requirements bids 118 to the market exchange 112. Then, in step 235, the market exchange 112 matches the resource offers 110 to the requirements bids 118 to determine an economical and efficient allocation of the resources to each application program. Step 235 will be described in further detail hereinafter with reference to Figure 6.

In step 240, the market exchange 112 completes a transaction to allocate the resources required for the application program based on matching offers and bids. Step 240 will be described in further detail hereinafter with reference to Figure 7.

In step 245, the application program's operations are migrated to the allocated resources. Step 245 will be described in further detail hereinafter with reference to Figure 8.

In step 250, the requirements broker 114 monitors the performance of the allocated resources and the service level requirements of each application program to insure that the

performance of the allocated resources meets the service level requirements of each application program. Step 250 will be described in further detail hereinafter with reference to Figure 10.

In step 255, based on the performance monitoring completed in step 250, the requirements broker 114 determines whether the allocated resources are under-utilized or underperforming with reference to the service level requirements of a particular application program. If yes, then the method 200 branches back to step 225 to obtain new resources that meet the service level requirements of the application program. If the requirements broker 114 determines in step 255 that the allocated resources meet the service level requirements of the application program, then the method branches to step 260.

In step 260, the method 200 determines whether operation of the application program is complete. If not, then the method 200 branches back to step 250 to continue monitoring the performance of the allocated resources and the service level requirements of the application program. If the method 200 determines in step 260 that the operation of the application program is complete, then the method 200 branches to step 265 in which the application program's operations are removed from the allocated resources.

The method 200 also includes managing the maintenance, acquisition, and divestment of the resources based on resource allocation over a predetermined period of time, as illustrated in step 270 of Figure 1. Step 270 will be described in further detail hereinafter with reference to Figure 11.

Figure 3 is a flow chart depicting a method 210 for creating offers to provide available virtual resources 109 at a specified price per unit of performance for each virtual resource 109 according to an exemplary embodiment of the invention, as referred to in step 210 of Figure 2. The method 210 will be described with reference to Figures 1 and 3.

In step 305, the resource broker 102 selects a physical resource 103 that is available for use by an application program. For example, the resource broker 102 can select a compute, network, or storage resource, such as the compute fabrics 104, the network fabrics 106, and the storage fabrics 108. In an exemplary embodiment, the resource broker 102 can monitor the use of each resource to identify excess capacity of each resource that is not being utilized. Such excess capacity can be identified as resources that are available for use by an application program.

Then, in step 310, the resource broker 102 identifies an amount of the selected physical resource 103 that is available for use by an application program. In an exemplary embodiment, the amount of each resource can comprise a hardware type and/or configuration, including a specific manufacturer and/or component, as well as the excess capacity currently available for each resource. In this regard, available physical resources 103 can be combined to create virtual resources 109, such as the virtual compute fabrics C_1, C_2, the virtual network fabrics N_1, N_2, and the virtual storage fabrics S_1, S_2. For example, computer processors available at distributed locations can be aggregated to create a virtual computing resource that is available for use by an application program. Representative resource capacities can comprise CPU cycles for computing resources, bandwidth for network resources, and disk space and/or memory for storage resources.

The amount of resource available also can comprise performance and reliability characteristics associated with each virtual resource 109. For example, such characteristics can comprise execution time, response time, accuracy of results (such as fault rate), availability, reliability, security, or other suitable characteristics indicating the performance of the virtual resources 109.

In step 315, the resource broker 102 identifies a unit amount of the virtual resource 109 to include in an offer to provide the virtual resource 109 for use by an application program. In this regard, the resource broker 102 can identify portions of a virtual resource 109 that are available for allocation in increments of the identified unit up to the maximum amount of the virtual resource 109 that is available. If the resource broker 102 identified multiple virtual resources 109, such as the virtual compute fabrics C_1, C_2, in step 310, then the resource broker 102 can identify a unit amount for each virtual resource 109.

In step 320, a price is established for each unit of virtual resource 109 that will be included in an offer to provide the virtual resource 109 for use by an application program. In an exemplary embodiment, the resource broker 102 can calculate a price per unit of virtual resource 109 based on resource sophistication, cost data, supply, demand, or other economic data input into the resource broker 102 by a user or obtained by the resource broker 102 based on historical data of completed resource allocation transactions. For example, more sophisticated resources can be more expensive than less sophisticated resources, and high-demand resources can be more expensive than lower-demand resources.

The price can comprise a monetary amount at which the resource will be sold for use by an application program. Alternatively, the value can represent a perceived value of the resource that is not based on actual monetary value. For example, the perceived value can be based on business requirements and priorities established within the business enterprise.

In step 325, the resource broker 102 generates one or more resource offers 110 to provide the available virtual resources 109 for use by an application program. The resource offers 110 can specify the virtual resource 109, the amount of virtual resource 109 available (including capacity and performance), and the unit price of the virtual resource 109, based on the information obtained in steps 310-320.

In step 330, the resource broker 102 determines whether to generate resource offers 110 for another virtual resource 109. For example, if the resource broker 102 has generated resource offers 110 for only available compute fabrics 104, then the resource broker 102 can decide to generate resource offers 110 for available network and storage fabrics 106, 108. In that case, the method 210 branches back to step 305 to generate resource offers 110 for another resource. If the method 210 determines in step 330 not to generate resource offers 110 for another virtual resource 109, then the method 210 branches to step 215 (Figure 2).

Figure 4 is a flow chart depicting a method 220 for identifying service level requirements required for operation of an application program according to an exemplary embodiment of the invention, as referred to in step 220 of Figure 2. The method 220 will be described with reference to Figures 1 and 4.

In step 405, an application program is selected. Then, specific service level requirements for the selected application program are identified in steps 410-435.

In step 410, a budget available for operating the application program is identified. In an exemplary embodiment, the budget can be input by a user of the application program and designed to meet the user's budgetary constraints. For example, a user can input the budget available for operating the application program based on known funding for a particular program. In another exemplary embodiment, the budget information can comprise a value associated with operating the application program. The value can comprise a monetary amount that the user is willing to pay to obtain the resources necessary to operate the application program. Alternatively, the value can represent a perceived value of the application program that is not based on actual monetary value. For example, the perceived

value can be based on a priority of the application program's operations to the user and/or to other users operating additional application programs within the distributed computing environment. Budget constraints can be provided in several alternative formats, such as commands to obtain the best resources available regardless of cost, to obtain the least expensive resources, to obtain resources at a specified total price or price per unit of resource, or to obtain resources via another suitable budget format.

In step 415, time periods during which the application program must operate are identified. In an exemplary embodiment, the time periods of operation can be input by a user of the application program and designed to meet the user's budgetary and customer service constraints. For example, a user can input the time periods during which the application program must operate, such as 24/7 (twenty-four hours per day, seven days a week), 24/5 (twenty-four hours per day, five days a week), 8:00 am to 5:00 pm Monday through Friday, or any other suitable time period during which the application program must operate. The user also can adjust the specified time periods of operation based on budgetary constraints. For instance, the user can specify operating the application program during off-peak time periods to reduce the cost of operating the application program.

In steps 420-435, computing resources, network resources, storage resources, and other resources required to operate the application program, respectively, are determined. According to an exemplary embodiment, steps 420-435 can comprise identifying a hardware type and/or configuration, including a specific manufacturer and/or component, as well as the capacity required for each resource. Representative resource capacities required can comprise CPU cycles for computing resources, bandwidth for network resources, and disk space and/or memory for storage resources. Steps 420-435 also can comprise identifying performance characteristics for each required resource to operate the application program within specific parameters. For example, such characteristics can comprise execution time, response time, accuracy of results (such as fault rate), availability, reliability, security, or other suitable characteristics indicating the performance of the resources.

In an exemplary embodiment, the requirements broker 114 can determine the required resources based on minimum or optimum resource requirements necessary to operate the application program as desired. In this case, the requirements broker 114 can obtain that information directly from the application program's specifications.

Alternatively, a user of the application program can input desired, configurable settings to specify an amount of one or more of the resources. In this regard, the user can input characteristics that will provide a desired level of service, which the requirements broker 114 can read in steps 420-435.

According to exemplary embodiments, service level requirements can be expressed in terms of thresholds or ranges. For example, a required response time can be established at less than 1 second, which allows a future determination of whether a resource's performance meets the service level requirement threshold. An alternative example would be that a required response time can be established in a range such as 0.5 seconds to 1.5 seconds. In that case, a resource meets that service level requirement if its response time falls within the specified range. Performance thresholds and ranges can be specified for each service level requirement.

In step 440, the method 220 determines whether to identify service level requirements for another application program. If yes, the method 220 branches back to step 405 to select another application program for which it will identify service level requirements. In this regard, the method 220 can identify service level requirements for multiple application programs. For example, service level requirements can be identified for applications utilizing one of the canonical architectures A, B, C illustrated in Figure 1. If the method 220 determines in step 440 that it will not identify service level requirements for another application program, then the method 220 branches to step 225 (Figure 2).

The specific service level requirements described with reference to steps 410-435 can depend upon the specific requirements of a particular application program and the specific requirements of a user of the application program or the owner of the distributed computing resources. Accordingly, the service level requirement may comprise all, or a subset, of the items described in steps 410-435, and additional service level requirements are within the scope of the invention.

Figure 5 is a flow chart depicting a method 225 for creating bids to obtain units of resources required to meet the service level requirements of an application program according to an exemplary embodiment of the invention, as referred to in step 225 of Figure 2. The method 225 will be described with reference to Figures 1 and 5.

In step 505, the requirements broker 114 selects a first application program for which it will create one or more requirements bids 118. And, in step 510, the requirements broker 114 selects a first resource required for the operation of the selected application program. For example, the requirements broker 114 can select one of compute, network, or storage resources required for operation of the application program.

In step 515, the requirements broker 114 reads the service level requirements for the selected resource, based on the service level requirements determined in steps 415-435 of Figure 4. Additionally, in step 520, the requirements broker 114 reads the budget information indicating the budget constraints for operating the application program, based on the budget determined in step 410 of Figure 4.

Then, in step 525, the requirements broker 114 establishes a price per unit of the selected resource that it will pay to obtain the selected resource for operation of the application program, based on the service level requirements and the available budget. In an exemplary embodiment, the requirements broker 114 can calculate a price per unit of resource based on cost data, supply, demand, or other economic data input into the requirements broker 114 by a user or obtained by the requirements broker 114 based on historical data of completed resource allocation transactions. The requirements broker 114 also can establish a price per unit based on commands to obtain the best resources available regardless of cost, to obtain the least expensive resources, to obtain resources at a specified total price or price per unit of resource, or to obtain resources via another suitable budget format, depending on the budget information and the priority of the application program.

In step 530, the requirements broker 114 generates one or more requirements bids 118 to obtain the selected resource for use by the selected application program. The requirements bids 118 can specify the resource, the service level requirements that the resource must meet, and the unit price that the user will pay to obtain the resource, based on the information obtained in steps 515-525. As discussed previously, the unit price can comprise an actual monetary value or a perceived value.

In step 535, the requirements broker 114 determines whether to generate requirements bids 118 for another resource needed to operate the application program. For example, if the requirements broker 114 has generated requirements bids 118 for only compute resources, then the requirements broker 114 can decide to generate requirements

bids 118 for network or storage resources. In that case, the method 225 branches back to step 510 to generate requirements bids 118 for another resource. If the method 225 determines in step 535 not to generate requirements bids 118 for another resource, then the method 225 branches to step 540.

Then, in step 540, the requirements broker 114 determines whether to generate requirements bids 118 for another application program. If yes, the method 225 branches back to step 505 to generate requirements bids 118 for another application program. If the method 225 determines in step 540 not to generate requirements bids 118 for another application program, then the method 225 branches to step 230 (Figure 2).

Figure 6 is a flow chart depicting a method 235 for matching resource offers 110 to requirements bids 118 to obtain virtual resources 109 for use by an application program according to an exemplary embodiment of the invention, as referred to in step 235 of Figure 2. The method 235 will be described with reference to Figures 1 and 6.

In step 602, the market exchange 112 selects an application program for which it will identify resources available to operate the selected application program. And, in step 605, the market exchange 112 selects a resource required to operate the selected application program. More specifically, if multiple resources are required for operation of the selected application program, then the method 235 selects one of those resources in step 605, thereby allowing the method 235 to match a resource offer 110 to a requirements bid 118 for the selected resource. Then, as described hereinafter, the matching steps can be repeated for other resources that are needed for operation of the application program.

In step 610, the market exchange 112 selects a requirements bid to purchase units of the selected resource for the selected application program. Then, in step 615, the market exchange 112 determines whether a resource offer exists to provide virtual resources 109 at the service level requirements and price parameters specified in the selected requirements bid. Accordingly, in step 620, the market exchange 112 determines whether such a matching offer exists. If yes, then the method 235 branches to step 630. If not, then the method 235 branches to step 625 in which the market exchange 112 allows the resource and requirements brokers 102, 114 to revise the resource offers 110 and/or the selected requirements bid until a matching offer is identified. The method 235 then proceeds to step 630.

The methods employed by the market exchange 112 to identify resource offers 110 that match a selected requirements bid can comprise any suitable format for allocating goods in an economic market. For example, the market exchange 112 can utilize methods such as a commodity market model, posted price model, tendering/contract-net model, auction model (including a Dutch auction model), monopoly/oligopoly model, and/or a bid-based proportional resource sharing model. In these exemplary embodiments, the market exchange 112 operates an economic market to efficiently allocate resources at a market clearing price and within the budget parameters specified in the selected requirements bid.

When considering the budget parameters specified in the selected requirements bid, the market exchange 112 can compare resource offers to identify the best resource offer that meets the requirements bid. For example, if multiple resource offers offer an appropriate type of resource, the market exchange 112 can identify the best resource offer under the budget constraints, such as a requirements bid's command to obtain the best resources available regardless of cost, to obtain the least expensive resources, to obtain resources at a specified total price or price per unit of resource, or to obtain resources via another suitable budget format.

In step 630, the market exchange 112 links the matching resource offer to the selected requirements bid. Then, in step 635, the market exchange 112 determines whether additional units of the selected resource are required to operate the selected application program. For example, if the matching resource offer provides only a portion of the resource required to meet the service level requirements, then the market exchange 112 can determine that additional units of the selected resource are required. If yes, then the method 235 branches back to step 610 to select another requirements bid to purchase units of the selected resource at a specified price. The newly selected requirements bid can be a revised version of the previously selected requirements bid, with a quantity of the selected resource reduced by an amount equal to the amount of resource provided by the matching resource offer.

Referring back to step 635, if additional units of the selected resource are not required, then the method 235 branches to step 640. In step 640, the market exchange 112 determines whether another resource is required to operate the selected application program. For example, if the market exchange 112 has identified matching resource offers 110 for

only compute resources, then the market exchange 112 can decide to identify matching resource offers 110 for network or storage resources needed to operate the application program. In that case, the method 235 branches back to step 605 to identify matching resource offers 110 for another resource. If the market exchange 112 determines in step 640 not to identify matching resource offers 110 for another resource, then the method 235 branches to step 645.

Then, in step 645, the market exchange 112 determines whether to match requirements bids 118 and resource offers 110 for another application program. If yes, the method 235 branches back to step 602. If not, then the method 235 branches to step 240 (Figure 2).

Figure 7 is a flow chart depicting a method 240 for completing a transaction to purchase resources required for operation of an application program based on matching resource offers 110 and requirements bids 118 according to an exemplary embodiment of the invention, as referred to in step 240 of Figure 2. The method 240 will be described with reference to Figures 1 and 7.

In step 702, the market exchange 112 selects an application program. And, in step 705, the market exchange 112 selects a requirements bid and its matching resource offer for the application program. In step 710, the requirements broker 114 commits to pay the resource broker 102 to utilize the virtual resource 109 specified in the resource offer. In step 715, the market exchange 112 accounts for the commitment to pay for use of the virtual resource 109 by debiting an account of the requirements broker 114 and crediting an account of the resource broker 102. Then, in step 720, the market exchange 112 issues a service level agreement between the requirements broker 114 and the resource broker 102 in which the resource broker 102 agrees to provide the virtual resources 109 specified in the resource offer (including a commitment to meet the service level requirements specified in the matching requirements bid) in exchange for the payment of costs, if any, specified in the resource offer.

In step 725, the market exchange 112 determines whether additional matching requirements bids 118 and resource offers 110 exist for the application program. If yes, then the method 240 branches back to step 705 to complete a transaction for another resource

required for operation of the application program. If not, then the method 240 branches to step 730.

In step 730, the market exchange 112 determines whether it will complete transactions to obtain resources for another application program. If yes, then the method 240 branches back to step 702 to select another application program. If not, then the method 240 branches to step 245 (Figure 2).

Figure 8 is a flow chart depicting a method 245 for migrating an application program's operations to purchased resources according to an exemplary embodiment of the invention, as referred to in step 245 of Figure 2. The method 245 will be described with reference to Figures 1 and 8.

In step 802, an application program is selected. In step 805, the resource broker 102 selects a virtual resource 109 that has been purchased for the selected application program. Then, in step 810, the resource broker 102 allocates the purchased virtual resource 109 to the application program, and, in step 815, the requirements broker 114 instructs the application program to utilize the allocated virtual resource 109 for operation of the application program.

In step 820, the method 245 determines whether another resource was purchased for the application program. If yes, then the method 245 branches back to step 805 to allocate another virtual resource 109 to the application program. If not, then the method 245 branches to step 825.

In step 825, the method 245 determines whether to migrate another application program's operations to purchased resources. If yes, then the method 245 branches back to step 802 to select another application program. If not, then the method 245 branches to step 250 (Figure 2).

Figure 9 is a flow chart depicting a method 250 for monitoring the performance of allocated resources and the service level requirements of an application program according to an exemplary embodiment of the invention, as referred to in step 250 of Figure 2. The method 250 will be described with reference to Figures 1 and 9.

In step 905, the requirements broker 114 selects a resource being utilized by the application program. For example, the requirements broker 114 can select one of the computing, network, or storage resources being utilized by the application program.

In step 910, the requirements broker 114 determines the application program's service level requirements specified in the requirements bid for the selected resource. In an exemplary embodiment, the requirements broker 114 can make that determination based on the service level requirements listed in the service level agreement. Then, in step 915, the requirements broker 114 determines whether new service level requirements (other than the service level requirements specified in the requirements bid for the selected resource) have been established for this resource. If yes, then the method 250 branches to step 920 to determine the application program's new service level requirements, for example, by reading the new requirements input by a user of the application program. The method 250 then proceeds to step 925. Referring back to step 915, if the requirements broker 114 determines that new service level requirements for the application program have not been established, then the method 250 can branch directly to step 925.

In step 925, the requirements broker 114 monitors the selected resource's performance as utilized by the application program. In step 930, the requirements broker 114 compares the selected resource's performance to the application program's service level requirements. Then, in step 935, the requirements broker 114 determines whether the resource's performance exceeds the application program's service level requirements. If yes, then the method branches to step 940.

In step 940, the requirements broker 114 determines whether it is paying for an excess of the resource. For example, the requirements broker 114 can determine that it is paying for an excess of the resource if the application program is operating at or near its maximum utilization of the resource and the resource has excess capacity. Alternatively, if the resource has excess capacity but the application program currently is operating below its maximum utilization of the resource, then the requirements broker 114 can determine that it is not paying for an excess of the resource. If the requirements broker 114 determines that it is paying for an excess of the resource, then the method 250 branches to step 255 (Figure 2) in which the requirements broker 114 determines that it is paying for under-utilized resources.

Referring back to step 940, if the requirements broker 114 determines that it is not paying for an excess of the resource, then the method 250 branches to step 955 to continue monitoring the selected resource.

Referring back to step 935, if the requirements broker 114 determines that the resource's performance is not exceeding the application program's service level requirements, then the method 250 branches to step 945. In step 945, the requirements broker 114 determines whether the selected resource's performance is not meeting the application program's service level requirements. If yes, then the method branches to step 950 in which the requirements broker 114 determines whether the resource is unable to meet the service level requirements. For example, the requirements broker 114 can determine that the selected resource is unable to meet the service level requirements if the application program is operating at or below its maximum utilization of the resource and the resource is not providing sufficient performance to meet the service level requirements. Alternatively, if the application program is temporarily operating beyond the utilization specified in the service level requirements, then the requirements broker 114 can determine that the resource is able to meet the service level requirements. If the requirements broker 114 determines that the resource is unable to meet the service level requirements, then the method 250 branches to step 255 (Figure 2) in which the requirements broker 114 determines that it is paying for under-performing resources.

Referring back to step 950, if the requirements broker 114 determines that it is not paying for an excess of the resource, then the method 250 branches to step 955 to continue monitoring the selected resource.

Referring back to step 945, if the requirements broker 114 determines that the resource's performance is not less than the service level requirements, then the method 250 branches to step 955 to continue monitoring the selected resource.

From step 955, the method 250 proceeds to step 960 in which the requirements broker 114 determines whether to monitor the performance of another resource being utilized by the application program. If yes, then the method 250 branches back to step 905 to select another resource for monitoring. If not, then the method branches to step 255 (Figure 2) in which the requirements broker 114 can determine that the resources utilized by the application program are not under-utilized or under-performing.

The method 250 can be performed for each application program that has been allocated resources in the distributed computing environment via service level agreements. In this manner, the performance of the allocated resources is continually or periodically

monitored to ensure that the resources meet the service level requirements. If the resources are not providing the required level of service, then the requirements broker 114 generates new requirements bids 118 and submits those bids to the market exchange 112 to obtain resources that will meet the service level requirements. Additionally, in this manner, the performance of the allocated resources is continually or periodically monitored to ensure that the performance of the resources does not exceed the service level requirements by an amount that would indicate the requirements broker 114 is paying for unused resources (in other words, excess capacity of the resource). If the resources are not being sufficiently utilized by the application program, then the requirements broker 114 generates new requirements bids 118 and submits those bids to the market exchange 112 to obtain resources that are more suitable and/or economical with regard to the service level requirements.

The method 250 illustrated in Figure 9 monitors the performance of allocated resources to determine whether those resources are under-utilized or under-performing. If so, then the method 250 returns to the method 200 illustrated in Figure 2 to perform steps 225-245. In steps 225-245 of the method 200, new resources are identified and allocated to the application program, and the application program's operations are migrated to the new resources.

Figure 10 is a block diagram depicting reallocation of distributed computing resources for an application program in the system 100 according to an exemplary embodiment of the invention. As illustrated in Figure 10, the requirements broker 114 has identified a breach in the service level agreement for currently allocated network and storage virtual resources 109. In other words, the performance of the currently allocated network and storage virtual resources 109 is not meeting the service level requirements specified the service level agreement related to those virtual resources 109. More specifically, the virtual network fabric N_2 is not sufficient to provide the necessary communication rates between allocated resources, and the requirements broker 114 needs to obtain faster network resources. Additionally, the virtual storage fabric S_2 is not sufficient to provide the necessary storage and retrieval rates, and the requirements broker 114 needs to obtain more resilient storage resources.

Accordingly, the requirements broker 114 generates new requirements bids 118 to obtain new network and storage resources that can meet the specified service level requirements and submits those bids to the market exchange 112. The market exchange 112 identifies matching resource offers 110 and completes a service level agreement to allocate new network and storage virtual resources 109 to the application program. Then, the operations of the application program are migrated to the new network and storage virtual resources 109. As illustrated in Figure 10, the application program's operations are migrated from virtual network fabric N_2 to virtual network fabric N_1 and from virtual storage fabric S_2 to virtual storage fabric S_1.

Also as illustrated in Figure 10, the requirements broker 114 has identified a breach in the budget for currently allocated compute virtual resources 109. In other words, the performance of the currently allocated virtual compute fabric C_1 is exceeding the service level requirements specified in the service level agreement related to those resources. More specifically, the requirements broker 114 is over-paying for unused compute resources and needs to obtain new resources that perform within the service level requirements, which likely will reduce the cost to operate the application program.

Accordingly, the requirements broker 114 generates one or more new requirements bids 118 to obtain new compute resources that perform within the specified service level requirements and submits those bids to the market exchange 112. The market exchange 112 identifies matching resource offers 110 and completes a service level agreement to allocate new compute virtual resources 109 to the application program. Then, the operations of the application program are migrated to the new compute virtual resources 109. As illustrated in Figure 10, the application program's operations are migrated from virtual compute fabric C_1 to virtual compute fabric C_2.

Figure 11 is a flow chart depicting a method 270 for managing the maintenance, acquisition, and/or divestment of resources based on resource allocation over a period of time according to an exemplary embodiment of the invention, as referred to in step 270 of Figure 2. The method 270 will be described with reference to Figures 1 and 11.

In step 1105, a resource manager, such as the resource broker 102, monitors revenue generated by each physical resource 103. In an exemplary embodiment, the resource manager can monitor such generated revenue by aggregating payments for service level

agreements that correspond to each particular physical resource 103. Each payment represents an amount paid by the requirements broker 114 to the resource broker 102 for utilization of a particular physical resource 103 that is included in a service level agreement. In that regard, the resource manager can maintain a running total of revenue generated by each physical resource 103 during the time period.

Then, in step 1110, the method determines whether the time period has elapsed. In exemplary embodiments, the time period can be quarterly, bi-annually, annually, or any other suitable time period for monitoring revenues generated by the physical resources 103. In another exemplary embodiment, the method 270 can determine whether the time period has elapsed based on a predetermined time period that is monitored by the resource broker 102, the expiration of which can trigger an alert that the time period has elapsed. Alternatively, the method 270 can determine whether the time period has elapsed based on a user manually accessing the generated revenue information from step 1105. In any event, if the time period has not elapsed, then the method 270 branches back to step 1105 to continue monitoring the revenue generated by each physical resource 103. If the time period has elapsed, then the method 270 branches to step 1115.

In step 1115, the method identifies costs and expenses associated with, among other things, procuring and maintaining each physical resource. In an exemplary embodiment, a user can input that information based on actual and/or projected procurement and maintenance costs. Then, in step 1120, the profit and loss of each physical resource 103 is determined by subtracting the costs and expenses associated with the physical resource 103 from the revenue generated by the physical resource 103.

In step 1125, a particular physical resource 103 is selected, and, in step 1130, it is determined whether the selected resource made a profit or loss during the time period. If the physical resource's revenue is greater than its costs and expenses, then the physical resource has generated a profit during the time period. Alternatively, if the physical resource's revenue is less than its costs and expenses, then the physical resource has generated a loss during the time period.

If the selected physical resource generated a profit, then the method 270 branches to step 1135. In step 1135, the profit of the selected physical resource is compared to the profit of other similar resources. Then, a determination is made in step 1140 whether to maintain

the current position of the selected physical resource or to invest in the selected physical resource. For example, if the resource is making only a small profit compared to other resource, then the user may decide to maintain the current position in the resource. In other words, the user will not purchase more of the resource. Alternatively, if the resource made a large profit compared to other resources, or if the demand for the resource is projected to increase, then the user may decide to invest in the resource by purchasing more of the resource or upgrading the existing resource. After determining whether to maintain or invest in the position of the selected physical resource, the method 270 proceeds to step 1150.

Referring back to step 1130, if the selected physical resource generated a loss during the time period, then the method 270 branches to step 1145. In step 1145, a determination is made whether to maintain the current position of the selected physical resource or to divest the selected physical resource. For example, if the resource experienced only a small loss, or if the resource meets a high priority need that justifies the cost, then the user may decide to maintain the current position in the resource. In other words, the user will not divest the resource. Alternatively, if the resource experienced a large, or otherwise undesirable, loss, then the user may decide to divest the resource by selling the resource or discontinuing the support or maintenance of the resource. In other embodiments, the user may decide to reduce use of the resource to reduce the loss associated with the resource, or the user may decide to subsidize continued use of the resource using profits generated by another resource. After determining whether to maintain or divest in the position of the selected physical resource, the method 270 proceeds to step 1150.

In step 1150, it is determined whether to evaluate the position of another resource. If yes, then the method 270 branches back to step 1125 to select another resource. If not, then the method 270, and the method 200 (Figure 2), ends.

Accordingly, the method 270 allows a user to evaluate physical resources to determine which resources are the most economical, including both price and performance. For example, compute resources can comprise different platforms that each has different price and performance characteristics. Comparison of the profit and loss statements of each resource will show which platforms are used most by application programs, thereby generating more revenue. Based on that information, the user can determine which

platforms to maintain or in which to increase a companies position and which platforms to divest. Because the resources have been allocated as described with reference to the method 200 (Figure 2), the method 270 provides an indication of which resources are more desirable, based on price and performance characteristics, for use by application programs.

Although the invention has been described in detail with reference to allocation and management of distributed computer resources, the invention also applies to allocation and management of other distributed resources. For example, the invention also can apply to a distributed workforce. In this case, resource offers 110 are generated to identify characteristics and prices associated with available individual or group members of the workforce, and those offers are submitted to the market exchange 112. Requirements bids 118 are generated to obtain services from the workforce for a particular work project, and those bids are submitted to the market exchange 112. Then, the market exchange 112 identifies matching bids and offers and allocates the workforce resources to the project. The performance of the allocated resources can be monitored and the resources can be reallocated as necessary to correct for under-utilized or under-performing members of the workforce. Over time, the profit and loss of individual or group members of the workforce can be determined, and the profit and loss information can be used to determine investment or divestment decisions regarding particular aspects of the workforce.

Figure 12 is a flowchart depicting a method 1200 for managing resource requirements based on a service level agreement for an application program according to an exemplary embodiment of the invention. The method 1200 will be described with reference to Figures 12 and 15, where Figure 15 is a block diagram depicting a system 1500 for managing resource requirements based on a service level agreement for an application program according to an exemplary embodiment of the invention.

In step 1205, a service level agreement is created to define the service level requirements for each resource to be utilized by the application program. In an exemplary embodiment, the required resources can comprise computing, network, and storage resources, such as one or more of the compute fabrics 104, network fabrics 106, and storage fabrics 108 illustrated in Figure 15. The service level agreement defines the unit amount of each resource that is required to operate the application program in a manner that is suitable to a user of the application program. The defined unit amounts are the corresponding

service level requirements for each resource. The service level requirements can include a minimum acceptable amount of each resource, a maximum amount of each resource, and/or a range of acceptable amounts of each resource.

In step 1210, thresholds are configured for each service level requirement for each resource above which action is initiated to increase an allocated amount of resource(s) to the application program. Specifically, for each resource, a threshold is established below the service level requirement for each resource. If appropriate or desired, the threshold can be established above a service level requirement depending on the type of resource and how utilization of the resource is measured. Regardless of whether the threshold is above or below the service level requirement, the threshold provides a benchmark to identify when utilization of a resource nears the service level requirement. The threshold represents the utilization of a resource at which additional amounts of the resource or alternative sources of the resource will be allocated, thereby taking action to allocate sufficient amounts of the resource prior to breaching the service level agreement. For example, a service level agreement may require between 90 and 100 Megabits/second ("Mbits/s") data transfer rate for network resources. An available network resource may have an available capacity of 1000 Mbits/s, and 90 Mbits/s of the network resource may be allocated to the application program. The threshold established for the network resource can be an amount below 90 Mbits/s. For example, the threshold can be 85 Mbits/s, 80 Mbits/s, or another suitable value less than the amount of network resource that is allocated to the application program. In an exemplary embodiment, the threshold can be established as a percentage of the allocated amount of the network resource, for example, 95%, 90%, 85%, 80%, or other suitable percentage. In the example provided above where the threshold is set at 85 Mbits/s, 85 Mbits/s represents the utilization of the network resource by the application program beyond which action will be taken to allocate additional amounts of network resources to the application program up to the service level requirement of 100 Mbits/s. For example, 100 Mbits/s of the network resource can be allocated to the application program prior to the utilization of the network resource exceeding the previously allocated 90 Mbits/s. In this embodiment, the threshold is based on the allocated amount of a resource where the allocated amount is less than the service level requirement for that resource. Alternatively, the allocated amount of the resource can equal the service level requirement for the resource,

and the threshold can represent the point at which the service level requirement is increased and the allocation of the resource is increased to the new service level requirement. For example, the service level requirement may be temporarily increased to accommodate a spike in utilization. Additionally, the service level requirement and the allocated amount of resources can be decreased after the spike in utilization has ceased.

The thresholds for each resource are stored in a quality of service "QOS" manager 1504 in the system 1500. In an exemplary embodiment, the thresholds can be preconfigured in the QOS manager 1504. Alternatively, a user can configure the thresholds by inputting the desired thresholds into the QOS manager 1504 via an input device (not shown).

In an exemplary embodiment, the QOS manager 1504 can comprise a software module operating in the distributed computing environment and functioning as described herein.

In step 1215, a resource allocation manager 1506 makes an initial allocation of resources to the application program. In an exemplary embodiment, the resource allocation manager 1506 can comprise the market exchange 112 described previously herein and can allocate resources in connection with the requirements broker 114 and the resource broker 102, as described previously with reference to Figures 1-11. Alternatively, the resource allocation manager 1506 can allocate resources in another manner without using the market economic process described with reference to the market exchange 112. For example, a user can define the initial allocation of resources to the application program via the resource allocation manager 1506, or the resource allocation manager 1506 can allocate available resources based on a defined priority of application programs or a defined initialization timeline of allocated resources (a "first come, first served" model). The resource allocation manager 1506 can allocate an amount of each resource for utilization by the application program in any suitable amount sufficient to meet the service level requirement defined in the service level agreement for the application program.

For example, the resource allocation manager 1506 can select a compute, network, or storage resource, such as the compute fabrics 104, the network fabrics 106, and the storage fabrics 108. In an exemplary embodiment, based on input from a monitoring module 1502 described in further detail hereinafter, the resource allocation manager 1506 can identify

excess capacity of each resource that is not being utilized. Such excess capacity can be identified as resources that are available for use by an application program.

The resource allocation manager 1506 identifies an amount of each physical resource 103 that is available for use by an application program. In an exemplary embodiment, the amount of each resource can comprise a hardware type and/or configuration, including a specific manufacturer and/or component, as well as the excess capacity currently available for each resource. In this regard, available physical resources 103 can be combined to create virtual resources 109, such as the virtual compute fabrics C_1, C_2, the virtual network fabrics N_1, N_2, and the virtual storage fabrics S_1, S_2. For example, computer processors available at distributed locations can be aggregated to create a virtual computing resource that is available for use by an application program. Representative resource capacities can comprise CPU cycles for computing resources, bandwidth for network resources, and disk space and/or memory for storage resources.

The amount of resource available also can comprise performance and reliability characteristics associated with each virtual resource 109. For example, such characteristics can comprise execution time, response time, accuracy of results (such as fault rate), availability, reliability, security, or other suitable characteristics indicating the performance of the virtual resources 109.

The resource allocation manager 1506 identifies a unit amount of each virtual resource 109 to allocate for use by the application program. In this regard, the resource allocation manager 1506 can identify portions of a virtual resource 109 that are available for allocation in increments of the identified unit up to the maximum amount of the virtual resource 109 that is available. If the resource allocation manager 1506 identified multiple virtual resources 109, such as the virtual compute fabrics C_1, C_2, then the resource allocation manager 1506 can identify a unit amount for each virtual resource 109.

In an exemplary embodiment, the resource allocation manager 1506 can comprise a software module operating in the distributed computing environment and functioning as described herein.

After the initial allocation of resources to the application program, the allocated resources are monitored to ensure that the allocated resources meet the requirements of the service level agreement for the application program. Accordingly, in step 1220, a resource

being utilized by the application program is selected. In step 1225, the threshold associated with the selected resource for the application program is identified from the thresholds established in step 1210. And, in step 1230, the monitoring module 1502 monitors the selected resource's performance as utilized by the application program. The monitoring module 1502 communicates the selected resource's performance to the QOS manager 1504.

In an exemplary embodiment, the monitoring module 1502 can comprise a software module operating in the distributed computing environment and functioning as described herein. In certain exemplary embodiments, the monitoring module 1502 can comprise off-the-shelf software designed to monitor resource utilization.

In step 1235, the QOS manager 1504 compares the selected resource's utilization to the threshold for the selected resource to determine in step 1240 whether the selected resource's utilization by the application program exceeds the threshold, or alternatively, meets or exceeds the threshold (both cases are referred to herein as exceeding the threshold). If yes, then the method 1200 branches to step 1245 to modify the allocation of the selected resource for the application program prior to breach of the service level agreement associated with the application program. Step 1245 will be described in further detail hereinafter with reference to Figure 13. The method 1200 then proceeds to step 1250.

Referring back to step 1240, if the QOS manager 1504 determines that the selected resource's utilization by the application program does not exceed the threshold, then the method 1200 branches directly to step 1250.

In step 1250, the QOS manager 1504 determines whether to monitor another resource's performance for the application program. If yes, then the method 1200 branches back to step 1220 to select another resource. If not, then the method 1200 ends.

The method 1200 can be performed simultaneously or linearly for multiple application programs and for multiple resources being utilized by or available to one or more of the application programs to continuously monitor and maintain allocated resources for multiple application programs.

Figure 13 is a flowchart depicting a method 1245 for modifying the allocation of a selected resource for an application program prior to breach of a service level agreement associated with the application program according to an exemplary embodiment, as referred

to in step 1245 of Figure 12. The method 1245 will be described with reference to Figures 13 and 15.

The method 1245 is performed when the utilization of a resource exceeds the threshold established for that resource for the application program. Because the threshold is established at a level below the allocated amount of the resource to the application program, step 1245 can be performed to modify the resource allocation prior to the utilization exceeding the allocated amount of the resource. In this manner, the methods and systems described herein can anticipate or predict an upcoming breach of a service level agreement and can take action prior to the breach to allocate adequate resources to the application program.

Referring to Figure 13, in step 1305, a resource allocation manager 1506 determines whether the selected resource has additional, unallocated capacity that can be allocated to the application program. If yes, then the method 1245 branches to step 1310 in which the resource allocation manager 1506 allocates additional capacity of the selected resource to the application program. If the initial allocation of the selected resource was less than the service level requirement for the resource, then additional amounts of the selected resource can be allocated to the application program up to the service level requirement. Alternatively, if the initial allocation of the selected resource equaled the service level requirement for the resource, then the service level agreement can be automatically adjusted to create a higher service level requirement, and additional amounts of the selected resource can be allocated to the application program up to the higher service level requirement. The increase in the service level requirement can be either temporary or permanent. From step 1310, the method 1245 proceeds to step 1355 described hereinafter.

Referring back to step 1305, if additional capacity of the selected resource is not available, then the method 1245 branches to step 1315. In step 1315, the resource allocation manager 1506 determines whether a lower priority application program is utilizing the selected resource at level above the lower priority application program's minimum requirement for the resource. If yes, then the method 1245 branches to step 1318 to determine whether sufficient capacity exists for the resource to accommodate an increased capacity for the monitored application program and a reduced capacity for the lower priority application program, based on the net amount of increase and decrease in allocation for both

programs. If not, then the method 1245 branches to step 1330 described hereinafter. If yes, then the method 1245 branches to step 1320 to shift allocation of the selected resource from the lower priority application program to the higher priority application program that is being monitored. Accordingly, in step 1320, the resource allocation manager 1506 reduces the allocation of the selected resource for the lower priority application program but maintaining the minimum requirement for the lower priority application program. Then, in step 1325, the resource allocation manager 1506 increases the allocation of the selected resource for the higher priority application program being monitored. Similar to step 1310, the amount of the selected resource allocated to the monitored application program can be increased up to the existing or newly-established service level requirement.

From step 1325, the method 1245 proceeds to step 1328 in which the resource allocation manager 1506 determines whether the net utilization of the resource has changed. If not, then the method 1245 branches to step 1250 (Figure 12). If yes, then the method 1245 branches to step 1355 described hereinafter.

Referring back to step 1315, if a lower priority application program is not operating on the selected resource, then the method 1245 branches to step 1330 to migrate the application program to more suitable resources. In step 1330, the resource allocation manager 1506 determines whether the application program is part of a family of application programs, where a "family" of application programs is a group of application programs that have service level agreements that depend upon each other. If yes, then the method 1245 branches to step 1335 to migrate the family of application programs to more suitable resources. Thus, in step 1335, the resource allocation manager 1506 identifies alternative source(s) of the resource that are capable of supporting the family of application programs according to the service level requirements defined for each application program in the family of application programs.

In step 1338, the resource allocation manager 1506 determines whether the identified resources are sufficient to accommodate the family of application programs according to the service level agreements for the family of application programs, including power and cooling requirements necessary to meet the service level requirements established in the service level agreements. If not, then the method 1245 branches to step 1348 in which the resource allocation manager 1506 issues an alert notification to a system administrator. In

exemplary embodiments, the alert notification can be displayed on a monitor, sent via e-mail or text message, or communicated in any other suitable manner. Then, the system administrator may modify the service level definitions for one or more application programs, and the resources can be allocated among the application programs according to the modified service level definitions.

Referring back to step 1338, if the identified resources are sufficient, then the method 1245 branches to step 1340. In step 1340, the resource allocation manager 1506 migrates the family of application programs to the alternative resource(s) identified in step 1335. In an exemplary embodiment, the migration can be a "live" migration without interrupting the operation of any of the application programs. From step 1340, the method 1245 proceeds to step 1355 described hereinafter.

Referring back to step 1330, if the resource allocation manager 1506 determines that the application program is not part of a family of application programs, then the method 1245 branches to step 1345 to migrate only the monitored application program to more suitable resources. Thus, in step 1345, the resource allocation manager 1506 identifies alternative source(s) of the resource which are capable of supporting the application program according to the service level requirements defined for the application program.

In step 1347, the resource allocation manager 1506 determines whether the identified resources are sufficient to accommodate the application program according to the service level agreement for the application program, including power and cooling requirements necessary to meet the service level requirements established in the service level agreement. If not, then the method 1245 branches to step 1348, described previously.

If the identified resources are sufficient, then the method 1245 branches to step 1350. In step 1350, the resource allocation manager 1506 migrates the application program to the alternative resource(s) identified in step 1345. In an exemplary embodiment, the migration can be a "live" migration without interrupting the operation of the application program. The method 1245 then proceeds to step 1355.

In step 1355, a power/cooling adjustment system 1508 adjusts the cooling and power requirements of the original and alternate resources based on the change in allocated resources conducted in steps 1310, 1340, or 1350. Step 1355 will be described in further

detail hereinafter with reference to Figure 14. From step 1355, the method 1245 proceeds to step 1250 (Figure 12).

Figure 14 is a flow chart depicting a method 1355 for adjusting the cooling and power requirements of original and alternate resources based on a change in allocated resources conducted in steps 1310, 1340, or 1350 (Figure 13) according to an exemplary embodiment, as referred to in step 1355 of Figure 13. The method 1355 will be described with reference to Figures 14 and 15.

In step 1405, the resource allocation manager 1506 determines the amount of increase or decrease of the allocation of a particular resource based on the allocation of additional or alternative resources conducted in steps 1310, 1340, or 1350 (Figure 13). For example, the resource allocation manager 1506 can make this determination by subtracting the initial allocation of a particular resource from the allocation existing after the performance of the resource allocations in steps 1310, 1340, or 1350. The difference is the increase or decrease in the allocation of the particular resource. The resource allocation manager 1506 reports this information to the power/cooling adjustment system 1508.

In step 1410, the power/cooling adjustment system 1508 determines the corresponding increase or decrease in power required to operate the particular resource based on the change in the allocation of the particular resource. Then, in step 1415, the power/cooling adjustment system 1508 adjusts the power supplied to the particular resource accordingly. For example, if the CPU cycles of a computing resource are reduced, the corresponding power required to operate the computing resource may be reduced. Alternatively, if the CPU cycles of a computing resource are increased, the corresponding power required to operate the computing resource may increase. In this regard, the power/cooling adjustment system 1508 can interface with a power controller (not shown) to change the amount of current and/or voltage supplied to the computing resource. Accordingly, power can be supplied efficiently to a resource by providing only the power necessary to operate the resource based on the amount of the resource that is allocated to application programs.

Similarly, in step 1420, the power/cooling adjustment system 1508 determines the corresponding increase or decrease in cooling required to operate the particular resource based on the change in the allocation of the particular resource. Then, in step 1425, the

power/cooling adjustment system 1508 adjusts the cooling supplied to the particular resource accordingly. For example, if the CPU cycles of a computing resource are reduced, the corresponding cooling required to operate the computing resource may be reduced. Alternatively, if the CPU cycles of a computing resource are increased, the corresponding cooling required to operate the computing resource may increase. In this regard, the power/cooling adjustment system 1508 can interface with a cooling system controller (not shown) to change the amount of cooling supplied to the computing resource. Accordingly, cooling can be supplied efficiently to a resource by providing only the cooling necessary to operate the resource based on the amount of the resource that is allocated to application programs. In an exemplary embodiment, the cooling system controller can control an HVAC system in a room in which the particular resource is provided. Alternatively, the cooling system controller can control a fan (not shown) that is internal or external to the particular resource.

From step 1425, the method 1355 proceeds to step 1250 (Figure 12).

In an exemplary embodiment, the power/cooling adjustment system 1508 can comprise a software module operating in the distributed computing environment and functioning as described herein.

Accordingly, the QOS manager 1504 can anticipate or predict when a potential breach of a service level agreement may occur and can act to allocate additional or alternative resources prior to breach of the service level agreement to thereby prevent a breach of the service level agreement.

In an exemplary embodiment, the QOS manager 1504 can run proofs continuously to determine the appropriate resources for each application program based on the resources available. Proofs can be run to detect the potential need for more resources based on new applications that will use the existing resources.

In another exemplary embodiment, an artificial intelligence application can continuously monitor resource utilization. The various service level agreements can be defined as described herein, and artificial intelligence can run proofs to determine efficient, initial, alternative, or other allocations of resources based on multiple possible scenarios. The resource allocation manager, for example, can determine the available resource capacity and associated costs, if any, and can communicate that information to the artificial

intelligence application for evaluating multiple scenarios for resource allocation among various application programs.

The present invention can be used with computer hardware and software that perform the methods and processing functions described above. As will be appreciated by those skilled in the art, the systems, methods, and procedures described herein can be embodied in a programmable computer, computer executable software, or digital circuitry. The software can be stored on computer readable media. For example, computer readable media can include a floppy disk, RAM, ROM, hard disk, removable media, flash memory, memory stick, optical media, magneto-optical media, CD-ROM, etc. Digital circuitry can include integrated circuits, gate arrays, building block logic, field programmable gate arrays (FPGA), etc.

Although specific embodiments of the present invention have been described herein in detail, the description is merely for purposes of illustration. The exemplary methods described herein are merely illustrative and, in alternative embodiments of the invention, certain steps can be performed in a different order, performed in parallel with one another, or omitted entirely, and/or certain additional steps can be performed without departing from the scope and spirit of the invention. Additionally, various modifications of, and equivalent steps corresponding to, the disclosed aspects of the exemplary embodiments, in addition to those described herein, can be made by those skilled in the art without departing from the spirit and scope of the present invention defined in the following claims, the scope of which is to be accorded the broadest interpretation so as to encompass such modifications and equivalent structures.

CLAIMS

What is claimed is:

1. A computer-implemented method for allocating computing resources, comprising the steps of:

allocating an amount of a resource to an application program based on an established service level requirement for utilization of the resource by the application program;

determining on a computer whether the application program's utilization of the resource exceeds a utilization threshold, the utilization threshold being based on the established service level requirement and being different than the established service level requirement; and

changing via a computer the allocated amount of the resource in response to a determination that the application program's utilization of the resource exceeds the utilization threshold.

2. The method of claim 1, wherein the amount of the resource allocated in the allocating step equals the service level requirement for the resource.

3. The method of claim 1, wherein the amount of the resource allocated in the allocating step is less than the service level requirement for the resource.

4. The method of claim 1, wherein the changing step occurs prior to the application program's utilization of the resource exceeding the service level requirement.

5. The method of claim 1, wherein the changing step comprises allocating an additional amount of the resource to the application program, and wherein a total amount of the resource allocated to the application program is less than or equal to the service level requirement.

6. The method of claim 1, wherein the changing step comprises allocating an additional amount of the resource to the application program, and wherein a total amount of the resource allocated to the application program is greater than the service level requirement.

7. The method of claim 1, wherein the changing step comprises reducing an amount of the resource allocated to another program to obtain an additional amount of the resource that is then allocated to the application program.

8. The method of claim 1, wherein the changing step comprises migrating the application program to a different resource to obtain sufficient capacity based on the service level requirement for the application program.

9. The method of claim 1, wherein the changing step comprises migrating a family of application programs, including the application program, to a different resource to obtain sufficient capacity based on the service level requirement for the application program and to accommodate the family of application programs.

10. The method of claim 1, further comprising the step of adjusting the power or cooling requirements for the resource in response to changing the allocated amount of the resource.

11. The method of claim 1, wherein the changing step is based on a market exchange model.

12. The method of claim 1, further comprising repeating the allocating, determining, and changing steps for a plurality of resources and a plurality of application programs.

13. A computer-readable medium having computer-executable instructions for performing the computer-implemented method of Claim 1.

14. A system for allocating computing resources, comprising:

a resource allocation manager that initially allocates an amount of a resource to an application program based on an established service level requirement for utilization of the resource by the application program; and

a quality of service ("QOS") manager that determines whether the application program's utilization of the resource exceeds a utilization threshold, the utilization threshold being based on the established service level requirement and being different than the established service level requirement,

wherein the resource allocation manager further changes the allocated amount of the resource in response to a determination that the application program's utilization of the resource exceeds the utilization threshold.

15. The system of claim 14, further comprising a monitoring module that monitors the application program's utilization of the resource and communicates the application program's utilization of the resource to the QOS manager.

16. The system of claim 14, wherein the initial amount of the resource allocated to the application program equals the service level requirement for the resource.

17. The system of claim 14, wherein the initial amount of the resource allocated to the application program is less than the service level requirement for the resource.

18. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource prior to the application program's utilization of the resource exceeding the service level requirement.

19. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource by allocating an additional amount of the resource to the application program, and wherein a total amount of the resource allocated to the application program is less than or equal to the service level requirement.

20. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource by allocating an additional amount of the resource to the application program, and wherein a total amount of the resource allocated to the application program is greater than the service level requirement.

21. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource by reducing an amount of the resource allocated to another program to obtain an additional amount of the resource that is then allocated to the application program.

22. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource by migrating the application program to a different resource to obtain sufficient capacity based on the service level requirement for the application program.

23. The system of claim 14, wherein the resource allocation manager changes the allocated amount of the resource by migrating a family of application programs, including the application program, to a different resource to obtain sufficient capacity based on the service level requirement for the application program and to accommodate the family of application programs.

24. The system of claim 14, further comprising a power adjustment system that adjusts power for the resource in response to the resource allocation manager changing the allocated amount of the resource.

25. The system of claim 14, further comprising a cooling adjustment system that adjusts cooling for the resource in response to the resource allocation manager changing the allocated amount of the resource.

26. The system of claim 14, wherein the resource allocation manager comprises a market exchange that allocates the resource based on a market exchange model.

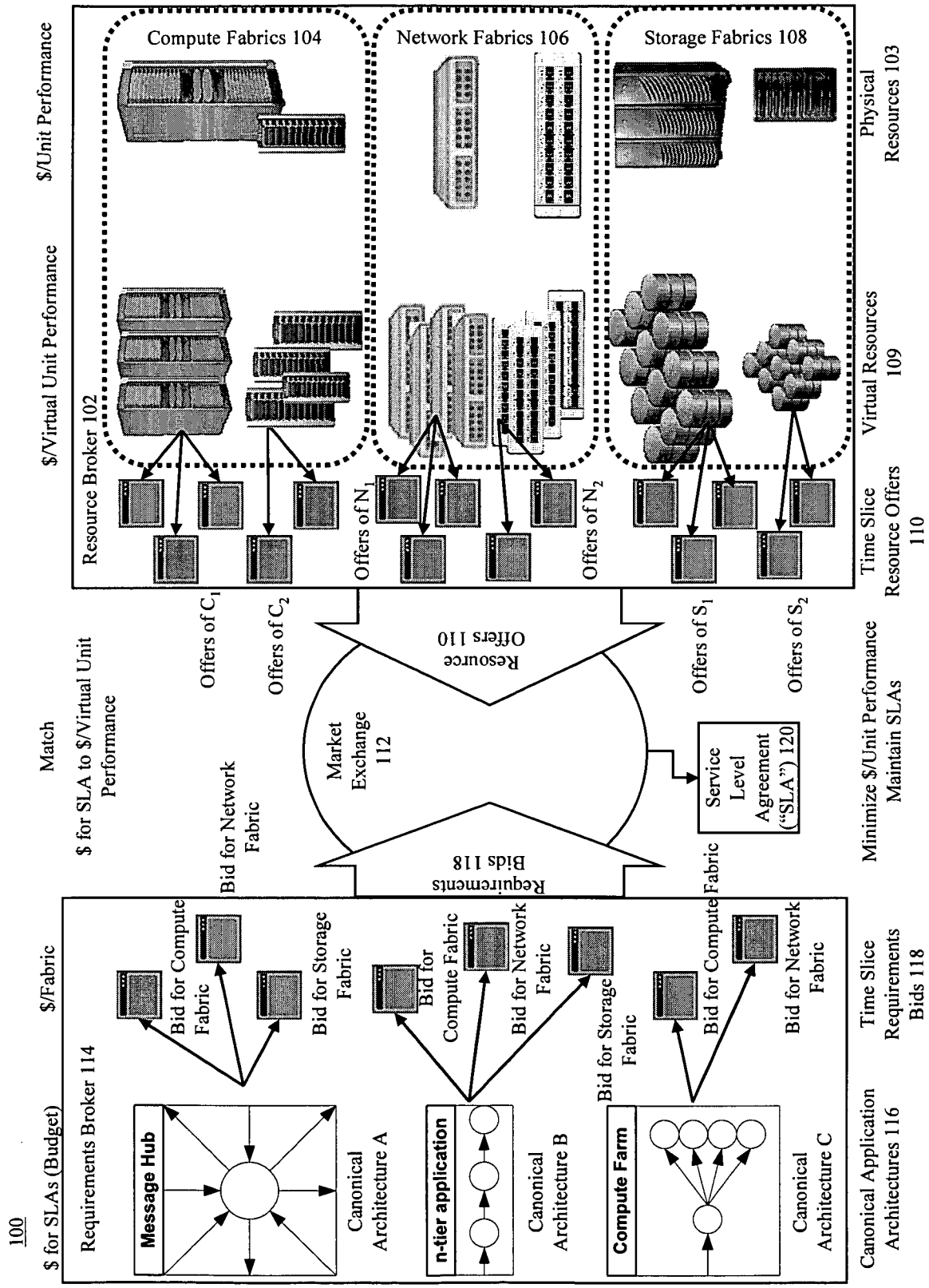

100
$ for SLAs (Budget)
$/Fabric
Requirements Broker 114
Message Hub
Bid for Compute Fabric
Bid for Storage Fabric
Canonical Architecture A
n-tier application
Bid for Compute Fabric
Bid for Network Fabric
Canonical Architecture B
Compute Farm
Bid for Storage Fabric
Bid for Compute Fabric
Bid for Network Fabric
Canonical Architecture C
Canonical Application Architectures 116
Time Slice Requirements Bids 118
Minimize $/Unit Performance
Maintain SLAs
Match
$ for SLA to $/Virtual Unit Performance
Bid for Network Fabric
Requirements Bids 118
Market Exchange 112
Resource Offers 110
Service Level Agreement ("SLA") 120
Offers of C_1
Offers of C_2
Offers of S_1
Offers of S_2
$/Virtual Unit Performance
$/Unit Performance
Resource Broker 102
Compute Fabrics 104
Network Fabrics 106
Storage Fabrics 108
Offers of N_1
Offers of N_2
Time Slice Resource Offers 110
Virtual Resources 109
Physical Resources 103

Figure 1

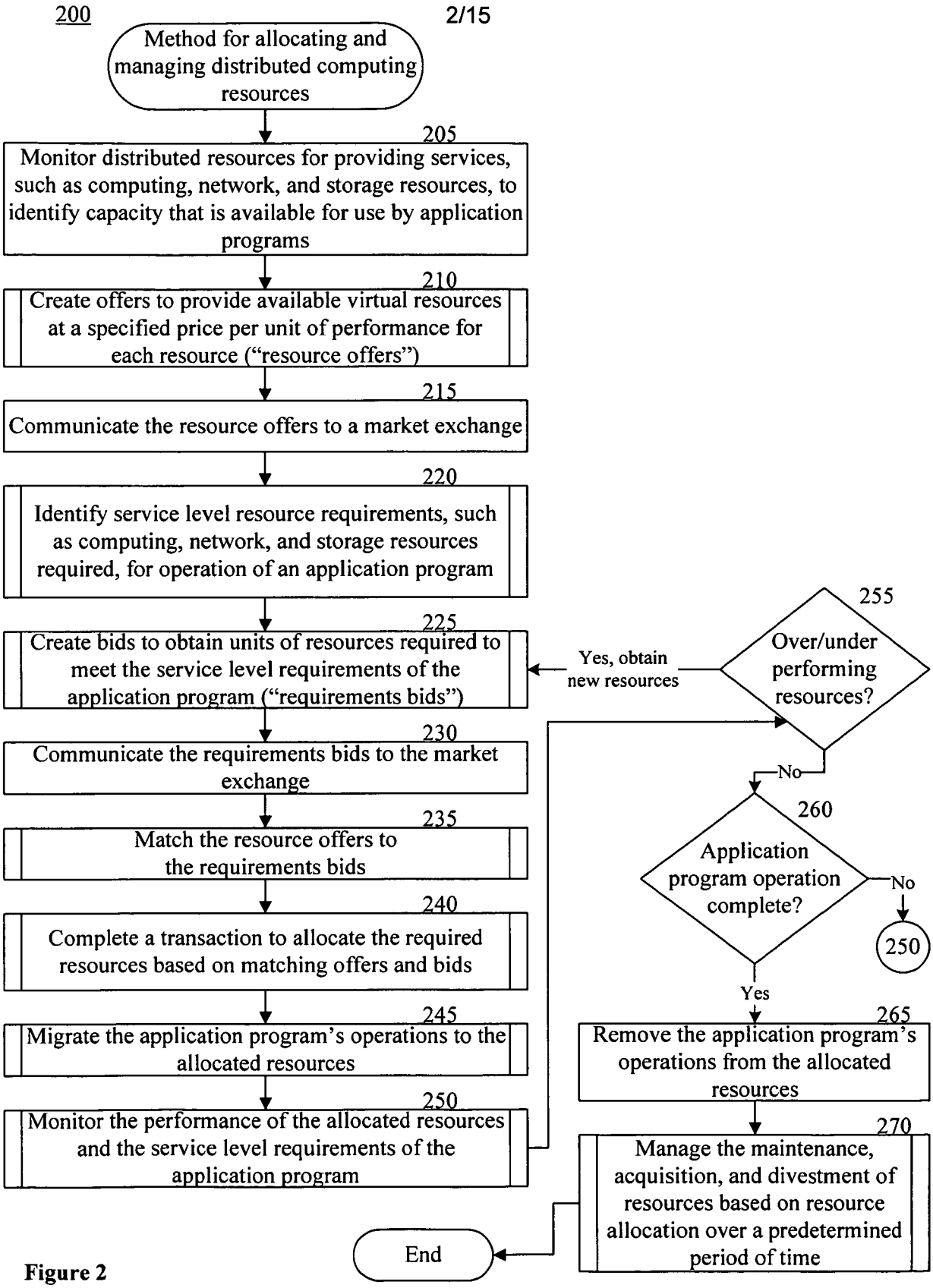

200
Method for allocating and managing distributed computing resources
205
Monitor distributed resources for providing services, such as computing, network, and storage resources, to identify capacity that is available for use by application programs
210
Create offers to provide available virtual resources at a specified price per unit of performance for each resource ("resource offers")
215
Communicate the resource offers to a market exchange
220
Identify service level resource requirements, such as computing, network, and storage resources required, for operation of an application program
225
Create bids to obtain units of resources required to meet the service level requirements of the application program ("requirements bids")
230
Communicate the requirements bids to the market exchange
235
Match the resource offers to the requirements bids
240
Complete a transaction to allocate the required resources based on matching offers and bids
245
Migrate the application program's operations to the allocated resources
250
Monitor the performance of the allocated resources and the service level requirements of the application program
255
Over/under performing resources?
Yes, obtain new resources
No
260
Application program operation complete?
No
250
Yes
265
Remove the application program's operations from the allocated resources
270
Manage the maintenance, acquisition, and divestment of resources based on resource allocation over a predetermined period of time
End

Figure 2

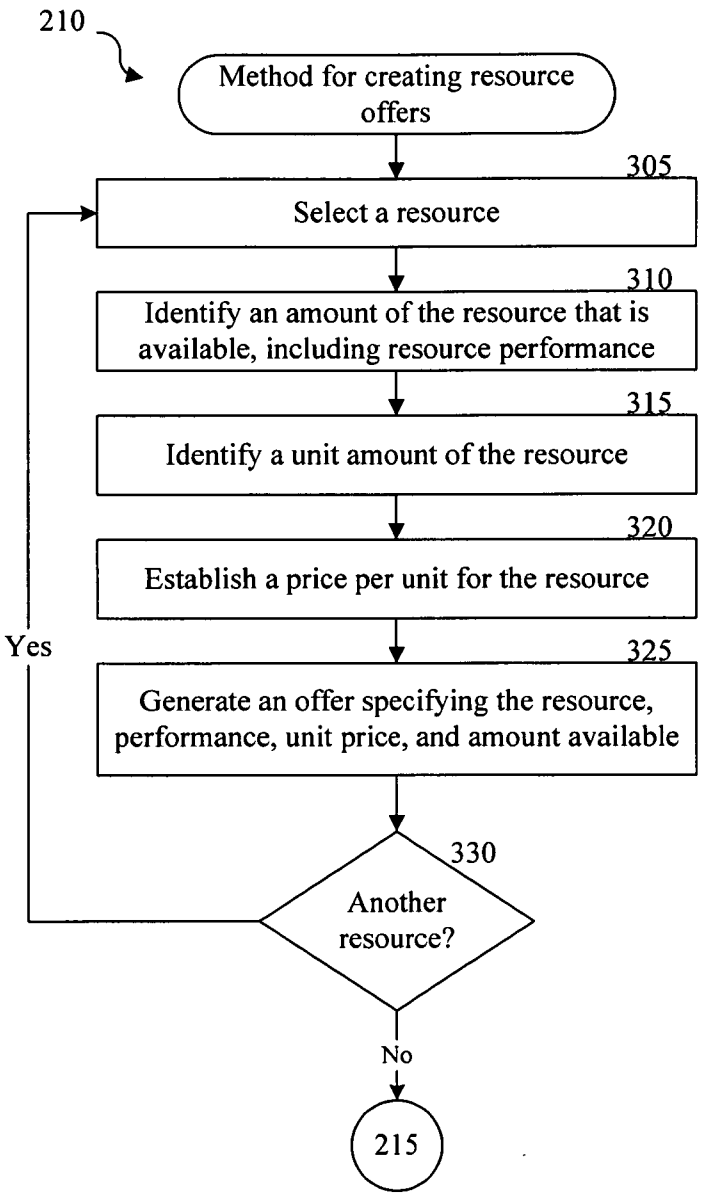

210
Method for creating resource offers
305
Select a resource
310
Identify an amount of the resource that is available, including resource performance
315
Identify a unit amount of the resource
320
Establish a price per unit for the resource
325
Generate an offer specifying the resource, performance, unit price, and amount available
330
Another resource?
Yes
No
215

Figure 3

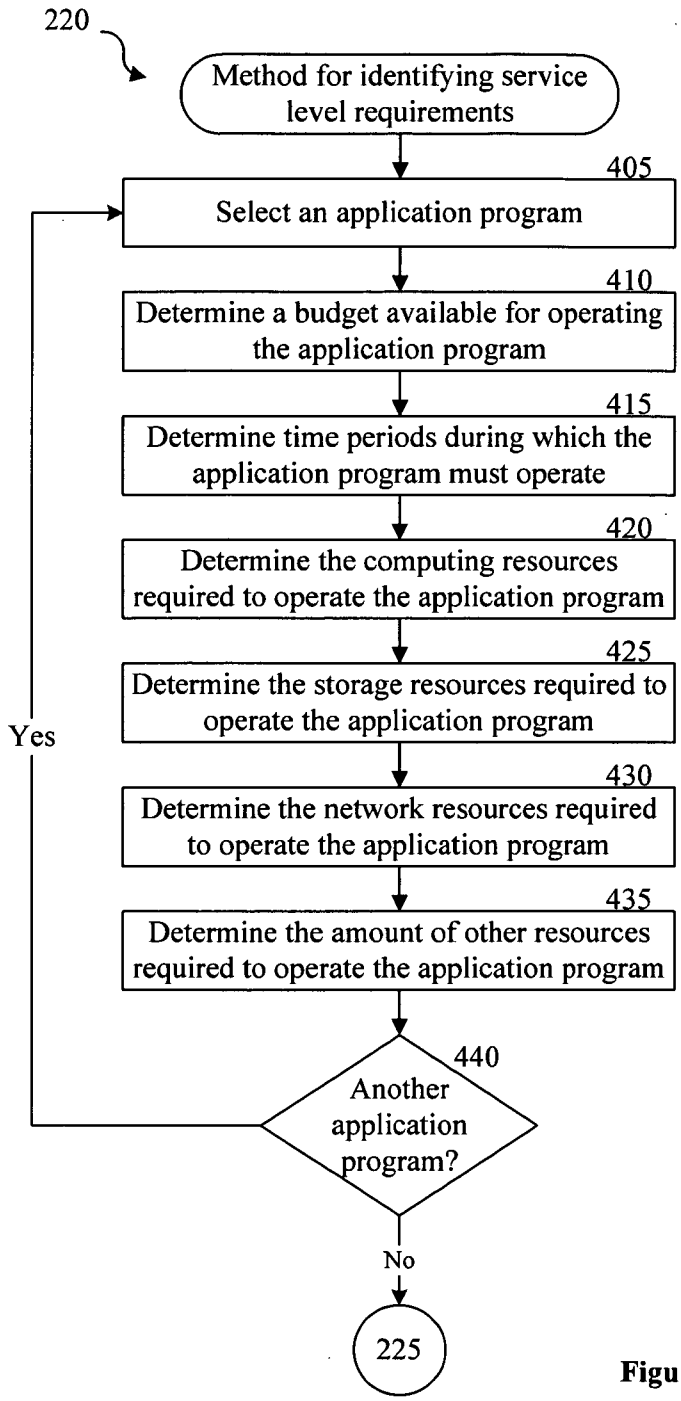

220
Method for identifying service level requirements
405
Select an application program
410
Determine a budget available for operating the application program
415
Determine time periods during which the application program must operate
420
Determine the computing resources required to operate the application program
425
Determine the storage resources required to operate the application program
430
Determine the network resources required to operate the application program
435
Determine the amount of other resources required to operate the application program
440
Another application program?
Yes
No
225

Figure 4

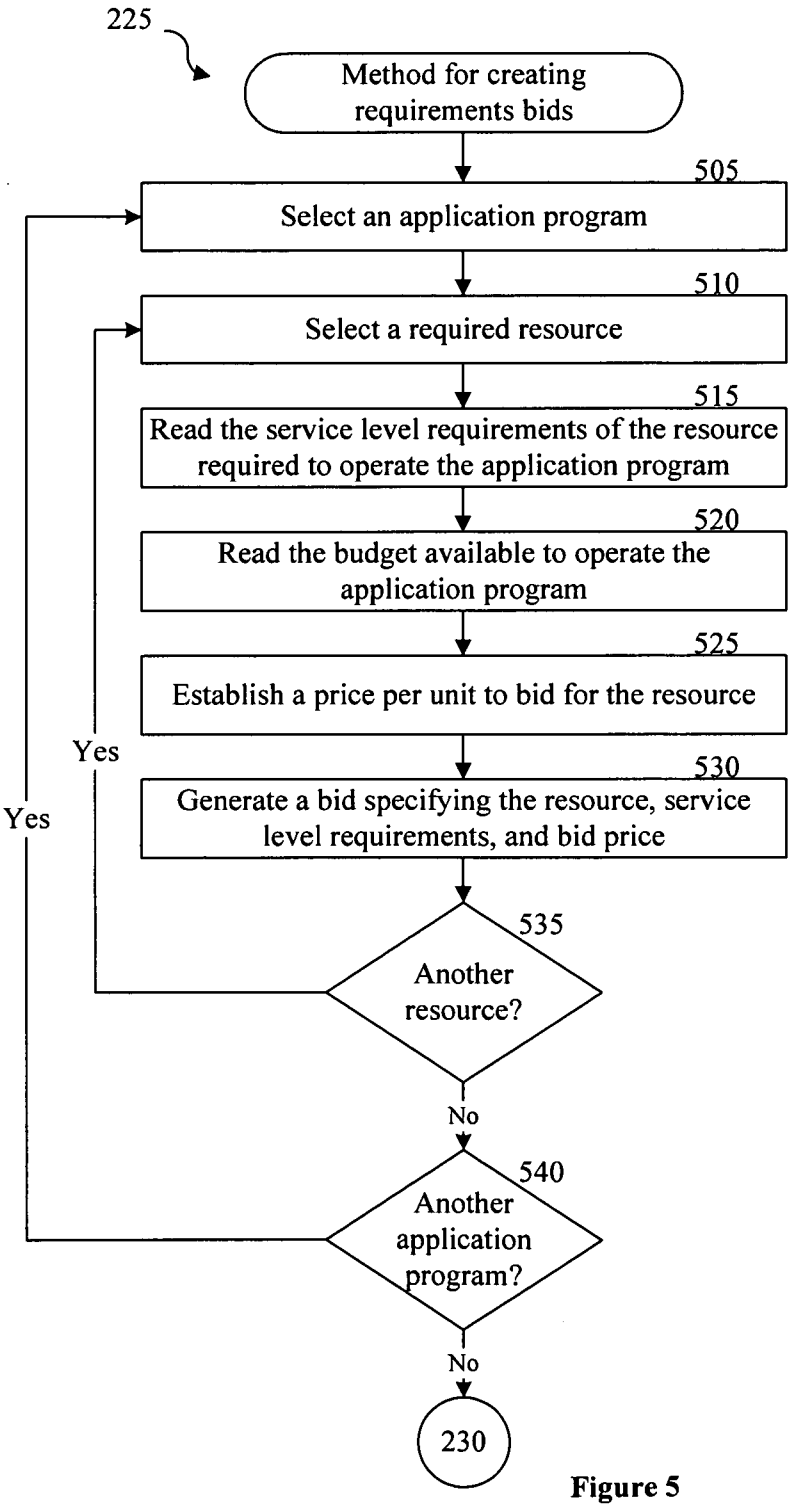

225
Method for creating requirements bids
505
Select an application program
510
Select a required resource
515
Read the service level requirements of the resource required to operate the application program
520
Read the budget available to operate the application program
525
Establish a price per unit to bid for the resource
530
Generate a bid specifying the resource, service level requirements, and bid price
535
Another resource?
Yes
No
540
Another application program?
Yes
No
230

Figure 5

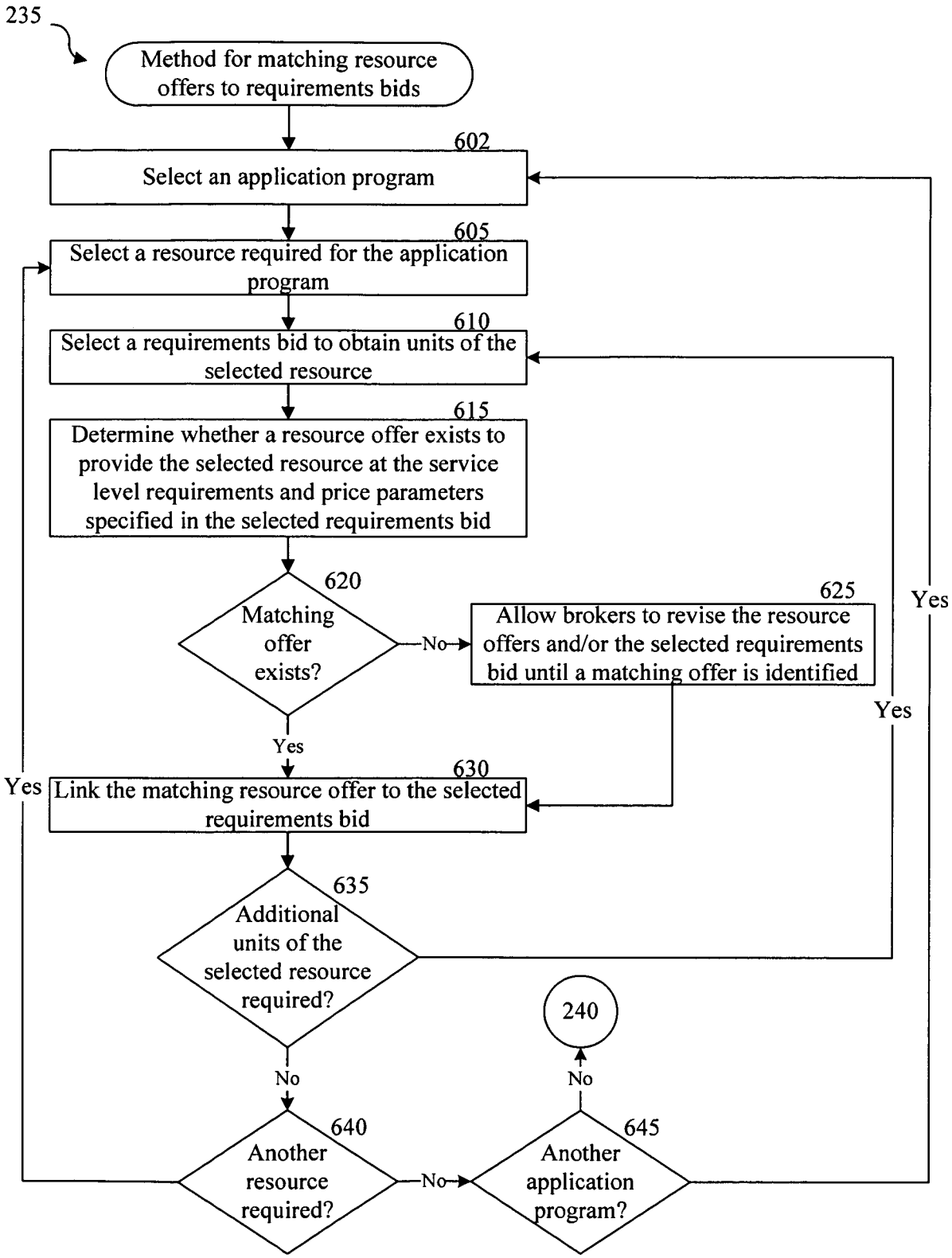

235
Method for matching resource offers to requirements bids
602
Select an application program
605
Select a resource required for the application program
610
Select a requirements bid to obtain units of the selected resource
615
Determine whether a resource offer exists to provide the selected resource at the service level requirements and price parameters specified in the selected requirements bid
620
Matching offer exists?
No
625
Allow brokers to revise the resource offers and/or the selected requirements bid until a matching offer is identified
Yes
Yes
630
Link the matching resource offer to the selected requirements bid
Yes
635
Additional units of the selected resource required?
Yes
240
No
No
640
Another resource required?
No
645
Another application program?

Figure 6

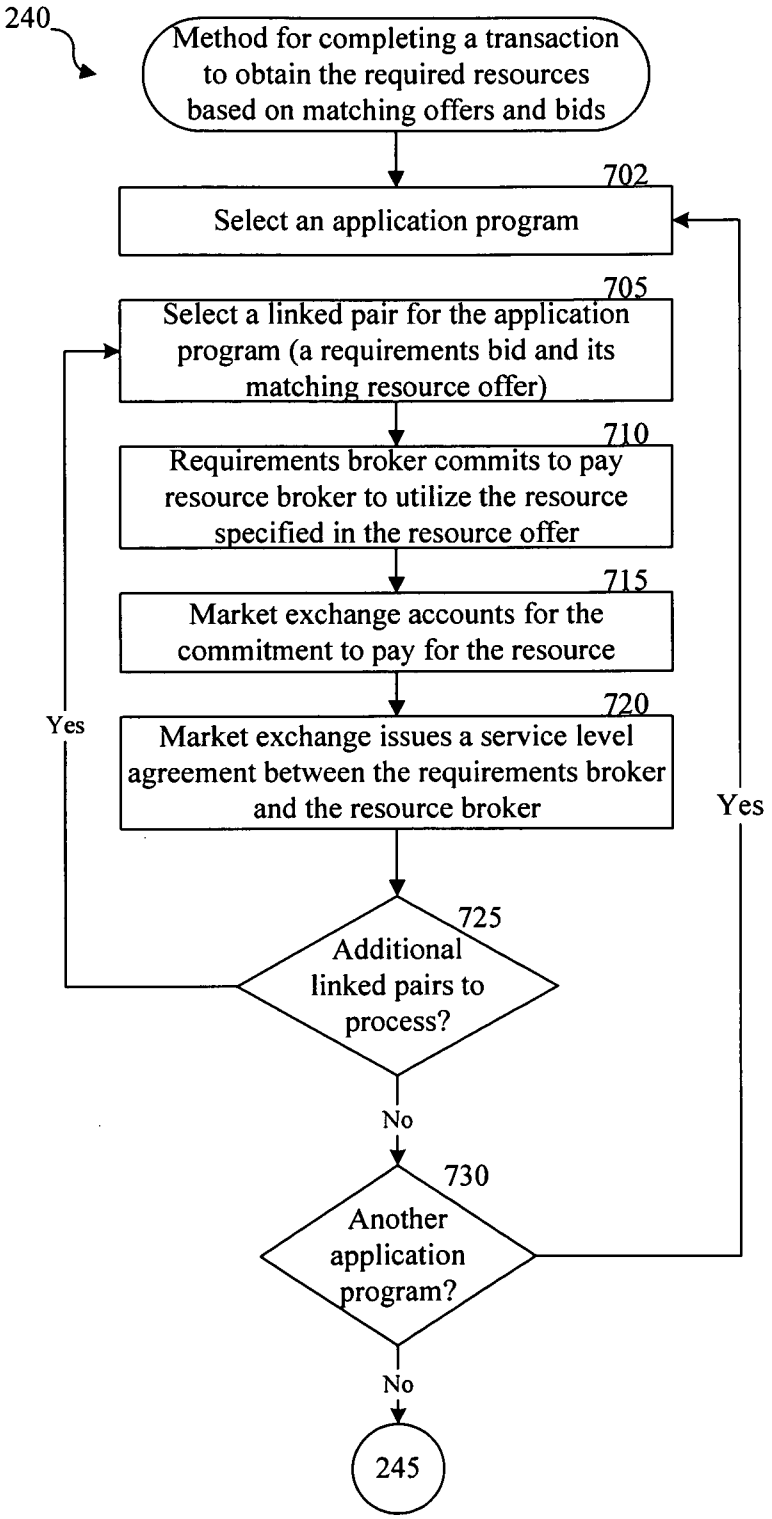

240
Method for completing a transaction to obtain the required resources based on matching offers and bids
702
Select an application program
705
Select a linked pair for the application program (a requirements bid and its matching resource offer)
710
Requirements broker commits to pay resource broker to utilize the resource specified in the resource offer
715
Market exchange accounts for the commitment to pay for the resource
720
Market exchange issues a service level agreement between the requirements broker and the resource broker
725
Additional linked pairs to process?
Yes
No
730
Another application program?
Yes
No
245

Figure 7

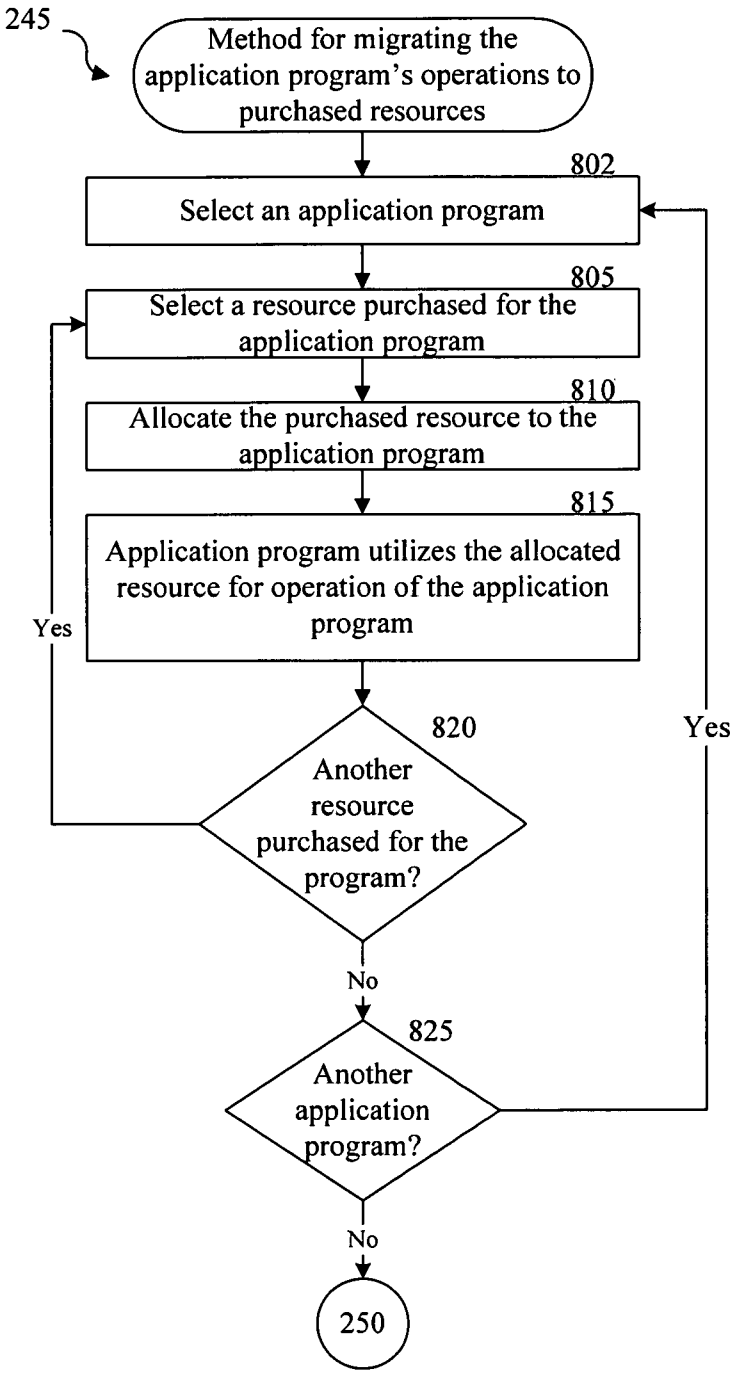

245
Method for migrating the application program's operations to purchased resources
802
Select an application program
805
Select a resource purchased for the application program
810
Allocate the purchased resource to the application program
815
Application program utilizes the allocated resource for operation of the application program
820
Another resource purchased for the program?
Yes
No
825
Another application program?
Yes
No
250

Figure 8

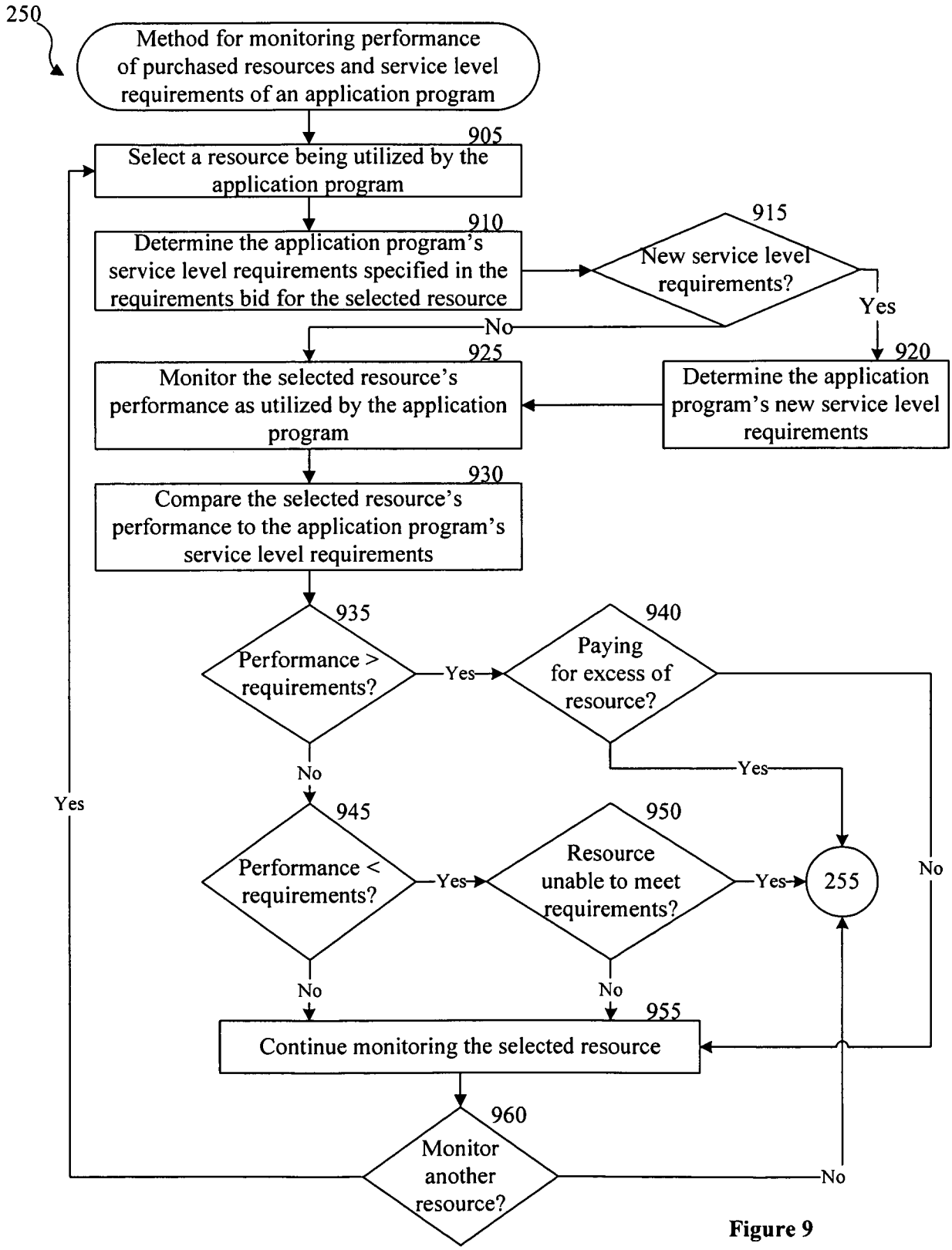

250
Method for monitoring performance of purchased resources and service level requirements of an application program
905
Select a resource being utilized by the application program
910
Determine the application program's service level requirements specified in the requirements bid for the selected resource
915
New service level requirements?
Yes
No
920
Determine the application program's new service level requirements
925
Monitor the selected resource's performance as utilized by the application program
930
Compare the selected resource's performance to the application program's service level requirements
935
Performance > requirements?
Yes
940
Paying for excess of resource?
Yes
No
945
Performance < requirements?
Yes
950
Resource unable to meet requirements?
Yes
255
No
No
No
955
Continue monitoring the selected resource
960
Monitor another resource?
Yes
No

Figure 9

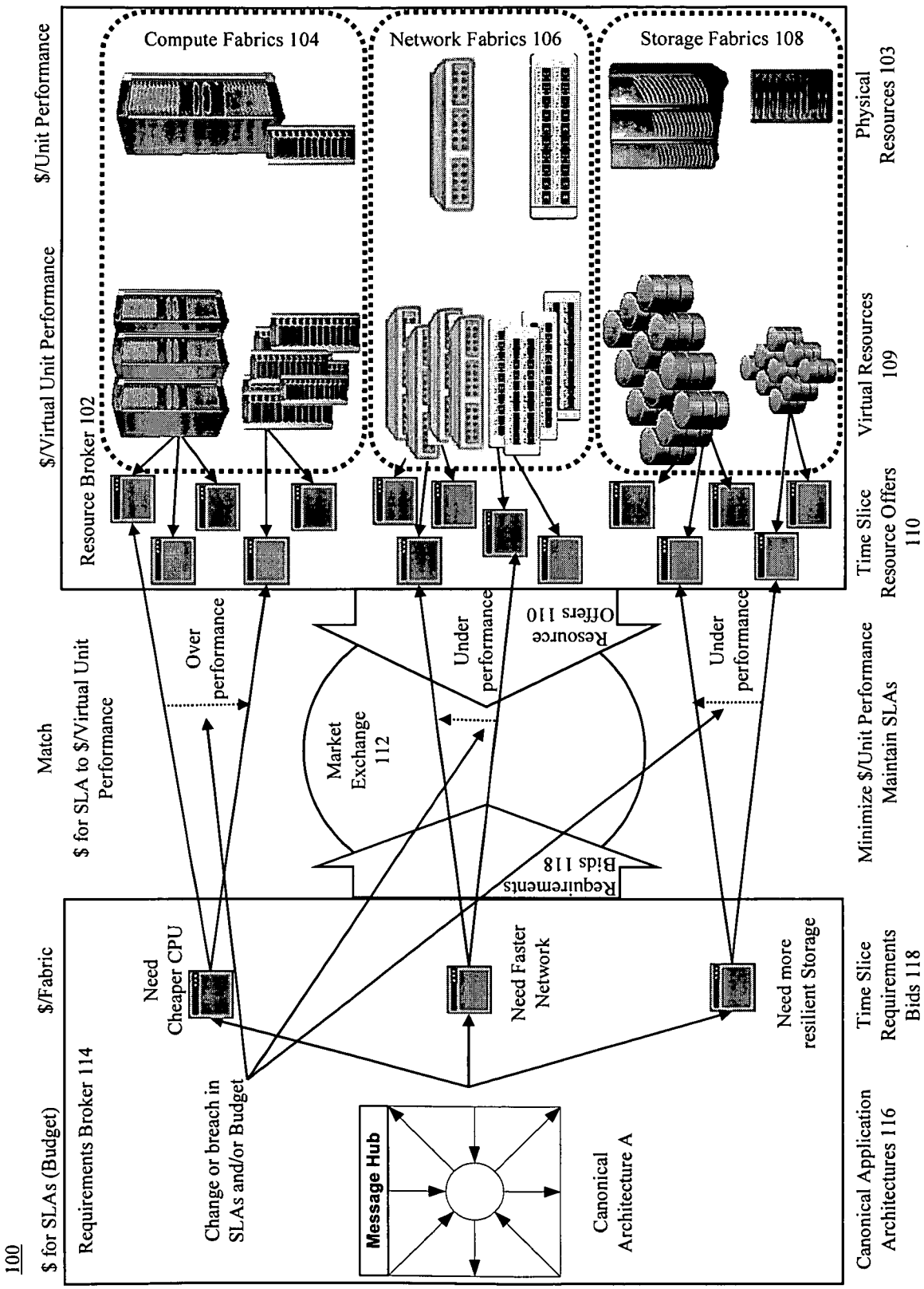

100
$ for SLAs (Budget)
$/Fabric
$/Virtual Unit Performance
$/Unit Performance
Requirements Broker 114
Resource Broker 102
Compute Fabrics 104
Network Fabrics 106
Storage Fabrics 108
Physical Resources 103
Virtual Resources 109
Time Slice Resource Offers 110
Time Slice Requirements Bids 118
Canonical Application Architectures 116
Match $ for SLA to $/Virtual Unit Performance
Minimize $/Unit Performance Maintain SLAs
Market Exchange 112
Resource Offers 110
Requirements Bids 118
Over performance
Under performance
Under performance
Need Cheaper CPU
Need Faster Network
Need more resilient Storage
Change or breach in SLAs and/or Budget
Message Hub
Canonical Architecture A

Figure 10

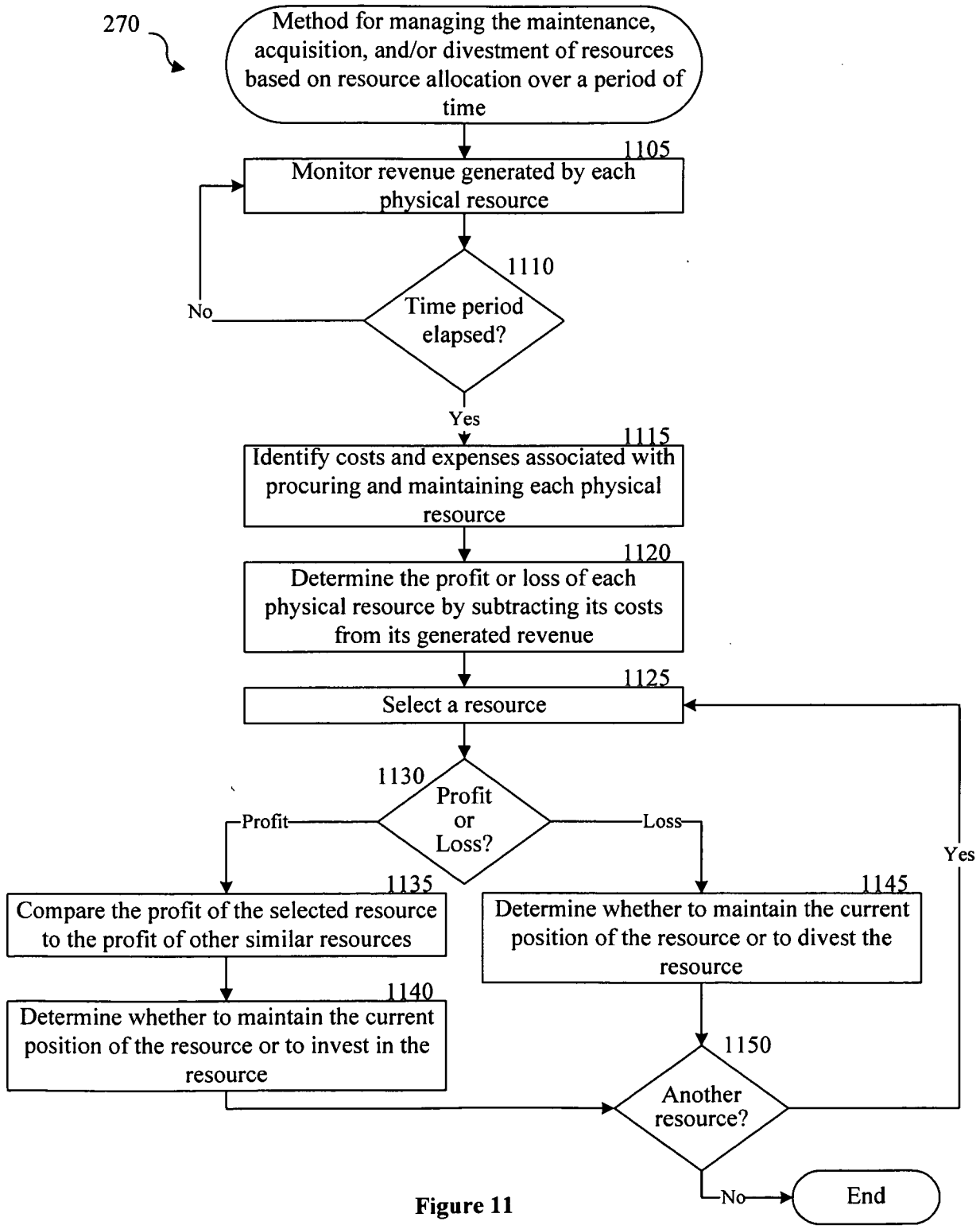

270
Method for managing the maintenance, acquisition, and/or divestment of resources based on resource allocation over a period of time
1105
Monitor revenue generated by each physical resource
1110
Time period elapsed?
No
Yes
1115
Identify costs and expenses associated with procuring and maintaining each physical resource
1120
Determine the profit or loss of each physical resource by subtracting its costs from its generated revenue
1125
Select a resource
1130
Profit or Loss?
Profit
Loss
1135
Compare the profit of the selected resource to the profit of other similar resources
1140
Determine whether to maintain the current position of the resource or to invest in the resource
1145
Determine whether to maintain the current position of the resource or to divest the resource
1150
Another resource?
Yes
No
End

Figure 11

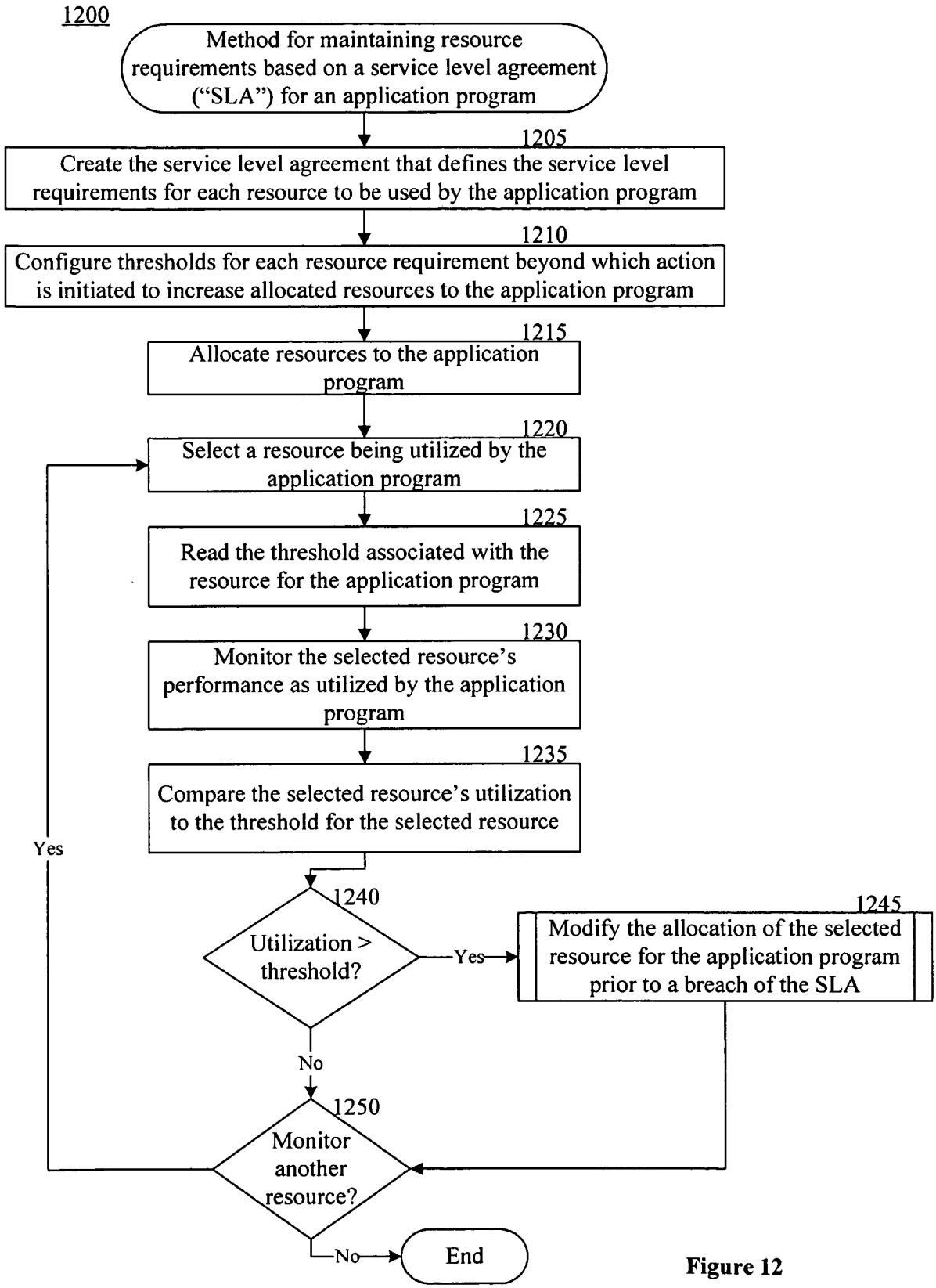

1200
Method for maintaining resource requirements based on a service level agreement ("SLA") for an application program
1205
Create the service level agreement that defines the service level requirements for each resource to be used by the application program
1210
Configure thresholds for each resource requirement beyond which action is initiated to increase allocated resources to the application program
1215
Allocate resources to the application program
1220
Select a resource being utilized by the application program
1225
Read the threshold associated with the resource for the application program
1230
Monitor the selected resource's performance as utilized by the application program
1235
Compare the selected resource's utilization to the threshold for the selected resource
1240
Utilization > threshold?
Yes
1245
Modify the allocation of the selected resource for the application program prior to a breach of the SLA
No
1250
Monitor another resource?
Yes
No
End

Figure 12

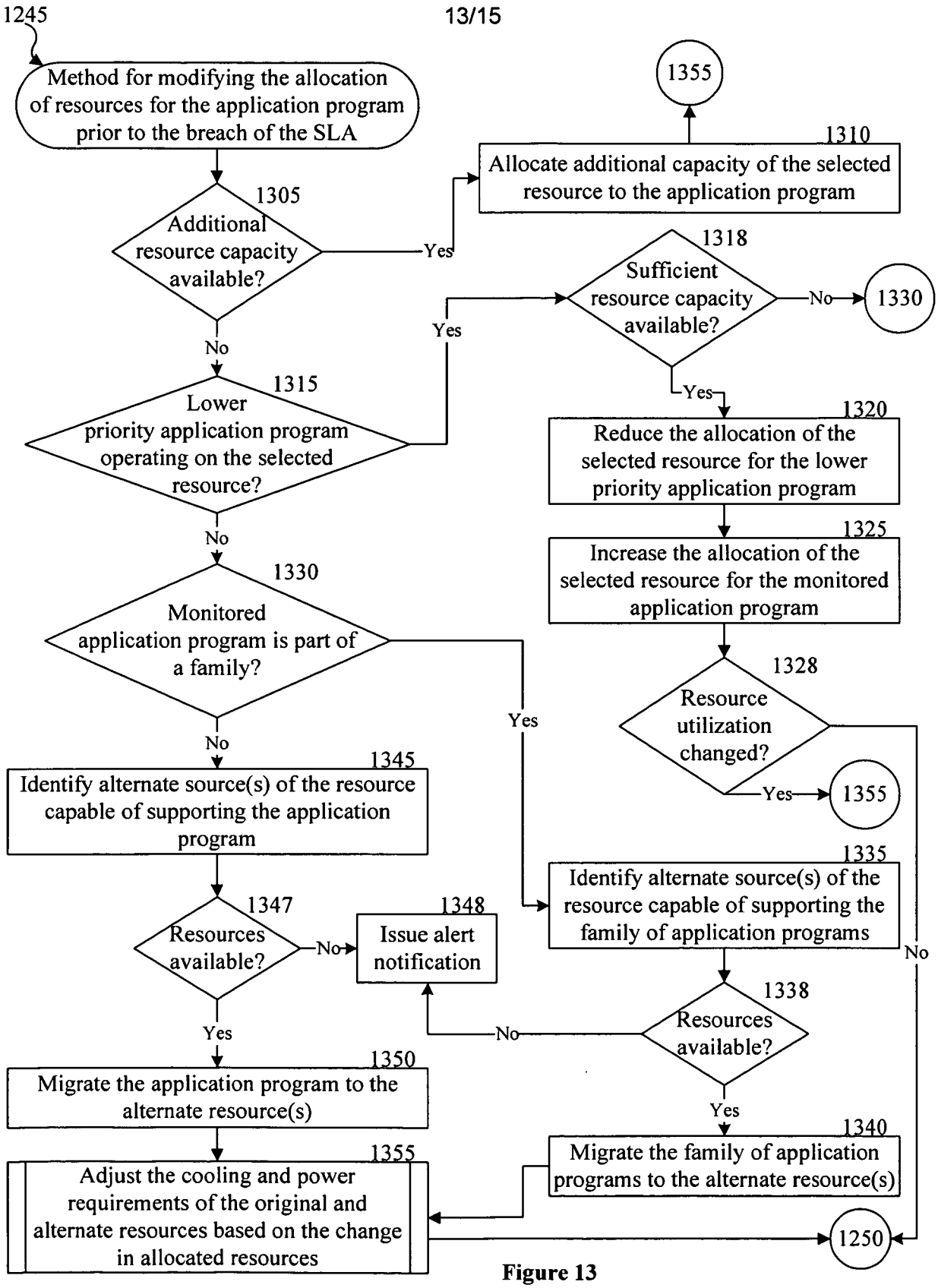

1245
Method for modifying the allocation of resources for the application program prior to the breach of the SLA
1305
Additional resource capacity available?
Yes
No
1310
Allocate additional capacity of the selected resource to the application program
1355
1315
Lower priority application program operating on the selected resource?
Yes
No
1318
Sufficient resource capacity available?
No
1330
Yes
1320
Reduce the allocation of the selected resource for the lower priority application program
1325
Increase the allocation of the selected resource for the monitored application program
1328
Resource utilization changed?
Yes
1355
No
1330
Monitored application program is part of a family?
Yes
No
1345
Identify alternate source(s) of the resource capable of supporting the application program
1335
Identify alternate source(s) of the resource capable of supporting the family of application programs
1347
Resources available?
No
1348
Issue alert notification
Yes
1338
Resources available?
No
Yes
1340
Migrate the family of application programs to the alternate resource(s)
1350
Migrate the application program to the alternate resource(s)
1355
Adjust the cooling and power requirements of the original and alternate resources based on the change in allocated resources
1250

Figure 13

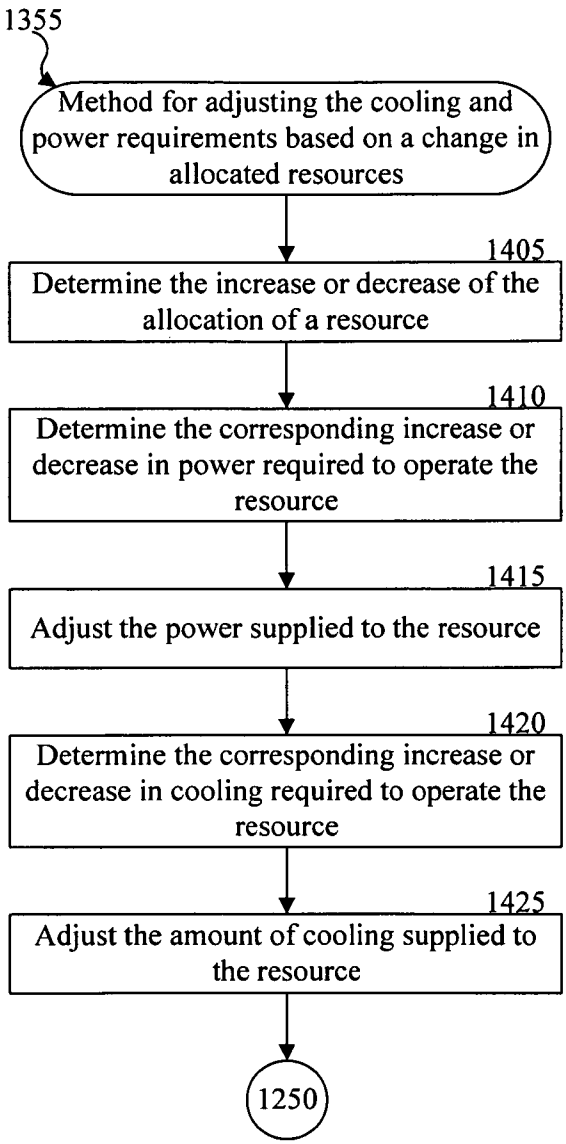

1355
Method for adjusting the cooling and power requirements based on a change in allocated resources
1405
Determine the increase or decrease of the allocation of a resource
1410
Determine the corresponding increase or decrease in power required to operate the resource
1415
Adjust the power supplied to the resource
1420
Determine the corresponding increase or decrease in cooling required to operate the resource
1425
Adjust the amount of cooling supplied to the resource
1250

Figure 14

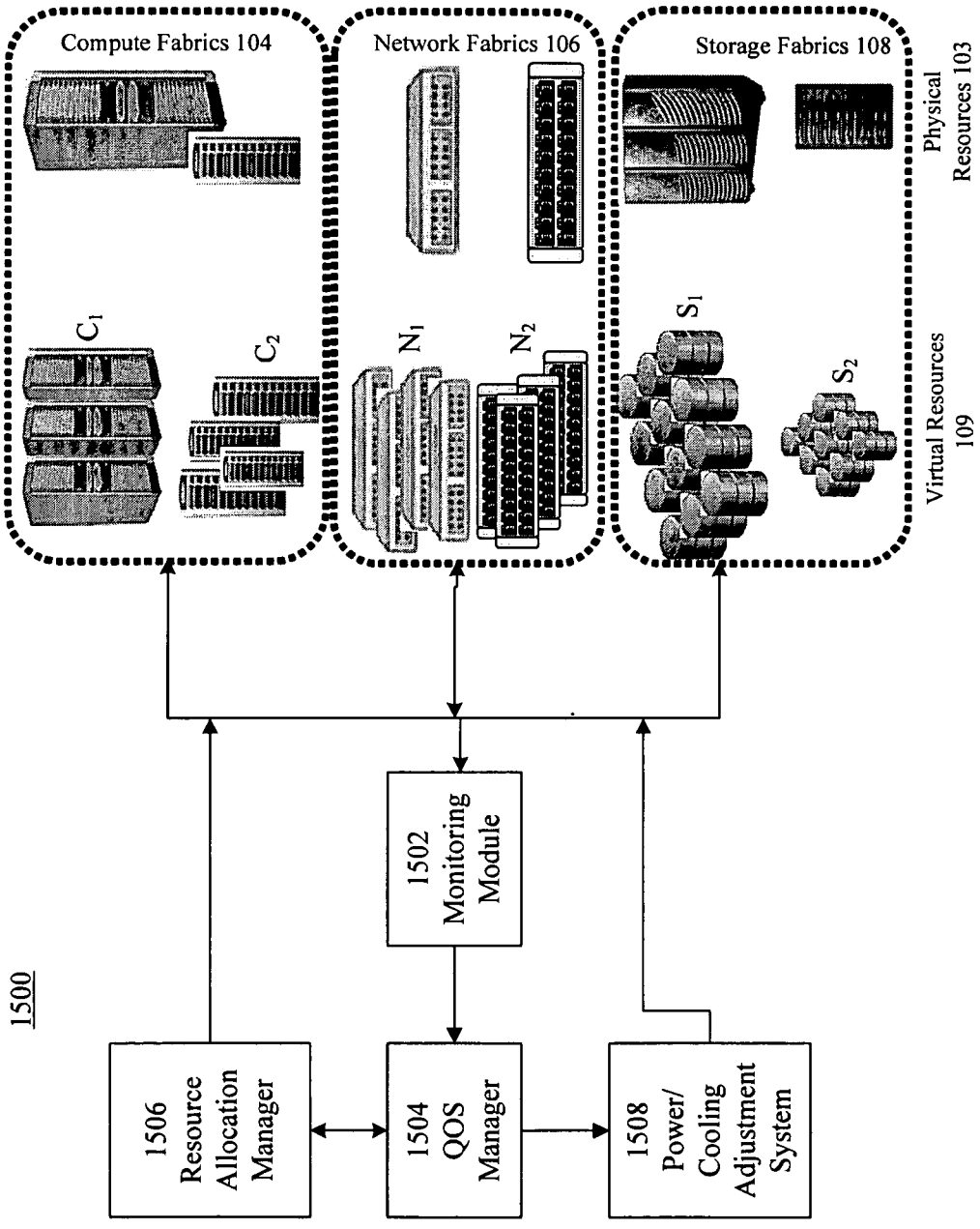

Compute Fabrics 104
Network Fabrics 106
Storage Fabrics 108
Physical Resources 103
Virtual Resources 109
C1
C2
N1
N2
S1
S2
1500
1502 Monitoring Module
1506 Resource Allocation Manager
1504 QOS Manager
1508 Power/ Cooling Adjustment System

Figure 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 08/12514

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 15/173 (2008.04)
USPC - 709/226
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)
USPC: 709/226

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC:709/201, 203, 223, 226, 229; 700/90, 91, 99

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWEST(PGPB, USPT, USOC, EPAB, JPAB); GOOGLE
Search Terms Used: Allocating or reallocating, resources, utilization, dynamically, threshold, market exchange, power, cooling.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2005/0044228 A1 (Birkestrand et al.) 24 February 2005 (24.02.2005), entire document, especially; abstract, para. [0003], [0006], [0032], [0045], [0052], [0066].	1-26
Y	US 2006/0123217 A1 (Burdick et al.) 08 June 2006 (08.06.2006), entire document, especially; abstract, para. [0008], [0026], [0039]-[0044], [0046].	1-26
A	US 2006/0143204 A1 (Fish) 29 June 2006 (29.06.2006), entire document.	1-26

☐ 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
22 December 2008 (22.12.2008)	08 JAN 2009
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: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (April 2007)