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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 November 2017 (02.11.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/188969 A1

(51) International Patent Classification:
G06F 9/46 (2006.01) *G06F 9/455* (2006.01)
G06F 9/38 (2006.01) *G06F 15/16* (2006.01)

(21) International Application Number:
PCT/US2016/029937

(22) International Filing Date:
29 April 2016 (29.04.2016)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **AUYOUNG, Alvin**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **MA, Yadi**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **BANERJEE, Sujata**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **LEE, Jung Gun**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **SHARMA, Puneet**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **YU, Da**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US).

(74) Agent: **FOUGERE, Jeffrey R.**; Hewlett Packard Enterprise, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(54) Title: WORK PLACEMENT IN A MULTI-TENANT SYSTEM FOR COMPLIANCE WITH SERVICE-LEVEL OBJECTIVES

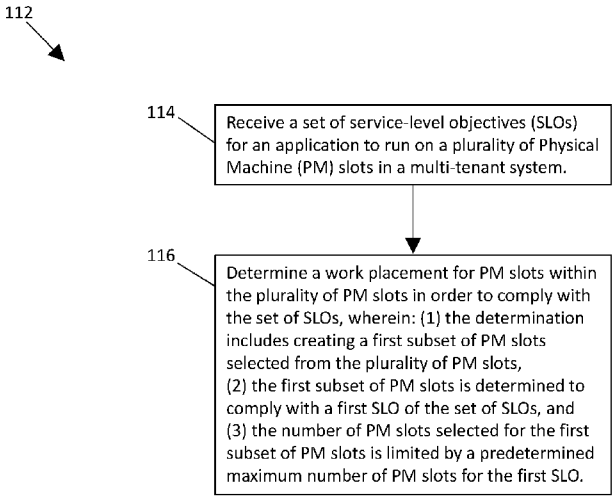


FIG. 4

(57) Abstract: In some examples, a method is described for work placement in a multi-tenant system for compliance with service-level objectives (SLOs). The method can include receiving a set of SLOs for an application to run on a plurality of Physical Machine (PM) slots in a multi-tenant system and determining a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs. Determining the work placement can include creating a first subset of PM slots selected from the plurality of PM slots. The first subset of PM slots can be determined to comply with a first SLO of the set of SLOs. The number of PM slots selected for the first subset of PM slots can be limited by a predetermined maximum number of PM slots for the first SLO.

WO 2017/188969 A1

[Continued on next page]

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

Published:

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

WORK PLACEMENT IN A MULTI-TENANT SYSTEM FOR COMPLIANCE WITH SERVICE-LEVEL OBJECTIVES

BACKGROUND

[0001] Data centers and other facilities can be used to house computing systems, such as various networking, server, and storage systems. Cloud computing and other on demand computing architectures may leverage such computing systems to provide shared resources and data to one or more tenants who share access to the computing systems.

BRIEF DESCRIPTION OF DRAWINGS

[0002] FIG. 1 is a diagram of a work placement environment, according to an example.

[0003] FIG. 2 is a diagram of a work placement environment, according to another example.

[0004] FIG. 3 is a diagram of a work placement environment, according to another example.

[0005] FIG. 4 is a flowchart for a method, according to an example.

[0006] FIG. 5 is a flowchart for a method, according to another example.

[0007] FIG. 6 is a diagram of system, according to another example.

[0008] FIG. 7 is a diagram of a machine-readable storage medium, according to an example.

DETAILED DESCRIPTION

[0009] The following discussion is directed to various examples of the disclosure. Although one or more of these examples may be preferred, the examples disclosed herein should not be interpreted, or otherwise used, to limit the scope of the disclosure, including the claims. In addition, the following description has broad application, and the discussion of any example is meant only to be descriptive of that example, and not intended to intimate that the scope of the disclosure, including the claims, is limited to that example. Throughout the present disclosure, the terms "a" and "an" are intended to denote at least one of a particular element. In addition, as used herein, the term "includes" means includes but not limited to, the term "including" means including but not limited to. The term "based on" means based at least in part on.

[0010] Certain implementations of the present disclosure describe multiple tenant ("multi-tenant") systems, such as certain Infrastructure-as-a-Service (IaaS) cloud systems, that can allow a client to request a set of Virtual Machines (VMs) and/or Physical Machines (PMs) with which to run a software application. Some software applications seek certain performance expectations for use of a multi-tenant system. Such performance expectations may be codified as "Service-Level Objectives" (SLOs) that attempt to guarantee a certain level of service or effort for client VMs and/or applications running on the client VMs. For example, an application may request one or more SLOs relating to a minimum level of availability, computational bandwidth, and/or network bandwidth that the client is to receive during its tenancy.

[0011] A work placement controller or other computing device connected to or within the multi-tenant system may be used to select which VMs and PMs should be used to run the application in order to provide the requested SLOs. As described in further detail herein, such work placement decisions can be automated by using an algorithm that can filter and rank hosts based on metrics such as available Central Processing Unit (CPU) capacity or memory. Moreover, in some implementations, work placement can be optimized by a combination of a sophisticated placement algorithm and various control settings (e.g., client consumption limits on various devices).

[0012] Certain implementations of the present disclosure are directed to methods, systems, mediums, etc., that can allow for improved work placement in a multi-tenant system. In one example implementation, a method can include: (a) receiving a set of SLOs for an application to run on a plurality of PM slots in a multi-tenant system and (b) determining a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs. In such an implementation, the determination can include creating a first subset of PM slots selected from the plurality of PM slots, with the first subset of PM slots being determined to comply with a first SLO of the set of SLOs. The number of PM slots selected for the first subset of PM slots can be limited by a predetermined maximum number of PM slots for the first SLO. Through the use of certain implementations of the present disclosure, a combination of SLOs

can be delivered simultaneously by only using individual, single-purpose work placement programs. It is appreciated that certain implementations of the present disclosure can allow for benefits relating to software modularity without unnecessarily limiting the types of programs that are supported. With software modularity, an administrator does not have to develop or rewrite existing modules to support a new SLO and adding and modifying features may become more manageable. By not imposing too many requirements on the programs supported, an administrator may be free to choose from a much larger set of implementations of VM placement programs without knowledge of the internal implementations. This may be useful for avoiding vendor lock-in or handling legacy program support. Other advantages of implementations presented herein will be apparent upon review of the description and figures.

[0013] FIGs. 1-3 are diagrams of example work placement environments 100 including a work placement determination module 102. Functional and structural aspects of work placement determination module 102 are described in further detail herein with respect to the methods of FIGs. 4-5, the system of FIG. 6, and the medium of FIG. 7. In FIG. 1, work placement environment 100 includes a client 104, a work placement controller 106 including work placement determination module 102, and a multi-tenant system 108 including multiple PMs 110. In FIG. 2, work placement determination module 102 is included on a PM 110 within multi-tenant system 108 rather than on a work placement controller. In FIG. 3, a work placement determination module 102 is included on client 104 rather than on a work placement controller 106 or on a PM within multi-tenant system 108. It is appreciated that one or more aspects of work placement determination module 102 may be included on one or more computing devices within a work placement environment or at another suitable location. For example, in some implementations, a first aspect of work placement determination module 102 (e.g., corresponding to a first function of the module) is included on client 104 and a second aspect of work placement determination module 102 (e.g., corresponding to a second function of the module) is included on PM 110.

[0014] The functionality of work placement controller 106 can, for example, be implemented in part via a software program on a standalone machine, such as a standalone server. In some implementations, work placement controller 106 can be implemented on multi-purpose machines, such as a suitable desktop computer, laptop, tablet, or the like. In some implementations, work placement controller 106 can be implemented on a suitable non-host network node, such as certain types of network switches. It is appreciated that the functionality of work placement controller 106 may be split among multiple controllers or other devices. For example, work placement controller 106 is described and illustrated as including only one controller 106. However, it is appreciated that the disclosure herein can be implemented in work placement environments with multiple controllers. For example, in some work placement environments, network devices are in communication with multiple work placement controllers such that work placement control can be smoothly handed over from a first controller to a second controller if a first controller fails or is otherwise out of operation. As another example, multiple controllers (or other computing devices) can work together to concurrently provide work placement control. In such a work placement environment, a first controller can, for example, control work placement for certain computing devices within multi-tenant system 108 while a second controller can control work placement for other computing devices within multi-tenant system 108 (or elsewhere). In view of the above, reference in this application to a single work placement controller 106 is intended to include such multiple controller configurations (and other suitable multiple controller configurations). Moreover, it is appreciated that the functionality of such a controller may be implemented within another computing device of work placement environment 100, such as on client 104 or on PM 110 of multi-tenant system 108.

[0015] Client 104, controller 106, and PMs 110 can, for example, be in the form of network hosts or other types of network nodes. For example, one or more of these computing devices can be in the form of suitable servers, desktop computers, etc. As but one example, client 104 can be in the form of a desktop computer including a monitor for presenting information to an operator and a keyboard and mouse for

receiving input from an operator, and PM 110 can be in the form of a standalone storage server appliance.

[0016] It is appreciated that work placement environment 100 can include one or more intermediary nodes such as, for example, suitable switches or other multi-port network bridges that process and forward data at the data link layer. In some implementations, one or more of the nodes can be in the form of multilayer switches that operate at multiple layers of the Open Systems Connection (OSI) model (e.g., the data link and network layers).

[0017] Various nodes within work placement environment 100 are connected via one or more data channels, which can, for example be in the form of data cables or wireless data channels. Although a single link between certain nodes is illustrated (e.g., a single line between work placement controller 106 and multi-tenant system 108 in FIG. 1), it is appreciated that each single link may include multiple wires or other wired or wireless data channels. Moreover, one or more nodes within work placement environment 100 may be connected via logical control channels. Such nodes can, for example, be directly connected to only one or a few network nodes, while being indirectly connected to other nodes of multi-tenant system 108 or work placement environment 100. As but one example, work placement controller 106 can be directly connected to a first PM 110 within multi-tenant system 108 via an Ethernet cable, while being indirectly connected to a second PM 110 (e.g., by relying on the first PM 110 as an intermediary for communication with the second PM 110).

[0018] In the example work placement environment 100 depicted in FIGs. 1-3, various computing devices are illustrated, such as a client 104, a standalone work placement controller (e.g., in FIG. 1), and a multi-tenant system including various PMs. It is appreciated however, that the implementations described herein can be used or adapted for work placement environments including more or fewer devices, different types of devices, and different arrangements. For example, there may be any suitable number of clients 104, multi-tenant systems 108, PMs 110, etc., within a work placement environment for use with certain implementations of the present disclosure.

[0019] FIG. 4 illustrates a flowchart for a method 112 according to an example of the present disclosure. The description of method 112 and its components make reference to example work placement environment 100 and elements thereof, such as for example work placement controller 106 and PMs 110, etc. However, it is appreciated that method 112 or aspects thereof can be used or otherwise applicable for any suitable work placement environment. For example, method 112 can be applied to a work placement environment 100 having a different arrangement than those illustrated in FIGs. 1-3.

[0020] In some implementations, method 112 can be implemented or otherwise executed through the use of executable instructions stored on a memory resource (e.g., the memory resource of FIG. 6), executable machine readable instructions stored on a storage medium (e.g., the medium of FIG. 7), in the form of electronic circuitry (e.g., on an Application-Specific Integrated Circuit (ASIC)), and/or another suitable form. Although the description of method 112 herein primarily refers to operations performed on work placement controller 106 for purposes of illustration, it is appreciated that in some implementations, method 112 can be executed on another computing device within work placement environment 100 or in data communication with work placement environment 100.

[0021] Method 112 includes receiving (at block 114) a set of service-level objectives (SLOs) for an application to run on a plurality of Physical Machine (PM) slots in multi-tenant system 108. As used herein, the term "PM slots" can, for example, refer to one or more slots on each PM that can be allocated for tenant requests and whereupon VMs can be placed. The term "VMs" can, in some implementations refer to virtualized machines that run a full copy of an operating system as well as a virtual copy of all the hardware that the operating system uses to run. It is appreciated that the term "VM" may have other meanings and that different types of virtualization techniques may be used. For example, in some implementations, containers are used as VMs or in the place of VMs. As used herein, the term "container" can, for example, refer a virtualized resource that only includes enough of an operating system, supporting programs and libraries, and system resources to run a specific program.

[0022] In certain multi-tenant systems, a single instance of software may run on a server and be used to serve multiple tenants. The term "tenant" can, for example, refer to one or more groups of users who share a common access with specific privileges to the software instance. With a multi-tenant system, applications can provide each tenant a dedicated share of the software instance—such as its data, configuration, user management, tenant individual functionality, and non-functional properties. In some implementations, the multi-tenant system can be in the form of an Infrastructure-as-a-Service (Iaas) cloud. IaaS can refer to online services that abstract a user from the details of infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc. IaaS clouds can, for example, be used to scale services up and down according to varying requirements of a user and may supply services on-demand from large pools of equipment installed in data centers or other facilities.

[0023] As used herein, the term "SLOs" can, for example, refer to service objectives, such as those relating to networking, storage, and/or processing performance. Such objectives can, for example, include: (1) a minimum level of availability during a tenancy of an application, (2) computational bandwidth during a tenancy of the application, and/or (3) network bandwidth during a tenancy of the application. In some implementations, SLOs can relate to QoS criteria, such as criteria relating to implementing time-sensitive network services, such as high speed computing networks where nodes connect compute servers, real-time multimedia services including Internet Protocol television (IPTV), video calls, online gaming, security camera streams, Voice over IP (VoIP) traffic, or other services.

[0024] Method 112 includes determining (at block 116) a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs. Block 116 can for example include creating a first subset of PM slots selected from the plurality of PM slots. The first subset of PM slots can, for example, be determined to comply with a first SLO of the set of SLOs. As used herein, the term "work placement" can, for example, refer to selecting, based on a work placement program, certain PM slots upon which to place a VM. In some implementations, "work placement" can refer to resource mapping of PM slots to assist in achieving the SLOs. The term

"resource mapping" can, for example, refer to mapping one or more threads to one or more cores of PM slots. Note that work placement can, in some implementations, refer to resource mapping problems, such as threads to cores, containers/tasks to servers, storage block to storage node, virtual link to physical link, etc.

[0025] It is appreciated that "compliance" with a SLO can, for example, be based on intra-PM capabilities for each PM within a set of candidate PMs in the multi-tenant system and/or inter-PM capabilities for multiple PMs within the set of candidate PMs. Intra-PM capabilities can, in some implementations, refer to capabilities relating to the use of a single PM, such as for example, certain processing, storage, networking capabilities of the PM. Inter-PM capabilities can, in some implementations, refer to capabilities between multiple PMs, such as for example a network link between multiple PMs, compatibilities between multiple PMs, etc. Determining whether a subset of PM slots can comply with an SLO can be based on specifications of the PM as well as monitored usage. In some implementations, compliance with an SLO can be based on a single threshold value, whereas in other implementations, compliance can be based on multiple criteria. For example, in some implementations, monitored usage can satisfy a SLO by: (1) being below a threshold value for jitter rate, (2) above a threshold value for packet rates, and (3) within a range of values for data rate. It is appreciated that more complicated criteria can be applied. For example, in some implementations the criteria is satisfied only if the packet rate is less than a threshold value and another condition is satisfied, such as a certain amount of time has elapsed since a starting time.

[0026] In block 116, the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots for the first SLO. In some implementations, the number of PM slots selected for the first subset of PM slots can be based on unused VM slots in the multi-tenant system. In some implementations, the number of PM slots selected for the first subset of PM slots can be based on availability of resources in the multi-tenant system. It is appreciated that the predetermined maximum number of PM slots can, in some implementations, be manually provided by an administrator. In some implementations, the predetermined maximum number of PM slots can be

automatically determined using by computer based on one or more static or dynamic parameters. The work placement algorithms presented herein following the description of FIG. 5 provide examples relating to the use of such predetermined maximum number of PM slots for SLOs.

[0027] In some implementations, block 116 can include creating a second subset of PM slots selected from the first set of PM slots. The second subset of PM slots can, for example, be determined to comply with a second SLO of the set of SLOs. The number of PM slots selected for the second subset of PM slots can, for example, be limited by a predetermined maximum number of PM slots for the second SLO. It is appreciated that the predetermined maximum number of PM slots for the first SLO may be different than the predetermined maximum number of PM slots for the second SLO.

[0028] In some implementations, block 116 can include returning a final subset of PM slots that are determined to comply with each SLO of the set of SLOs when the number of PM slots of the final subset of PM slots is greater than a predetermined minimum number of PM slots. In some implementations, block 116 can include returning a null subset of PM slots when the number of PM slots of the final subset of PM slots is less than a predetermined minimum number of PM slots. Additional description relating to the use of determining work placement for PM slots is provided in the various algorithms following the description of FIG. 5.

[0029] Although the flowchart of FIG. 4 shows a specific order of performance, it is appreciated that this order may be rearranged into another suitable order, may be executed concurrently or with partial concurrence, or a combination thereof. Likewise, suitable additional and/or comparable operations may be added to method 112 or other methods described herein in order to achieve the same or comparable functionality. In some implementations, one or more operations are omitted. For example, in some implementations, block 114 of receiving a set of SLOs can be omitted from method 112. It is appreciated that blocks corresponding to additional or alternative functionality of other implementations described herein can be incorporated in method 112. For example, blocks corresponding to the functionality of various aspects of the system of FIG. 6 otherwise described herein

can be incorporated in method 112 even if such functionality is not explicitly characterized herein as a block in a method.

[0030] FIG. 5 illustrates another example of method 112 in accordance with the present disclosure. For illustration, FIG. 5 reproduces various blocks from method 112 of FIG. 4, however it is appreciated that method 112 of FIG. 5 can include additional, alternative, or fewer blocks, functionality, etc., than method 112 of FIG. 4 and is not intended to be limited by the diagram of FIGs. 1-3 (or vice versa) or the related disclosure thereof. It is further appreciated that method 112 of FIG. 5 can incorporate one or more aspects of method 112 of FIG. 4 and vice versa. For example, in some implementations, method 112 of FIG. 4 can include the additional blocks described below with respect to method 112 of FIG. 5.

[0031] Method 112 of FIG. 5 includes creating (block 118) a VM cluster based on the determined work placement, wherein the VM cluster is to run the application to comply with the set of SLOs and running (block 120) the application using the created VM cluster. VM clusters can, for example, be built with VMs installed at distributed servers from one or more physical clusters. VMs in a virtual cluster can, for example, be interconnected logically by a virtual network across several physical networks. In some implementations, running the application includes running the application using a container platform.

[0032] Various example algorithms for providing work placement in a multi-tenant system for compliance with SLOs will now be provided and described. It is appreciated that these example algorithms may include or refer to certain aspects of other implementations described herein (and vice-versa), but are not intended to be limiting towards other implementations described herein. Given a set of independent VM placement programs (also referred to as programs, or modules), a method for automatically creating a logical composition of a subset of these programs is presented. This logical composition can, for example, retain the properties of the individual programs.

[0033] For example, given a set of SLOs, the techniques described herein attempt to find a set of N (e.g., 10) landing spots for VMs such that all of the SLOs are satisfied. For

each of the SLOs, a "black box" is provided that can return us a set of landing spots (e.g., PM slots) that satisfy its own SLO. In accordance with certain implementations of the present disclosure, each black box is asked for more than 10 landing spots (e.g., 50). If all of the black boxes return about 50 locations, then certain implementations will then seek a set of 10 common spots from each of the black boxes. If 10 such spots can be determined, then the spots will represent a set of landing spots satisfying all the SLOs.

[0034] It is appreciated that an exhaustive search for a solution may be too computationally expensive. Instead, the algorithms presented herein can represent a trade-off between a fast heuristic algorithm and one that is likely to find an answer if it exists. However, the algorithms presented herein are not guaranteed to find an answer. For example, in a first heuristic algorithm, the first black box may be queried for 50 locations. If 50 such locations cannot be found, then the black box is queried for 49, 48, 47, etc., locations. Supposing that 45 locations are found, the second black box can be instructed to choose 45 from this set. If it cannot (which is likely), then the second black box can be queried for 46, 45, 44, etc., locations. This process can be repeated until a set of at least 10 locations (N in the algorithm) is provided or a NULL set is returned. Various extensions described herein can be used to speed up the search, perform more search permutations, etc.

[0035] Given a set of N programs P = {p_1, p_2, p_3,.... p_N}. Assuming that each program p_i performs a placement that achieves a property, or set of properties, SLO_i. For example, p_1 (n, P, [s]) can return a set R\subset P, where assigning work to the n elements in R can result in an SLO property SLO_i, being satisfied for the requesting tenant. Given an arbitrary subset Q\subset P, certain implementations of the present disclosure can automatically generate a logically equivalent placement program p_1', that has the properties SLO_1' = {SLO_1 and SLO_2 and . . . SLO_i . . . }, where this new SLO satisfies the goals of each of the individual SLOs

[0036] In these example algorithms, the parameter 'n' refers to a number of VMs requested, the parameter 'P' refers to a list of candidate physical machines PMs where VMs can be placed, the parameter 's' refers to an optional description or parameter for the specific SLO implemented. Each VM placement program returns a

list (which can, if applicable, be an ordered list) of n physical machines where VMs can be placed or a null set if the VMs cannot be placed. The example algorithms are in the form of simplistic pseudo-code. Certain example algorithms presented herein provide techniques for VM placement. However, these algorithms can be applicable for other work placement. For example, it is appreciated that containers or tasks can be substituted for VMs in the algorithms and the techniques may still apply.

[0037] Certain implementations of the present disclosure may be limited by the type of programs Q that can be supported. For example, in some implementations only a single program is supported (q\in Q, whose placement strategy for individual work items are dependent). Such a program can be referred to as a *combinatorial placement*. As an example, consider the example of availability_vm_placement. Let q represent the algorithm that represents this placement. Further, let q(n, P, [s]) = r. As stated earlier, the elements r, satisfy some property, SLO_q. However, if an arbitrary subset r' of r (s.t. |r'| < n) is selected, then it is not guaranteed that the element r' also satisfy the property SLO_q. However, for the other examples, latency_vm_placement and guaranteedbw_vm_placement, this property may be retained. The difference between availability_vm_placement and the other two modules is in the SLO that it satisfies. Certain implementations of the present disclosure rely on the SLO property being maintained for all subsets of the chosen set r.

[0038] In some implementations, proactive migration is not provided. That is, when a set of VMs is placed, existing VM placement algorithms are relied upon to know whether or not these current tenants will incur a violation due to this new placement. Therefore, the ability to deliver SLOs for the duration of the lifetime of a cluster is dependent on the implementation of the individual placement algorithms. An example header for the example VM placement algorithms can be as follows:

```
def generic_vm_placement_program(n, P, [s])
```

[0039] In this example, the placement programs receive two elements as input: 'n' and 'P'. For a system with full capacity available for a request, this set P would correspond to all physical machines in the system. The programs return a subset C/subset P, where

|C| = n, and represents the physical machines that the n VMs should be placed. If two VMs are placed on the same physical machine, p, then p will appear in the return list twice. If no set C exists, then the algorithm will return an empty list. In some implementations, the algorithm may also induce side effects to the system, such as reserving bandwidth or create network transmission limits (i.e., "rate limits") to limit the interference between co-located VMs. Moreover, when applicable, the return set may return an ordered list where items are ordered by decreasing preference (e.g., ability to serve the SLO or another suitable preference).

[0040] A first example VM placement program can have an objective of minimizing latency, such as for example, reducing inter-VM communication latency. Such a program can, for example, be used to return physical machines that are under the same rack, or perhaps, even a single physical machine/enclosure, such that communication does not have to traverse extra network switches per transmission. In some implementations, such a program may consider bandwidth rates and CPU contention induced by neighboring VMs to minimize the potential interference from other applications on various shared queues, such as switch queues, or a CPU scheduling queue. Moreover, in some implementations, this program can reserve bandwidth or create rate limiting mechanisms on relevant network switches to limit interference between co-located VMs. An example heuristic that approximates this type of VM placement is provided below. It is appreciated that a heuristic is used for illustrative purposes and not intended to demonstrate algorithm effectiveness.

[0041] This example algorithm is programmed to return an ordered list of n PMs from set P, using optional SLOs. It is appreciated that a set larger than n (or only constrained by |P|) may also be returned. The example algorithm is provided below with comments preceded by #:

[0042]
```
def latency_vm_placement(n, P, [s]):
    # Attempt to create an ordered list of PM pairs pm1,pm2 based on
    some latency metric.
    # The first one is some notion of best. While the next set is based on
    the first 2 elements.
    pm_pairs = [] # Empty list
    for pm1, pm2 in generate_all_unordered_pairs(P):
```

```
        # Some historical data regarding latency between pm1 and
        pm2 described by s.
        latency_value = measure_latency(pm1, pm2, [s])
        pm_pairs.append([(pm1,pm2), latency_value])
pm_pairs.sort() # Sort based on latency values, lowest latency first
return_set = [] # Eventually return this list
# Seed list with lowest latency pair
pm1, pm2 = pm_pairs.pop(0)
return_set.append(pm1)
return_set.append(pm2)
# Now keep adding nodes to this pair in order of preference
while len(return_set) < n:
        # No more candidates to consider
        if len(pm_pairs) < 1:
        break
        else:
        # Find pairs that have exactly 1 in the current solution set
        curr_cand_pairs = filter(lambda c: c[0] in return_set xor c[1] in
        return_set, pm_pairs)
        if len(curr_cand_pairs) < 1:
        # Pick best one; list should still be ordered
        pm1, pm2 = curr_cand_pairs.pop(0)
        # Exactly one of these PMs is already in the list
        if pm1 in return_set:
        next_cand = pm2
        else:
        next_cand = pm1
        return_set.append(next_cand)
        # Prune all edges who have both members in return set
        (optimization)
        pm_pairs = filter(lambda c: c[0] not in return_set and c[1] not
        in return_set, pm_pairs)
if len(return_set) < n:
        raise Exception # Implementation is code dependent
else:
```

```
        return return_set[0:n]
```

[0043] A second example VM placement program can have an objective of high-availability VM placement, such as by minimizing an impact of a single infrastructural failure. Such a program can, for example, attempt to place a set of N VMs on N different physical machines. In this case, the impact of a single machine failure would be minimized to the disruption of 1/Nth of the hosted service/application. The example algorithm provided below returns an unordered set of PMs and includes a simplifying assumption that ignores machine capacity constraints The example algorithm is provided below with comments preceded by #:

[0044]
```
def availability_vm_placement(n, P, [s]):
    # Map candidate PMs to their fault-domain
    # Each PM belongs to exactly one fault domain in this example
    fault_domains = {} # key=fault_domain_id, val=set of PMs belonging
    to that fault_domain
    # Assume we have the following method for building the mapping of
    fault-domain:servers
    for pm in P:
        f_id = get_fault_domain(pm)
        if f_id not in fault_domains:
            fault_domains[f_id] = Set()
        fault_domains[f_id].insert(pm)
    return_set = []
    for i in range(n):
        # Choose next fault_domain round robin style
        rand_f_id = fault_domains.keys()[i modulo
        len(fault_domains.keys())]
        # Choose next PM within set randomly
        pm = choose_random_item(fault_domains[rand_f_id])
        return_set.append(pm)
    return return_set
```

[0045] A third example VM placement program can have an objective of guaranteed hose bandwidth VM placement, such as for example an algorithm that seeks provide a level of predictability to network performance to clients. In such an example, a

client may want an assignment that can guarantee a minimum throughput volume between (all pairs) of its VMs in the physical infrastructure. Such a placement module would search the infrastructure for an assignment that has sufficient network capacity on links, and NIC capacity on physical machines to guarantee this minimum bandwidth. In addition, this placement algorithm would also actively reserve bandwidth to enforce this assignment regardless of future requests. This example program returns a set R of n PMs from set P, where any placement of said VMs on the set R will results in a guaranteed VM-VM bandwidth B. The example algorithm is provided below with comments preceded by #:

[0046]
```
def guaranteedbw_vm_placement(n, P, B, [s]):
    # Get residual (current capacity) bandwidth on current network
    res_cap = read_current_topology()
    # Note that this is a greedy heuristic and therefore not
    # guaranteed to find a placement even if one exists.
    # return_set = []
    # Keep picking host with largest outbound BW
    for i in range(n):
        # Descending order PMs by outbound NIC capacity
        pm_list.sort(lambda x,y: -1*cmp(res_cap[x], res_cap[y]),
        P.to_list())
        # Pick one
        cand_pm = pm_list[0]
        update_residual_capacity(cand_pm, B)
        return_set.append(cand_pm)
    return return_set
```

[0047] Another example VM placement program is provided below. In this example, a set of black boxes B = {B_1, B_2, B_3, ...} is given, with each B_i being the implementation of an algorithm described in the Model section, and a required cluster size N. In this example, parameter B is the set of VM placement algorithms, or black boxes. In this implementation, set B is ordered such that any combinatorial black box appears at the end of the sequence. Parameter N is a desired cluster size and parameter M is an internal parameter chosen based on domain specific knowledge. This example

returns a list (which can be ordered, if desired) of N physical machines where VMs can be placed or a NULL set if they cannot be. The example algorithm is provided below with comments preceded by #:

```
def composer(B, N, M):
    # Retrieve all unused VM slots on physical machines
    C = getAllAvailableVMSlots()
    # The parameter M is chosen such that M >> N, and M <= C,
    # based on some domain specific knowledge.
    for Bi in B:
        # Try to find a candidate set of size M. If not, try M-1..down to N
        for (m in [M, N]):
        C' = Bi (m, C)
        if C' != None:
            break # Found answer
        # else try a smaller m
    if C' == None || size(C') < N:
        return None # Could not find satisfying answer
    else: # Move on to next black box
        C = C'
        C' = None
        M = min(M, size(C)) # Update M after each iteration
    if size(C) < N:
        return None
    CN = C[0:N] # Return the first N elements from set C
    return CN
```

[0048] The above algorithm, as well as other algorithms described herein, may have multiple variations. For example, the selection of M can, in some implementations, be tailored for a particular deployment environment. In such an implementation, a very large M can increase a likelihood of finding a solution, but may, in some situations, use more computation time, and may, in some situations, result in a less "efficient" solution, since each black box may be given an artificially more difficult problem to fit. It is appreciated that a smaller M may make the reverse trade-off. In some implementations, the above algorithm may try several values of M (for

example by doing a binary search of M = |C|, M = |C|/2,...) until it finds an approximate smallest value of M to return a solution. This approach can, for example, try to strike a balance between the tradeoffs to find an efficient but quick solution. A modification to the above algorithm to achieve this functionality can, for example, indicate that m is not to be assigned every value between M and N. For example, statements in the for-loop statement may be modified to provide such functionality.

[0049] In some implementations, the above algorithm may be modified so as to compute a first |B|-1 candidate sets in parallel. This can, in some implementations, be more efficient as |B| becomes larger, or if a computation time of each B_i is a bottleneck. In such an implementation, a last B_i, where i = |B|, can be computed separately, if, for example, it exhibits combinatorial properties described herein. Certain implementations described herein can, for example, be limited so as to support one such B_i. It is appreciated that using the above algorithm, the last L can be generalized by increasing an exhaustiveness of a search, as described for example, in the below algorithm:

[0050] def composer_parallel(B, N, M):

```
# Retrieve all unused VM slots on physical machines
C = getAllAvailableVMSlots()
# Note that unlike the other for-loop, each iteration of this for each loop is
independent, and can be done with parallel threads. Except for the last item, which
requires separate computation if it exhibits combinatorial properties.
for each Bi in B[0:|B|-1]:
        Ci = Bi (min(M, size(C)), C)
O = intersection(Ci) # Compute the intersection over all sets {C1, C2, ....}
CN = B_[|B|](min(M,O), O)
if size(CN) < N:
        return None
return CN.
```

[0051] It is appreciated that using any of the algorithms above, one can test multiple values of M, as in a default algorithm that searches m in [M,N]. As provided above, in some implementations, an exhaustiveness of a search may be increased such as by

permuting an order of black boxes in a composition. In such an implementation, all possible permutations of the B_i sequence may be tried. This can, in some implementations, include generating a set of all such permutations, and running the algorithm for each until a desired set is found. For example, for a set of B black boxes, this can, at most, be B! permutations or (B-1)! if keeping a last black box the same for all permutations due to combinatorial placement.

[0052] In some implementations, multiple candidate sets can be generated from each black box. For example, a black box B_i, may return different sets, B_i (N, C), even with the same candidate C, due to any pre-programmed randomness within it. Moreover, if an overlap using the simple heuristics above cannot be found, then candidate set C can be refined to find alternative candidate sets. An example algorithm for a more exhaustive search is described in the below algorithm:

[0053]
```
def composer_exhaustive(B, N, M, L):

    # Retrieve all unused VM slots on physical machines
    C = getAllAvailableVMSlots()
    do:
        for each Bi in B[0:|B|-1]:
            Ci = Bi (min(M, size(c)), C)
        O = intersection(Ci) # Compute the intersection over all sets {C1, C2, ....}
        if size(O) <= N:
            # Find L culprit VMs that appears the least often, and remove that
            from candidacy
            U = union(Ci) # Take union of all candidate sets.
            Unique = extract_unique(U) # Remove duplicates.
            # Sort unique items by their frequency in each Bi's output.
            # Sorted in order of decreasing frequency.
            Unique.sort(lambda x,y: -1*compare(U.count(x), U.count(y)))
            # Remove L least-frequent elemtents from the candidate list.
            # If there aren't L elements anymore, then abort
        if size(Unique) < L:
            return None
        C = C.remove(Unique[-L:])
```

$C_{|B|} = B_{|B|}(\min(M,C), O)$

If this candidate set isn't large enough, it'll re-run with a pared down set C.

while size($C_{|B|}$) < N

if size(C_N) < N:

return None

return $C_{|B|}$ (0:N)

[0054] In some implementations, the above techniques may be applied to a single B_i that exhibits combinatorial properties. However, using this technique, techniques can be generalized to K such properties. For example, with a remaining B_1 to B_k combinatorial boxes and a candidate set C, which is the result of composing some prior set of B_i. In such a situation, compose_exhaustive({B_1, . . . B_k}, N, C, X) can be called to search for a solution that satisfies all remaining B_i. It is appreciated that setting X can be determined and in some implementations can be determined within a range from 0 and |C|-N.

[0055] FIG. 6 is a diagram of a system 122 in accordance with the present disclosure. As described in further detail below, system 122 includes a processing resource 124 and a memory resource 126 that stores machine-readable instructions 128, 130, 132, and 134. For illustration, the description of system 122 of FIG. 6 makes reference to various aspects of the diagram of FIGs. 1-3 as well as method 112 of FIGs. 4-5. Indeed, for consistency, the same reference number for system 122 is used for the work placement controller of FIG. 106. However it is appreciated that system 122 can be in the form of one or more other suitable computing devices within one or more work placement environments described herein or other suitable environments. For example, in some implementations, system 122 can be in the form of client 104 and/or PM 110 within multi-tenant system. Moreover, system 122 of FIG. 6 can include additional, alternative, or fewer aspects, functionality, etc., than the implementation described with respect to method 112 and is not intended to be limited by the related disclosure thereof.

[0056] Instructions 128 stored on memory resource 126 are, when executed by processing resource 124, to cause processing resource 124 to receive first and second SLOs for an application. Instructions 128 can incorporate one or more aspects of blocks of

method 112 or another suitable aspect of other implementations described herein (and vice versa).

[0057] Instructions 130 stored on memory resource 126 are, when executed by processing resource 124, to cause processing resource 124 to determine a first set of PM slots in a multi-tenant system based on the first SLO such that the determined first set of PM slots is less than a predetermined maximum number of PM slots. Instructions 130 can incorporate one or more aspects of blocks of method 112 or another suitable aspect of other implementations described herein (and vice versa).

[0058] Instructions 132 stored on memory resource 126 are, when executed by processing resource 124, to cause processing resource 124 to determine a second set of PM slots from the first set of PM slots based on the second SLO such that the determined second set of PM slots is greater than a predetermined minimum number of PM slots. Instructions 132 can incorporate one or more aspects of blocks of method 112 or another suitable aspect of other implementations described herein (and vice versa).

[0059] Instructions 134 stored on memory resource 126 are, when executed by processing resource 124, to cause processing resource 124 to run the application using the second set of PM slots. Instructions 134 can incorporate one or more aspects of blocks of method 112 or another suitable aspect of other implementations described herein (and vice versa).

[0060] Processing resource 124 of system 122 can, for example, be in the form of a central processing unit (CPU), a semiconductor-based microprocessor, a digital signal processor (DSP) such as a digital image processing unit, other hardware devices or processing elements suitable to retrieve and execute instructions stored in memory resource 126, or suitable combinations thereof. Processing resource 124 can, for example, include single or multiple cores on a chip, multiple cores across multiple chips, multiple cores across multiple devices, or suitable combinations thereof. Processing resource 124 can be functional to fetch, decode, and execute instructions as described herein. As an alternative or in addition to retrieving and executing instructions, processing resource 124 can, for example, include at least one integrated

circuit (IC), other control logic, other electronic circuits, or suitable combination thereof that include a number of electronic components for performing the functionality of instructions stored on memory resource 126. The term "logic" can, in some implementations, be an alternative or additional processing resource to perform a particular action and/or function, etc., described herein, which includes hardware, e.g., various forms of transistor logic, application specific integrated circuits (ASICs), etc., as opposed to machine executable instructions, e.g., software firmware, etc., stored in memory and executable by a processor. Processing resource 124 can, for example, be implemented across multiple processing units and instructions may be implemented by different processing units in different areas of environment 100.

[0061] Memory resource 126 of system 122 can, for example, be in the form of a non-transitory machine-readable storage medium, such as a suitable electronic, magnetic, optical, or other physical storage apparatus to contain or store information such as machine-readable instructions 128, 130, 132, and 134. Such instructions can be operative to perform one or more functions described herein, such as those described herein with respect to method 112 or other methods described herein. Memory resource 126 can, for example, be housed within the same housing as processing resource 124 for system 122, such as within a computing tower case for system 122 (in implementations where system 122 is housed within a computing tower case). In some implementations, memory resource 126 and processing resource 124 are housed in different housings. As used herein, the term "machine-readable storage medium" can, for example, include Random Access Memory (RAM), flash memory, a storage drive (e.g., a hard disk), any type of storage disc (e.g., a Compact Disc Read Only Memory (CD-ROM), any other type of compact disc, a DVD, etc.), and the like, or a combination thereof. In some implementations, memory resource 126 can correspond to a memory including a main memory, such as a Random Access Memory (RAM), where software may reside during runtime, and a secondary memory. The secondary memory can, for example, include a nonvolatile memory where a copy of machine-readable instructions are stored. It is appreciated that both machine-readable instructions as well as related data can be stored on memory mediums and that multiple mediums can be treated as a single medium for purposes of description.

[0062] Memory resource 126 can be in communication with processing resource 124 via a communication link 136. Each communication link 136 can be local or remote to a machine (e.g., a computing device) associated with processing resource 124. Examples of a local communication link 136 can include an electronic bus internal to a machine (e.g., a computing device) where memory resource 126 is one of volatile, non-volatile, fixed, and/or removable storage medium in communication with processing resource 124 via the electronic bus.

[0063] In some implementations, one or more aspects of system 122 (as well as controller 106, PMs, or other devices of work placement environment 100) can be in the form of functional modules that can, for example, be operative to execute one or more processes of instructions 128, 130, 132, or 134 or other functions described herein relating to other implementations of the disclosure. As used herein, the term "module" refers to a combination of hardware (e.g., a processor such as an integrated circuit or other circuitry) and software (e.g., machine- or processor-executable instructions, commands, or code such as firmware, programming, or object code). A combination of hardware and software can include hardware only (i.e., a hardware element with no software elements), software hosted at hardware (e.g., software that is stored at a memory and executed or interpreted at a processor), or hardware and software hosted at hardware. It is further appreciated that the term "module" is additionally intended to refer to one or more modules or a combination of modules. Each module of a controller 106 or other computing device can, for example, include one or more machine-readable storage mediums and one or more computer processors.

[0064] In view of the above, it is appreciated that the various instructions of system 122 described above can correspond to separate and/or combined functional modules. For example, instructions 128 can correspond to a "receiving module" to receive a set of SLOs for an application to run on a VM cluster over a plurality of PM slots in a multi-tenant system and instructions 130 can correspond to a "work placement determination module" to determine a work placement for PM slots within the plurality of PM slots based on the identified number of VM slots and a first subset of PM slots from the plurality of PM slots that comply with the first SLO. It is further

appreciated that a given module can be used for multiple functions. As but one example, in some implementations, a single module can be used to both receive a set of SLOs (e.g., corresponding to the functionality of instructions 128) as well as to determine a work placement for PM slots (corresponding to the functionality of instructions 130 and/or 132).

[0065] One or more nodes within system 122 (e.g., client 104, work placement controller 106, PM 110) can further include a suitable communication module to allow networked communication between nodes within work placement environment 100. Such a communication module can, for example, include a network interface controller having an Ethernet port and/or a Fibre Channel port. In some implementations, such a communication module can include wired or wireless communication interface, and can, in some implementations, provide for virtual network ports. In some implementations, such a communication module includes hardware in the form of a hard drive, related firmware, and other software for allowing the hard drive to operatively communicate with other hardware of work placement environment 100. The communication module can, for example, include machine-readable instructions for use with communication the communication module, such as firmware for implementing physical or virtual network ports.

[0066] FIG. 7 illustrates a machine-readable storage medium 138 including various instructions that can be executed by a computer processor or other processing resource. In some implementations, medium 138 can be housed within a controller, such as work placement controller 106, or on another computing device within work placement environment (e.g., client 104 or PM 110) or in local or remote wired or wireless data communication with work placement environment 100.

[0067] For illustration, the description of machine-readable storage medium 138 provided herein makes reference to various aspects of system 122 (e.g., processing resource 124) and other implementations of the disclosure (e.g., method 112). Although one or more aspects of system 122 (as well as instructions such as instructions 128, 130, 132, and 134) can be applied or otherwise incorporated with medium 138, it is appreciated that in some implementations, medium 138 may be stored or housed separately from such a system. For example, in some implementations, medium 138

can be in the form of Random Access Memory (RAM), flash memory, a storage drive (e.g., a hard disk), any type of storage disc (e.g., a Compact Disc Read Only Memory (CD-ROM), any other type of compact disc, a DVD, etc.), and the like, or a combination thereof.

[0068] Medium 138 includes machine-readable instructions 140 stored thereon to cause processing resource 124 to receive a set of SLOs for an application to run on a VM cluster over a plurality of PM slots in a multi-tenant system. Instructions 140 can, for example, incorporate one or more aspects of method 112 or system 122 or another suitable aspect of other implementations described herein (and vice versa).

[0069] Medium 138 includes machine-readable instructions 142 stored thereon to cause processing resource 124 to identify a number of VM slots for use in determining a VM cluster. Instructions 142 can, for example, incorporate one or more aspects of method 112 or system 122 or another suitable aspect of other implementations described herein (and vice versa).

[0070] Medium 138 includes machine-readable instructions 144 stored thereon to cause processing resource 124 to determine a work placement for PM slots within the plurality of PM slots based on the identified number of VM slots and a first subset of PM slots from the plurality of PM slots that comply with the first SLO, wherein the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots. The predetermined maximum number of PM slots can, for example, is based on the number of SLOs in the set of SLOs and the identified number of VM slots. Instructions 144 can, for example, incorporate one or more aspects of method 112, system 122, or another suitable aspect of other implementations described herein (and vice versa).

[0071] While certain implementations have been shown and described above, various changes in form and details may be made. For example, some features that have been described in relation to one implementation and/or process can be related to other implementations. In other words, processes, features, components, and/or properties described in relation to one implementation can be useful in other implementations. Furthermore, it should be appreciated that the systems and

methods described herein can include various combinations and/or sub-combinations of the components and/or features of the different implementations described. Thus, features described with reference to one or more implementations can be combined with other implementations described herein.

[0072] As used herein, "logic" is an alternative or additional processing resource to perform a particular action and/or function, etc., described herein, which includes hardware, e.g., various forms of transistor logic, application specific integrated circuits (ASICs), etc., as opposed to machine executable instructions, e.g., software firmware, etc., stored in memory and executable by a processor. Further, as used herein, "a" or "a number of" something can refer to one or more such things. For example, "a number of widgets" can refer to one or more widgets. Also, as used herein, "a plurality of" something can refer to more than one of such things.

CLAIMS

What is claimed is:

1. A method comprising:

receiving a set of service-level objectives (SLOs) for an application to run on a plurality of Physical Machine (PM) slots in a multi-tenant system; and

determining a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs,

wherein the determination includes creating a first subset of PM slots selected from the plurality of PM slots,

wherein the first subset of PM slots is determined to comply with a first SLO of the set of SLOs, and

wherein the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots for the first SLO.

2. The method of claim 1, wherein determining the work placement includes:

creating a second subset of PM slots selected from the first set of PM slots,

wherein the second subset of PM slots is determined to comply with a second SLO of the set of SLOs, and

wherein the number of PM slots selected for the second subset of PM slots is limited by a predetermined maximum number of PM slots for the second SLO.

3. The method of claim 1, wherein determining the work placement includes:

returning a final subset of PM slots that are determined to comply with each SLO of the set of SLOs when the number of PM slots of the final subset of PM slots is greater than a predetermined minimum number of PM slots.

4. The method of claim 1, wherein determining the work placement includes:

returning a null subset of PM slots when the number of PM slots of the final subset of PM slots is less than a predetermined minimum number of PM slots.

5. The method of claim 1, further comprising:

creating a Virtual Machine (VM) cluster based on the determined work placement, wherein the VM cluster is to run the application to comply with the set of SLOs; and

running the application using the created VM cluster.

6. The method of claim 1, wherein the multi-tenant system is an Infrastructure-as-a-Service (IaaS) cloud.

7. The method of claim 1, wherein the number of PM slots selected for the first subset of PM slots is based on availability of resources in the multi-tenant system.

8. The method of claim 1, wherein work placement in a VM includes selecting, based on the work placement program, PM slots upon which to place a VM.

9. The method of claim 1, wherein work placement in a VM includes resource mapping of PM slots to assist in achieving the SLOs.

10. The method of claim 1, wherein resource mapping includes mapping one or more threads to one or more cores of PM slots.

11. The method of claim 10, wherein the number of PM slots selected for the first subset of PM slots is based on unused Virtual Machine (VM) slots in the multi-tenant system.

12. A non-transitory machine readable storage medium having stored thereon machine readable instructions to cause a computer processor to:

receive a set of service-level objectives (SLOs) for an application to run on a Virtual Machine (VM) cluster over a plurality of Physical Machine (PM) slots in a multi-tenant system;

identify a number of VM slots for use in determining a VM cluster; and

determine a work placement for PM slots within the plurality of PM slots based on the identified number of VM slots and a first subset of PM slots from the plurality of PM slots that comply with the first SLO,

wherein the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots.

13. The medium of claim 12, wherein the predetermined maximum number of PM slots is based on the number of SLOs in the set of SLOs and the identified number of VM slots.

14. A system comprising:

a processing resource; and

a memory resource storing machine readable instructions to cause the processing resource to:

receive first and second service-level objectives (SLOs) for an application;

determine a first set of Physical Machine (PM) slots in a multi-tenant system based on the first SLO such that the determined first set of PM slots is less than a predetermined maximum number of PM slots;

determine a second set of PM slots from the first set of PM slots based on the second SLO such that the determined second set of PM slots is greater than a predetermined minimum number of PM slots; and

run the application using the second set of PM slots.

15. The system of claim 14, wherein running the application using the second set of PM slots includes running the application using a container platform.

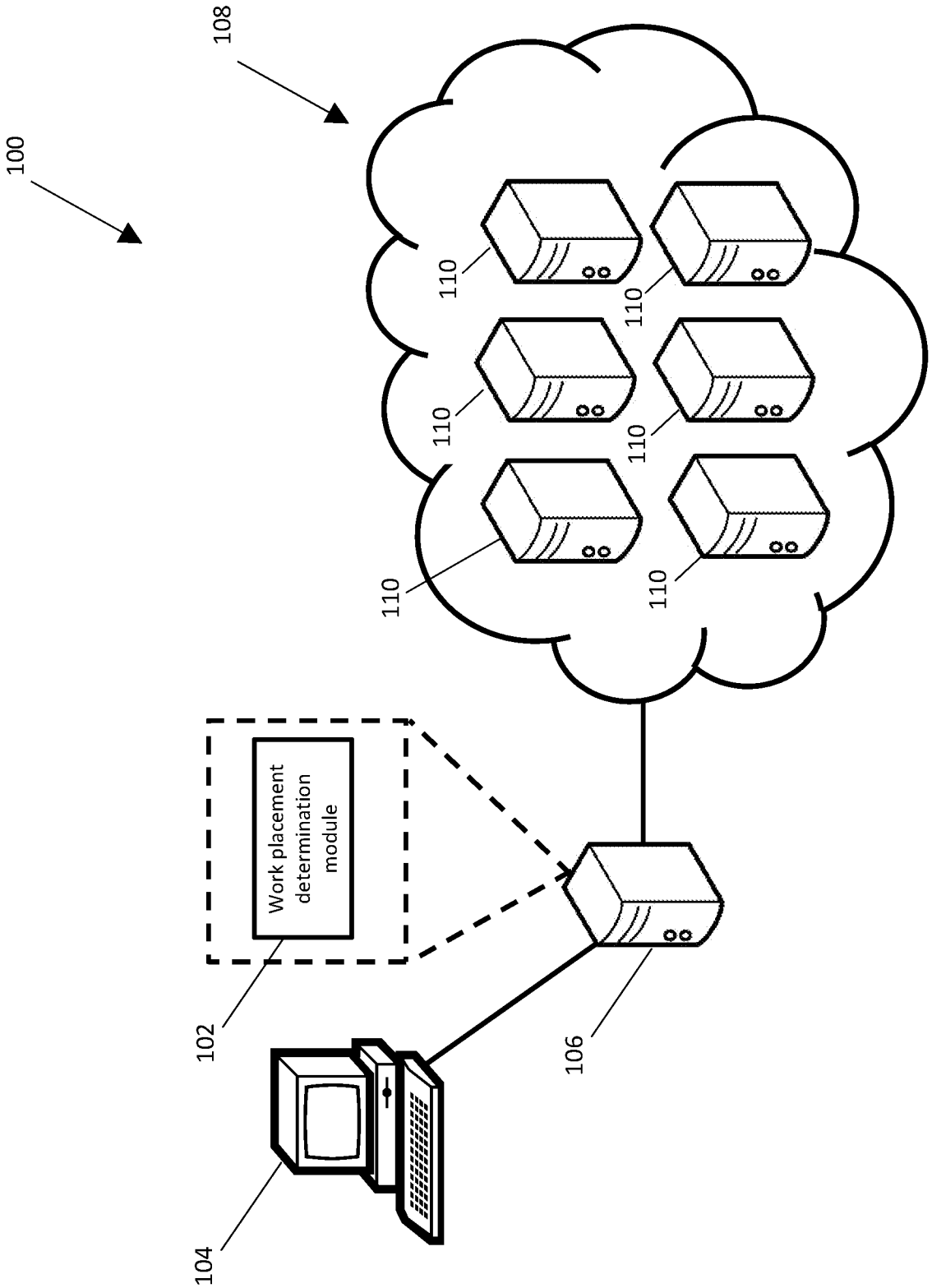

100
108
110
110
110
110
110
110
Work placement determination module
102
104
106

FIG. 1

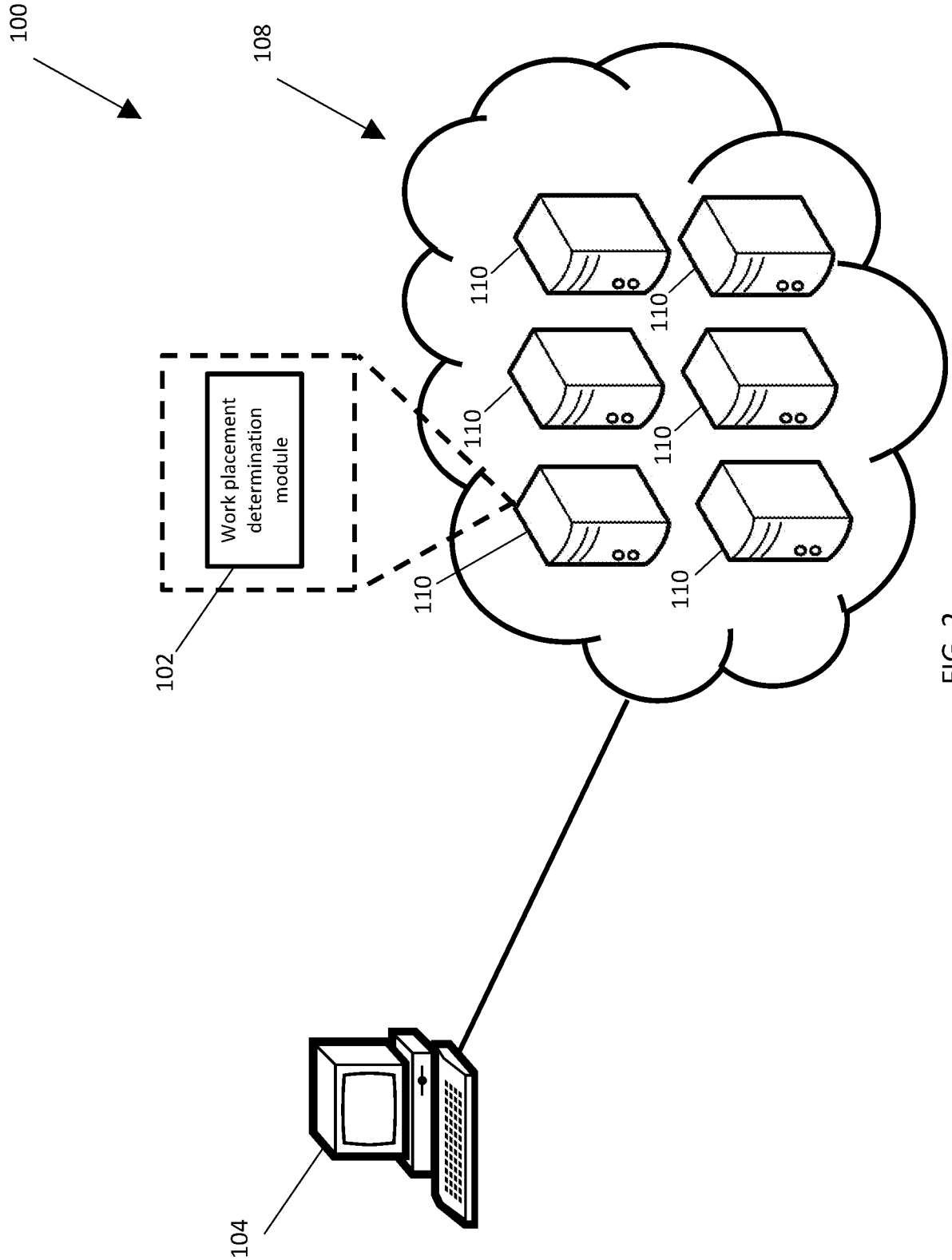

100
108
110
110
110
110
110
110
Work placement determination module
102
104

FIG. 2

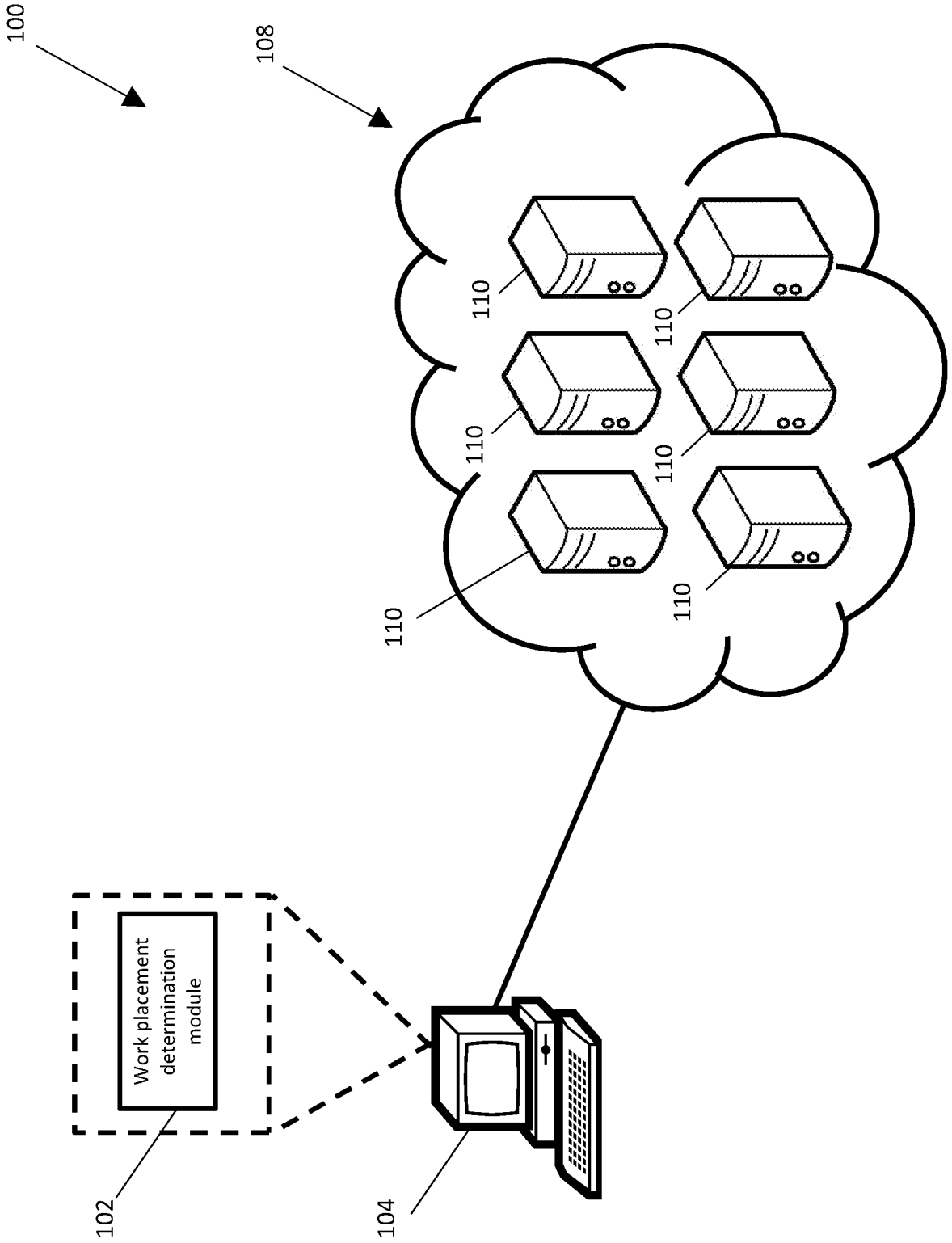

100
108
110
110
110
110
110
110
Work placement determination module
102
104

FIG. 3

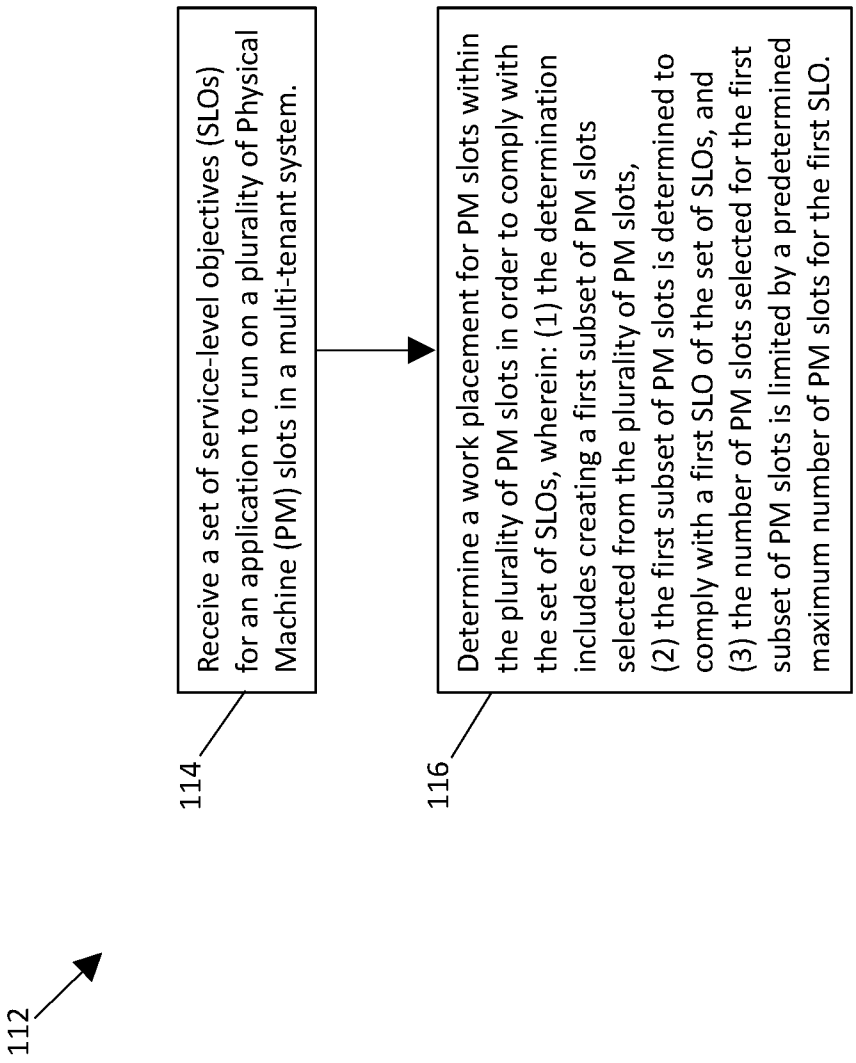

112
114
Receive a set of service-level objectives (SLOs) for an application to run on a plurality of Physical Machine (PM) slots in a multi-tenant system.
116
Determine a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs, wherein: (1) the determination includes creating a first subset of PM slots selected from the plurality of PM slots, (2) the first subset of PM slots is determined to comply with a first SLO of the set of SLOs, and (3) the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots for the first SLO.

FIG. 4

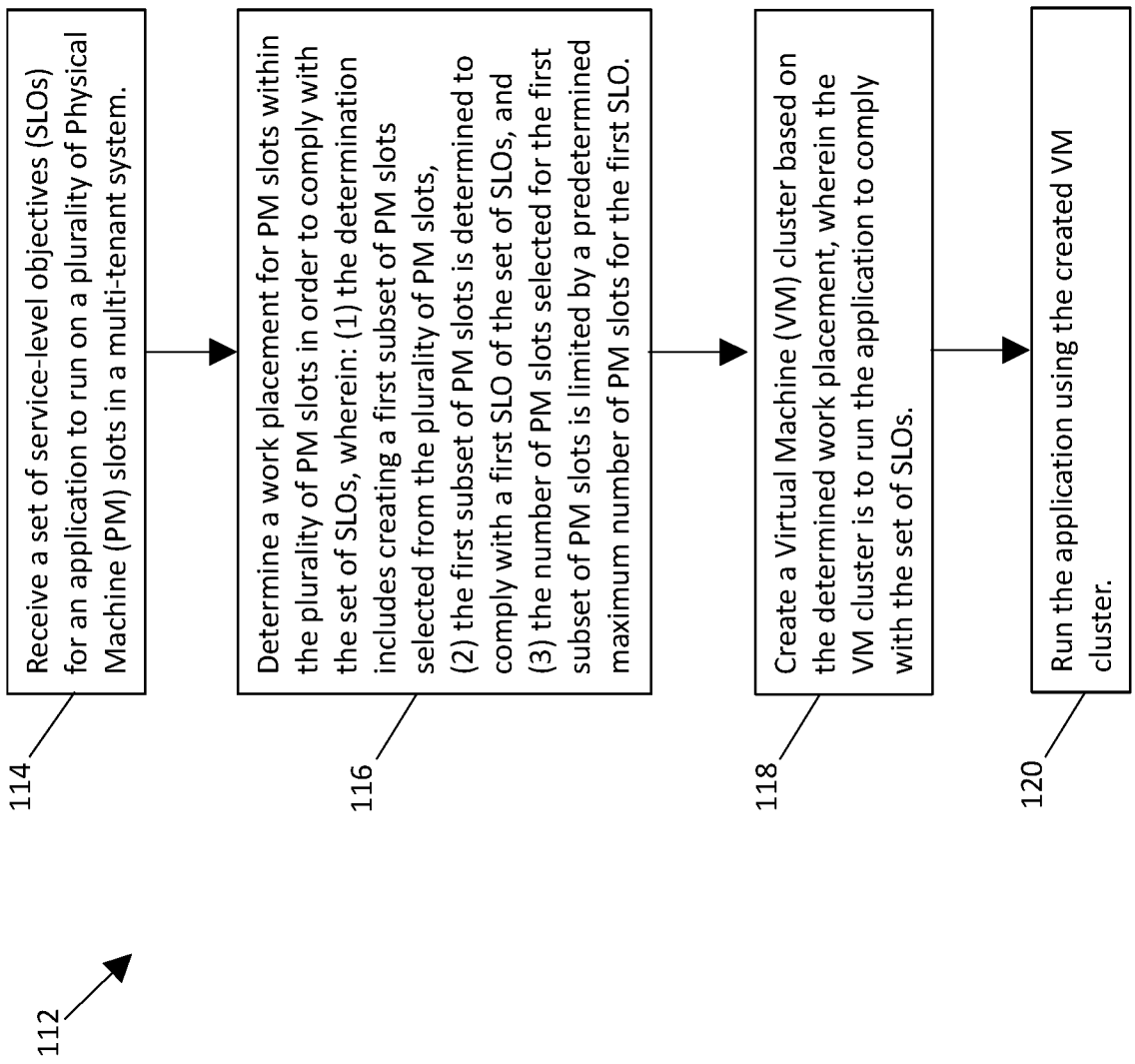

112
114
Receive a set of service-level objectives (SLOs) for an application to run on a plurality of Physical Machine (PM) slots in a multi-tenant system.
116
Determine a work placement for PM slots within the plurality of PM slots in order to comply with the set of SLOs, wherein: (1) the determination includes creating a first subset of PM slots selected from the plurality of PM slots, (2) the first subset of PM slots is determined to comply with a first SLO of the set of SLOs, and (3) the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots for the first SLO.
118
Create a Virtual Machine (VM) cluster based on the determined work placement, wherein the VM cluster is to run the application to comply with the set of SLOs.
120
Run the application using the created VM cluster.

FIG. 5

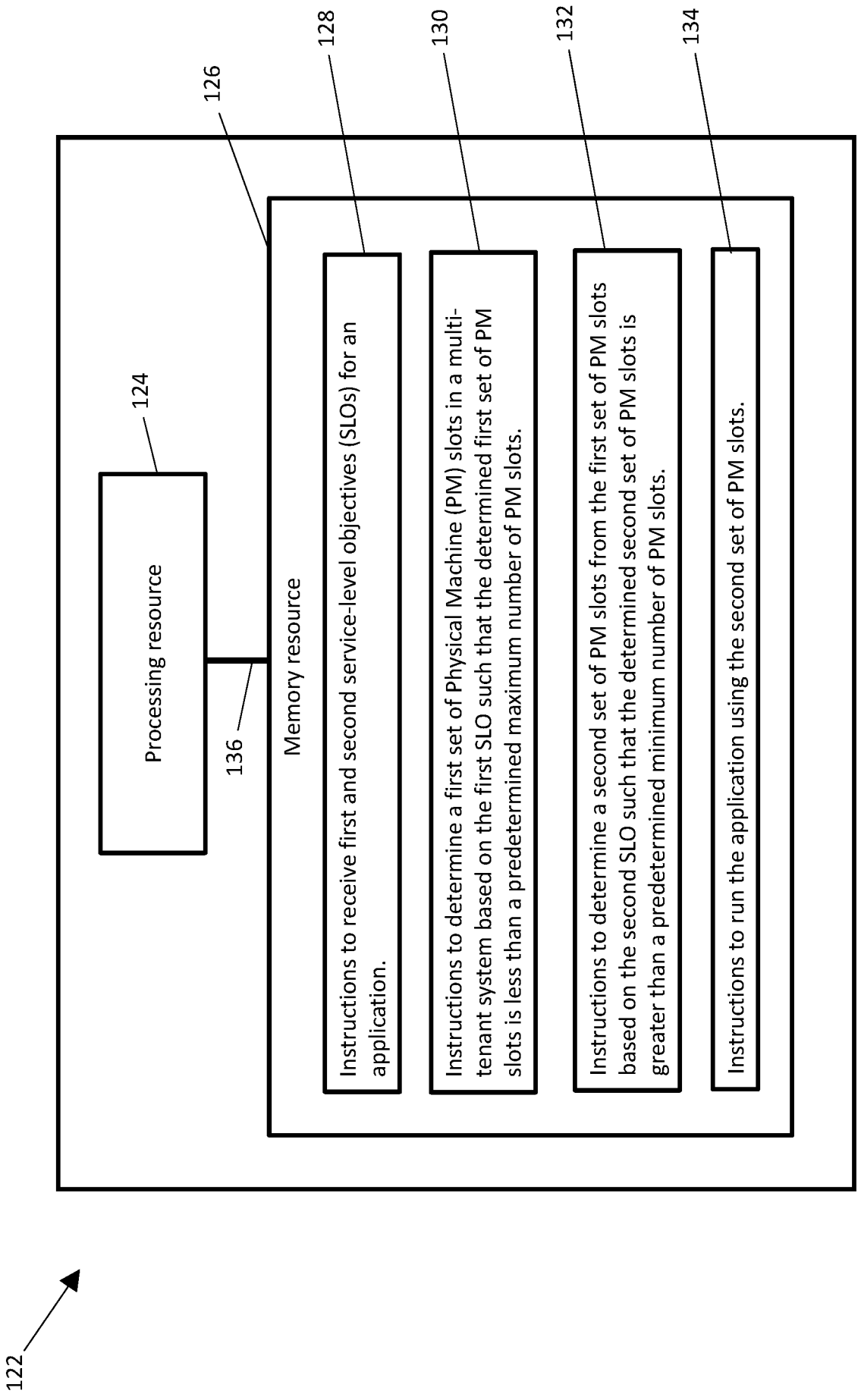

122
126
128
130
132
134
124
Processing resource
136
Memory resource
Instructions to receive first and second service-level objectives (SLOs) for an application.
Instructions to determine a first set of Physical Machine (PM) slots in a multi-tenant system based on the first SLO such that the determined first set of PM slots is less than a predetermined maximum number of PM slots.
Instructions to determine a second set of PM slots from the first set of PM slots based on the second SLO such that the determined second set of PM slots is greater than a predetermined minimum number of PM slots.
Instructions to run the application using the second set of PM slots.

FIG. 6

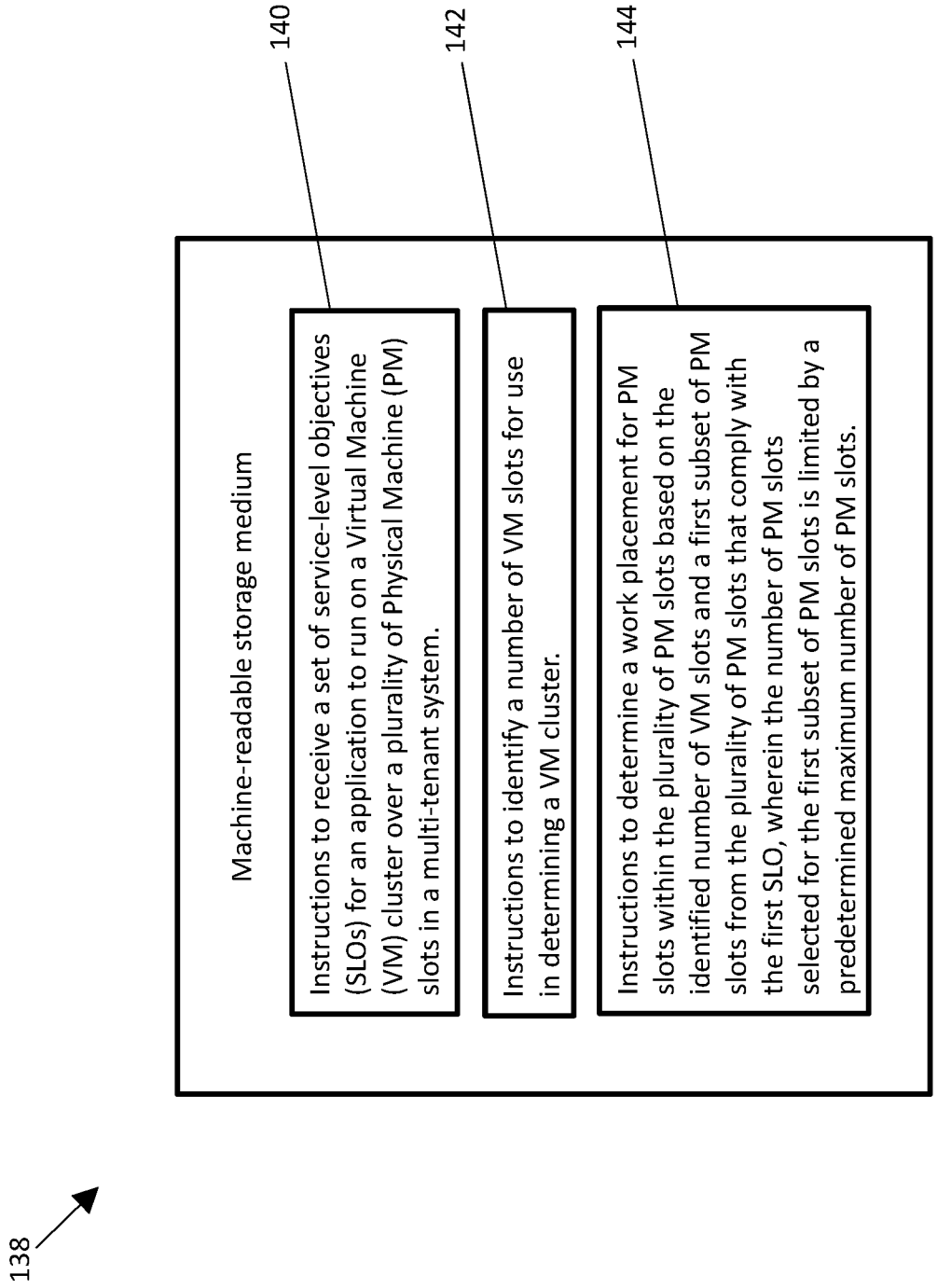

138
Machine-readable storage medium
140
Instructions to receive a set of service-level objectives (SLOs) for an application to run on a Virtual Machine (VM) cluster over a plurality of Physical Machine (PM) slots in a multi-tenant system.
142
Instructions to identify a number of VM slots for use in determining a VM cluster.
144
Instructions to determine a work placement for PM slots within the plurality of PM slots based on the identified number of VM slots and a first subset of PM slots from the plurality of PM slots that comply with the first SLO, wherein the number of PM slots selected for the first subset of PM slots is limited by a predetermined maximum number of PM slots.

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/029937

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/46(2006.01)i, G06F 9/38(2006.01)i, G06F 9/455(2006.01)i, G06F 15/16(2006.01)i

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)
G06F 9/46; G06F 15/16; G06F 9/445; G06F 15/173; G06F 9/44; G06F 9/50; G06F 3/01; G06F 9/38; G06F 9/455

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: service-level objectives (SLOs), Physical Machine (PM) slot, multi-tenant, subset, virtual machine, cloud, resource

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2014-0033218 A1 (PATRICK CHARLES MCGEER et al.) 30 January 2014 See paragraphs [0008]-[0043]; and figures 1-2.	1-15
Y	US 2016-0054991 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 25 February 2016 See paragraphs [0007], [0022]-[0032]; and figures 1, 3A.	1-15
A	US 2011-0029882 A1 (DEVENDRA RAJKUMAR JAISINGHANI) 03 February 2011 See paragraphs [0058]-[0067]; and figures 3C-6.	1-15
A	US 2013-0138806 A1 (ATUL A. GOHAD et al.) 30 May 2013 See paragraphs [0024]-[0060]; claim 1; and figures 1A-2.	1-15
A	US 2014-0189639 A1 (MICROSOFT CORPORATION) 03 July 2014 See paragraphs [0025]-[0046]; claims 1-7; and figure 1.	1-15

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

☒ See patent family annex.

* 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
07 February 2017 (07.02.2017)	**09 February 2017 (09.02.2017)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/029937

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0033218 A1	30/01/2014	US 9396034 B2	19/07/2016
US 2016-0054991 A1	25/02/2016	GB 2529475 A	24/02/2016
US 2011-0029882 A1	03/02/2011	CN 102576354 A	11/07/2012
		CN 102576354 B	06/07/2016
		DE 112010003144 T5	05/07/2012
		GB 2485712 A	23/05/2012
		GB 2485712 A8	10/06/2015
		GB 2485712 B	20/05/2015
		GB 2485712 B8	10/06/2015
		US 2011-0029673 A1	03/02/2011
		US 2011-0029810 A1	03/02/2011
		US 2011-0029981 A1	03/02/2011
		US 2012-0102180 A1	26/04/2012
		US 2013-0080902 A1	28/03/2013
		US 2015-0220408 A1	06/08/2015
		US 2016-0043903 A1	11/02/2016
		US 2016-0197851 A1	07/07/2016
		US 8316305 B2	20/11/2012
		US 9009521 B2	14/04/2015
		US 9201557 B2	01/12/2015
		US 9329951 B2	03/05/2016
		US 9442810 B2	13/09/2016
		US 9491117 B2	08/11/2016
		WO 2011-014827 A1	03/02/2011
		WO 2011-014830 A1	03/02/2011
		WO 2011-014835 A1	03/02/2011
US 2013-0138806 A1	30/05/2013	US 2013-0138798 A1	30/05/2013
		US 9274850 B2	01/03/2016
US 2014-0189639 A1	03/07/2014	CN 103226539 A	31/07/2013
		CN 103226539 B	08/06/2016
		EP 2834738 A1	11/02/2015
		US 2013-0268914 A1	10/10/2013
		US 8707254 B2	22/04/2014
		US 9015662 B2	21/04/2015
		WO 2013-152245 A1	10/10/2013