SYSTEM AND METHOD OF SHARING COMPUTING RESOURCES

TECHNICAL FIELD

[0001] The following generally relates to computing environments and more specifically relates to distributed virtualization computing environments.

BACKGROUND

[0002] A virtual machine (VM) is a software computer used to emulate a physical computer. A virtual server operates in a “multi-tenant” environment, meaning that multiple VMs run on the same physical hardware. In some embodiments, the computing resources of a physical server are virtualized and shared amongst all VMs running on it. FIG. 1 shows the architecture of a typical VM environment. The architecture of a virtual server is typically more complex than that of a physical server and thus, a hypervisor is installed on top of physical hardware. A hypervisor is a computer software, firmware, or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine. A hypervisor can be used to create and manage VMs, which have their own virtual computing resources. Multiple guest operating systems and server applications can be loaded on top of the virtual hardware. Each of the guest operating systems is able to compute any number of individual, different tasks. Thus, virtual servers can allow several operating systems and applications to run on the basis of the shared physical hardware.

[0003] However, one limitation of typical virtualization environment is that the virtual machines are limited by the physical hardware resources of the single host machine. That is, the hardware resources (i.e. the RAM, CPU, I/O, storage, GPU, etc.) of the single host machine are shared between the multiple virtual machine instances. If there are a limited amount of physical hardware resources available to the virtual machines, the performance of the virtual machines can be negatively affected. This can cause an impact to the users as more compute time may be required to do the same tasks.

[0004] It is possible to overcome the limitation of hardware resources by using distributed or grid computing. Grid computing is the use of widely distributed computer resources to reach a common goal. FIG. 2 shows a diagram in which grid computing is used. In this embodiment, a plurality of computers (worker nodes) are typically connected to a master node to solve a single complex problem. That is, the master node can split any given large task into a plurality of smaller computing tasks and distribute the smaller tasks among the plurality of worker nodes.

Each worker node is only required to complete a portion of the large task. The master node may then compile the individual tasks completed by the worker nodes to produce the desired output of the single complex task. In this way, the physical hardware resources of the plurality of computers can be used for computing and the system is typically not limited by the physical hardware resources. However, the distributed computing system would not be useful for users who wish to compute individual, different tasks of varying complexities using the plurality of individual computers.

[0005] It is therefore an objective of the present invention to overcome the disadvantages of the prior art.

SUMMARY

[0006] In one embodiment, there is provided a method of sharing computing resources, the method comprising: distributing a virtual machine layer amongst a plurality of compute units located on an internal network and supplying surplus hardware resources from the plurality of compute units to the virtual machine layer. The surplus hardware resources are available on the virtual machine layer and accessible by the plurality of compute units located on the internal network.

[0007] In another embodiment, there is provided a system of sharing computing resources, the system comprising: an internal network having a plurality of compute units connected thereto; and a virtual machine layer distributed amongst the plurality of compute units. The plurality of compute units may have surplus hardware resources which are supplied from the plurality of compute units to the virtual machine layer; and the surplus hardware resources are available on the virtual machine layer and accessible by the plurality of compute units located on the internal network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Embodiments will now be described with reference to the appended drawings wherein:

[0009] FIG. 1 is a representation of a typical virtual machine environment hosted on a single physical machine;

[0010] FIG. 2 is a representation of a typical distributed computing environment;

[0011] FIG. 3 is a representation of a plurality of compute units on an internal network;

[0012] FIG. 4 is a representation of the internal network of FIG. 2 having a cloud network connected thereto;

[0013] FIG. 5A is a schematic representation of a first step in providing burstable distributed computing;

[0014] FIG. 5B is a schematic representations of a second step in providing burstable distributed computing;

[0015] FIG. 5C is a schematic representations of a third step in providing burstable distributed computing;

[0016] FIG. 6 is a chart demonstrating the provisioning of local hardware resources of a single compute unit;

[0017] FIG. 7 is an embodiment use of resources for an example network of two nodes;

[0018] FIG. 8 is a display of a virtual machine on two connected nodes;

[0019] FIG. 9 is a flowchart outlining the installation process for the distributed network;

[0020] FIG. 10A is a flowchart outlining the steps user can take in order to execute tasks on node machines; and

[0021] FIG. 10B is a flowchart outlining the process used by the software in order to execute tasks on node machines.

DETAILED DESCRIPTION

[0022] A system for sharing computing resources is provided herein. A virtual machine can be distributed amongst a plurality of compute units (‘nodes’) in order to complete a variety of general compute tasks. The compute units are internally connected to a network such as a local area network (LAN), a virtual private network (VPN), and the like. The internally connected compute units supply directly provisioned hardware (CPU, RAM, GPU, TPU, QPU, etc.) to the distributed virtual machine in order to provide communal computing across the internal network.

[0023] Turning now to the figures, FIG. 1 is a representation of a typical virtual machine environment 100. Typically, a hypervisor 101 is installed on top of a single host machine 102 having a number of physical hardware resources 103. Physical hardware resources 103 include RAM (random access memory), CPU (central processing unit), I/O (input output), GPU (graphical processing unit), disk space including storage or memory. Furthermore, various other types of physical hardware resources 103 are known such as QPU (quantum processing unit) and TPU (tensor processing unit). A hypervisor 101 is typically a computer software, firmware or hardware that creates and runs virtual machines (VMs). The VMs 104 are typically machines running an operating system (OS) 104a and can have their own virtual computing resources 105 such as CPU, I/O, RAM, disk, etc. Multiple guest operating systems and server applications can be loaded on top of the virtual hardware 105. Each of the guest operating systems 104a can then compute any number of individual, different tasks. Thus, virtual servers can allow several operating systems 104a and applications 104b to run on the basis of the shared physical hardware 103. One limitation of typical virtualization environment 100 is that the virtual machines 104 are limited by the physical hardware resources 104 of the single host machine 102. That is, the physical hardware resources 103 (i.e. the RAM, CPU, I/O, storage, GPU, etc.) of the single host machine 102 are shared between the multiple virtual machine instances 104.

[0024] FIG. 2 is a representation of a typical distributed computing environment 200. A plurality of computers (i.e. worker nodes 201) are connected to the master node 202 in order to solve a single, large compute task. That is, the master node can split the large task into a plurality of smaller computing tasks and distribute the divided task among the plurality of worker nodes 201. It can be appreciated that any number of worker nodes can be acceptable, i.e. a first worker node 201a completes a first task, a second worker node 202b completes a second task, and an "nth" worker node 201n presumably completes the last task provided by the master node 202. Each worker node 201 is then only required to complete a portion of the large task. The master node 202 may then compile the individual tasks completed by the worker nodes 201 to produce the desired output of the single complex task. Each of the worker nodes 201 has its own set of physical hardware resources 103 (i.e. the RAM, CPU, I/O, storage, GPU, etc.). Furthermore, the physical hardware resources 103 are not shared between the worker nodes 201. However, the distributed computing system may not be useful for users who wish to compute individual, different tasks of varying complexities using the plurality of individual computers. That is, the worker nodes 201 share the task to be completed, and work in parallel to complete the task provided by the master node 202.

[0025] FIG. 3 is a representation of a plurality of compute units 301 on an internal network 300. A compute unit 301 is defined herein as a device having at least one or more of the physical hardware resources 103. Each compute unit 301 may include RAM, CPU, I/O, storage, GPU, QPU, TPU, and the like. In this embodiment, each compute unit 301 is connected or interconnected to an internal network 302 such as a LAN (local area network), a WAN (wide area network) or a VPN (virtual private network). It can be appreciated that the compute units 301 are not necessarily tied to hardwired connections.

[0026] In one instance, the network 300 may be an office setting, having a plurality of compute units 301 connected thereto. In this instance, the compute units 301 may include laptops, personal computers, printers, servers, routers and the like. In another instance, the network 300 may be a personal use setting such as a home. In this instance, the compute units 301 may include laptops, mobile phones, personal computers, gaming consoles, micro-consoles, and the like. It can be appreciated that each of the compute units 301 will have the physical hardware resources 103 available in varying amounts. That is, a first compute unit 301a may be a laptop having: four CPU cores, 8GB of RAM and 1TB of storage available for use. The second compute unit 301b may be a personal computed having: two CPU cores, 8GB of RAM and 500GB of storage available for use. Furthermore, it can be appreciated that the users of these compute units 301 may be using them for a variety of tasks. For instance, the first compute unit 301a may be used for a high-process task such as gaming and the second compute unit 301b may be used for a low-process task such as word processing. A high-process task is herein defined as a task requiring a high amount of physical hardware resources 103. A low-process task is herein defined as a task requiring a low amount of physical hardware resources 103. It can be appreciated that the tasks may be relative to one another. For example, a high-process task may be considered a low-process task in view of another higher process task.

[0027] FIG. 4 is a representation of the internal network 300 of FIG. 2 having a cloud network 303 connected thereto. The cloud network 303 can be optionally connected to the internal network 300. The cloud network 303 provides additional compute units 304 having their own physical hardware resources 304 (i.e. RAM, CPU, I/O, storage, GPU, QPU, TPU, and the like).

[0028] FIG 5A, 5B and 5C provide schematic representations of the system and method for providing burstable distributed computing. In FIG. 5A, a VM image 501, or “virtual layer”, is shared across a plurality of compute units 301 located in an internal network 300. The virtual layer 501 is uniform across all machines and runs in the background of each compute unit 301. Before the VM image 501 is shared across the internal network 300, all of the physical hardware resources 503 are provisioned locally at each of the compute units 301. The physical hardware resources 503 include RAM, CPU, I/O, storage, GPU, QPU, TPU, and the like. At this time, the compute units 301 may be completing a variety of tasks having different processing requirements. For example, a first compute unit 301a may be computing a high-process task such as gaming; a second compute unit 301b may be computing a low-process task such as word processing; a third compute unit 301c may be computing a very high-process task such as 3D image modelling; and a fourth compute unit may be sitting idle.

[0029] The distribution of physical hardware resources is possible once the virtual layer is installed across the devices in the network. In FIG. 5B, each of the compute units 301 passes any unused physical hardware resources 503a to the virtual layer 501. For instance, the first compute unit 301a may require 60% of its physical hardware resources for gaming; the second compute unit 301b may require 40% of its physical hardware resources for word processing; the third compute unit 301c may require 80% of its physical hardware resources for 3D image modelling; and the fourth compute unit 301d may require only 20% of its physical hardware resources as it is sitting idle. As such, the surplus physical hardware resources 503a are passed to the virtual layer 501.

[0030] Once the surplus physical hardware resources 503a are passed to the virtual layer 501, they are available to, and accessible by, all the compute units 301 located on the internal network 300. This is shown in FIG. 5C. In example embodiment described, the first compute unit 301a may pass up 40% of its physical hardware resources 503; the second compute unit 301b may pass up 60% of its physical hardware resources 503; the third compute unit 301c may pass up 20% of its physical hardware resources 503; and the fourth compute unit may pass up 80% of its physical hardware resources 503 to the virtual layer 501. The surplus physical hardware resources 503a (“virtual hardware resources”) are then available on the virtual layer 501 and accessible by all the compute units 301 located on the internal network 300. This can be helpful as each of the compute units 301 may utilize as much or as little of the virtual hardware resources 503a available. For instance, it may be beneficial for the third compute unit 301c to utilize more of the virtual hardware resources 503a as it may require more resources to efficiently compute the 3D image modeling task. Thus, the third compute unit 301c would be able to utilize the resources provisioned by the remaining compute units. That is to say when the third compute unit 301c requires an increase in available compute power the amount of available virtual hardware resources 503a (RAM, CPU, I/O, storage, GPU, QPU, TPU, and the like) on the virtual layer 501 are expanded so that some of the surplus compute power from the remaining compute units (301a, 301b, 301d) is used.

[0031] FIG. 6 shows a chart demonstrating the provisioning of physical hardware resources 503 of a single compute unit 301. Each compute unit 301 may include a number of physical hardware resources (CPU 601, RAM 602, storage 603, GPU 604, QPU 605, TPU 605, I/O 605, etc.). The percent of total physical hardware resources available on a single compute unit are depicted in FIG. 6. For instance, if a compute unit has four CPU cores 601, 8GB of RAM 602 and 1TB of storage 603 available for use, then each of these physical hardware resources 503 would comprise the 100% of the total available resources. Once the virtual layer is installed on this compute unit, the provisioning of surplus physical hardware resources is possible. In one embodiment, the compute unit may require only 1 CPU core, 1GB of RAM, and 250GB of storage for computation. Thus, 3 CPU cores, 7GB RAM and 750GB of storage would be made available to the virtual layer 501 in the form of virtual hardware resources 503a. In FIG. 6, the physical hardware resources (503b) are shown using grey shading and the virtual hardware resources (503a) are shown using no colour. The virtual hardware resources 503a may be made available to the virtual layer 501 on-demand or in real time. For example, this can be computed using artificial intelligence or machine learning to determine which compute unit typically requires more compute power than another compute unit. Optionally, the provisioning of resources may be determined in-advance or set-in-stone. For example, a user may set thresholds for each compute unit to provision a set number of resources to the virtual layer 606. In this instance, the user may determine which compute unit typically requires more compute power than another compute unit. Additionally, performance tests or monitoring may be used to determine which compute unit typically requires more compute power than another compute unit.

[0032] FIG. 7 shows an embodiment use of resources for an example network of two nodes. Each compute unit 301 may include a number of physical hardware resources (RAM, CPU, I/O, storage, GPU, QPU, TPU, etc). The percent of total physical hardware resources available on the two compute units 301a, 301b are depicted in the two smaller charts of FIG. 7. For instance, if a first compute unit 301a has four CPU cores, 8GB of RAM and 1TB of storage available for use, then each of these physical hardware resources would comprise the 100% of the total available resources of the first compute unit 301a. If a second compute unit 301b has two CPU cores, 8GB of RAM and 500GB of storage available for use, then each of these physical hardware resources comprises the 100% of the total available resources of the second compute unit 301b. Once the virtual layer 501 is installed on the compute units 301, the provisioning of physical hardware resources 503 is possible. In one embodiment, the first compute unit 301a may require only 1 CPU core, 1GB of RAM, and 250GB of storage for computation. Thus, 3 CPU cores, 7GB RAM and 750GB of storage would be made available to the virtual layer 501 in the form of virtual hardware resources 503a. In the same instance, the second compute unit 301b may require only 1 CPU core, 500MB of RAM, and 250GB of storage for computation. Thus only 1 CPU core, 7.5GB RAM and 250GB of storage would be made available to the virtual layer 501 in the form of virtual hardware resources 503a. In FIG. 7, the physical hardware resources 503b of the first compute unit 301a are shown using grey diagonal shading, the physical hardware resources 503b of the second compute unit 301b are shown using grey wavy shading and the virtual hardware resources 503a are shown using no colour. It can be appreciated that the virtual hardware resources 503a are available for communal use once they are on the virtual layer 501.

[0033] FIG. 8 provides a display of a virtual machine on two connected nodes. A VM image 501, or “virtual layer”, is shared across a plurality of compute units 301 located in an internal network. The virtual layer 501 is uniform across all compute units and runs in the background of each compute unit 301. Before the VM image 501 is shared across the internal network 300, all of the physical hardware resources 503 are provisioned locally at each of the compute units 301. The compute unit 301 users are presented with a window 801, 802 containing a requested software’s display. By interacting with this display 801 802, the user may perform all of the same operations they would be able to, had the software been running on their local compute unit. Additional software instances may also be used by the same user. In the embodiment shown in FIG. 8, a first compute unit 301a is being used for a high-process task such as gaming 801 and the second compute unit 301b is being used for a low-process task such as word processing 802. The second compute unit 301b can pass any surplus physical resources 503 to the virtual layer 501, where the first compute unit 301a can access the resources and compute more efficiently.

[0034] FIG. 9 shows a flowchart outlining the installation process for the communal network. First, the resources to be provisioned are earmarked 901. Then a base virtual machine is created 902 and launched on a first compute unit or node 903. The first node is given a machine ID 904. The machine ID is an identifier used to differentiate between the compute units. A machine ID can be as simple as a letter or sequence of letters and can be more complicated for a larger system involving many compute units. Optionally, a test may be run on the first node to determine a performance score 905 of the node and sent to the internal network. Additional nodes are added 906 by launching the virtual machine on each of the nodes connected on an internal network. Each of the additional nodes is also given a machine ID. A test may optionally be run on the additional nodes to determine a performance score of all of the nodes 907. The performance score of all the nodes is sent to the internal network. The node performance is ranked and ordered based of the performance scores of each node 908. It can be appreciated that the performance score can be based on a computed value or other method such as all equal scores, based on hardware specs, manual assignment or any other suitable method.

[0035] The step 901 is used to mark resources to be provisioned for use in the virtual machine layer. For instance, this may be useful for employees of an organization where each employee is only allotted a limited number of resources to be used locally. The remaining, or surplus resources may be earmarked to be provisioned for use by the virtual layer. The virtual layer may act as one physical machine having a large number of computing resources available for use.

[0036] In one embodiment, the virtual machine system can be based on a single block chain that is distributed across all network computers. The blockchain includes features for both security and fault tolerance. The blockchain system keeps a running log of all compute transactions. In another embodiment, the virtual machine system can be used on a computer clustering software, a toolkit, or a bundle or software.

[0037] Additionally, there are encryption benefits to the virtual layer software. Computing tasks are distributed in such a manner that each compute task runs in isolation. For example, if one of the connected compute units was disconnected, its compute tasks could be distributed to other machines on the network.

[0038] FIG. 10A provides a flowchart outlining the steps user may take in order to execute tasks on node machines. Once the local compute unit (host machine) is turned on 1001, users can be are presented with a borderless window containing only the requested enterprise software's window 1002. By interacting with this window, they can perform any of the same operations they would be able to, had the software been running on their local machine's desktop. This process can be repeated by the same user multiple times for additional software.

[0039] FIG. 10B provides a flowchart outlining the process used by the virtual layer software in order to execute tasks on node machines. Enterprise software such as an application can also be installed on the VM. This can be beneficial in many environments including an office environment. For example, the office IT department may install any enterprise software to the virtual layer. When an employee wishes to use the software, it can be accessed from the employee's compute unit, such as a laptop. The employee compute unit can run the algorithm shown in FIG. 10B. When an application is launched 1003, the compute unit will determine if the application can be run on the virtual layer 1004. If yes, the application is run on the virtual layer using the surplus resources provisioned by the virtual layer 1006. The results of the application can be displayed on the employee's compute unit 1007. If the application cannot be run on the virtual layer, the compute unit will run the application using the locally available resources 1005.

[0040] For simplicity and clarity of illustration, where considered appropriate, reference numerals may be repeated among the figures to indicate corresponding or analogous elements. In addition, numerous specific details are set forth in order to provide a thorough understanding of the examples described herein. However, it will be understood by those of ordinary skill in the art that the examples described herein may be practiced without these specific details. In other instances, well-known methods, procedures and components have not been described in detail so as not to obscure the examples described herein. Also, the description is not to be considered as limiting the scope of the examples described herein.

[0041] It will be appreciated that the examples and corresponding diagrams used herein are for illustrative purposes only. Different configurations and terminology can be used without departing from the principles expressed herein. For instance, components and modules can be added, deleted, modified, or arranged with differing connections without departing from these principles.

[0042] It will also be appreciated that any module or component exemplified herein that executes instructions may include or otherwise have access to computer readable media such as storage media, computer storage media, or data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Computer storage media may include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. Examples of computer storage media include

RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by an application, module, or both. Any application or module herein described may be implemented using computer readable/executable instructions that may be stored or otherwise held by such computer readable media.

[0043] The steps or operations in the flow charts and diagrams described herein are just for example. There may be many variations to these steps or operations without departing from the principles discussed above. For instance, the steps may be performed in a differing order, or steps may be added, deleted, or modified.

[0044] Although the above principles have been described with reference to certain specific examples, various modifications thereof will be apparent to those skilled in the art as outlined in the appended claims.

Claims:

1. A method of sharing computing resources, the method comprising:

distributing a virtual machine layer amongst a plurality of compute units located on an internal network; and

supplying surplus hardware resources from the plurality of compute units to the virtual machine layer;

wherein the surplus hardware resources are available on the virtual machine layer and accessible by the plurality of compute units located on the internal network.

2. The method according to claim 1, wherein the internal network is a virtual private network.

3. The method according to claim 1, wherein the internal network is a local area network

4. The method according to claim 1, wherein the plurality of compute units are selected from the group consisting of: laptops, personal computers, printers, servers, routers, mobile phones, gaming consoles and micro-consoles.

5. The method according to claim 1, wherein the computing resources to be shared are selected from the group consisting of: random access memory (RAM), computer processing unit (CPU), input/output (I/O), storage, graphic processing unit (GPU), quantum processing unit (QPU), and tensor processing unit (TPU).

6. A system for sharing computing resources, the system comprising:

an internal network having a plurality of compute units connected thereto; and

a virtual machine layer distributed amongst the plurality of compute units;

wherein the plurality of compute units have surplus hardware resources which are supplied from the plurality of compute units to the virtual machine layer; and

wherein the surplus hardware resources are available on the virtual machine layer and accessible by the plurality of compute units located on the internal network.

7. The system according to claim 6, wherein the internal network is a virtual private network.

8. The system according to claim 6, wherein the internal network is a local area network

9. The system according to claim 6, wherein the plurality of compute units are selected from the group consisting of: laptops, personal computers, printers, servers, routers, mobile phones, gaming consoles and micro-consoles.

10. The system according to claim 6, wherein the computing resources to be shared are selected from the group consisting of: random access memory (RAM), computer processing unit (CPU), input/output (I/O), storage, graphic processing unit (GPU), quantum processing unit (QPU), and tensor processing unit (TPU).

ABSTRACT

A system of communal computing is provided herein. Communal computing refers to the idea of sharing computing resources amongst devices. A virtual machine can be distributed amongst a plurality of compute units (‘nodes’) in order to complete a variety of general compute tasks. The compute units are internally connected to a network such as a local area network (LAN) or virtual private network (VPN). The internally connected compute units supply surplus hardware (CPU, RAM, GPU, storage etc.) to the distributed virtual machine in order to provide communal computing across the internal network. The surplus physical hardware resources are then available on the virtual layer and accessible by any or all the compute units located on the internal network.