MACHINE MANAGER SERVICE FABRIC

BACKGROUND

[0001] Web-based services include files that are located on web servers along with data that is stored in databases. For example, there are a large number of servers located within different networks to handle the traffic that is directed to the service. Managing and deploying the large number of servers is a time consuming process that requires a large operations staff that is subject to human error.

SUMMARY

[0002] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0003] A machine manager controls the deployment and management of machines for an online service The machines are spread across one or more networks in farms that each may include different configurations. The machine manager is configured to manually/automatically deploy farms, upgrade farms, add machines, remove machines, start machines, stop machines, and the like. The machine manager keeps track of the locations of the machines, the roles of the machines within the networks, as well as other characteristics relating to the machines (e.g. health of the machines). The machines within the network are arranged in farms and are configured to perform one or more roles. Instead of upgrading software on the machines in a farm that are currently handling requests, one or more machines are configured in a new farm with the selected disk images and then the requests are moved from the old farm to the new farm.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] FIGURE 1 illustrates a cloud manager system for managing networks that are associated with an online service, such as a content management service;

[0005] FIGURE 2 shows a cloud manager including managers and associated databases;

[0006] FIGURE 3 shows an exemplary job record stored within a row of a database;

[0007] FIGURE 4 shows an example system for a network including front-end and back-end servers for an online service;

[0008] FIGURE 5 illustrates a computer architecture for a computer;

[0009] FIGURE 6 shows a machine manager controlling and managing a deployment of a large number of servers; and

[0010] FIGURE 7 shows a process for deploying a farm.

DETAILED DESCRIPTION

[0011] Referring now to the drawings, in which like numerals represent like elements, various embodiment will be described.

[0012] Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Other computer system configurations may also be used, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. Distributed computing environments may also be used where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0013] FIGURE 1 illustrates a cloud management system for managing networks that are associated with an online service. System 100 illustrates cloud manager 105 that is connected to and manages different networks potentially distributed across the world. Each of the networks is configured to provide content services for one or more tenants (e.g. clients, customers). The networks may be hosted within a cloud service and/or in an on-premises data center. Cloud manager 105 is used in deploying, configuring and managing the networks. The cloud manager is configured to receive requests through an idempotent and asynchronous application web service application programming interface (API) 150 that can tolerate intermittent network failures.

[0014] As illustrated, cloud manager 105 comprises work manager 110, machine manager 115, application specific manager 120, scripts 130 and a central repository, such as data store(s) 140 (e.g. databases). The functionality that is not included within one of the illustrated managers may reside in some other location of the cloud manager. According to one embodiment, application manager 120 is a SharePoint tenant manager that comprises SharePoint specific logic.

[0015] Work manager 110 manages the execution of tasks and enables scheduling and retry of longer running tasks. Work manager 110 starts jobs stored in job queue 112 and keeps track of running jobs. When a predetermined time has elapsed, work manager 110 may automatically cancel the task and perform some further processing relating to the task. According to one embodiment, the tasks in job queue 112 are executed by work manager 110 by invoking one or more scripts 130. For example, a scripting language such

as Microsoft's PowerShell® may be used to program the tasks that are executed by work manager 110. Each script may be run as a new process. While executing each script as a new process may have a fairly high CPU overhead, this system is scalable and helps to ensure a clean environment for each script execution plus full cleanup when the script is completed.

[0016] Machine manager 115 is configured to manage the physical machines in the networks (e.g. Network 1, Network 2, Network 3). Generally, machine manager 115 understands Networks, Physical Machines, Virtual Machines (VMs), VM Images (VHDs), and the like. The machine manager does not have a strong binding to the specific services running within the networks but keeps track of the various components in the networks in terms of "roles." For example machine manager 115 could be requested through API 150 to deploy a VM of type "Foo" with version 12.34.56.78 on Network 3. In response to a request to cloud manager 105, machine manager 115 locates a suitable Physical Machine that is located on Network 3 and configures the VM according to the VM Image associated with the VM's Role. The physical machine is configured with a VHD of type Foo with version 12.34.56.78 that is stored within a data store, such as data store 140. The images used within the network may also be stored in other locations, such as a local data share for one or more of the networks. Scripts may be run to perform the installation of the VHD on the physical machine as well as for performing any post-deployment configuration. Machine manager 115 keeps track of the configuration of the machines each network. For example, machine manager 115 may keep track of a VM's role (type of VM), state of the VM (Provisioning, Running, Stopped, Failed), version and whether the VM exists in a given farm (which implies their network).

[0017] Scripts 130 is configured to store scripts that are executed to perform work both locally for cloud manager 105 and remotely on one or more of the networks. One or more of the scripts 130 may also be stored in other locations. For example, scripts to be performed on a network (e.g. Network 1, Network 2, Network 3) may be stored locally to that network. The scripts may be used for many different purposes. For example, the scripts may be used to perform configurations of machines in one or more of the networks, changing settings on previously configured machines, add a new VM, add a new database, move data from one machine to another, move tenants, change schemas, and the like. According to one embodiment, the scripts are Microsoft's PowerShell® scripts. Other programming implementations may be used. For example, a compiled and/or early-bound programming language may be used to implement the functionality. Scripting, however,

is a fairly concise language to express many of the tasks that are to be performed. Programming the equivalent in a programming language, such as C#, would often require much more verbose implementations. The scripts are also late-bound, meaning that multiple versions of underlying code-bases can be targeted without having to constantly link to different interface DLLs. Using PowerShell scripts allows a process to be started locally by cloud manager 105 that may in turn start a process on a remote machine (i.e. a physical machine in one of the attached networks). Other techniques may also be used to start a process on a remote machine, such as Secure Shell (SSH) and the like.

[0018] Application specific information that cloud manager 105 is managing is performed by application manager 120. According to one embodiment, the application specific information relates to Microsoft SharePoint®. As such, application manager 120 is configured to know about SharePoint Tenants, Site Collections, and the like.

[0019] Each network may be configured as a dedicated network for a tenant and/or as a multi-tenant network that services more than one client. The networks may include a changing number of physical/virtual machines with their configuration also changing after deployment. Generally, a network may continue to grow as long as the networking limits (e.g. load balancer and network switches) are not exceeded. For example, a network may start out with ten servers and later expand to one hundred or more servers. The physical machines within a network may be assigned a class or type. For example, some of the machines may be compute machines (used for web front ends and app servers) and other machines may be storage machines that are provisioned with more storage than compute machines. According to an embodiment, cloud manager 105 configures the machines within a network with multiple versions of the image files. According to an embodiment, farms usually have a same version of image files.

[0020] According to one embodiment, the software limits are managed by the cloud manager system 100 within the network by virtualizing the machines and managing independently acting "Farms" inside the network. Each network may include one or more farms (e.g. see Network 1). According to one embodiment, a network is considered a single cluster of network load balanced machines that expose one or more VIP (Virtual IP) to the outside world and can route that traffic to any of the machines within the network. The machines in the network generally are tightly coupled and have minimum latencies (i.e. < 1ms ping latency).

[0021] Farms are the basic grouping of machines used to coordinate applications that need tightly bound relationships. For example, content farms may be deployed within

each of the networks for a content management application, such as Microsoft SharePoint®. Generally, the set of machines in each of the farms provide web service and application server functions together. Typically, the machines inside the farm are running the same build of an application (i.e. SharePoint) and are sharing a common configuration database to serve specific tenants and site collections.

[0022] Farms can contain heterogeneous sets of virtual machines. Cloud manager 105 maintains a "farm goal" within data store 140 which is a target number of machines of each role for each farm. Some roles include Content Front End, Content Central Admin, Content Timer Service, Federated Central Admin, Federated App Server etc. For example, content farms are the basic SharePoint farm that handles incoming customer requests. Federated Services farms contain SharePoint services that can operate cross farms such as search and the profile store. Farms may be used for hosting large capacity public internet sites. Some farms may contain a group of Active Directory servers and a Provisioning Daemon. Cloud manager 105 automatically deploys and/or decommissions virtual machines in the networks to help in meeting the defined target. These farms goals may be automatically and/or manually configured. For example, the farm goals may change to respond to changes in activity and capacity needs. Network Farm- there is one network farm per Network that contains all the VM roles that scale out easily as a resource to the whole Network.

[0023] The Cloud Manager Web Service APIs 150 are designed to work in the context of a massively scalable global service. The APIs assume that any network request might fail and/or hang in transit. Calls to cloud manager 105 are configured to be idempotent. In other words, the same call may be made to cloud manager 105 multiple times (as long as the parameters are identical) without changing the outcome.

[0024] Cloud manager 105 is designed to do very little processing (<10ms, <50ms) before returning a response to any given request. Cloud manager 105 maintains records to keep track of current requests. For example, cloud manager 105 updates records in a local database and if necessary schedules a "job" to perform more lengthy activity later.

[0025] Cloud manager keeps track of Images (such as Virtual Disk Images) that are the templates used to deploy new machines within a network. The Image references may be stored in a database, such as database 140, and/or in some other location. The images may be stored in one or more shared data stores that are local to the network(s) on which the image will be deployed. According to one embodiment, each Image includes a virtual machine (VM) role type that specifies the type of VM it can deploy, the number of

processors that it should use, the amount of RAM that it will be assigned, a network ID used to find a nearby install point (so they don't get copied repeatedly over the cross data-center links) and a share path that the deployment code can use to access the VHD.

[0026] Generally, machines in the networks being managed by cloud system 100 are not upgraded in the traditional manner by downloading data and incorporating the data into the existing software on the machine. Instead, machines are updated by replacing a VHD with an updated VHD. For example, when a new version of software is needed by a farm, a new farm is deployed that has the new version installed. When the new farm is deployed, the tenants are moved from the old farm to the new farm. In this way, downtime due to an upgrade is minimized and each machine in the farm has a same version that have been tested. When a virtual machine needs to be upgraded, the VM on the machine may be deleted and replaced with the VM that is configured to run the desired service.

[0027] While upgrades to existing software are not optimal, some servers within the networks do utilize the traditional update procedure of an in-place upgrade. For example, Active Directory Domain Controllers are upgraded by updating the current software on the server without completely replacing an image on the machine. The cloud manager may also be upgraded in place in some instances.

[0028] FIGURE 2 shows a cloud manager including managers and associated databases. As illustrated, cloud manager 200 comprises work manager 210, work database 215, machine manager 220, machine database 225, tenant manager 230, tenant database 235, secrets database 245 and web service APIs 240.

[0029] Generally, databases used within a cloud management system (e.g. system 100) are sized to enable high performance. For example, a database (such as work database 215, machine database 225, tenant database 235 and secrets database 245) may not exceed a predefined size limit (e.g. 30 GB, 50 GB, 100 GB, and the like). According to an embodiment, a database is sized such that it is small enough to fit in-memory of a physical machine. This assists in high read I/O performance. The size of the database may also be selected based on performance with an application program, such as interactions with a SQL server. The databases used in the farms may also be sized to enable high performance. For example, they may be sized to fit in-memory of the host machine and/or sized such that backup operations, move operations, copy operations, restore operations are generally performed within a predetermined period of time.

[0030] Cloud manager 200 divides the cloud manager data into four databases. The work database 215 for the work manager. The machine database 225 for the machine manager 220. The tenant database 235 for the tenant manager 230 and a secrets database 245 for storing sensitive information such as system account and password information, credentials, certificates, and the like. The databases may be on the same server and or split across servers. According to an embodiment, each database is mirrored for high availability and is a SQL database.

[0031] Cloud manager 200 is configured to interact with the databases using a reduced set of SQL features in order to assist in providing availability of the cloud manager 200 during upgrades of the databases. For example, foreign keys or stored procedures are attempted to be avoided. Foreign keys can make schema changes difficult and cause unanticipated failure conditions. Stored procedures place more of the application in the database itself.

[0032] Communications with the SQL servers are attempted to be minimized since roundtrips can be expensive compared to the cost of the underlying operation. For example, it is usually much more efficient if all of the current SQL server interactions to a single database are wrapped in a single round-trip.

[0033] Constraints are rarely used within the databases (215, 225, 235). Generally, constraints are useful when it helps provide simple updates with the right kind of error handing without extra queries. For example, the fully qualified domain name (FQDN) table has a constraint placed on the “name” to assist in preventing a tenant from accidentally trying to claim the same FQDN as is already allocated to a different tenant.

[0034] Caution is used when adding indices. Indices typically improve read performance at the cost of extra I/Os for write operations. Since the data within the databases is primarily RAM resident, even full table scans are relatively fast. According to an embodiment, indices may be added once the query patterns have stabilized and a performance improvement may be determined by proposed indices. According to an embodiment, if adding the index will potentially take a long time the "ONLINE=ON" option may be specified such that the table isn’t locked while the index is initially built.

[0035] According to an embodiment, upgrades to databases within the cloud manager may be performed without causing downtime to the cloud manager system. In other words, even during an upgrade of the cloud manager, the cloud manager continues processing received requests. As such, changes made to the schema are to be compatible with the previous schema. The SQL schema upgrade is run before the web servers used

by the cloud manager are upgraded. When the web servers are upgraded they can start to use the new features enabled in the database. Database upgrades are limited such that operations involved in the upgrade are quick and efficient. For example, tables may be added and new nullable columns may be added to existing columns. New columns may be added at the end of a table. Generally, time consuming operations to the databases are avoided. For example, adding a default value to a newly added column at creation time may be a very time consuming operation when there is a large amount of data. Adding a nullable column, however, is a very quick operation. As discussed above, adding new indices are allowed, but caution should be taken when adding a new constraint to help ensure sure that the schema upgrade won't break with the existing data. For example, when a constraint is added it may be set to a state that is not checked and avoids a costly validation of existing rows and potential errors. Old tables and unused columns are removed after a new version is being used and the cloud manager is not accessing those tables and columns.

[0036] Generally, a single row in each of the databases is used to indicate a task and/or a desired state. For example, the tenant database 235 includes a single row for each tenant. A given tenant may include a Required Version record. This record is used to help ensure that the tenant is placed on a farm running the required version. For example, for tenant 1 to stay on SharePoint 14 SP1, the required version for tenant could be set to "14.1." and any version including 14.1 would match and any other versions (e.g. 14.2.xxxx) would not match. The tenant records may include other items such as authorized number of users, quotas (e.g. allowed total data usage, per user data usage, etc.), time restrictions, and the like. Some organization might have multiple tenants that represent different geographies, organizations or capabilities. According to an embodiment, tenants are walled off from each other without explicit invitation of the users (via extranet or other features).

[0037] According to one embodiment, each tenant is locked into a specific network. Tenants are kept localized to a small set of databases. A tenant is either small (smaller than would fill one database) in which case it is in exactly one database, shared with other tenants. This implies that all the tenants sharing that database need to upgrade at the same time. When a tenant grows larger it may be moved to its own dedicated database(s) and now might have more than one, but is not sharing databases with other tenants. Maintaining a large tenant in one or more dedicated databases helps in reducing a number of databases that are needed to be upgraded simultaneously in a single upgrade.

[0038] Similarly, the work database 215 includes a single row for each job. The machine database 225 may include a row for each physical machine, VM, farm, and the like. For example, machine manager database 225 may include a version string. According to an embodiment, each VHD, Farm, and VM within a network has an associated version string.

[0039] According to one embodiment, the cloud manager includes a simple logging system that may be configured to record a log entry for each web service call. A logging system may be implemented that includes as few/many features as desired. Generally, the logging system is used for measuring usage and performance profiling.

[0040] According to an embodiment, the Web Service APIs 240 are built using SOAP with ASP.net. The various Web Methods in the APIs follow two main patterns- Gets and Updates. Generally, the update methods take a data structure as the input and return the same structure as the output. The output structure returns the current state of the underlying object in the database, potentially differing from the input object if validation or other business logic changed some properties or else with additional properties filled in (for example record IDs or other values calculated by the cloud manager). The update methods are used for initial object creation as well as subsequent updates. In other words, callers to the web service APIs 240 can simply request the configuration they want and they don't need to keep track of whether the object already exists or not. In addition this means that updates are idempotent in that the same update call can be made twice with the identical effect to making it only once. According to an embodiment, an update method may include a LastUpdated property. When the LastUpdated property is present, the cloud manager 200 rejects the Update if the value of LastUpdate does not match the one currently stored in the database. Some Update methods include properties that are set on the first invocation of the method and are not set on other invocations of the method.

[0041] Cloud manager 200 is configured to avoid the use of callbacks. Since callbacks may be unreliable, clients interacting with cloud manager 200 may check object status using a web service API when they want to check a status of an update. According to an embodiment, a call to an update method causes cloud manager 200 to set the state of the underlying object to "Provisioning" and when the updates are completed the state is set to "Active".

[0042] FIGURE 3 shows an exemplary job record stored within a row of a database. As illustrated, record 300 comprises job identifier 302, type 304, data 306, owner 308, step 310, last run 312, expire time 314, next time 316, state 318 and status 320.

[0043] Generally, for each task that is requested to be performed, the cloud manager creates a record in database 350 (e.g. work database 215 in FIGURE 2).

[0044] Job identifier 302 is used to specify a unique identifier for the requested task.

[0045] Type 304 specifies the task to perform. For example, the type may include a name of the script to be executed. For example, when the task is to run the script named "DeployVM.ps1" then the data 306 may include the identifier (e.g. "-VMID 123"). This allows new task types to be added to the system without requiring any changes to compiled or other binary parts of the system.

[0046] Data 306 is used to store data that is associated with the task. For example, the data may be set to the tenant, machine, network, VM, etc. on which the task is to be performed. The data 306 may also store one or more values to which a value in a database is set. The process running the task may look to the job record to see what value the desired number of machines is set to. The script uses the value in the database to perform the operation.

[0047] Owner 308 specifies a process/machine that is executing the process. For example, when a cloud manager machine starts execution of a job, the machine updates the owner 308 portion of the record with an ID of the machine.

[0048] Step 310 provides an indication of a step of the current script. For example, the script may divide a task into any number of steps. As the process completes a step of the script, step 310 is updated. A process may also look at step 310 to determine what step to execute in the script and to avoid having to re-execute previously completed steps.

[0049] Last run 312 provides a time the script was last started. Each time a script is started, the last run time is updated.

[0050] Expire time 314 is a time that indicates when the process should be terminated. According to an embodiment, the expire time is a predetermined amount of time (e.g. five minutes, ten minutes…) after the process is started. The expire time may be updated by a requesting process through the web service API.

[0051] Next time 316 is a time that indicates when a task should next be executed. For example, a process may be stopped after completion of a step and be instructed to wait until the specified next time 316 to resume processing.

[0052] State 318 indicates a current state and Status 320 indicates a status of a job (e.g. Created, Suspended, Resumed, Executing, Deleted).

[0053] Duplicate rows in the database can be removed before they are performed if they have the same task type and data values. For example, multiple requests may be made to perform the same task that are stored in multiple rows of the database.

[0054] A job can have one or more locks 355 associated with it. If locks are not available then a job will not be scheduled to run until the locks are available. The locks may be configured in many different ways. For example, the locks may be based on a mutex, a semaphore, and the like. Generally, a mutex prevents code from being executed concurrently by more than one thread and a semaphore restricts a number of simultaneous uses of a shared resource up to a maximum number. According to an embodiment, a lock is a character string that represents a resource. The resource may be any type of resource. For example, the lock may be a farm, a machine, a tenant, and the like. Generally, the locks are used to defer execution of one or more tasks. Each job may specify one or more locks that it needs before running. A job may release a lock at any time during its operation. When there is a lock, the job is not scheduled. A job needing more than one lock requests all locks required at once. For example, a job already in possession of a lock may not request additional locks. Such a scheme assists in preventing possible deadlock situations caused by circular lock dependencies amongst multiple jobs.

[0055] FIGURE 4 shows an example system 400 for a network including front-end and back-end servers for an online service. The example system 400 includes clients 402 and 404, network 406, load balancer 408, WFE servers 410, 412, 414 and back-end servers 416-419. Greater or fewer clients, WFEs, back-end servers, load balancers and networks can be used. Additionally, some of the functionality provided by the components in system 400 may be performed by other components. For example, some load balancing may be performed in the WFEs.

[0056] In example embodiments, clients 402 and 404 are computing devices, such as desktop computers, laptop computers, terminal computers, personal data assistants, or cellular telephone devices. Clients 402 and 404 can include input/output devices, a central processing unit ("CPU"), a data storage device, and a network device. In the present application, the terms client and client computer are used interchangeably.

[0057] WFEs 410, 412 and 414 are accessible to clients 402 and 404 via load balancer 408 through network 406. As discussed, the servers may be configured in farms. Back-end server 416 is accessible to WFEs 410, 412 and 414. Load balancer 408 is a dedicated network device and/or one or more server computers. Load balancer 408, 420, WFEs 410, 412 and 414 and back-end server 416 can include input/output devices, a central

processing unit ("CPU"), a data storage device, and a network device. In example embodiments, network 406 is the Internet and clients 402 and 404 can access WFEs 410, 412 and 414 and resources connected to WFEs 410, 412 and 414 remotely.

[0058] In an example embodiment, system 400 is an online, browser-based document collaboration system. An example of an online, browser-based document collaboration system is Microsoft Sharepoint® from Microsoft Corporation of Redmond, Washington. In system 400, one or more of the back-end servers 416-419 are SQL servers, for example SQL Server from Microsoft Corporation of Redmond, Washington.

[0059] WFEs 410, 412 and 414 provide an interface between clients 402 and 404 and back-end servers 416-419. The load balancers 408, 420 direct requests from clients 402 and 404 to WFEs 410, 412 and 414 and from WFEs to back-end servers 416-419. The load balancer 408 uses factors such as WFE utilization, the number of connections to a WFE and overall WFE performance to determine which WFE server receives a client request. Similarly, the load balancer 420 uses factors such as back-end server utilization, the number of connections to a server and overall performance to determine which back-end server receives a request.

[0060] An example of a client request may be to access a document stored on one of the back-end servers, to edit a document stored on a back-end server (e.g. 416-419) or to store a document on back-end server. When load balancer 408 receives a client request over network 406, load balancer 408 determines which one of WFE server 410, 412 and 414 receives the client request. Similarly, load balancer 420 determines which one of the back-end servers 416-419 receive a request from the WFE servers. The back-end servers may be configured to store data for one or more tenants (i.e. customer).

[0061] Referring now to FIGURE 5, an illustrative computer architecture for a computer 500 utilized in the various embodiments will be described. The computer architecture shown in FIGURE 5 may be configured as a server, a desktop or mobile computer and includes a central processing unit 5 ("CPU"), a system memory 7, including a random access memory 9 ("RAM") and a read-only memory ("ROM") 11, and a system bus 12 that couples the memory to the central processing unit ("CPU") 5.

[0062] A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM 11. The computer 500 further includes a mass storage device 14 for storing an operating system 16, application programs 10, data store 24, files, and a cloud program 26 relating to execution of and interaction with the cloud system 100.

[0063] The mass storage device 14 is connected to the CPU 5 through a mass storage controller (not shown) connected to the bus 12. The mass storage device 14 and its associated computer-readable media provide non-volatile storage for the computer 500. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, the computer-readable media can be any available media that can be accessed by the computer 100.

[0064] By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes 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. Computer storage media includes, but is not limited to, RAM, ROM, Erasable Programmable Read Only Memory ("EPROM"), Electrically Erasable Programmable Read Only Memory ("EEPROM"), flash memory or other solid state 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 the computer 500.

[0065] According to various embodiments, computer 500 may operate in a networked environment using logical connections to remote computers through a network 18, such as the Internet. The computer 500 may connect to the network 18 through a network interface unit 20 connected to the bus 12. The network connection may be wireless and/or wired. The network interface unit 20 may also be utilized to connect to other types of networks and remote computer systems. The computer 500 may also include an input/output controller 22 for receiving and processing input from a number of other devices, including a keyboard, mouse, or electronic stylus (not shown in FIGURE 5). Similarly, an input/output controller 22 may provide output to a display screen 28, a printer, or other type of output device.

[0066] As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 14 and RAM 9 of the computer 500, including an operating system 16 suitable for controlling the operation of a networked computer, such as the WINDOWS® operating systems from MICROSOFT® CORPORATION of Redmond, Washington. The mass storage device 14 and RAM 9 may also store one or more program modules. In particular, the mass storage device 14 and the RAM 9 may

store one or more application programs, such as cloud program 26, that perform tasks relating to the cloud system.

[0067] FIGURE 6 shows a machine manager controlling and managing a deployment of a large number of servers for an online service. As illustrated, system 600 shows a cloud manager 605 comprising machine manager 610 that controls the automatic deployment of large numbers of servers in specific topologies across different networks (Network 1, Network 2). While only two networks are shown, many more networks are generally managed (e.g. ten, one hundred, one thousand, ten thousand, and the like). Cloud manager 605 operates and is configured similarly to the cloud manager system shown and described above. According to an embodiment, machine manager 610 is configured to manage and deploy physical and virtual machines that are used in an online content collaboration service, such as a SharePoint Online service. Machine manager 610 may be used to deploy and manage machines for other online services.

[0068] Machine manager 610 understands the physical topology of the networks and keeps track of the location of the physical machines which are members within each of the networks. According to an embodiment, machine manager 610 knows a location of each rack within the networks and each machine that is located within the rack. Machine manager 610 also keeps track of the specific versions of software that is being used on each of the machines as well as the Virtual Machine (VM) Images that are installed on each of the machines. Each VM image corresponds to a different server role.

[0069] Machine manager 610 is also configured to determine the roles for each of the machines within each of the farms. This determination may be performed manually and/or dynamically. For example, a farm may be initially defined to include fifteen physical machines, five of which are located within a content farm 660, five in a federated services farm 665, and five in a SQL farm 670. During operation of the online service, machine manager 610 may collect performance characteristics relating to the farms and machines within a network and dynamically adjust the resources of the network based on the collected performance characteristics. For example, when it is determined that a content farm is overloaded, machine manager 610 may create a job that deploys another machine within the content farm to provide additional bandwidth. The health of the machines/networks may also be monitored. Machine manager 610 may replace machines within a network, direct traffic to a new set of machines, and/or perform some other actions in response to the determination of the health of the machines. For example, if one

or more farms go down, machine manager 610 deploys a new farm and directs the traffic to the newly deployed farms.

[0070] Machine manager 610 keeps track of virtual machines which are the actual servers that do the work of the service and stores this information in a data store, such as machine database 620. As discussed, each of the VMs have a specific role representing the function of the specific server and they are further grouped into farms which are typically a group of machines running the exact same version of the software that work together. According to an embodiment, machine manager 610 stores a table in machine database 620 for each farm that specifies the goals and the roles within the farm. A number of machines for each role and farm is also stored within machine database 620. The number of machines within a farm may be manually configured and/or automatically configured. For example, a background process can monitor the load and determine dynamically the farm goals. The machine manager may start machines and/or stop machines based on the current/anticipated network characteristics. Each farm within a network may be configured with a same number of machines as other farms or a different number of machines. For example, one content farm may include six machines whereas another content farm may use only three machines.

[0071] When new software is available, machine manager 610 manages the deployment of the new software. Generally, individual machines within a farm/network are not upgraded or patched. Instead, machine manager 610 starts jobs managed by the work manager (e.g. work manager 110 in FIGURE 1) that deploys new machines and farms that include the new software. After the new farm is configured with the new software, machine manager 610 moves the traffic/load to the new farm and stops the traffic from going to the old farm.

[0072] Images 640 are configured to store Virtual Hard Disk (VHD) images that are in use and/or are to be deployed on one or more of the machines in one or more of the networks. According to an embodiment, the MICROSOFT® VHD file format is used that specifies a virtual machine hard disk that can reside on a native host file system encapsulated within a single file. Generally, the VHD format is broadly applicable since it is agnostic to the virtualization technology, host operating system, or guest operating system with which it is used. Images that are used within a specific network may be moved to a global share 645 and/or to a network share that is local to a network (e.g. network share 655). Storing the images on a network share saves time in a deployment of images since network communication time is reduced.

[0073] Differential VHDs may also be used. For example, only the differences between the latest version of a VHD and the previous version may be deployed. Different techniques may be used for this differential comparison. For example, Remote Differential Compression techniques may be used to determine the differences and copy only the changes to the network shares. This allows a process to speed up deployment and updates to machines within a network since the full copy of the VHD does not need to be copied. Deltas within a network may also be employed. Deltas are sent as files and then within the local machine/local network the files can be used to create the complete VHD.

[0074] According to an embodiment, machine manager 610 doesn't actually install the software on the machines. Instead, as discussed previously, a job is placed in a job queue that when executed performs actions to complete the task. Once the images are installed on the machines, machine manager 610 starts the machine running. Zero or more role-specific scripts that can do role-specific customizations of the deployment may be run after the Virtual Machine is started. These scripts may be located within the network's share (e.g. network share 655), a global share 645, scripts 630 or some other location.

[0075] According to an embodiment, the VHDs are non-changeable once they are running on the machines. The operating system files may also be locked down such that they can not be changed. Each farm runs its services in one of two accounts including an application pool account and an administrative account. These accounts are created for each farm and are generated by machine manager 610. The accounts include randomly generated passwords. These passwords may be stored in a secure manner within machine database 620 or some other location. Generally, a human is not supplied with the passwords for the farm. Instead, application accounts are used to run processes relating to the online service (e.g. SharePoint). According to an embodiment, the application pool account is not logged into and there is not a web application interface to access the database that stores the passwords. During configuration of the farm, the process executing the script is provided with the required password(s) to set up and deploy the machine. For example, when a process to deploy a farm is executed, the passwords are supplied to the deploy farm process so that the farm may be configured and started. During the installation of the images and any customizing of the configuration, the farm is not connected to the online service and as such does not have any traffic. Once traffic is directed to the farm, the machine is locked down and the passwords are not accessible on the machine or through a web interface. According to an embodiment, unique accounts are automatically generated as the farm is deployed, with no human being allowed to ever

see the actual system account password. While the farm is processing requests, machine manager 610 may monitor the machines and the VHD images to determine if they are attempted to be changed. If they are, machine manager 610 may deploy a new farm with newly generated system accounts to replace the potentially comprised farm. In this way, any code that was placed onto the old farm is not replicated to the newly deployed farm.

[0076] As discussed above, when a farm is upgraded (e.g. farm 680), a new farm (e.g. farm 681) is provisioned and deployed. Generally, once farm 681 has been configured with the specified VHDs and is running, the traffic that was being directed to farm 680 is now directed to farm 681. Data that is associated with farm 680 may be kept for a period of time such that traffic may be returned to farm 680 if a problem is detected (See FIGURE 7 and related discussion for more detail on the farm deployment process). The databases that are associated with the farms are copied to the new farms such that there is minimal time in which a database is not available for writing (See FIGURE 7).

[0077] During the upgrade, the new farm may be configured to better handle the traffic that was being directed to the old farm. For example, it may be determined from monitoring of farm 680, that more SQL machines are needed to handle the load. During the upgrade from farm 680 to farm 681, machine manager 610 may add/remove machines from the farm.

[0078] Referring now to FIGURE 7 a process for managing and deploying machines in an online service will be described.

[0079] When reading the discussion of the routines presented herein, it should be appreciated that the logical operations of various embodiments are implemented (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations illustrated and making up the embodiments described herein are referred to variously as operations, structural devices, acts or modules. These operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof.

[0080] FIGURE 7 shows a process for deploying a farm. After a start operation, the process 700 flows to operation 710, were a determination is made to deploy a farm. The farm may be deployed for many different reasons. For example, one or more farms may be deployed to run a different version of software from existing farms, a new network may

be deployed, equipment may fail, and the like. The determination may occur from a request through an API, such as a Web Service API as described above, and/or may be determined automatically. For example, when the goals of a farm change, a new farm may be manually/automatically deployed.

[0081] Moving to operation 720, the provisioning of the machines in the new farm is started. According to an embodiment, the provisioning of the machines is performed asynchronously such that the machine manager can continue to perform other actions. The provisioning may include many different steps. For example, the VHDs may be moved to the network(s) that are going to use them such that copy operations are performed within a network rather than across networks. A process also installs the VHDs on the machines whose roles match the VHDs, perform any customizations to the installations and starts the machines.

[0082] Flowing to operation 730, the data from the old farm is backed up. For example, the database in the farm being upgraded is backed up. The backup may be performed while the machines in the new farm are being provisioned.

[0083] Moving to operation 740, the data is copied from the old farm to the new farm. The farm being upgraded continues to receive and process requests during the upgrade process. According to an embodiment, the data is copied on a farm by farm basis within a network. For example, content farm data is copied before federated services farm data and before SQL farm data. The data may also be copied concurrently.

[0084] Transitioning to operation 750, any upgrades to the new farm are performed. For example, the databases in the new farm may be upgraded to handle different operations as compared to the old farm. As discussed above, the new farm is not receiving any of the requests during the upgrade process.

[0085] Moving to operation 760, the changes made to the data on the old farm since the backup was initially performed is obtained and added to the data copied to the new farm. During this operation, the data on the old farm is marked as read only such that for a short period of time, writes are not allowed to the data on the old farm. Since the time period between the full backup and the transactional backup is relatively short (e.g. a few minutes) the time period that the database is unavailable for writing is also short.

[0086] Flowing to operation 770, when the provisioning and deployment of the new farm is successful, the traffic from the old farm is directed to the new farm. At this point, the newly deployed farm receives and processes all of the requests that were previously directed to the old farm. If a problem is detected during the deployment of the farm, the

old farm may continue to be utilized. Further, traffic may also be redirected back to the old farm. The process then moves to an end block and returns to processing other actions.

[0087] The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

WHAT IS CLAIMED IS:

1. A method for managing servers in an online service, comprising:

storing by a central service a configuration of machines in different networks for an online service, wherein the configuration includes a location of each of the machines that includes a rack location of the machine and roles of the machines in the networks; wherein the machines are arranged within farms within each of the networks, wherein the role is used to determine one or more virtual machines to install the machines, wherein the configuration stores a goal for each of the farms and a role for each of the farms;

determining when to deploy a new farm within one or more of the networks;

from the central service automatically starting a process to deploy the new farm within one of the networks; wherein deploying the farm comprises provisioning software on the machines within the farm using virtual hard disk (VHD) images, starting the machines, and directing traffic to the new farm.

2. The method of Claim 1, wherein a VHD image is used for each role within the farm, wherein each machine within the farm uses a same version of the VHD image for each role.

3. The method of Claim 1, wherein determining when to deploy the new farm comprises at least one of: determining when to upgrade the software on one or more of the farms; determining when a farm has failed; and determining when to reconfigure a configuration of a farm.

4. The method of Claim 1, wherein deploying the new farm comprises backing up data for an old farm being replaced within a same network of the new farm while continuing to allow reads and writes to the data being backed up; copying the backed up data to the new farm while continuing to allow reads and writes to the data previously backed up in the old farm; and updating the data copied to the new farm with any data that changed on the old farm since a time the data was backed up on the old farm.

5. The method of Claim 2, further comprising copying the VHD images from the central service to the network that includes the new farm where the VHD images are to be installed and when a previous version of the VHD image exists on the network then copying a file that includes only the differences between the VHD image and the previous version of the VHD image.

6. The method of Claim 1, wherein the farm goals are dynamically determined by calculating machine performance for machines within the farm.

7. A computer-readable storage medium having computer-executable instructions for deploying farms for an online service, comprising:

storing by a central service that manages an online content management services a configuration of machines in different networks for an online service, wherein the configuration includes a location of each of the machines that includes a rack location of the machine and roles of the machines in the networks; wherein the machines are arranged within farms within each of the networks, wherein the role is used to determine one or more virtual machines to install the machines, wherein the configuration stores a goal for each of the farms and a role for each of the farms;

determining when to deploy a new farm within one or more of the networks; wherein deploying the new farm replaces one of the farms within one of the networks; wherein determining when to deploy the new farm comprises at least one of: determining when to upgrade the software on one or more of the farms; determining when a farm has failed; and determining when to reconfigure a configuration of a farm;

from the central service automatically starting a process to deploy the new farm within one of the networks; wherein deploying the farm comprises provisioning software on the machines within the farm using a same version of virtual hard disk (VHD) images, starting the machines, and directing traffic to the new farm.

8. The computer-readable storage medium of Claim 7, wherein deploying the new farm comprises backing up data within databases for an old farm being replaced within a same network of the new farm while continuing to allow reads and writes to the databases being backed up; copying the backed up databases to the new farm while continuing to allow reads and writes to the data previously backed up in the old farm; and updating the databases copied to the new farm with any data that changed on the databases in the old farm since a time the databases were backed up on the old farm.

9. The computer-readable storage medium of Claim 7, wherein deploying the new farm comprises locking down the VHD images and operating system files on the machines within the network after the machines are running with the VHD images such that the VHD images and the operating system files are not changeable.

10. A system for deploying farms for an online service, comprising:

a network comprising servers and databases; wherein the servers are arranged in farms that each perform a role; wherein the farms comprise content farms, federated services farms and SQL farms;

a processor and a computer-readable storage medium;

an operating environment stored on the computer-readable medium and executing on the processor; and

software that is operative to:

store a configuration of machines in farms of different networks for the online service, wherein the configuration includes a location of each of the machines that includes a rack location of the machine and roles of the machines in the networks; wherein the role is used to determine one or more virtual machines to install the machines, wherein the configuration stores a goal for each of the farms and a role for each of the farms;

determine when to deploy a new farm within one or more of the networks; and

automatically starting a process to deploy the new farm within one of the networks; wherein deploying the farm comprises provisioning software on the machines within the farm using a same version of virtual hard disk (VHD) images , starting the machines, and directing traffic to the new farm.