SYSTEM AND METHOD FOR COMPUTER CLUSTER VIRTUALIZATION USING DYNAMIC BOOT IMAGES AND VIRTUAL DISK

TECHNICAL FIELD

This disclosure relates generally to the field of data processing and, more specifically, to a system and method for computer cluster virtualization using dynamic boot images and virtualized disk access.

BACKGROUND OF THE INVENTION

Typically, enterprise applications are executed on dedicated compute resources. Often, an enterprise will include a variety of computing environments for different instances of the application such as production, test, and development. These multiple computing environments are typically the same size and capacity as the live or production instance. Moreover, the non-production environments are frequently idle for extended periods of time. This normally results in large amounts of wasted computing resources and labor expense in maintaining and administering these various environments.

Currently, enterprises may use provisioning as an attempt to address these issues. Generally, provisioning is the process of instantiating compute resources to the enterprise application by copying the local disk from a repository to the resource. The resource is then booted with the provisioned operating system and software through a process that normally takes over ten minutes.

SUMMARY OF THE INVENTION

This disclosure provides a system and method for computer cluster virtualization that includes selecting a distributed application. A policy associated with the distributed application is retrieved. One of a plurality of nodes is dynamically selected, possibly based on the policy. Then, a boot image of the selected node is reset based, at least in part, on the retrieved policy, with the boot image being compatible with the distributed application. Then, a virtual disk image is associated with the node. At least a portion of the distributed application is then executed on the reset node using the associated virtual disk image.

According to a first aspect of the present invention, there is provided a method for computer cluster virtualization comprises: selecting a distributed application; retrieving a policy associated with the distributed application; dynamically selecting one of a plurality of nodes; resetting a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and associating a virtual disk image with the selected node based, at least in part, on the retrieved policy; and executing at least a portion of the distributed application on the reset node using the associated virtual disk image.

According to a second aspect of the present invention, there is provided software for computer cluster virtualization operable to: select a distributed application; retrieve a policy associated with the distributed application; dynamically select one of a plurality of nodes; reset a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and associate a virtual disk image with the selected node based, at least in part, on the retrieved policy; and execute at least a portion of the distributed application on the reset node using the associated virtual disk image.

According to a third aspect of the present invention, there is provided a system for computer cluster virtualization comprises: a plurality of computing nodes; and a management node operable to: select a distributed application; retrieve a policy associated with the distributed application; dynamically select one of a plurality of nodes; reset a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and associate a virtual disk image with the selected node based, at least in part, on the retrieved policy; and execute at least a portion of the distributed application on the reset node using the associated virtual disk image.

Embodiments of the present invention have several important technical advantages. For example, one possible advantage of the embodiments of the present invention is that it allows for computing nodes to be reprovisioned on-the-fly to become a member of a virtual cluster for a distributed application, thereby possibly reducing provisioning times to fifteen seconds or less. Another possible advantage of the present disclosure may be a reduction in Information Technology (IT) hardware and maintenance costs by at least thirty percent. Moreover, when an application is not at a peak processing period, idles nodes of that application may be dynamically reallocated or reprovisioned to other distributed applications. Yet another possible advantage is that it provides centralized capacity planning, performance monitoring, and simplified administration. Further, the embodiments of the present invention may allow for better node failure management. Various embodiments of the invention may have none, some, or all of these advantages. Other technical advantages of the embodiments of the present invention will be readily apparent to one skilled in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present disclosure and its advantages, reference is now made to the following descriptions, taken in conjunction with the accompanying drawings, in which:

FIGURE 1 illustrates an example distributed system providing dynamic booting in accordance with one embodiment of the present disclosure; and

FIGURE 2 illustrates an example method for dynamically rebooting a node within one embodiment of the present disclosure.

DETAILED DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram illustrating a distributed computing system 100 for executing software applications 114 and processes using dynamic boot images 131. Generally, system 100 is a scalable distributed computing environment for enterprise or other distributed applications. System 100 provides a scalable, fault-tolerant computing environment, which can dynamically grow based on computing needs and can simultaneously provide computing resources to multiple applications 114 through providing each application 114 with its own scalable virtual cluster. For example, system

100 may include server 102 that is connected, through network 116 to one or more administration workstations or local clients 120. But system 100 may alternatively be a standalone computing environment or any other suitable environment. In short, system 100 is any computing environment that automatically allows nodes 108 to be dynamically allocated on-the-fly as application 114 requirements, parameters, and processing needs change. The term "dynamically," as used herein, generally means that certain processing is determined, at least in part, at run-time based on one or more variables. The term "automatically," as used herein, generally means that the appropriate processing is substantially performed by at least part of system 100. It should be understood that "automatically" further contemplates any suitable user or administrator interaction with system 100 without departing from the scope of this disclosure.

Server 102 comprises any local or distributed computer operable to execute a plurality of applications 114 across one or nodes 108. Generally, server 102 comprises a distributed computer such as a rack-mounted server, blade server, or other distributed server. Nodes 108 comprise any computer or processing device such as, for example, blades, general-purpose personal computers (PC), Macintoshes, workstations, Unix-based computers, or any other suitable devices. Generally, FIGURE 1 provides merely one example of computers or blades that may be used with the disclosure. For example, although FIGURE 1 illustrates one blade server 102 that may be used with the disclosure, server 102 can be implemented using computers other than servers, as well as a server pool. In other words, the present disclosure contemplates computers other than general purpose computers as well as computers without conventional operating systems. As used in this document, the term "computer" is intended to encompass a personal computer, workstation, network computer, or any other suitable processing device. Server 102, or the component nodes 108, may be adapted to execute any operating system including Linux, UNIX, Windows Server, or any other suitable operating system. According to one embodiment, server 102 may also include or be communicably coupled with a remote web server.

Illustrated server 102 includes a management node 104 communicably coupled with a plurality of nodes 108 and operable to execute dynamic boot engine 105. But it will be understood that server 102 and nodes 108 may not include all of the illustrated

components. Management node 104 comprises at least one blade or computing device substantially dedicated to managing server 102 or assisting an administrator. For example, management node 104 may comprise two hot-swappable blades, with one of the two blades or rack-mounted servers being redundant (such as an active/passive configuration).

Dynamic boot engine 105 could include any hardware, software, firmware, or combination thereof operable to dynamically allocate and manage nodes 108 and execute applications 114 using virtual clusters of nodes 108 (or application environments). For example, dynamic boot engine 105 may be written or described in any appropriate computer language including C, C++, Java, Visual Basic, assembler, any suitable version of 4GL, and others or any combination thereof. It will be understood that while dynamic boot engine 105 is illustrated in FIGURE 1 as a single multi-tasked module, the features and functionality performed by this engine may be performed by multiple modules such as, for example, a physical layer module, a virtual layer module, a job scheduler, and a presentation engine. Moreover, dynamic boot engine 105 may be a child or sub-module of another software module without departing from the scope of this disclosure. Therefore, dynamic boot engine 105 comprises one or more software modules operable to intelligently manage nodes 108 and applications 114 based on policies 132.

Generally, dynamic boot engine 105 manages one or more applications 114 by starting and stopping application environments on the individual nodes 108. For example, dynamic boot engine 105 may reset the particular node 108 with a different boot image 131 from boot image file 130, which is specific to or compatible with the desired application environment. In other words, dynamic boot engine 105 supports dynamically booting any suitable application environment on any controlled node 108. Accordingly, dynamic boot engine 105 may also support dynamically setting IP or MAC addresses for the public IP interface on any controlled computer. Dynamic boot engine 105 may also boot any node 108 directly from the network using a network boot protocol or by booting from attached disk storage. Dynamic boot engine 105 may also utilize high speed network access to a virtual local disk image containing the operating system, services, and applications for any controlled computer. It will be understood that dynamic boot engine 105 may start up or shut down application environments based on calendar date and times or using any other predetermined parameter.

Dynamic boot engine 105 may also support various fault tolerance and recovery techniques. For example, boot engine 105 may automatically recover server 102 from single hardware component failures by automatically replacing and dynamically rebooting a replacement node 108 for the failed node 108. Moreover, installing a new node 108 may be facilitated because of dynamic boot engine's 105 ability to automatically recognize the new node 108 and do any required configuration, resetting, or booting.

Nodes 108 comprises any computer, blade, or server operable to execute at least a portion (such as a task or process) of application 114. Illustrated node 108 includes, at a high level, memory 109 and processor 110. Memory 109 may include any memory or database module and may take the form of volatile or non-volatile memory including, without limitation, magnetic media, optical media, random access memory (RAM), read-only memory (ROM), removable media, or any other suitable local or remote memory component. Memory 109 may include any of a variety of local information. Node 108 also includes processor 110. Processor 110 executes instructions and manipulates data to perform the operations of server 102 such as, for example, a central processing unit (CPU) or field programmable gate array (FPGA). Although FIGURE 1 illustrates a single processor 110 in each node 108, multiple processors 110 may be used according to particular needs and reference to processor 110 is meant to include multiple processors 110 where applicable. Processor 110 may include any pointer to a boot image such as, for example, Electronically Erasable Programmable Read-Only Memory (EEPROM) 111. But it will be understood that node 108 may comprise any number of components, configured in any appropriate fashion, without departing from the scope of this disclosure. Node 108 may also include one or more local hard drives for the purposes of providing local temporary file space and virtual memory swap space.

Application 114 may comprise any enterprise or distributed application such as, for example, a database management system (DBMS), financial software, and others. Typically, application 114 is comprised of software written in any suitable language and operable to perform any data processing. But unconventional applications are also within the scope of this disclosure. Applications 114 may run in an application environment, or virtual cluster, which logically defines the environment for application execution. In one embodiment, an application environment comprises i) name and description of the

application environment; ii) minimum / maximum number of nodes 108; iii) software configuration information, such as operating system software version and application 114 software version; and iv) hardware configuration of each node 108 such as boot image, hostname and IP address, custom configuration applied after node 108 booting, virtual local disk image, local file systems, file systems to mount, and network configuration. But it will be understood that any suitable parameter, variable, or characteristic may be used to assist dynamic boot engine 105 with defining, locating, and processing the application environment. For example, the application environment may also include information on application 114 startup, shutdown, and health monitoring.

Server 102 may include interface 115 for communicating with other computer systems, such as client 120, over network 116 in a client-server or other distributed environment. In certain embodiments, server 102 receives boot images 131, virtual local disk images 134, policies 132, or application data 140 from network 116 for storage or processing via high-speed interface 115. Generally, interface 115 comprises logic encoded in software and/or hardware in a suitable combination and operable to communicate with network 116. More specifically, interface 115 may comprise software supporting one or more communications protocols associated with communications network 116 or hardware operable to communicate physical signals.

Network 116 facilitates wireless or wireline communication between computer server 102 and any other computer, such as clients 120. Indeed, while illustrated as residing between server 102 and client 120, network 116 may also reside between various nodes 108 without departing from the scope of the disclosure. In other words, network 116 encompasses any network, networks, or sub-network operable to facilitate communications between various computing components. Network 116 may communicate, for example, Internet Protocol (IP) packets, Frame Relay frames, Asynchronous Transfer Mode (ATM) cells, voice, video, data, and other suitable information between network addresses. Network 116 may also process and route data packets according to any other suitable communication protocol, for example, InfiniBand (IB), Gigabit Ethernet (GE), or FibreChannel (FC). Data packets are typically used to transport data within application data 140. A data packet may include a header that has a source identifier and a destination identifier. The source identifier, for example, a source

address, identifies the transmitter of information, and the destination identifier, for example, a destination address, identifies the recipient of the information. Network 116 may include one or more local area networks (LANs), radio access networks (RANs), metropolitan area networks (MANs), wide area networks (WANs), all or a portion of the global computer network known as the Internet, and/or any other communication system or systems at one or more locations.

Boot table 130 is any disk farm or network file system that includes a plurality of boot images 131. While illustrated as remote, boot images 131 may be preloaded by dynamic boot engine 105 to simplify initialization and installation. Boot image 131 is any form, image, pointer, or reference to at least a portion of the boot drive primary operating system partition. Boot image 131 is typically in binary form. Boot image types include kernel images from a file, kernel images from a block device or floppy disk, or the boot sector of some operating system. For example, a Linux boot image might appear as:

```
0x1B031336,
0x4,
0x90000000,
0x90000200,

0x4,
0x90200,
0x800,
0x800,

0x4,
0x10000,
0x80000,
0x80000,

0x04000004,
0x100000,
0x80000,
0x80000
```

It will be understood that above example boot image 131 is for illustration purposes only and may include none, some, or all of the illustrated elements as well as additional elements not shown. Moreover, boot image 131 may be in a different layout or format than the above example without departing from the scope of this disclosure.

Policies table 132 includes any parameters for managing nodes 108 and applications 114. For example, policies 132 may be for automatically adding or subtracting nodes 108 to application environments. Alternatively or in combination, policies 132 may be used by server 102 to resolve issues between competing applications 114. Generally, policies table 132 may comprise one or more tables stored in a relational database described in terms of SQL statements or scripts. In another embodiment, policies table 132 may store or define various data structures as XML documents, Virtual Storage Access Method (VSAM) files, flat files, Btrieve files, or comma-separated-value (CSV) files. Policies table 132 may also comprise a plurality of tables or files stored on one computer or across a plurality of computers. Moreover, policies table 132 may be local or remote without departing from the scope of this disclosure and store any type of appropriate data. For example, policies table 132 may store individual virtual cluster policies including: i) minimum / maximum number of nodes 108 assigned to an application environment; ii) default number of servers assigned to the application; iii) conditions to dynamically add node 108 to the application environment; iv) conditions to dynamically remove node 108 from the application environment; v) conditions to remove node 108 (such as turning off network access), but leave it up for problem investigation; and vi) conditions under which node 108 should not be removed because application 114 is actively running a transaction or process.

In another example, policies table 132 may include any number of inter-virtual cluster policies such as priority, resource sharing, and preemption policies. Priority typically determines which application environment gets the resources if there is a policy conflict. For example, if the priority of a particular application environment is higher, it may get prioritized access to nodes 108. Resource sharing is often based on defined entitlement of the application environments. For example, each application environment may be granted an entitlement to a percentage of nodes 108. Resource sharing may also be based on computer usage over a sliding window of time. Preemption policies may allow high priority application environments to take over nodes 108 from lower priority application environments.

Virtual local disk image table 133 is any disk farm or network file system that includes a plurality of virtual local disk images 134. While illustrated as remote, virtual

local disk image 134 may be preloaded with the operating system and application software to simplify initialization and installation. Virtual local disk image 134 is any form, image, pointer, or reference to the local disk storage of each virtual node for each application. Virtual local disk image 134 will typically include the operating system, configured services, and installed applications of each application's virtual node. Each virtual local disk image 134 may contain multiple file systems, which may be read-only for sharing between multiple nodes, or modifiable file systems, which are normally specific to an application node. Virtual local disk image 134 may be stored in a hierarchical directory within a traditional file system or may be stored in a recoverable database with a network file system interface provided to the application nodes.

In general, application data 140 is any memory, database, storage area network (SAN), or network-attached storage (NAS) for storing data for applications 114. Application data 140 may comprise one or more tables stored in a relational database described in terms of SQL statements or scripts. In another embodiment, application data 140 may store or define various data structures as XML documents, VSAM files, flat files, Btrieve files, or CSV files. Application data 140 may also comprise a plurality of tables or files stored on one computer or across a plurality of computers. Moreover, application data 140 may be local or remote without departing from the scope of this disclosure.

Client 120 is any device operable to present the user with an administration screen via a graphical user interface (GUI) 122. At a high level, illustrated client 120 includes at least GUI 122 and comprises an electronic computing device operable to receive, transmit, process and store any appropriate data associated with system 100. It will be understood that there may be any number of clients 120 communicably coupled to server 102. Further, "client 120" and "user of client 120" may be used interchangeably as appropriate without departing from the scope of this disclosure. Moreover, for ease of illustration, each client is described in terms of being used by one user. But this disclosure contemplates that many users may use one computer to communicate commands or view graphical presentations using the same GUI 122.

As used in this disclosure, client 120 is intended to encompass a personal computer, touch screen terminal, workstation, network computer, kiosk, wireless data port, cell phone, personal data assistant (PDA), one or more processors within these or other

devices, or any other suitable processing device. For example, client 120 may comprise a computer that includes an input device, such as a keypad, touch screen, mouse, or other device that can accept information, and an output device that conveys information associated with the operation of server 102 or clients 120, including digital data, visual information, or GUI 122. Both the input device and output device may include fixed or removable storage media such as a magnetic computer disk, CD-ROM, or other suitable media to both receive input from and provide output to users of clients 120 through the administration and job submission display, namely GUI 122.

GUI 122 comprises a graphical user interface operable to allow the system (or network) administrator to interface with system 100 to monitor applications 114 or system performance, modify virtual clusters, or any suitable supervisory purpose. Generally, GUI 122 provides the user of client 120 with an efficient and user-friendly presentation of data provided by system 100. GUI 122 may comprise a plurality of customizable frames or views having interactive fields, pull-down lists, and buttons operated by the user. In one embodiment, GUI 122 presents display that presents the various graphical views of application environments or policy screens and receives commands from the user of client 120 via one of the input devices. These graphical views may include i) graphical representations of the current status of application environments, nodal resources, and monitored loads; ii) graphical representations of application environment and nodal loads and usage over time; iii) wizards; and iv) views of which application 114 is running in each application environment and on each node 108. In short, GUI 122 may present any physical and logical status or characteristics of nodes 108 to the system administrator and receive various commands from the administrator.

In one embodiment, GUI 122 may allow an administrator to create, delete, copy, and modify application environments. The administrator may also set up application environment sharing policies, activate and deactivate application environments, monitor states and loads of application environments and nodes 108 using GUI 122. Further, GUI 122 may allow the adding or subtracting of nodes 108 from active application environments. GUI 122 may also present alerts to an administrator based on various system 100 characteristics such as, for example, configurable load levels were reached on node 108 or within an application environment, a node 108 became unavailable,

application environment started or stopped, node 108 was added or subtracted from application environment, server 102 was unable to meet minimum application environment requirements, or a level of service requirement (such as transaction response time) was exceeded.

It should be understood that the term graphical user interface may be used in the singular or in the plural to describe one or more graphical user interfaces and each of the displays of a particular graphical user interface. Therefore, GUI 122 contemplates any graphical user interface, such as a generic web browser, that processes information in system 100 and efficiently presents the results to the user. GUI 122 also contemplates a secured browser operable to communicate via SSL–HTTPS. Server 102 can accept data from client 120 via the web browser (e.g., Microsoft Internet Explorer or Netscape Navigator) and return the appropriate HTML or XML responses using network 116.

In one aspect of operation, dynamic boot engine 105 selects a distributed application 114. Based on one or more associated policies 132, dynamic boot engine 105 may dynamically add or subtract one or more selected nodes 108 to the particular application environment or virtual cluster. Based on the retrieved policies 132, dynamic boot engine 105 selects the appropriate boot image 132 for the selected nodes 108. For example, if there are already four nodes 108 executing a portion of application 114, then dynamic boot engine 105 automatically selects the fifth boot image 132 (at least partially based on node's 108 hardware and other characteristics and the one or more policies 132) that is compatible with application 114. Based on the retrieved policies 132, dynamic boot engine 105 may also select the appropriate virtual local disk image 134 for the selected nodes 108. Once the appropriate boot image 132 and/or virtual local disk image 134 are selected, dynamic boot engine 105 flashes node 108 with a pointer or other reference to the selected boot image 132 and virtual local disk image 134 and reboots node 108. Once node 108 is initialized (normally less than fifteen seconds), dynamic boot engine 105 (or some other job scheduler) executes the appropriate task, process, or other portion of application 104 on the selected node 108.

FIGURE 2 is a flowchart illustrating an example method 200 for dynamically rebooting a node 108 within one embodiment of the present disclosure. FIGURE 2 illustrates method 200, which generally describes a dynamic allocation of one of a

plurality of nodes 108 to a virtual cluster or application environment. Of course, any number of nodes 108 may be sequentially or concurrently reset, rebooted, or otherwise allocated within the scope of this disclosure. At a high level, method 200 includes selecting node 108 for allocation to an application's 114 environment, resetting boot image 132 of the selected node 108, and rebooting the node 108. The following description focuses on the operation of dynamic boot engine 105 in performing method 200. But system 100 contemplates using any appropriate combination and arrangement of logical elements implementing some or all of the described functionality.

Method 200 begins at step 205, where dynamic boot engine 105 determines that software application 114 would should be allocated more nodes 108. This determination may occur using any appropriate technique. For example, the administrator may manually add node 108 to the application environment for application 114. In another example, dynamic boot engine 105 may dynamically determine that nodes 108 may or should be used based on policies 132. Next, at step 210, dynamic boot engine 105 determines if there are any unutilized computing nodes 108 available. If there are more nodes 108 available, then dynamic boot engine 105 selects first available computing node 108 using any suitable technique at step 215. For example, dynamic boot engine 105 may select node 108 based on physical location, virtual location, application 114 compatibility, processor speed, or any other suitable characteristic. At decisional step 220, dynamic boot engine 105 determines if the selected node is compatible with application 114. If node 108 is not compatible with application 114, then dynamic boot engine 105 brings down the selected node using any suitable technique at step 225. Next, dynamic boot engine 105 dynamically selects policy 132 based on the software application 114 at step 230. For example, dynamic boot engine 105 may determine that three nodes 108 are currently executing software application 114. Based on this determination, dynamic boot engine 105 locates the fourth logical node 108 in policy 132. Based on the selected policy 132, dynamic boot engine 105 flashes the selected node with a pointer to a new boot image 131 at step 235 and associates virtual local disk image 134 at step 237. As described above, dynamic boot engine 105 may flash EEPROM 111 or any other suitable component. Next, dynamic boot engine 105 boots the selected node 108 using the new boot image 131 at step 240. Once the node 108 has been rebooted (or if the node was already compatible

with application 114), then dynamic boot engine 105 executes application 114 on the selected node 108 at step 245 and method 200 ends.

Returning to decisional step 210, if there were no computing nodes 108 available, then dynamic boot engine 105 selects an optimum utilized node 108 for application 114 at step 250. This selection of optimum node 108 may occur in any appropriate fashion such as, for example, determining the least utilized node 108, selecting a compatible node 108, or determining some other "best fit". At step 255, dynamic boot engine 105 kills the current processing on selected node 108 at step 255. Dynamic boot engine 105 may terminate the processing using any suitable technique such as executing an application-specific command, killing a process using the operating system, and others. At decisional step 260, dynamic boot engine 105 determines if the selected node 108 is compatible with application 114. If node 108 is not compatible with application 114, then dynamic boot engine 105 brings down the selected node using any suitable technique at step 265. Next, dynamic boot engine 105 dynamically selects policy 132 based on the software application 114 at step 270. For example, dynamic boot engine 105 may determine that three nodes 108 are currently executing software application 114. Based on this determination, dynamic boot engine 105 locates the fourth logical node 108 in policy 132. Based on the selected policy 132, dynamic boot engine 105 flashes the selected node with a pointer to a new boot image 131 at step 275 and associates virtual local disk image 134 at step 277. As described above, resource management engine may flash EEPROM 111 or any other suitable component. Next, dynamic boot engine 105 boots the selected node 108 using the new boot image 131 and virtual local disk image 134 at step 280. Once the node 108 has been rebooted (or if the node was already compatible with application 114), then dynamic boot engine 105 executes application 114 on the selected node 108 at step 285 and method 200 ends.

The preceding flowchart and accompanying description illustrate only exemplary method 200. System 100 contemplates using any suitable technique for performing these and other tasks. Accordingly, many of the steps in this flowchart may take place simultaneously and/or in different orders than as shown. Moreover, system 100 may use methods with additional steps, fewer steps, and/or different steps, so long as the methods remain appropriate.

Although this disclosure has been described in terms of certain embodiments and generally associated methods, alterations and permutations of these embodiments and methods will be apparent to those skilled in the art. Accordingly, the above description of example embodiments does not define or constrain this disclosure. Other changes, substitutions, and alterations are also possible without departing from the spirit and scope of this disclosure.

WHAT IS CLAIMED IS:

1. A method for computer cluster virtualization comprises:

selecting a distributed application;

retrieving a policy associated with the distributed application;

dynamically selecting one of a plurality of nodes;

resetting a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and

associating a virtual disk image with the selected node based, at least in part, on the retrieved policy; and

executing at least a portion of the distributed application on the reset node using the associated virtual disk image.

2. The method of Claim 1, the application executing on a subset of the plurality of nodes and the method further comprising:

comparing the subset of nodes with the retrieved policy; and

selecting one of a plurality of compatible boot images based on the comparison.

3. The method of Claim 2, wherein comparing the subset of nodes with the retrieved policy comprises:

determining a count of nodes in the subset; and

selecting the boot image based on a link in the policy and the count of nodes.

4. The method of Claim 2, each of the subset of nodes associated with one of the plurality of compatible boot images.

5. The method of Claim 1, wherein dynamically selecting one of the plurality of nodes comprises:

determining if one or more of the plurality of nodes is unutilized by a second distributed application; and

in response to at least one of the nodes being unutilized, selecting one of the unutilized nodes.

6. The method of Claim 5, in response to none of the nodes being unutilized, further comprising selecting one of the nodes utilized by the second distributed application based on one or more of the following:

the retrieved policy;

low utilization of the selected node;

priority of the selected distributed application; and

compatibility of the selected node with the selected distributed application.

7. The method of Claim 6, wherein resetting the boot image of the selected node comprises:

automatically shutting down the selected node;

resetting the boot image of the selected node; and

restarting the selected node using the reset boot image.

8. The method of Claim 7, further comprising terminating any processes associated with the second distributed application prior to shutting down the node.

9. The method of Claim 1, the policy comprising a plurality of links to boot images, each link associated with one of a count of nodes compatible with the distributed application.

10. The method of Claim 1, the policy comprising one or more parameters for determining the timing of the selection of the node.

11. The method of Claim 1, the boot image comprising a remote boot image stored in a Storage Area Network (SAN).

12. The method of Claim 1, the node associated with a first boot image prior to the reset and associated with a second boot image from the reset, the first and second boot image differing in at least one of the following characteristics:

operating system;

system configuration; and

distributed application parameters.

13. The method of Claim 1, further comprising:

determining that one of the plurality of nodes failed, the failed node executing at least a portion of the selected distributed application; and

wherein selecting one of the plurality of nodes comprises selecting one of the remaining nodes in response to the failure.

14. The method of Claim 1, each of the plurality of nodes comprising the same processor architecture.

15. The method of Claim 1, wherein selecting one of the plurality of nodes comprises selecting one of the plurality of nodes at a predetermined time.

16. Software for computer cluster virtualization operable to:

select a distributed application;

retrieve a policy associated with the distributed application;

dynamically select one of a plurality of nodes;

reset a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and

associate a virtual disk image with the selected node based, at least in part, on the retrieved policy; and

execute at least a portion of the distributed application on the reset node using the associated virtual disk image.

17. The software of Claim 16, the application executing on a subset of the plurality of nodes and the software further operable to:

compare the subset of nodes with the retrieved policy; and

select one of a plurality of compatible boot images based on the comparison.

18. The software of Claim 17, wherein the software operable to compare the subset of nodes with the retrieved policy comprises software operable to:

determine a count of nodes in the subset; and

select the boot image based on a link in the policy and the count of nodes.

19. The software of Claim 17, each of the subset of nodes associated with one of the plurality of compatible boot images.

20. The software of Claim 16, wherein the software operable to dynamically select one of the plurality of nodes comprises software operable to:

determine if one or more of the plurality of nodes is unutilized by a second distributed application; and

in response to at least one of the nodes being unutilized, select one of the unutilized nodes.

21. The software of Claim 20, in response to none of the nodes being unutilized, further operable to select one of the nodes utilized by the second distributed application based on one or more of the following:

the retrieved policy;

low utilization of the selected node;

priority of the selected distributed application; and

compatibility of the selected node with the selected distributed application.

22. The software of Claim 21, wherein the software operable to reset the boot image of the selected node comprises software operable to:

automatically shut down the selected node;

reset the boot image of the selected node; and

restart the selected node using the reset boot image.

23. The software of Claim 22, further operable to terminate any processes associated with the second distributed application prior to shutting down the node.

24. The software of Claim 16, the policy comprising a plurality of links to boot images, each link associated with one of a count of nodes compatible with the distributed application.

25. The software of Claim 16, the policy comprising one or more parameters for determining the timing of the selection of the node.

26. The software of Claim 16, the boot image comprising a remote boot image stored in a Storage Area Network (SAN).

27. The software of Claim 16, the node associated with a first boot image prior to the reset and associated with a second boot image from the reset, the first and second boot image differing in at least one of the following characteristics:

operating system;

system configuration; and

distributed application parameters.

28. The software of Claim 16, further operable to:

determine that one of the plurality of nodes failed, the failed node executing at least a portion of the selected distributed application; and

wherein the software operable to select one of the plurality of nodes comprises software operable to select one of the remaining nodes in response to the failure.

29. The software of Claim 16, each of the plurality of nodes comprising the same processor architecture.

30. The software of Claim 16, wherein the software operable to select one of the plurality of nodes comprises software operable to select one of the plurality of nodes at a predetermined time.

31. A system for computer cluster virtualization comprises:

a plurality of computing nodes; and

a management node operable to:

select a distributed application;

retrieve a policy associated with the distributed application;

dynamically select one of a plurality of nodes;

reset a boot image of the selected node based, at least in part, on the retrieved policy, the boot image compatible with the distributed application; and

associate a virtual disk image with the selected node based, at least in part, on the retrieved policy; and

execute at least a portion of the distributed application on the reset node using the associated virtual disk image.

32. The system of Claim 31, the application executing on a subset of the plurality of nodes and the management node further operable to:

compare the subset of nodes with the retrieved policy; and

select one of a plurality of compatible boot images based on the comparison.

33. The system of Claim 32, wherein the management node operable to compare the subset of nodes with the retrieved policy comprises the management node operable to:

determine a count of nodes in the subset; and

select the boot image based on a link in the policy and the count of nodes.

34. The system of Claim 32, each of the subset of nodes associated with one of the plurality of compatible boot images.

35. The system of Claim 31, wherein the management node operable to dynamically select one of the plurality of nodes comprises the management node operable to:

determine if one or more of the plurality of nodes is unutilized by a second distributed application; and

in response to at least one of the nodes being unutilized, select one of the unutilized nodes.

36. The system of Claim 35, in response to none of the nodes being unutilized, selecting one of the nodes utilized by the second distributed application based on one or more of the following:

the retrieved policy;

low utilization of the selected node;

priority of the selected distributed application; and

compatibility of the selected node with the selected distributed application.

37. The system of Claim 36, wherein the management node operable to reset the boot image of the selected node comprises the management node operable to:

automatically shut down the selected node;

reset the boot image of the selected node; and

restart the selected node using the reset boot image.

38. The system of Claim 37, the management node further operable to terminate any processes associated with the second distributed application prior to shutting down the node.

39. The system of Claim 31, the policy comprising a plurality of links to boot images, each link associated with one of a count of nodes compatible with the distributed application.

40. The system of Claim 31, the policy comprising one or more parameters for determining the timing of the selection of the node.

41. The system of Claim 31, the boot image comprising a remote boot image stored in a Storage Area Network (SAN).

42. The system of Claim 31, the node associated with a first boot image prior to the reset and associated with a second boot image from the reset, the first and second boot image differing in at least one of the following characteristics:

operating system;

system configuration; and

distributed application parameters.

43. The system of Claim 31, the management node further operable to:

determine that one of the plurality of nodes failed, the failed node executing at least a portion of the selected distributed application; and

wherein the management node operable to select one of the plurality of nodes comprises the management node operable to select one of the remaining nodes in response to the failure.

44. The system of Claim 31, each of the plurality of nodes comprising the same processor architecture.

45. The system of Claim 31, wherein the management node operable to select one of the plurality of nodes comprises the management node operable to select one of the plurality of nodes at a predetermined time.