(57) Abrégé/Abstract:
Mobile code infrastructure for the mobility of management components associated with virtual machines is provided. The system includes a set of software services that execute within a management plane associated with a set of virtual machines. Virtual and physical management planes are described. Mobile of management components is provided by embots that execute within an autonomic manager that is embedded within the management plane.
ABSTRACT

Mobile code infrastructure for the mobility of management components associated with virtual machines is provided. The system includes a set of software services that execute within a management plane associated with a set of virtual machines. Virtual and physical management planes are described. Mobile of management components is provided by embots that execute within an autonomic manager that is embedded within the management plane.
Management Of Virtual Machines Using Mobile Autonomic Elements

FIELD OF INVENTION

[0001] The present invention relates to management of virtual machines using mobile autonomic elements.

BACKGROUND OF THE INVENTION

[0002] The drive to make more effective use of physical resources within an enterprise information technology (IT) infrastructure has recently led to the introduction of virtual machine technology. Virtual machine technology allows one or more guest operating systems to run concurrently on one physical device. There are several approaches to providing virtualization technology, the most recent being paravirtualization and native CPU with BIOS or EFI support. Concurrent with the above has been the emergence of the management plane as the means by which hardware, operating system and applications are managed within the service plane.

[0003] The separation of management and service functionality has a number of documented distinct advantages that include separation of concerns, management of change and security improvements.

[0004] Finally, delegated management through the paradigm of Autonomic Computing has emerged. Autonomic Computing is a relatively recent field of study that focuses on the ability of computers to self-manage [Ref.1]. Autonomic Computing is promoted as the means by which greater dependency [Ref.2] will be achieved in systems. This incorporates self-diagnosis, self-healing, self-configuration and other independent behaviors, both reactive and proactive. Ideally, a system will adapt and learn normal levels of resource usage and predict likely points of failure in the system. Certain benefits of computers that are capable of adapting to their usage environments and recovering from failures without human interaction are relatively obvious; specifically the total cost of ownership of a device is reduced and levels of system availability are increased. Repetitive work performed by human administrators is reduced, knowledge of the system’s performance over time is retained (assuming that the machine records or publishes information about the problems it detects and the solutions it applies), and events of significance are
detected and handled with more consistency and speed than a human could likely provide.

[0005] The introduction of virtualization along with management and service plane separation has produced a new important problem. Specifically, if a virtual machine migrates, the associated units of manageability need to move as well. The problem extends to more than simply moving code; state accumulated for the element(s) being managed must also migrate.

[0006] Environments for the general mobility of software and state have been built. However, there has been no such infrastructure for an autonomic element, which applies specifically to the system management domain where virtual machines are under management.

SUMMARY OF THE INVENTION

[0007] It is an object of the invention to provide a method and system that obviates or mitigates at least one of the disadvantages of existing systems.

[0008] It is an object of the invention to provide an improved infrastructure that supports the synchronization of management and service plane states.

[0009] It is an object of the invention to provide an improved infrastructure that supports the migration of autonomic elements from one physical machine to another.

[0010] According to an aspect of the present invention there is provided a system and method for migrating an autonomic manager, which includes a mobile code environment.

[0011] According to a further aspect of the present invention there is provided a system and method for replication and distribution of an autonomic manager.

[0012] According to a further aspect of the present invention there is provided a system and method for configuration of an autonomic manager based upon an existing autonomic manager.
[0013] According to a further aspect of the present invention there is provided a system and method for suspending and resuming an autonomic manager.

[0014] According to a further aspect of the present invention there is provided a system and method for removing an autonomic manager.

[0015] This summary of the invention does not necessarily describe all features of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings wherein:

[0017] Figure 1 is a diagram showing an example of an autonomic element to which virtualization infrastructure in accordance with an embodiment of the present invention is suitably applied;

[0018] Figure 2 is diagram showing that the autonomic element of Figure 1 can be achieved by separating the autonomic manager from the managed element;

[0019] Figure 3 is a diagram where a single management plane contains a single autonomic manager for each service plane under management;

[0020] Figure 4 is a diagram that shows the movement of a service plane from one host to another;

[0021] Figure 5 is a diagram that shows the movement of the autonomic manager from the original host to the host where the service plane now resides;

[0022] Figure 6 is a diagram that shows the interaction between a policy that is involved in the migration of a service virtual machine and the policies that manage that virtual machine;

[0023] Figure 7 shows the one-to-many relationship that exists between an embot and the policies that effect autonomic management; and
[0024] Figure 8 shows the pluggable service architecture used to support migration where a migration service is shown as a plug-in.

DETAILED DESCRIPTION

[0025] The present invention provides a system and method for embedded system administration that uses autonomic computing principles to manage virtual machines. The invention is concerned with the scenario in which management and service are separated in distinct execution environments, also referred to as management and service planes respectively. A single management plane may provide manageability for one or more service planes. A management plane may be instantiated as a privileged virtual machine or may be an embedded single board computer running Linux. Multiple embodiments are possible. Service planes are instantiated as virtual machines in this embodiment of the invention. Individuals skilled in the art of virtualization will note that other forms of abstract execution environment are possible; a full virtualization of the operating system being the most extreme. The invention provides infrastructure to support the mobility of manageability components that provide autonomic management for a migrating virtual machine – or more generally execution container -- which constitutes a service plane.

[0026] Figure 2 is a block diagram of a management system according to an embodiment of the invention. The management system runs on a computer that has the characteristics of a typical desktop computer. In all embodiments of the current invention the management system consists of software that runs in a variety execution environments. One embodiment of an execution environment is domain 0 of the Xen virtualization environment, which is an example of a privileged execution environment. Another embodiment of an execution environment is an Open Platform Management Architecture (OPMA) or Advanced System Management Interface (ASMI) card, which runs Linux and the invention software runs on it.

[0027] The management plane shown in Figure 2 runs an application framework in this embodiment of the invention that provides a set of management services. One service is the management module runtime, which provides an execution environment for embots. This is identified clearly in Figure 2. All embots execute within this environment, which provides significant abstractions with respect to the service plane
being managed. Embots are the smallest runtime units of manageability as provided by this invention. Embots are created when a management module is deployed to the management plane and loaded. A management module is the smallest unit of deployable system administration. The nature of the management module is the subject of separate applications [Refs. 3, 4, 18, and 19]. By way of an analogy, a management module can be thought of as being a web archive (WAR) or enterprise java bean archive (EAR). Embots could be thought of as providing session bean functionality; i.e. business logic specific to the system management domain. Embots are instantiated by the application framework in a way that is directly comparable to the instantiation of session beans in a deployed J2EE application. Individuals skilled in the art of J2EE application design will understand the role of the home interface in the lifecycle of a session bean; embots employ a similar concept for their lifecycle management. Refs. 4 and 19 provide details of several embodiments of an invention that provides management module creation, editing and deployment mechanisms for system management knowledge. Refs. 3 and 18 provides details of an embodiment of an invention that uses a control (here management) plane for system management. Specifically, Refs. 3 and 18 provides information on how an embodiment of a management module can be accessed, interpreted and used to construct a fault model of a set of interacting applications, services and operating system components running on a server or workstation.

[0028] Figure 2 shows that the embots running in the embot execution environment interact through the embot application framework with the service plane through sensor and effectors running on the service plane. While Figure 2 shows a single service plane, a one-to-many management to service plane interaction is supported as would be typical in the scenario where the management plane is instantiated in a privileged virtual machine and the service planes are guest operating systems running within individual unprivileged virtual machines.

[0029] Figure 1 illustrates an example of autonomic element to which virtualization infrastructure in accordance with an embodiment of the present invention is applied. In Figure 1, an autonomic element separates management from a managed element function, providing sensor (S) and effector (E) interfaces for management. It should minimally impact the functions of the managed element. The managed element does
not dominate, override or impede management activity. For example, if the managed
element and autonomic manager share the same processor or memory address space
this cannot be guaranteed owing to the management of these shared resources by a
shared operating system. True autonomy requires a control plane, which has long been
the view in the telecommunications domain.

[0030] In one embodiment of the invention, referring to Figures 1-2, an embot
represents the monitor, analyze, plan, execution and knowledge parts of an autonomic
manager. In other embodiments of the invention, several embots communicating
through the channels shown by arrows connecting them in Figure 2 could collectively
constitute the same functionality.

[0031] Referring to Figures 3, 4, and 5 these figures graphically demonstrate a
scenario in which a single management plane manages 2 service planes in a
virtualized environment. In Figures 3, 4, and 5, "ACE" represents "Autonomic
Controller Engine", which is a software component running in the management plane
that forms the autonomic element; "VMM" represents "Virtual Machine Manager";
"VM" represents "Virtual Machine". An example of such a scenario would be the use
of Xen for the VMM with domain 0 hosting the privileged management domain and
service planes running Windows Server 2003 operating systems. Several policies
execute within the management plane, the policies being implemented within one or
more embots. In Figure 3, policy \( p_{a1} \) is related to the management of virtual machine
\( VM_{a1} \), policy \( p_{a2} \) is related to the management of virtual machine \( VM_{a2} \). Figure 4
shows that \( VM_{a2} \) has migrated to a new host, host B. In order for \( VM_{a2} \) to continue to
be managed autonomically the policies used to manage it must be migrated too. This
is necessary as a result of the requirement to maintain management and service planes
resident the same autonomic element. Figure 5 indicates this and implies a
requirement for code mobility. References 5, 6 and 7 provide examples of the utility
of this mechanism in the network management domain. References 8, 9, 10 and 11
provide details of a mobile code framework designed specifically for usage in the
network and system management domains. Reference 12 provides information related
to important mobile code standards; i.e. the MASIF specification. Reference 13
provides details on one of the more influential mobile agent environments – Aglets
from IBM. Reference 14 provides a Java programming language code base for code
mobility that is specifically targeted at network and system management. Reference 15 provides details on how the OSGi framework can be used to provide a code mobility framework. Individuals knowledgeable in the art of mobile agents will realize that many instantiations of mobile code (or agents, the words are used interchangeably in this document) are possible, the aforementioned references being a small sample of the available references on the subject.

[0032] In one embodiment of the invention, the management VM is represented by domain 0 of the Xen operating system. The service VMs are represented by domains 1, 2, etc. The operating systems running within domains 1, 2, etc may be Windows Server 2003, Window XP, for example. The hosts could be Intel-based server machines that are enabled with Virtualization Technology (VT); e.g. 940 CPUs. Reference 14 provides a framework for code mobility; the details will not be repeated here. As described in Ref. 1, policies consume observations made on the virtual machine being managed. Under certain conditions a policy determines that a particular virtual machine should be moved to another physical host. These conditions might be that the available processing power of the host is insufficient to meet the service level agreement associated with the applications hosted by the virtual machine. The realization of this need for migration might be through the interaction with an external monitoring system such as provided by a conventional management system. The virtual machine hypervisor through system administrator action may also provide such a stimulus to a policy. Individuals such as system administrators skilled in the art of system management will realize that there are many situations under which host resources can be insufficient to meet service level agreements.

[0033] In one embodiment of the invention, the policy having determined that migration should occur contacts the management plane on the destination host and notifies it that migration is just about to begin. It then executes the “xm migrate --live mydomain destination.ournetwork.com” command. This causes the VM to move to the destination host. The return code of the xm command is tested to ensure successful migration of the VM. If unsuccessful, the destination host is notified that migration was unsuccessful and a system administrator notified. If successful, the policy sets the state of the VM managed element just migrated to “migrated” and the location of the VM managed element to be “destination.ournetwork.com”. These changes are
propagated to policies that listen to changes in the state of the VM. These policies would include policies that are responsible for providing autonomic management functionality for the operating system, and for applications hosted within the VM; e.g. Microsoft Exchange or Oracle. Figure 6 describes this message flow.

[0034] Referring to Figure 6, the notified policies contact their embot containers indicating that migration should occur. One embot can contain one or more policies. This is shown in Figure 7. Embot containers include behavior that supports movement of manageability from one management plane to another, including the ability to move both code and state. In the current embodiment of the invention, a series of services are used to provide embot lifecycle management. References 1 and 2 provide details on these services. One such service in the migration service, which is based upon a mobile code infrastructure, or toolkit. Using the migration service provided as part of a mobile code infrastructure (see Ref. 14 for details), the affected embots schedule themselves for migration. Reference 4 provides details of the infrastructure used to support such a service; Figure 8 provides a view of the plug-in nature of one embodiment of the infrastructure used to support migration.

[0035] The migration service is responsible for transmission of the embots from the source host to the destination host. In the current embodiment, which is based upon the Mobile Code Toolkit (MCT) described in Ref. 14, transmission occurs by invoking the onMigration API of each embot followed by serializing the embot and transferring it to the destination migration service. The onMigration API allows the embot to unlink itself from the sensors and effectors on the source management plane. When de-serializing the embot a custom class loader is invoked to load the class associated with the embot. Should the class not exist locally, a request is made of the source migration service to send it. This process is repeated for all classes required to fully instantiate the embot on the destination management plane. Successful transmission of the embot is seen to have terminated when all required classes have been transferred and the embot instantiated on the destination platform. The onInit API instantiates the embot on the destination management plane and allows it to link into the sensor and effector framework on the destination management plane. When the embot reports that it has been successfully started, having invoked the onInit and onStart APIs, a success return code is sent back to the source migration service. The
migration service then removes the embot from the list of embots scheduled to be migrated. Once successfully transferred, the embot on the source management plane stops. This is achieved by executing the onStop API. All affected policies are notified through their stop API. If an embot migration fails, an exception is raised by the destination migration service that is caught by the sending migration service. The individual embots are then notified of the failure to migrate through the use of the onMigrateFailure API. In the current embodiment it is the responsibility of individual embots to decide what to do in this scenario. Examples of embot actions are: notify a system administrator or attempt to manage the (now) remote service virtual machine and retry the migration later.

[0036] Another embodiment might use the CORMAT infrastructure for mobility as described in Ref. 15. Individuals skilled in the art of mobile agents will realize that many alternatives exist for migration strategies. A rich literature exists in this area.

[0037] The embodiment(s) of the present invention have the following features:

- Migration of an autonomic manager.
- Management state preservation during migration.
- Lifecycle maintenance of management software in a virtualized environment.
- Fault recovery of the management plane when migration of management components cannot be moved in conjunction with a migrated VM.
- Self-management of the autonomic manager; i.e. autonomic manager can diagnose faults in its own operation and act to recover from them.

[0038] The embodiment(s) of the present invention have the following advantages:

- Improved system management through effective delegation.
- Results in reduced cost of ownership of system.
Higher system availability.

Management is delegated; management infrastructure responds dynamically to changes in service infrastructure.

Ability to dynamically react to changes in the applications deployed on a system; e.g. if a new application is deployed the system can automatically acquire and configure management functionality for it.

Provides a platform for coherent management of heterogeneous virtualized platforms; e.g. Windows and Linux operating systems.

[0039] Further detail can be found in Appendixes A, B, C and D which form an integral part of the Detailed Description Section of this application.

[0040] In addition, all references cited and listed in this application are hereby incorporated by reference, and form an integral part of this application.

[0041] The virtual machines of the present invention may be implemented by any hardware, software or a combination of hardware and software having the above described functions. The software code, instructions and/or statements, either in its entirety or a part thereof, may be stored in a computer readable memory. Further, a computer data signal representing the software code, instructions and/or statements, which may be embedded in a carrier wave may be transmitted via a communication network. Such a computer readable memory and a computer data signal and/or its carrier are also within the scope of the present invention, as well as the hardware, software and the combination thereof.

[0042] The present invention has been described with regard to one or more embodiments. However, it will be apparent to persons skilled in the art that a number of variations and modifications can be made without departing from the scope of the invention as defined in the claims.
WHAT IS CLAIMED IS:

1. A system for migrating an autonomic manager, comprising.
   
a mobile code environment for transmission of policies; and
   
mechanisms for configuration of policies associated with mobile execution environments

2. A system according to claim 1, wherein the mobile code environment is in an embedded management plane.

3. A system according to claim 1, wherein the mobile execution environment is a virtual machine.

4. A system according to claim 2, wherein the embedded management plane is an embedded single board computer.

5. A system according to claim 2, wherein the embedded management plane is a privileged virtual machine.

6. A system according to claim 1, wherein the mobile code environment is embedded in a single board computer.

7. A system according to claim 1, wherein the mobile code environment is embedded in a privileged virtual machine.

8. A system for distribution of groups of policies associated with a virtual machine.

9. A system according to claim 8, wherein the distribution system is embedded in a management plane.

10. A system according to claim 9, wherein the management plane is an embedded single board computer.
11. A system according to claim 9, wherein the management plane is a privileged virtual machine.


17. A method of implementing the system of claim 11.
The Virtual Machine Problem

Service VM_{b1}
Management VM ACE_{b} p_{b1}

Service VM_{a2}
Service VM_{a1}
Management VM ACE_{a} p_{a1} p_{a2}

VMM Resources Host_{b}
VMM Resources Host_{a}

Fig. 3
The Virtual Machine Problem (2)
The Virtual Machine Problem (3)

Fig. 5