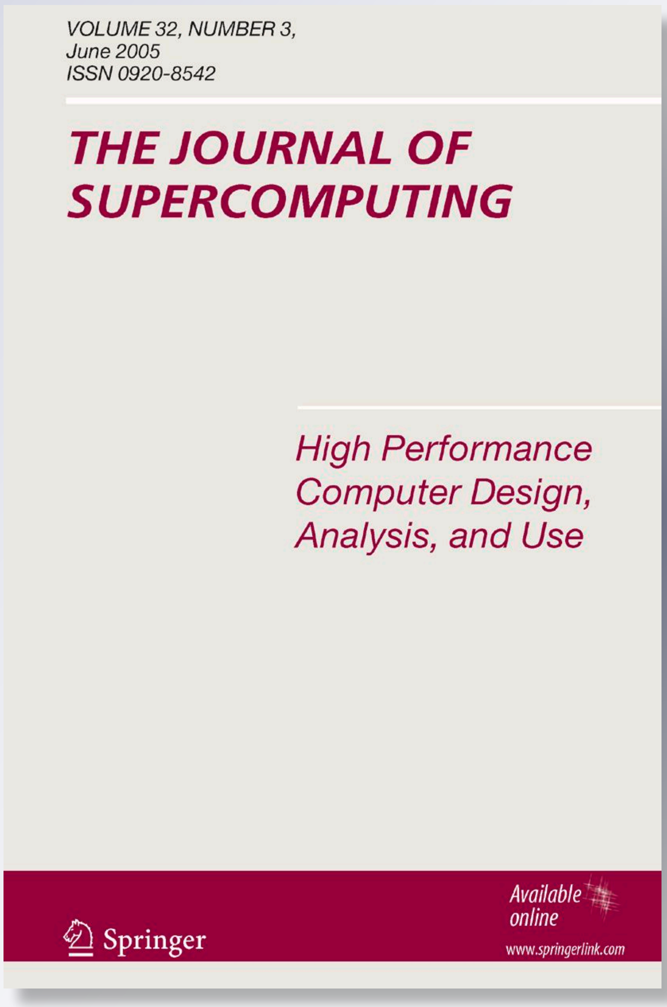
Adapting grid computing environments dependable with virtual machines: design, implementation, and evaluations

Xuanhua Shi, Hai Jin, Song Wu, Wei Zhu & Li Qi

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542

J Supercomput
DOI 10.1007/s11227-011-0664-7



J Supercomput
DOI 10.1007/s11227-011-0664-7

Adapting grid computing environments dependable with virtual machines: design, implementation, and evaluations

Xuanhua Shi · Hai Jin · Song Wu · Wei Zhu · Li Qi

Abstract Due to its potential, using virtual machines in grid computing is attracting increasing attention. Most of the researches focus on how to create or destroy a virtual execution environments for different kinds of applications, while the policy of managing the virtual environments is not widely discussed. This paper proposes the design, implementation, and evaluation of an adaptive and dependable virtual execution environment for grid computing, ADVE, which focuses on the policy of managing virtual machines in grid environments. To build a dependable virtual execution environments for grid applications, ADVE provides an set of adaptive policies managing virtual machine, such as when to create and destroy a new virtual execution environment, when to migrate applications from one virtual execution environment to a new virtual execution environment. We conduct experiments over a cluster to evaluate the performance of ADVE, and the experimental results show that ADVE can improve the throughput and the reliability of grid resources with the adaptive management of virtual machines.

X. Shi (✉) · H. Jin · S. Wu
Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer, Huazhong University of Science and Technology, Wuhan, 430074, China
e-mail: xhshi@hust.edu.cn

H. Jin
e-mail: hjin@hust.edu.cn

S. Wu
e-mail: wusong@hust.edu.cn

W. Zhu
Microstrategy, Hangzhou, 310012, China
e-mail: zw283927@gmail.com

L. Qi
IT Department, Operation Center, China Development Bank, Beijing, 100037, China
e-mail: quick.qi@gmail.com

Published online: 05 August 2011

Keywords Adaptive · Dependable · Virtual execution environment · Grid computing

1 Introduction

Nowadays, grid computing attracts more and more attentions in high performance computing area. By defining standardized protocols for discovering, accessing, monitoring, and managing remote computers, storage systems, networks, and other resources, grid technologies make it possible to allocate resources to applications dynamically, in an on-demand fashion [14]. However, while grids offer users access to many diverse and powerful resources, they do little to ensure that once a resource is accessed, it fulfills user expectations for quality of service. The problem is that most grid platforms today do not support performance isolation: activities associated with one user or virtual organization (VO) [15] can influence the performance seen by other processes executing on the same platform in an uncontrolled way. Another serious issue is that while grids provide access to many resources with diverse software configurations, a user's application will typically run only in a specific, customized software environment. Variations in operating systems, middleware versions, library environments, and file system layouts all pose barriers to application portability.

Due to its potential, virtual machine is attractive to high performance computers. Virtual machines present the image of a dedicated raw machine to each user [22]. This abstraction is very powerful for grid computing because users then become strongly decoupled from the system software of the underlying resource, and other users sharing the resource. In terms of administration, virtual machines allow the configuration of an entire operating system to be independent from that of the computational resource; it is possible to completely represent a VM "guest" machine by its virtual state (e.g., stored in a conventional file) and instantiate it in any VM "host", independently of the location or the software configuration of the host. Furthermore, we can migrate running VMs to appropriate resources. Upon the above reasons, the merge of virtual machine and grid computing is promising.

In this paper, we present an adaptive and dependable virtual environment for grid computing, ADVE, which allows a grid client to define an environment in terms of its requirements (such as resource requirements or software configuration), manage it, and then deploy the environment in the grid. Moreover, ADVE can change its configuration during the runtime. For example, ADVE can handle the problem when and where to create a virtual execution environment, when to change the number of the virtual execution environments, and what kind of situation to migrate one application to another virtual execution environment. ADVE is implemented with virtual machine technology and the adaptive component [6]. The virtual machine technology supports the definition, deployment, and destroying of the virtual execution environment, and the adaptive component handles the dynamic management of the virtual execution environment.

We summarize our contributions as follows: (1) Using virtual machines, we implement an adaptable virtual execution environment, which can be adaptable according to user requirement and system status; (2) The real system evaluation shows that ADVE can make grid application more reliable.

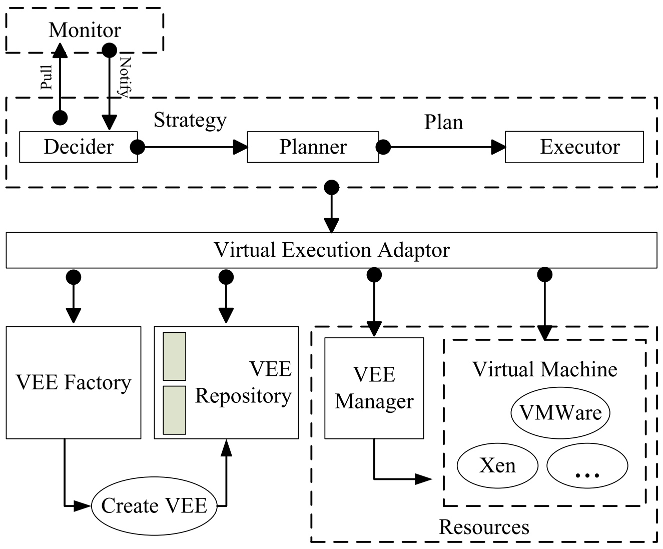


Fig. 1 Components of ADVE

The remainder of this paper is organized as follows. Section 2 describes the architecture of ADVE. The implementation of *VEE manager*, a key component in ADVE which controls virtual machines, is presented in Sect. 3. In Sect. 4, the performance evaluations are presented. Section 5 discusses the related work, and we conclude our work in Sect. 6 with a brief look at the future work.

2 Architecture of ADVE

In this section, we first present the architecture of ADVE, and then discuss the adaptive model in ADVE, which makes dynamic actions according to the dynamic changes of the grid environments and users' requests. Later, the situations for adaptation is presented, and the policies and plans are also presented in this section.

2.1 Components of ADVE

As illustrated in Sect. 1, ADVE is implemented with virtual machines and adaptive components. The adaptive capability of ADVE is fulfilled by the Dynaco component [10]. Dynaco is a component which provides the capability for developers to make adaptive decisions with policies and plans. The components of ADVE is shown in Fig. 1. ADVE is composed by three parts: the adaptive component, the virtual execution adaptor (VEA), and the virtual execution environment (VEE).

– **Adaptive components**. Adaptive components are implemented with Dynaco. In ADVE, the Dynaco gets the system information of the grid environment from *Monitor* with pull or push method, the *Monitor* collects the information of the grid environment and job status, such as the network traffic, cpu load, and job requests.

The Dynaco make adaptive decisions with the internal components of *Decider* and *Planner*. Decider makes *strategies* to guide the planner to make an adaptive *plan* for *executor*. The *Decider* makes the strategy based on the *policy*, which describes component-specific information required to make the decisions. The *Planner* is a program that modifies the components. The planner makes an adaptation plan that makes the component adopt a given strategy by the decider, and the plan-making is based on the *guide*. The *executor* take the adaptive action to the outsider. The *Executor* is a virtual machine to interpret the plan provided by the planner by means of *actions*. For example *creating a virtual execution environment* is an action. Dynaco has implemented the dynamic framework; we only add adaptive policies and guides to guide the adaptation, for example, *upon system overload, create an new virtual execution environment* is a policy, and *transfer the new image and deploy the new image* can be the guide for the strategy of creating a new virtual execution.

– **Virtual execution adaptor**. The virtual execution adaptor is a bridge between the Dynaco and the virtual execution environment. Because Dynaco only handles the adaptive policy, the VEE creation and management should be handled by the grid middleware, and interaction between the virtual machine and the grid also needs a bridge to fix the gap, for example, creating a virtual execution environment and executing grid application on it should handle the interaction between grid middlewares and virtual machines. The VEA have the following functions: passing the request of a grid user to VEA and transferring the responses back from the VEEs to the grid user; pulling and pushing the monitoring information of the VEEs to the monitor; passing the deployment and suspending action from Dynaco to VEEs; starting the applications on the virtual environment; and migrating one application from one VEE to another environment.
– **Virtual execution environment**. The VEE is based on Globus virtual workspace [20]. The VEE refines the execution environment layer in grid architecture: rather than mapping jobs directly onto hardware resources, VEE maps jobs to preconfigured execution environments which can then be mapped to grid resources. Since a VEE may exist beyond its deployment, and may in fact be deployed many times on different resources during its lifetime, two new services are introduced: VEE Repository which provides a management interface to execution environment, and VEE Manager which orchestrates their deployment. In order to create a VEE instance, VEA contacts the VEE Factory with a VEE description by XML schema. A negotiation process may take place to ensure that the VEE is created in a policy controlled way. The newly created VEE is registered with a VEE Repository, which provides a grid service interface allowing for inspection and management of VEE and keeps track of resources implementing VEE such as virtual machine images. As a result of creation the client is returned a WSRF [8] endpoint reference (EPR) to the workspace. To deploy a VEE on a specific resource, VEA contacts a VEE Manager grid service on that resource and presents it the VEE's EPR. The VEE Manager allows a client to deploy/undeploy, start/stop, and also pause/unpause execution environment. Based on such functions, the adaptive components can make adaptive actions to improve the execution performance.

2.2 Model of adaptive virtual environment

As mentioned above, ADVE tries to provide a flexible and dependable environment for grid applications. The targets of ADVE system is to improve the ratio of the grid resources' usage and to enhance the application reliability. These targets are fulfilled by the adaptive model in ADVE system. To illustrate the adaptive model, some definitions are presented first.

Definition 1 Throughput (T): Throughput of a grid resource (R) is defined as the number of requests that R can handle in a specific time interval (t).

Definition 2 Reliability (r): Reliability of a grid application is defined as the possibility that no application failure has occurred in a specific time interval (t).

Definition 3 Adaptive Overhead (AO): The adaptive overhead is defined as the sum of the Dynaco starting time and passing adaptive instruction to actions. The time for passing adaptive instruction to actions refers to the time that the instruction passing from *decider* to *VEA*.

Definition 4 Virtual Machine Overhead (VMO): The virtual machine overhead is defined as the execution time for starting/stopping, pausing/unpausing, and deploying/undeploying of the virtual machine.

To improve the throughput of the grid resources, ADVE creates a new VEE dynamically according to user requests and resources usage. To enhance the reliability of an application, ADVE creates a new VEE to migrate the application or to create an replica of the application. ADVE uses a decision-making method to make adaptive actions to improve the system performance. The target equation for decision making is shown in (1).

$$\begin{cases} Maxmum(T); \\ Maxmum(r); \\ Minmum(AO * i + VMO * j), \end{cases} \tag{1}$$

where i in (1) refers to the number of actions that Dynaco takes, and j refers to the number of virtual machine management. According to (1), the adaptive model in ADVE is a multiobjective decision making problem, and ADVE takes *Analytic Hierarchy Process Method* (AHP) to make decisions [30].

2.3 Situations for adaptation

To illustrate the adaptive model, we first present the situations for adaptation. Generally, there are three situations that ADVE will make adaptive actions to improve the performance, they are as follows:

– Number of request changes. When there are new users' requests arrive, the load for each VEE changes, and ADVE will compute the decision-making target function

Table 1 Policies for the change of user request numbers

Situation	Strategy
{*upon request increase over a threshold*}	*create a new VEE*
{*upon request decrease than a threshold*}	*undeploy a VEE*

Fig. 2 Plan template for creating a new VEE

```
Algorithm: Create a new VEE
        Preparing a virtual machine image;
        Searching available resources for deploying;
        Transferring virtual machine as grid services;
        Returning VEEID;
```

to decide whether to create a new VEE or not. At the other side, when the number of users' request decrease, ADVE will compute the decision-making target function to decide whether to stop a VEE to decrease the management overhead and execution management of a virtual machine.

– Computational load changes. When one computational resources is overload, for example CPU is overload, ADVE will create a new VEE to handle this. When some computational load decrease, ADVE will undeploy a VEE to reduce the management overhead.
– Resources down. In grid environment, it is very common that some resources are down during job execution. When some resources down, ADVE will create new VEEs on new available resources to take over the request on failed resources.

2.4 Adaptive policies and plans

Based on the situation analysis above, we can present the adaptive policies and the plans in ADVE, the policies and the plans are defined with Java language which can be recognized and interpreted by Dynaco system. To illustrate policies and plans simply, we take a natural language to present them. The policy for the change of request numbers can be given as Table 1.

In Table 1, there are two actions that need plans to guide, and they are *create a new VEE* and *undeploy a VEE*. Figure 2 shows the plan template for creating a VEE. As shown in Fig. 2, to create a new VEE, ADVE needs to take the following actions: preparing a virtual machine image for the VEE; searching available resources to deploy the virtual machine; transferring the virtual machine to the selected resources; deploying the virtual machine as grid services; and returning the ID of VEE to the VEE manager.

3 Implementation of VEE manager

As illustrated above, the VEE manager is a controller for virtual machines; the controlling function includes monitoring, deploying, and load balancing. The load balancing function refers to live migration of virtual machines which can lead to load balancing for back-end severs.

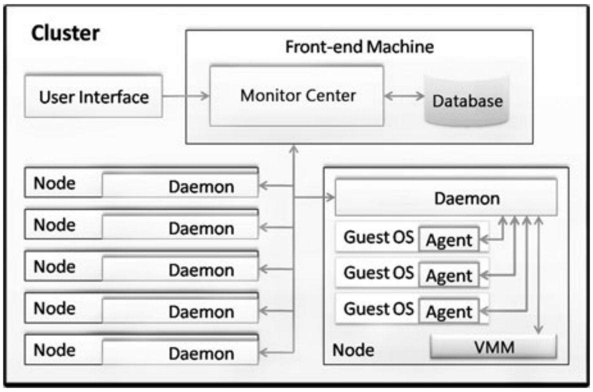


Fig. 3 Architecture of the monitor

3.1 Monitor

To launch a VEE, the VEE manager needs a clear understanding of the up-to-the-minute system conditions. *Monitor* is the fundamental component of VEE manager. In ADVE, the Monitor is implemented by three module, the *Monitor Center*, which is deployed on the front-end of the servers, the *Daemon*, which is a distributed module deployed on Virtual Machine Manager (VMM), and the *Agent* which is deployed inside the guest OS on the cluster nodes. The architecture of the Monitor is shown in Fig. 3.

Initially, the *Agent* obtains system information of guest OS while *Daemon* acquires physical resource information and VMM information of Host OS. Next, *Monitor Center* gathers information from each *Daemon* and performs analysis processing to maintain an overall information record. Then, according to the status, the VEE manager determines what to do, e.g., migrating one VM from one node to another node.

Agent mainly concerns about processes or jobs running inside a virtual machine as well as their resource usage information. Such information provides basis for adjusting reserved physical computing resource for virtual machines. Meanwhile, many types of statistical data and analytical data are offered to administrators to help them gain a deeper insight into the system. *Daemon* focuses on physical computing resource usage in a physical machine. Such information is crucial for deploying virtual machines. As resource is limited, we cannot deploy too many virtual machines on a single node arbitrarily. Before creating a virtual machine, VEE manager examines the information of physical machine carefully to make sure that there is enough physical computing resource left. Moreover, to migrate a virtual machine also needs checking the information beforehand because it requires sufficient memory available in target physical machine.

Monitor Center supervises the availability of every single physical node. Each physical node has two states: valid or invalid. A physical node is invalid implies that any operation related to this node would fail. For that a valid node could become invalid at any moment and vice versa, *Monitor Center* updates the states of physical machines periodically. Sometimes analysis processing is necessary especially when dealing with many virtual machines. For example, IP addresses of virtual machines could be displayed in a list, which, apparently, fails to intuitively reveal the network interconnections between virtual machines. *Monitor Center* analyzes the relationship of IP addresses and then generates a diagram depicting network topological structure.

3.2 Deployment

To launch a VEE for an application, VEE manager needs to deploy a virtual machine; the traditional process is extremely time-consuming. The process goes like this: first, create a virtual machine with a blank disk image; second, install an operating system (Guest OS) on this virtual machine; and finally, setup the application and its requirements. In ADVE, we present several ways to decrease the deployment time.

Firstly, VEE manager offers dozens of virtual machine images with preinstalled operating system as virtual machine templates. By cloning virtual machine template, we can deploy one or more virtual machines very easily and quickly. Therefore, it frees administrators from installing the operating system over and over. Templates are managed by *the Monitor Center* while cloned images are accessed by cluster nodes on which *Daemon* runs.

Secondly, VEE manager takes incremental template deployment for virtual machines. It takes a long time to duplicate dozens of disk images for grid users. An incremental disk image can be used to store the changes to another disk image, without actually affecting the contents of the original image; incremental disk image and base disk image combined to be an integrated disk image for a virtual machine. Base disk image is read-only, which means that all write operation is acted on the incremental disk image. When a new job wants to read an area, the VEE manager first checks if that area is allocated within the incremental disk image. If not, the VEE manager reads the area from the base disk image. The VEE manager takes the QCOW image format for storing the virtual machine image. The QCOW image format is one of the disk image formats supported by the QEMU processor emulator. The share-base disk image and incremental disk image deployment management is shown in Fig. 4.

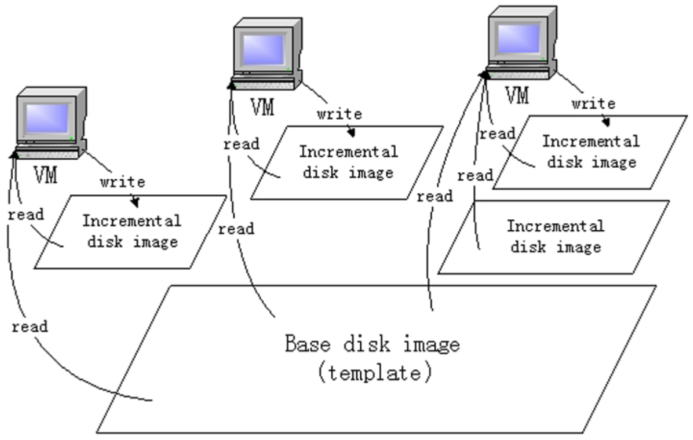


Fig. 4 Incremental template deployment

3.3 Load balancing

To raise resource utilization ratio, the VEE manager presents two levels of load-balancing functions: The first level is dynamic allocating memory and CPU allocation for virtual machines on a single physical machine (single-node level), and the other level refers to live migration of virtual machines which can adjust resources between physical nodes (multinode level). In single-node level, if a virtual machine is far from fully utilized, the VEE manager will cut down its computing resource and give more resources to other virtual machines. Reversely, if a virtual machine is overloaded, the VEE manager is supposed to promote its performance by appropriately reserving more physical resources for it. As to the multinode level, one case is that a physical machine would be overloaded due to too many virtual machines competing for limited CPU and memory resources. *Monitor Center* would detect this situation and generate a migrating plan for the VEE manager, then the VEE manager executes the plan to move virtual machines from an overloaded physical machine to comparatively idle physical machines. Another case is that the physical resource is too fragmentary resulting from virtual machines being created and destroyed. Resource fragments may prevent us from deploying more virtual machines. If this happens, *the Monitor Center* would suggest another migrating plan for the VEE manager, and the VEE manager will combine resource fragments into big pieces by making virtual machines assembled in the same physical machine.

4 Performance evaluation

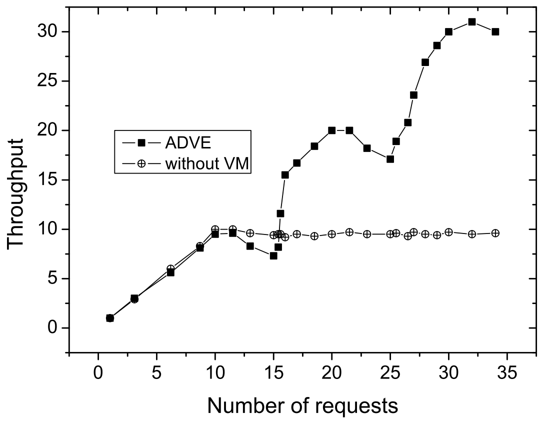
In this section, we present the performance results of the implementation. Firstly, we evaluate the throughput of ADVE compared with classical grid implementation, Secondly, we evaluated the reliability of application deployed on ADVE. Finally, we present the performance for the VEE manager.

4.1 Throughput

As mentioned above, ADVE is implemented with Globus virtual workspace. ADVE uses Xen (Version 3.0) as the implementation of the virtual machine. The grid services and infrastructure are implemented by using GT4 (Version 4.0.5). The VM image is Debian (Version 3.1) with the size of about 1 GB. To stage a VEE, the VEE Manager transfers the virtual machine image (including description metadata and the implementation-specific image) from the VEE Repository to the host node by using GridFTP [2]. One node at the Wuhan site of ChinaGrid [18] is chosen to evaluate the throughput of ADVE. The node is equipped with a 2.33 GHz Xeon CPU, 4 GB memory, and 60 GB disk. The node is configured to run single-CPU guest VMs. Once the transferring of VM image is complete, the VEE Manager waits for the VEA to start the VEE, which includes creating a VEE resource, loading the VM image into memory, and booting the VM.

The performance impact of virtual machines on applications has been shown to be small (typically under 5% of slowdown) for different application classes [3]. In

Fig. 5 Throughput of ADVE



our evaluation, we first explore the throughput of the adaptive implementation. In our preliminary evaluation, we explored the performance impact of different ways of using VMs as part of grid infrastructure. We also conducted a preliminary evaluation of VM usage with applications. The application service is an image grey transformation algorithm, and the request of a user is a call to rend a image on specific grid resources. The throughput between ADVE and the grid system without adaptive VMs is shown in Fig. 5.

In Fig. 5, the x-axis shows the numbers of request in one minute, the y-axis shows the throughput in one minute. From Fig. 5, we can see the following:

(1) If the number of requests in one minute is less than 10, the throughput of ADVE and the grid system without adaptive VMs are almost the same, and the throughput grows linearly. This shows that when the pressure for user requests is not so high, ADVE and the grid system without VMs perform almost the same. Also, we can see that the throughput of ADVE is a little lower than the system without VMs; this throughput difference comes from the small overhead of VMs.
(2) If the number of requests in one minute is more than 10 and less than 17, the throughput of ADVE is lower than the system without VMs. The reason for this is that ADVE will create new VEEs to improve the throughput, while the deployment of VEE will create new overhead, and the throughput will decrease.
(3) If the number of requests in one minute is more than 17, the throughput of ADVE is much higher than the system without adaptive VMs. The reason for this is that there are more VEEs to handle the request, and the throughput grows.
(4) There are three decreasing arcs in the line describes the throughput of ADVE in Fig. 5. The reason for these decreasing is that ADVE is creating new VEEs, and the deployment of a new VEE brings extra overhead, while the numbers of VEEs does not change during the deployment process.

4.2 Application reliability

The second evaluation explored the reliability of applications on ADVE. The evaluation environment is the same as illustrated in Sect. 4.1. We ran image grey transformation program, the failure-free execution time is about 60 seconds. We set the

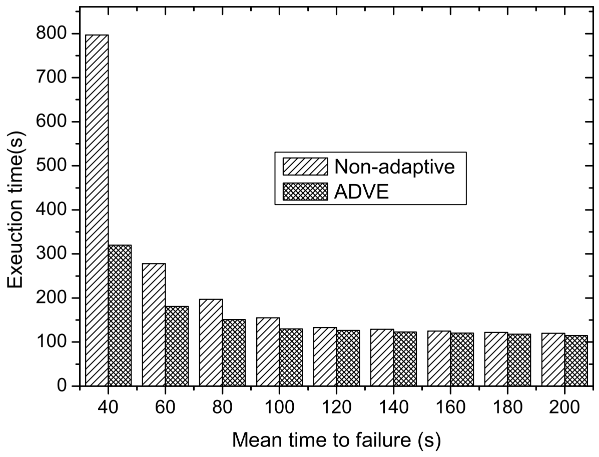


Fig. 6 Reliability of image grey transformation program

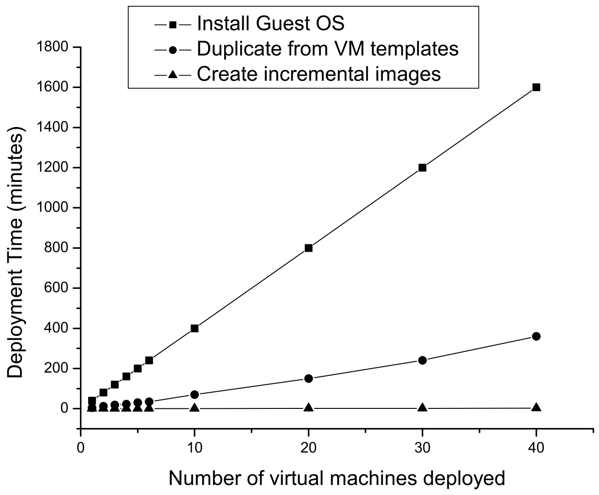
checkpoint interval of the application is about 6 seconds, and the checkpoint overhead and the recovery time from the checkpoint image are about 0.5 seconds. We set the down time of the grid middleware and the VEE as 60 seconds. The evaluation results are shown in Fig. 6. We experiment one thousand jobs for each mean time to failure (MTTF) testing, and the execution time in Fig. 6 is the mean time of these executions. Figure 6 shows the relationship between the execution time and the mean time to failure. In Fig. 6, the x-axis shows the MTTF, the y-axis shows the mean time of these executions. From Fig. 6, we can see the following:

(1) When the MTTF is less than 60 seconds, the execution time over ADVE is much less than the system without adaptive VMs. This shows that ADVE is suitable for long-duration jobs over unreliable grid environments.
(2) If the MTTF is longer than 120 seconds, the execution time difference between the ADVE and the non-adaptive system is very small. The difference in decreasing can be explained as this: an application runs over a reliable environment can get a good reliability, so the improvement will not be so obvious. Figure 6 also shows that the execution time over ADVE is a little smaller than over nonadaptive systems; this shows that an application can more reliability of the applications even over reliable environments with ADVE than with nonadaptive system, despite of the extra overhead of ADVE.

4.3 VM deployment

VM deployment is a time consuming work in ADVE, we evaluate the deployment on an IBM HS21 cluster at the Wuhan site of ChinaGrid, each HS21 node is equipped with 2-way, 4-core Intel Xeon CPU E5345 2.33 GHz and 8 GB memory, and each HS21 node features 73 SCSI hard disk. The HS21 nodes are connected with a Gigabit Ethernet network and a 10 Gbps Infiniband. We evaluate three VM deployment methods: (1) create blank images and install operating system; (2) create images by duplicating from VM templates; (3) create incremental image based on VM templates. We are trying to deploy a certain number of VMs with Windows XP installed.

Fig. 7 Time for deployment



Each VM is allocated 1 processor and 512 MB memory. On each single node, we deploy 4 VMs at most, thus the deployment time would not be extended due to slow response of overloaded nodes.

Figure 7 shows the deployment time of three different methods. From Fig. 7, we can observe that installing operating system for all VMs is a time-consuming work, about 40 minutes for each VM. By duplicating from VM templates, the VEE manager reduces the deployment time to approximately 6 minutes per VM which accounts for 1/7 of that of manual deploying. The deployment time is dramatically decreased again by utilizing incremental images; the average deployment time is just 0.3 minutes per VM which only accounts for 1/20 of the second method and accounts for 1/120 of original method. Besides the shortening of deployment time, the I/O consumption is also reduced, which is an important feature for running data-intensive applications.

To evaluate the overhead for incremental deployment, we run vBench [29] in each VM and compare the result with that of a VM using a nonincremental image. The overhead is shown in Fig. 8.

From Fig. 8, we can observe that the disk I/O overhead is around 10% if there is only one incremental image basing on a VM template. When the number of VMs sharing one base image is increased, the average disk I/O overheads slowly decrease and reach the bottom of 5% at the point of 20 VMs. The reason for the decreasing of overheads is that Linux caches disk blocks, which leads to an improved performance for a frequently accessed base image. The result shows that the incremental deployment in the VEE manager is suitable for large-scale grid environments.

5 Related work

For the flexibility of resource management and user management, the use of the virtual machine is very popular in grid computing. Figueiredo et al. [13] proposes an overview of merge virtual machine and grid computing. The In-Vigo project [1] [21]

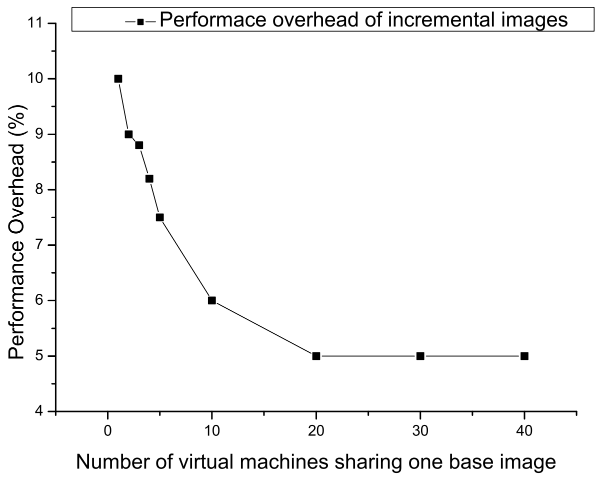


Fig. 8 Deployment overhead

made substantial progress in this direction while the Virtuoso [27] and VIOLIN [17] projects explored networking issues arising from use of VMs in this setting. Globus virtual workspace proposes the virtual workspace (VW) abstraction to describe such environments and showed how this abstraction can be implemented by using virtual machines for grid computing [20]. Sotomayor et al. propose a system, Haizea, combining batch execution and leasing using virtual machines [25], which implements leases as virtual machines (VMs), leveraging their ability to suspend, migrate, and resume computations and to provide leased resources with customized application environments in a cluster. ADVE share some features of Haizea. Attempts to capture requirements of an execution environment to some extent and automate their deployment have also been made before, for example, the virtual appliance project [24] uses virtual machines configured based on descriptions in a configuration language to ease administrative burden, and the Cluster on Demand (COD) project [7] allows a user to choose from a database of configurations to configure a partition of a cluster. The Xenoserver project [28] is building an infrastructure for wide-area distributed computing based on virtual machines similar to Globus virtual workspace. We differ from these projects by our focus on the dynamic features of managing virtual execution environment for grid applications during the application runtime. OpenNebula [23] and Eucalyptus [12] are two open-source projects, which aimed at building cloud computing [26] tool to manage the complexity and heterogeneity of distributed data center infrastructures. These two projects do the similar work as the VEE in ADVE, but ADVE address more on the adaptive issue for grid computing.

The adaptive computing is not a new concept, for example, an adaptive scheduling method is presented in [4], while this method does not give full control of applications to the developers. In [11], a Program Control Language is proposed, which provides a novel means of specifying adaptations in distributed applications. Gorender et al. presented an adaptive programming model for fault-tolerant distributed computing, which provides upper-layer applications with process state information according to the QoS [16]. In [5], a component-based autonomous repair management in distributed systems is presented, which provides the adaptive replication ability to manage the subsystems. The related work has the similar concepts to ADVE, whereas, ADVE focuses on providing an adaptive execution environment for grid applications which targets fault-tolerance and load balancing for grid computing with adaptive technology. DRIC framework proposed an policy-based fault-tolerance architecture for grid applications [19]. DRIC has the similar concept to make a dependable grid computing environment for grid applications as ADVE, while DRIC focus on provide fault-tolerance technique for applications themselves with the help of the grid middleware, such as checkpointing the application, resubmitting the application, while ADVE focuses on creating adaptive execution environments for grid applications. An adaptive replication middle-ware is presented by MEAD research group [9]. Just like DRIC, MEAD also targets the fault-tolerance of the applications themselves, not the environments.

6 Conclusions and future work

In this paper, we present the design, implementation, and evaluation of an adaptive and dependable virtual execution environment, ADVE. ADVE merges the idea of adaptive computing and virtual machine technology to provide a virtual dynamic execution environment for grid applications. ADVE deploys or undeploys virtual machines over physical grid resources based on grid environment changes and grid users' requests changes, such as number of service requests changes, resource load changes. The experiment conducted in this paper shows that ADVE can improve the throughput of the grid resources and the reliability of grid applications comparing with the classical grid environment.

Our future work includes the following directions: (1) We will improve ADVE with more dynamic policies to enhance the reliability of grid applications and to improve the flexibility of management of virtual machines. (2) We will improve the performance of virtual machine image transferring with I/O scheduling technology. (3) To improve the performance of the virtual machine deployment, we will design new virtual machine deployment techniques, such as template inside memory.

Acknowledgements This work is partly funded by the National 973 Basic Research Program of China under Grant No. 2007CB310900, NSFC under Grant No. 60973037 and 61073024, Wuhan Chenguang Program under Grant No. 201050231075, the SRF for ROCS, SEM, NCET Program Grant NCET-07-0334.

References

1. Adabala S, Chadha V, Chawla P, Figueiredo R, Fortes J, Krsul I, Matsunaga A, Tsugawa M, Zhang J, Zhao M, Zhu L, Zhu X (2004) From virtualized resources to virtual computing grids: the In-VIGO system. In: Future generation computer systems
2. Allcock W, Bresnahan J, Kettimuthu R, Link M, Dumitrescu C, Raicu I, Foster I (2005) The globus striped GridFTP framework and server. In: Proceedings of the 2005 ACM/IEEE conference on supercomputing
3. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebar R, Pratt I, Warfield A (2003) Xen and the art of virtualization. In: ACM symposium on operating systems principles (SOSP)

4. Berman F, Wolski R, Casanova H, et al (2003) Adaptive computing on the grid using AppLes. IEEE Trans Parallel Distrib Syst 14(4):369–382
5. Bouchenak S, Boyer F, Hagimont D, et al (2005) Architecture-based autonomous repair management: an application to J2EE clusters. In: Proceedings of the IEEE symposium on reliable distributed systems
6. Buisson J, André F, Pazat J (2005) A framework for dynamic adaptation of parallel components. In: Proceeds of ParCo 2005, Sep
7. Chase J, Grit L, Irwin D, Moore J, Sprenkle S (2003) Dynamic virtual clusters in a grid site manager. In: Proceedings of 12th international symposium on high performance distributed computing (HPDC-12)
8. Czajkowski K, Ferguson D, Foster I, Frey J, Graham S, Sedukhin I, Snelling D, Tuecke S, Vambenepe W (2004) The WS-Resource framework. www.globus.org/wsrf
9. Dumitras T, Srivastava D, Narasimhan P (2005) Architecting and implementing versatile dependability. In: Architecting dependable systems, vol 3
10. Dynaco (2011) http://gforge.inria.fr/projects/dynaco
11. Ensink B, Stanley J, Adve V (2003) Program control language: a programming language for adaptive distributed applications. J Parallel Distrib Comput 62(11):1082–1104
12. Eucalyptus (2011) http://www.eucalyptus.com
13. Figueiredo R, Dinda P, Fortes J (2003) A case for grid computing on virtual machines. In: Proceedings of the 23rd international conference on distributed computing systems
14. Foster I (2002) The grid: a new infrastructure for 21st century science. Phys Today 55(22):42–47
15. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J Supercomput Appl 15(3):200–222
16. Gorender S, de Araújo Macêdo RJ, Raynal M (2007) An adaptive programming model for fault-tolerant distributed computing. IEEE Trans Dependable Secure Comput 4(1):18–31
17. Jiang X, Xu D (2003) VIOLIN: virtual internetworking on overlay infrastructure. Department of Computer Sciences Technical Report CSD TR 03- 027, Purdue University
18. Jin H, Shi X, Qi L (2005) Use case study of grid computing with CGSP. In: Proceedings of the human society@Internet
19. Jin H, Shi X, Qiang W, Zou D (2006) DRIC: dependable grid computing framework. IEICE Trans Inf Syst, E89-D(2):612–623
20. Keahey K, Foster I, Freeman T, Zhang X, Galron D (2005) Virtual workspaces in the grid. In: Proceedings of Europar 2005, Lisbon, Portugal, September
21. Krsul I, Ganguly A, Zhang J, Fortes J, Figueiredo R (2004) VMPlants: providing and managing virtual machine execution environments for grid computing. In: Proceeding of the supercomputing (SC'XY) conference, Pittsburgh, PA
22. Meyer RA, Seawright LH (1970) A virtual machine time sharing system. IBM Syst J 9(3):199–218
23. OpenNebula (2011) http://opennebula.org/
24. Sapuntzakis C, Brumley D, Chandra R, Zeldovich N, Chow J, Lam MS, Rosenblum M (2003) Virtual appliances for deploying and maintaining software. In: Proceedings of the 17th large installation systems administration conference (LISA '03)
25. Sotomayor B, Keahey K, Foster I (2008) Combining batch execution and leasing using virtual machines. In: Proceedings of the 17th international symposium on high-performance distributed computing (HPDC)
26. Sotomayor B, Montero RS, Llorente IM, Foster I (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 13(5):14–22
27. Sundararaj A, Dinda P (2004) Towards virtual networks for virtual machine grid computing. In: Proceedings of the 3rd USENIX conference on virtual machine technology
28. Xenoserver project (2011) http://www.xenoservers.net/
29. Yuan P, Jin H, Ye D, Cao W, Yan Y, Xie X (2009) vBench: a micro-benchmark for file—I/O performance of virtual machines. In: Proceedings of the IEEE Asia-Pacific services computing conference (APSCC 2009)
30. Zahedi F (1986) The analytic hierarchy process—a survey of the method and its applications. Interfaces 16(4):96–108