Model-Based Resource Selection for Efficient Virtual Cluster Deployment

Shohei Yamasaki[†], Naoya Maruyama[†], and Satoshi Matsuoka[†,‡]
[†] Tokyo Institute of Technology, [‡] National Institute of Informatics
yamasaki@matsulab.is.titech.ac.jp, {naoya.maruyama, matsu}@is.titech.ac.jp

Abstract

When installing virtual cluster on Grid environments, randomly selecting nodes from arbitrary computing resources can radically increase installation time. This is because installation time of each node can vary greatly in heterogeneous Grid environments and the total installation time of a virtual cluster is bottlenecked by the slowest node. To achieve fast virtual cluster installation, we propose a model-based resource selection policy that chooses a near-optimal hosting node combination to assemble each cluster. We divide the VM setup process into five logical steps and construct a performance model for each step. The model represents the execution time of each step as a linear combination of hardware and software parameters, including CPU frequency, disk I/O performance, and installing package size. We have extended our virtual cluster installer, VPC, to select nodes in the increasing order of predicted installation time. Experimental results show that the model-based selection policy is indeed effective, especially when the package size differs depending on sites. The proposed policy has shown to reduce the installation time by up to 68% compared to the most naïve policy that selects nodes in a random order, 60% and 58% to the policies considering either CPU speed or disk I/O performance, respectively.

1 Introduction

Virtual clusters as job execution platforms for Grid environments have been receiving much attention as they can provide user-specific computing environments without destructively interfering with underlying hosting machines [3, 4, 6]. A virtual cluster is a virtualized computing cluster that consists of underlying multiple clusters on the Grid interconnected by physical or overlay networks. By combining multiple clusters, virtual clusters could achieve extremely higher performance or better system utilization compared to each single cluster.

One of the challenges in realizing such Grid resource sharing is how to efficiently deploy large-scale virtual clusters over heterogeneous environments. The time to deploy virtual machines on each node can be significant overhead for user jobs, limiting the applicability of virtual clusters only to long-running jobs. Furthermore, selection of hosting nodes can have huge impact on the total installation time, especially on heterogeneous Grid environments. This is because VM installation time of each hosting node can greatly vary in heterogeneous Grid environments, and because the total virtual cluster installation time is bottlenecked by the slowest node. We have actually observed such variation of installation time, where the difference between the fastest node and the slowest node was more than 100 seconds even within a single site. Thus, instead of randomly selecting nodes from any available hosting resources, a more intelligent selection policy prioritizing nodes with faster installation time is necessary for rapidly constructing large-scale virtual clusters over heterogeneous Grids.

Some of the issues in the implementation of virtual clusters have been addressed by Krsul [6], Foster [4], and Nishimura [8], but none of them consider the resource selection policy for heterogeneous environments. VMPlants by Krsul et al. [6] provides a graph-based VM creation and customization framework for virtual Grid environments such as In-VIGO [11], enabling user-specific flexible customization. Virtual Workspace by Keahey et al. defines interoperable interfaces with existing services in the context of the Globus toolkit [4, 5]. Our previous work addresses efficient, scalable installation of fully-customizable virtual environments [8]. Although these issues are important per se, they assume homogeneous physical hosting environments, which is rarely the case in large-scale Grid platforms or even within a single datacenter.

To achieve fast virtual cluster installation, we propose a model-based resource selection policy that selects near-optimal node combination to assemble each cluster. We derive a performance model of virtual cluster installation in our prototype installer VPC [8] by statistical linear regression with pre-stage performance profiles. We divide the whole VM installation process into five logical steps, and construct the model for each step. The model represents the

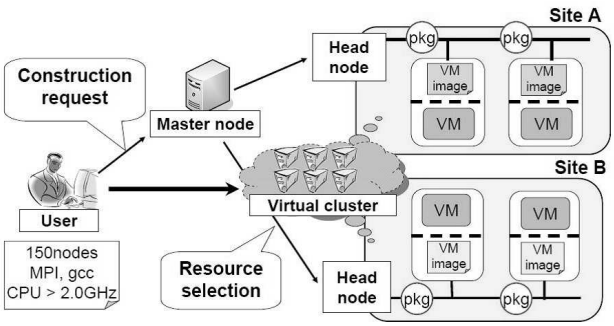

Figure 1. VPC Overview

```
# hardware specification
[hardware]
NumberOfNodes: 32
CPUArch: x86
CPUSpeed: 3GHz
RAM: 2GB
Disk: 8GB
# software specification
[software]
User: root:$1$abcdefgh$rV6Rh...
User: shohei:$1$abcdefgh$6DEi2...
Hostname: vc%02d
Network: 192.168.10.128/25
Packages: gcc mpich python...
```

Figure 2. A sample install request to create 32-node virtual cluster

execution time of each step as a linear combination of hardware and software parameters, including CPU frequency, disk I/O performance, and installing package size. We determine the coefficient of each variable by multiple regression analysis using pre-stage performance profiles. Our model-based policy selects nodes in the increasing order of predicted installation time, and transparently avoids the installation time increase due to heterogeneity in the underlying resources.

To evaluate the proposed model-based resource selection policy, we have extended the resource selection functions of our virtual cluster installer VPC. Preliminary experiments with the extended VPC show that the model-based selection policy is indeed effective, especially when the installing package size differs depending on sites. The proposed policy has shown to reduce the installation time by up to 68% compared to the most naïve policy that randomly select nodes, 60% and 58% to the policies considering either CPU speed or disk I/O performance, respectively.

2 Overview of the VPC Virtual Cluster Installer

VPC is a virtual cluster installer that achieves efficiency, scalability, and yet simultaneously fine-grained customization of virtual clusters [8]. Unlike other typical virtual cluster installer that assumes the existence of pre-built appropriate VM images [5, 6], VPC can dynamically instantiates VMs from scratch by using an existing package-based cluster management system, such as Lucie [12] and Rocks [9]. As such, it completely eliminates manual preparation of VM images and allows the user to easily enjoy much finer-grained customization. Furthermore, VPC reduces software installation time in common cases by dynamically creating *VM images*, which are virtual disk image where frequently-requested software is installed, but yet not fully configured. VPC automatically determines the packages to install in VM images by finding frequently-appearing packages in user-request history, thus requiring no manual configuration. We further reduce installation time by exploiting a scalable pipelined data distribution technique for deploying installing packages and VM images [7]. Experimental results show that VPC can install a 50-VM virtual cluster in 43 to 148 seconds; with VM images in place, the installation times are further reduced to 27 to 63 seconds. Further experiments show that VPC can install with a 200-VM virtual cluster in as fast as 40 seconds. Extrapolation with these results suggest high scalability of VPC: installation of even a thousand-VM virtual cluster could be done in less than two minutes.

Figure 1 illustrates the overview of the system architecture, which consists of three key components, including *master node*, *head node*, and *VM hosting nodes*. The master node receives user requests, and schedules and dispatches installation requests to the head nodes. The head node for each site manages the VM hosting nodes, where user-customized VMs are actually scheduled to run. In the rest of this section, we describe the installation flow in VPC that is relevant to the performance models presented in the next section. More detailed information on VPC itself can be found in [8].

2.1 Install Request Submission

As illustrated in Figure 1, the user initiates a virtual cluster installation by sending a request to the master node. The request consists of hardware and software specifications, each of which describes the requirements of either hardware or software. The hardware specification includes CPU type and speed, RAM amount, disk space, and the number of nodes. The software specification describes OS kernel and other user-level packages to be installed. The structure of the software specification depends on the particular cluster installer being used. Figure 2 is an example of configuration file when installing a 32-node virtual cluster.

2.2 Virtual Cluster Creation for a User Request

When receiving a user request, the master node initiates virtual cluster creation using appropriate VM hosting nodes on which the user-customized VMs actually run. To do so, the master node contacts the head node for each site for VM hosting nodes that satisfy the hardware specification in the user request, and selects appropriate VM hosting nodes. For each selected host, the master node requests the head node to create the requested VMs on the VM hosting nodes, and contacts the base cluster installer to install the requested software specified in the software specification to the VMs.

2.3 Software Installation onto Virtual Cluster Nodes

When receiving the installation request from the master node, the head node in each site identifies the VM image with whom the given request shares the largest commonality in installed packages. The missing packages in the VM images are fetched by the cluster installer on the head node from the nearest package repository. To avoid network bandwidth bottleneck in the head node, VPC exploits O(1) pipelined data transfer for every data distribution.

On each VM hosting node, a fresh VM instance is running and waiting for the head node's installation request. After the VM image and remaining packages are made available from the head node, the VM mounts the VM image as its base file system, and uses the employed cluster installer to install the remaining packages into the file system. Finally, the cluster installer performs the system configuration specified by the user request.

3 Modeling of the Virtual Cluster Installation in VPC

To derive a model for virtual cluster installation, we first create a node-level model that predicts the installation time at each node. The model for the whole installation of a virtual cluster takes the maximum of the node-level model, since the total installation time is determined by the slowest node. We construct the node-level model by dividing the process into five logical steps, and fitting a linear model to pre-stage performance profiles by multiple regression analysis. In this section, we first describe the process decomposition into the sub steps, and next the linear modeling of the time in each step.

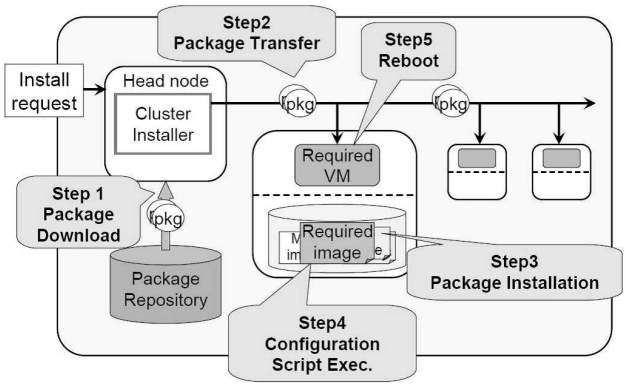

Figure 3. VPC Steps within a Site.

3.1 Logical Steps in the Virtual Cluster Installation

We decompose the installation process into the following five steps, as illustrated in Figure 3:

Step 1: Package Download As the head node receives a virtual cluster installation request from the master node, it decides which VM image to use and downloads the requested packages missing in the image from the local package repository.

Step 2: Package Transfer The head node transfers the downloaded packages to every hosting node.

Step 3: Package Installation Each VM hosting node mounts the VM image, and installs transferred packages into the image. Note that our current model assumes that the VM images are already staged to each node.

Step 4: Configuration After the package installation, each node executes configuration scripts for the standard system properties, such as host name and IP address, as well as, for particular packages.

Step 5: Reboot After the node finishes installing the user-requested VM image, it boots the VM with the customized image.

In Step 3, we do not consider the time for VM image transfer, assuming that every VM image is already staged to each node. The reason for this for our VPC in particular is because each node locally caches the transferred VM image within each local disk, and because the "cache miss" at the first use of an image is likely to be a rare event in the long run due to an effective caching algorithm. Of course, this assumption would not be valid if VM images are not reused

Table 1. Models for the installation steps

Step Names	Assumed Models
Package Download	$\alpha_1(PkgSize) + \lambda_1$
Package Transfer	$\alpha_2(PkgSize)$ $+\beta_2(TransferOrder) + \lambda_2$
Package Installation	$\alpha_3(PkgSize) + \gamma_3(CPUFreq)^{-1}$ $+\delta_3(DiskWrite)^{-1} + \lambda_3$
Configuration	$\alpha_4(PkgSize) + \gamma_4(CPUFreq)^{-1}$ $+\delta_4(DiskWrite)^{-1} + \lambda_4$
Reboot	$\alpha_5(PkgSize) + \gamma_5(CPUFreq)^{-1}$ $+\epsilon_5(DiskRead)^{-1} + \lambda_5$

many times, but in that the user request would vary significantly.

However, because VPC only generates images whose package combination is frequently requested in user-request history, our assumption is likely to be valid in common cases. Experimental evaluation remains a subject of future work.

3.2 Model of Each Installation Step

Our model represents the execution time of each step as a linear combination of hardware and software parameters, including CPU frequency, disk I/O performance, installing package size, and the package transfer order of each node. We obtain performance profiles using our virtual cluster installer VPC and determine the coefficients in the models by multiple regression analysis.

Table 1 shows the linear model for each step. *PkgSize* is the size (MB) of packages to be added to the VM image for a given user request, *TransferOrder* is the order of each VM hosting node when packages are transferred from the head node in a pipelined fashion, *CPUFreq* is CPU frequency (GHz), *DiskRead* is disk read speed (MB/s), and *DiskWrite* is disk write speed (MB/s). We use these values as parameters because they vary depending on hosting nodes and particular installation requests. As of current, we assume that other possible parameters that potentially affect the installation time, such as network bandwidth and latency within a site, are uniform, and do not include them in the models. It is worth noting that our modeling can accommodate the difference of network performance among sites since we derive a different model for each site.

We have conducted 200-node virtual cluster installation experiments using VPC on our PrestoIII cluster. The PrestoIII cluster consists of four different types of machines

Table 2. PrestoIII cluster node specifications.

	CPU (freq.)	RAM	HDD
Type 0	Athlon2000+ (1.6GHz)	1GB	IDE
Type 1	Opteron242 (1.6GHz)	2GB	IDE
Type 2	Opteron280 (2.4GHz)	4GB	SATA
Type 3	Opteron250 (2.4GHz)	2GB	SCSI

Table 3. Result of Multiple Regression and Adjusted Coefficients of Determination.

Step Names	Models with Determined Coefficients	R^2
Package Download	$0.312 \times PkgSize + 0.72$	0.99
Package Transfer	$0.022 \times PkgSize$ $+0.04 \times TransferOrder + 0.14$	0.99
Package Installation	$0.784 \times PkgSize + 56 \times (CPUFreq)^{-1}$ $+308 \times (DiskWrite)^{-1} - 47$	0.92
Configuration	$0.016 \times PkgSize + 1.1 \times (CPUFreq)^{-1}$ $+13 \times (DiskWrite)^{-1} - 0.8$	0.56
Reboot	$0.045 \times PkgSize + 9.9 \times (CPUFreq)^{-1}$ $+62 \times (DiskRead)^{-1} + 6.7$	0.47

as listed in Table 2. We have created 31 patterns of virtual clusters whose installation sizes ranges from 0MB to 30MB by 5MB increments, and measured the installation time for each virtual cluster 50 times.

Table 3 shows the results of multiple regression and analysis with Adjusted Coefficients of Determination. A coefficient of determination R^2 is a statistical measure of how well the regression line approximates the real data points. R^2 of 1.0 indicates that the regression line perfectly fits the data. In each step except the last two steps, R^2 is greater than 0.9, which means that the models approximate the real data points well. However, both the configuration and reboot steps have relatively lower coefficient values. The low accuracy of the configuration step has little effect on the overall installation time modeling, because the ratio of the time in the overall configuration step is typically very small. On the other hand, in our performance profiles, the reboot step can occupy more than 30% of the total installation time; thus, the accuracy of the reboot step does matter in the overall modeling effectiveness. Further analysis remains a subject of future work.

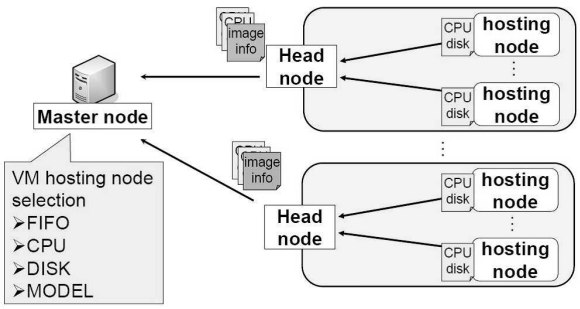

Figure 4. The extensions for the existing VPC.

4 Evaluation of the Model-based Resource Selection Policy

To evaluate the effectiveness of the model-based resource selection policy, we have implemented the policy within our virtual cluster installer, VPC [8], as well as other simple policies for comparison. This section describes the overview of the extensions for the VPC and presents experimental results showing the effectiveness of our proposed method.

4.1 Overview of the Extensions for the VPC

We extended the node selection function in the VPC for implementing our model-based policy as well as other simpler three policies. As illustrated in Figure 4, the head node in each site collects from each VM hosting nodes such information as CPU frequency and disk I/O performance, and sends it to the master node. The head node also notifies the master node of the VM images already transferred over the site. This information allows the master node to determine the size of additional packages that must be dynamically downloaded.

Our node selection policies include FIFO, CPU, DISK, and MODEL. The FIFO policy is the most naïve method in which the master node selects VM hosting nodes in a random order, and is the default policy in the current VPC. The CPU and DISK policies select VM hosting nodes in the decreasing order of CPU and disk I/O performance, respectively. With the MODEL policy, the master node predicts the virtual cluster installation time for a given user request, and selects the nodes with the fastest predicted installation times.

4.2 Experimental Methods

We compare the four selection policies by varying both the number of VMs and the sizes of additional packages to VM images. To evaluate the installation time with multiple sites, we partition a single cluster into two clusters, and use them as two emulated sites. Each of the two sites, called SITE0 or SITE1, consists of 50 VM hosting nodes with three different VM images. We evaluate three patterns of additional package sizes, 50, 100, and 150 MB, by changing the sizes of VM images. We also vary the number of VMs to be 5, 25, 50, and 75.

As an experimental environment, we use our PrestoIII cluster consisting of 256 Linux machines, and partition them into two emulated sites. The specification of each node is one of the four types listed in Table 2. As shown in the table, the PrestoIII cluster is not a typical homogeneous cluster, but has performance heterogeneity due to its partial system upgrades. The site SITE0 consists of nodes of type 0 and 1, and its head node is of type 0. The types of nodes in site SITE1 include of type 1, 2, and 3, and its head node is a type 1 node. We use a type 3 node for the master node. Each hosing node runs the Linux kernel v2.6.16 with Xen v3.0.2-2 patch applied, while the head nodes and the master node run the standard Linux kernel v2.6.12.6. All the nodes, including the VM hosting nodes, the head nodes, and the master node, are connected to 13 gigabit Ethernet switches, which are then interconnected by two gigabit switches.

4.3 Experimental Results

We have conducted two types of experiments using the prototype system: one is the situation where every node uses the same VM image, and another is where images are different between sites. The former situation can occur when both sites have been used in a similar way in its history, while the latter can occur each site has hosted different kinds of virtual clusters. We present the result of each experiment in the rest of this section.

4.3.1 The Same VM Image for Both Sites

Figure 5 and Figure 6 compare the installation times when using the same image in both sites. The model policy reduced the whole virtual cluster installation times by up to 38% compared to the FIFO policy, 20% to the CPU policy, and 11% to the DISK policy.

The reason of the slowest installation with the FIFO policy is that our PrestoIII cluster include several nodes with degraded disk I/O performance. Further examination of the results revealed that the FIFO policy had one of the slowest nodes at the very beginning of the node list, thus resulting in much slower performance even in the 5-node case.

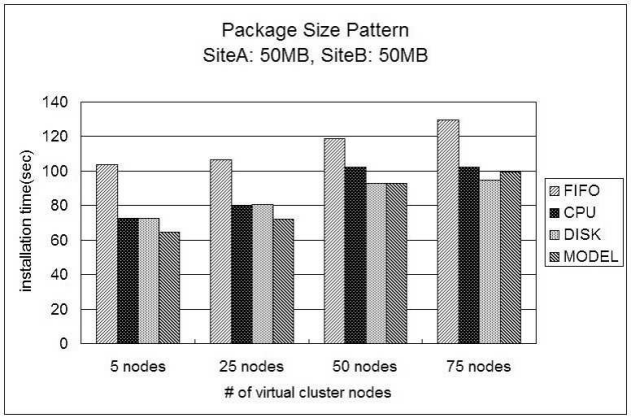

Figure 5. Comparison of the 4 policies: package size 50 MB in both sites.

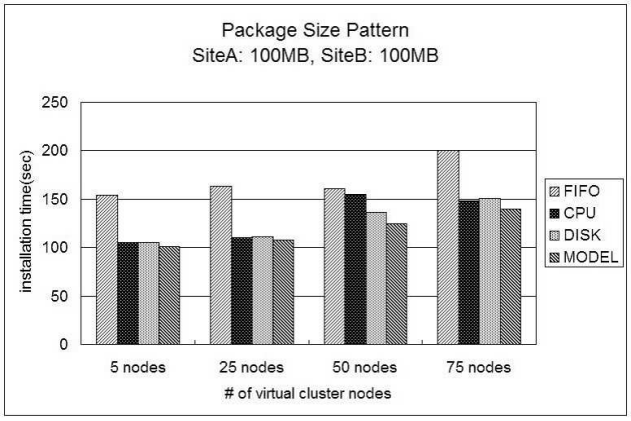

Figure 6. Comparison of the 4 policies: package size 100 MB in both sites.

4.3.2 Different VM Image for Each Site

As shown in Figure 7 and Figure 8, the model policy reduced the whole virtual cluster installation time by up to 68%, 60%, and 58% to the FIFO, CPU, and DISK policies, respectively. The results indicate that the proposed model-based policy is effective especially in the case that the size of the VM image used varies among sites.

With the CPU and DISK policies, although they are not bad selection policies in the case that the same VM images are used in both sites, the installation time increased greatly in the case that different VMs images are used, while with the proposed model the installation time did not increase as much as the other policies. The most effective situation for the MODEL policy was when the package size is 150MB

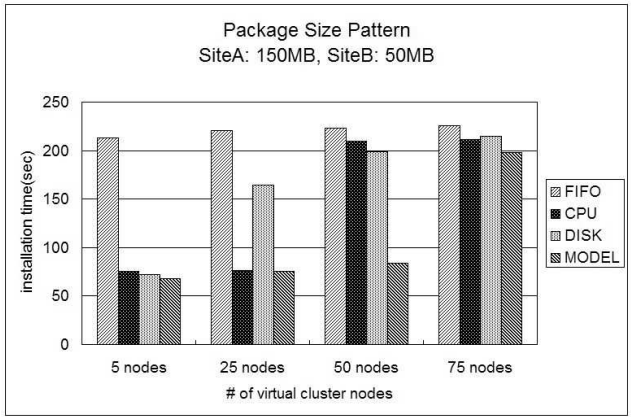

Figure 7. Comparison of the 4 policies: package size 150 MB in SITE0 and 50 MB in SITE1.

in the site SITE0, the package size is only 50MB in the site SITE1, and 50 VMs are installed. The model-based policy takes into consideration the difference in installation package sizes and therefore it uses only the nodes in SITE1, where only 50MB of packages are to be installed. However, the CPU and DISK policies do not consider the large difference in package installation sizes between the two sites, installing a virtual cluster using nodes in both sites. As a result, the CPU and DISK policies increase the installation time due to the additional 100MB package installation compared to the MODEL policy.

It is also worth noting that in the case of 75-VM virtual cluster, the installation time with the MODEL policy was nearly the same as the other policies since each site only has 50 VMs, and constructing a 75-VM cluster requires the use of both sites irrespective of the used policy.

5 Related Work

While several virtual cluster installation approaches have been proposed, most of them do not pay particular attention to scalability and heterogeneity with increasing number of nodes. Examples of such previous research include environment customization [6, 8] and interoperability with existing infrastructure [2, 4, 5]. Although these issues are important per se, they assume homogeneous physical hosting environments, which is rarely the case in large-scale Grid platforms.

More closely related to our work, Sotomayor et. al proposed an accurate resource allocation method for virtual computing environments [10]. Job execution time accounting in typical batch job schedulers include the time for staging executables to compute nodes, since the staging

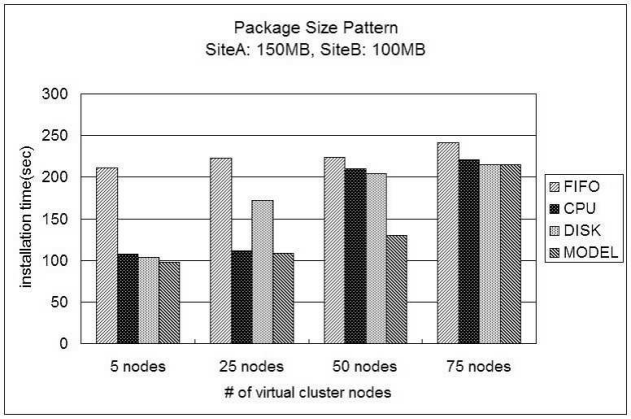

Figure 8. Comparison of the 4 policies: package size 150 MB in SITE0 and 100 MB in SITE1.

cost is typically very small compared to its execution time. However, deploying a virtual cluster can take significantly longer, thus including its time in execution time accounting can be unfair for the user. Sotomayor et. al proposed a VM-aware resource accounting method by estimating the time to deploy virtual machines on a single cluster. Similar to us, their estimation uses the information of the size of the VM image and the number of VMs to deploy. Unlike us, they assume homogeneous performance of compute nodes. Thus, their estimation would not be as accurate as ours in the presence of significantly different speeds of disk I/O.

Another project related to our work is French Grid'5000 [1]. Grid'5000 is built on a large collection of computing clusters distributed across WANs, and provides an on-demand customizable physical testbed for Grid computing research [1]. Although it installs systems on multiple distributed sites, it does not take the performance heterogeneity into account but rather relies on user specifications.

6 Conclusion

6.1 Summary

To realize a fast virtual cluster installation on heterogeneous Grid environments, we have presented a performance model that predicts the virtual cluster installation time, and have proposed a model-based resource selection policy. We divide the whole installation time of a virtual cluster into five logical steps, and derive a model for each step. Each step is represented as a linear combination of software and hardware parameters, including CPU speed, disk I/O performance, and installing package sizes. To evaluate the effectiveness of the model in installation time reduction, we have extended our virtual cluster installer VPC to select nodes in the increasing order of predicted installation time. Experiments using the extended VPC showed that the model-based selection policy is indeed effective, especially in a heterogeneous situation where the installation package size differs depending on sites. The proposed policy has shown to reduce the installation time by up to 68% compared to the most naïve policy that randomly select nodes, 60% and 58% to the policies considering either CPU speed or disk I/O performance, respectively.

6.2 Future Work

Modeling VM Image Transfer Time As we have discussed in Section 3.1, our current installation model does not consider the time for VM image transfer, assuming that every VM image is already staged to each node. We will experimentally evaluate the installation time including the VM image transfer and examine effects on the accuracy of our installation model.

Modeling Package Distribution from Remote Repositories We have assumed that each site has a locally-mirrored package repository. However, in addition to the site-wide repository configuration, the user might require remote non-mirrored repositories as well. In such a case, our model must reflect the network performance to the remote repositories.

Resource Selection Considering Performance Characteristics of Each Job This paper has addressed the reduction of time from when the install request is submitted until the time when its cluster is available to login and launch jobs. We are also considering further improving node selection by matching the performance characteristics of each job and underlying resources. For example, network-bottleneck jobs should be scheduled on tightly-coupled clusters with relatively small inter-node latencies. On the other hand, CPU-bottleneck jobs could improve the total performance by exploiting larger number of nodes with little overhead due to longer inter-node latencies.

Acknowledgments

This work is supported in part by the Ministry of Education, Culture, Sports, Science, and Technology, Grant-in-Aid for Scientific Research on Priority Areas, 18049028.

References

[1] F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, and O. Richard. Grid'5000:

a large scale, reconfigurable, controllable and monitorable Grid platform. In *International Workshop on Grid Computing*, Seattle, USA, Nov 2005.

[2] W. Emeneker and D. Stanzione. Dynamic virtual clustering. In *Proceedings of the IEEE International Conference on Cluster Computing (Cluster'07)*, pages 84–90, September 2007.

[3] R. Figueiredo, P. Dinda, and J. Fortes. A case for grid computing on virtual machines. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 550–559, 2003.

[4] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual clusters for grid communities. In *CCGRID '06*, pages 513–520, Singapore, May 2006.

[5] K. Keahey, I. Foster, , T. Freeman, X. Zhang, and D. Galron. Virtual workspaces in the grid. In *Euro-Par*, pages 421–431, 2005.

[6] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, pages 7–18, Pittsburgh, PA, November 2004.

[7] A. Manabe. Disk cloning program 'dolly+' for system management of pc linux cluster. In *Computing in High Energy Physics and Nuclear Physics*, 2001.

[8] H. Nishimura, N. Maruyama, and S. Matsuoka. Fast, Scalable, Fully-Customizable Installation for Virtual Clusters. In *CCGrid'07*, pages 549–556, 2007.

[9] P. M. Papadopoulos, M. J. Katz, and G. Bruno. Npaci rocks: Tools and techniques for easily deploying manageable linux clusters. In *Proceedings of the International Conference on Cluster Computing*, 2001.

[10] B. Sotomayor, K. Keahey, and I. Foster. Overhead matters: A model for virtual resource management. In *First International Workshop on Virtualization Technology in Distributed Computing (VTDC'06)*, Nov 2006.

[11] A. Sumalatha, C. Vineet, C. Puneet, F. Renato, F. Jose, K. Ivan, M. Andrea, T. Mauricio, Z. Jian, Z. Ming, Z. Liping, and Z. Xiaomin. From virtualized resources to virtual computing grids: the In-VIGO system. *Future Generation Computer Systems*, 21(6):896–909, 2005.

[12] Y. Takamiya. *Large-Scale Configuration Management and Installation of Commodity Clusters*. PhD thesis, Tokyo Institute of Technology, March 2006.