A Multi-Site Virtual Cluster System for Wide Area Networks

Takahiro Hirofuchi[*] *Takeshi Yokoi*[*] *Tadashi Ebara*[*†] *Yusuke Tanimura*[*]
Hirotaka Ogawa[*] *Hidetomo Nakada*[*] *Yoshio Tanaka*[*] *Satoshi Sekiguchi*[*]
[*]*National Institute of Advanced Industrial Science and Technology (AIST)*
[†]*Mathematical Science Advanced Technology Laboratory Co., Ltd.*

Abstract

A virtual cluster is a promising technology for reducing management costs and improving capacity utilization in datacenters and computer centers. However, recent cluster virtualization systems do not have the maximum scalability and flexibility required, due to limited hardware resources at one site. Therefore, we are now developing an advanced cluster management system for multi-site virtual clusters; which provides a virtual cluster composed of distributed computer resources over wide area networks. This system has great advantages over other cluster management systems designed only for single-site resources; users can create a cluster of virtual machines from local and remote physical clusters in a scalable manner, and dynamically change the number of cluster nodes on demand, seamlessly. In our system, a multi-site cluster achieves a monolithic system view of cluster nodes to enable existing applications to be deployed quickly and managed flexibly, just as in physical clusters. In this paper, we propose an advanced cluster virtualization mechanism composed of two key techniques. An L2 network extension of virtual machine networks allows transparent deployment over networks for distributed virtual cluster nodes, and a transparent package caching mechanism greatly optimizes data transfers in virtual cluster deployment over network latencies. Experimental results show multi-site virtual clusters have sufficient feasibility in WAN environments and promise great scalability for a large-scale number of virtual nodes.

1 Introduction

Large-scale datacenters and computer centers need to maintain enough computer system resources for their customers in a cost-effective manner. However, it is difficult to make appropriate investments against unpredictable demands on processing powers. Although recent virtualization technologies contribute to great improvements in efficient resource usage, scalability and flexibility are limited by the total amount of physical resources at one site. For virtualization of high performance computing environments, emerging virtual cluster systems allow rapid deployment of customized cluster systems. They basically reduce deployment costs and improve resource sharing efficiencies. However, existing virtual cluster systems focus only on single-site computer resources; users cannot create large-scale clusters beyond single-site hardware resources and dynamically extend running clusters in a flexible manner, on demand.

We designed and implemented a multi-site virtual cluster management system, which enables administrators to create a virtual cluster composed of virtual machines from distributed computer centers easily, and dynamically change its components for ever-changing demands from users, making efficient use of surplus computational resources.

In this paper, we propose two key mechanisms for multi-site support of cluster virtualization: An L2 extension of virtual machine networks allows transparent virtual cluster deployment over WANs, and contributes to the same usability as a single physical cluster, through unmodified application programs. In addition, a package caching mechanism is an essential component for scalable deployment of multi-site virtual clusters. It dramatically reduces backbone network traffic among sites and alleviates network resource requirements to the lowest feasible levels.

Section 2 clarifies the concept of multi-site virtual clusters and Section 3 presents key mechanisms for multi-site support. Section 4 gives design details and Section 5 evaluates prototype implementation through experiments. Section 6 notes additional discussions, and Section 7 summarizes related work. Finally, we conclude this paper in Section 8.

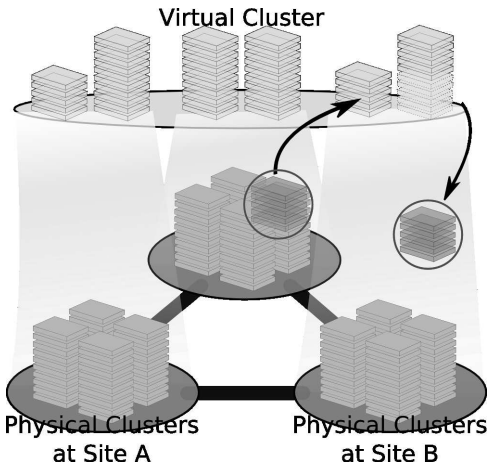

Figure 1: Multi-site virtual cluster concept

2 Multi-Site Virtual Cluster

A multi-site virtual cluster is an advanced cluster virtualization concept for greater scalability and flexibility, as well as sufficient usability. Beyond physical hardware limitations at one site, it aims to integrate distributed computational resources into a single cluster form by exploiting recent broad bandwidth backbone advances in WAN environments. A conceptual overview is illustrated in Figure 1, and its advantages are summarized as follows.

Scalability: A virtual cluster overlaying multiple sites can be composed of a greater number of computational nodes than can single-site hardware resources. This breaks hardware resource limitations at a single site, and integrates distributed computer resources as virtual clusters. In recent years, broadband networks over 1Gbps have been widely deployed between large-scale data centers and computer centers, and end-to-end reservations of network resources are becoming possible in academic and commercial backbone networks. These networks enable virtual clusters to be extended to other remote sites in a scalable manner.

Flexibility: A virtual cluster can be dynamically expanded according to user requests, by adding more and more virtual machines from remote sites. It is also possible to shrink it by releasing virtual machines. The virtual nodes belonging to a virtual cluster can be substitutable with other virtual machines of remote sites in a fairly transparent manner; for instance, some virtual nodes can be moved to other sites for hardware maintenance or power saving of physical clusters.

Usability: Even though a virtual cluster consists of distributed virtual machines located at remote sites, it can be seen as a single system image ”cluster” for users and applications. Its underlying networking topology is appropriately hidden from the inside of the virtual cluster. Remote computer resources are integrated into a uniform computing environment over its underlying heterogeneity, so that existing cluster applications can be applied to distributed environments with smaller efforts. A user basically operates a virtual cluster in the same manner as a physical one, using unmodified programs designed for physical clusters.

3 Virtual Cluster Management System

This section describes our virtual cluster management system. We recently decided to extend our single-site virtual cluster management system [10] for multi-site cluster virtualization. Its system architecture was carefully designed to be composed of standard technologies, and its components were well-modularized for further extension and customization. Therefore, multi-site extensions can be seamlessly integrated with it without large modifications. The basic mechanism of cluster virtualization is the same as that of multi-site cluster virtualization.

In our system, virtual clusters are based on three types of hardware virtualization; virtual machine monitors (VMMs) for processor virtualization, logical volume manager (LVM) and iSCSI [16] for storage access, and VLAN [6] for network isolation. A VMM works on each node of a physical cluster and hosts multiple virtual machines on it. Storage devices of a virtual machine are attached via iSCSI from dedicated storage servers, in which an LVM allocates storage space from large hard disks. A physical node hosts several virtual machines, each of which is a member of different virtual clusters. The private network of a virtual cluster is logically separated from the physical cluster’s networks by VLAN. Virtual clusters are completely isolated from each other, so that users can never use sniffer and tamper with network packets of other clusters.

A user interface is provided through a web browser. An administrator can specify the upper limit of physical resources allocated for a virtual cluster, and users make reservations of virtual clusters with the number of virtual nodes, the amount of the memory and storage required, and the start / end time. Users can also specify an operating system and applications which are then automatically installed in a reserved virtual cluster.

Our system is fully compatible with a widely-used cluster management framework, NPACI Rocks [14], in order to allow users to utilize its pre-packaged software repository of various cluster applications. A self-contained ISO image, called a Roll, includes a set of RPM files for a cluster application, as well as its cluster-aware configuration scripts. It should be noted that our system is also distributed as a Roll, which means Rocks users can easily install virtual cluster management sys-

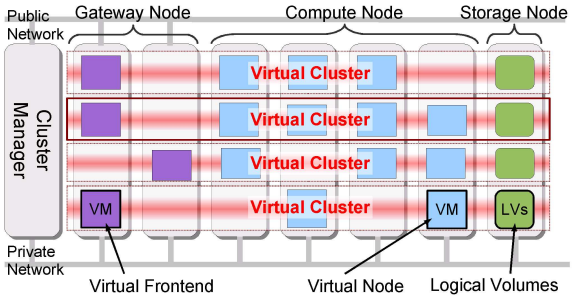

Figure 2: Virtual cluster management system for a single site

tems in their physical clusters. It includes a VMware server, and Linux iSCSI initiator / target systems, as well as our virtual cluster management programs.

An overview of our system is illustrated in Figure 2. Our system is composed of four types of physical nodes. A **cluster manager node** supports users and administrators of web interfaces, and sends administrative control messages to other nodes, such as those for the creation or destruction of virtual machines belonging to a virtual cluster. It also works as a frontend node of a Rocks cluster, which runs a PXE boot server for installation of other physical nodes. **Gateway nodes** and **compute nodes** host virtual machines on a VMM, which are launched or stopped after control messages are received from the cluster manager.

In our system, virtual clusters on a physical cluster are also installed and managed by Rocks; unmodified Rocks and cluster applications work for a virtual cluster. A virtual machine on a gateway node works as a frontend node of a virtual cluster, which provides other virtual machines on compute nodes of a PXE boot installation and a NAT service.

Storage nodes allocate logical volumes of storage space according to requests from the cluster manager, and export them to virtual machines via the iSCSI protocol.

All virtual clusters are fully isolated by a tagging VLAN. A cluster manager assigns a unique VLAN ID for a virtual cluster, and physical nodes create tagging VLAN interfaces (e.g., `eth0.123` on Linux) which are bridged to the pseudo network interfaces of hosted virtual machines.

3.1 Multi-Site Support Requirements

The requirements of multi-site support are summarized as follows: First, the multi-site extension needs to have compatibility with existing components and transparency for the inside of a hosting virtual cluster. Our single-site virtual clusters are basically designed to be scalable, flexible, and usable as much as possible for available hardware resources. We consider this basic design architecture is also applicable to multi-site virtual clusters. Its multi-site support should be achieved by a straightforward extension of the existing design; scalability, flexibility, and usability are improved in a transparent manner for users, applications, and cluster management systems. Users should be able to utilize multi-site virtual clusters seamlessly by using existing programs for physical clusters.

Second, multi-site virtual clusters should be allocated rapidly, and their internal operating systems and applications should be also installed and configured quickly, with the minimum manual configurations. These installations and configurations need to be performed dynamically over networks for additional virtual machines, so that users can fully customize their virtual clusters anytime. Both efficient deployment and full customization must be supported in this mechanism, even for remote nodes over network latencies.

Third, flexible relocation of virtual machines is required for management flexibility. Some virtual machines of a virtual cluster should be able to be moved to other host machines at remote sites; however the virtual cluster must continue to work properly during the move. Without this feature, it is difficult to maintain a large-scale virtual cluster distributed, across many sites consistently.

3.2 Multi-site Extension Mechanism

To meet the requirements above, we propose two key extension mechanisms for multi-site support of virtual clusters, an L2 extension of a virtual cluster network and transparent package caching. All components of a virtual cluster, such as virtual machines and storage services, are located under the same network segment, retaining the same addressing and naming as a single physical cluster. All programs designed for local area networks work correctly, including administrative services such as DHCP, PXE boot, and cluster monitoring tools, which use multicast and broadcast messages. In addition, it is basically possible to utilize the live migration features of VMMs (e.g., Xen [2] and VMware Infrastructure [20]), designed for a single network segment.

A transparent package caching service works toward rapid installation of virtual clusters by dramatically reducing network traffic among sites. It transparently intercepts download requests of packages and saves them into a local cache repository. Our system exploits a package-based cluster installer (i.e., one not based on preauthored virtual machine images), by which users can customize their virtual clusters with unlimited combinations of pre-packaged applications including cluster-

wide setting scripts. This deployment approach also enables an installer on each node to absorb hardware differences, even for the case of virtual machines (e.g., CPU architecture). In addition, every node in a cluster can be customized easily in a different manner, as in our virtual cluster system. As for multi-site cluster deployment over a WAN, this package-based installation approach is more suitable than VM-imaged based ones since both customizability and caching efficiency are achieved.

It should be noted that these extension mechanisms for multi-site support are not specific to our virtual cluster management system, however, and can be applied to other virtual cluster projects for multi-site support.

4 Design and Implementation

A multi-site virtual cluster is composed of virtual clusters at its master site and worker sites, and these virtual clusters at the sites are interconnected by Ethernet VPNs to be a single cluster. The cluster manager in Figure 2 is modified to send/receive allocation requests of virtual clusters and also reserve network resources if possible. A virtual frontend node of the local virtual cluster in the master site also works as the Rocks frontend node of the multi-site virtual cluster; other virtual nodes at its worker sites are also installed and managed from the remote virtual frontend. Ethernet VPN services and transparent cache servers are incorporated into the virtual cluster management system as independent components for multi-site support.

4.1 L2 Network Extension

An L2 extension of virtual cluster networks is performed by an Ethernet VPN service as illustrated in Figure 3. As mentioned in Section 3, the private network of a virtual cluster is completely isolated from other networks by means of a tagging VLAN. A VLAN tag [1] is added by a host operating system, and Ethernet frames with a VLAN tag are sent to other physical nodes through a LAN. To extend this private network of a virtual cluster, our Ethernet VPN service gets rid of the VLAN tag from an Ethernet frame and redirects the frame to other interconnected sites in which virtual machines of a multi-site virtual cluster exist. This mechanism avoids sharing VLAN IDs among sites, of which only up to 4096 are allowed, and enables VPN services to send only the internal traffic of a virtual cluster.

Our system works with an appropriate Ethernet VPN mechanism from commercial services and appliances, such as WAN Ethernet VPN services from ISPs, and proprietary VPN appliances of L2TPv3 [7] or EtherIP [3].

[1] An additional field in an Ethernet frame header in which a unique VLAN ID is embedded.

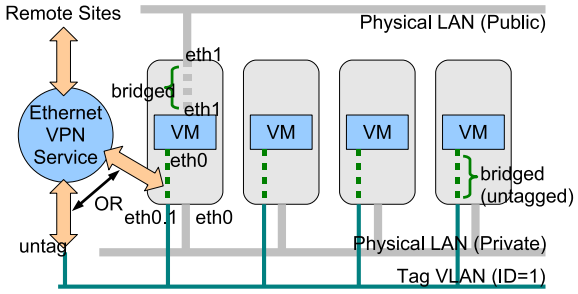

Figure 3: L2 extension of virtual cluster networks

In these cases, a VPN appliance is attached to a LAN among physical clusters, and configured from a cluster manager via wrapper scripts of its console interface.

On the other hand, administrators can decide to utilize software VPN programs (e.g., OpenVPN [12], Vtun [21], or PacketiX [13]) for multi-site support, instead of external VPN appliances. A VPN program is installed into a gateway node which has both global and private network interfaces, is launched to establish Ethernet VPN connections for the untagged private network (e.g., `eth0.123`) of a local virtual cluster. In our prototype implementation, we utilize an open source VPN software program, OpenVPN, since it can be distributed with our virtual cluster package for the Rocks toolkit.

4.2 Transparent Package Caching

Figure 4 shows the transparent caching mechanism for downloading packages. All package retrieval requests to a remote virtual frontend node are redirected to a caching service on a local gateway node. The caching service downloads packages not cached yet, instead of downloading by local virtual nodes, and saves them for later requests. Concurrent requests from local nodes are merged into one remote request by copying partial retrieving data to local downloading streams. In the best case, network traffic between a pair of master and worker sites is reduced to the total amount of required package sizes for one virtual node. When building a large scale virtual cluster, this mechanism is essential for practical deployment time; without caching packages efficiently, network traffic for installations becomes proportional to the number of remote nodes. Basically, package cache repositories are separated for virtual clusters respectively, for security reasons. However, it is possible to share common packages among virtual clusters if their digital signatures are verified and their licenses allow redistribution.

Actually, Rocks has a bittorrent-like mechanism that transfers downloaded packages among installation nodes. Downloading network traffic from a frontend

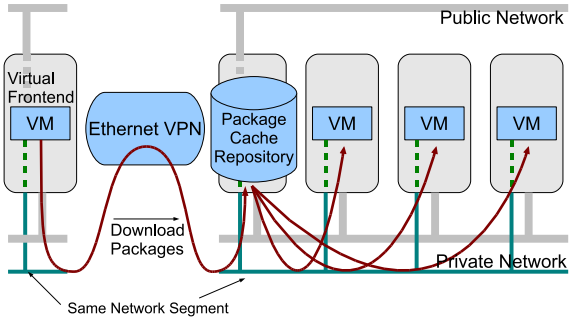

Figure 4: Transparent package caching mechanism

node is reduced due to end-to-end package deliveries. However, this mechanism, only available among installing nodes, does not work for an initial boot image of approximately 250MB, and unlike bittorrent, only completely-downloaded files are transferable to other nodes. Therefore, our system needs another acceleration mechanism specially designed for reducing network traffic between remote sites over a WAN.

In a prototype implementation, a web proxy server, Squid [18], is exploited for caching packages, since it supports a transparent proxy mode and concurrent downloading aggregation for one target file. A Squid server is launched for a virtual cluster, and all outgoing HTTP requests to package files on a virtual frontend are redirected to its listening port. It is carefully configured to be fully transparent for Rocks components. Node-specific requests, such as getting configuration files, are redirected to the virtual frontend without passing through a caching engine.

4.3 Deployment Steps

Multi-site virtual clusters are deployed in the following steps. First, a master site, which receives a reservation request, allocates a single-site virtual cluster with a virtual frontend node. Next, to extend this to a multi-site virtual cluster, the master site sends allocation requests of virtual nodes to other sites. If successfully allocated, VPN sessions are established and virtual machines at worker sites are launched to start node installation from the virtual frontend node. Node installation is based on the Rocks installer; an Ethernet VPN transparently extends a private network of the virtual frontend node, except for its bandwidth and latency restrictions.

A node installation consists of three phases: **Phase 1.** After a bootstrap system starts, it tries to get a kickstart file from a frontend node. The kickstart file includes all configuration information required to automate an Anaconda-based installer. A Rocks frontend node dynamically generates this file on request for node-by-node customization. **Phase 2.** An installing node starts to download three file system images which include installer-related programs and data. It expands them and launches the advanced installer. The file system images are `update.img`, `product.img`, and `stage2.img`, which total 250MB. **Phase 3.** The Anaconda installer starts to run with the kickstart file. It creates target file systems on disks, and then continues to download and expand many packages. After all configurations have been made, the node reboots and its installation is finished.

5 Evaluation

In this section, we evaluate the proposed extension mechanisms through experiments with a large number of physical nodes under emulated networks. We focused on the deployment feasibility of large-scale multi-site virtual clusters in WAN environments. An overview of our experiments is illustrated in Figure 5 and Table 1. We used two physical clusters to emulate a multi-site virtual cluster of approximately 150 nodes. Cluster A is a master site cluster and it hosts virtual nodes including a virtual frontend. Cluster B works as a remote worker site of the multi-site virtual cluster. Its virtual nodes are installed from the virtual frontend node in Cluster A over an emulated WAN. Both physical clusters are interconnected with Gigabit Ethernet via a hardware network emulator, GtrcNET-1 [4]. GtrcNET-1 guarantees very precise emulated latencies for such high throughput network media. The private network for Cluster B is switched by Cisco 4006; its switching performance was sufficient for the whole range of node-to-node throughputs in the experiments. An OpenVPN session is used to make a single network segment by establishing an Ethernet VPN between the clusters.

5.1 Virtual Cluster Installation via Ethernet VPN

We first tested whether a multi-site virtual cluster is possible in WAN environments by extending the cluster's private network with an Ethernet VPN. 10 virtual machines, respectively launched on physical nodes in Cluster B, are installed through an OpenVPN tunnel over an emulated WAN with different latencies. A network latency through the OpenVPN tunnel is always under 1ms without latency emulation. Its maximum throughput is up to 160Mbps, limited by the processing power of the OpenVPN nodes, due to Blowfish encryption overhead. 495 RPM packages (900MB) are installed on each virtual node; they include many Grid and clustering middleware programs (e.g., Globus Toolkit, Condor, and Sun Grid

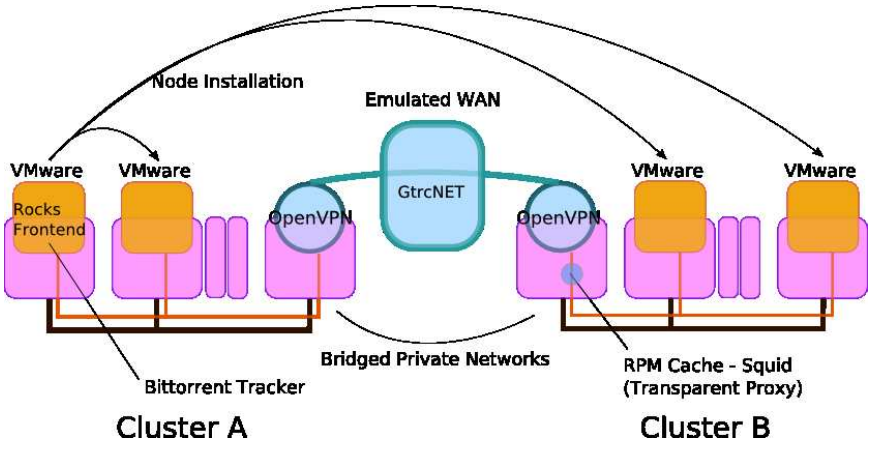

Figure 5: Experiment overview

Table 1: Experimental environment

Cluster A	16 nodes	AMD Opteron Processor 244 3GB memory Gigabit Ethernet (Broadcom BCM5703X) x2
Cluster B	134 nodes	AMD Opteron Processor 246 6GB memory Gigabit Ethernet (Broadcom BCM5704) x2

Engine). The cache server is disabled during this experiment. For the network latencies tested, all virtual nodes were successfully installed, and all applications basically worked correctly.

Figure 6 shows the installation time for 10 nodes under different network latencies. [2] Installation time becomes larger under longer network delays, since package downloading takes a longer time due to a TCP congestion algorithm. A well-known solution for reducing download rates over long network latencies is to optimize TCP window sizes to be sufficient values. However, this approach is not appropriate for the installer, which downloads a large number of small files with different connections.

We also tested the installation of a larger number of virtual nodes. Huge amounts of network traffic went through the VPN tunnel, and deployment became much slower.

The L2 network extension of a virtual cluster is a straightforward method to create a multi-site virtual cluster, which enables existing installer tools and most applications to be also used for it without any modification. However, without package caching, it is difficult to deploy large-scale virtual clusters over WANs for actual use.

[2] All virtual nodes finished their installation approximately at the same time.

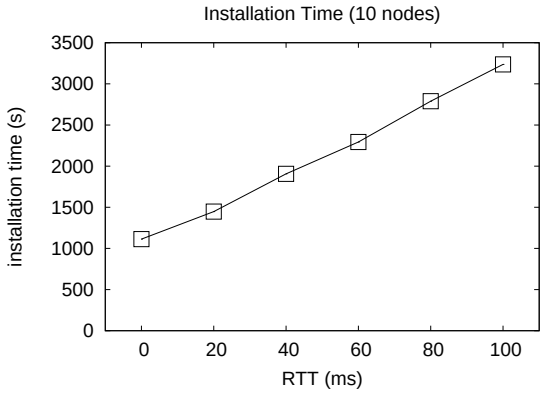

Figure 6: Installation time for 10 nodes under emulated network latencies

5.2 Transparent Package Caching

Next, we tested the transparent package caching mechanism for virtual cluster installation over an Ethernet VPN, which aimed to resolve the problem of huge network traffic during virtual cluster deployment.

In this experiment, we started by installing 134 virtual machines on physical nodes of Cluster B at one time. The virtual machines were booted from their virtual CDROM drives only for bypassing a PXE boot stage; some nodes

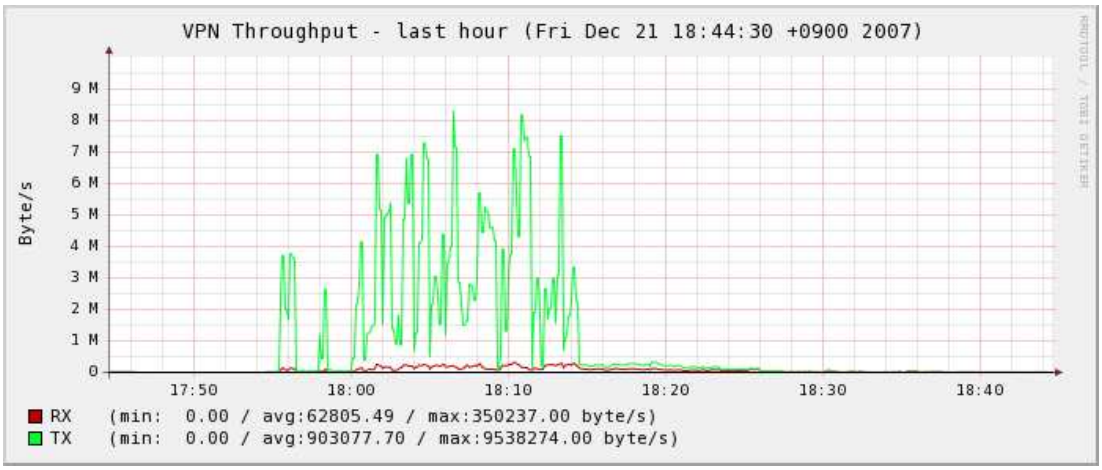

Figure 7: VPN throughput with package caching

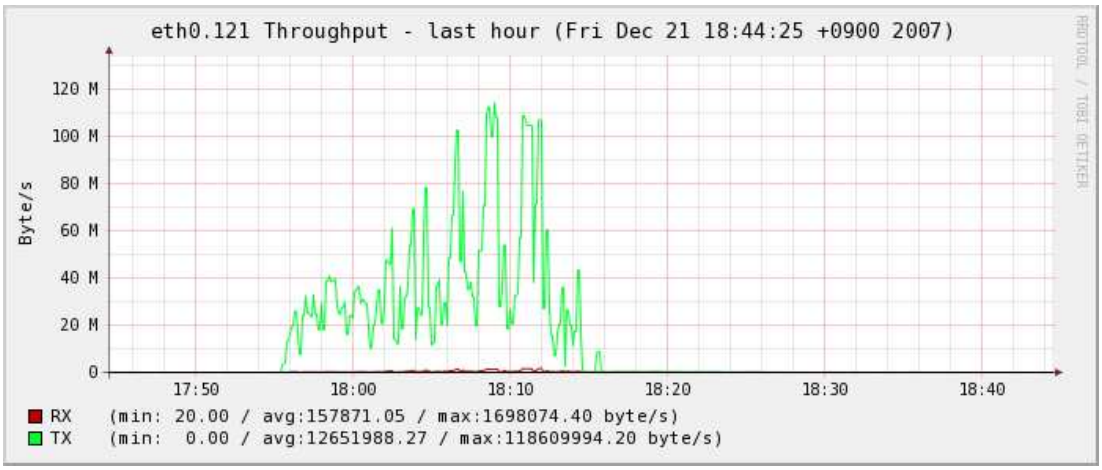

Figure 8: Private network interface throughput of the OpenVPN node in Cluster B with package caching

stopped at PXE booting due to a download error of a kernel and a small bootstrap system image (`initrd.img`). A PXE boot utilizes TFTP to get initial boot files, however their UDP packets are likely to be lost under a congested network. After the kernel and `initrd.img` are loaded from a CDROM image, an installing node starts to download initial images and packages from the virtual frontend in Cluster A. A cache server saves all downloaded files into its repository; the total amount of them is only 1.1GB for all Rocks Rolls. The test network is set to emulate a 20ms RTT latency, which is a slightly larger value than that for average RTTs disclosed by many ISPs for SLA in Japan.

Figure 7 shows the network traffic throughput inside the Ethernet VPN. During this experiment, network traffic in the downloading direction (TX in Figure 7) is much greater than that requesting messages to the virtual frontend. However, the throughput retains a much lower rate vis-a-vis installations of 138 nodes; it is dramatically reduced to approximately the amount of a set of downloaded files for one node. As in Figure 8, which shows outgoing network traffic from the cache server to installing nodes, the cache server correctly multiplexes concurrent downloading requests to target files. Its high throughput rate is reduced to the VPN throughput rate by package caching.

Figure 9 shows installation progress bars of all target nodes. We made this figure by analyzing HTTP server logs in the virtual frontend. A target node accesses several CGI scripts to get node settings and update information during installation. These access records are exploited in our installation progress analyzer.

As in Figure 9, dynamic kickstart generation is fairly slow and installation processes are ordered by its retry count. It is necessary to fix this issue for faster installation.

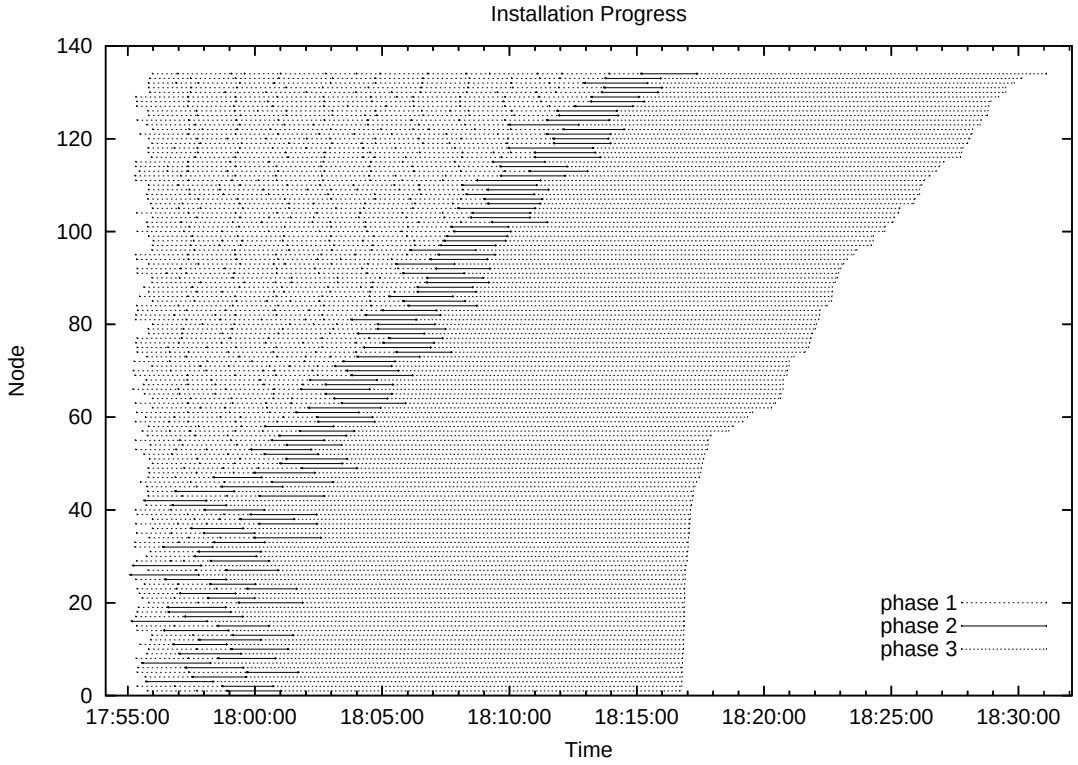

Figure 9: Installation progress with package caching. See Section 4.3 about installation phases. Note that progress bars are sorted by finish time for installation.

The fastest 50 nodes finished approximately at the same time, just after both VPN and cache download traffic goes down. This means these nodes retrieved the same downloading files simultaneously from the cache server which multiplexed them to unique file requests. Since the Rocks bittorrent-like mechanism does not work for partial file downloading, the fastest nodes need to get new package files from the cache server. In cases without it, these requests got directly to the virtual frontend node; a huge amount of data is transferred over the network, and it takes over 2 hours to complete a cluster setup in the tested environments. Other backbone traffic is suppressed for a long time.

Other slower nodes efficiently exploit the bittorrent-like mechanism. They download packages from other nodes which already have them. As Figure 8 shows, after the fastest nodes have downloaded all packages, all requests went to installing nodes, not to the cache server.

In the case where all packages are already cached before deployment, VPN traffic during installation is very small (e.g., under 400KB in our experiments). All packages are downloaded from the cache server, not from the remote virtual frontend over the VPN. The installation time of the fastest node becomes only 12 minutes in the emulated WAN. Therefore, a pre-caching mechanism, which stores common files in advance, is considered quite an effective solution to alleviate harmful network impact on other background traffic and speed up virtual cluster deployment over networks.

6 Discussion

In our implementation, all virtual nodes in a multi-site virtual cluster are located in a single network segment. Existing programs basically work for it in the same manner as in a physical cluster, without any modification. However, network intensive programs need special care to avoid performance degradation due to network latency and insufficient bandwidth over an Ethernet VPN. One possible solution is that users run network-intensive parts of programs at each site; virtual nodes in a multi-site virtual cluster are named in a different manner, based on their physical locations, so that users can optimize the running of their applications. The physical locations of the virtual nodes can be distinguished by their MAC address, which includes a physical cluster ID number used in management databases. Also, their IP addresses are selected from different ranges, and their host names are assigned with a prefix part for physical locations (e.g.,

`vcompute-7-123`, `7` is a site ID number). In addition, it is possible to utilize special middleware libraries addressing heterogeneous network topology (e.g., MPI library for WAN [8]).

Although the current implementation still does not utilize the migration technologies of virtual machines, we found that it could provide management flexibility to a degree. Since the (re)installation cost of a virtual compute node is very small, users can dynamically substitute some virtual nodes of a multi-site virtual cluster with other virtual machines at other sites with the minimum interaction. In addition, running jobs can be submitted to new nodes automatically by simple scripts. However, this approach is not fully transparent for users and applications; users may have to extend target applications to support dynamic node configuration. Moreover, it is difficult to move a virtual frontend node to a remote site by this approach; it has all management databases and tools for use inside of a virtual cluster, and cannot be reinstalled while the virtual cluster is working. We consider it possible, however, to exploit the live migration features of VMMs to move all virtual nodes transparently to other sites. Further studies of this point will be made in our future work.

7 Related Work

Although in the HPC and Grid research area many studies have been conducted for virtual clusters (e.g., [5, 9, 11, 22]), there are few virtualization systems which allow single-image virtual clusters over the Internet. VNet [19] is an experimental VPN program with dynamic adaption of VPN topology, which is used to connect distributed private networks of virtual machines served under a marketplace-based resource manager [17]. VioCluster [15] is a computational resource sharing system based on machine and network virtualization. Virtual machines on physical clusters are automatically grouped with VPNs on demand, in accordance with borrowing and lending policies between different administrative domains. The work queue of PBS is exploited to measure the demand for processing power in a domain. Compared with these systems, our system is intended to build large-scale virtual clusters over a WAN with physical clusters in a seamless manner. However, their key mechanisms are also applicable to our system with a straightforward extension of the current implementation. It is possible to establish more complicated network topologies rather than only a master/worker star topology, and dynamically change them in order to adapt to network conditions and application requirements. A multi-site virtual cluster can be expanded/shrunk automatically according to resource usage policies. It is developed to support cloud computing of virtualized computing resources. Compared with Amazon EC2 [1] and Virtual Workspace [22], our system focuses scalable management of distributed virtual machines at many sites.

8 Conclusion

For achieving the maximum scalability and flexibility of resource management in datacenters and computational centers, we have presented an advanced cluster management system for multi-site virtual clusters over WANs, composed of virtual machines hosted on widely-distributed physical clusters. The L2 extension mechanism of virtual machine networks enables existing cluster application programs to be utilized also for distributed computational resources in the same manner as on a single physical cluster. In addition, its package caching mechanism allows rapid deployment of large-scale multi-site virtual clusters under network latencies in WANs. Our experiments showed multi-site virtual clusters were feasible through the L2 network extension mechanism, and that the package caching mechanism was quite essential for scaling them up under WAN latencies. In future work, we will address fully-transparent migration of virtual clusters and latency-aware optimization with HPC libraries and advanced networking mechanisms.

Acknowledgment

The authors would like to thank the GtrcNET team of Grid Technology Research Center for their kind support, and for their hardware network emulator GtrcNET.

References

[1] AMAZON ELASTIC COMPUTE CLOUD. `http://aws.amazon.com/ec2`.

[2] BARHAM, P., DRAGOVIC, B., FRASER, K., HAND, S., HARRIS, T., HO, A., NEUGEBAUER, R., PRATT, I., AND WARFIELD, A. Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (2003), ACM Press.

[3] HOUSLEY, R., AND HOLLENBECK, S. EtherIP: Tunneling Ethernet Frames in IP Datagrams. RFC 3378, Sept. 2002.

[4] KODAMA, Y., KUDOH, T., TAKANO, R., SATO, H., TATEBE, O., AND SEKIGUCHI, S. GNET-1: Gigabit Ethernet network testbed. In *Proceedings of Cluster 2004: IEEE International Conference on Cluster Computing* (2004), IEEE Computer Society.

[5] KRSUL, I., GANGULY, A., ZHANG, J., FORTES, J. A. B., AND FIGUEIREDO, R. J. VMPlants: Providing and managing virtual machine execution environments

for grid computing. In *Proceedings of the ACM/IEEE Supercomputing 2004 Conference* (2004).

[6] LAN MAN STANDARDS COMMITTEE OF THE IEEE COMPUTER SOCIETY. *IEEE standards for local and metropolitan area networks: virtual bridged local area networks*. IEEE, 1999.

[7] LAU, J., TOWNSLEY, W. M., AND GOYRET, I. Layer Two Tunneling Protocol - Version 3 (L2TPv3). RFC 3931 (Proposed Standard), Mar. 2005.

[8] MATSUDA, M., KUDOH, T., KODAMA, Y., TAKANO, R., AND ISHIKAWA, Y. Efficient MPI collective operations for clusters in long-and-fast networks. In *Proceedings of Cluster 2006: IEEE International Conference on Cluster Computing* (2006), IEEE Computer Society.

[9] MCNETT, M., GUPTA, D., VAHDAT, A., AND VOELKER, G. M. Usher: An extensible framework for managing clusters of virtual machines. In *Proceedings of the 21st Large Installation System Administration Conference (LISA 2007)* (2007), USENIX Association.

[10] NAKADA, H., YOKOI, T., EBARA, T., TANIMURA, Y., OGAWA, H., AND SEKIGUCHI, S. The design and implementation of a virtual cluster management system. In *Proceedings of the first IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management (EVGM2007)* (2007).

[11] NISHIMURA, H., MARUYAMA, N., AND MATSUOKA, S. Virtual clusters on the fly - fast, scalable, and flexible installation. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)* (2007), IEEE Computer Society.

[12] OPENVPN. `http://openvpn.net/`.

[13] PACKETIX VPN. `http://www.softether.com/`.

[14] PAPADOPOULOS, P. M., KATZ, M. J., AND BRUNO, G. NPACI Rocks: Tools and techniques for easily deploying manageable Linux clusters. In *Proceedings of Cluster 2001: IEEE International Conference on Cluster Computing* (2001), IEEE Computer Society.

[15] RUTH, P., MCGACHEY, P., AND XU, D. VioCluster: Virtualization for dynamic computational domains. In *Proceedings of Cluster 2005: IEEE International Conference on Cluster Computing* (2005), IEEE Computer Society.

[16] SATRAN, J., METH, K., SAPUNTZAKIS, C., CHADALAPAKA, M., AND ZEIDNER, E. Internet Small Computer Systems Interface (iSCSI). RFC 3720, Apr. 2004.

[17] SHOYKHET, A. I., LANGE, J., AND DINDA, P. A. Virtuoso: A system for virtual machine marketplaces. Tech. Rep. NWU-CS-04-39, Northwestern University, July 2004.

[18] SQUID: OPTIMIZING WEB DELIVERY. `http://www.squid-cache.org/`.

[19] SUNDARARAJ, A. I., AND DINDA, P. A. Towards virtual networks for virtual machine grid computing. In *Proceedings of the Third Conference on Virtual Machine Research And Technology Symposium (VM'04)* (2004), USENIX Association.

[20] VMWARE INFRASTRUCTURE. `http://www.vmware.com/`.

[21] VTUN: VIRTUAL TUNNELS OVER TCP/IP NETWORKS. `http://vtun.sourceforge.net/`.

[22] ZHANG, X., FREEMAN, T., KEAHEY, K., FOSTER, I., AND SCHEFTNER, D. Virtual clusters for grid communities. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID2006)* (2006), IEEE Computer Society.