Efficient Distribution of Virtual Machines for Cloud Computing

Matthias Schmidt, Niels Fallenbeck, Matthew Smith, Bernd Freisleben
Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str. 3, D-35032 Marburg, Germany
{schmidtm, fallenbe, matthew, freisleb}@informatik.uni-marburg.de

Abstract— **The commercial success of Cloud computing and recent developments in Grid computing have brought platform virtualization technology into the field of high performance computing. Virtualization offers both more flexibility and security through custom user images and user isolation. In this paper, we deal with the problem of distributing virtual machine (VM) images to a set of distributed compute nodes in a Cross-Cloud computing environment, i.e., the connection of two or more Cloud computing sites. Ambrust et al. [3] identified data transfer bottlenecks as one of the obstacles Cloud computing has to solve to be a commercial success. Several methods for distributing VM images are presented, and optimizations based on copy on write layers are discussed. The performance of the presented solutions and the security overhead is evaluated.**

I. Introduction

The Cloud computing paradigm is sometimes viewed as the commercial successor of the academic Grid computing paradigm. Although this view is debatable, there is a strong trend towards this new paradigm. Cloud computing as offered by companies such as Amazon (also called Infrastructure as a Service (IaaS)) is easier to use than the typical Grid and offers more user rights (i.e. super-user rights) and tighter security. This has led to much greater commercial adoption of Cloud computing than of Grid computing. However, unlike a Grid, a Cloud does not assist its users with the selection and use of different remote Cloud sites. Due to commercial interests, a Cloud provider usually has a proprietary system where a user creates an image that only runs on the provider's site. This represents a significant step back compared to Grid computing. While many cross-site technologies applied in Grid computing such as virtual organization management and single sign-on can quite easily be adopted by Cloud computing, the actual configuration and management of user software in the Cloud is significantly different compared to the traditional Grid. In the Cloud, users expect to be able to install complex software with super-user privileges on-demand.

If the Cloud paradigm is to be extended to facilitate cross-provider utilization of resources, several challenges need to be solved. One of the requirements for Cross-Cloud computing discussed in this paper is the capability to create and efficiently distribute interoperable virtual machines. First, we briefly present a Cross-Cloud virtual machine creation solution that is based on a layered approach to allow a single user image to be deployed to multiple Cloud sites, including desktop Cloud sites. Next, we analyze several methods for the bulk transfer of these layered images to the Cloud sites. Unlike the transfer of virtual machines in Grid computing [17], [18], which can be planned and executed in advance by the meta-scheduler, this transfer must be performed on-demand without pre-knowledge of the transfer or the recipients of the transfer. This creates a much tighter time constraint. We therefore present a hybrid solution that adaptively selects the optimal transfer method depending on network capabilities and Cloud site constraints. Finally, we present a study of the performance of our system with and without security and discuss a re-transfer optimization method based on copy-on-write layers. The proof of concept implementation is integrated into the XGE v2 [17].

The paper is organized as follows. Section II states the problems involved in virtual machine distribution. Section III presents several algorithms for distributing virtual machine images to a number of nodes. Section IV discusses implementation issues. Section V presents results of several performance measurements. Section VI discusses related work. Section VII concludes the paper and outlines areas for future work.

II. Problem Statement

In a setup with a small number of compute nodes, the problem of distributing VM images to physical compute nodes may be negligible, but in a common Cloud computing setup with hundreds or thousands of nodes, the time needed to distribute virtual machines images to their destinations is crucial. On the one hand, it is important for a cluster administrator who is interested in an efficient utilization of his/her resources. On the other hand, it is important for a user who is interested in fast response times.

Apart from the VM image distribution time, the network load caused by the distribution process itself is a critical factor. A simple and inconsiderate distribution method could lead to significant traffic on the core network components, possibly leading to long lasting transfers, congestion or, in the worst-case, packet loss. All stated networking problems prevent the continuous utilization of resources: If the network between the resource manager/cluster scheduler and the compute nodes is congested, the execution of commands and new tasks is delayed or even prevented. An existing interactive session (i.e. an administrative SSH session) can become unresponsive or is even interrupted.

Furthermore, while current Cloud solutions usually run in a single domain using dedicated clusters, future Cloud systems will likely become more heterogeneous and distributed. The deployment mechanism should be able to cope with both cluster environments and more heterogeneous, distributed Cross-Cloud computing environments, which also includes desktop systems as found in private Clouds.

Another issue is the redistribution of already transferred VM images. If a user's task is started for the first time on a new infrastructure, his/her data VM image is distributed to all destination nodes. After the computation, the image remains on the local disk of the compute node, only the parts that change are stored on shared storage. If the user starts another task that is scheduled on the same compute nodes, it would be a waste of time and bandwidth to retransfer the VM image. Thus, a technique is needed that not only avoids useless retransmissions, but also allows the distribution of updates (i.e. security updates for the VMs) for the images already stored on the compute nodes. Furthermore, cross-site distribution must be supported, i.e. it is desirable to reuse as much components of an already installed VM image on a remote site as possible.

From the discussion above, the following requirements can be derived:

1) The primary goal for efficient VM image distribution is to avoid congested links between the node where a VM image resides and the compute nodes.
2) The communication time between the compute nodes should be minimized, whereas the actual data exchange should be maximized: distribute maximum data in minimal time.
3) To face the problems regarding VM image updates, an integrated solution is needed that reduces transmission overhead and allows an easy update to be performed.

III. VM Image Distribution Methods

This section presents several methods for distributing VM images from one node as the source to a set of selected compute nodes as the destinations.

A. Network File System

The traditional way to distribute a number of VM images in a Cloud site would be to use the Network File System (NFS). NFS is well established and provides a simple way, both for the administrator as well as the user, to make files available remotely. The central NFS server stores the VM images and the worker nodes retrieve copies on demand. This leads to multiple point-to-point transfers. Furthermore, when multi-gigabyte files are accessed by a large number of nodes simultaneously, NFS shows erratic behavior. This lead to a number of crashes during our tests. To avoid this behavior, the workers would need to synchronize their transfers, so as not to interfere with each other. Further common NFS problems like stale NFS file handles (which can, for instance, be caused by a re-export of the NFS exports) can lead to stalling VMs or even Xen Domain 0 nodes. Finally, NFS is not well suited for use in a Cross Cloud Computing scenario. Exporting NFS outside a local network is not trivial and is difficult to secure. For these reasons, a simple unicast deployment algorithm was developed to serve as a benchmark instead of NFS.

B. Unicast Distribution

A straightforward method to distribute VM images is sequentially copying them to the destination nodes. The benefits of this method are that it is fairly simple to understand and to implement and works in Cross Cloud scenarios, but its drawbacks are long transfer times and network congestion.

C. Binary Tree Distribution

To avoid network congestion and to allow parallel transfers, a binary-tree based distribution method can used. In this method, all compute nodes are arranged in a balanced binary tree with the source node as its root. The balanced tree property guarantees that the depth of the leaves differs by at most one. Since a balanced tree has a height of $log_2(n)$ with n being the number of compute nodes, the transmission time is $O(t \cdot log_2 n)$ where t is the time needed to transfer a VM image from source to destination.

All transfers are synchronized to avoid that a transfer on a child node starts before the data from the parent is available. Correct synchronization can be achieved by either synchronizing every level of the tree or by synchronizing every compute node. Whereas the first method is easier to implement, the second method guarantees a higher throughput and thus lower transmission times.

This method reduces the time needed to transfer a VM image to all compute nodes as compared to the unicast distribution method. The method can be used for Cross-Cloud computing, if either all compute nodes are located inside the same subnet (e.g., if two remote sites are connected with a Virtual Private Network (VPN) or if the compute nodes have public IP addresses ([15], [1]).

Fibonacci Tree: Our binary tree is in fact a Fibonacci tree. A special feature of the Fibonacci tree is that it is rebalanced after insertions or deletions using rotations. Thus, if the distribution of a VM image is interrupted due to a node failure, the tree needs to be rebalanced to guarantee seamless transfer. Besides rebalancing, there is some further action involved to resume the transfer to all nodes that are now rebalanced. It could be possible that a node has a different parent after rebalancing so the transmission for that node must be restarted. The child nodes of that node stall automatically until it has sufficient data to continue the transmission.

D. Peer-to-Peer Distribution

Another method is to use peer-to-peer (P2P) technologies for VM image transfer. We have chosen the BitTorrent protocol [4] that is briefly described below.

BitTorrent Protocol: BitTorrent was designed as a protocol for fast, efficient and decentralized distribution of files over a network. Every recipient downloading a file supplies this file (or at least parts of it) to newer recipients also downloading the file. This reduces the overall costs in terms of network traffic, hardware costs and overall time needed to download the whole file.

The node hosting the source file starts a *tracker* that coordinates the distribution of the file. Furthermore, a file (a so called *torrent file*) containing meta-data about the source file (URL of the tracker) is generated and must be distributed to all clients (either active with a push mechanism or passive with all clients downloading the torrent file from a web server). Clients, also called *peers*, connect to the tracker that tells them from which other peers pieces of the file are available for download. A peer that shares parts or the complete file is called a *seeder*. Using this technique, sharing files between multiple peers benefits from high speed downloads and reduced transmission times compared to other techniques.

Due to its distributed nature, BitTorrent perfectly fulfills the needs of Cross-Cloud computing. It can be used to distribute VM images between two dedicated VM image pool nodes on remote sites if the networks where the actual compute nodes reside are private or not connected in a VPN. The distribution from the pool node to the compute nodes can also be accomplished by BitTorrent or another suitable protocol. If the network setup permits it (compare subsection III-C), a direct VM image transfer to the compute nodes is desired to save additional local distribution time on each Cloud site. More on this issue is discussed in Subsection III-F.

VM Image Distribution: To distribute a VM image to the compute nodes, a torrent file containing the URL of the tracker needs to be generated. The tracker in this case is the source node hosting the VM images. Furthermore, a seeder for the VM image needs to be started on the source node. To begin with the actual distribution process, BitTorrent clients are started remotely on all compute nodes. They connect to the tracker, load and seed the VM image immediately. After the process is finished, all nodes carry the complete and correct VM image.

E. Multicast

The distribution of VM images via multicast is the most efficient method in local area network environment. The design of the multicast module can be kept simple if no ordering guarantees are given by the master node. Ensuring reliability in this case is delegated to both the sender and the receivers. The former can handle data reliability by tuning either the number of redundant packets (increasing the CPU load) or by tuning the stripe size, i.e., the number of packets sent in a block (this could decrease the CPU load but increase the data loss). IP based multicast can also be used to transfer the data. It is supported by most networking hardware out of the box. Modern hardware is able to handle multicast transfers even better, i.e. it is possible to distribute the multicast packets only over selected links according to their multicast group membership. This is accomplished if the switches support *IGMP Snooping* or the *Cisco Group Management Protocol* (CGMP). The switches can inspect the Internet Group Management Protocol (IGMP) packets and adjust their switching tables accordingly. This transfer method is likely to be scalable to large-scale installations, e.g. the ones used by modern Cloud computing infrastructure providers, since the used hardware is capable of scaling to thousands of hosts and multicast groups.

F. Cross-Cloud Transfer

Although multicast is an ideal method to transfer data within a local area network, it is not the preferred method to transfer data over network boundaries, since routing multicast packets between networks requires special protocol support at the network core, such as the Distance Vector Multicast Routing Protocol (DVMRP). Furthermore, it could be possible that data is shared between two private, physically separated desktop Clouds within the same organization, but the network policy forbids that data is transferred via multicast over the network backbone connecting the two desktop Clouds. Choosing BitTorrent or the binary tree based distribution mechanisms to overcome this limitation is an interesting option.

It is also possible to use a combination of two methods when performing Cross-Cloud computing. If the compute nodes in the private Cloud are not directly accessible, because they do not have public IP addresses, it is not possible to use e.g. BitTorrent directly. Here, it is possible to transfer the VM images or only selected layers to publicly accessible pool nodes to cache the data there and transfer it locally via multicast afterwards.

G. Encrypted vs. Unencrypted Transfer

If private data (whether it is sensitive data or not) is transferred over an insecure link or over network boundaries, it should be encrypted. However, encryption involves additional costs: The cryptographic computations produce CPU load on both the sender and receiver, and the average transfer rate is reduced. Unencrypted transfers are favorable if the data itself is public and nobody cares if it is stolen. Furthermore, if the network is isolated from the outside world (which is common in high performance computing setups today), the increased transfer rate is a benefit compared to encrypted transfer.

H. Avoiding Retransmission Overhead

To avoid the retransmission of a VM image that is already present and to avoid the need for huge amounts of disk space at the destination nodes, an approach based on copy-on-write (COW) layers is proposed. A layered filesystem is a virtual filesystem built from more than one individual filesystem (layer) using a COW solution like UnionFS. A VM image is realized by a COW virtual disk consisting of three or more layers. At first, only a *base layer* is present. This layer contains a complete installation of a Linux operating system. On top of the base layer, a *site layer* is placed. It contains all modifications needed to run this VM image on a particular site, such as a local or a remote one or e.g. Amazon's EC2 service. Special configuration information depending on the infrastructure, e.g. LDAP settings, name servers, NFS server, are stored in the site layer. The third layer is the *user layer* that contains all modifications made by the user. The size of this layer ranges from small (only a basic software stack with special libraries and e.g. MPI programs is installed) to very large (a complex, proprietary application with a license server is installed).

There are different usage scenarios for layered VM images, as shown in Figure 1. The user may set up the user layer of

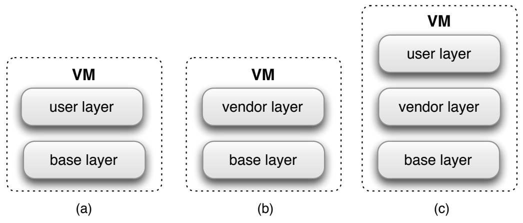


Fig. 1. Usage scenarios for a layered VM

a VM completely on his or her own, leading to a two-layered image (a). Alternatively, a software vendor (or one of its sales partners) may provide a layer containing one of its products (b). Even in this case, the user may set up a custom layer on top of the vendor layer, containing extensions or other required tools (c). Nevertheless, other usage scenarios might be possible and should be supported.

Usually, a large number of similar jobs are submitted to a site, where each job represents a part of the problem to be solved. A simulation process is typically divided into numerous independent tasks that will be concurrently executed on many compute nodes. These jobs are executed in multiple instances of the VM, and they are most likely executed consecutively. Retransmitting the user layer again every time is contradicting to the whole idea of image distribution compared to the use of NFS. Thus, the user layer (as well as the base and possible vendor layers) should be cached locally on the individual nodes. To ensure that a cached layer is in a working state when it is used to form a root filesystem, it is best to make sure that it is not changed with respect to its original state. This can only be achieved by prohibiting write access to the layer during runtime.

Using COW virtual disks has several advantages. The base layer remains nearly the same over a long period of time. An update, for example, is only necessary if security or errata patches have to be installed. This allows us to store the base layer on the compute nodes and avoid expensive retransmissions. It also removes the need for a large amount of disk space because the large base layer is cached. Updates to the site layer are also quite rare, thus it can also be stored on the compute nodes. Only the user layer changes often, because users tend to install new software, reinstall existing software or change settings. Thus, most of the time, the user layer is the only part of a VM image that needs to be retransmitted, leading to significant savings in the overhead for retransmission.

IV. Implementation

In this section, the implementation of the proposed methods in a Cloud computing environment is described. To leverage existing software components, the proposed methods have been implemented in the context of the XGE v2 [17], [16]. It supports VM handling and task management and one instance is installed on every Cloud site to support the Cross Cloud Computing paradigm. Communication between the instances is possible over an encrypted channel. The software infrastructure used on every site is as follows. Every valid user can use the Image Creation Station (ICS) to create his/her own customized VMs to use as a computing base in the Cloud. A user can log into his or her created VM and install and customize the VM to his or her needs. After this user launches a task, the XGE chooses the user's own VM image, generates several virtualized execution environments, distributed them to the compute nodes (either local or cross-site) and boots them. The integration of the distribution algorithm is discussion in the following subsections.

A. Unicast Distribution

The implementation of the unicast distribution method is simple as the user's VM image is copied sequentially to all destination compute nodes. For obvious reasons this method is not used in production environments.

B. Binary Tree Distribution

Binary tree distribution is more complex than using the unicast distribution method. A binary tree is generated once all destination nodes are known by the resource manager. The root of the tree is always the node carrying the VM image. A multi-threaded implementation ensures that the copy operations are performed in parallel, i.e. the VM image is copied from the first node to its two children in the tree. Once a reasonable amount of data (a variable threshold; in our implementation it is set to 50 MB) has arrived at the children, a new copy-thread for every children is spawned. Thus, a continuous flow of data through the entire tree is maintained.

To ensure that a new copy-thread is started without having any data to transfer, all copy actions are synchronized. A node must not send data until it has sufficient data to start a transfer. For example, if the network transfer is interrupted between two nodes in the tree, all children are blocked until the interruption is resolved. To avoid endless waiting, the copy process is aborted after a certain time interval. A repeated transmission or a complete cancelation is possible, depending on the local administrator's preferences.

C. Peer-to-Peer Distribution

As already mentioned, the peer-to-peer distribution method is based on BitTorrent. *Enhanced ctorrent*[1] is used for the seeders and the tracker. Once the user's VM image is ready for distribution, the XGE creates a torrent file. This small file is distributed to all destination nodes with the binary tree algorithm. The destination nodes connect automatically to the tracker (which itself acts as the initial seeder) and start downloading pieces of the VM image. Due to the nature of BitTorrent, multiple transfers of different files to the same node are possible. Of course, the total bandwidth is shared between all running BitTorrent peer instances on a node. Once the download is finished, all peer processes are terminated and the tracker shuts down itself. Apart from that, BitTorrent takes care of the integrity of the VM images. Additionally, the XGE can also compute a cryptographic checksum (SHA-256) to ensure the integrity. This is optional, because computing a hash over a large file is time and hardware intensive.

[1]http://www.rahul.net/dholmes/ctorrent/

D. Multicast

The multicast distribution method is based on UDPcast[2]. It provides an easy and seamless way to transfer data via multicast. Every compute nodes has a receiving client installed on the Xen domain 0 and the sender is installed on the machine hosting the VM images. Once a transmission is started, the sender starts the multicast distribution and all receivers store the VM image at a predefined location. All distributing threads are synchronized and shutdown once the transfer is complete.

E. Cross-Cloud Transfer

Depending on the network structure, the XGEs are either aware of the public IP addresses of all compute nodes in Cross-Cloud computing environments or know at least the IP addresses of a publicly reachable VM image pool node. By default, the XGEs assume that the compute nodes are publicly reachable. Otherwise, it is up to the local administrator of the Cloud site to configure it appropriately. BitTorrent is used when VM images are transferred between two remote Cloud sites, and multicast for the local distribution.

F. Encrypted vs. Unencrypted Transfer

Depending on the distribution method, the transfer can be either encrypted or unencrypted. For example, BitTorrent and Multicast traffic is commonly unencrypted and thus additional security mechanisms like IPsec [8] must be put in place. The binary tree method can be used with unencrypted (to save bandwidth and CPU overhead in protected network environments) or encrypted (for use in non-confidential networks) transfer. Authentication in our implementation is achieved by using public key cryptography. The source node has a secret key and all destination nodes have the registered public key of the source node. Traffic encryption is performed by using OpenSSL [13] that itself uses an asymmetric key for the handshake and a symmetric key for the actual encryption procedure.

G. Avoiding Retransmission Overhead

The presented three-layer COW disk image architecture is based on UnionFS [19] that is part of the Linux mainline kernel since 2007. The base layer hosts a Debian GNU/Linux stable based Linux distribution with a Xen-capable kernel. All tools needed to ensure seamless out-of-the-box operation are included, such as the complete Debian package management, standard compilers and interpreters and a basic set of libraries including development headers. The base layer is able to run on every infrastructure supporting the Xen virtual machine monitor. The site layer contains several additions needed to run the generic base image on different sites. The changes to the base layer include special settings such as name resolution (DNS), IP configuration (DHCP-based, static addresses or peer-to-peer auto configuration) or user management (LDAP or NIS interfaces). Typically, the size of the site layer is small, often between 10 KB or 1 MB.

[2]http://udpcast.linux.lu/

The user layer contains all the modifications made by the owner. The Image Creation Station is provided to create, manage and customize VM images. The ICS offers either a web frontend intended for non-computer science users or a web service interface intended for advanced users who want to integrate the image creation/modification process, for example, within BPEL workflows [2].

To ensure that the modifications made by the user will not prevent the VM image from running (imagine that the user deleted some crucial system files), the base layer as well as the site layer are mounted read-only during execution time. Furthermore, the site layer overwrites any changes of the user to configuration files that are also touched by the site layer. Of course, this limits the user's freedom of image customization, but we think that having a limited freedom is better than having no running VM at all (or ever worse, a VM that is stuck in the boot process and blocks resources).

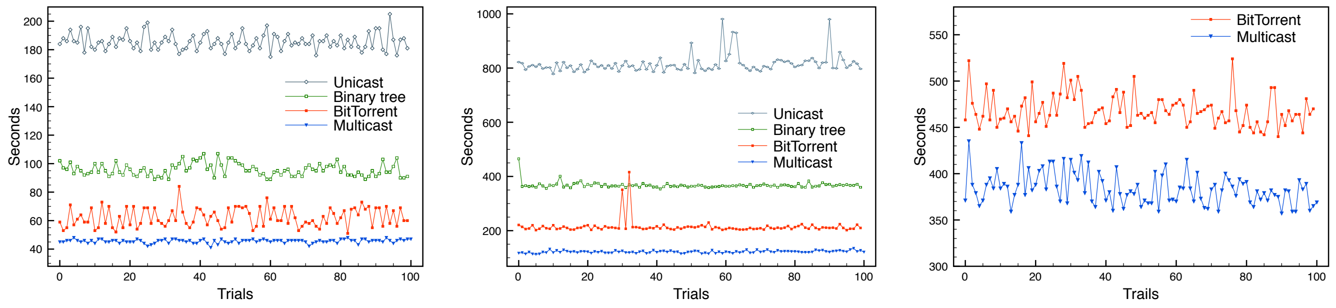
V. Experimental Results

All presented distribution mechanisms have advantages and disadvantages. While multicast and BitTorrent are promising candidates to distribute large amounts of data to hundreds or thousands of clients, they are not suitable for every scenario. As already mentioned, routing multicast packets between networks is not possible without additional protocols, and BitTorrent needs a torrent file for every transfer of a different VM image. This generates an overhead if only small pieces of data are transferred. On the contrary, the tree-based distribution method only scales to a certain degree, but it is ready to go and needs no preliminary work like generating and distributing torrent files. Thus, choosing the right method depends on the amount of data to distribute and the number of receiving nodes. By default, the XGE transfers data smaller than 300 MB with a tree based method if the number of target nodes is smaller than 10.

A. VM Distribution Methods

To evaluate the performance of the presented distribution methods, several measurements have been performed. Our test environment consists of Intel Xeon 2.5 GHz machines with a total of 80 cores connected with a 1 Gbit/sec ethernet. All machines have 16 GB RAM and Debian Linux stable with Xen 3.0.2 installed. The local hard disks have a capacity of 250 GB.

In Figure 2, all measurements with unencrypted transfer are shown. As already stated, the unicast distribution method is the slowest method to distribute the VM images and is thus not really suitable for practical use, it only serves as a reference value. The binary tree distribution method is significantly faster than unicast distribution, but slower than the P2P distribution method based on BitTorrent and the multicast method. Even though BitTorrent was designed with desktop file sharing in mind, it performed very well in the structured cluster environment. Multicast is significantly faster than all other algorithms. In Figure 2 (c) the results of unicast and the binary tree are skipped, because they are about three times slower and thus unusable to transfer larger data sizes. The



(a) 1024 MB (b) 4096 MB (c) 8192 MB

Fig. 2. Deployment times for different data sizes and algorithms

mean values (of 100 measurements) of all transfers are shown in Table I. We measured the transfer times with different VM image disk sizes, where 512 MB refers to a basic image or layer with few modifications and 8192 MB to a highly customized VM image or layer. Based on our experiments with several users, the average VM image size is about 4 GB and the average user layer size is about 1 GB.

Image Size	**Unicast**	**Binary tree**	**BitTorrent**	**Multicast**
512 MB	89.4 s	50.0 s	29.0 s	27.9 s
1024 MB	191.1 s	95.7 s	62.3 s	45.5 s
2048 MB	401.0 s	183.8 s	120.9 s	90.6 s
4096 MB	815.2 s	367.4 s	214.3 s	121.8 s
8192 MB	1634.1 s	727.6 s	465.0 s	383.5 s

TABLE I
MEASURED MEAN VALUES OF ALL DEPLOYMENT METHODS

When encrypted transfer over the binary tree is used, encryption, transfer and decryption are performed in parallel. The encryption methods offered by BitTorrent clients mainly aim to prevent filtering of BitTorrent traffic by ISPs. Thus, both the algorithm (RC4) and the bit lengths (60-80) used are not sufficient for confidential data in Cloud environments. Thus, for the BitTorrent and multicast cases the entire VM is encrypted using 128-bit blowfish before transfer, and can only be decrypted once the entire file has been received, which significantly slows down the approach.

In the P2P distribution method, a torrent file needs to be generated for every distribution process, thus the time needed for the generation of the torrent file must be added to the actual transfer time. The generation time grows with the size of the file – generating a torrent file for a 512 MB VM image takes less time than for a 8192 MB VM image. As a compromise, we measured the generation time needed for a 4096 MB VM image. We conducted 100 trials to calculate a mean of about 27 seconds. Thus, the complete time needed to transfer a 4096 MB VM image is about 241 seconds (instead of 214 seconds without considering the generation of the torrent file) on the average. Obviously, this is still better than the transfer time for a 4096 MB VM image (367 seconds) using the binary tree distribution method.

The jitter that can be seen in the figures is caused by network activity produced by other users in our test environment. The effect of others users' activities is, however, small compared to the jitter effect of the BitTorrent protocol. Thus, the jitter that can be seen is mainly caused by the BitTorrent protocol itself.

B. Multi-Layered Virtual Machines

Adding COW layers to VMs using UnionFS produces additional costs when intensive file related tasks are performed. We conducted measurements using the bonnie++ [5] benchmark as a well-known testing suite for file systems to investigate this overhead. The used VMs have 1024 MB RAM and a single assigned CPU core. A total of 100 tests was performed. The average of the results is shown in Figure 3. When writing the file block by block, the non-layered VM outperformed the layered VM (206277 KB/s vs. 198015 KB/s), thus the COW layer introduces a slight performance reduction. The character test does not reveal any notable difference (48775 KB/s vs. 48319 KB/s), whereas in the rewrite test the layered VM had a significantly higher throughput than the non-layered VM (67951 KB/s vs. 33472 KB/s). This is due to the COW cache of the UnionFS file system. As a conclusion, it can be stated that the introduction of the additional layers consumes some performance if files are written in large blocks. Once this step has been performed, the performance benefits from the effective caching of the layered filesystem are evident. Due to the fact that most files of a regular job are written in the user's home directory that is natively accessible, the overhead only comes into play in certain, special circumstances.

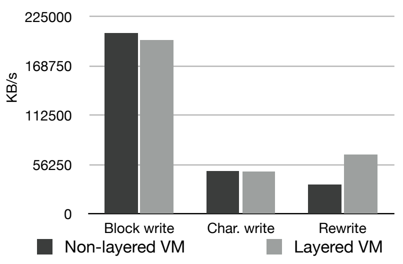


Fig. 3. Results of the bonnie++ benchmark with layered and unlayered VMs

Without the multi-layered file system, every time the user updates his or her VM and submits a job, the complete disk image of this machine has to be copied to the particular nodes (because a VM that was cached earlier is marked as invalid after a software update). When the multi-layered file system is used, only the user layer needs to be copied, which is significantly smaller. The base layer needs to be copied only once, because it is cached locally at the compute nodes. In case of an update, only the changes are copied and merged into the corresponding layer. We conducted a measurement of the transfer time in both cases, comparing a VM with and without a multi-layered file system.

		Transfer times		
Size (MB)		single site	multi site	
			uncompressed	compressed
Single disk image	691	40.59 secs	660.83 secs	460.12 secs
Base layer (BL)	666	39.12 secs	636.92 secs	443.47 secs
BL update	72	15.06 secs	106.23 secs	100.11 secs
User layer	67	14.45 secs	101.51 secs	91.58 secs

TABLE II
TRANSFER TIMES OF VM IMAGES AND FILESYSTEM LAYERS

Table II shows the measured time needed to transfer different images from one compute node to another without compressing the data during the copy process. The difference between the size of the single disk image and the sum of the sizes of base and user layer (691 MB vs. 733 MB) is due to the fact that the base layer as well as the user layer each contain a package database. Furthermore, the updated packages are cached within the respective layers, which also adds some space to the layers. We conducted 60 transfer operations to calculate a robust mean value. When VM images must be copied between remote sites, the time needed for the copy operations increases dramatically. The table also shows the measurements of uncompressed and gzip-compressed data transfer between compute nodes on two different academic locations connected by the German Research Network (DFN).

Summing up, without the multi-layered file system the amount of data to be transferred for the VM including the update is about 1380 MB, taking about 81, 1330 or 925 seconds (LAN, WAN, WAN compressed). Using our solution, the amount of data reduces to 140 MB, taking about 34, 212 or 196 seconds when the base image is already cached or 805 MB and 73, 850 or 640 seconds otherwise, although the latter case should be rare. This means that the use of the multi-layered file system saves up to 90% traffic and 60% – 85% of the time in the scenario.

VI. Related Work

A. VM Distribution and Migration

Most work done in the area of VM image distribution has been performed in conjunction with VM migration. This means that an application or the operating system as a whole is migrated over a network. All important operating system aspects (i.e., saving the state of the CPU, network, memory etc.) are covered by VM migration. Thus, distributing the VM image over the network is only part of a complete procedure and not covered in detail.

Sapuntzakis et al. [14] show how to quickly move the state of a running computer across a network, including the state in its disks, memory, CPU registers and I/O devices. The authors use several techniques to migrate a VMWare *x86* VM from one machine to another. The paper focuses on distribution over slow links (the primary example is a 384 kps DSL link). To achieve this goal, a self-made COW layer connected to the VMWare GSX Server is used. This layer attaches a bitmap to every COW disk. If a block is written/freed on one of the disks, the associated bitmap entry changes. Thus, instead of transmitting the entire COW disk, only the bitmap file is transferred, compared against an older, already present version and only the blocks that have changed since the last update are transmitted. To speed up the transfer over low-bandwidth links, only a hash instead of the data itself is transferred. If the receiver has the data matching the hash on local storage, it uses this data. If not, it requests the data from the server. As the authors state in the paper, the presented approach is not intended for high-bandwidth environments. Nevertheless, their approach presents a number of good inspirations related to VM image distribution.

Nelson et al. [12] describe the design and implementation of a system that uses virtual machine technology to provide fast, transparent application migration. The system is called *VMMotion* and part of the VMWare VirtualCenter product. The actual migration involves severals steps: the selected VM's memory is pre-copied to the destination, while the original VM continues running. Then, the VM is suspended and the VM is transferred to the destination. The destination takes over control and resumes the suspended VM. Finally, the remaining memory state is copied. The actual transfer process happens over SCSI storage. All VMs are attached to a Storage Area Network (SAN) and thus the SCSI disk can be reconnected to the destination machine. Due to the fact that we do not use VMWare for our approach and thus have no access to this feature, this approach is not feasible for us. Furthermore, we do not have access to a SAN to use related Xen capable techniques.

Kozuch et al. [9] present an approach called Internet Suspend/Resume (ISR). ISR is a hypothetical capability of suspending a machine on one Internet site, traveling to another and resuming it there. To achieve this, the authors use VM technologies (VMWare) and distributed file systems (NFS). All VMs are stored on a shared folder that itself is shared with all participating machines. On suspend, the VM is shut down and saved on the disk. This disk can now be used to resume the VM on a remote destination. While this scenario is feasible for closed environments, it is not feasible for Grid computing. Due to the fact that the VM disk could be accessed by others, this could lead to a potential information leak. The authors also present some thoughts on further improvement, especially in the area of image deployment. Their thoughts served as an inspiration for the work done in this paper.

Wolinsky et al. [21] describe a system of VM-based sandboxes deployed in wide-area overlays of virtual workstations (WOWs). They feature a DHCP-based virtual IP address allocation, a self-configured virtual network supporting peer-to-peer NAT traversal, stacked file systems and IPSec-based host authentication and end-to-end encryption of communication channels. Due to the use of stacked file systems, their approach is similar to ours, but they do not cope with the aspect of VM image distribution. The authors assume that all layers (called Base, Module and Home) are already installed.

B. Grid Virtualization

The Globus Team [6], [7] has identified the need for integrating the advantages of virtual machines in the Cluster and Grid area. It is argued that virtual machines offer the ability to instantiate and independently configure guest environments for different users. Nimbus (with its predecessor Virtual Workspaces) is able to provide virtual clusters on demand. In contrast to the XGE, Nimbus is not able to use scheduling decisions of resource managers for jobs in a Cloud environment.

VMPlant [10] is a Grid service for automated configuration and creation of virtual machines based on VMware that can be cloned and dynamically instantiated to provide homogeneous execution environments within distributed Grid resources. The focus of this work is the definition of a framework for virtual machine management and the representation of software requirements through a directed acyclic graph.

VSched [11] is a system for distributed computing using virtual machines to mix batch and interactive VMs on the same hardware. Implemented as a user level program, it schedules virtual machines created by VMware GSX Server [20]. VSched is designed to execute processes within virtual machines during idle times on workstations. Processes are executed while users are not producing a high CPU load, e.g. while only a word processor is used or the web is surfed.

VII. Conclusions

In this paper, we have presented several methods for distributing virtual machine images to physical machines in a Cloud computing environment no matter if it is a private Cloud connected via a fast local area network or several remote Cloud computing sites connected via the Internet. We have studied their performance in case of encrypted and unencrypted data transfer: unicast distribution, binary tree distribution, peer-to-peer distribution based on BitTorrent and finally multicast. Our evaluation showed that multicast offers the best performance. Nevertheless, when transferring VM images between remote sites, BitTorrent is the method to choose. It is even possible to combine two methods to overcome limitations caused by the transfer over network boundaries. To avoid the retransmission of a VM image that is already present at a destination node and to make Cross-Cloud computing work, an approach based on copy-on-write layers has been presented. The evaluation of the layered file system showed that it saves a considerable amount of traffic, up to 90%.

There are several areas for future work. For example, integration of strong encryption and decryption directly into the BitTorrent client would significantly increase the performance and security of encrypted BitTorrent image deployment. Furthermore, the scalability of the presented solutions should be investigated in a test environment with machines in a wide area network environment. Finally, studying efficient change detection methods for storage blocks to avoid retransmissions is an interesting area for further research.

VIII. Acknowledgements

This work is partly supported by the German Ministry of Education and Research (BMBF) (D-Grid and HPC Initiative).

References

[1] Amazon Web Services LLC. Feature Guide: Amazon EC2 Elastic IP Addresses. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1346, 2008.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. *Business Process Execution Language for Web Services, Version 1.1*. Microsoft, IBM, Siebel, BEA und SAP, 1.1 edition, May 2003.

[3] M. Armbrust, A. Fox, R. Griffith, and A. Joseph. Above the Clouds: A Berkeley View of Cloud Computing. *University of California*, Jan 2009.

[4] BitTorrent. BitTorrent. http://www.bittorrent.com/, June 2008.

[5] R. Coker. bonnie++ Benchmark. http://www.coker.com.au/bonnie++/, 2009.

[6] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual Clusters for Grid Communities. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 513–520. IEEE Computer Society, 2006.

[7] K. Keahey and T. Freeman. Contextualization: Providing one-click Virtual Clusters. *IEEE International Conference on eScience*, 0:301–308, Jan 2008.

[8] S. Kent and R. Atkinson. RFC 2401: Security Architecture for the Internet Protocol. http://tools.ietf.org/html/rfc2401, 1998.

[9] M. Kozuch, M. Satyanarayanan, I. Res, and P. Pittsburgh. Internet Suspend/Resume. *Mobile Computing Systems and Applications*, pages 40–46, Jan 2002.

[10] 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*, page 7. IEEE Computer Society, 2004.

[11] B. Lin and P. A. Dinda. VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, page 8. IEEE Computer Society, 2005.

[12] M. Nelson, B. Lim, and G. Hutchins. Fast Transparent Migration for Virtual Machines. *Proceedings of the USENIX Annual Technical Conference 2005*, pages 391–394, Jan 2005.

[13] OpenSSL Project. OpenSSL: The Open Source Toolkit for SSL/TLS. http://www.openssl.org/, December 2008.

[14] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the Migration of Virtual Computers. *In Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, pages 377–390, 2002.

[15] M. Schmidt, N. Fallenbeck, M. Smith, and B. Freisleben. Secure Service-Oriented Grid Computing with Public Virtual Worker Nodes. In *Proceedings of 35th EUROMICRO CONFERENCE on Internet technologies, quality of service and applications (ITQSA)*, page (to appear). IEEE press, 2009.

[16] M. Schmidt, M. Smith, N. Fallenbeck, H. Picht, and B. Freisleben. Building a Demilitarized Zone with Data Encryption for Grid Environments. In *Proceedings of First International Conference on Networks for Grid Applications*, pages 8–16, 2007.

[17] M. Smith, M. Schmidt, N. Fallenbeck, T. Doernemann, C. Schridde, and B. Freisleben. Secure On-demand Grid Computing. In *Journal of Future Generation Computer Systems*, pages 315–325. Elsevier, 2008.

[18] B. Sotomayor, K. Keahey, and I. Foster. Overhead Matters: A Model for Virtual Resource Management. *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, pages 5–5, Nov 2006.

[19] Unionfs Developers. Unionfs: A Stackable Unification File System. http://www.filesystems.org/project-unionfs.html, June 2008.

[20] VMWare Inc. VMWare GSX Server. http://www.vmware.com/products/server/, 2008.

[21] D. Wolinsky, A. Agrawal, P. Boykin, and J. Davis. On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide-area Overlays of Virtual Workstations. *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, page 8, Jan 2006.