Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation

Kaveh Razavi, Liviu Mihai Razorea, and Thilo Kielmann

Dept. of Computer Science, VU University Amsterdam
{k.razavi,l.m.razorea,thilo.kielmann}@vu.nl
www.conpaas.eu

Abstract. Elastic cloud applications rely on fast virtual machine (VM) startup, e.g. when scaling out for handling increased workload. While there have been recent studies into the VM startup time in clouds, the effects of the VM image (VMI) disk size and its contents are little understood. To fill this gap, we present a detailed study of these factors on Amazon EC2. Based on our findings, we developed a novel approach for consolidating size and contents of VMIs. We then evaluated our approach with the ConPaaS VMI, an open-source Platform-as-a-Service runtime. Compared to an unmodified ConPaaS VMI, our approach results in up to four times reduction of the disk size, three times speedup for the VM startup time, and three times reduction of storage cost.

1 Introduction

Traditionally, program binaries have been the abstraction used for executing code on a certain computing resource. In contrast, to execute an application on an Infrastructure-as-a-Service (IaaS) cloud, the user has to provide a virtual machine image (VMI) that contains all the execution dependencies, typically containing a fully-featured operating system plus application-specific software. One can see the VMI as a new abstraction for executing code on computing resources. The implications of using VMIs should be clearly studied and proper methods be deployed as necessary.

One of these implications is the fact that VMIs are large and abundant, existing on behalf of different users. Thus, it is not feasible to store all of the VMIs on every compute node. In clouds today, a number of storage nodes are dedicated to storing VMIs. Upon user request to create a new VM from a specific VMI, the VMI is transfered from the storage node to a designated compute node either fully and in the beginning, or partially and on demand. In both cases, the virtual disk size and the contents of the VMI become important factors in terms of startup time and the consumption of storage and network resources.

While there have been recent studies into the startup of VMs in clouds [8, 12], the effects of the VMI's disk size and contents are hardly understood. To fill this gap, we have studied the impact of these factors on Amazon EC2[1].

[1] Similar experiments on our own private cloud based on OpenNebula showed results consistent with the ones from EC2. We omit them to comply with space limitations.

We use the results of this study to develop a novel approach for consolidating VMIs. We evaluate our approach with the VMI of ConPaaS [15], an open-source Platform-as-a-Service (PaaS) runtime.

The structure of this paper is as follows: Section 2 studies the effects of disk size and contents on the startup time of VMs. We describe how to consolidate the contents of the ConPaaS service VMI, our primary source of motivation for this work, in Section 3. We evaluate our approach in Section 4. Related work is discussed in Section 5, and Section 6 concludes.

2 Analyzing VM startup time in clouds

In this section, we analyze startup delays on Amazon EC2 while varying the VMI disk size and its contents. We describe the VM startup process on Amazon EC2, before we present the experimental results.

2.1 VM startup process on Amazon EC2

Upon a user request to start a VM from a VMI, four steps are executed. **(1)** The EC2 scheduler identifies a suitable physical machine for the VM. Depending on the properties of the requested VM (i.e., instance type), the availability of physical resources, and the number of concurrent requests, this operation can take different amounts of time [8]. As the scheduler delay is beyond the user's control, we can not apply optimizations here. Therefore, we fixed the instance type to *m1.small* for the rest of this paper. However, we do report on the estimated delay of this operation in Section 4.

(2) Next is the transfer of the VMI from the storage node(s) to the designated compute node. Amazon EC2 has two different types of VMIs: instance-store (S3-backed) VMIs, and EBS-backed VMIs. S3-backed VMIs are first transferred from S3 to the compute node before VM startup. With EBS-backed VMIs, the location of the VMI is mounted on the designated compute node, and the VMI is read on-demand by the VM. While EBS-backed VMIs offer slightly faster startup times, each I/O operation from the VM to the VMI will cause a network transfer that costs a small amount of money and a network delay.

(3) In order to save on required storage (costs) and on network transfer from S3 to the EC2 compute node, S3-backed images are stored in a compressed format. This requires a third step, where the S3-backed VMI is decompressed. EBS-backed VMIs are not compressed on storage and do not need this step.

(4) In the final step, the VM starts booting and reads sectors of data from its VMI. The VM booting delay depends on the operating system (e.g. Linux), distribution (e.g. Debian), and on the enabled system services (e.g. a web server). While it is possible to fiddle with the booting process of a specific operating system or distribution, users generally avoid making low-level changes to the operating system stack for reasons of complexity and forward compatibility. There are, however, standardized distribution-specific tools such as [4, 19] that can be used to add or remove services from the booting process.

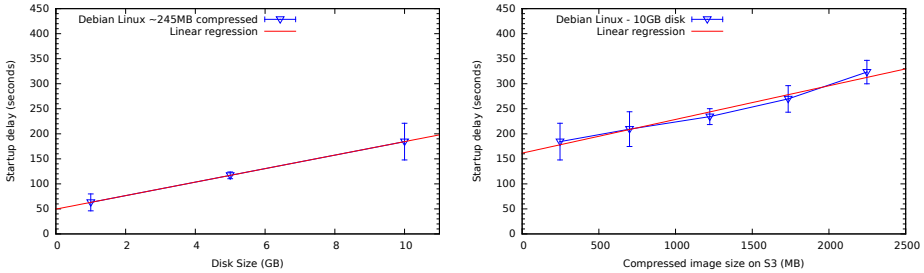

Fig. 1: EC2/S3 startup delay, varying the disk size and contents of the VMI.

2.2 Startup delay while varying disk size and content

We have used *build-debian-cloud* [2] to build Debian images that can boot on Amazon EC2. The tool takes the desired disk size as a parameter (and fills the image with empty blocks). We start from a Debian distribution and add a file to the image with random ("compression proof") content of a given size to change the compressed size of the image. This way, we can independently study the effects of the uncompressed and compressed image sizes.

Figure 1 shows the result of the experiments while varying the disk size (left) and compressed size (right) with S3-backed images. We have repeated each experiment five times. When varying the disk size, the content size remains the same (except for some negligible file system metadata for the bigger disk size). Thus, the reverse of the regression line's slope estimates the decompression rate (75.9 MB/sec). While varying the compressed size, the disk size remains constant (10 GB) and thus, the the reverse of the regression line's slope gives the S3/EC2 network transfer rate (14.9 MB/sec). These results suggest that, for S3-backed VMIs, it is important to:

a) Reduce the disk content to save on the network transfers for a faster startup. The reduced size of the disk contents also saves required S3 storage space.
b) Reduce the disk size to save on the decompression time for a faster startup.

For the EBS-backed VMIs, regardless of the disk size or content, the startup time remains constant at about 51 seconds. Reduced disk content, and subsequently disk size, however, does save on the EBS storage cost. As mentioned before, the benefit of an S3-backed VMI (vs. EBS-backed) comes from the fact that VM-generated I/O requests to the VMI do not cause network transfers, removing network delays and costs.

3 Consolidating the ConPaaS services VM image

Before looking at possible mechanisms to reduce the disk content and size of VMIs, we look at the ConPaaS VMI as a motivating use-case of a typical cloud

service VMI. This VMI is scaled out frequently depending on the user's application and workload. Thus, an effective VMI optimization strategy will have a big impact on the perceived performance and/or user experience. To discover prominent optimization strategies, we investigated the process of VMI generation in ConPaaS.

ConPaaS comes with two scripts to generate VMIs that run on either the Xen or KVM virtual machine monitors (VMMs). First, an empty disk with the default size of 3 GB is created and then formatted. Second, using *debootstrap* [5], a minimal Debian system is installed on the disk. Third, a proper Linux kernel is installed and then, the *grub* bootloader is configured. Finally, the required packages for each service are installed and then configured. For example, the MapReduce service installs the packages of *apache-hadoop*.

Looking at these steps, we can make a number of interesting observations. First, there is a large redundancy in the two provided scripts for different VMMs. This means that changing one will most likely result in changing the other. Second, since the required disk size of the VMI is not known in advance, the ConPaaS developers had to add a default value and increase it whenever the free disk space was not sufficient for a new service that brings in new packages. Third, *debootstrap*, while creating a minimal Debian installation, still brings in certain packages such as *man pages* that are not necessary for a production VMI on clouds. Fourth, all the packages of all the services are present in the VMI. This suggests that there is room for image specialization depending on the users' needs of a certain ConPaaS deployment. Based on these observations, we proceed to our three-step approach for consolidating the contents of the ConPaaS service VMI, consisting of (1) specializing to the services needed, (2) pruning the tree of software packages to the bare minimum, and (3) reducing the disk size accordingly.

3.1 VMI specialization

VMI generation starts from a basic VMI without services, consisting of a minimal Debian installation and the ConPaaS service core. During image specialization, the user specifies which services he wants to add to the image, the type of VMM, as well as the system architecture (e.g., x86/x64). Based on these choices, a customized script is generated automatically, and used to create the VMI.

For ConPaaS, this leads to unifying the two independent scripts (for Xen or KVM) to a single set of scripts as well as the possibility for users to choose various sets of services, depending on their needs.

3.2 Pruning

After creating the image, the next stage uses shell tools to inspect the database of the package management system, in our case apt/dpkg [1, 7]. This step automatically analyzes the dependencies between currently installed software packages and decides which of the packages installed by *debootstrap* are not necessary and can safely be removed. Limiting the analysis to the information provided by the

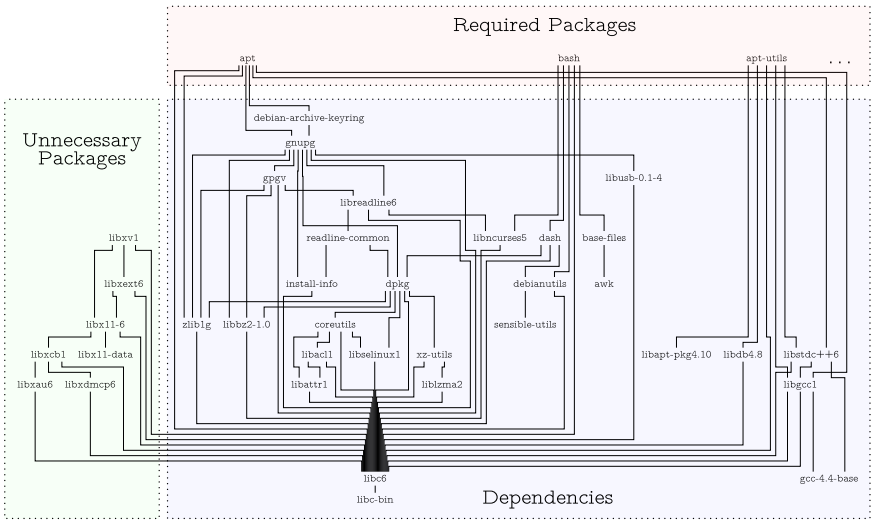

Fig. 2: Example of a partial dependency graph, based on required packages.

package manager reduces the number of opportunities to save disk space, but ensures that the pruning step will not break the package management system.

The pruning algorithm constructs a partial dependency graph by following only dependencies of packages required by the user. Figure 2 shows an example of such a graph. In this diagram, we distinguish packages that are directly *required* by the services the user wants to be hosted on the VMs. Their *dependencies* are determined for inclusion in the VMI. All other packages, that are not part of this hierarchy, are considered *unnecessary*, and will be removed.

3.3 Disk size reduction

The previous phase has reduced the contents of the image, but not the image size. The final step calculates the necessary image size, creates a new bootable disk, and copies the contents of the VM to this disk.

Copying the files to a fresh, smaller VMI disk has two benefits. **(1)** For clouds that do not implement compression, the smaller VMI disk results in faster network transfers. **(2)** For clouds that do implement compression, the new VMI disk results in a better compression and thus, faster network transfers. The latter is because most filesystems only remove the i-nodes (vs. the data) when files (from the unnecessary packages) are removed. This means that the corresponding blocks still contain non-zero data that will be included in the compressed image, taking up space needlessly. Starting from a fresh image, with all empty blocks containing zero data, improves the compression rate.

4 Experiments

We tested our VMI consolidation approach with various configurations. With VMI specialization, users now have the option to determine the services they want included in their ConPaaS VMI. We experimented with a number of proposed configurations, covering a wide range of use cases in the context of ConPaaS, shown in Table 1. In addition, we experimented with the ConPaaS *core* (without any service-specific packages installed) and the *complete* ConPaaS, including all of the developed services so far; the services from Table 1, as well as a *Selenium* web-application testing service, and a content delivery network service (CDN).

Table 1: Interesting VMI configurations for ConPaaS

Config.	List of services	Use-case
C1	XtreemFS	Scalable storage
C2	Hadoop	Scalable MapReduce
C3	PHP, MySQL	Web application
C4	PHP, MySQL, Scalaris	Web application with caching
C5	PHP, MySQL, Scalaris, XtreemFS	Web application with caching & storage
C6	HTCondor, XtreemFS	High-throughput computing & storage

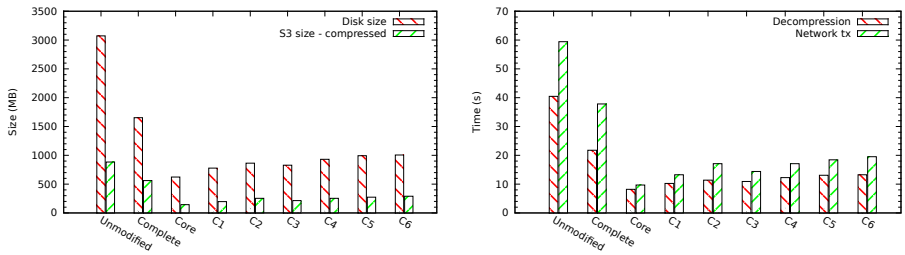

Fig. 3: Left: disk size and contents with service selection and optimizations. Right: the estimated network transfer and disk decompression time on EC2, based on Figure 1.

Figure 3 (left) compares the disk size and S3-compressed size of various configurations with the unmodified ConPaaS VMI. The difference between *unmodified* and *complete* is the result of the optimizations from Section 3.2 (reducing the disk contents) and from Section 3.3 (reducing the disk size). The differences between *unmodified* and the configurations described in Table 1, are the result of VMI specialization from Section 3.1, in addition to the other optimizations.

Using the estimated S3/EC2 network performance and decompression rates obtained in Section 2, it is possible to convert the size values in Figure 3 (left) to

time estimates. Figure 3 (right) shows the estimated delay in VM startup caused by the network transfer and by decompression. We estimate that pruning and disk resizing together result in an about 40 seconds faster VM startup: 46% faster decompression and 36% faster network transfer. VMI specialization saves from 27 seconds (C6) to 36 seconds (C1): 39% to 53% faster decompressions, and 48% to 65% faster network transfers.

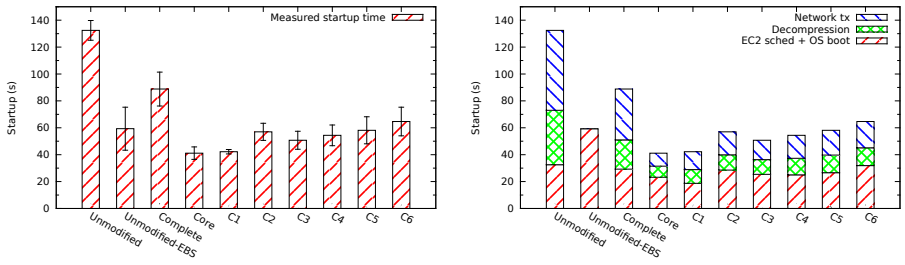

Fig. 4: Left: measured startup time of ConPaaS VMIs with various configurations and optimizations. The unmodified EBS startup time is included as well for comparison. Right: decomposed startup time based on the estimates of Figure 3 (right).

Figure 4 (left) shows the measured startup times of ConPaaS VMIs with various configurations and optimizations. We have repeated each experiment five times. Although variations in startup times are present, they tend to remain comparably small and stable over time as independently shown by [8]. Figure 4 (Right) shows the decomposed startup times into various operations discussed in Section 2.1. Since we have an estimate of network transfer and decompression delay from Figure 3 (right), we can assume the rest of the startup time involves the EC2 resource scheduler and the OS boot time.

In total, depending on service selection, our optimizations result in two to three times faster VM startups for S3-backed VMIs. Figure 4 also includes the startup time of an unmodified, EBS-backed ConPaaS VMI. As expected, due to the on-demand nature of EBS, the startup delay of unmodified ConPaaS (59 seconds), remains close to that of a minimal Debian (53 seconds). While our optimizations do not affect the startup time of EBS-backed AMIs, they do save up to three times in storage cost. Another interesting observation is that, the S3-backed configurations C1-C5 start slightly faster than the EBS image.

In this section, we described the results of our experiments with the proposed approach in generating VMIs. The optimizations mandated by our approach resulted in up to three times faster startup times and up to three times saving on the costs of storing ConPaaS VMI.

5 Related work

In this section, we discuss related work in three areas: *VM consolidation* generates special-purpose VMIs. *Efficient VMI transfer* deals with the problem of moving VMIs from storage nodes to compute nodes, and *Cloud instance startup time* studies the startup time of VM instances in different clouds.

5.1 VM consolidation

VMPlant [10] is a service for generating custom VMIs. The user provides VMPlant with 1) machine requirements (e.g. OS, size of memory and disk, etc.), and 2) a set of configuration scripts and their dependencies in the form of a directed acyclic graph (DAG). VMPlant then accordingly generates a new VMI based on previously cached VMIs. In contrast, we generate VMIs from scratch; our optimizations, however, could complement VMPlant's current functionality.

Jebessa et al. [9] argue for purpose-driven VMs to enhance security by means of VMI code reduction. They use a domain-specific language to declare the properties of the desired target VMI such as the required software package set and the target VMM. We use a similar approach to reduce the content of the VMI by looking at the dependency tree of user-selected services. Further, we reduce the necessary disk size of the VMI in the final generation process.

Quinton et al. [16] use package dependency information to install the minimum number of packages for a given virtual appliance. To minimize the disk size, they estimate the installation size of each package. Our final VMI copying step renders such estimations unnecessary.

Charon [6] is a tool that builds on top of NixOS [13]. NixOS has a declarative package management system and Charon uses that to configure virtual machines with the right package set for the provisioned services.

5.2 Efficient VMI transfer

In our work, we focus on minimizing the size of the VMIs themselves. Complementary, a large body of work is dealing with speeding up the deployment of given VMIs. There are two different scenarios in which efficient VMI transfer becomes a primary concern. In the first scenario, a single VMI needs to be transferred to many physical nodes. Here, peer-to-peer networking is a commonly used technique [3, 17, 20]. SnowFlock [11] clones VMs from already running ones in less than one second.

In the second scenario, many VMIs need to be transferred to many physical nodes concurrently and start executing on behalf of different users. Schmidt et al. use Unionfs [18], a sophisticated, layered file system along with multicasting to minimize network transfer times of VMIs. VDN [14] is a network hierarchy-aware system for transferring VMIs to compute nodes in units of small chunks that can be retrieved from local neighbors.

All these approaches can be used orthogonally to our VMI content consolidation work. The advantage of our approach is that it does not require any

changes to the cloud infrastructure and can hence also be used with commercial IaaS clouds like EC2.

5.3 Cloud instance startup time

Mao et al. [12] study the startup performance of various commercial IaaS clouds by various factors such as OS image size, instance-type and number of instances acquired at the same time. Their experiments only consider the image size, neglecting the effects of the actual disk content. In contrast, we have used a set of experiments that separates the performance implications of disk size and content. Mixing of compression time into network transfer time is apparent in their reported transfer rate of 10.9 MB/s, which is lower than our estimated value of 14.9 MB/s.

Iosup et al. [8] benchmark various aspects of cloud computing. They extensively study resource acquisition delay on Amazon EC2 while comparing different factors such as different instance-types and number of concurrent requests. In this work, we have done a break-down of acquisition delay of Amazon EC2 to find promising areas for optimizing the resource acquisition performance.

6 Conclusions

Fast VM startup times are essential for dynamic scaling of cloud-based services. Both disk size and contents of the user-provided VMIs play an important role in the VM startup time. Furthermore, VMI disk sizes directly translate to monetary costs for storage and network transfer of the VMIs that users have to pay to their cloud providers.

In this paper, we studied the VM startup time as a function of VMI disk size and content. Based on these results, we introduced a novel approach for reducing the disk size and content of VMIs. Using the example of the ConPaaS service VMI, we let users select the desired set of services, based on which the strictly necessary set of software packages is determined and installed on the VMI. A final pass creates a VM disk of the minimal required size. This approach, in general, can be applied to all VM images for which the user can express which top-level software packages are required for its desired functionality.

Our experiments show up to four times reduction in the disk size, and up to three times speedup in the VM startup time for S3-backed VMIs on Amazon EC2. Furthermore, EBS-backed VMIs created with our approach require up to three times less storage costs, compared to an unoptimized VM image.

Acknowledgments

This work is partially funded by the FP7 Programme of the European Commission in the context of the Contrail project under Grant Agreement FP7-ICT-257438, and by the Dutch public-private research community COMMIT/.

References

1. apt-rdepends. http://man.devl.cz/man/8/apt-rdepends, [Online; 4-June-2013]
2. Debian bootstrapping script for Amazon machine images and Google Compute Engine images. https://github.com/andsens/build-debian-cloud, [Online; 4-June-2013]
3. Chen, Z., Zhao, Y., Miao, X., Chen, Y., Wang, Q.: Rapid Provisioning of Cloud Infrastructure Leveraging Peer-to-Peer Networks. In: 29th IEEE Int. Conf. on Distributed Computing Systems Workshops. pp. 324–329. ICDCSW '09 (2009)
4. chkconfig. http://linux.die.net/man/8/chkconfig, [Online; 4-June-2013]
5. debootstrap. http://linux.die.net/man/8/debootstrap, [Online; 6-June-2013]
6. Dolstra, E., Vermaas, R., Levy, S.: Charon: Declarative Provisioning and Deployment. In: 1st International Workshop on Release Engineering 2013. pp. 17–20. RELENG '13 (2013)
7. dpkg-query. http://man.he.net/man1/dpkg-query, [Online; 4-June-2013]
8. Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. IEEE Trans. Parallel Distrib. Syst. 22(6), 931–945 (2011)
9. Jebessa, N.D., van 't Noordende, G., de Laat, C.: Towards Purpose-Driven Virtual Machines. In: ESSoS Doctoral Symposium 2013 (2013)
10. Krsul, I., Ganguly, A., Zhang, J., Fortes, J.A.B., Figueiredo, R.J.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In: 2004 ACM/IEEE Conference on Supercomputing. SC '04 (2004)
11. Lagar-Cavilla, H.A., Whitney, J.A., Scannell, A.M., Patchin, P., Rumble, S.M., de Lara, E., Brudno, M., Satyanarayanan, M.: SnowFlock: rapid virtual machine cloning for cloud computing. In: 4th ACM European conference on Computer systems. pp. 1–12. EuroSys '09 (2009)
12. Mao, M., Humphrey, M.: A Performance Study on the VM Startup Time in the Cloud. In: 5th Int. Conf. on Cloud Computing. pp. 423–430. CLOUD '12 (2012)
13. NixOS. http://nixos.org/nixos, [Online; 29-May-2013]
14. Peng, C., Kim, M., Zhang, Z., Lei, H.: VDN: Virtual machine image distribution network for cloud data centers. In: The 29th Conference on Computer Communications. pp. 181–189. INFOCOM '10 (2012)
15. Pierre, G., Stratan, C.: ConPaaS: a Platform for Hosting Elastic Cloud Applications. IEEE Internet Computing 16(5), 88–92 (2012)
16. Quinton, C., Rouvoy, R., Duchien, L.: Leveraging feature models to configure virtual appliances. In: 2nd International Workshop on Cloud Computing Platforms. pp. 2:1–2:6. CloudCP '12 (2012)
17. Reich, J., Laadan, O., Brosh, E., Sherman, A., Misra, V., Nieh, J., Rubenstein, D.: VMTorrent: scalable P2P virtual machine streaming. In: 8th Int. Conf. on Emerging networking experiments and technologies. pp. 289–300. CoNEXT '12 (2012)
18. Schmidt, M., Fallenbeck, N., Smith, M., Freisleben, B.: Efficient Distribution of Virtual Machines for Cloud Computing. In: 18th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing (PDP). pp. 567–574. PDP '10 (2010)
19. update-rc.d. http://www.tin.org/bin/man.cgi?section=8&topic=update-rc.d, [Online; 4-June-2013]
20. Wartel, R., Cass, T., Moreira, B., Roche, E., Guijarro, M., Goasguen, S., Schwickerath, U.: Image Distribution Mechanisms in Large Scale Cloud Providers. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. pp. 112–117. CloudCom '10 (2010)