Operational Experiences with Disk Imaging in a Multi-Tenant Datacenter

Kevin Atkinson, Gary Wong, and Robert Ricci, ***University of Utah***

https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/atkinson

This paper is included in the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14).

April 2–4, 2014 • Seattle, WA, USA

ISBN 978-1-931971-09-6

Open access to the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14) is sponsored by USENIX

Operational Experiences with Disk Imaging in a Multi-Tenant Datacenter

USENIX Symposium on Networked Systems Design and Implementation—Operational Systems Track

Kevin Atkinson * Gary Wong Robert Ricci

University of Utah School of Computing
{kevina, gtw, ricci}@cs.utah.edu www.emulab.net

Abstract

Disk images play a critical role in multi-tenant datacenters. In this paper, the first study of its kind, we analyze operational data from the disk imaging system that forms part of the infrastructure of the Emulab facility. This dataset spans four years and more than a quarter-million disk image loads requested by Emulab's users. From our analysis, we draw observations about the nature of the images themselves (for example: how similar are they to each other?) and about usage patterns (what is the statistical distribution of image popularity?). Many of these observations have implications for the design and operation of disk imaging systems, including how images are stored, how caching is employed, the effectiveness of pre-loading, and strategies for network distribution.

1 Introduction

Computers in datacenters are frequently re-allocated from one purpose to another, need to have their software upgraded, or need to be returned to a known "clean" state. This type of re-provisioning is particularly important in *multi-tenant* datacenters [4], which are shared by a large number of applications running on behalf of different clients. Notably, this is the model adopted by "Infrastructure as a Service" (IaaS) clouds such as Amazon EC2 [2], Rackspace [11], and datacenters managed with software such as OpenStack [15]. These facilities provide physical or virtual servers (infrastructure) on which users run their own operating systems and applications [9, 15].

The primary means for initializing user resources is to load them with an initial *disk image*, which is a block-level snapshot of a filesystem containing an installed operating system and set of applications. Typically, a cloud will provide a set of images that any user may install on servers that they provision (*facility images*). Users may also create their own images (*user images*): this is commonly accomplished by loading a facility image, customizing it, and taking a snapshot of the resulting disk.

*Work done at the University of Utah; now at Rice University

Large multi-tenant facilities have hundreds to hundreds of thousands of servers and thousands to millions of users [5]. A busy facility may have many thousands of user images and provision tens of thousands of servers per day. Disk images are commonly written to drives attached to the host; EC2, for example, calls this "instance storage" [1], and it is available on nearly all VM types. Disk imaging is on the critical path for provisioning servers, which cannot be booted until the requested image has been loaded. Images can consume significant resources on the facility, including the space used to store them and the network bandwidth required to distribute them to the hosts on which they are to be used. Thus, understanding disk images and their use is important to the design and operation of multi-tenant datacenters.

In this paper, we study four years' worth of data from the operation of the Emulab testbed [16], a multi-tenant facility with approximately six hundred hosts and over five thousand user accounts. The data we examine covers 279,972 requests for disk images (Section 2) and is, to our knowledge, the only dataset currently available to the public that contains detailed traces of disk imaging in a multi-tenant datacenter. It allows us to study properties of the disk images themselves as well as how they are used by the facility's users, and we draw a number of conclusions that are applicable to the design and operation of imaging systems. Our key findings include:

- **Section 3**: There is substantial block-level similarity between many images, suggesting that deduplicating storage is appropriate. The lifespan of images varies greatly, from days to years, and many images go unused for months at a time, making multi-tier data storage attractive.

- **Section 4**: The working set of images is quite small (mean: 12 per day, 30 per week), making caching of frequently used images potentially effective. However, the makeup of this working set changes frequently, and there are no dominant images. The daily working set size grows linearly with the number of users, but the total number of facility and user

images follow different curves.

- **Section 5**: The popularity of user images follows a heavy-tailed distribution, while the popularity of facility images does not. Most users skew heavily towards using either facility provided images or custom images, not both. While most users do not create their own images, those who do number among the facility's heaviest users.

- **Section 6**: We consider the technique of pre-loading popular facility images, allowing some requests to be satisfied without waiting for the image to load. We find that two factors control the potential benefit from this strategy: (a) the ratio of the working set size to the number of idle disks available for pre-loading, and (b) the ratio of the rate at which the facility can load disks to the arrival rate of requests.

- **Section 7**: Differential loading (pre-loading a base image, then transferring only differing blocks as required) shows potential. In order to be effective, it will require development of sophisticated prediction techniques that take into account both the popularity of images themselves and their block-level similarity to each other.

We conclude in Section 8 with several concrete suggestions regarding the design and operation of disk imaging systems, and point to fertile areas for future work.

2 Dataset

Emulab is a network testbed widely used by the distributed systems and networking communities. An experimenter describes a network in terms of links and hosts. Included in this specification is the disk image to be loaded on each host. Emulab then provisions servers, physical or virtual, loading the requested disk image. This provisioning is done on demand as requests come in, and there is only limited support for ahead-of-time scheduling or batch jobs. The facility provides a number of standard images, including "default" images that are used if the user does not explicitly request an image. Many users create their own images by booting from a facility image, customizing it (for example, by installing software packages or modifying the operating system), and taking a snapshot. This user image can be referenced in future requests, saving the user the effort of re-installing the packages they use, or to scale out to much larger experiments. This basic model of image usage and creation is similar to that used in most IaaS clouds [14].

Emulab is capable of provisioning both physical and virtual machines; physical machines are the most commonly allocated resource. While many IaaS clouds provision solely virtual machines, we believe that this difference does not have a significant impact on conclusions drawn from the dataset: in either case, the user is presented with the abstraction of an PC on which they may load and boot an operating system. While the details of operating systems that run within physical and virtual machines may vary, the quantity and diversity of users' desired images is unlikely to be affected.

Emulab uses block-level disk images and distributes them using the Frisbee [6] disk imaging system. The format uses filesystem-aware compression, meaning that it does not store disk blocks that are not used by the filesytem, and compresses the allocated blocks with `zlib` [7] for efficient storage. Frisbee uses IP multicast to distribute images, and is highly optimized so that the bottleneck in image distribution and installation is the write speed of the target disk. The amount of time required to load a disk image depends on the number of used blocks in the filesystem that it contains, but is typically on the order of a few minutes. Facility images are visible to and may be requested by all users. User images are visible only to their creators unless the creator decides to make the image public, which few do.

2.1 Dataset Details

The dataset that we study covers four years of disk image requests on Emulab, from March 2009 to March 2013. The dataset covers a total of 279,972 requests for 714 unique images. The requests were made by 368 users, at an average rate of 192 disk images loaded per day. The records cover the identity of the image, the user making the request, and the timestamp at which the request was made. Furthermore, the data indicates whether each image was a facility image or a user image, and whether it was requested explicitly by the user or was chosen as a default because the user did not specify an image. To preserve user anonymity, users and user images are assigned random integers as identifiers in this paper. We present the names of facility images using their Emulab-assigned names; user images are presented as $user/image$ pairs.

One of the things we studied was the block-level differences between images. Our primary interest in examining the contents of images is to determine the potential savings from loading a "base" image (usually a facility image), then transferring and writing only the disk blocks required to transform it into a particular "derived" image (usually a user image). We define the difference of two images A and B as:

$$\Delta(A, B) = |\forall i \in b : B[i] \neq A[i]| \quad (1)$$

where b is the set the indices of allocated storage blocks in image B, and $A[i]$ and $B[i]$ are the contents of images A and B, respectively, at index i. This measure directly

captures the numbers of blocks in image B that would need to be written to a disk that already contains image A. We define $\delta(A, B)$ as the fraction of blocks that would need to be written: that is,

$$\delta(A, B) = \frac{\Delta(A, B)}{|b|} \quad (2)$$

The Emulab dataset does not record the provenance of images (that is, which user images where based on which facility images). We assume that each user image U was based on the facility image F for which $\Delta(F, U)$ is minimized. For a particular image U, we refer to this base image as U_B. Emulab allow users to delete their image files: only 37.4% (267) of the images found in the request traces were available for analysis of block-level similarities. Though large in number, the missing images were relatively unpopular, accounting for only 15.8% of all requests. Emulab also allows its users to modify images, so the image files that we analyzed represented a snapshot of image contents at a particular point in time.

2.2 Removing Sources of Bias in the Dataset

We filtered the dataset to remove certain biases. First, we omit all uses of the facility by its operational staff: the maintenance, testing, etc. that they perform is likely to follow different patterns than users of the facility. Second, as a network testbed, Emulab supports a feature known as "delay nodes," [13, 12] which perform a traffic-shaping role that does not represent a function present in most multi-tenant datacenters. Third, Emulab includes some resources that are not the standard PC servers used in clouds and datacenters: these include wireless nodes, programmable network hardware, and sensors. This filtering removed 183,824 of the original 463,796 requests (39.6%), 215 images (23.1%), and 30 users (7.5%), leaving us with the 279,972 requests, 714 images, and 368 users that we studied.

It is worth making special note of Emulab's "default" images. If an Emulab experimenter does not specify a particular disk image in their experiment description, they get a default that is, for historical reasons, quite old. Due to their ages, the default images are not very popular. Most users select the facility image that best meets their needs; as a result, the presence of a default does not have a dominating effect on the way that users select images.

2.3 Users and Projects

For the purposes of this study, we consider users at the level of *organizations*. Emulab groups individual users into "projects." These loosely-defined groups represent research groups, classes, or cross-institution collaborations. Because of this, they are analogous to businesses that purchase time on a cloud such as EC2, or individual business units that share a company-wide datacenter. In the remainder of this paper, we consider all individuals who are part of the same project to be a singe "user" of the facility—when we refer to "users," we are referring to Emulab projects. The number of individuals who requested disk image loads over this time period was 1,301.

2.4 Limitations of This Study

The Emulab dataset is, to our knowledge, the only one of its type currently publicly available. Therefore, we cannot quantitatively assess the degree to which it matches other multi-tenant facilities. We believe our analysis remains valuable nonetheless, for two reasons. First, it is the only analysis to date to apply such a large quantity of real-world data to the problem of improving disk imaging systems. Second, we conjecture that the most *fundamental* findings in our work remain applicable in other environments, even if specifics (such as the λ parameter to the facility image popularity distribution) differ.

Our dataset covers a large number of disk image loads, but comes from a mid-sized facility. We attempt to analyze the effects of facility size in Section 4.3, but application of our conclusions to larger facilities necessarily involves extrapolation. In addition, two features unique to Emulab affected our ability to run certain analyses.

First, the nature of resource allocation in Emulab makes it difficult to study the inter-arrival times of image requests. Emulab's primary unit of resource allocation is the *experiment*: a collections of hosts that together make up a network experiment. In contrast, most IaaS clouds consider only individual servers or "instances," and the cloud has no semantic information about which instances are contributing to the same application. Thus, image requests in Emulab arrive in well-defined bursts that do not have a direct analog in many other datacenters. Deploying an application in a datacenter or cloud does often involve provisioning of multiple machines in a short timeframe; however, we have no data that would allow us to analyze whether experiment sizes in Emulab are representative of burst sizes in other environments. For this reason, we avoid analyzing this aspect of the dataset, and all of our analyses are with respect to individual loads of disk images rather than Emulab experiments.

Second, we chose not to analyze the relative popularity of the operating systems contained in the images (eg. Linux vs. BSD, or the relative popularities of Linux distributions). Emulab's user base is overwhelmingly comprised of academic researchers and students, and their OS preferences may not be representative of a broader population. In particular, while Emulab supports Windows, it constitutes a small fraction of all Emulab use—almost certainly a smaller fraction than would be seen in other

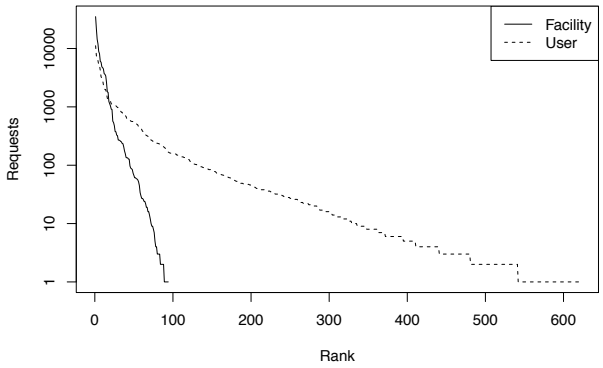

Figure 1: Requests for facility and user images, sorted on the x axis by popularity. Note the log-scale y axis.

settings. We restrict our analysis to the popularity of disk images rather than the operating systems they contain, and it is possible that this distribution is affected by the operating system preferences of Emulab's user base.

3 Storage of Disk Images

We begin our study by examining the basic properties of the images in our dataset, with an eye towards understanding how they should be stored. We pay special attention to the relationship between images that are provided by the facility and images that are created by users; as we will see through further analysis, these images have different characteristics that warrant different treatment.

3.1 Prevalence of User Images

During the 48 months covered by our dataset, there were a total of 368 users. Of these, nearly two thirds (231) used only facility images, and slightly over one third (137, or 37.2%) used at least one user image. This implies that optimizing the provisioning of facility images can improve the experience of a majority of users. For example, if a suitable set of facility images can be identified for preloading on to servers, this could take image loading out of the critical path for creation of those users' instances. We explore this issue further in Section 6.

The number of users who request user images, however, is non-negligible, suggesting that an imaging system should also take their needs into account. In fact, we find that there are more user images in our dataset (619) than facility images (94), meaning that, on average, each user who creates at least one disk image creates 4.5 of them.

3.2 Popularity of User Imags

The top of Table 1 shows the relative popularity of facility and user images. We see that the percentage of requests for user images is over 44%; since only 37.2% of users create their own images, this imples that this set of users are heavier users of the testbed by at least 18%. Table 1 also shows all images that made up at least 1% of the requests. Of these twenty four images, ten are user images. Note that RHL90-STD and FEDORA10-STD each appear twice, because they are both common explicitly requested images and also images loaded by default. The complete image popularity data is plotted in Figure 1. We can see that the number of user images is much larger than the number of facility images, but that the population of user images contains many images that are used few times. Together, the top 17 facility images are more popular than the top 17 user images (the 17th facility image had 1,772 requests, and the 17th user 1,330). From the 18th image onwards, the user images are more popular—the 18th user image had 1,260 requests and the 18th facility image had

	Image name	Requests	%
	All facility images	155,617	55.6%
u	All user images	124,355	44.4%
	RHL90-STD [D]	21,993	7.9%
	FEDORA10-STD	18,042	6.4%
	UBUNTU10-STD	14,402	5.1%
	RHL90-STD	13,182	4.7%
	FC4-UPDATE	12,097	4.3%
u	715/10	11,156	4.0%
	FBSD410-STD	8,916	3.2%
	FEDORA8-STD	8,153	2.9%
u	237/69	7,512	2.7%
u	296/35	7,179	2.6%
u	787/24	6,243	2.2%
	UBUNTU70-STD	6,021	2.2%
	UBUNTU12-64-STD	5,834	2.1%
u	787/14	5,231	1.9%
u	226/44	5,198	1.9%
	FEDORA10-UPDATE	4,861	1.7%
	CENTOS55-64-STD	4,710	1.7%
	FC6-STD	4,455	1.6%
u	762/69	4,213	1.5%
	FC4-WIRELESS	3,700	1.3%
	FC4-STD	3,615	1.3%
	FEDORA10-STD [D]	3,604	1.3%
	UBUNTU11-64-STD	3,383	1.2%
u	624/89	3,277	1.2%
u	238/50	3,113	1.1%
u	226/51	2,899	1.0%

Table 1: Total requests for all user and facility images. Also shown are the number of requests for all images that account for more than 1% of all requests. User images are marked with a 'u' in the left column, and images requested implicitly as defaults are marked with a '[D]'; explicit requests for default images are counted separately.

1,233. Both facility and user images have tails consisting of images that were requested fewer than ten times, but this tail is much more prevalent in the case of user images, where the tail represents nearly half of all user images.

From this data, we can conclude that facility images dominate, but that there are a small number of user images that are as popular as some facility images.

3.3 Image Lifespan

The Emulab dataset does not include explicit creation and deletion dates for images. Thus, we define the lifespan of an image to be the number of days between when the image was first seen in the request stream and when it was last seen. Note that this will tend to underestimate lifespan: some images were likely first used before our dataset begins, and some will continue to be used after the end of the dataset.

A histogram of user image lifespans can be seen in Figure 2. While the majority of images have very short lifespans, there is a long tail: several were used throughout the entire four years covered by the dataset. The observed mean lifespan is 100.4 days.

We found the number of images with short lifespans to be quite surprising, so we examined them in greater detail, and it became clear that a large majority of these short-lifespan images were requested only on a single day: 196 of the 619 user images (31%) fall into this usage pattern. This suggests that a number of users create images for the purposes of running a single experiment, a conclusion borne out by looking at the experiment metadata.

Finally, we looked at how long user images "go idle". We found that it is common for user images to have gaps of months in between requests for them. During this time, there is no need to have the images constantly available; they could be moved to cheaper, but slower, storage. The distribution of the maximum idle periods for the 214 user images with a lifespan of at least 30 days is shown in Figure 3. In total, 162 of the images (76% of long-lived images, and 26% of images overall) had gaps in usage of one month or more. Two images even had gaps of over two years between successive uses.

3.4 Block-Level Differences Between Images

We next examined how much user images differ from the facility images they are based on. We use the definitions of $\Delta(A, B)$, $\delta(A, B)$, and "base" images given in Section 2.1. Figure 4 shows a histogram of similarities between user images and their associated base facility images. From this figure, it is clear that many user images do show significant similarities to their bases—most are more than 50% similar, with a significant peak in the 60%–80% range. This is in line with findings from smaller studies in the past [8]. There is also a significant tail of more than twenty images images with very low similarity (below 10%) to their base images.

Overall, these numbers point to two potential strategies for improving disk imaging systems. First, they suggest that significant storage savings can be had by storing images in a deduplicating storage system [10], which would

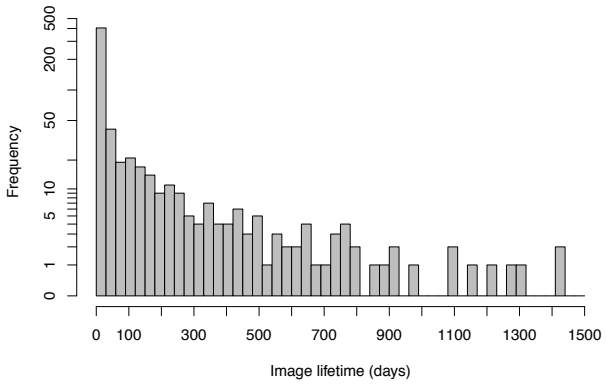

Figure 2: Histogram of the lifespans of user images. Note that the y axis is log-scale.

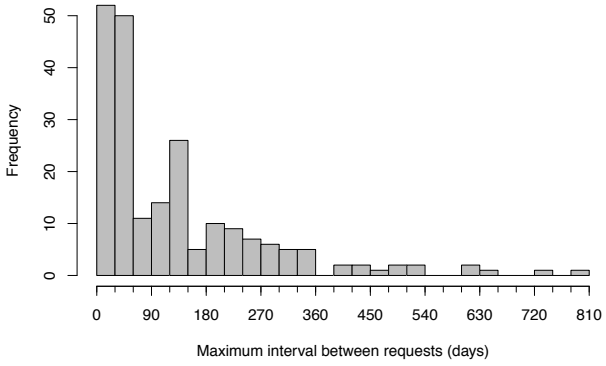

Figure 3: Histogram of usage gaps for user images.

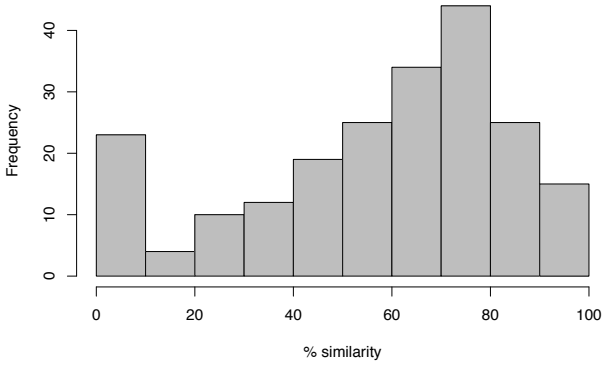

Figure 4: Histogram of similarity ($1 - \delta(U_B, U)$) between user images and their associated base images. Higher percentages indicate more similarity.

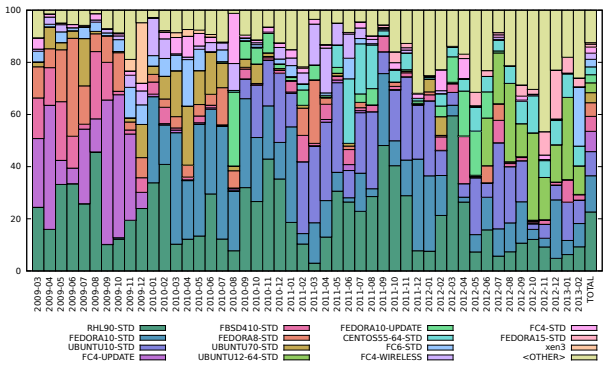

Figure 5: Variation of facility image popularity over time. The fifteen most popular facility images are shown.

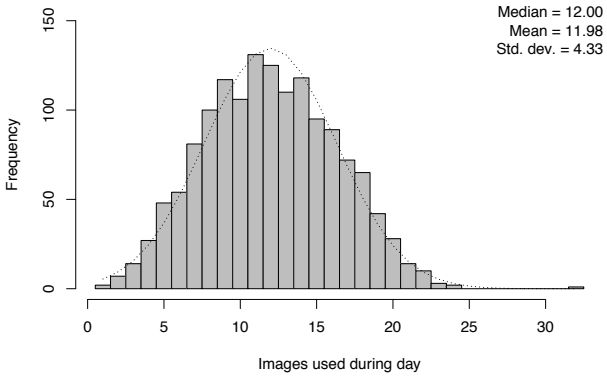

Figure 6: Histogram showing the distribution of the working set size over one-day periods (midnight to midnight).

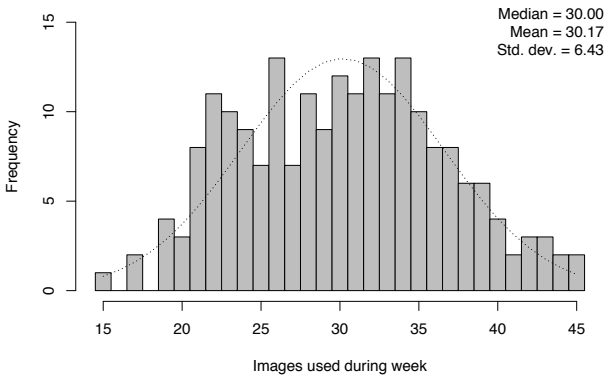

Figure 7: Histogram showing the distribution of the working set size over one week periods (Sunday to Saturday).

store only one copy of the blocks that the base and derived images have in common. Second, they suggest that the technique of differential disk loading, which transforms a base image into a derived image by writing only the blocks that differ, has a potential for reducing the time and bandwith for distributing the user images. We explore the latter in detail in Section 7.

4 Working Set Size and Caching Potential

Having looked at the images themselves, we turn our attention to trends of usage over time, paying particular attention to the working set; understanding the size and composition of the working set is critical to designing strategies for caching and pre-loading.

4.1 No Dominant Images

If a small set of facility images dominates the request stream, it would be possible to design the disk imaging system around that fact. In particular, it would make sense to pre-load most or all idle disks with popular images, allowing user requests to be satisfied without waiting for a disk to load. This is the policy adopted by Emulab: the images labeled '[D]' in Table 1 are loaded as part of the process of freeing machines for the next user.

As we can see in Figure 5, there is no such dominant image. The popularity of all facility images fluctuates wildly from month to month, with new images becoming popular quickly, old images falling out of favor, and some images swinging between popular and unpopular. Even the default images, which remain active throughout the entire time period, sees large changes in popularity. Note that we do not distinguish between explicit and implicit requests for default images as we did in Table 1; for the purposes of disk loading, these two cases are equivalent.

As a result, we conclude that the strategy of pre-loading a single default image is unhelpful. It is, in fact, counterproductive: servers must be taken out of circulation while they are loaded with the default image, and most are re-loaded a second time when requested by a user. If pre-loading strategies are to be useful, they will require more sophisticated methods for predicting future requests.

4.2 Size and Variation of the Working Set

Figure 6 depicts the working set size (number of unique images requested) over one-day periods. The mean working set size is quite small, at a mean of 11.98 images per day—this represents only 1.7% of the total number of images. While there is some variation in the working set size, it is not large: it follows a normal distribution with a standard deviation of 4.33. This result is encouraging from the perspective of caching: it suggests that only a small fraction of images need to be available for quick loading at any point in time, and that others could be stored in cheaper, slower storage systems. Figure 7 shows the distribution over week-long periods. The average working set size is approximately two and a half times larger than the daily average, and again follows a normal distribution with a reasonably small standard deviation.

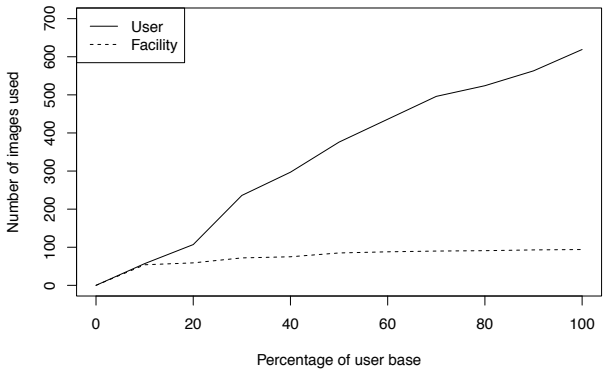

Figure 8: Total number of images used over four years when considering random subsamples of the Emulab userbase.

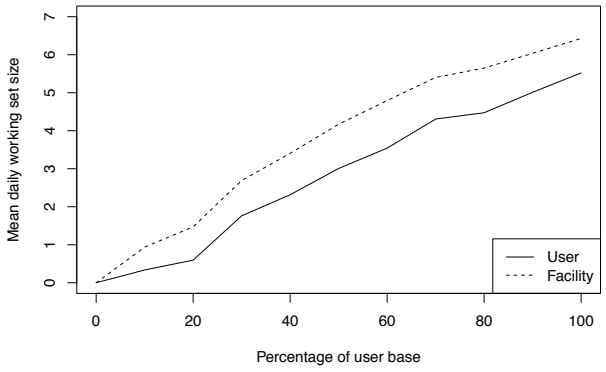

Figure 9: Mean daily working set size when considering random subsamples of the Emulab userbase.

4.3 Scaling of the Working Set

To get a feel for how the size of the working set might vary on facilities larger or smaller than Emulab, we subsampled our data to simulate differently sized userbases. Figure 8 shows the total number of images used over the 4-year period when considering only 10% of the userbase, 20%, etc. The set of facility images quickly reaches saturation (all images are used at least once) and stops growing with additional users. The set of user images, on the other hand, grows linearly with respect to the number of users. This is explained by simple intuition: the set of useful facility images is more a function of the facility than of the userbase, while more users mean more user-created images. Thus, we can expect that a facility with many more users than Emulab will have a greater number of user images in proportion roughly to its greater userbase, but that its set of facility images will not be larger by the same proportion. Indeed, Amazon EC2, which has a userbase that is at least three orders of magnitude larger than Emulab, advertises less than thirty images provided directly by AWS [3] and less than a hundred public images provided by their business partners. In comparison, Emulab has 94 public facility-provided images.

However, this does not quite tell the whole story. Figure 9 shows the same subsampings, but this time looks at the mean daily working set size. Here, we see that the number of images loaded in a typical day increases linearly with the userbase for both facility and user images. Thus, we can expect that facilities much larger than Emulab do exhibit larger working sets. The working set of facility images is capped by the total number of such images, so very large facilities are likely to include most or all of their facility images in the daily working set.

The general trend we can expect, is that for small facilities, the daily image working set size is in direct proportion to the size of the userbase. For large facilities, the working set will contain a relatively small set of facility images, and a very large set of user images; however, we find that the fraction of requests that are for user images stays fairly constant regardless of the size of the userbase, meaning that these requests must necessarily be diverse.

5 Users' Behavior

We now turn our attention to the behavior of individual users; a facility that understands how its users interact with images is in a better position to provide the interfaces and image management tools that they require.

5.1 Distribution of Image Popularity

In distributions with "light" tails, such as the normal distribution, a relatively small subset of the population accounts for most of the popularity. For "heavy tailed" distributions (defined as those whose tail is not bounded by the exponential [17]), this effect is less pronounced, and it takes more of the population to cover the same level of popularity. We compared the popularity distributions of facility and user images separately to exponential distributions chosen to match the sample means. We found that facility images are a reasonably good match for the corresponding exponential distribution (with Kolmogorov-Smirnov statistic $\sqrt{n}D_n = 1.13$), but user images are not ($\sqrt{n}D_n = 5.54$). As can be seen in Figure 10, the tail for user images lies substantially above the exponential.

This is a key finding: user-created images have a significant heavy tail, while facility-provided images do not. The primary consequence of this discrepancy is that strategies that depend on being able satisfy a large number of requests with a relatively small number of images (such as pre-loading, examined in detail in Section 6), will be more effective with facility images than with user images.

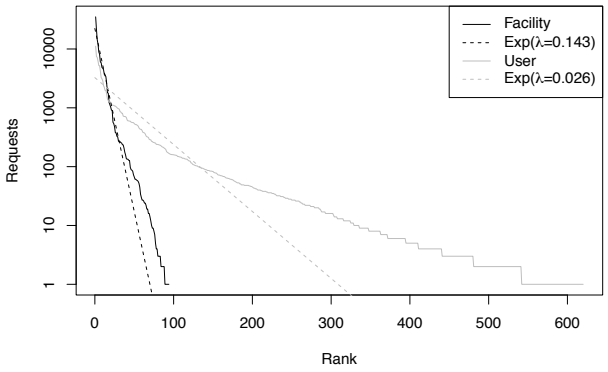

Figure 10: Distribution of image popularity compared to the exponential (shown as dashed lines); note the log-scale y axis.

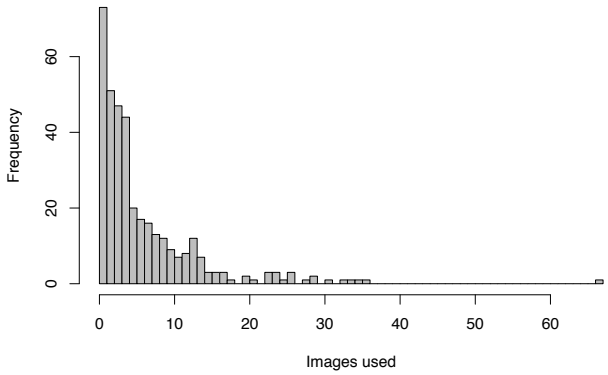

Figure 11: Histogram showing the number of users who use different quantities of images.

5.2 Users and Images

As we can see in Figure 11, most users use a relatively small set of images. There are, however, two surprising features of this data.

Only 20% of users used a single image—a large majority used two or more. We believe that this is due to three factors. First, since our sample period covers four years, many users likely migrated to newer versions of images as operating systems were updated. Second, any user who creates a custom image will use at least two images: they will request the base facility image at least once, then move to the custom image they create. Third, users may have started off using the default images provided by Emulab, found them unsuitable for their needs, and switched to non-default images.

Another surprising feature is that there are a small number of users who use a very large number of images. Twenty users use at least 20 images, and one outlier uses more than 60.

Figure 12: Profile of users making at least 500 disk image requests. Requests for facility images are shown as bars above the axis, and user images are below the axis.

5.3 Behavior of Heavy Users

Because user images are created by customizing facility images, we can expect that all users will employ facility images at least once, and likely a few times. The question remains, however, whether users tend to use primarily facility images, primarily their own images, or some balanced mixture of the two. We are particularly interested in the answer to this question for heavy users of the facility.

Figure 12 shows a profile of the heaviest users (those who made at least 500 image requests) from the Emulab dataset. Two important facts are evident. First, while a few users do mix facility and user images (i.e. have bars both above and below the axis in the figure), most tend to skew heavily towards one or the other. Second, among the twenty heaviest users, twelve employ primarily user images. Past this point, facility images dominate. This clearly establishes that custom user images are a "power user" feature: their dominant use is by a relatively small number of users, who use them heavily.

6 Prediction and Pre-Loading

We now turn our attention to techniques that may allow the facility to service user requests more quickly. The first technique that we consider is pre-loading: if it is possible to predict which images will be requested in the near future, the facility can pre-load them onto idle disks. If the predictions are correct, users requests may be satisfied immediately; if not, the user will have to wait for their image to be loaded. Note that this strategy does not save bandwidth on the datacenter's image distribution network; it simply shifts the image distribution to before the user's request arrives. In fact, pre-loading may *increase* the bandwidth used for distributing images: in the case of mispredictions, a node pre-loaded with one disk image may need to be re-loaded with another.

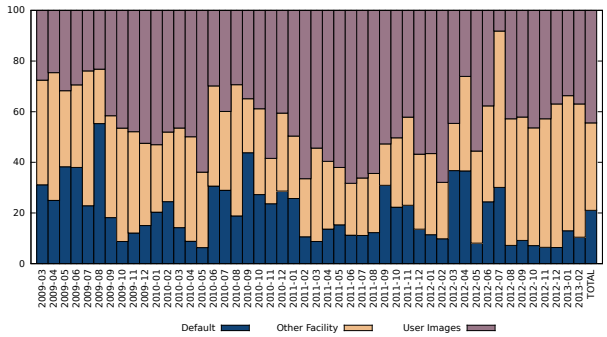

Figure 13: Percentage of requests for three classes of images.

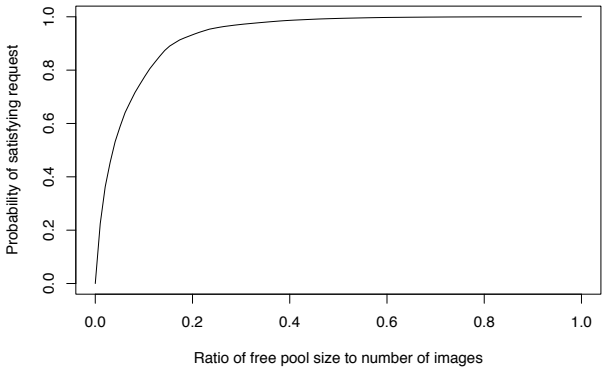

Figure 14: Fraction of requests satisfied from pre-loaded images for varying ratios of free pool size to the working set size.

6.1 Free Pool vs. Working Set Size

We begin with the observation from Section 3.2 that the popularity of user images has a much longer, heavier tail than the set of requests for facility images. Therefore, strategies targeting prediction of facility images are likely to bear more fruit. We also recall from Section 4.1 that there does not exist a consistently dominant image, though Section 4.2 showed us that the working set size over a day is fairly small. This small working set size is encouraging from a prediction standpoint.

An illustration of the potential for prediction can be found in Figure 13, which shows three classes of image requests. Requests for default images can be satisfied by simply pre-loading default images without complicated prediction strategies. This strategy is clearly ineffective in Emulab, as few requests are for the defaults. On top of these are requests for non-default facility images, which represent attractive targets for pre-loading. Finally, we see that approximately 40% of requests are for user images, which are a poor target for prediction because of their long tail. Thus, we target the 60% of requests that are for the relatively predictable facility images.

We first consider how the size of the free pool affects the potential for prediction, where the free pool is defined as the set of idle nodes or disks that are not in use and are thus available for pre-loading. We consider a simple model in which we assume that the inter-arrival time of requests is greater than the time required to load an image. (We will relax this assumption below.) In this model, the determinant of prediction accuracy is the ratio between the size of the free pool and the working set size. In this scenario, the best prediction mechanism is to pre-load those N disks with the N most popular images.

Figure 14 shows the percentage of requests for facility images satisfied under this model, using the empirical request and working set data from Emulab. Intuitively, if there are no disks available for pre-loading, it is not possible to satisfy any requests from pre-loaded machines, and if one can pre-load the entire working set of images (the ratio is 1.0 or greater), it is possible to satisfy all requests. Because the distribution of facility image popularity is roughly exponential, the ability to load the top 25% of images satisfies 95% of all facility requests.

It is interesting to consider how this result applies to different sizes of facilities. In many cases, the size of the free pool will be a fraction of the physical resources, meaning that it is much larger, in absolute terms, for larger facilities. At the same time, we have seen that the working set size of facility images grows linearly with the userbase, but is capped at a relatively small size by the total number of facility images. The practical effect is that small facilities (tens of nodes) are likely to fall on the left side of the curve in Figure 14, meaning that pre-loading is not likely to be particularly effective. Large facilities (thousands of nodes), on the other hand, are likely to be on the far right, with free pool sizes that far exceed the number of facility images—for them, pre-loading is likely to be able to satisfy all requests for facility images. In between these extremes, a facility needs to carefully consider the free pool to working set ratio to determine whether pre-loading makes sense.

6.2 Reload Rate vs. Arrival Rate

Our previous experiment made the simplifying assumption that request inter-arrival time was smaller than the time required to re-load an image; this enables the facility to ensure that the N most popular facility images are loaded at all times, and that only one copy of each image needs to be kept pre-loaded. We now consider the relationship between the arrival rate of new requests and the rate at which the facility can pre-load images in response. If bursts of requests arrive at a faster rate the the facility can re-image, it is useful to have more than one pre-loaded copy of each image. It is also possible for bursts of requests to outpace the facility's ability to keep the image loaded, meaning that there can be mispredictions even for very popular images.

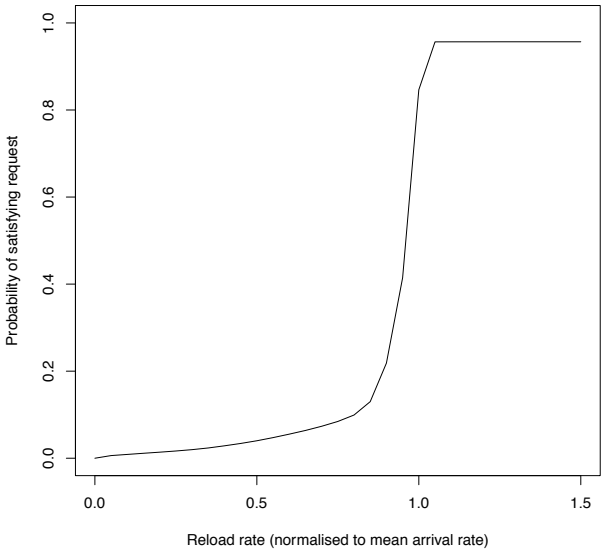

Figure 15: Fraction of requests satisfied against the rate at which images can be pre-loaded.

We model this scenario using standard tools from queuing theory: each image is modeled as a queue, with a number of queue slots equal to the number of disks onto which the image is pre-loaded. The distribution of pre-loaded images is taken directly from the observed distribution of requests; using our results from Section 5.1, we model this distribution as being exponential with $\lambda = 0.143$. We make the standard queuing theory assumption that requests arrive according to a Poisson process [18]. We picked a facility size of 1,000 disks, with an average utilization rate of 90%, meaning that on average, 100 disks are available for pre-loading.

Figure 15 shows the results of a Monte Carlo simulation using this model. We varied the ratio of reload rate to the mean request arrival rate, and find that this ratio is critical. If the facility can reload images at a faster rate than requests arrive (the area to the right of the 1.0 ratio), it can easily keep the proper set of facility images preloaded and can satisfy most requests for these images; this matches the case modeled in Figure 14. If the reload rate is lower (to the left of the 1.0 ratio), the value of preloading falls quickly, as bursts of requests overwhelm the facility's ability to keep a pre-loaded pool that contains the appropriate set of images.

We conclude that pre-loading facility images can be an effective strategy for reducing user wait time, but that the critical determining factors for its success are: (1) the ratio betwen the size of the free pool and the working set size; and (2) the ratio between the facilities' reload rate and the mean arrival rate.

7 Differential Disk Loading

The second optimization we consider targets requests for user images: it may be possible to pre-load facility images, and when requests for user images arrive, load only the blocks that differ. This differential loading strategy is attractive for two reasons. As we saw in Section 5.1, the distribution of user image popularity has a heavy tail, making it difficult to pre-load enough of them to satisfy many requests. But, as we saw in Section 3.4, user images have high levels of similarity to the smaller set of facility images. Thus, we have the potential to reduce user wait times by picking a pre-loaded facility image and doing a fast load of just the blocks that differ. In this section, we develop metrics that quantify the potential benefits of differential disk loading and give us a general understanding of the potential effectiveness of this technique. In order to realize these benefits, additional methods for predicting future requests would need to be developed, which take into account not only image popularity, but also block-level similarity between the pre-loaded images and the images that may be layered on top of them.

We consider only the problem of finding the differences between two disk images, and not the more general problem of taking the difference between a disk image and arbitrary disk state (i.e. the state in which the disk is left by the previous user). Earlier work [6] has shown that disk distribution and installation can run at the full write speed of the target disk, meaning that schemes that require reading disk contents before writing are likely to slow the process down, and are likely to be fruitful only in cases where users do not write much to the disk.

7.1 Limits to Savings

As we have seen, the set of facility images is smaller and more predictable than the set of user images. Thus, as with the last section, we continue to pre-load only facility images; when a user image U is requested, if its base image U_B has been pre-loaded, we need to transfer only $\Delta(U_B, U)$ blocks instead of the full $|u|$ blocks belonging to the image. Clearly, this strategy relies on having the correct set of base images pre-loaded. To simplify, we start by assuming that we have an oracle that tells us what facility images to pre-load or sufficient capacity to preload all facility images; we relax this assumption below.

We begin by defining the number of disk blocks loaded for user images when differential loading is not in use (i.e. the entire user image must be loaded). For an individual image U, this quantity is:

$$|u| \cdot U_C \tag{3}$$

Recall that u is the set of block addresses with defined values in image U, and therefore $|u|$ represents the size

of the image. We define U_C to be the number of times the image is loaded. Intuitively, then, this quantity is simply the number of blocks in the image multiplied by the number of times the image is used.

To obtain the total number of blocks loaded across the universe of all user images, $\mathbb{U}$, we sum the total blocks loaded for each image $U \in \mathbb{U}$:

$$\sum_{U \in \mathbb{U}} |u| \cdot U_C \tag{4}$$

To adapt these equations for differential loading, we substitute $\Delta(U_B, U)$ for $|u|$, giving us the number of blocks that must be loaded assuming the base image has been pre-loaded. This gives us the total number of blocks:

$$\sum_{U \in \mathbb{U}} \Delta(U_B, U) \cdot U_C \tag{5}$$

Differential Savings Potential (DSP): The maximum relative savings from differential loading (assuming the correct U_B images are always loaded) is derived by combining Equation 4 and Equation 5:

$$\text{DSP} = \sum_{U \in \mathbb{U}} \frac{|u| - \Delta(U_B, U)}{|u|} U_C \tag{6}$$

In the Emulab dataset, the values for Equation 4 and Equation 5 are 174 TB and 78 TB, giving a DSP of 0.55. This indicates that, in the presence of an oracle, the Emulab facility could save over half of the blocks it transfers for user images at request time, potentially halving the average time users must wait for custom images to load.

Adjusted Differential Savings (ADS): We next relax the assumption of an oracle. To do so, we use the notation $P[I]$ to indicate the probability that image I is pre-loaded on the facility. We adjust Equation 6 to indicate that with some probability, the user request can be fulfilled with differential loading because the requisite base image is loaded. If not, the entire image must be loaded (resulting in no savings):

$$\text{ADS} = \sum_{U \in \mathbb{U}} P[U_B] \frac{|u| - \Delta(U_B, U)}{|u|} U_C \tag{7}$$

Note that if $P[U_B] = 1$ for all images (perfect prediction), this gives us Equation 6. For smaller $P[U_B]$ values (worse predictions), the adjusted savings are lower than the savings potential, which fits with the intuitive notion that sub-optimal pre-loading will reduce the value of differential loading.

7.2 Savings With Predictions

Figure 16 shows the effectiveness of differential loading as a function of the fraction of facility images that are pre-loaded. The y axis of this graph represents the fraction of blocks that must be loaded at request time, with lower numbers being better, and the limit being $1 - \text{DSP}$ (0.45). Along the x axis, we show the fraction of facility images loaded—we rank facilitiy images by an adjusted popularity that is the sum of their own popularity and the popularity of all users images that use that facility image as a base, and then pre-load the x most popular. What we can see is that relatively few facility images act as bases for user images, so it is necessary to pre-load only a small subset of them (approximately 20%) in order to get most of the benefit of differential loading. This implies that this technique can be effective even on facilities that have low free pool to working set ratios.

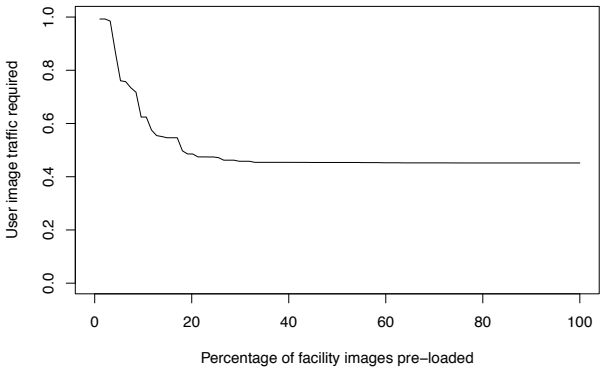

Figure 16: Network traffic required to load user images, when various facility images may be pre-loaded.

Also of interest in Figure 16 is that, for our dataset, the most popular facility images (the default images) are not commonly used as bases for user images—this accounts for the small plateau on the left of the graph. We hypothesize that this is due to the age of Emulab's defaults.

8 Recommendations and Future Work

In our exploration of the Emulab disk image request dataset, we have uncovered a number of properties that can be used to guide the operation and design of disk image storage and installation systems. Based on our analysis, we make the following recommendations:

- Storing images in a deduplicating image store is likely to result in substantial savings. Reads from deduplicating stores can be slow, but the working set size is small enough that it is possible to cache images in faster storage.
- Focusing pre-loading strategies on facility images is likely to produce the best results. The tail of user images is much longer and heavier than the one for facility images, and only a few user images approach the popularity of the heaviest-used facility

images. For very large facilities, it is likely that most facility images appear in the daily working set, making prediction straightforward.

- Pre-loading of a single default image is not a useful strategy, as the diversity of user requests means that no one image, even the default, is dominant on any time scale.

- For small facilities (those where the number of idle disks is significantly smaller than the working set size), pre-loading is likely not a valuable strategy. For large facilities, the number of idle disks is likely to be much larger than the working set size, making simple pre-loading strategies highly effective. To accurately model the effectiveness of pre-loading for mid-sized facilities, additional study of request inter-arrival distributions is necessary.

- Large facilities would do well to focus on techniques that allow them to sustain high reload rates. The only way for pre-loading to be effective is to keep this rate significantly above the request arrival rate, which is likely to be high for large facilities. Techniques of interest include distribution using multicast and image distribution servers spread throughout the datacenter.

- Differential loading has the potential to be effective, especially on facilities with limited free pools. It shows the potential to halve the number of disk blocks transferred to satisfy user requests, but that potential depends on correct predictions when pre-loading the appropriate base images. This changes the criteria for pre-loading, since base images should be selected not only on their own popularity, but also on the popularity of images that may be laid down on top and their block-level similarly with the base image. This complex optimization problem presents an interesting area for future study.

An anonymized version of the dataset used for thus study, plus all code used to analyze it and produce the figures for this paper, can be found at:

`http://aptlab.net/p/tbres/nsdi14`

Acknowledgments

We would like to thank the administrators of Emulab for their assistance in collecting the data used for this study. We would also like to thank Dave Andersen, our shepherd Bruce Maggs, and the anonymous reviewers for their valuable comments. This work was supported by NSF under award CNS-0709427.

References

[1] Amazon Web Services. Amazon EC2 instance store: User guide. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html`.

[2] Amazon Web Services. Amazon Elastic Compute Cloud website. `http://aws.amazon.com/ec2/`.

[3] Amazon Web Services. Amazon Machine Images (AMIs). `https://aws.amazon.com/amis`.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[5] L. A. Barroso and U. Holzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, volume 6 of *Synthesis Lectures on Computer Architecture*. Morgan and Claypool, 2009.

[6] M. Hibler, L. Stoller, J. Lepreau, R. Ricci, and C. Barb. Fast, scalable disk imaging with Frisbee. In *Proc. of the USENIX Annual Technical Conference (ATC)*, pages 283–296, San Antonio, TX, June 2003.

[7] Jean-loup Gailly and Mark Adler. zlib website. `http://www.zlib.org`.

[8] K. Jin and E. L. Miller. The effectiveness of deduplication on virtual machine disk images. In *Proc. of SYSTOR, the Israeli Experimental Systems Conference*, May 2009.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *Proc. of the Workshop on Cloud Computing and its Applications (CCA)*, 2008.

[10] S. Quinlan and S. Dorward. Venti: A new approach to archival storage. In *Proc. of the USENIX Conference on File and Storage Technologies (FAST)*, pages 89–101, Jan. 2002.

[11] Rackspace US, Inc. Rackspace hosting website. `http://www.rackspace.com/`.

[12] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *Computer Communication Review*, 27(1):31–41, Jan. 1997.

[13] P. Sanaga, J. Duerig, R. Ricci, and J. Lepreau. Modeling and emulation of Internet paths. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, Apr. 2009.

[14] The OpenStack Team. OpenStack user documentation. `http://docs.openstack.org/user-guide/`.

[15] The OpenStack Team. OpenStack website. `http://www.openstack.org`.

[16] The University of Utah. Emulab website. `http://www.emulab.net/`.

[17] Wikipedia: Heavy-tailed Distribution. `http://en.wikipedia.org/wiki/Heavy-tailed_distribution`.

[18] Wikipedia: Poisson Process. `http://en.wikipedia.org/wiki/Poisson_process`.