Seagull: Intelligent Cloud Bursting for Enterprise Applications

Tian Guo
UMASS Amherst

Upendra Sharma
UMASS Amherst

Timothy Wood
The George Washington University

Sambit Sahu
IBM Watson

Prashant Shenoy
UMASS Amherst

Abstract

Enterprises with existing IT infrastructure are beginning to employ a hybrid cloud model where the enterprise uses its own private resources for the majority of its computing, but then "bursts" into the cloud when local resources are insufficient. However, current approaches to *cloud bursting* cannot be effectively automated because they heavily rely on system administrator knowledge to make decisions. In this paper we describe Seagull, a system designed to facilitate cloud bursting by determining which applications can be transitioned into the cloud most economically, and automating the movement process at the proper time. We further optimize the deployment of applications into the cloud using an intelligent precopying mechanism that proactively replicates virtualized applications, lowering the bursting time from hours to minutes. Our evaluation illustrates how our prototype can reduce cloud costs by more than 45% when bursting to the cloud, and the incremental cost added by precopying applications is offset by a burst time reduction of nearly 95%.

1 Introduction

Many enterprise applications see dynamic workloads at multiple time scales. Since predicting peak workloads is frequently error-prone and often results in underutilized systems, cloud computing platforms have become popular due to their ability to rapidly provision server and storage capacity to handle workload fluctuations. At the same time, many medium and large enterprises have significant current investments in IT data centers that house compute and storage systems. This IT infrastructure is often sufficient for the majority of their computing needs, while offering greater control and lower operating costs than the cloud. However, workload spikes, both planned and unexpected, can sometimes drive the resource needs of enterprise applications above the level of resources available locally. Rather than incurring capital expenditures for additional server capacity to solely handle such infrequent workload peaks, a hybrid model has emerged where an enterprise leverages its local IT infrastructure for the majority of its computing needs, and supplements with cloud resources whenever local resources are stressed.

Employing cloud bursting can save enterprises a significant amount of money. Figure 1 illustrates a scenario where a business typically requires five "extra large" servers for its daily needs, but two days a week experiences a spike up to ten servers. Using Amazon's EC2 Cost Calculator [2], we can see that a hybrid approach is most efficient and lowers costs by up to 29% a year.

This hybrid technique, which is referred to as "cloud bursting", allows the enterprise to expand its capacity as needed while making efficient use of its existing resources. While commercial and open-source virtualization tools are beginning to support basic cloud bursting functionalities [11, 10, 14], the primary focus has been on the underlying mechanisms to enable the transition of virtual machines between locations. These systems leave significant policy decisions in the hands of system administrators to determine when to invoke cloud bursting and which applications to "burst". This may lead to poor choices in terms of minimizing cloud costs or reducing downtime during the transition, especially when there are a large number of diverse applications in the data center and different cloud platform pricing models.

We have developed Seagull to alleviate the above challenges; Seagull dynamically decides which applications can be moved to the cloud at lowest cost, and then performs the migrations needed to dynamically expand capacity as efficiently as possible. By automating these processes, Seagull is able to respond quickly and efficiently to workload spikes.

The first *insight* of our work is that rather than naïvely moving an overloaded application to the cloud, it may be cheaper and faster to move *different* applications and

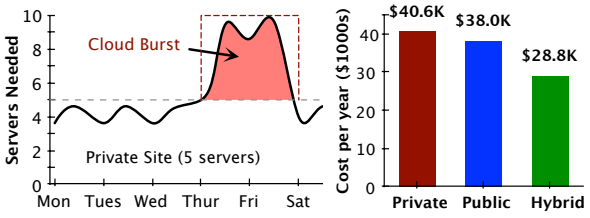

Figure 1: Hybrid clouds can utilize cheaper private resources the majority of the time and burst to the cloud only during periods of peak demand, providing lower cost than exclusively private or public cloud based solutions.

then assign the freed-up server resources to the overloaded application. Bursting an application to the cloud involves copying its disk image and any application data. Since this disk state may be large, a pure on demand migration to the cloud may require hours to copy this large amount of data. The second *insight* of our work is that periodic background precopying of disk snapshots of candidate applications can significantly reduce the cloud bursting latency–since only the incremental delta of the disk state needs to be transferred to reconstruct the disk image in the cloud.

Our paper makes several contributions: **(i)** a placement algorithm that determines which applications should be moved to minimize cost; **(ii)** a precopying algorithm that decides which applications should be proactively replicated to the cloud to enable much faster VM migrations; and **(iii)** a prototype of Seagull and an experimental evaluation of it on a Xen-based local data center and the Amazon EC2 cloud platform. We show Seagull's placement algorithm can make intelligent decisions about which applications to move, lowering the cost of resolving an overloaded large scale data center by over 45%, while precopying significantly lowers burst time with only a modest increase in cost.

2 System Model and Problem Statement

Seagull seeks to enable more agile cloud bursting that can respond to moderate workload spikes within hours or even minutes. We assume that each application is composed of one or more virtual machines that are housed in private data centers, which offer mechanisms for dynamic scaling of server capacity. We assume that applications support either or both of the following mechanisms to scale capacity: i) horizontal scaling: additional replicas are started on demand to increase the capacity ii) vertical scaling: an application's VM is allocated more resources such as more CPU cores. For simplicity, we assume access to a workload forecaster that can predict when a data center is becoming overloaded. We also assume that public cloud follows a resource pricing model

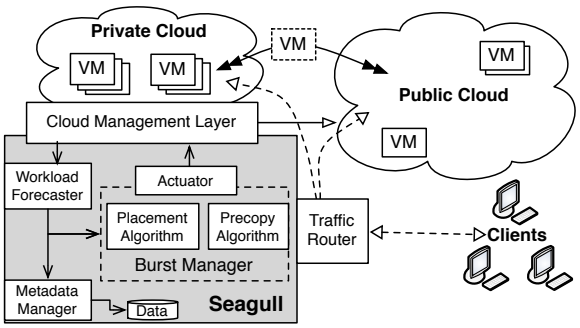

Figure 2: Seagull architecture

similar to Amazon EC2.

We have designed Seagull to both automate and optimize cloud bursting tasks within this setting. In this work, we focus on answering the following two questions: i) Which applications to cloud burst so that cloud server and I/O costs are optimized? ii) How to use judicious precopying to achieve the tradeoff between cloud bursting latency and cost?

3 Placement & Precopying in Seagull

Seagull is composed of multiple interacting components as shown in Figure 2. The *Cloud Management Layer* offers a common interface to interact with both the private and public clouds. The actions performed by this layer are determined by the *Burst Manager*, which is responsible for important decisions about application placement, bursting, and precopying. This section describes the placement and precopying algorithms used by Seagull.

Intelligent Placement The intuition behind the placement algorithm is to maximize the utilization of local resources, which are cheaper than public resources, and migrate the cheapest applications when local resources are insufficient to handle the overload. To do so in a cost-effective manner, the algorithm greedily picks those applications to move that free up the most units of local resources relative to their cost of running in the cloud.[1]

To determine which applications should be moved, we assume the duration of the workload spike, L, and the desired capacity C for each virtual machine are known. Note that C is a vector representing the CPU, disk, network and memory capacity needs of each VM. We define the cost of bursting an application, say A, that is composed of n virtual machines in terms of the cost of transferring its memory and storage, storing the data, and then

[1] Additional administrative criteria such as security policies may also preclude some applications from being valid cloud burst targets; we assume that system administrators provide this information as a cloud bursting black list.

running it in the public cloud:

$$Cost = \sum_{j=1}^{n} C\_tran_j + C\_stor_j + C\_run_j * L, \quad (1)$$

where C_tran_j and C_stor_j are calculated based on the amount of data that must be transferred and stored in the cloud to run the j^{th} VM of A (i.e. VM_j); C_run_j is determined based on the capacity requirements, C, of the virtual machine (e.g., the number of cores it requires) and must be multiplied by L to account for the length of time the VM would need to remain in the cloud before the workload spike passes. We sum the cost across all VMs in the application to account for the constraint that all virtual machines that comprise an application be grouped together either in the local data center or on the cloud. Notice that we can easily plug in different cost functions to account for different pricing models and scenarios such as deploying VMs from the same applications across different data centers.

We can use Equation 1 to calculate the cost of bursting the overloaded application, however, Seagull must decide whether to simply move the overloaded application itself, or to find a different set of applications that can be moved more cheaply in its place. To this end, Seagull must consider each of the VMs that make up the overloaded application and see if there is a way to meet their resource requirements in the local data center by either local reconsolidation or selecting one or more different applications to burst.

The virtual machines of the overloaded application are considered in decreasing order of their resource requirements. For each of these virtual machines, Seagull considers the potential hosts in the local data center sorted by two criteria: 1) their free capacity in descending order and 2) the total cost, in increasing order, of moving all *applications* (including related VMs) on the host to the cloud. The first criteria biases Seagull towards utilizing the free capacity in the local data center first, potentially reducing the number of applications that need to be moved to the cloud. The second criteria ensures that hosts running low cost applications are considered first.

When hosts have been sorted in this way, the algorithm considers the first host and attempts to decide if a set of VMs on that host can be moved in order to create space for the overloaded VM. Each virtual machine, VM_j on the host is ranked based on: $num\_cores_j/Cost$, where $Cost$ is the cost of moving the full application that VM_j is part of, and num_cores_j is the number of CPU cores currently in use by the virtual machine. The VMs on the host are considered in decreasing order of this criteria, and the first k VMs are selected such that the free capacity they will generate is sufficient to host the overloaded virtual machine. The intuition behind this greedy heuristic is that it *optimizes the amount of local capacity freed per dollar spent running applications in the cloud*.

Each of the overloaded applications is considered for bursting using this metric. When a solution is found, the total cost of moving all of the marked applications is compared to moving just the overloaded application; the cheaper of the two options is chosen in each case.

Opportunistic Precopying In general, an application's state may be very large. Migrating all of this data at cloud bursting time can take hours or even days, significantly reducing the agility with which a data center can respond to rising workloads.

Seagull performs precopying by transferring an incremental snapshot of a virtual machine's disk-state to the cloud. Seagull's precopying technique must make two important decisions: i) *which* applications to precopy, and ii) *how frequently* to precopy each one. Each of these decisions leads to a cost-benefit tradeoff. The larger the set of candidate applications chosen for precopying, the greater the chances Seagull's cloud bursting algorithm will pick one of the precopied applications to burst to the cloud when the peak workload arrives, increasing the agility of the system to respond to local stress. Similarly, the more frequently each application is precopied to the cloud, the smaller the delta will be, leading to a smaller bursting latency. Thus a careful choice of the candidate set of applications to precopy and precopying frequency can both reduce the overheads. We have implemented a strategy that computes a set of candidate applications to balance the benefits of precopying against its cost, and also two baseline strategies for comparison.

Our cost-benefit tradeoff strategy first generates an *overload list*, i.e. a list of applications likely to become overloaded[2]. Seagull then runs its cloud bursting algorithm, from the previous section, in an *offline* mode over the *overload list*. That is, for each application A on the overload list, Seagull runs its algorithm to see which application(s) get chosen for bursting if A were to become overloaded. These applications form the *precopy list*.

The disk state of applications in the *precopy list* is replicated to the cloud based on a frequency strategy. In the simplest case the precopy frequency can be chosen statically—say once a day or once a week. However, Seagull can analyze the write rates to the virtual disks to "tune" the precopy frequency for each application in the list, managing overall cloud costs while retaining agility.

The two other strategies that we use as baselines for our comparitive evaluation are: i) *Random Precopying:* selects a random set of applications to be precopied based on the maximum expected overload (e.g., ran-

[2]Seagull can generate such a list based on the history of prior cloudburst instances and system administrators can alter it, based on their expert knowledge of which overload scenarios are still likely in the future.

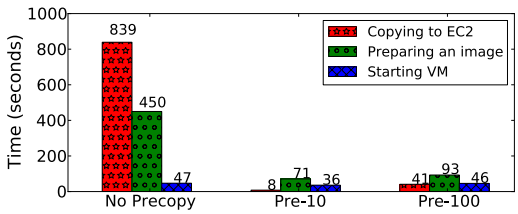

Figure 3: Without precopying, a cloud burst can take tens of minutes to copy data.

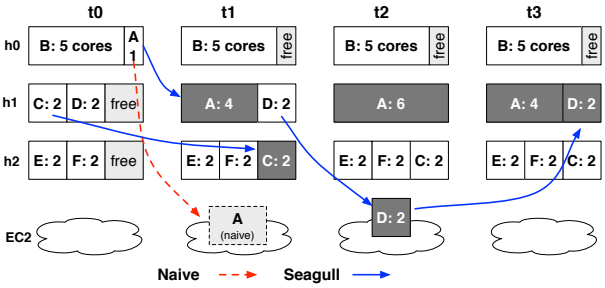

Figure 4: The naïve approach uses only one migration, immediately moving A from h_0 to the cloud. Seagull initially avoids any cloud costs by rebalancing locally, and is able to move back from the cloud sooner than the naïve approach.

domly precopy 20% of the data center's applications). ii) *Naïve Precopying:* selects the set of applications that is predicted to become overloaded for precopying.

4 Experimental Setup and Evaluation

We have implemented Seagull's placement and precopying algorithms as modules that extend the OpenNebula cloud management software. We have created a private cloud environment on a lab cluster using OpenNebula over the Xen-hypervisor and used Amazon EC2 as our public cloud. We use three applications, TPC-W, Wikibooks and CloudStone for our evaluation. We have created private-cloud as well as public-cloud appliances for each of these three applications and their respective client applications. An appliance instance will create the virtual machine(s) which house the complete application. We warm up each application, using its clients, for two minutes before collecting data.

Cloud Bursting Time The total time to perform a cloud burst can be decomposed into three major parts: copying data to the cloud, preparing an application image, and booting up the virtual machine. To measure each of these components, we migrate a virtual machine running the CloudStone application with a disk-state size of 5GB.

As shown in Figure 3, the total time to migrate an application with even a very small 5GB disk state, is 1336 secs ($\sim$ 22 mins); this clearly illustrates the need for precopying in real applications that may have ten or more times as much state. We next precopy the application and reduce the delta (i.e. difference between the original and precopied snapshot) to 10MB or 100MB; the total time to burst the application significantly reduces to less than 200 secs for a delta of 100 MB. Note that as delta reduces the image preparation time and boot time start to flatten around 120 secs and become the prime component of total bursting time.

Placement Algorithm In this experiment, we analyze the placement efficiency of Seagull compared to a naïve algorithm (which always cloudbursts the overloaded application), in a small scenario that demonstrates the intuition behind Seagull's decision making. We show that when a hotspot occurs, Seagull is able to make better use of local resources as well as pick cheaper applications to move to the cloud.

We use three 6-core physical servers, each hosting a pair of different applications: TPC-W (VMs A and D) Wikibooks (B, E), and CloudStone (C, F). Each application is running inside a single VM and can be scaled up vertically. The initial arrangement of applications and the number of cores dedicated to each is shown under t_0 in Figure 4. To simplify the scenario, we assume that all applications have identical storage requirements.

We change application A's workload every hour (marked by instants t_i, where $i = 1 \ldots 3$) such that its CPU requirement increases to four cores, then six cores, before falling back to four cores at t_3. To eliminate the impact of prediction errors in this experiment we assume a perfect forecaster.

Results: When Seagull detects the first upcoming workload spike at t_1, it attempts to resolve the hotspot by repacking the local machines, shifting application C to h_2 and then moving A to h_1 at effectively no cost. In the naïve solution, application A is cloud burst to EC2 directly without considering local reshuffling.

In the workload's second phase, Seagull migrates a cheaper application, D, to EC2 since the local data center could not provide enough capacity needed for A. On the other hand, the naïve algorithm had already moved A to the cloud, so it simply allocates extra resources to it making it more expensive.

Eventually, the workload spike for application A passes, Seagull migrates D back to the local data center while the naïve algorithm, lacking the ability to perform local reshuffling, still needs to keep A in the cloud, wasting more money.

Time and Monetary Cost: The use of local resources in Seagull allows it to respond to overload faster than the naïve approach. Figure 5 shows the amount of time spent by each approach to resolve the hotspots at each measurement interval; note that for both systems we pre-

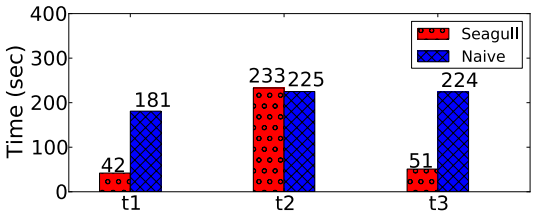

Figure 5: Seagull uses local, live migrations at t_1, and benefits from reverse pre-copying at t_3, substantially reducing the time spent at each stage compared to naïvely cloud bursting at t_1 and restarting instances at t_2 and t_3.

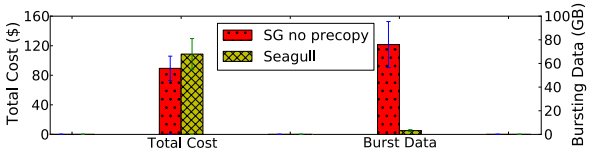

Figure 6: Precopying causes a marginal increase in cost, but a dramatic reduction in burst time.

copy all applications once to the cloud before the experiment begins. Seagull is substantially faster because it uses only a local, live migration at t_1 whereas the naïve approach approach requires a full cloud burst. Subsequent actions performed by Naïve also incur substantial downtime since VMs must be rebooted in the cloud to adjust their instance type to obtain more cores. Seagull's migration back from the cloud at t_3 is also quite fast because it does not require the full image registration process needed for moving into the cloud. Most importantly, the fact that Seagull only requires a virtual machine in the cloud for the hour starting at t_2 means that it pays 30% less in cloud data transfer and instance running costs.

Precopying Algorithm In order to study the impact of precopying, we conducted an experiment by simulating a data center comprised of 200 quad-core hosts and populated it with three types of applications with different disk size and update rate[3] .

We first instruct Seagull to precopy 60 applications to the cloud, and then simulate a hotspot scenario where 30% of the data center becomes overloaded. With this level of precopying, Seagull was able to resolve the hotspot solely by bursting applications which had already been precopied, dramatically reducing the cloud burst time compared to an approach with no precopying. Figure 6 show a modest 22% increase in cost due to precopying, but a substantial 95% saving in data transfer. However, this trade-off can be further tuned based on the precopying algorithm and parameters.

Comparisons: We next evaluate the effectiveness of Seagull's intelligent precopying strategy compared to the random (*SG-random*) and naïve precopying strategies at a larger scale. We use Seagull's placement algorithm to determine the total burst cost when using each of these precopying techniques. We study the decisions made when the level of overload in the data center increases from 10 to 30 percent. Figure 7 presents the average performance of these three strategies when the simulation is repeated 40 times for each level of overload.

In Figure 7(a), Seagull achieves the lowest precopying cost across all overload levels. The benefits of Seagull increase with rising overload levels, and it is able to lower precopying costs by up to 75%. The naïve approach shows the highest cost because there are often applications which can be precopied more cheaply than those which are expected to become overloaded.

Figure 7(b) shows the total cost including both precopying and cloud bursting. Seagull reduces the cost by 45% compared to the Naïve approach because naïvely running the overloaded applications in the public cloud is more costly. SG-Random and Seagull have similar total cost because they select the same applications to burst.

Figure 7(c) shows the total amount of data sent during cloud bursting, which can be used as a proxy for total burst time. Our intelligent precopying strategy far outperforms SG-Random because the latter has a poor chance of precopying the applications that will be selected by the placement algorithm. The naïve algorithm precopies and bursts the same applications, giving performance similar to Seagull, although at higher cost.

5 Related Work

Cloud Computing covers a wide range of types of systems; in this work we focus on Infrastructure as a Service (IaaS) platforms such as Amazon's Elastic Compute Cloud. Armbrust et al. provide a survey of cloud computing [1], and specifically list "scaling quickly" as one of the key opportunities in cloud computing. Many recent projects automate virtual machine or storage migration to balance the CPU, memory, or I/O loads within a single data center [6, 13]. It is increasingly common for businesses and service providers to own multiple data centers, so managing resources across data centers is a growing challenge [12, 4]. We believe that operational expense will naturally expand the automated resource management techniques to include cross data center management approaches like cloud bursting as data centers become connected by increasingly high bandwidth links.

Cloud Bursting was first proposed by Amazon's Jeff Barr as a way to allow enterprises who already own significant amounts of IT infrastructure to still make use of the cloud during periods of high demand [5]. Re-

[3]To eliminate the impact of Seagull's local reshuffling on precopying efficiency, we assume that the data center runs only horizontal-scaling applications, preventing the need for local reconsolidating.

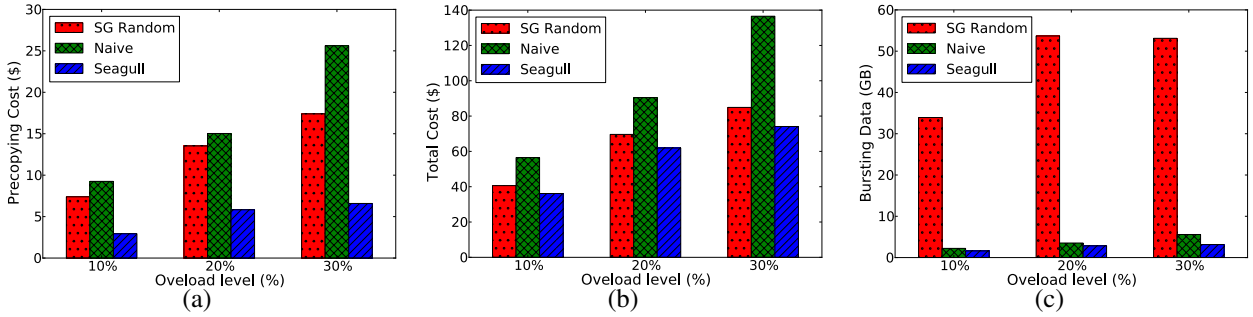

Figure 7: Intelligent precopying reduces total cost and data transferred by over 45% compared to the naïve algorithm.

searchers have been investigating the potential economic savings by using cloud bursting in specific domains such as medical image processing [8] and publishing [7].

Live VM migration over WAN attempt to seamlessly move the memory and storage of a virtual machine between data center sites, usually by building upon the existing LAN migration tools included in modern hypervisors [3]. Alternatively, storage migration tools move only the disk state of applications [9]. Our prototype focuses on storage migration due to limitations of current cloud platforms; however, we note that Seagull could easily be enhanced to support full VM live migration.

6 Conclusions and Future Work

Cloud bursting is a technique to dynamically move applications running in a private data center to the public cloud to take advantage of additional resources there. In this work we propose Seagull, a cloud bursting system that efficiently precopies and migrates applications to the cloud when local infrastructure becomes overloaded. This allows Seagull to perform agile provisioning of resources across a private data center and the cloud, resulting in more efficient utilization of local resources while incurring only minimal expense in the cloud.

In future work, we plan to extend our Seagull prototype to provide complete cloud bursting automation. There are a number of challenges remaining, such as determining when to initiate a cloud burst even if accurate prediction models of future workloads are not available, defining standardized cloud interfaces to enable live WAN migration to public clouds, and integrating Seagull's placement algorithm with business policy requirements.

Acknowledgements: We thank our shepherd Pradeep Padala and reviewers for their comments. This research was supported in part by NSF grants CNS-1117221, CNS-0916972, CNS-0855128 and OCI-1032765 and an Amazon AWS grant. Upendra Sharma was supported by an IBM Graduate fellowship.

References

[1] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. Above the clouds: A berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, UCB, Feb. 2009.

[2] AWS Economics Center. http://aws.amazon.com/economics/.

[3] BRADFORD, R., KOTSOVINOS, E., FELDMANN, A., AND SCHIÖBERG, H. Live wide-area migration of virtual machines including local persistent state. In *VEE* (San Diego, California, USA, 2007), ACM, pp. 169–179.

[4] BUYYA, R., RANJAN, R., AND CALHEIROS, R. N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing* (2010).

[5] Cloudbursting - hybrid application hosting. http://aws.typepad.com/aws/2008/08/cloudbursting-.html, Aug. 2008.

[6] GULATI, A., SHANMUGANATHAN, G., AHMAD, I., WALDSPURGER, C., AND UYSAL, M. Pesto: online storage performance management in virtualized datacenters. In *SOCC* (New York, NY, USA, 2011), SOCC '11, ACM, pp. 19:1–19:14.

[7] KAILASAM, S., GNANASAMBANDAM, N., DHARANIPRAGADA, J., AND SHARMA, N. Optimizing service level agreements for autonomic cloud bursting schedulers. In *ICPP Workshops* (2010), pp. 285–294.

[8] KIM, H., PARASHAR, M., FORAN, D. J., AND YANG, L. Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *GRID* (2009), IEEE, pp. 34–41.

[9] MASHTIZADEH, A., CELEBI, E., GARFINKEL, T., AND CAI, M. The design and evolution of live storage migration in vmware esx. In *USENIX ATC* (Berkeley, CA, USA, 2011), pp. 14–14.

[10] Open Nebula: The Open Source Toolkit for Data Center Virtualization. http://www.opennebula.org.

[11] openstack: Cloud Software. http://www.openstack.org.

[12] ROCHWERGER, B., BREITGAND, D., EPSTEIN, A., HADAS, D., LOY, I., NAGIN, K., TORDSSON, J., RAGUSA, C., VILLARI, M., CLAYMAN, S., LEVY, E., MARASCHINI, A., MASSONET, P., MUNOZ, H., AND TOFFETTI, G. Reservoir - when one cloud is not enough. *Computer 44* (2011), 44–51.

[13] SHEN, Z., SUBBIAH, S., GU, X., AND WILKES, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. SOCC '11, ACM, pp. 5:1–5:14.

[14] VMware: Public & Hybrid Cloud Computing. http://www.vmware.com/solutions/cloud-computing/public-cloud/products.html .