Cost-Aware Live Migration of Services in the Cloud

David Breitgand
IBM, Haifa Research Lab

Gilad Kutiel, Danny Raz
Department of Computer Science Technion, Israel

Abstract

Live migration of virtual machines is an important component of the emerging cloud computing paradigm. While live migration provides extreme versatility of management, it comes at a price of degraded service performance during migration. The bulk of studies devoted to live migration of virtual machines focus on the duration of the copy phase as a primary metric of migration performance. While shorter down times are clearly desirable, the pre-copy phase imposes an overhead on the infrastructure that may result in severe performance degradation of the migrated and collocated services offsetting the benefits accrued through live migration.

We observe that there is a non-trivial trade-off between minimizing the copy phase duration and maintaining an acceptable quality of service during the pre-copy phase, and introduce a new model to quantify this trade-off. We then show that using our model, an optimal migration schedule can be efficiently calculated. Finally, we simulate, using real traces, live migrations of a virtual machine running a web server and compare the migration cost using our algorithm and commonly used live-migration methods.

1 Introduction

Cloud computing platforms allow hosting of multiple services on a globally shared resource pool where resources are allocated to services on demand. Recent advances in the server virtualization technologies radically improve flexibility and versatility of resource provisioning. This is done through the ability to collocate several virtual machines (VM) on the same physical host, to dynamically change VM resource allotments and to migrate VMs across physical servers both within the same data center and across disparate data centers.

Live migration of VMs is an essential tool for the management of Cloud Computing. For example, using live migration, service SLA (Service Level Agreement) compliance can be improved by migrating excessive load to a less loaded host or by migrating a VM from the host that will go down for maintenance, thus complying with performance and availability SLA of the service respectively. However, if used carelessly on the shared network infrastructure, live migration overhead may cause the network-bound services to violate their SLAs.

The migration process consists of transferring the service's memory image from the source host to the destination host. In the off-line migration process the source VM is suspended, the entire VM image is copied to the destination physical host, and then the copied VM is restarted on the destination host. With live migration, most of the migration process takes place while the source VM continues to run, and the service is alive. With this method the service is suspended for a very short period of time before it is restarted on the destination host. Clearly, live migration has an advantage of keeping the service's availability with only a short period of service downtime.

One approach for live migration is the *pre-copy* approach in which memory pages are iteratively copied from the source machine to the destination one, all without stopping the execution of the VM being migrated. The page copying process may take several iterations during which dirty pages are continuously transferred. Normally, since the server's pages are being modified continuously, one must, at some point, stop the server until all the pages have been fully transferred to the destination (referred in what follows as the *copy phase*). Subsequently, the VM can be resumed at the destination host. The actual task of migrating the pages consumes computation resources and thus may degrade the service performance. Furthermore, if live migration is being performed *in-band* (i.e., the same network bandwidth is being used by the migration process and by the service running in the VM) then we expect even a more severe degradation in the QoS (quality of service) due to the fact that migration consumes some of the bandwidth used by clients of the service.

There is a non-trivial trade-off between minimizing the copy phase duration and maintaining an acceptable quality of service during the pre-copy phase. In this paper we concentrate on modeling and optimizing the live migration process assuming the pre-copy approach. Our results, however, are general and apply to post-copy migration as well.

In order to conduct a quantitative study of migration in this context, we start by evaluating the expected degradation in service level due to bandwidth limitations. Without loss of generality, we consider response time SLAs for network-bound services. As expected, the probability of service request to be satisfied by its SLA determined deadline, decreases with the available bandwidth. Where the exact parameters vary from service to service. These results give rise to a cost function model that quantifies service degradation as a function of the residual band-

This research was partly supported by The Israel Science Foundation ISF, and by the European Union's Seventh Framework Programme under grant agreements n° 257448, 215605.

width (i.e., not used by the migration process).

Using this cost function, we can quantify the overall cost of a live migration which is the sum of the pre-copy phase cost and the down time cost, and study the algorithmic aspects of finding an optimal bandwidth allocation schedule for in-band live migration. In other words, we define the amount of bandwidth to be used for the migration in each step (time) of the pre-copy phase[1].

We first address a simpler model in which the bandwidth used is fixed throughout the entire pre-copy phase. We develop optimal strategy for this case, which is of an independent interest, and test it using real traces. We then use similar building blocks to develop an optimal migration strategy for the general case, where the amount of bandwidth used for migration can vary over time.

Our main contributions are as follows:

- We formulate a novel analytical framework that allows to quantify the total cost of live in-band migration including both pre-copy and copy phases;
- We develop an optimal bandwidth allocation algorithm for in-band live migration;
- We perform a thorough trace-driven simulation study of the proposed algorithms where the traces are obtained from a real, albeit intentionally simplified system.

2 Related Work

Clark et al. studied in [5] live migration of entire OS instances. In their implementation they addressed several of the issues and trade-offs involved in live local-area migration. These issues are (a) minimizing the downtime during which services are entirely unavailable, (b) minimizing the total migration time, during which state on both machines is synchronized and thus may affect reliability and (c) ensuring that migration does not unnecessarily disrupt active services through resource contention (e.g., CPU, network bandwidth) with the migrating OS. The algorithm they used is as follows: in the first pre-copy round they transfer pages using a minimum bandwidth value. Each subsequent round counts the number of pages dirtied in the previous round, and divides this by the duration of the previous round to calculate the dirtying rate. The bandwidth limit for the next round is then determined by adding a constant increment to the previous amount, compensation for the dirtying process. They empirically determined that 50Mbit/sec is a suitable value for this constant.

Timothy Wood et al. [10] propose a smart stop and copy mechanism to optimize WAN VM migration.

Voorsluys et al. [11] presented a performance evaluation of the effect of live migration of virtual machines on the performance of applications running inside Xen VMs. They found that in most cases, migration overhead is acceptable but cannot be disregarded, especially in systems where service availability and responsiveness are governed by strict SLAs. In such systems, service providers and consumers agree upon a minimum service level and non-compliance to such agreement may incur penalties to providers [2].

Sherif et al. [1] characterize the parameters that affect migration time and provide two simulation models that are able to predict migration time to within 90% accuracy for both synthetic and real-world benchmarks.

Ming Zhao and Renato J. Figueiredo [12] conduct an experimental study and showed that based on a VMs migration time it is possible to estimate the expected time of migration of other VMs having other configuration.

Checconi et al. [3] introduced a stochastic model for the live migration process. In this model the migration is done with a fixed bandwidth and each memory page of the migrated VM has a different probability to be accessed during a given time frame. It is shown, under these assumptions, that there exists an optimal ordering of pages being migrated which minimizes the expected migration time.

A number of other studies presented individual and side by side measurements of VM run-time overhead imposed by hyper-visors on a variety of workloads [6, 4]. To the best of our knowledge, this work is the first one in which the total cost of a live migration of virtual machine is modeled and migration algorithms for an optimal cost are studied.

3 The Cost Of Live Migration

As described in the introduction, in-band live-migration consumes part of the bandwidth used to serve clients and thus may affect user's perspective of the service. In particular, we may expect to see a degradation in the QoS. In order to study optimal migration strategies, we need first to understand this phenomenon in a quantitative manner. We assume that an SLA is associated with the service, that specifies the maximum response time of a query to the service[2]. Then, we define the Service Level (SL) to be the percentage of requests that are being served within this time bound (deadline). The service SL depends on the workload and specifically on the request's rate, the service itself, and the availability of resources (CPU, memory access, etc.) at the server. However, it also depends on the bandwidth available on the links from the servers to the clients.

Assume that the total bandwidth available (both for the

[1] Clearly, the copy phase will use all the available bandwidth since we do not have to consider the service level since the service is down.

[2] This is just a one reasonable KPI, the algorithm presented in this paper is general and holds for any other restrictions implied by the SLA

Figure 1: Olio Service Degradation

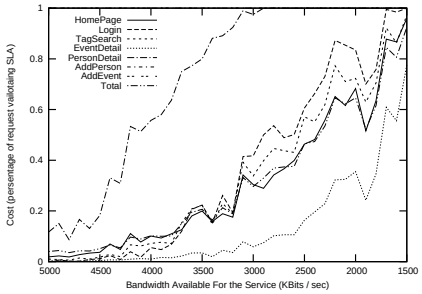


Table 1: Olio Services Distribution and Deadline

Operation	Distribution	Deadline
HomePage	26.15%	1 second
Login	10.22%	1 second
TagSearch	33.45%	2 second
EventDetail	24.68%	2 second
PersonDetail	2.61%	2 second
AddPerson	0.84%	3 second
AddEvent	2.84%	3 second

migration process and for the service itself) is B[3] and the migration process consuming $B_m \leq B$ of it, then the residual bandwidth utilized by the running service is $B_s = B - B_m$. Clearly, increasing B_m would speed up the migration process, but at the same time it will reduce B_s and thus may also reduce the service level during this time. We define the *cost function* $F(B_s)$ to be the portion of the requests that are not satisfied by their deadline. In other words, if S denote the serving time of a request and t_{SLA} denote the period, specified in the SLA, for a request to be served then we can say that: $F(B_s) = P[S > t_{SLA}]$.

While the algorithms presented in this paper do not specifically depend on this cost function, it is important to understand what is a typical cost function for a Web based service, in order to better understand the effect of in-band migration on the services. We conducted a set of experiments in which we emulated users requests to several web based services, and measured the portion of the requests not satisfied within the SLA time bound as a function of the residual bandwidth.

Figure 1 depicts the cost function of seven request types that are part of the services offered by Apache Olio [7] as a function of the available bandwidth. Olio is a toolkit which is part of the Apache framework that helps developers evaluate the suitability, functionality and performance of various web technologies. The workload was generated using Faban [9] by emulating 100 concurrent users. Each user performs one of a seven possible operations every 5 seconds. Operations are chosen randomly with probabilities and deadlines as depicted in Table 1, a detailed description of the workload can be found in [8].

The curves, presented in Figure 1 show that if the residual bandwidth is large enough all requests are satisfied within their SLA time bound. When the residual bandwidth decreases, the probability of a request not to be satisfied by this time limit increases and when the bandwidth is very limited, almost non of the requested are satisfied by their SLA based time bound.

To construct a first order approximation of response time and simplify the analysis, we consider M/M/1 model for the server. As our experimentation shows, this model is sufficient to gain insights into the problem. Denote by $\mu(B_s)$ the rate at which the server is capable of sending responses given residual bandwidth B_S, then $\mu(B_s) = B_s/Average\ Response\ Size$. Denote by $E(S)$ the expected serving time of a request, then, as known from the basic queuing theory, the probability that the time it takes to handle request is greater than $aE(S)$ is given by $P[S > aE(S)] = e^{-a}$ where $E(S) = \frac{1}{1-\rho} \cdot \frac{1}{\mu}$, $\rho = \frac{\lambda}{\mu} \Rightarrow E(S) = \frac{1}{\mu-\lambda}$. If $t_{SLA} = aE(S)$ then

$$F(B_s) = P[S > t_{SLA}] = e^{\frac{-t_{SLA}}{E(S)}} = e^{t_{SLA}(\lambda - \mu(B_s))} \quad (1)$$

Now, given t_{SLA}, the time limitation for a request to be served, λ, the rate at which requests arrive to the server and the average response size, one can find the expected cost for a given service as a function of the residual bandwidth for the service B_s.

To test this model we conducted an experiment as follows. An Apache Tomcat served a single 1 MB static file. A client machine generated requests to this server in a Poisson process with average rate of 20 requests per second. The available bandwidth of the server was gradually throttled while we measured the percentage of requests that were not completed within the 1 second limitation. The cost function for such server, using Equation: 1 is $F(B_s) = e^{20-\frac{B_s}{8}}$. This cost function alongside the measurements we performed are depicted in Figure 2.

One can see the good correlation between the modeled function and the actual measurements. In our evaluations throughout the rest of the paper we use the same workload on the server being migrated, thus, applying the cost function $F(B_s) = e^{20-\frac{B_s}{8}}$.

4 Fixed Bandwidth Migration

In this section we focus on a model where migration bandwidth B_m remains constant throughout the pre-copy

[3]For simplicity we assume that each service has a bandwidth allocation that is used both for serving clients and for management tasks.

Figure 2: Cost Function Model

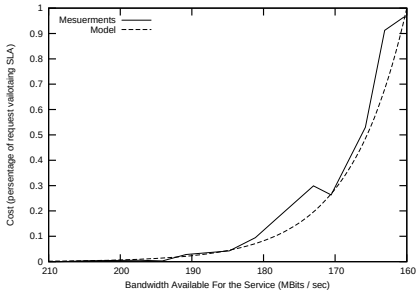
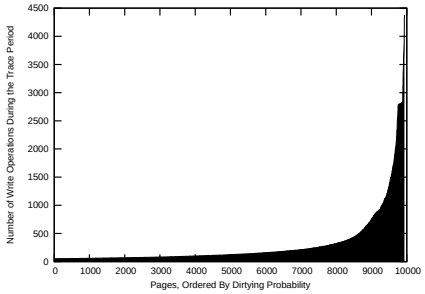


Figure 3: Write Frequency Distribution Over Pages



phase. Our objective is twofold: we seek (a) an optimal B_m to minimize the total cost of the live in-band migration process, and (b) an optimal stopping condition for the pre-copy phase expressed in terms of the number of pages that should be transferred before the copy phase starts.

During the pre-copy phase, pages transferred to the destination while VM remains alive at the source may become dirty again. To analyze this process, we define the dirtying probability of page i (denoted by p_i) to be the probability of page i to be written in a single time unit. In practice different pages may have different dirtying-probability and a write operation to page i may increase the probability to observe such operation in adjacent pages. However, to simplify the analysis of the algorithm we assume that p_i is uniform and independent. We discuss the actual distribution of the dirtying probability in real traces later in this section.

As explained in Section 3, the cost function is given by $F(B - B_m) = F(B_s)$, where $F(0) = 1$, i.e., when no bandwidth is available for the service the cost function is normalized to one. A page is *clean* if it was sent to the destination and no write operation was experienced by it since the last time it was sent; a page that is not clean is *dirty*. We use $clean(t)$ to indicate the number of clean pages at time t.

First, we give an expression of the number of clean pages after t time units using a bandwidth of B_m (where B_m is expressed in pages per time unit). Assume that we have N_1 clean pages at time $t = 0$, then the expected number of clean pages can be expressed in terms of t and B_m as follows:

$$E(clean(t)) = N_1 \cdot q^t + \sum_{i=0}^{B_m \cdot t - 1} q^{\frac{i}{B_m}} = N_1 \cdot q^t + \frac{(1 - q^t)}{1 - q^{\frac{1}{B_m}}} \quad (2)$$

We obtain Equation 2 as follows, the probability of a clean page to stay clean after t time units is q^t and thus if we already have N_1 clean pages the expected number of clean pages after t time units is $N_1 \cdot q^t$. Now, using a bandwidth B_m for t time units we transfer $B_m \cdot t$ pages. The first page that is transferred, becomes clean after $\frac{1}{B_m}$ time units and thus it has a probability of $q^{t - \frac{1}{B_m}}$ to stay clean at the end of this process. The second page that is transferred, becomes clean after $\frac{2}{B_m}$ time units and thus it has a probability of $q^{t - \frac{2}{B_m}}$ to stay clean at the end of this process and in general the ith page has a probability of $q^{t - \frac{i}{B_m}}$ to remain clean at the end of the phase. Summing the probabilities of the $B_m \cdot t$ pages that being transferred we get $\sum_{i=0}^{B_m \cdot t - 1} q^{\frac{i}{B_m}}$.

We now want to compute the expected time it takes to move from a state of N_1 clean pages to a state of $N_2 > N_1$ clean pages, using bandwidth B_m. Using Equation 2 we get:

$$t = \frac{1}{ln(q)} \cdot ln\left(\frac{N_2 \cdot (1 - q^{\frac{1}{B_m}}) - 1}{N_1 \cdot (1 - q^{\frac{1}{B_m}}) - 1}\right) \quad (3)$$

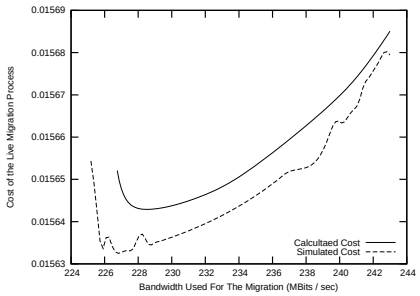
The expected cost of the pre-copy phase ($N_1 = 0, N_2 = N$) is then: $Cost_{pc} = \frac{1}{ln(q)} \cdot ln(1 - N \cdot (1 - q^{\frac{1}{B_m}})) \cdot F(B - B_m)$ and the total cost of the migration process is given by: $Cost_{pc} + \frac{M-N}{B} \cdot F(0)$ where M is the total number of pages to be migrated. One can now minimize this value by finding the optimal bandwidth (and optimal cost) per each N value, and then find the overall minimal N[4].

We now test the proposed solution on a real-world application. To do that we simulate a live migration of a VM with 512MB RAM and link capacity of 512MBit / sec, running an Apache Tomcat web server as described in Section 3, where the cost function is given by $F(B_s) = e^{20 - \frac{B_s}{8}}$.

Pages of a real-world application have, of course, a different dirtying behavior, thus we start by examining the dirtying pattern of the Tomcat server under the described load. We generated a trace of the write operations to memory pages in such a VM during a period of 2 minutes, the number of write operations to each page during

[4]The more general case where several services being migrated sequentially can be handled in a similar manner where there is a global cost function that is the sum of the cost functions of those services.

Figure 4: Comparing The Calculated Cost With The Simulated Cost (Fixed Bandwidth Migration



this period can be seen in Figure 3. This figure displays the 10,000 pages (out of 133,233) with the highest dirtying probability. One can see that a very small fraction (about 1%) of the pages has a very high dirtying rate (this group of pages was referred as *Writable Working Sets* at [5]), and there is the major group of the pages with a low dirtying probability.

Using this trace of write operations, we simulate a live migration using a range of (fixed throughout the migration process) bandwidths and compare the simulated cost with the calculated one. The stop-condition during the simulations is fixed and was arbitrarily set to 256 pages. The results can be seen in Figure 4. As one might expect, there is a difference between the calculated cost and the simulated one. This difference is the result of the non-independent and non-uniform dirtying probability of a real-world application. However, one can see that the optimal calculated bandwidth value (228Mbit/sec) is very close to the optimal simulated bandwidth value (227Mbit/sec). Furthermore, using the bandwidth value resulting from calculation for the real migration will result in a cost very close to the minimal cost possible. The algorithm described in this section will be referred from now on as CALM-fixed (Cost Aware Live Migration).

5 Variable Bandwidth Migration

We now consider the case where the bandwidth used for the migration (B_m) can vary over time. That is, the algorithm needs to decide at each point in time how much bandwidth to use. Recall that we start with 0 clean pages (since no page is transfered at this time) and we need to end the pre-copy phase with a certain number of clean pages and then perform the copy phase. Our algorithm works in steps, where in step i we move from $i-1$ clean pages to i clean pages, until we decide to terminate the pre-copy phase, and we copy the rest of the pages while the service is down.

Using Equation 3 we get that the step cost, C_i is:

$$C_i = F(B - B_i)\frac{1}{ln(q)} \cdot ln(\frac{i \cdot (1 - q^{\frac{1}{B_i}}) - 1}{(i-1) \cdot (1 - q^{\frac{1}{B_i}}) - 1})$$

for $i > 1$. If F is known, the optimal migration bandwidth B_i for step i can be derived analytically. Otherwise, B_i can be obtained using numerical methods. Now, for each step i we also have to consider ending the pre-copy phase. The cost of copying a single page during the copy phase is given by $\frac{F(0)}{B} = \frac{1}{B}$ since during the copy phase all available bandwidth is being fully utilized by copying and the service is unavailable. Thus $\min_i C_i \geq \frac{1}{B}$ is the step in which we stop the pre-copy phase, halt the service, and continue with the copy step. We will refer to the above algorithm as CALM-adaptive.

Theorem 1 *If the dirtying probability is uniform and independent, then the CALM-adaptive algorithm achieves the minimal possible cost for the migration.*

Proof 1 *It is easy to see that if $C_i \leq \frac{1}{B}$ then for every $j > i$ we also have $C_j \leq \frac{1}{B}$. This together with the computation of the optimal B_i proves the theorem.*

We compare the performance of CALM-adaptive to the algorithm by Clark et. al. [5] that serves as a basis of the XEN live migration implementation and will be referred from now on as the XEN-basic algorithm. As described in Section 2, XEN-basic algorithm has 4 parameters that needs to be determined: the minimum and maximum bandwidth to be used in the migration, the fixed bandwidth increment, and the stop condition. To the best of our knowledge, there is no automated way to find optimal values for these parameters and they need to be tuned manually. In our experiment we used the proposed values as described in [5]. Namely, the minimum bandwidth was set to 100MBits/sec, there is no limitation on the maximum bandwidth, the fixed increment was set to 50MBits/sec, and the stop condition was set to 256KB.

The comparison is done by simulating live migration using 3 different traces of page access. These traces were generated from a VMs with 1GB, 512MB and 256MB RAM respectively, each of them running Apache Tomcat with the load described in Section 3. We tested live migration with total available bandwidth capacities of 1Gbit/s, 512 Mbit/s, and 256 Mbit/s. In our evaluation we considered the overall migration cost, the overall migration time and duration of the downtime period. Table 2 presents the results of this study. We also show the performance of the CALM-fixed migration algorithm for reference.

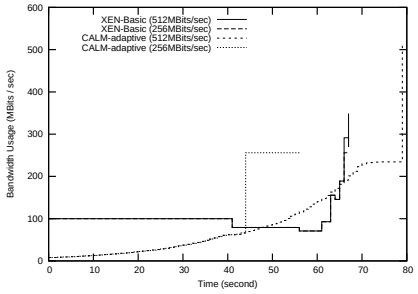
As one can see, performance of CALM-adaptive algorithm is significantly better in terms of cost under all configurations. It can also be noticed that when the link capacity is relatively low XEN-basic algorithm has a much shorter down-time but this comes at the cost of longer service degradation and a higher total cost.

In Figure 5 we compare the bandwidth usage of XEN-basic and CALM-adaptive algorithm throughout the migration process for a dirty trace generated from the VM

Table 2: Simulation Results

RAM	Bandwidth	Algorithm	Cost	Total Time (sec)	Down Time (sec)
1GB	1GBits/sec	XEN-basic	0.04	105.2	0.0424
		CALM-adaptive	0.01	85.78	0.0078
		CALM-fixed	0.01	19.93	0.0035
512MB	512MBits/sec	XEN-basic	0.27	67.99	0.0039
		CALM-adaptive	0.01	79.03	0.0078
		CALM-fixed	0.02	142.55	0.0069
512MB	256MBits/sec	XEN-basic	48.71	67.22	0.6412
		CALM-adaptive	12.05	56.58	11.940
		CALM-fixed	12.42	31.90	12.232
256MB	256MBits/sec	XEN-basic	36.86	40.38	2.1160
		CALM-adaptive	4.05	56.37	0.3940
		CALM-fixed	4.52	35.60	4.2325

Figure 5: Bandwidth Usage During Migration



with 512MB RAM, and two different link capacities. As one can see, XEN-basic algorithm behaves exactly the same (except when reaching the bandwidth limitation) regardless of the link capacity, while CALM-adaptive algorithm adjust itself according to the available bandwidth and the cost function.

6 Conclusions

Currently, the bulk of work on live migration performance evaluation focuses on the duration of the copy phase. While this is well founded for out-of-band migration, for the in-band use case, this measure of performance is insufficient since it does not account for performance degradation due to possible bandwidth contention. The concept of the cost function, introduced in this paper, makes it possible to evaluate the cost of the migration in terms of SLA violations. We introduced an efficient algorithm which, under some assumptions, performs a live migration with the minimal cost possible. We also provide a simpler algorithm for scenarios where the optimal algorithm is not applicable, and a fixed amount of bandwidth should be used. As our simulations show, our algorithms have a good chance of outperforming existing algorithms on realistic scenarios. A more extensive performance study with other realistic workloads is required to validate that point. The preliminary results presented in this work are encouraging, however. Many interesting points are left for further study. One is the ability to separate the pages to groups according to their dirtying probability and tune the algorithm to address each group separately. Another important factor is the change in the request rate. We assumed that this rate is fixed, but one can try to use periods in which the load is low, and increase the migration rate during these time periods.

References

[1] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Predicting the performance of virtual machine migration. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, 0:37–46, 2010.

[2] A.C. Barbosa, J. Sauvé, W. Cirne, and M. Carelli. Evaluating architectures for independently auditing service level agreements. *Future Generation Computer Systems*, 22(7):721–731, 2006.

[3] F. Checconi, T. Cucinotta, and M. Stein. Real-Time Issues in Live Migration of Virtual Machines. In *Euro-Par 2009–Parallel Processing Workshops*, pages 454–466. Springer, 2010.

[4] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, page 24. USENIX Association, 2005.

[5] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[6] T. Deshane, Z. Shepherd, J.N. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao. Quantitative comparison of Xen and KVM. *Xen Summit, Boston, MA, USA*, pages 1–2, 2008.

[7] The Apache Software Foundation. `http://incubator.apache.org/olio/`.

[8] The Apache Software Foundation. `http://incubator.apache.org/olio/the-workload.html`.

[9] Sun Microsystems. `http://faban.sunsource.net/`.

[10] Jacobus van der Merwe Timothy Wood, K.K. Ramakrishnan and Prashant Shenoy. CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. In *International Conference on Virtual Execution Environments (VEE)*, 2011.

[11] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation.

[12] M. Zhao and R.J. Figueiredo. Experimental study of virtual machine migration in support of reservation of cluster resources. In *Proceedings of the 3rd international workshop on Virtualization technology in distributed computing*, page 5. ACM, 2007.