IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Live Migration Modeling and Regression Analysis

Kutcherlapati Hima Bhargavi, G.M.Padmaja, K. Kanaka Sanjana, Kattamuri Sai Satya Sanjana, Nallala Anirudh.

Student, Assistant professor, Student, Student, Student

Computer Science and Engineering

Raghu Institute Of Technology,Visakhapatnam,India

Abstract –

One of the most important innovations for improving utilization and maintenance of the virtual machine during the live migration process to increase the power efficiency of data is one the main task which contains the high risk. Several live migration algorithms have been proposed, each with its own set of characteristics in by using completion time and other characteristics features related to live migration machine, virtual machine (VM) downtime, and VM performance degradation. Choosing the best live migration strategy has been a challenge so far, even with service-level agreements and organizational constraints in place. We propose Regression based Machine learning model that can predict key characteristics of live migration with high accuracy, depending on the migration algorithm and workload running within the VM. In contrast to previous research, we are not only able to model all widely used migration algorithms, but also significant metrics that have not been considered previously, such as VM performance degradation.

Keywords: Terms— live migration, Regression Techniques, virtualisation, Performance metrics

Problem Definition

The downtime and the total migration time are the two key parameters that quantify the performance of VM live-migration. These two quantities have a tendency to behave in opposite ways, so they must be carefully balanced. The cumulative migration time, on the other hand, measures the effect of the migration on the Cloud network infrastructure, while the downtime measures the impact on the end-perceived user's quality of service.

Proposed Approach

We use machine learning (ML) techniques to create a flexible model capable of accurately predicting key metrics of various live migration algorithms. The presented model predicts six main metrics of live migration (complete VM migration time, total amount of data transferred, VM downtime, VM performance degradation, and CPU and memory use on the physical hosts) with high accuracy given the resource usage of the physical hosts and the characteristics of the VM's workload. The model can be incorporated into established migration frameworks to determine the live migration algorithm is best for a VM migration.

- We illustrate that there is no such thing as a one-size-fits-all live migration algorithm. Using the 'correct' algorithm will help you save time and money by reducing resource waste and SLA violations.

- Here we implement a modeling approach for predicting key performance metrics of live migration algorithms for a specific virtual machine. The work presented here is currently the only method that can predict multiple target metrics in a scalable and automated manner for all widely used live migration algorithms.
- We show how incorporating the model into an existing live migration framework to automatically select the best live migration algorithm reduces the total number of SLA violations while also improving resource utilisation.

Algorithms Used

The point in time where the volatile state of the VM is copied to the destination host may be labelled as pre-copy, stop-and-copy, or resume (post-copy). Other algorithms are combinations of these three methods or concentrate on one

Introduction

The market for strategies for complex resource management in data centers has skyrocketed in the last decade. The aim is to reduce operating costs and environmental effects by minimizing energy consumption while optimizing hardware resource usage [30]. Virtualization [23] is a crucial technology for efficient data centre operation because it allows for better resource utilization by running multiple virtual machines on a single physical host. Virtual machines are live migrated [10, 20], that is, transferred from one physical host to another while the virtual machine (VM) is still operating, to adapt to fluctuating workloads and dynamically optimize resource usage. A live VM migration is an expensive process since it requires sending several gigabytes of volatile VM state from the source to the destination host. Several live migration algorithms have been proposed over the years [10, 19, 22, 28, 31, 24, and 27]. each algorithm has different performance characteristics that are dependent on the state of the host system, the interconnection network, and, to a greater extent, the workload running within the VM itself.

Choosing the best migration strategy based on operating policies, workload characteristics in the VM, the status of the involved hosts, and existing service-level agreements is a major challenge with major cloud platform Many companies are using virtualization strategies in their data centers [3, 16, 31]. (SLAs). There have been numerous attempts to model live migration efficiency [1, 13, 26, 27, 14, 11, 13], but analytical or simple probabilistic models do not achieve adequate prediction accuracy due to the various migration algorithms and large parameter space. We use machine learning (ML) techniques to create a flexible model capable of accurately predicting key metrics of various live migration algorithms. The presented model predicts six main metrics of live migration (complete VM migration time, total amount of data transferred, VM downtime, VM performance degradation, and CPU and memory use on the physical hosts) with high accuracy given the resource usage of the physical hosts and the characteristics of the VM's workload. The model can be incorporated into established migration frameworks to determine the live migration algorithm is best for a VM migration.

Literature survey

While current Cloud products include a number of options for managing multiple VMs running multi-tier applications [6], they do not support simultaneous VM live migration, do not account for possible future failures, or optimise the migration bandwidth allocated to each memory migration round, resulting in unnecessarily longer service interruption times. Following the pioneering work on live-migration [3,] a large number of implementations and research efforts focused on moving VMs with minimal service disruption [2], [7], and [9]. The majority of current work focuses on single VM migration; however, few solutions address the problem of migrating groups of similar VMs, such as those running multi-tier applications. VMFlockMS [10] focuses in particular on the migration of large VM disc images between data centres. This strategy, unlike ours, is primarily for non-live VM migrations, and the optimum allocation of inter-data centre link bandwidth is not taken into account. [11] The role of different resource reservation techniques and migration strategies on the live-migration of multiple VMs was evaluated experimentally. In both VM consolidation and dispersion studies, Kikuchi et al. [12] investigated the efficiency of concurrent live-migrations. These solutions do not account for the substantial impact of network resources on live-migration efficiency, unlike our approach, which considers an optimum bit-rate allocation. Other implementation-based experiments were performed to assess simultaneous live-migration under various assumptions and with various goals in mind. However, for each memory transfer round, they do not optimise bandwidth allocation. The only online algorithm designed for VM live migration that we are aware of was proposed in [17], where VM placement heuristics take into account server workload, VM performance degradation, and energy but do not account for network topology and bandwidth allocation for server interconnection, as we do in our approach. Their migration model also fails to account for memory dirtying rates.

aspect of a process. We simulate all five virtualization techniques provided by the major virtualization platforms.

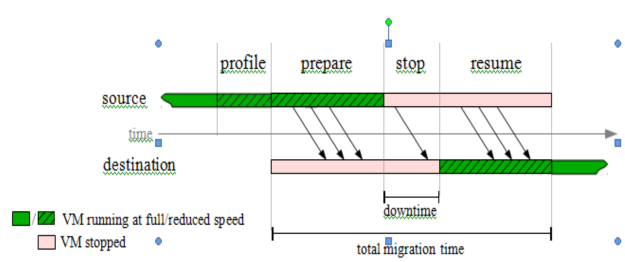

Fig 1:- **live migration** model

Support vector Regression

The SVR can be used as a regression tool while preserving all of the algorithm's key characteristics (maximal margin). With a few slight variations, the Support Vector Regression (SVR) uses the same rules for classification as the SVM. For instance, since production is a real number, predicting the information at hand, which has an infinite number of possibilities, becomes extremely difficult. In the case of regression, a tolerance margin (epsilon) is set as a rough approximation to the SVM that would have already been requested from the problem. However, there is another aspect to consider: the algorithm is more complex.

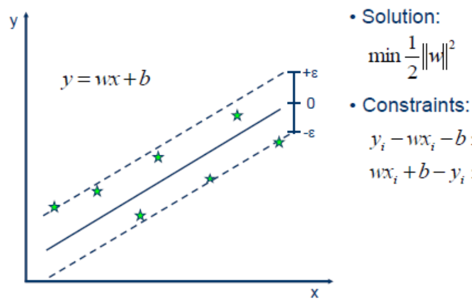

Fig 2:- **live migration** with SVR solution and constraint

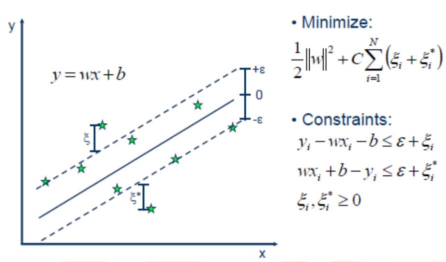

Fig 3:- **live migration** with SVR minimize constraint

SVR with Bagging

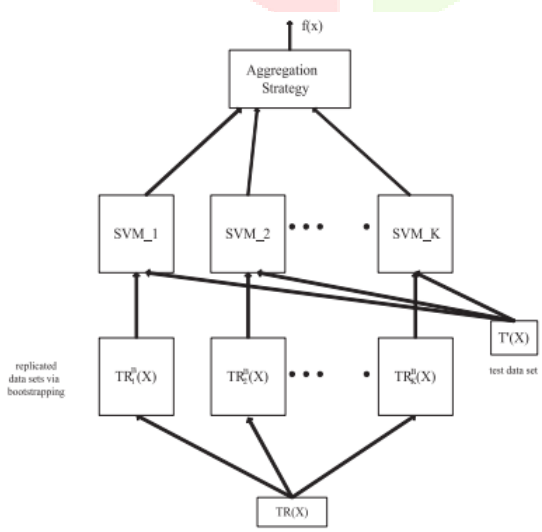

Fig 4: architecture of the SVR with Bagging

By randomly resembling, but with replacement, from the given training data set T R, bootstrapping generates K replicate training data sets T RB k |k = 1, 2,...,K. In any replicate training data set, each example xi from the given training set T R can appear multiple times or not at all. A different SVM will be trained for each replicate training package.

Random Forest Regression

Candidate split dimension A dimension along which a split may be made

Candidate split point One of the first m structure points to arrive in a leaf

Candidate split A combination of a candidate split dimension and a position along that dimension to split. These are formed by projecting each candidate split point into each candidate split dimension

Candidate children Each candidate split in a leaf induces two candidate children for that leaf. These are called as right and left split of child nodes

```
Algorithm 1 BuildTree
Require: Initially the tree has exactly one leaf (TreeRoot) which covers the whole space
Require: The dimensionality of the input, D. Parameters λ, m and τ.
  SelectCandidateSplitDimensions(TreeRoot, min(1 + Poisson(λ), D))
  for t = 1 ... do
    Receive (X_t, Y_t, I_t) from the environment
    A_t ← leaf containing X_t
    if I_t = estimation then
      UpdateEstimationStatistics(A_t, (X_t, Y_t))
      for all S ∈ CandidateSplits(A_t) do
        for all A ∈ CandidateChildren(S) do
          if X_t ∈ A then
            UpdateEstimationStatistics(A, (X_t, Y_t))
          end if
        end for
      end for
    else if I_t = structure then
      if A_t has fewer than m candidate split points then
        for all d ∈ CandidateSplitDimensions(A_t) do
          CreateCandidateSplit(A_t, d, π_d X_t)
        end for
      end if
      for all S ∈ CandidateSplits(A_t) do
        for all A ∈ CandidateChildren(S) do
          if X_t ∈ A then
            UpdateStructuralStatistics(A, (X_t, Y_t))
          end if
        end for
      end for
      if CanSplit(A_t) then
        if ShouldSplit(A_t) then
          Split(A_t)
        else if MustSplit(A_t) then
          Split(A_t)
        end if
      end if
    end if
  end for
```

Fig 5:-Random forest regression Algorithm

Results Analysis

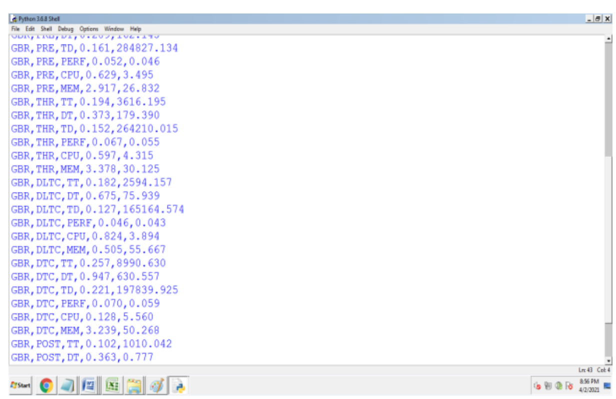

Fig **6**: Train the live migration model using Gradient Boosting Regression

Algo.	TT	DT	TD	PERF	CPU	MEM
PRE	0.99	0.99	0.99	0.24	0.18	0.69
THR	0.99	0.99	0.99	0.52	0.45	0.71
DLTC	0.97	0.89	0.97	0.18	0.30	0.78
DTC	0.97	0.99	0.99	0.25	0.70	0.65

Table1:- **Aggregated CoDs of the input features using gradient boosting**

Algorithm	Learning Time (s)	Prediction Time (ms)
Linear	8.0	0.74
SVR	239.0	5.11
SVR.Bagg	6617.7	188.63
RFR	6819.9	176.7
GBR	6987.5	198.6

Table 2: Learning and prediction overhead of the model for all classification

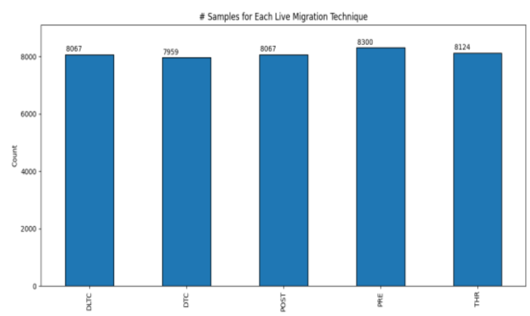

Fig **7 :-samples of live migrations**

	capability	workload_type	VM_size	...	Performance	SRC.CPU	SRC.MEM
0	POST	parsec	783.109375	...	0.751104	-50.083714	51.253906
1	POST	synthetic	1714.863281	...	0.415234	-3.184370	-55.632812
2	DTC	oltpbench	325.093750	...	0.555271	27.083333	77.238281
3	POST	synthetic	1615.085938	...	0.478092	-30.179749	-47.285156
4	DLTC	synthetic	995.621094	...	0.993469	1.638168	127.949219
5	DLTC	bzip	2038.121094	...	0.974335	4.552393	600.000000
6	DLTC	specweb	1512.390625	...	1.896274	-4.455539	8.183594
7	POST	synthetic	1459.621094	...	0.402481	-22.006000	80.437500
8	DTC	synthetic	977.519531	...	0.815634	79.640680	526.875000
9	DLTC	bzip	2036.808594	...	0.868442	-3.022056	-1.101562

Fig 8: The 20 input features of the ML model

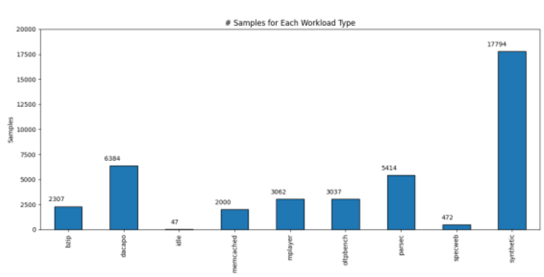

Fig 9:- features of work load type

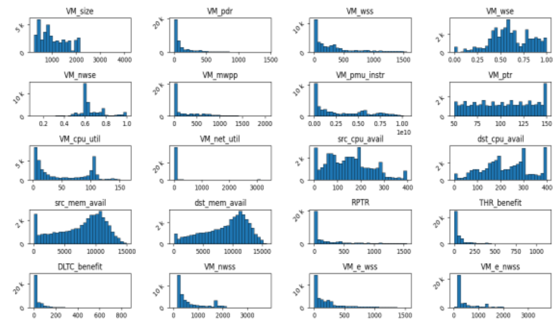

Fig **10 Impact of dataset size to the model accuracy**

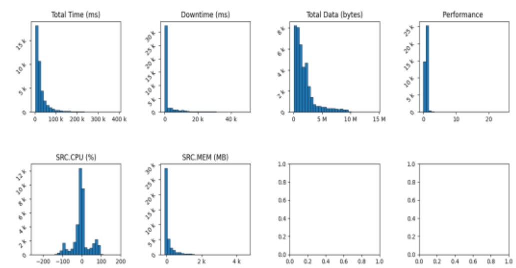

Fig 11:- total downtime and uptime with data analysis

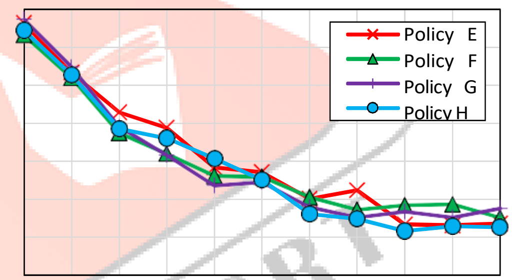

fig 12: error ration using policy

```
[ PRE] Training Time: 88.731s, Prediction Throughput: 10751 / sec
[ THR] Training Time: 86.780s, Prediction Throughput: 11177 / sec
[DLTC] Training Time: 85.756s, Prediction Throughput: 11157 / sec
[ DTC] Training Time: 88.548s, Prediction Throughput: 8072 / sec
[POST] Training Time: 86.215s, Prediction Throughput: 10449 / sec
```

fig 13: training time and prediction time accuracy

	capability	workload_type	VM_size	...	Performance	SRC.CPU	SRC.MEM
0	POST	parsec	783.109375	...	0.751104	-50.083714	51.253906
1	POST	synthetic	1714.863281	...	0.415234	-3.184370	-55.632812
2	DTC	oltpbench	325.093750	...	0.555271	27.083333	77.238281
3	POST	synthetic	1615.085938	...	0.478092	-30.179749	-47.285156
4	DLTC	synthetic	995.621094	...	0.993469	1.638168	127.949219
5	DLTC	bzip	2038.121094	...	0.974335	4.552393	600.000000
6	DLTC	specweb	1512.390625	...	1.896274	-4.455539	8.183594
7	POST	synthetic	1459.621094	...	0.402481	-22.006000	80.437500
8	DTC	synthetic	977.519531	...	0.815634	79.640680	526.875000
9	DLTC	bzip	2036.808594	...	0.868442	-3.022056	-1.101562

References

[1] A. Colin Cameron and Frank A.G. Windmeijer. 1997. *Journal of Econo- metrics* 77, 2 (1997), 329 – 342. https://doi.org/10.1016/S0304-4076(96)01818-0

[2] Ron C. Chiang, Jinho Hwang, H. Howie Huang, and Timothy Wood. 2014. Ma- trix: Achieving Predictable Virtual Machine Performance in the Clouds. In *11th International Conference on Autonomic Computing (ICAC 14)*. USENIX Asso- ciation, Philadelphia, PA, 45–56.

https://www.usenix.org/conference/icac14/ technical-sessions/presentation/chiang

[3] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Chris- tian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live Migration of Virtual Machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2 (NSDI'05)*. USENIX Association, Berkeley, CA, USA, 273–286. http://dl.acm.org/citation.cfm?id=1251203.1251223

[4] Christina Delimitrou and Christos Kozyrakis. 2013. QoS-Aware Scheduling in Heterogeneous Datacenters with Paragon. *ACM Trans. Comput. Syst.* 31, 4, Article 12 (Dec. 2013), 34 pages. https://doi.org/10.1145/2556583

[5] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware Cluster Management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 127–144. https://doi.org/10. 1145/2541940.2541941

[6] Li Deng, Hai Jin, Huacai Chen, and Song Wu. 2013. Migration Cost Aware Mitigating Hot Nodes in the Cloud. In *Proceedings of the 2013 International Conference on Cloud Computing and Big Data (CLOUDCOM-ASIA '13)*. IEEE Computer Society, Washington, DC, USA, 197–204. https://doi.org/10.1109/ CLOUDCOM-ASIA.2013.72

[7] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudre- Mauroux. 2013. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. *Proc. VLDB Endow.* 7, 4 (Dec. 2013), 277–288. https://doi.org/10.14778/ 2732240.2732246

[8] Jim Gao and Ratnesh Jamidar. 2014. Machine learning applications for data center optimization. *Google White Paper* (2014).

[9] Google Compute Engine 2017. https://cloud.google.com/compute. (2017). Online; accessed August 2017.

[10] Google Compute Engine uses Live Migration technology to service infrastructure without application downtime 2017. https://goo.gl/Ui3HFd. (2017). Online; accessed August 2017.

[11] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. 2009. Entropy: A Consolidation Manager for Clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*. ACM, New York, NY, USA, 41–50. https://doi.org/10. 1145/1508293.1508300

[12] Michael R. Hines and Kartik Gopalan. 2009. Post-copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*. ACM, New York, NY, USA, 51–60. https://doi.org/10.1145/1508293.1508301

[13] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, and Xiaodong Pan. 2009. Live vir- tual machine migration with adaptive, memory compression. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–10. https: //doi.org/10.1109/CLUSTR.2009.5289170

[14] Changyeon Jo and Bernhard Egger. 2013. Optimizing Live Migration for Vir- tual Desktop Clouds. In *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom '13)*, Vol. 1. 104–111. https://doi.org/10.1109/ CloudCom.2013.21

[15] Changyeon Jo, Erik Gustafsson, Jeongseok Son, and Bernhard Egger. 2013. Efficient Live Migration of Virtual Machines Using Shared Storage. In *Pro- ceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '13)*. ACM, New York, NY, USA, 41–50. https: //doi.org/10.1145/2451512.2451524

[16] Jonathan Koomey. 2011. Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times* 9 (2011).

[17] Sajib Kundu, Raju Rangaswami, Ajay Gulati, Ming Zhao, and Kaushik Dutta. 2012. Modeling Virtualized Applications Using Machine Learning Techniques. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE '12)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2151024.2151028

[18] Jianxin Li, Jieyu Zhao, Yi Li, Lei Cui, Bo Li, Lu Liu, and John Panneerselvam. 2014. iMIG: Toward an Adaptive Live Migration Method for KVM Virtual Machines. *Comput. J.* 58, 6 (2014), 1227. https://doi.org/10.1093/comjnl/bxu065

[19] Haikun Liu and Bingsheng He. 2015. VMbuddies: Coordinating Live Migration of Multi-Tier Applications in Cloud Environments. *IEEE Transactions on Parallel and Distributed Systems* 26, 4 (April 2015), 1192–1205. https://doi.org/10.1109/TPDS.2014.2316152

[20] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. 2013. Performance and energy modeling for live migration of virtual machines. *Cluster Computing* 16, 2 (2013), 249–264. https://doi.org/10.1007/s10586-011-0194-3

[21] Zhaobin Liu, Wenyu Qu, Weijiang Liu, and Keqiu Li. 2010. Xen Live Migration with Slowdown Scheduling Algorithm. In *Proceedings of the 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '10)*. IEEE Computer Society, Washington, DC, USA, 215–221. https: //doi.org/10.1109/PDCAT.2010.88

[22] Vijay Mann, Akanksha Gupta, Partha Dutta, Anilkumar Vishnoi, Parantapa Bhattacharya, Rishabh Poddar, and Aakash Iyer. 2012. *Remedy: Network-Aware Steady State VM Management for Data Centers*. Springer

Berlin Heidelberg, Berlin,
Heidelberg, 190–204. https://doi.org/10.1007/978-3-642-30045-5_15

[23] Memcached - a distributed memory object caching system 2017. https:// memcached.org/. (2017). Online; accessed August 2017.

[24] Microsoft Azure - Virtual Machines 2017. https://azure.microsoft.com/en-us/ services/virtual-machines/. (2017). Online; accessed August 2017.

[25] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo. 2012. Dynamic resource manage- ment using virtual machine migrations. *IEEE Communications Magazine* 50, 9 (September 2012), 34–40. https://doi.org/10.1109/MCOM.2012.6295709

[26] MPlayer - The Movie Player 2017. http://www.mplayerhq.hu/design7/news.html. (2017). Online; accessed August 2017.

[27] Senthil Nathan, Umesh Bellur, and Purushottam Kulkarni. 2015. Towards a Com- prehensive Performance Model of Virtual Machine Live Migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*. ACM, New York,
NY, USA, 288–301. https://doi.org/10.1145/2806777.2806838

[28] Senthil Nathan, Umesh Bellur, and Purushottam Kulkarni. 2016. On Selecting the Right Optimizations for Virtual Machine Migration. In *Proceedings of the12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '16)*. ACM, New York, NY, USA, 37–49. https://doi.org/10.1145/2892242.2892247

[29] Senthil Nathan, Purushottam Kulkarni, and Umesh Bellur. 2013. Resource Avail- ability Based Performance Benchmarking of Virtual Machine Migrations. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engi- neering (ICPE '13)*. ACM, New York, NY, USA, 387–398. https://doi.org/10.1145/ 2479871.2479932

[30] Michael Nelson, Beng-Hong Lim, and Greg Hutchins. 2005. Fast Transparent Migration for Virtual Machines. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '05)*. USENIX Association, Berkeley,
CA, USA, 25–25. http://dl.acm.org/citation.cfm?id=1247360.1247385

[31] Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Sethuraman Subbiah, and John Wilkes. 2013. AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. USENIX, San Jose, CA, 69–82. https://www.usenix.org/conference/icac13/ technical-sessions/presentation/nguyen