Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers

Kui Su, Lei Xu, Cong Chen, Wenzhi Chen, Zonghui Wang
College of Computer Science
Zhejiang University, Hangzhou, 310027, China
Email: {sukuias12, leixu, 21221240, chenwz, zjuzhwang }@zju.edu.cn

***Abstract*—Virtual machine placement (VMP) problem has been a key issue in IaaS/PaaS cloud infrastructures. Many recent works on VMP prove that inter-VM relations such as memory share, traffic dependency and resource competition should be seriously considered to save energy, increase the performance of infrastructure, reduce service level agreement violation rates and provide better administrative capabilities to the cloud provider. However, most existing works consider the inter-VM relations without taking the heterogeneity of cloud data centers into account. In practice, heterogeneous physical machines (PM) in a heterogeneous data center are often partitioned into logical groups for load balancing and specific services, cloud users always assigned their VMs with specific PM requirements, which make the inter-VM relations far more complex. In this paper, we propose an efficient solution for VMP with inter-VM relation constraints in a heterogeneous data center. The experimental results prove that our solution can efficiently solve the complex problem with an acceptable runtime.**

***Keywords*-Virtual machine placement; Affinity; Conflict; Heterogeneity; Cloud data centers;**

I. Introduction

With the popularity of cloud computing and the numerous benefits of virtualization technology, more and more companies, enterprises and organizations have shifted a significant part of their businesses from local physical servers to virtual machines (VMs) in data centers to provide cloud computing services. In a cloud data center, most distributed applications such as parallel computing applications[1] and multi-tier e-business web applications[2], are encapsulated within multiple VMs, the execution of these application jobs inside VMs generates a large amount of communications or data exchanges, besides, these VMs always run the same operating system and libraries that there may exist a lot of duplicate memory pages in these VMs[3]. The above dependencies between VMs are identified as affinity relation. While allocating given VMs on PMs in a cloud data center, VMs with affinity relation should be placed on the same physical machine (PM) to improve application performance and save resource[4–10]. On the other hand, there also exists conflict relation between VMs that these VMs should not be placed on the same PM. For example, if VMs running CPU-intensive applications are allocated on the same PM, it is easy to reach the CPU hotspot for the PM. Besides, VMs belonging to different users in a cloud may cause security risk such as cross-VM attacks[11] that they should be placed on different PMs for security[12–14]. Moreover, to improve fault tolerance and high availability for clouds, there may be a number of duplicate VMs for backups [15, 16], which should not be allocated on the same PM with the original VMs.

Many recent works on VMP have proved that the affinity and conflict relations between VMs should be seriously considered in order to provide quality and secure cloud services. However, few of them take the impacts of heterogeneity[17] in cloud data centers into account. Unlike more traditional application and organization-specific clusters, modern consolidated cloud environments are likely to be constructed from a variety of machine classes, representing different points in the con-figuration space of processing to memory to storage ratios. Several generations of machines, with different specifications, are likely to be encountered, as the underlying machine types evolve over time with respect to economically attractive price-performance. A subset of machines with specialized accelerators, such as graphics processors, may also be available in limited numbers. Finally, as the workload spans multiple organizations, it is likely to be inherently more diverse in its resource demands than one from any single organization. Should such a high degree of heterogeneity and variability in workload demand be encountered[18], it will significantly complicate inter-VM relations in VMP since heterogeneous physical machines (PM) in a heterogeneous data center are often partitioned into logical groups for load balancing and specific services, cloud users always assigned their VMs with specific PM requirements. Such as, some cloud users may require their VMs to be allocated on a PM group with high-end CPU and other users' VMs may require PMs with trusted computing hardware.

Addressing this problem, we propose a heuristic backtracking algorithm for VMP with inter-VM relations in heterogeneous data centers. In our experiments, we have collected data sets from a production cloud data center to validate the availability of our algorithm, the experimental results prove that our solution can efficiently solve the complex problem with an acceptable runtime.

978-1-4799-8261-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ISADS.2015.42

The paper is organized as follows. In Section II we define the VMP with complex inter-VM relations under heterogeneous environments. In Section III we describe our proposed solutions. The simulation results for our algorithm are given in Section IV. Finally, we describe related work and present our conclusions.

II. Problem Statement

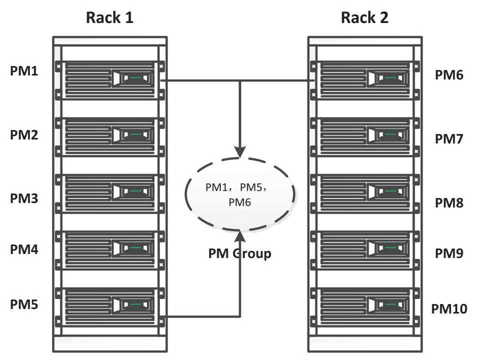

Figure 1: Relations between PMs in cloud data centers

In a cloud data center, PMs are distributed on a number of Racks, the network bandwidth within Racks are much better than that across Racks. Due to the heterogeneity, PMs are often partitioned into logical groups for specific applications. As Fig.1 shows, There exist four relations between PMs: same rack, different racks, same group and different groups. For the placement of VMs, as introduced in section I, VMs with affinity relations such as memory share or traffic dependency should be allocated on the same PM to save resource and improve performance, VMs with resource competition or belonging to different cloud users should not be allocated on the same PM for load balance and security. But,in the end these relations can be defined in the perspective of physical location as follows: (1), same PM (SP); (2), different PMs (DP); (3), same Rack (SR); (4), different Racks (DR); (5), same Group (SG); (6), different Group (DG).

Assuming the above scenario, we address a VM placement problem for cloud data centers in which VMs have resource demands, and the above six relations are taken as the affinity and conflict between VMs. Formally, the problem can be described as: Given a set of VMs, all the VMs have resource requirements such as (CPU, memory, bandwidth, etc.), a part of VMs have affinity requirements such as SP, SR and SG, a part of VMs with conflict have isolation requirements such as DP, DR and DG. Since the limited network bandwidth often becomes a bottleneck resource in modern data centers, network delay requirement and traffic value between communicating VMs are also assigned. We give an simple instance for our problem as follows:
1. Given VMs: $VM_1, VM_2, VM_3, VM_4, VM_5, VM_6$. Each of them is a vector of resource parameters: (CPU, memory, network and I/O).
2. Affinity requirements:
$(VM_1, VM_2, SP), (VM_1, VM_3, SP), (VM_4, VM_5, SR),$
(VM_3, VM_6, SG).
3. Isolation requirements:
$(VM_1, VM_4, DP), (VM_5, VM_6, DG)$.
4. Network-delay requirements: $(VM_1, VM_6, 110ms)$.
5. Traffic value: $(VM_3, VM_5, 850MB)$.

Our prior objective is to allocate these VMs on suitable PMs while meeting affinity, isolation and network delay requirements, the second objective is to minimize the total traffic across PMs by allocating VMs with large amount of traffic on the same PM or the same Rack. As same as traditional VM bin packing problem, we also consider to reach the above objectives by consuming as minimum active PMs as possible.

III. VM placement in heterogeneous data centers

VM placement has been explored for years and many excellent algorithms have been developed for it, most of existing algorithms are based on greedy algorithms. Greedy algorithms such as First Fit (FF), Best Fit (BF), Nest Fit (NF), First-Fit-Decreasing (FFD) can be used to solve most of VM placement problem and achieve an approximate-optimal solution. However, they are not suitable for our problem because of the multiple requirements and the heterogeneity in heterogeneous data centers. As a result, we need to develop a new algorithm for our problem.

Our solution can be divided into three steps: 1. pre-processing; 2. Grouping all the VMs based on their relations; 3.a group-based VM placement algorithm.

A. Pre-processing

Because of the hierarchical structure of physical topology in cloud data centers, network delay between different pairs of PMs is also distinct, we compute each PM of their network delay with other PMs and give a ascending order of them. In a cloud data center, we compute the network delay between PMs according to the distance between the switchboards they connect with. If two PMs connect the same switchboard, the network delay between them is zero.

Then, because the given affinity relation set is a binary relation, according to the directionality of affinity, we construct a relation table to find all the direct and indirect affinity relation. For example, if VM_1 and VM_2 have "SP" relation, VM_2 and VM_3 have "SP" relation, then VM_1 and VM_3 also have "SP" relation. But for conflict between VMs, the transitivity is not considered in our problem.

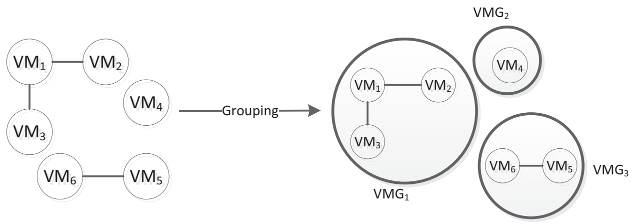

Figure 2: Virtual Machine Grouping

B. VM grouping

Before allocated onto PMs, all the VMs are divided into different groups according to their affinity and traffic relations that VMs with “SP” affinity or traffic relation will be taken as a group. For example, there is SP relation between VM_1 and VM_2, there is traffic relation between VM_1 and VM_3, VM_5 and VM_6 also have SP relation, VM_4 is irrelevant with other VMs, then VM_1, VM_2, VM_3 will be taken as a group, VM_5 and VM_6 are in a group, the last group contains only VM_4, each VM group will be allocated on one PM. Fig.2 gives an instance for VM grouping. The primary benefit of VM grouping is that if VMs with communication dependency or “SP” are bundled as whole unit and allocated onto the same PM, then we can minimize the communication overheads across physical networks and enhance application performance.

The grouping process finally generates a variety of VM groups. It should be noted that one VM group should be allocated on one PM thus the total resource requirements of one VM group must be less than the total capacity of this PM, besides, there may exist isolation requirements within certain VM groups. Therefore, it is necessary to further divide those unreasonable VM groups into smaller ones. In this step, we divide those groups using max-flow-min-cut theorem which can reduce the overall traffic across PMs .

C. Group-based VM placement

After VM grouping, all the VMs are divided into different groups that our problem is reduced to a group-based VM placement problem, we formulate it as a searching problem based on graph theory and design an iteratively heuristic searching (IHS) algorithm to achieve satisfactory results. Our algorithm is inspired by IDA*[19]. The details of IHS will be described as follows:

1. We construct a graph with all the VM groups based on their affinity and traffic relations. Affinity relation between VM groups is “SR” or “SG” and the traffic relation mainly results from the cutting process for the unreasonable VM groups in VM grouping.

2. To further simplify the problem, we search the VM-group graph with breadth-first search (BFS) algorithm and divide it into several subgraphs. For each subgraph, we iteratively search and place VMs in the subgraph on suitable PMs. For each iteration in a subgraph, we set a max value for the number of active PMs, if it fails to place VMs of the subgraph with the max PMs, we increase the max value and go on the next iteration, otherwise we go on to place another subgraph. To avoid meaningless searching and iterations, we set a threshold of runtime for each subgraph searching.

3. If certain subgraph is placed unsuccessfully , we search all the VM groups in the subgraph and find those groups which can be further cut into smaller ones and return to place the subgraph repeatedly. To avoid over-cutting and over-searching, we also set another runtime threshold for this process. The details of our algorithm are presented in Algorithm 1 and Algorithm 2.

Algorithm 1 VM PLACEMENT

```
1: RET⟵ false
2: while $RET \equiv false$ do
3:   MAX_AVAIL_PM ⟵ 1
4:   ret⟵ false
5:   for all $vmg \in V$ do
6:     if $vmg\ is\ placed$ then
7:       continue
8:     end if
9:     queue ⟵ $Get\_SubGraphs\_ByBFS(vmg)$
10:    while $ret \equiv false\ and\ time\ not\ runs\ out$ do
11:      ret⟵$RecursivePlace(0, queue)$
12:      if $ret \equiv false$ then
13:        MAX_AVAIL_PM ++
14:      end if
15:    end while
16:    if $time\ runs\ out$ then
17:      break
18:    end if
19:  end for
20:  for all $vmg \in V1$ do
21:    if $Cut(vmg) \equiv true$ then
22:      break
23:    end if
24:  end for
25:  if $time\ runs\ out$ then
26:    break
27:  end if
28: end while
```

As shown in the Algorithm 1, First we get all the VM groups related to the current VM group from the VM group sets V with BFS algorithm and append these related groups into a queue, then we place these VM groups on suitable PMs recursively. If a certain subgraph fails to be placed, we cut some lager VM groups of the subgraph into smaller ones and repeat to place this subgraph until time runs out.

As presented in the Algorithm 2, we check the current depth, if it is equal to the size of the queue (subgraph),

Algorithm 2 RecursivePlace

```
1:  if $depth \equiv queue\_size$ then
2:      return success;
3:  end if
4:  thisVMG $\longleftarrow$ queue_VMG[depth]
5:  total_requirement $\longleftarrow$ $Estimate\_PM\_Number(depth, queue\_VMG)$
6:  if $total\_requirement > MAX\_AVAIL\_PM$ then
7:      return place_failed
8:  end if
9:  for all $pm \in G$ do
10:     if $CheckRequirements(thisVMG, pm) \equiv success$ then
11:         Place this VMG on this PM.
12:     end if
13:     if $RecursivePlace(depth + 1, queue\_VMG) \equiv place\_success$ then
14:         return place_success
15:     else
16:         continue
17:     end if
18: end for
19: return place_failed
```

it means that all the VM groups in this queue are placed successfully, we return to Algorithm 1 and place the next subgraph. Otherwise, we begin to allocate the VM group in current depth of the queue on a suitable PM. To improve efficiency, we estimate the number of requiring PMs for all the VMs in our problem. This estimation take the well placed VMs into consideration and compute the total number of PMs with greedy algorithms in which only resource requirement are considered. Then we compare the PM number with the max value for PMs in this iteration. If it is greater than the max value, the rest searching steps of this iteration will be skipped. This estimation is a pruning to reduce unavailable searching steps and try to consume as minimum PMs as possible.

IV. Experiments and Evaluation

In this section, we first describe the characteristics of our data sets and then we simulate our algorithm on the data sets. At the same time, some comparisons are made between IHS and traditional greedy algorithms.

our data sets are collected from a production cloud data center and we independently and randomly sample 20, 50, 100, 150, 200 VMs from the data sets for many times and compute the average results. The data sets are similar with the sample introduced in section 3. We evaluate IHS in terms of three metrics: 1. average runtime for a successful VM placement; 2. total traffic across PMs; 3. total consumption of active PMs for VM placement.

Table I: Average runtime for a successful VM placement

VM Number	Average Runtime(ms)
20	414
50	505
100	580
150	662
200	731

A. Average Runtime

To prove the availability of our algorithm, we test the runtime for a successful VM placement. The runtime indicates that for given VMs, how long our algorithm will achieve an available solution. As shown in table 1, because of the pruning process in IHS, the average runtime for a successful VM placement in our simulation is less than one second which is acceptable for real cloud data centers.

B. Total traffic across PMs

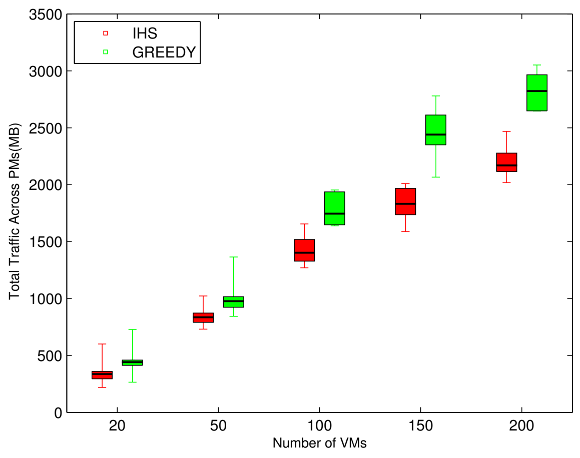

Figure 3: Total traffic across PMs

The second metric we have tested for IHS is the total traffic across PMs. In modern data centers especially for telecom cloud data centers, it has been found that the network bandwidth often becomes the bottleneck resource, causing both high network contention and reduced performance for communication and data-intensive applications. In our work, we try the best to allocate VMs with traffic relationship on the same PM to minimize the total traffic across PMs. As shown in Fig.3, its apparent that IHS reduces the total traffic across PMs comparing to greedy algorithms. Though greedy algorithms can also utilize the traffic between VMs, the affinity and conflict relations make it difficult to allocate most of VMs with traffic on the same PM.

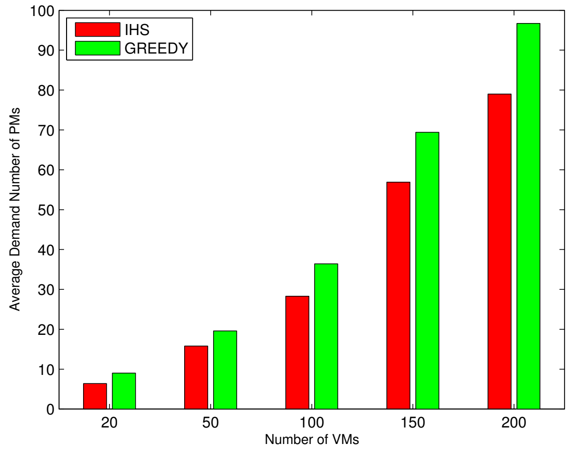

Figure 4: Total number of active PMs

C. Total number of PMs

For VM packing problem, the total number of active PMs is an important metric. Because the more PMs consume more power. Though in our problem, we consider not only resource requirements but also affinity and conflict relations between VMs, we still make an optimization to reduce the consumption of active PMs. In IHS, it utilizes a heuristic method based on the well placed VMs to estimate the optimal number of active PMs in each iteration. We can see from Fig.4 that IHS consumes fewer active PMs than greedy algorithms and as the problem scale increases, the superiority is more obvious.

V. Related Work

Affinity and conflict between VMs have been concerned in VM placement for years, cloud providers can significantly lower operational costs, and improve hosted application performance, by accounting for affinities and conflicts between co-placed virtual machines and various related algorithms for VM placement have been developed. We briefly introduce some of them in this section.

Affinity-aware algorithms: In virtualized data centers, a few studies have been done on affinity. In[7], Chen and Li employ affinity to implement a new schedule strategy to improve the efficiency of virtualized resource scheduling. The proposed affinity is used to identify the relation between a virtual CPU and a CPU in VMM or Hypervisor. [9]presents an affinity-aware VM migration technique to minimize the communication overhead on a virtualized platform. The affinity identifies a policy or a technique of VM migration for a dynamic resource allocation. [8]proposes a traffic-aware VM placement to improve the network scalability. To save memory, [5, 6] determine the sharing potential among a set of VMs and compute more efficient placements. Recently Sudevalayamet al.[20] attempt to evaluate performance of virtualized applications hosted among two VMs with co-location affinity. It focuses on performance evaluation but not resource allocation. Moreover, VMWare[21] uses affinity to signify the relationship between VMs which are kept together as one unit in VM placement. The above studies prove the practicality of our study on affinity-aware VM placement in cloud data centers.

Conflict-aware algorithms: Security is one of the top concerns in clouds. Indeed, cross-VM attacks are an important and real world threat. [12]presents the problem that scheduling algorithms can be used to place VMs on the same PM which leads to the risk of cross-VM attacks between adversary users in cloud environments. However, among existing VM placement algorithms there is only one which takes into account the notion of security, Zaina Afoulki al.[11] propose a security-aware scheduler that implements security policies expressing isolation requirements. The policies are expressed by the cloud users themselves by giving them the possibility to choose their own adversaries and they enforce these policies within the VM placement and migration algorithms.

As above-mentioned, many existing algorithms consider the affinity or conflict between VMs for VM placement, which really have brought great benefits for data centers. But none of them consider the both and the heterogeneity is also neglected, which is not suitable for modern cloud data centers. In our work, the main difference between our algorithm with the above studies is that we concurrently accounting for the affinity and conflict between VMs and propose an available algorithm for VM placement. Besides, we also take the heterogeneity such as PM grouping into account..

VI. Conclusion and Future Work

In this paper, we address VM placement problem in heterogeneous data centers while concurrently considering affinity and conflict between VMs. Firstly, we analysis the affinity and conflict between VMs in data centers and demonstrate the benefits of accounting for the affinity and conflict in the placement of VMs. Then we introduce the impact of cloud heterogeneity on inter-VM relations. Lastly, we propose IHS to solve the placement of VMs for heterogeneous data centers and simulate it on data sets from a production cloud data center. The results prove that our algorithm is actual available for real cloud data centers and the performance outperform traditional greedy algorithms.

In the future, we plan to further optimize the performance of our algorithm, compare it with more related algorithms and implement it in Openstack to validate our solution. We believe that our algorithm will be widely applied for VM placement in cloud data centers.

Acknowledgment

This work is supported by the National Science and Technology Major Project of the Ministry of Science and Technology of China under grant 2013ZX03003010-002.

References

[1] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi, "Introduction to the hpc challenge benchmark suite," 2005.

[2] X. Wang, Z. Du, Y. Chen, and S. Li, "Virtualization-based autonomic resource management for multi-tier web applications in shared data center," *Journal of Systems and Software*, vol. 81, no. 9, pp. 1591–1608, 2008.

[3] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference engine: Harnessing memory redundancy in virtual machines," *Commun. ACM*, vol. 53, no. 10, pp. 85–93, Oct. 2010. [Online]. Available: http://doi.acm.org/10.1145/1831407.1831429

[4] J. Sonnek and A. Chandra, "Virtual putty: Reshaping the physical footprint of virtual machines," *Proc of HotCloud*, 2009.

[5] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, and M. D. Corner, "Memory buddies: exploiting page sharing for smart colocation in virtualized data centers," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. ACM, 2009, pp. 31–40.

[6] M. Sindelar, R. K. Sitaraman, and P. Shenoy, "Sharing-aware algorithms for virtual machine colocation," in *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*. ACM, 2011, pp. 367–378.

[7] H. Chen, H. Jin, and K. Hu, "Affinity-aware proportional share scheduling for virtual machine system." in *GCC*, 2010, pp. 75–80.

[8] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[9] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," in *Parallel Processing (ICPP), 2010 39th International Conference on*. IEEE, 2010, pp. 228–237.

[10] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: accelerating resource allocation in virtualized environments," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1. ACM, 2012, pp. 423–436.

[11] Z. Afoulki, A. Bousquet, and J. Rouzaud-Cornabas, "A security-aware scheduler for virtual machines on iaas clouds," *Report 2011*.

[12] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[13] K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," *IT professional*, vol. 12, no. 5, pp. 20–27, 2010.

[14] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 693–702.

[15] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 161–174. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387601

[16] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "Snowflock: Rapid virtual machine cloning for cloud computing," in *Proceedings of the 4th ACM European Conference on Computer Systems*, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1519065.1519067

[17] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, 2012.

[18] ——, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 7.

[19] R. E. Korf, "Iterative-deepening-a: an optimal admissible tree search," in *Proceedings of the 9th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc., 1985, pp. 1034–1036.

[20] S. Sudevalayam and P. Kulkarni, "Affinity-aware modeling of cpu usage for provisioning virtualized applications," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 139–146.

[21] V. Infrastructure, "Resource management with vmware drs," *VMware Whitepaper*, 2006.