Resource management in *IaaS*-Clouds

Konstantinos Tsakalozos*

National and Kapodistrian University of Athens
Department of Infromatics and Telecommunications
k.tsakalozos@di.uoa.gr

Abstract. In this thesis, we present our work on four fundamental problems current *IaaS* clouds face: *a)* How consumers can assist the cloud middleware in resource management while the internal structure of the cloud is never revealed. *b)* The *VM*-to-*PM* placement in large non-homogeneous physical infrastructures. *c)* Load balancing though live *VM* migration under time constraints while not depleting the cloud resources. *d)* The maximization of the financial profit when using the cloud. The proposed solutions on these problems show significant improvements in the efficiency of *IaaS* clouds.

1 Introduction

In our everyday interaction with computing systems, we regularly use services that are classified as cloud enabled. Most of these cloud services are made available through the *Internet*, while others are restricted for use within organizations. The large variety and versatility of cloud services may discourage us from finding common properties among them, yet, they are all hosted by the "misty" cloud.

Concealing services inside clouds allows us to concentrate on how consumers are expected to use them. Consumers are kept agnostic to the details of *how* services are provided. In effect, users focus on what really matters to them, consuming a service. Similarly the cloud service providers focus only on aspects of their domain that are largely transparent to the end consumers. This separation of concern introduces a very elegant and convenient way for building services and interacting with them. It essentially sets the interface between service providers and service consumers.

Users contacting an *IaaS* cloud often request multiple *VM*s that collectively form an entire virtual infrastructure. Each *VM* is coupled with specific service level agreements (SLAs) regarding its availability and performance. Often these SLAs involve absolute values when they refer to space shared resources (e.g., RAM) and average values for time shared resources (e.g., network bandwidth). The reason for this is that due to the multi-tenancy, time-shared resources are offered to multiple users simultaneously and they are often over-committed in order to increase hardware efficiency. Most commercial cloud providers charge *VM* access depending on system characteristics and the SLAs offered.

* Dissertation Advisor: Alex Delis, Professor.

Internally, *IaaS* clouds produce *VM*s hosted on physical nodes. The dedicated software that undertakes the task of serving a user request is called *Cloud Middleware* [1–4]. The cloud middleware decides which physical machine (*PM*) is going to host the requested *VM*. All *PM*s are equipped with software capable of virtualizing hardware, that is made available to the customers. The cloud middleware also configures the network so that the provided *VM*s operate in isolation. The end result is that the user has a service that provides access to processing power upon request.

The cloud middleware monitors the overall performance of the cloud and may help facilitate the sharing of load across physical nodes. Hot nodes are cooled down by migrating *VM*s away from them. *VM* migration essentially changes the source of resources consumed by the *VM*. Migrations also serve administrative tasks that require certain parts of the physical infrastructure to be shutdown for maintenance reasons.

In this thesis, address four fundamental problems current *IaaS* clouds face: First, we show *how* consumers may assist the cloud middleware in its decision on where to place the *VM*s requested by the user. The challenge in this is that the internal structure of the cloud should never be revealed to the consumer. Our approach is based on user-provided hints that describe an ideal placement of the *VM*s to *PM*s. The cloud middleware tries to reach a *VM* placement that resembles the ideal. The second problem we discuss is how the *VM*-to-*PM* placement decision can take into account the heterogeneity of large physical infrastructures. The third problem is how load balancing though *VM* migration can take place under time constraints. We address this issue by combining a special purpose file-system with a high level resource allocation policy that oversees the progress of simultaneous *VM* migration tasks. Our goal is to migrate *VM*s under time-constraints while not completely depleting the cloud resources. The final problem we discuss is how to maximize the financial profit of a cloud whose actual performance may change overtime due to events the consumer is not aware of (e.g., sharing cloud resources with other consumers).

2 Related Work

The work in this thesis targets four distinct areas of *IaaS* clouds.

2.1 *VM* Placement Based On Constraints

The *VM*s-to-*PM*s placement policy in [5] exploits the tendency of *VM*s to have certain properties in common. [6] reformulates the problem as a multi-unit combinatorial auction. In [7], placement constraints are treated as separate dimensions in a multi-dimensional *Knapsack* problem. User and administrative preferences expressed through constraints are also employed in [8–11]. Often, as the number of constraints increases more resources are needed to solve the constraint satisfaction problem.

2.2 Workloads and System Consolidation

The allocation of resources in dynamic distributed environments [12] where load and resource availability change over time requires adaptive policies. In [13, 14], such resource sharing policies are proposed for the execution of jobs on the Grid.

Sandpiper [15] detects and monitors performance bottlenecks in a cluster hosting *VM*s. Two approaches are evaluated in the decision making mechanism that produces the *VM* migration actions: the first, termed *black-box*, remains fully OS-and-applications agnostic while the second, termed *gray-box*, exploits statistics originating from both the OS and the application-layer. Modeling the *VM* load is deemed important in the *black-box* approach used in IaaS clouds. [16–18] classify *VM* workloads and develop metrics to model the encountered workloads in an effort to reduce *VM* migration costs.

2.3 System Migration

Virtualization used in *IaaS*-clouds offers a simple, yet, powerful solution to load balancing. "Hot" *PM*s can be cooled down by having *VM*s moved elsewhere. Live migration [19] could potentially reduce the downtime of migrating *VM*s to the scale of milliseconds. Current *VM* Monitors (*VMM*s) [20–22] place a hard requirement for live migration as both the source and target *PM*s must share a storage layer. To address this requirement, small and medium-size clouds resort to the use of distributed file systems (*DFS*s). Red Hat's Global File System (*GFS*) [23] allows for several nodes to combine their storage resources. Each client accessing *GFS* must first acquire appropriate locks by contacting a set of master nodes. *GlusterFS* [24] is a user-space *DFS* in which every node can act both as a storage repository and/or a client. *Ceph* [25] uses separation between metadata and storage nodes and although scales well, its efficiency is hampered by saturated remote components such as switches.

An alternative proposal to the scalability issues of *DFS*s is the use of incremental transfer of virtual systems to different locations [26]. In [27, 28], live *VM* migration in WANs is advocated as a way to relieve overloaded physical nodes. In such environments, migration must handle challenges emanating from rerouting, significant latencies and low bandwidth rates. A workload-driven migration of virtual storage is discussed in [29].

2.4 The Finance of Highly Scalable, Elastic Infrastructures

Often in commodity selling markets, user demand reduces as resource price increases. This kind of user behavior is assumed to study the conditions that maximize the resource provider's profit [30]. Markets with more than one type of commodity (such as CPU, network, storage) are well-suited for use in multi-user computing clusters. Here, policies set the price of all sold resources considering their correlations [31]. Policies that take into account only high-level performance metrics, such as the application response time [32], seem more appealing to the end users since less intervention is required on their part.

By combining clouds with high resource availability [33] and a massively scalable programming framework [34], a user can process large amounts of data. *Dynamic Hadoop* [35] and Condor [36] can harvest the resources of large cloud installations and offer virtual elastic infrastructures on demand. These frameworks add a software layer that efficiently manages administrative concerns of scaling virtual infrastructures.

The work in this thesis addresses problems in the aforementioned areas and greatly improves the efficiency and features of *IaaS* clouds. The instantiation of complex virtual infrastructures is assisted by enhancements in the interaction between the cloud provider and customers [11, 37]. The latter are able to make use of cloud internal properties, such as high availability features, that are normally known only to the cloud administration. Through *VM* deployment hints the customers offer advice to the cloud's resource management, while the physical substrate remains concealed. Taking into account the provided hints, the cloud middleware can make intelligent resource allocation decisions not only during the initial instantiation of *VM*s, but also throughout the *VM*'s life-cycle. Well-informed resource appointment is displayed when the need for balancing load or maintenance tasks rises. Our work allows for the re-organization of virtual infrastructures, through *VM* migrations, under time-constraints [38]. By exploiting this feature the cloud middleware can schedule migrations in periods of low demand, thus customers can benefit from peak performance and reduced SLA failure rates. Furthermore, part of the enhanced interaction between providers and consumers is the cost efficiency of elastic infrastructures. We show how the elasticity of virtual infrastructures can be tuned to match the budget of their owners. As a result, we maximize the profit entailed in consuming cloud resources [39].

Since the success of cloud installations is based on economies of scale, we have put significant effort in promoting scalability. Our approach in matching the user needs with the available resources is computationally demanding. Yet, its design takes advantage of the distributed nature of the cloud and keeps up with the growth of the physical infrastructure [40]. In similar spirit, the *VM* migration facilities we propose, offer live migration in a true share nothing cloud architecture; thus we impose no limits to the size of the physical substrate. Overall, the scalability, our work offers, combined with the profit maximization methods we propose, can greatly benefit huge-data applications setup on the cloud.

3 Hint-based Execution of Workloads in Clouds with Nefeli

In our thesis, we present the design, implementation, and evaluation of a *cloud gateway*, *Nefeli*. *Nefeli* performs intelligent placement of *VM*s onto physical nodes by exploiting user-provided deployment *hints*. Hints realize placement preferences based on knowledge only the cloud consumer has regarding the intended

usage of the requested *VM*s. By modeling workloads as patterns of data flows, computations, control/synchronization points and necessary network connections, users can identify favorable *VM* layouts. These layouts translate to deployment hints. Such hints articulate *1)* resource consumption patterns among *VM*s, *2) VM*s that may become a performance bottleneck and *3)* portions of the requested virtual infrastructure that can be assisted by the existence of special hardware support. For instance, the fact that two *VM*s in a virtual infrastructure will hold mirrors of a database is only known to the cloud consumer. This information should be communicated to the cloud as a deployment hint so that the respective *VM*s will not be deployed on the same host. We refer to *VM* layout patterns as *task-flows* to distinguish them from the traditional workflow concept. Specifically, task-flows illustrate "ideal" deployments of *VM*s described by the cloud consumers using deployment hints. *Nefeli* exploits these hints so as to (re)deploy *VM*s in the cloud and achieve efficient task-flow execution. However, hints must not reveal any cloud internal properties to the consumers. Although hints may offer a desired *VM*-deployment for consumer workloads, *Nefeli* may ultimately elect to ignore part or all of them based on the available physical resources. In addition to hints, *Nefeli* also takes into account high-level *VM* placement policies, set by the cloud administration, whose objectives may entail energy efficiency and load balancing.

The main contribution of our approach is that we present a complete solution in extracting and exploiting the knowledge cloud consumers posses regarding the operational aspects of their virtual infrastructures. Our approach is compatible with the cloud abstractions that dictate users are kept agnostic of the physical infrastructure properties at all times. Furthermore, our approach is able to adapt to dynamic environments where both task-flows and user preferences change over time. *Nefeli* produces suitable *VM* to physical node mappings in response to signals coming from the infrastructures (both physical and virtual) or any other external notification mechanism. The produced mappings are applied through appropriate *VM* placement calls to an underlying cloud middleware.

We have created a detailed prototype and experimented with both simulated and real cloud environments. We compare *Nefeli VM*-placement against *a)* random placement, *b)* a placement that evenly distributes *VM*s among physical nodes, *c)* a policy that minimizes the number of physical nodes used and thus reduces the power footprint of the cloud, and *d)* the match making policy used by the open–source cloud middleware *OpenNebula v.1.2.0*. Our approach consistently displays significant performance improvements when compared to the aforementioned policies. In video transcoding, *Nefeli* achieves 17% reduced processing times compared to the *VM* placement decided by *OpenNebula*. In scientific task-flows and for a variety of simulated clouds, *Nefeli* demonstrates significantly higher throughput rates compared to other *VM* placement policies. Noteworthy savings in terms of power consumption are reported as well. We also present the performance overheads involved in the operation of *Nefeli* as the cloud infrastructure scales out.

4 Handling Non-homgenious Clouds

Infrastructure-as-a-Service cloud providers often face the following challenge: they must offer uniform access (resource provision) over a non-uniform hardware infrastructure. Non-homogeneous infrastructures may be the product of hardware upgrades, where old resources are left operational alongside new ones, or federated environments, where several parties are willing to share hardware resources with diverse characteristics.

In our work, we focus on the problem of instantiating entire virtual infrastructures in large non-homogeneous IaaS clouds. We introduce a service implementing a two-phase mechanism. In the first phase, we synthesize infrastructures out of existing promising physical machines (*PM*s). These dynamically-formed physical infrastructures, termed *cohorts*, host the user-requested *VM*s. In the second phase, we determine the final *VM*-to-*PM* mapping considering all low-level constraints arising from the particular user requests and special characteristics of the most promising selected *cohorts*. Compared to other constraint-based *VM* scheduling systems [8–10], the novelty of our approach mainly lies in the first phase. During this phase, besides resource availability, we also take into account properties such as migration capabilities, network bandwidth connectivity, and user-provided deployment hints. This helps prune out many possible *cohorts* within the cloud, and thus reduces the time required to produce a deployment plan in the second phase. We express both the selection of hosting nodes and the production of *VM*-to-*PM* mappings as constraint satisfaction problems and we use cloud-resources to solve these problems. We harvest these resources with the help of an elastic virtual infrastructure that can be dynamically enhanced with additional nodes depending on the needs of cloud users and the quality of *VM*-to-*PM* mappings cloud administrators want to provide. Our evaluation shows that this approach *1)* scales effectively for hundreds of *PM*s, *2)* reduces plan production time by up to a factor of 9, and *3)* improves plan quality by up to a factor of 4, when compared to a single-phase *VM* placement approach.

5 Time Constrained Live *VM* Migration in Share-Nothing *IaaS*-Clouds

Live migration requires that both the source and target *PM*s (used for hosting the *VM*) have access to the virtual disks of the *VM*. There are three alternatives in achieving this:

Single storage node: having a single common storage node that retains all virtual disk images is often impractical. This directly impacts the cloud scalability. Furthermore, such "single-node" solutions tend to become a performance bottleneck as well as a point of failure.

Distributed file systems: to tackle inefficiencies of a common storage node, clouds may employ distributed file systems such as *GFS* [23] and *GlusterFS* [24]. The latter allow for several *PM*s to collectively form a persistence layer, and thus exchange *VM*s and share load. The key objective of

distributed file systems is to present the *same* view of *all* files to *all PM*s at *all* times. Even though such file systems employ data replication and caching mechanisms they are often unable to comply with the requirements of large cloud installations regarding a *POSIX* API and low I/O latency.

Synchronization of virtual disks: as *VM*s could benefit from the exploitation of local resources including low-latency access to data and availability of RAID arrays, migration of *virtual disks on-demand* is an attractive choice to help re-arrange *VM*s [26, 28]. This approach is very different from that of distributed file systems as it attempts to place *VM*s on *PM*s on-the-fly implementing a scheduling policy of choice. On-demand synchronization entails the *VM* disk images that are being migrated, a target *PM*, and how the actual migration happens [27]. As there are no restrictions regarding the sizes (often multiple Gigabytes) of *VM*s to move, there are always significant overheads that have to be weighted against benefits to be reaped in the long term.

The feasibility of the on-demand synchronization has been established in prior efforts [26–28] and we use it as the foundation of our proposal. In our approach, we employ on-demand *virtual disk synchronization* to accommodate large numbers of migration tasks -involved in operations such as load balancing- across large cloud installations. The novelty of our approach is in the policy appointing resources to migrations. Our objective is to complete each migration within a designated time frame while not depleting the resources of the cloud. In effect, we attempt to achieve real-time load balancing and reduce the performance penalty that migrations inflict. In our approach, each migration task is paired with a *time-constraint* regarding its completion. By complying with their designated time-constraints, *VM*s do not display degraded performance due to migration in periods of high utilization. Our approach employs resource management features that help effectively synchronize virtual disk images across *PM*s. These low-level features predominantly deal with the consumption of *PM*-resources and are realized in the context of our *MigrateFS* file system. Instances of *MigrateFS* communicate over the network for moving virtual disk images between any two *PM*s. During the shipment, we are allowed to set: *a)* the disk bandwidth available to the *VM* internal processes accessing the virtual disks under migration, and *b)* the network bandwidth used for the purposes of migration. In this way, we are able to yield safe estimates on the exact time the migration will finish.

Our approach is weaved around a coordinating *Migrations Scheduler* and a distributed *network of Brokers*. The *scheduler* prioritizes migration tasks while the *Brokers* monitor the progress and appoint resources to each migration. The end-effect is that migrations finish within specified time-limits and "hot" physical network links are not further stressed by virtual disk shipments. The latter is achieved through constraints set by *MigrateFS* on the network bandwidth. Our evaluation, based on both a *MigrateFS* prototype and simulation of large infrastructures, shows up to 24% less stress on saturated *PM*s during migration at the expense of minimal administration effort.

6 Flexible Use of Cloud Resources through Profit Maximization and Price Discrimination

Cloud computing introduces a number of challenges regarding both performance and financial issues. On the one hand, consumers of cloud services try to minimize the execution time of their submitted tasks without exceeding a given budget and on the other, cloud providers are keen on maximizing their financial gain while keeping their customers satisfied. With this work we focus on virtual infrastructures following the master-worker paradigm hosted in a cloud. In this type of infrastructure, there is a master node that dispatches jobs to worker nodes and increasing performance is a matter of adding extra worker nodes. This ease of expansion has been put into use by a number of programming frameworks, such as MapReduce[34, 35], and resource allocation management tools, including Condor[36] and TORQUE[41].

Much of the previous work on master-worker architectures targets scalability bottlenecks. Such bottlenecks may develop due to two factors: first, each infrastructure has its own hardware limitations. Second, the data processing algorithms implemented by the jobs cannot always be efficiently parallelized and therefore are not suited for this type of distributed environment. Research on the aforementioned bottleneck factors has resulted in solutions that can efficiently harvest the hardware resources of infrastructures of any size. Experience in Grids has shown that the combination of a resource allocation tool with a programming library for distributed programming (e.g., MPI) can fully utilize small to medium computer clusters [36, 41]. For larger computing infrastructures, frameworks such as MapReduce are shown to display outstanding scalability[34]. Yet, purchasing and maintaining such large physical infrastructures involves a high investment risk. *Infrastructure as a Service* (*IaaS*) clouds have greatly reduced the investment risk of owning an infrastructure, but introduce a new performance scaling factor: the user's financial capacity to rent virtual resources. This additional factor complicates the deployment and management of cloud architectures.

The value for money spent in a multi-tenant, elastic cloud renders resource sharing policies a key player in both cloud performance and user satisfaction. These policies must take into account the fact that cloud providers need strategies to evaluate and reduce possible financial risks while maximizing their profit. In addition, resource sharing policies must take into consideration the user's budget. Such policies often employ either auctioning [42, 43] or attempt to estimate user demand for resources [31]. In this way, the consumer needs are quantified based on their willingness to pay for the resources available. Microeconomic-based resource sharing policies thus allow for the distributed computing infrastructure to reach an equilibrium where the quality of service provided reflects the money spent.

In our thesis, we propose a virtual-machine provision policy based on marginal cost and revenue functions. Each cloud customer announces her budget as a function of the execution time of the tasks she submits. Knowledge of this function, combined with the machine-hour cost, allows for educated decisions regarding

the amount of virtual resources allocated per customer in the context of an *IaaS-Cloud*. The main contribution of our approach is that we provide an answer to the question of exactly how many virtual machines (*VM*s) a consumer should request from a cloud within a budget. In light of scalable, master-worker-based, virtual infrastructures and the seemingly endless resources of a physical cloud, specifying the exact amount of resources needed must be based on a) the consumer's budget and b) the performance bottlenecks which are known only at runtime. We propose a mechanism that continuously monitors user application performance and either "removes" or "adds" *VM*s in response to the observed performance fluctuations serving the needs of autonomic systems [44]. Our approach automatically adjusts to the ever-changing equilibrium point caused by dynamic workloads and thus ensures that resources are shared proportionally to money spent by the users. In addition, our approach is applicable to a wide range of computational environments since it does not enforce the use of a specific job submission tool. We allow each user to select any virtual infrastructure equipped with the tools of her preference.

7 Conclusions

Cloud computing changes our perspective of the services we provide and consume. Cloud services are meant to be scalable, elastic and support a well defined business model. Infrastructure as a Service (*IaaS*) clouds include services that offer infrastructures on-demand. *IaaS* clouds build their success on virtualization so as to over-commit hardware resources and to take advantage of economies of scale.

Often *IaaS* clouds are seen as the evolution of the computing clusters. In both clouds and clusters the goal is the same: the provision of distributed resources. What has changed with the advent of the clouds is the resource provisioning mechanisms and abstractions. Users contact an *IaaS* cloud in search of processing power offered in the form of virtual machines (*VM*s). *VM*s have different specifications that also define their performance. Cloud service consumers have to choose the *VM* that will satisfy their need for computational power. Resource allocation policies map the *VM*s to the physical nodes.

Resource management within the physical infrastructure needs to be revisited as there are several properties that differentiate *IaaS* clouds from other distributed infrastructures. The physical infrastructure is never revealed to the consumers of the services. Current cloud abstractions do not allow cloud consumers to assist in resource management. This allows clouds to transparently support multi-tenancy and offer service scaling. Furthermore, cloud internal tasks such as migration of virtual resources are entirely concealed from the customers. In effect, users are not aware of the fact that the resources/system they are using are virtual, thus they are not aware of any on-going *VM* migration operations.

In the future, we will attempt to have some of the ideas described in this thesis incorporated into commercial and open-source cloud software. This will require us to examen the effectiveness and robustness of available constraint satisfaction

solvers and realize standard interfaces. Access to a variety of cloud installations will reveal several opportunities for improvement in both our implementation and our approach to certain cloud aspects. We will populate the set of deployment hints and we will attempt to form cohorts in more effective ways. Large cloud installations will allow us to deploy our approach regarding time-constrained migrations and evaluate several resource management policies. Finally, we will have the chance to evaluate alternative profit maximization models and test them in *PaaS* clouds.

References

1. OpenStack: http://www.openstack.org (May 2012)
2. OpenNebula: http://www.opennebula.org (May 2012)
3. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The Eucalyptus Open-Source Cloud-Computing System. In: 9th IEEE/ACM Int. Symposium on Cluster Computing and the Grid (CCGRID), Shanghai, China (May 2009) 124–131
4. VMware: VMware hypervisor. http://www.vmware.com/ (2011)
5. Sindelar, M., Sitaraman, R.K., Shenoy, P.: Sharing-Aware Algorithms for Virtual Machine Colocation. In: Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, California, USA (June 2011)
6. Breitgand, D., Epstein, A.: SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds. In: IFIP/IEEE International Symposium on Integrated Network Managemen, Dublin, Ireland (May 2011)
7. Singh, A., Korupolu, M., Mohapatra, D.: Server-Storage Virtualization: Integration and Load Balancing in Data Centers. In: Proc. of the 2008 ACM/IEEE conference on Supercomputing (SC'08), Austin, TX (2008) 53:1–53:12
8. Hermenier, F., Lorca, X., Menaud, J., Muller, G., Lawall, J.: Entropy: a Consolidation Manager for Clusters. In: Proc. of the 2009 ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments, Washington, DC (March 2009)
9. Hyser, C., McKee, B., Gardner, R., Watson, B.J.: Autonomic Virtual Machine Placement in the Data Center. http://www.hpl.hp.com/techreports/2007/HPL-2007-189.html (2008)
10. Wang, X., Lan, D., Wang, G., Fang, X., Ye, M., Chen, Y., Q.B. Wang, Q.: Appliance-Based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center. In: Proc. of the 4th Int. Conf. on Autonomic Computing, Washington, DC (2007) 29
11. Tsakalozos, K., Roussopoulos, M., Floros, V., Delis, A.: Nefeli: Hint-based Execution of Workloads in Clouds . In: Proc. of the 30th IEEE Int. Conf. on Distributed Computing Sytems (ICDCS'10), Genoa, Italy (June 2010)
12. Keahey, K., Tsugawa, M., Matsunaga, A., Fortes, J.: Sky Computing. IEEE Internet Computing **13** (September 2009)
13. Lee, K., Paton, N., Sakellariou, R., Deelman, E., Fernandes, A., Mehta, G.: Adaptive Workflow Processing and Execution in Pegasus. In: Proc. of 3rd IEEE Int. Conf. on Grid and Pervasive Computing Workshops, Kunming, PR China (2008) 99–106
14. Lee, K., Paton, N., Sakellariou, R., Fernandes, A.: Utility Driven Adaptive Workflow Execution. In: Proc. of 9th IEEE/ACM Int. Symposium on Cluster Computing and the Grid, Shanghai, PR China (2009) 220–227

15. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and Gray-box Strategies for Virtual Machine Migration. In: Proc of the 4th USENIX Symposium on Networked Systems Design and Implementation, Cambridge, MA (2007)
16. Khanna, G., Beaty, K., Kar, G., Kochut, A.: Application Performance Management in Virtualized Server Environments. In: Proc of the 10th IEEE/IFIP Network Operations and Management Symposium, Vancouver, Canada (April 2006)
17. Bobroff, N., Kochut, A., Beaty, K.: Dynamic Placement of Virtual Machines for Managing SLA Violations. In: Proc of the 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany (May 2007)
18. Weng, C., Li, M., Wang, Z., Lu, X.: Automatic Performance Tuning for the Virtualized Cluster System. In: Proc. of the 29th IEEE International Conference on Distributed Computing Systems, Montreal, Canada (June 2009)
19. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live Migration of Virtual Machines. In: Proc. of the 2nd Symposium on Networked Systems Design & Implementation - Volume 2. NSDI'05, Boston, MA, USENIX Association (May 2005)
20. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T.L., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: SOSP, New York, NY (2003) 164–177
21. Kernel Based Virtual Machine: http://www.linux-kvm.org (May 2012)
22. VMware: VMware ESXi hypervisor. http://www.vmware.com/products/vsphere/esxi-and-esx/index.html (May 2012)
23. Red Hat: Global File System. http://www.redhat.com/gfs/ (May 2012)
24. Red Hat: GlusterFS. http://www.gluster.org/ (May 2012)
25. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: A Scalable, High-Performance Distributed File System. In: OSDI'06, Seattle, WA (Nov. 2006) 307–320
26. Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y., Chen, H.: Live and Incremental Whole-System Migration of Virtual Machines Using Block-Bitmap. In: Cluster Computing, 2008 IEEE International Conference on, Tsukuba, Japan (Sept. 2008)
27. Bradford, R., Kotsovinos, E., Feldmann, A., Schiberg, H.: Live Wide-Area Migration of Virtual Machines Including Local Persistent State. In: In VEE 07: Proc. of the 3rd Int. Conf. on Virtual Execution Environments, San Diego, CA (June 2007)
28. Wood, T., Ramakrishnan, K.K., Shenoy, P., van der Merwe, J.: CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. SIGPLAN Not. (March 2011) 121–132
29. Zheng, J., Ng, T.S.E., Sripanidkulchai, K.: Workload-Aware Live Storage Migration for Clouds. In: Proc. of the 7th ACM Int. Conf. on Virtual Execution Environments. (VEE '11), Newport Beach, CA (2011)
30. Marbukh, V., Mills, K.: Demand Pricing & Resource Allocation in Market-Based Compute Grids: A Model and Initial Results. In: ICN '08: Proceedings of the Seventh International Conference on Networking, Cancun, Mexico, IEEE Computer Society (Apr. 2008) 752–757
31. Subramoniam, K., Maheswaran, M., Toulouse, M.: Towards a Micro-Economic Model for Resource Allocation. In: In IEEE Canadian Conference on Electrical and Computer Engineering, IEEE Press (2002) 782–785
32. Volker, C.E., Hamscher, V., Yahyapour, R.: Economic Scheduling in Grid Computing. In: JSSPP '02: Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, London, UK, Springer (2002) 128–152

33. Amazon: Elastic Cloud. http://aws.amazon.com/ec2/ (2010)
34. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA (Dec. 2004)
35. HP Labs: Dynamic Hadoop Clusters. http://wiki.smartfrog.org/ /wiki/display/sf/Dynamic+Hadoop+Clusters (Feb. 2010)
36. University of Wisconsin Madison: Condor. http://www.cs.wisc.edu/condor/ (Nov. 2010)
37. Tsakalozos, K., Roussopoulos, M., Delis, A.: Hint-based Execution of Workloads in Clouds with Nefeli. In IEEE Transactions on Parallel and Distributed Systems, Under Minor Revision (2012)
38. Tsakalozos, K., Roussopoulos, M., Delis, A.: Time Constrained Live VM Migration in Share-Nothing IaaS-Clouds VM Placement in non-Homogeneous IaaS-Clouds. Under Submission (2012)
39. Tsakalozos, K., Kllapi, H., Sitaridi, E., Roussopoulos, M., Paparas, D., Delis, A.: Flexible Use of Cloud Resources through Profit Maximization and Price Discrimination. In: Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE'11), Hannover, Germany (Apr. 2011)
40. Tsakalozos, K., Roussopoulos, M., Delis, A.: VM Placement in non-Homogeneous IaaS-Clouds. In: Proc. of the 9th Int. Conf. on Service Oriented Computing, Paphos, Cyprus (Dec. 2011)
41. http://www.clusterresources.com/pages/products/torque-resource manager.php: TORQUE Resource Manager (2010)
42. Grosu, D., Das, A.: Auctioning resources in Grids: model and protocols: Research Articles. Concurrent Computation : Practice and Experience **18**(15) (2006) 1909–1927
43. Chen, C., Maheswaran, M., Toulouse, M.: Supporting Co-allocation in an Auctioning-based Resource Allocator for Grid Systems. In: Proc. of the International Parallel and Distributed Processing Symposium, Fort Lauderdale, Florida, USA (Apr. 2002) 89–96
44. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36**(1) (2003) 41–50