A SLA-Based Method for Big-Data Transfers with Multi-Criteria Optimization Constraints for IaaS

Mihaela-Catalina Nita, Cristian Chilipirea
Faculty of Automatic Control and Computers
University *Politehnica* of Bucharest
Bucharest, Romania
{mihaela.nita, cristian.chilipirea}@cti.pub.ro

Ciprian Dobre, Florin Pop
Faculty of Automatic Control and Computers
University *Politehnica* of Bucharest
Bucharest, Romania
{ciprian.dobre, florin.pop}@cs.pub.ro

***Abstract*—When one pays for a Cloud Service he wants the Service to be compliant with standards and to respect his needs (respect the Service Level Agreement). When manipulating big-data (like picture collections, satellite images or digital libraries), transfers must be optimized. The Cloud Infrastructure itself utilizes big-data transfers while migrating Virtual Machines inside an Infrastructure as a Service system. In this paper we discuss data transfers in the cloud, which affects performance in the case of Virtual Machine migration and of user submitted big-data transfers, by moving resources where they are needed. We suggest a scheduling policy and offer two greedy scheduling algorithms that minimize individual transfer times. This is mostly important for being able to have a High Performance Computing environment inside the cloud, for scenarios in which individual data transfer times need to be but when the data is transferred is not an issue. We present an empirical validation based on simulation experiments.**

Keywords—Data Transfer, SLA, Optimization, IaaS, Cloud Computing, Scheduling

I. INTRODUCTION

Cloud Computing has brought a new paradigm in Computer Science: one in which, for the first time, processing power is infinite. Users can use the elasticity of the cloud to compute difficult problems without the need for extensive hardware setup. But this elasticity comes with a cost: Virtual Machines need to be created, migrated or replicated; Back-ups need to be created on the fly. All of these operations take certain amounts of time, depending on the infrastructure the cloud provider has to offer, and the Service Level Agreement constraints, which cannot be ignored.

In this paper we discuss big-data transfers in the cloud. They affect performance in the case of user submitted data transfers, by moving resources to the virtual machine on which they are needed and in the migration of Virtual Machine migration.

An efficient data transfer mechanism will provide a High Performance Computing environment inside the cloud and it will provide more transparency to the user in the migration process of a Virtual Machine from one part of the cloud to the other. How Virtual Machine migration, copy and saving its current state affects performance is studied in [1]. It has been shown that these operations have a big effect on the performance and on the amount of time in which a virtual machine stays inactive.

With the growth in storage capacity and processing power, we can predict that Moore Law will continue to be obeyed and that Virtual Machine size inside the cloud will grow. This makes Virtual Machine migration even more difficult, by requiring bigger amounts of bandwidth and by raising the amount of downtime while a migration is in progress. This problem is even more important in the case of hybrid Clouds where the interconnection between individual clouds tends to have smaller amounts of bandwidth and a more unreliable connection.

Virtual Machine migration, Virtual Machine deployment and Big-data transfers in which the data needs to be transferred as fast as possible, but when the data is transferred is not relevant, raise the need for a policy where individual transfer time needs to be minimized, even at the cost of total transfer time.

For this we offer a greedy scheduling algorithm that will minimize the transfer duration for individual transfers, inside a hybrid cloud topology. To improve on this initial algorithm we add a priority, given throw the SLA, to each transfer that needs to be executed and modify the algorithm to account for this priority.

We test the algorithms inside a simulation environment with a simulator we implemented for this particular problem. We describe the simulator in the following sections.

II. VIRTUAL MACHINE MIGRATION INSIDE A HYBRID CLOUD

In the context of big-data transfers, a few big questions need to be answered in order to have an efficient cloud environment: when and where to migrate. In this context, an efficient data migration method, focusing on the minimum global time, is presented in [2]. The method however does not try to minimize individual migrations duration. In [3] is proposed a system that detects the need of VM migration, in the context of a SLA violation. The system is called Sandpiper and it also estimates the amount of additional resources needed. In [4] two migrations models are described: offline and online. The offline scheduling model has as main target the minimization of the maximum bandwidth usage on all links for all time slots of a planning period. In the online scheduling model, the scheduler has to make fast decisions and the migrations are revealed to the migration scheduler in an a priori undefined sequence. *Jung et al.* treats in [5] the data

mining parallelization by considering the data transfer delay between two computing nodes. This delay is estimated by using the auto-regressive moving average filter. In [6], the impacts of two resource reservation methods are tested: reservation in source machine and reservation in target machine. Experimental results proved that the resource reservation in target machines is needed, in order to avoid the migration failure, and the performance overheads of live migration are affected by memory size, CPU, and workload types.

None of these papers tries to minimize the duration of individual virtual machine migration. The time it takes to suspend the virtual machine, to transfer its data, which can have tens of GBs, and to start the virtual machine may bring a significant downtime to the clients system. If the virtual machine size grows this problem becomes even more pressing.

We estimate that scheduling data transfers, so that they do not overlap, will reduce individual transfers duration and thus will shorten downtimes experienced by the end user. In the following sections we will prove these concepts.

III. Experimental Scenarios

A Cloud offers to the end user the needed processing power, using its dynamically allocation mechanisms of resources to match the user's needs. The distribution of CPU, memory, bandwidth and storage is made in a transparent way for each user but cannot exceed the physical capabilities of the network. It also cannot offer smaller latency.

The power of balancing the resources over the cloud is a key feature when a High Performance Computing is implemented over a Cloud domain. Cloud providers can offer a fully redundant infrastructure for HPC application and simulate a full mesh distributed network. This is achieved by using a layered topology described in [4] and [7], where the physical hosts are connected to the first level, called "Edge Switch Level", typically through a 1Gb/s connection. The edge switches are connected to the core switches through a 10Gb/s, and the core switches to the routers with the same 10Gb/s connection. As a result of the high number of ports a switch can support this topology can create a full mesh connection with a large number of physical units. Multiple routers are used to connect to different areas of the Internet or to interconnect with other Clouds. This topology can be visualized in Fig. 1.

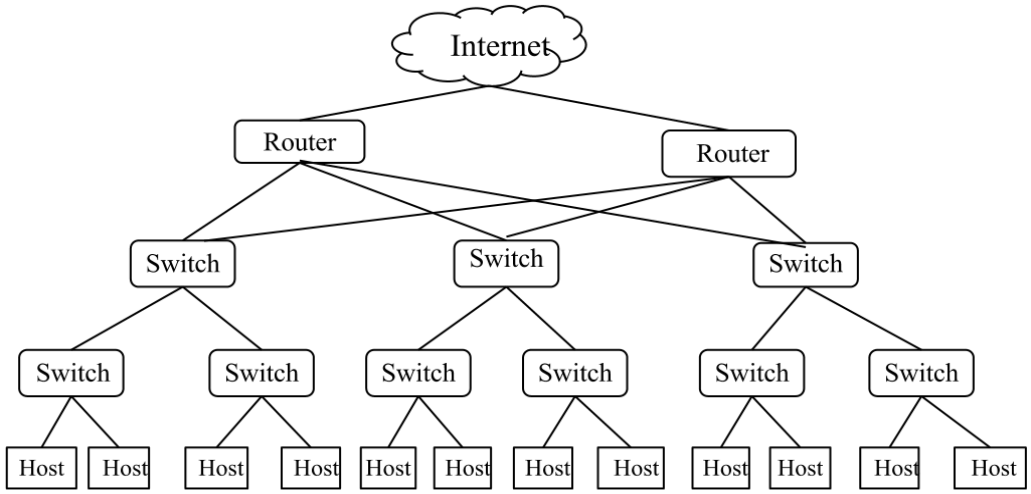

Fig. 1. Cloud Network Topology

Taking into consideration the described topology, our simulation is based on multiple Cloud domains with physical machines connected in a full mesh network. Looking at the whole map of domains, they are also connected directly or over the Internet in a full mesh topology. In our simulation we presume each cloud has three physical processing units and one routing unit, that is unable to create messages but can forward them. In the presented topology, four Clouds with the described infrastructure are interconnected. We presume an ideal environment and we do not take into consideration real life events, like temporary disconnects or bandwidth variation over time. Bandwidth and latency are constant over time and all connections have the same bandwidth. The network topology can be observed in Fig. 2. The physical machines have a number of messages with random destinations inside the hybrid cloud network. Messages are broken into several packets and the packets are scheduled for delivery. Over one connection two packets can be sent in every time unit, one in each direction.

The interactions (over Internet) between customers and providers increase in an exponential way, so the key to a profitable PaaS is user density per resources (physical or virtual machine). The scenario proposed in Fig. 1 highlights the model of Cloud System Networks. Scientific applications (addressing data intensive services or computationally intensive processing) follow the distributed computing paradigm (Cloud Computing) addressing different fields: medicine, physics, astronomy, chemistry, and even economy.

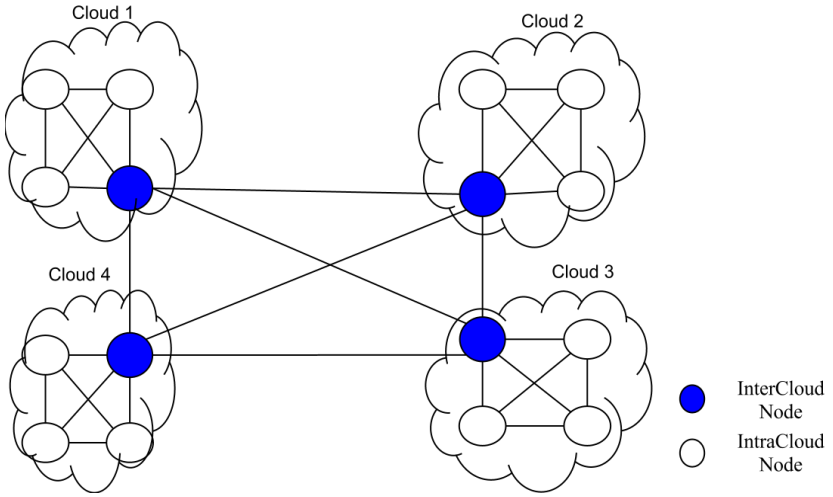

Fig. 2. The network topology

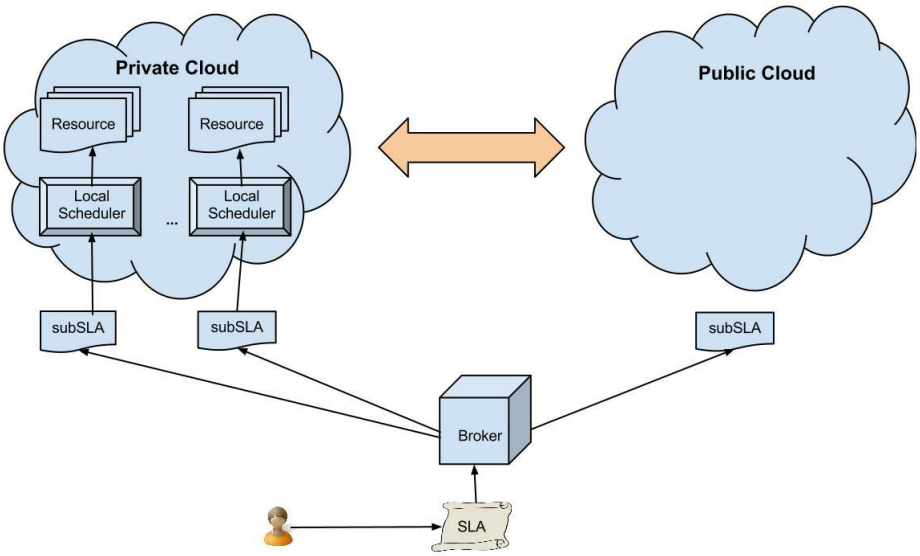

Fig. 3. General Schema

IV. SIMULATOR DESIGN

The general flow is: the customer makes a request for a certain service respecting a specific SLA; the broker receives the SLA and may decide to divide it in many sub SLAs. For example, the client will require three nodes, each with a specific CPU and Memory requirements. In this case, the broker will realize three others SLAs, for each node in part, and will try to schedule it on different hosts. The broker may schedule it in the private cloud or in the public one. The private and public clouds are connected with each other. Depending on where the task is scheduled Virtual Machines might need to be migrated and a substantial amount of time is spent in the migration process.

The user could also interact with the broker through an API and let it know Big-data transfers need to be done. The broker would then take the SLA into account and schedule the transfer as needed. This requires a minimum guaranteed bandwidth on all the connections so that the scheduling can be done correctly. We analyze the effects of big data transfers and virtual machine migration. The need to migrate and analyzing when a migration should be executed or a transfer is studied in [3].

To test our scheduling algorithm for big data transfers like virtual machine migration we have created our own simulator that has support for multiple clouds. Its schema is described in Fig. 4. The simulator goes through a few stages, it loads the node topology and then it generates messages. Each message

represents a Big-data transfer such as a Virtual Machine migration, it has a source and a destination, and other data may be added. The message is then taken by the simulator and split into packets and then ordered in an outgoing queue. The rest of the simulator deals with the actual transfers. The simulator takes into account bandwidth and any other needed statistics and generates logs at the end of each simulation.

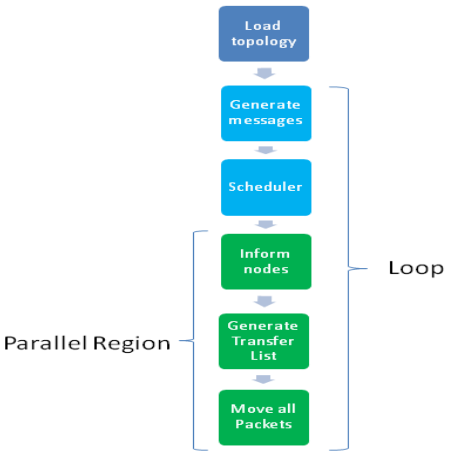

Fig. 4. Simulator design

It is important to mention that the modules in the figure are interchangeable, ergo if one wants to write a different scheduler, that individual module can be changed.

V. Greedy Scheduling With and Without Priority

We have implemented three scheduling algorithms inside our simulator. The first one is the simplest one: it does not perform any scheduling at all, it simply sends the packets in a first come first serve order. This is how standard networking devices work and how all transfers are treated in a real life environment. This algorithm sends the packet as it receives, so it should have a smaller arrival time for all packets than any other algorithm. Because of the way the packets can get mixed, individual message arrival time can be high.

Our first proposal is the greedy scheduling algorithm. This algorithm get the transfer requests in a first come first serve order and sets a time interval in which they can be sent. This interval is reserved on all the connections the packet has to go through (in our case there is a maximum of three hops to destination, because of the full mesh infrastructure). This is done until there are no more messages to schedule, taking into account the previously reserved time frames for each individual connection. It is important that we treat connections individually because some may prove to be a bottleneck. For instance the connections between individual clouds need to send more messages than connections inside the cloud. This way, even if the connection from a physical machine to a router is unused, the connection between the routers can be oversaturated. There is no point in scheduling the migration until the transfers that are currently running between the routers ends, even if the connection to the router is unused.

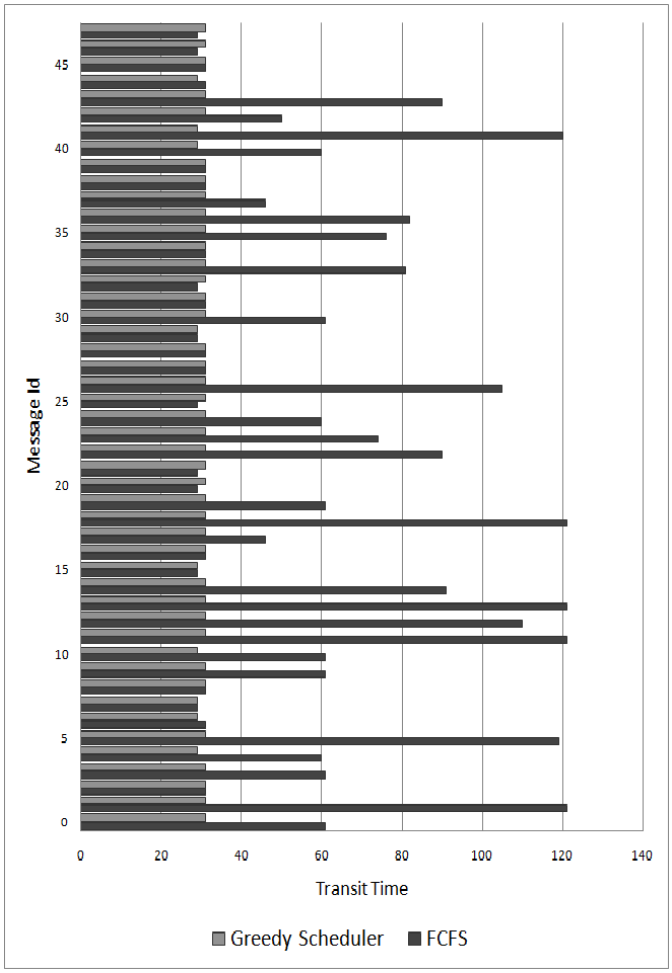

Fig. 5. Individual Transit Time comparison between Greedy and FCFS

We extended this algorithm by adding a priority marker to the messages, a priority that is set through the Service Level Agreement. It has values from 0 to 10, with higher values for better priorities. This priority is similar to the one some Cloud providers offer for CPU. The virtual machines have a higher priority when using the CPU and get to use it for longer periods of time. The messages are sorted according to the priority and afterwards scheduled. The bigger the priority, the earlier the packets will be scheduled.

VI. Simulation Results

We have simulated the algorithms that were described in the previous section. The first come first serve (FCFS) algorithm was used, as a comparison for the proposed greedy scheduling algorithm. This can be seen in Fig. 5, where we compared the transit time for individual packets in the two versions. We tested with a message size of 30 packets and 1 packet per each connection. It is clearly visible how the greedy scheduling transit time per message is about 30 time parts (it differs if the message is sent in the same cloud or in a different

cloud). The default method has bigger transit times for most of the messages. This proves that the scheduling mechanism works and provides the expected results.

In the next figure (Fig. 6), we compared the messages arrival time between the greedy scheduler and FCFS. The results are comparable. In some cases Greedy is slower, but the goal of this algorithm is to obtain smaller transit times and not arrival times. Other papers, like [2],[4] or [6] treat the problem of arrival time. Although we do not try in any way to minimize arrival times, the graph in figure 6 shows how the results are comparable and the proposed algorithm doesn't add significant latencies.

We also propose the greedy scheduling with priority, which acts the same way as the classical greedy but takes into account the message priority, priority given by the SLA. This priority is negotiated between the user and the cloud, the higher the priority is the higher the price paid for the service will be.

Fig. 7 compares the arrival times of a messages sequence, for Classical Greedy and Greedy with priority. By analyzing the trend lines, it can be observed that the Greedy Scheduling with Priority manages to deliver the messages with higher priority faster than the others.

In this context, Cloud computing has the potential to be the major solution for scalability, mobility, reliability, fault tolerance and security for business and academic environments in the same time.

As computing needs increase, multiple administrative domains will manage the resources in order to ensure the QoS and guarantee the SLA. Considering this, finding a way to a smart Cloud infrastructure will consider dependable management services to control the infrastructure and provide essential system-level functionality.

It is important to outline that the two algorithms only try to minimize the transit time. Both accomplished this goal and are best fitted for environments in which transition times need to be minimized and arrival times for most messages can be sacrificed.

Ergo the algorithms are best suited for virtual machine migrations and users big-data transfers that can be delayed but need to be executed in the shortest time possible. This happens when the user handles data that can be modified until the transfer starts.

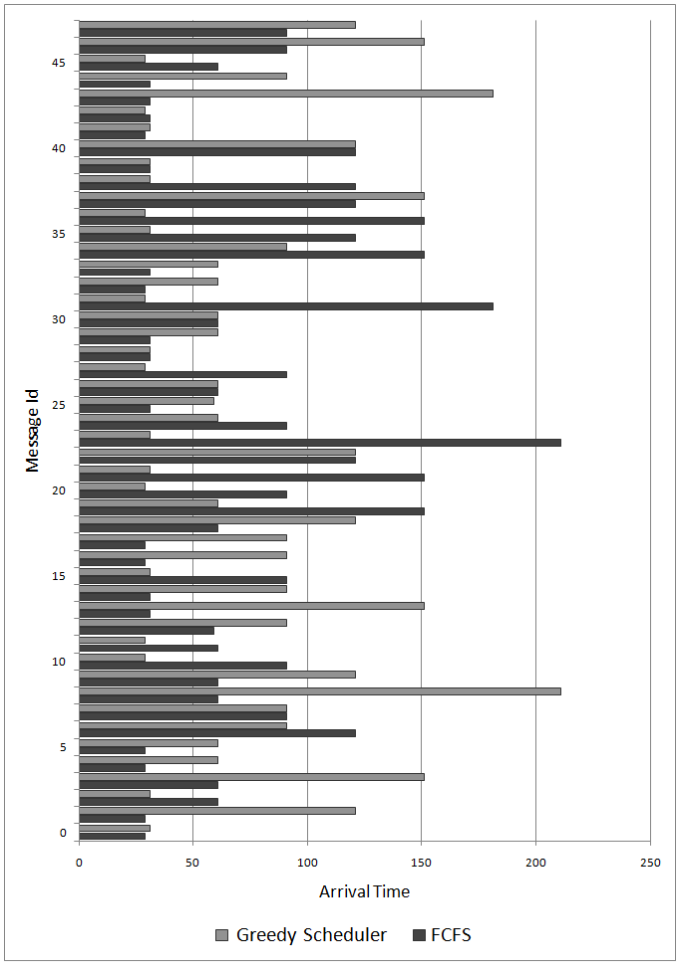

Fig. 6. Individual Arrival Time comparison between Greedy and FCFS

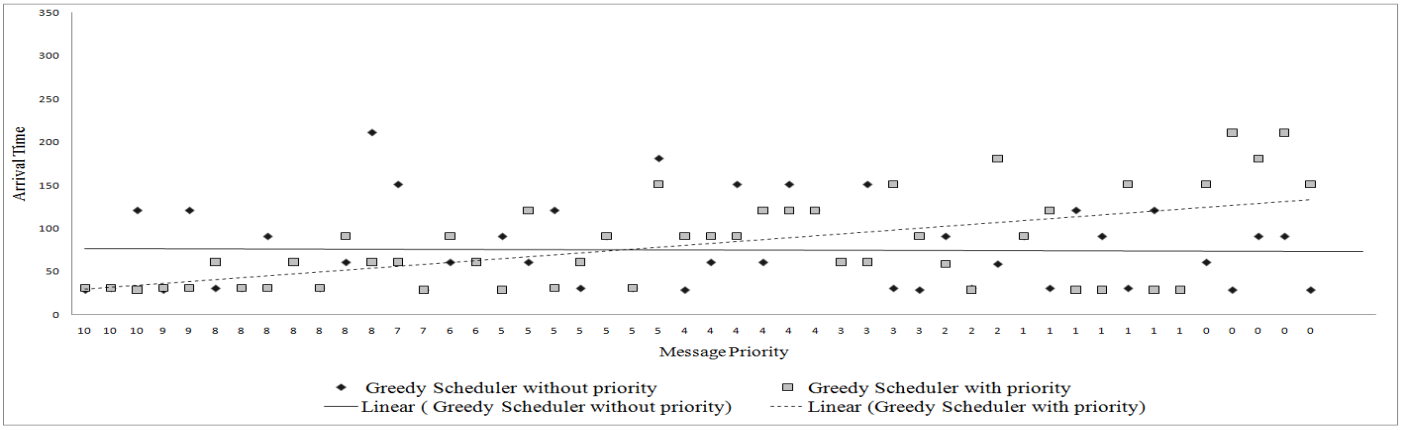

Fig. 7. Comparison between greedy with and without Priority

VII. Conclusion and future work

This paper highlights the problem of data transfers in the cloud, transfers that affect cloud performance by moving resources to the virtual machine on which they are needed. We described a method for big-data transfer optimization based on network characteristics and we considered the constraints introduced by SLA. We proposed two greedy scheduling algorithms and we compared them to a standard First Come First Served, packet transfer order. Several simulation experiments highlight the obtained improvement.

Having a scheduler for transfers that takes into account global transfers can bring an improvement to applications and to the user experience. We believe that better algorithms may be obtained that further improves on the scheduling so that cloud resource usage is minimized. We also considered using the discussed algorithms for Cloud user data transfers, for this to be practical an API needs to be provided to the Cloud user, throw which to schedule only the big-data transfers. The API should offer some sort of feedback, which can be utilized by the user to further optimize it's processing.

Acknowledgment

This work was partially supported by project "ERRIC - Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1", ID: 264207. The work has been co-funded by the Sectorial Operational Program Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557

References

[1] Ming Zhao and Renato J. Figueiredo. 2007. Experimental study of virtual machine migration in support of reservation of cluster resources. In *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing* (VTDC '07). ACM, New York, NY, USA, Article 5, 8 pages.

[2] Joseph Hall, Jason Hartline, Anna R. Karlin, Jared Saia, and John Wilkes. 2001. On algorithms for efficient data migration. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms* (SODA '01). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 620-629

[3] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. 2007. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX conference on Networked systems design and implementation* (NSDI'07). USENIX Association, Berkeley, CA, USA, 17-17.

[4] Alexander Stage and Thomas Setzer. 2009. Network-aware migration control and scheduling of differentiated virtual machine workloads. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing* (CLOUD '09). IEEE Computer Society, Washington, DC, USA, 9-14.

[5] Gueyoung Jung, Nathan Gnanasambandam, and Tridib Mukherjee. 2012. Synchronous Parallel Processing of Big-Data Analytics Services to Optimize Performance in Federated Clouds. In*Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing* (CLOUD '12). IEEE Computer Society, Washington, DC, USA, 811-818.

[6] Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, and Bei Wang. 2011. Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing* (CLOUD '11). IEEE Computer Society, Washington, DC, USA, 267-274.

[7] Albert Greenberg, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta Microsoft Research, Redmond, WA, USA. *Towards a Next Generation Data Center Architecture: Scalability and Commoditization*