Procedia Computer Science

Procedia Computer Science 101, 2016, Pages 351 – 358

YSC 2016. 5th International Young Scientist Conference on Computational Science

# Implementing, Evaluating and Extending the Model of LIT Cloud Infrastructure at JINR

Vagram Airiian[1,2,4], Vladimir Korenkov[2,3*], Andrey Nechaevskiy[2]

[1]*Institute of System Analysis and Control, Dubna State University, Dubna, Russian Federation*
[2]*Laboratory of IT, Joint Institute for Nuclear Research, Dubna, Russian Federation*
[3]*Plekhanov Russian University of Economics, Moscow, Russian Federation*
[4]*Department of Informatics, Technical University of Munich, Munich, Germany*
*wagram@jinr.ru, korenkov@jinr.ru, nechav@mail.ru*

**Abstract**

The project is aimed at modelling of LIT Cloud Infrastructure at the Joint Institute for Nuclear Research (JINR) and solving a problem of effective migration algorithms considering virtual machines deployment at the facilities. The main intention is to distribute virtual machines over involved resources with the highest density possible. In order to fulfil the requirements regarding a successful solution of the problem, the assessment of existing modelling frameworks and their comparison has been carried out. Furthermore, an initial to-be-optimized model has been implemented and evaluated. Currently, the project is focused on researching the Bin Packing Problem in respect of virtual machines migration and developing the migration scheme for the cloud infrastructure extending the existing model.

*Keywords:* modelling, SimGrid, simulation, virtual machines, live migration, Bin Packing Problem

## 1 Introduction

Nowadays, the use of simulations regarding computing resources is increasingly popular in order to, for instance, detect bottlenecks in deployed architecture of data center, experiment with rearranging topology and resources prior to physical intervention, model functioning of not yet existing computing infrastructures. It is a very good way to examine supposed solutions to a problem without interfering with actual working facilities which can result in them being out of service, data loss and corruption. Furthermore, modelling frameworks allow to debug and evaluate heuristics regarding resource management much easier and faster.

The problem considered in this paper is related to effective resource utilization. The main idea is to place as many virtual machines as possible to the first server, then to the second, then to the third and

* This work is supported by RFBR grant № 15-29-07027

Peer-review under responsibility of organizing committee of the scientific committee of the 5th International Young Scientist Conference on Computational Science

doi:10.1016/j.procs.2016.11.041

so on. Therefore, the servers will be arranged into two categories of densely populated and unpopulated ones. The unpopulated servers can be involved in solving a different task, for instance, WLCG TIER ones. In the worst case, if there are no any extra computing jobs, idle servers can be put into a sleep state to save energy.

To provide the highest density on a separate node memory and CPU overcommitment will be used. The overcommitment is a policy of allocating virtual memory, virtual CPU cores and/or other virtual resources with no guarantee that sufficient amount of physical representation for them exist. For instance, if a virtual machine utilizes no more than 20% of each physical CPU core, but wants more parallelism for some processes, we can assign additional virtual CPU cores getting at least four times of their initial amount without stability impact — physical CPU utilization will be a bit more than 80% (Achtemichuk, 2014).

Currently there are no heuristics to manage virtual machines migration and virtual resources reallocation, therefore, either the overcommitment or the migration are performed manually by system administrators. It is a very ineffective approach, which does not allow to accomplish the goal successfully as a person can generally notice mostly very emerging cases of intervention necessity. To get rid of this difficulties, the Cloud Infrastructure Group at LIT JINR has made the decision of developing a smart scheduler for the cloud infrastructure.

The main goal of the plan is to develop, test and apply an effective elaborate heuristic (i.e., a virtual machines migration scheme — a set of scheduling algorithms for migration), for freeing a maximum amount of computing resources at each given moment. This paper is dedicated to modelling the cloud infrastructure, testing it and defining the basis of the further research regarding some of the first inevitable steps on the roadmap of the project.

# 2 Modelling Frameworks

In order to model the cloud infrastructure, we need either to write our own simulation engine, which is a superfluous activity, or review and choose one of the existing modelling frameworks. There is a large number of papers dedicated to comparison and assessments of the different frameworks, thus we have used some of them to gather information about the four open source packages, we believe are among the most popular for cloud computing modelling. They are CloudSim, GreenCloud, iCanCloud and SimGrid (see Table 1).

| | CloudSim | GreenCloud | SimGrid | iCanCloud |
|---|---|---|---|---|
| Language | Java | C++/OTcl | C, Lua, Java, Ruby | C++ |
| Distribution | Source code | Source code / prebuilt VM | Source code / prebuilt binary | Source code |
| Stable Release | 02.05.2013 | 13.02.2016 | 13.10.2015 | 16.02.2015 |
| Latest Release | 02.05.2013 | 13.02.2016 | 17.05.2016 | 16.02.2015 |
| VM Live Migration | Yes | Yes | Yes | No |

**Table 1:** Basic parameters of the modelling frameworks

Despite a huge amount of characteristics featured by the packages, we used only small number of parameters to make the comparison clearer. The vital one is a live migration ability as this is the feature we are seeking for seamless implementation of our simulations which will not distract us by development of our own migration module (Wenhong, et al., 2015) (Mihailov & Radchenko, 2014) (Casanova, et al., 2014).

Although iCanCloud is a powerful tool for modelling computing infrastructures featuring POSIX, a bunch of communications protocols (UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF), storage systems support, GUI and cost accounting, it lacks an ability to migrate virtual

machines in a cloud, which is a major shortcoming (Korenkov, et al., 2014). Despite its rather recent release and requirement for ordinary C++, we could not recommend it for own usage because of this.

Furthermore, we have also refused the use of CloudSim and GreenCloud due to several reasons. Considering CloudSim it had an unacceptable state of development support with the last stable release submitted three years ago (at the start of our project — currently the latest release is 24.05.2016), being unaware of its further destiny we have decided to reject it (University of Melbourne, 2016). Although GreenCloud was a good candidate with a great dashboard, we have declined the idea of using it because of its meaningless documentation and requirement for C++/OTcl questioning proper support of the model in the future (University of Luxembourg, 2013).

Therefore, the only modelling framework meeting our expectations is SimGrid. SimGrid is a versatile modelling framework which allows to simulate the behavior of distributed systems such as Grids, Clouds, HPC or P2P. Figure 1 illustrates the main components and the key concepts of SimGrid.

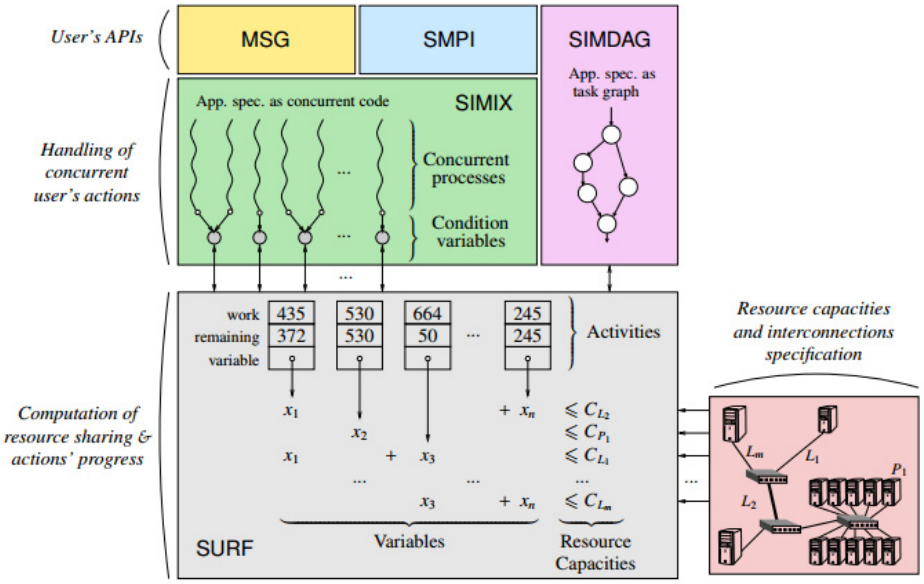


**Figure 1:** Design and internals of SimGrid

The top layer of abstraction comprises three APIs through which a simulator can be developed: MSG, SMPI and SIMDAG. In our project we are using the MSG API. MSG allows users to describe a simulated application as a set of concurrent processes by executing code implemented in C, C++, Java, Lua, or Ruby, and place MSG calls to simulate computation and communication activities. The mechanisms for simulating the concurrent processes for MSG are implemented as part of a layer called SIMIX, which is a kernel that provides process control and synchronization abstractions. The set of concurrent processes is depicted in the SIMIX box in the figure. All processes synchronize on a set of condition variables, also shown in the figure. Each condition variable corresponds to a simulated activity, computation or data transfer, and is used to ensure that concurrent processes wait on activity completions to make progress throughout time. The simulation core, i.e., the component that simulates the execution of activities on resources, is called SURF and is shown in the bottom-left of the figure. Each activity is defined by a total amount of work to accomplish and a remaining amount of work. When its remaining amount of work reaches zero the activity completes, signaling the corresponding SIMIX condition variable. This simulating approach is purely analytical. Formally, given a resource r, and a set of simulated activities, A, the model specifies the constrained Max-Min optimization problem (Casanova, et al., 2014).

## 2.1 Model of Cloud Infrastructure

LIT Cloud Infrastructure is based on OpenNebula. Cloud Nodes (CNs) support either OpenVZ or KVM virtualization. CNs are connected to two 1 Gbps subnetworks: private one used for exchanging data between cloud facilities and public one used for interaction with users and external services to maintain sufficient level of service. All virtual machines and containers are initialized on CNs (Baranov, et al., 2015).

The physical equipment is represented by:

- 30 servers with 4 cores CPU, 8 GB of RAM and 250 GB of HDD each;
- servers with 2 cores CPU, 4 GB of RAM and 800 GB of HDD each;
- 1 server with 4 cores CPU, 8 GB of RAM and 2330 GB of HDD;
- servers with 12 cores CPU, 36 GB of RAM and 1000 GB of HDD each;
- 1 server with 24 cores CPU, 24 GB of RAM and 1000 GB of HDD.

In SimGrid compute resources are defined in terms of compute capacities (e.g., CPU cycles per time unit). They are interconnected via a network topology that comprises network links and routing elements, defined by bandwidth capacities and latencies. All resources can be optionally associated with time-stamped traces of available capacity values including possible downtime. An example specification of available resources is depicted in the bottom-right of Figure 1, highlighting three network links (L1, L2, Lm) and one compute resource (P1) (Casanova, et al., 2014).

Despite rather sophisticated configuration of the infrastructure, we have decided to simplify a model leaving only the parts required for simulations of migration processes. To describe a simplified topology, a platform description file was created. It is written in XML and contains a description of a single Autonomous System (AS) representing the cloud infrastructure in whole. AS comprises 39 hosts combined of five host types, four HDD and four RAM types; one router connecting all hosts with one type of 1 Gbps link forming 39 routes (780 auxiliary routes) as shown in Figure 2.

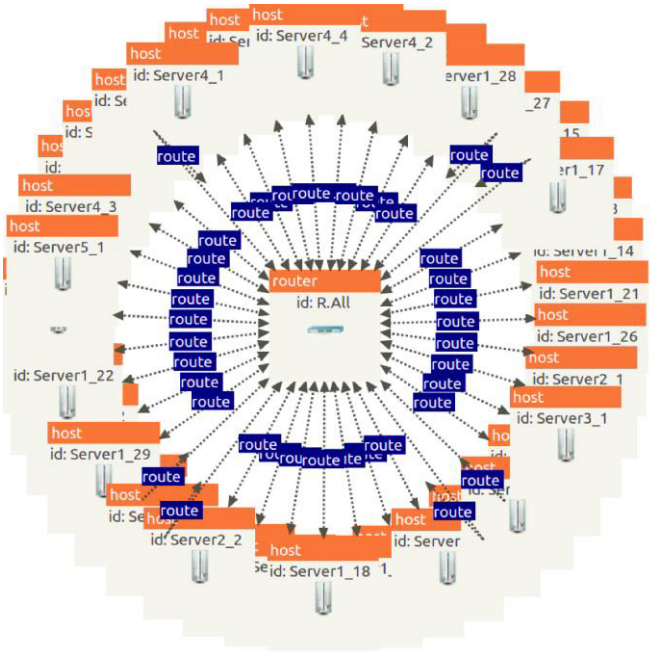


**Figure 2:** Scheme of LIT Cloud Infrastructure model

# 3 Evaluating Model

To evaluate the model, it is necessary to carry out a simulation experiment. However, since the current configuration of the cloud infrastructure lacks any virtual machines migration scheme and the migration process is performed manually, currently there is no way to evaluate the model statistically by simulating the same migration scheme as could be implemented in the real life configuration and comparing the output.

Considering the situation, we have suggested an idea to run a minor virtual machines migration scheme which we could also easily assess, as we have called it — a migration game, to check a logical consistency of the model and the simulator we have written using SimGrid.

In terms of the process of migration on the basic level of exchanging data between hosts we have used a standard pre-copy algorithm, which is default for SimGrid (supposedly we will consider optimizing it after accomplishing general stages of the whole project).

Figure 3 illustrates the pre-copy algorithm, which consists of six stages (Clark, et al., 2005):

- Pre-Migration. An active VM on a physical host A.
- Reservation. A request is issued to migrate an OS from host A to host B.
- Iterative Pre-Copy. During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.
- Stop-and-Copy. We suspend the running OS instance at A and redirect its network traffic to B.
- Commitment. Host B indicates to A that it has successfully received a consistent OS image.
- Activation. The migrated VM on B is now activated.

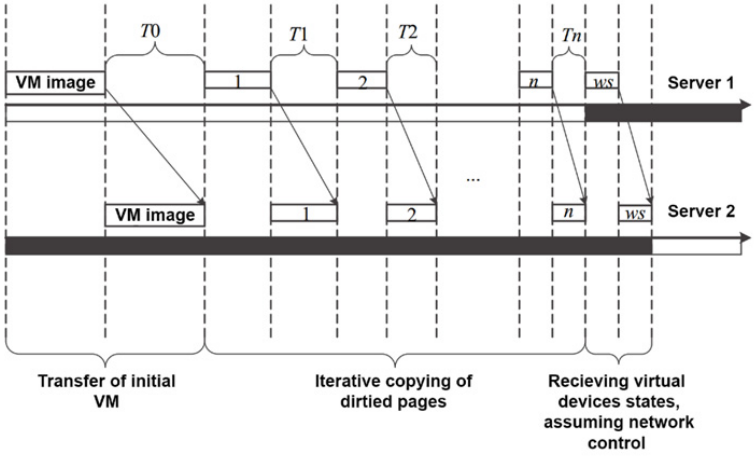


**Figure 3:** The Pre-copy algorithm in operation **(Aleksankov, 2015)**

The migration game is a simple algorithm. It starts the simulation with *N* virtual machines with randomly assigned load from 0 to 90 percent of virtual CPU (vCPU), which are randomly distributed over 39 hosts. Before the first migration is performed, all values are saved for a further assessment. The load on each vCPU stays constant until the end of the simulation, otherwise it is hard to assess the whole set of dynamically changing variables. On each step of the simulation the migration scheme detects a host with the lowest resource utilization and places all virtual machines with the lowest resource usage on the host with overcommitment. If any host became idle, it would be immediately put into a sleep state. After successful migration of all machines on the current step the process repeats.

Figure 4 illustrates the results of the simulation, where $t$ is an internal simulation time reported by SimGrid MSG. Each of three simulations has been run 60 times. Average results are mostly equal to individual ones (there were small fluctuations due to randomness of initial data). Results show that, although eventually there are a lot of free machines, however, the number of starving hosts (that are overloaded and suffer from shortage of resources) is incredibly high by the end of the simulation (beginning from some $t_i$ the number of each type of hosts stabilizes). Anyway, the migration scheme has not been intended to be applied to the real life facilities.

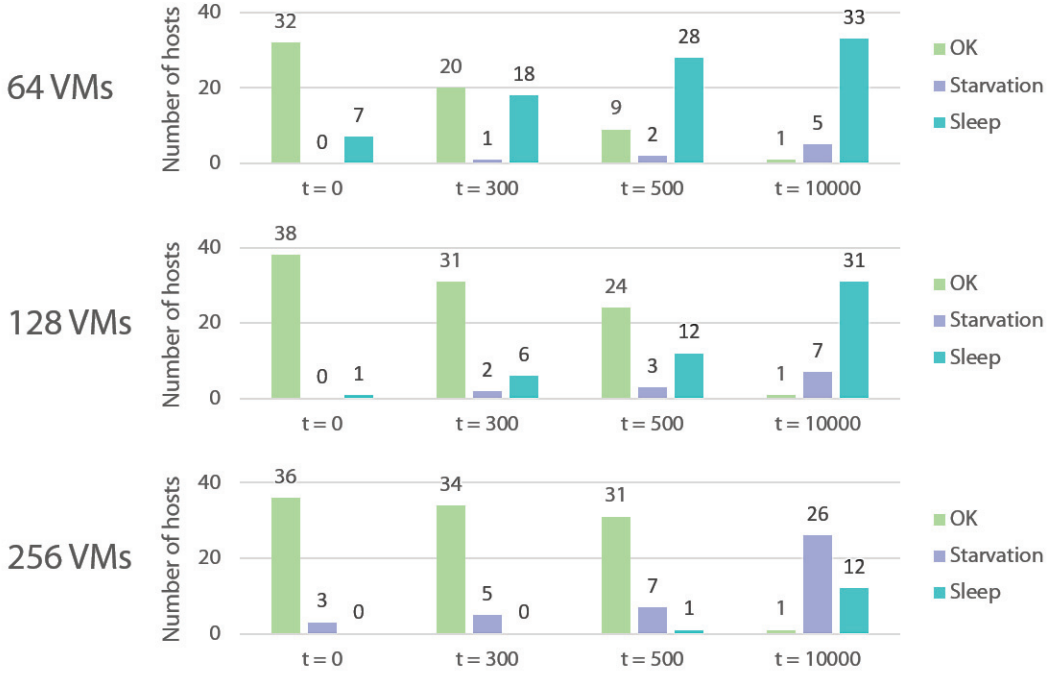


**Figure 4:** Results of the simulation for 64, 128 and 256 virtual machines

The successful assessment of the results has been carried out with a Python program using values obtained from the simulator, which proven the logical consistency of the model (the same results have been got).

# 4 Bin Packing Problem

As seen from the obtained results we need a way to effectively rearrange virtual machines regarding their resource utilization while keeping number of free hosts as high as possible. It means that we have to decide which number of utilized hosts is sufficient for a presented set of virtual machines. This is the problem known as the Bin Packing Problem. It can be formulated as following.

Given: $n$ objects need to be placed in bins of capacity $L$ each. Object $I$ requires $l_i$ units of bin capacity. Object: determine the minimum number of bins needed to accommodate all $n$ objects (Xin & Zhuanzhuan, 2015).

For virtual machines the problem does not sound so straightforward as this is a variant of Bin Packing when items share space when packed together known as VM packing (Sindelar, et al., 2011).

As a decision problem the Bin Packing Problem is NP-complete and as an optimization problem Bin Packing is NP-hard. Approximation Algorithm for Bin Packing includes First Fit, Best Fit, First Fit Decreasing and Best Fit Decreasing (Xin & Zhuanzhuan, 2015).

The First Fit algorithm provides a fast but a non-optimal solution, placing each item into the first fitting bin. The algorithm can be made more effective by pre-sorting the list of objects into decreasing order (the First Fit Decreasing algorithm), although this still does not guarantee an optimal solution. On the contrary, the Best Fit algorithm tries to makes the best use of space in bins fitting each item into the most suitable bin, so does the Best Fit Decreasing algorithm. Both First Fit and Best Fit require *O(nlogn)* time, where n is the number of objects to be placed (Lewis, 2009).

## 4.1 Solution Proposal for Migration Scheme

Since we are interested in the most efficient distribution of virtual machines and not in the fastest migration scheduling, the Best Fit algorithms should be considered in the development of the migration scheme for the cloud infrastructure.

However, to apply an approach regarding the Bin Packing Problem we need to introduce a value $L$ defining the bin capacity, where bins are hosts. Thus, $L_A$ is how many virtual machines we can place on a host $A$. In order to fulfil this requirement, the Cloud Infrastructure Group has come out with the following ranging algorithm of defining the capacity.

All hosts are assessed for the next characteristics: number of CPUs, RAM capacity, HDD capacity, number of virtual machines presented on the host. The hosts with similar characteristics are combined in groups and each group $G$ is assigned a rank $L_G$. Similarly, ranging of all virtual machines is performed (on the basis of requested CPUs, actual CPU load, requested RAM capacity, actual RAM usage, requested HDD capacity, actual HDD usage) and $l_i$ is assigned to each group $I$. Scales, according to which capacities $L_G$ and $l_i$ are assigned, are defined with the utilization statistics acquired by a monitoring system of the cloud infrastructure (Balashov, et al., 2015).

# 5 Future Plans

Given the presented solution proposal, the next step to be done is extending the model with a definition of host capacity LA and developing our own migration scheme based on the Bin Packing Best Fit (or Best Fit Decreasing) algorithm according to the proposed strategy in respect of the long-time statistics, which is now being gathered by the Cloud Infrastructure Group.

Furthermore, research regarding application of the expected migration scheme to OpenVZ and KVM usage in the cloud will be done and tested.

Finally, the model will be revaluated and adjusted according to the results gained from the real life experiment and optimization of the migration scheme and algorithms will take place.

# 6 Conclusion

The Cloud Infrastructure Group is carrying out an ambitious and challenging project of implementing the smart scheduler for the Cloud Infrastructure to manage resources effectively. Obviously, modelling is unavoidable to accomplish such a great goal as in most other cases of fully operating network systems in order not to render them out of service in a process of development. In this paper we have started from the basics, specifically, have studied and compared the popular modelling frameworks, created the simple model using the chosen framework, evaluated the model regarding the logical consistency and outlined the path for further research to gradually come to a satisfactory solution.

# References

Achtemichuk, M., 2014. *When to Overcommit vCPU:pCPU for Monster VMs.* [Online] Available at: https://blogs.vmware.com/vsphere/2014/02/overcommit-vcpupcpu-monster-vms.html

Aleksankov, S., 2015. Models of live migration with iterative approach and move of virtual machines. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics,* 15(6), pp. 1098-1104.

Balashov, N. et al., 2015. JINR Cloud Service: Status and Perspectives. *Trudy Instituta sistemnogo programmirovania RAN,* 27(6), pp. 345-354.

Baranov, A., Balashov, N., Kutovskiy, N. & Semenov, R., 2015. Cloud Infrastructure at JINR. *Компьютерные исследования и моделирование,* 7(3), pp. 463-467.

Casanova, H. et al., 2014. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms.. *Journal of Parallel and Distributed Computing,* 74(10), pp. 2899-2917.

Clark, C. et al., 2005. *Live migration of virtual machines.* Berkeley, CA, USENIX Association, pp. 273-286.

Korenkov, V., Muravyev, A. & Nechaevsky, A., 2014. Cloud Computing Simulation Packages. *Системный анализ в науке и образовании,* Issue 2.

Lewis, R., 2009. A General-Purpose Hill-Climbing Method for Order Independent Minimum Grouping Problems: A Case Study in Graph Colouring and Bin Packing. *Computers and Operations Research,* 36(7), pp. 2295-2310.

Mihailov, P. & Radchenko, G., 2014. Modeling and Performance Evaluation of Cloud Systems. *Bulletin of the South Ural State University,* 3(3), pp. 109-123.

Sindelar, M., Sitaraman, R. & Shenoy, P., 2011. *Sharing-Aware Algorithms for Virtual Machine Colocation.* San Jose, CA, s.n., p. 367–378.

University of Luxembourg, 2013. *GreenCloud User Manual.* [Online] Available at: https://greencloud.gforge.uni.lu/ftp/greencloud-user-manual.pdf

University of Melbourne, 2016. *Cloudslab/cloudsim.* [Online] Available at: https://github.com/Cloudslab/cloudsim/releases

Wenhong, T. et al., 2015. *Open-Source Simulators for Cloud Computing: Comparative Study and Challenging Issues.* [Online] Available at: https://arxiv.org/abs/1506.01106

Xin, L. & Zhuanzhuan, Z., 2015. A Virtual Machine Dynamic Migration Scheduling Model Based on MBFD Algorithm. *International Journal of Computer Theory and Engineering,* 7(4), pp. 278-282.