First Published, 2015

ISBN : 978-93-81212-92-9

Printed in India:
BHARTI PUBLICATIONS
4819/24, 3rd Floor, Ansari Road, Darya Ganj
New Delhi-110002
Mobile : +91-989-989-7381
E-mail : bhartipublications@gmail.com
info@bharatipublications.com
Website : www.bhartipublications.com

PRINTED IN INDIA

Published by Onkar Bharti for Bharti Publications. Typeset by Gaurav Graphices, Rajouria Printers, Delhi.

Proceedings of

2nd International Conference On

Innovation & Sustainability

Managing for Change

(Management and Computer Science)

63
CHAPTER

Analysis of a Novel Weighted-Priority Task Scheduling Algorithm in Cloud Computing through Simulation

Bhawna Taneja, Dr. Rajender Nath** & Bharat Chhabra****

ABSTRACT

The idea of Cloud Computing is to enable its users to exploit the full spectrum of available resources to match the varying requirements of their jobs. This capability of delivering the services by the cloud is practically feasible only if it is supported by some efficient task scheduling algorithm at the Virtual Machine level. This paper examines the available literature in this direction and also proposes a new weighted-priority algorithm for scheduling the cloudlets. The algorithm has also been simulated using CloudSim toolkit to analyze its performance in varying workload conditions.

Keywords: *Cloudlet, CloudSim, Virtual Machine.*

1. Introduction

With the rapid growth of virtualization and wide deployment of cloud computing technologies, the quantum of tasks arriving at any cloud are increasing steeply. These all tasks need to be carefully serviced by the cloud resources to maintain the acceptable throughput and hence profits. This responsibility is borne by efficient task scheduling algorithms for deciding the number of resources and timing for allocation of resources to a task. The available task scheduling algorithms may be classified as static or dynamic algorithm, pre-emptive or non-preemptive algorithm and centralized or distributed algorithms etc. A lot of research has been carried out by researchers in each of these categories of algorithms. Despite of some algorithm being a descendant of any category, the major goals for each one of these is to achieve maximum throughput rate, minimum makespan time, maximum resource utilization rate, meeting the SLA parameters and

* Research Scholar, DCSA, Kurukshetra University Kurukshetra, Haryana.

** Chairman, DCSA, Kuruskshetra University Kurukshetra, Haryana.

*** Asstt. Prof., Govt. College Julana (Jind), Haryana.

QoS parameters for each task. The idea of this keynote paper is to highlight the major features of these already available priority aware task scheduling algorithms and to propose a newly developed weighted-priority task scheduling algorithm.

2. Related Work

To meet the varying demands of different jobs, priority algorithms that are based on priority values of jobs and resources, have to select one more prior task before other tasks for execution. The decision of setting the priority value of one resource or task may be dependent on several factors such as bandwidth or memory requirements, QoS requirements, SLA requirements etc.

The authors[2] proposed scheduling algorithm that integrated prioritization of task as per bandwidth requirements and then follows SJF algorithm for allocation of the resources to user tasks. Authors used CloudSim framework to simulate the algorithm and compared the results with existing algorithms for evaluation.

A new priority based job scheduling algorithm based on the theory of Analytical Hierarchy Process (AHP) was proposed by [1]. It was based upon a comparison matrix. The algorithm considered priority at three levels namely at scheduling-level, resources level and job level. They have verified the algorithm using an example considering 3 resources and 4 jobs and compute makespan for the algorithm.

A user-priority Guided Min-Min scheduling algorithm was proposed by [3]. The objective of the algorithm was to overcome the limitation of load unbalancing in traditional Min-Min scheduling algorithm. The authors proposed two task scheduling algorithms namely (Load Balance Improved Min-Min) LBIMM and PA-LBIMM (User-Priority Aware Load Balance Improved Min-Min). LBIMM does not consider user priority and only load balanced the resources whereas PA-LBIMM algorithm considered the user priority. They divided the tasks into two groups namely G1 and G2, G1 for VIP user tasks and G2 for ordinary tasks. Min- Min algorithm was used to schedule all the tasks in G1 first followed by tasks in G2. The authors simulated the algorithm using Matlab toolbox.

The authors in the paper [4] utilized a combination of Batch Mode Heuristic Priority and Round Robin scheduling for load balancing. They took load balancing factor as a heuristic priority factor. They calculated the load balancing factor of each server using a formula. This factor was updated after execution of each task. The authors allocated the jobs according to calculated priority and in Round Robin fashion so that workload across cloud is managed.

A threshold based priority scheduling was proposed by [5]. They assigned the priority to the job based on the submission time or type of job request. They further compared it with Shortest Job First Scheduling and Round Robin scheduling algorithm.

3. Problem Formulation

In the traditional Priority Task Scheduling algorithm, all of the submitted tasks are ranked from highest priority task to lowest priority task and according to these ranks, tasks are executed. The method of assigning the priority to each task is generally achieved

[illegible]

4. Proposed Algorithm

In this algorithm, four weights are assigned to major features and minor features of the task and resources. [illegible] algorithm starts with calculating the priorities of the tasks submitted and of the resources, which is followed by assigning the [illegible] VM to the highest priority task. [illegible] the first step, features of task like length, filesize etc. are initialized with weights and similarly all features of resources like MIPS, RAM, Bandwidth etc. are also assigned certain weights. [illegible] is followed by the process of calculating the cumulative priorities for all tasks and resources as per the weights assigned to their specific features. Once the priorities are calculated successfully, the task scheduling is processed by allocating the resources to the submitted tasks as governed by these priority values of tasks and resources. The following data structures have been used in this algorithm:

1. Task is a set of size n and represents user tasks submitted to the cloud provider for execution. These are to be bind to the suitable VM. Task = [illegible] where N is the set of natural numbers.
2. Resource is a set of m resources available with the cloud provider. These are potential VMs that are to be allocated to tasks in [illegible]. Resource = [illegible] where N is the set of natural numbers.
3. Length is a set of size n that denotes the corresponding length of each task. Length = [illegible] where [illegible] is the length of the [illegible] task [illegible]
4. Filesize is a set of size n that denotes the corresponding filesize of each task. Filesize = [illegible] where [illegible] and [illegible] is the filesize of the [illegible] task [illegible]
5. [illegible] is a set of n elements corresponding to the [illegible] of each submitted task. [illegible] where [illegible] and [illegible] is the [illegible] of the [illegible] task [illegible]
6. [illegible] is a set of size m that represents the computing power of each VM in terms of its [illegible] capability. [illegible] = mips, mips, ..., mips where [illegible]

and $mips_k$ is the mips of resource r_k | 1 <= k <= m.

7. Ram is a set of size m that represents the main memory of each VM in terms of bytes. Ram(m) = {ram_1, ram_2,..... ram_m}, where m å N and ram_k is the ram of resource r_k | 1 <= k <= m.
8. BandW is a set of m elements corresponding to the bandwidth of each VM. BandW(m) = {bw_1, bw_2, ..., bw_m}, where m E N and bw_k is the bandwidth of resource r_k | 1 <= k <= m.
9. Pri_Task is a set of n elements that contains the computed priorities of tasks in T according to the proposed algorithm. Pri_Task(n) = { pt_1, pt_2, ..., pt_n} where pt_i is the computed value of priority of task t_i, 1 <= i <= n.
10. Pri_Res is a set of m elements which contains the computed priorities of resources in R according to the algorithm. Pri_Res(m) = { pr_1, pr_2, ..., pr_m} where pr_k is the computed value of priority of resource r_k, 1 <= k <= n.

Besides all these sets of elements, some static constants have also been used to represent the weights such as length_wt, filesize_wt and outputsize_wt have been used to represent the weight being assigned to length, filesize and outputsize properties of the submitted task and these have been initialized to 0.6, 0.3 and 0.1 respectively. Similarly, some other static constants have also been used to assign the weights to properties of VMs. These are mips_wt, ram_wt and bandwidth_wt. these are initialized to 0.5, 0.3 and 0.2 repectively. The values for these constants have been carefully selected that indicates the weightage i.e. the impact factor of the respective feature of a task or VM.

Algorithm 1: Weighted priority Algorithm

Input: Task(n), Resource(m)

Step 1:

Initialization: // assign weights to various attributes of task and of resource

Set length_wt = 0.6, filesize_wt = 0.3, outputsize_wt = 0.1

Set mips_wt = 0.5, ram_wt = 0.3, bandwidth_wt = 0.2

Step 2: // Computing the priority values for each task

Compute the priorities for each task t_i in Task, where 1 <= i <= n

pt_i = length_wt * lt_i + filesize_wt * fst_i + outputsize_wt * ost_i;

Step 3: // Computing the priority values for each resource

Compute the priorities of each resource r_k in Resource, where 1 <= k <= m

pr_k = mips_wt * $mips_k$ + ram_wt * ram_k + bandwidth_wt * bw_k;

Step 4: // Allocating the VM to each task according to priority.

While set Task not empty, assign highest priority resource from set Resource to each highest priority submitted task in set Task

Bind r_k with t_i such that

(i) $pr_k > pr_j$ where $1 <= j <= m$, $1 <= k <= m$ and $k < > j$ and

(ii) $pt_i > pt_q$ where $1 <= i <= n$, $1 <= q <= n$ and $i < > q$

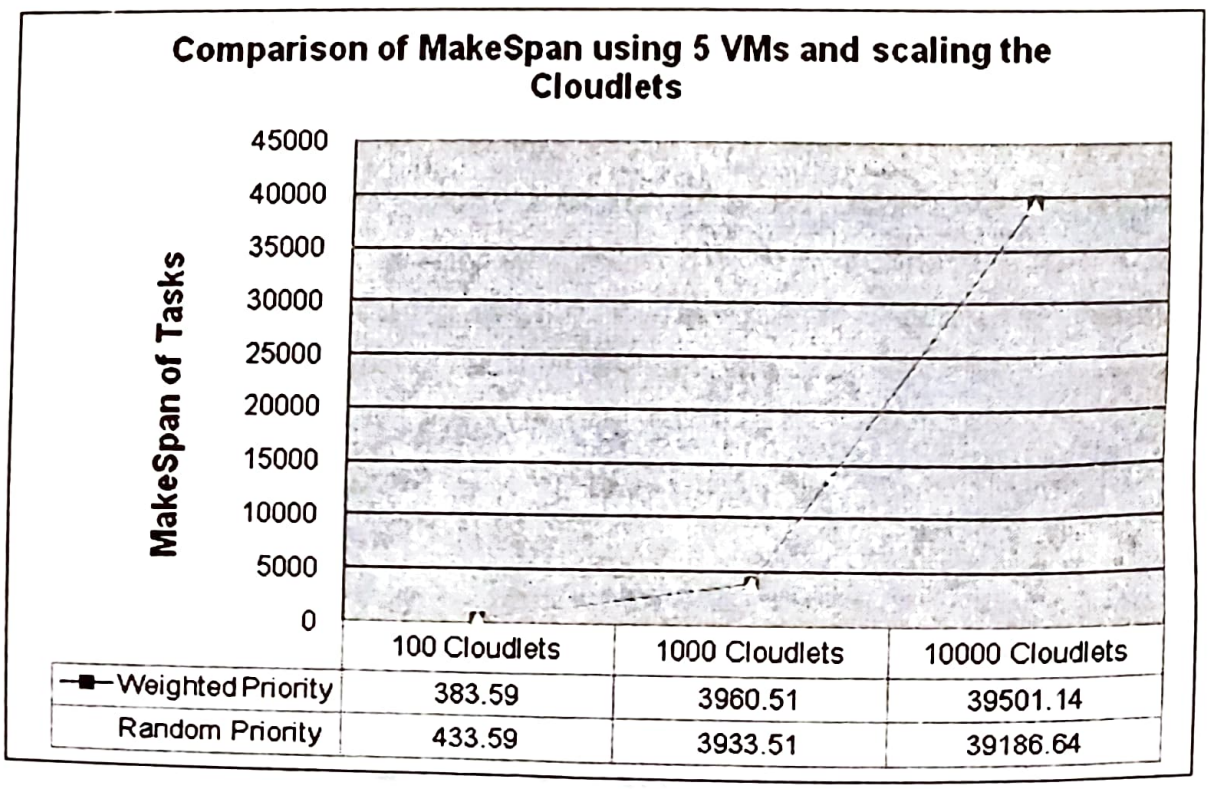
5. Experimental Setup

In order to evaluate the proposed weighted-priority task scheduling algorithm, it is simulated with CloudSim toolkit 3.0.3. [11] under various workload conditions. The algorithm has been simulated under three different workload conditions.

Workload Condition 1

Computing power of VMs is randomly selected from 3000, 4000 and 5000. Length of cloudlets is also assigned randomly from a range starting with 30000 instructions to 50000 instructions. In this first workload condition, cloudlets were taken to be 100, 1000 and 10000 with 5 resources only.

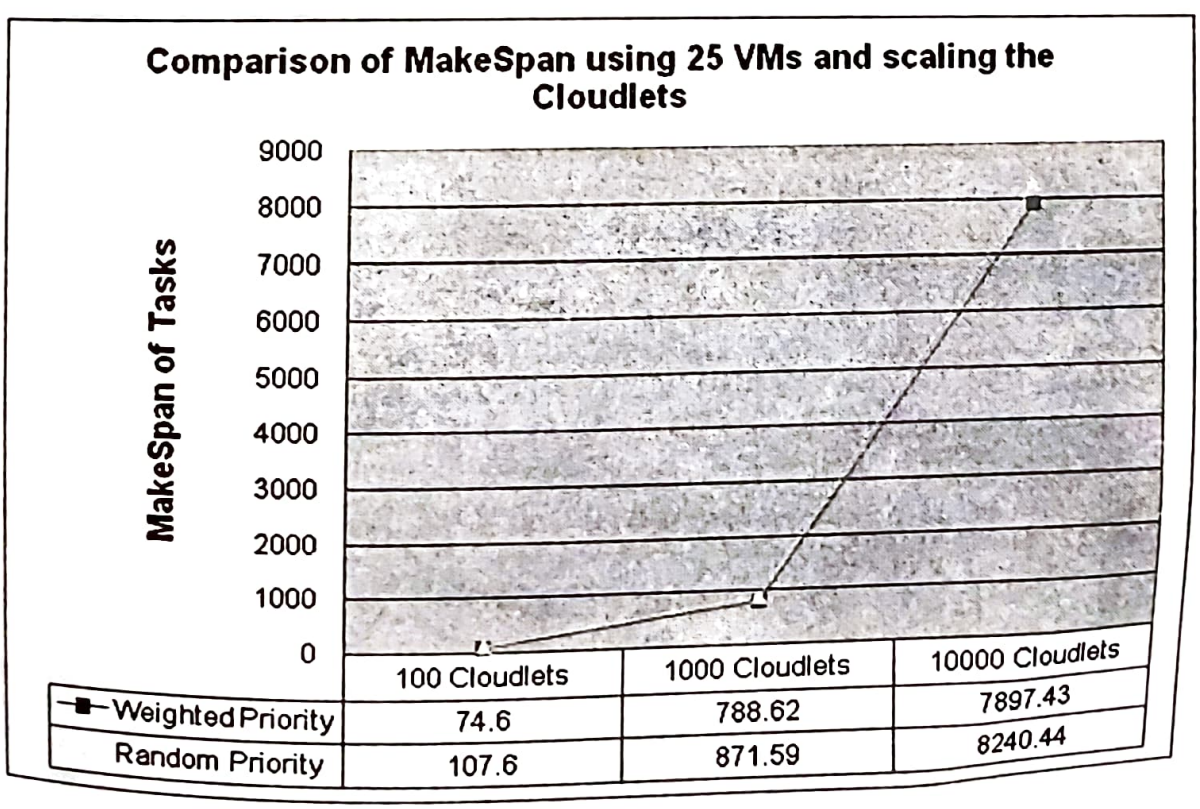
Virtual Machine	*Computing Power (MIPS)*	*Bandwidth (Mbps)*	*RAM*
VM#1	2000	4400	2
VM#2	4800	2400	2
VM#3	3600	2100	2
VM#4	4100	3300	2
VM#5	2700	3000	2



Workload Condition 2

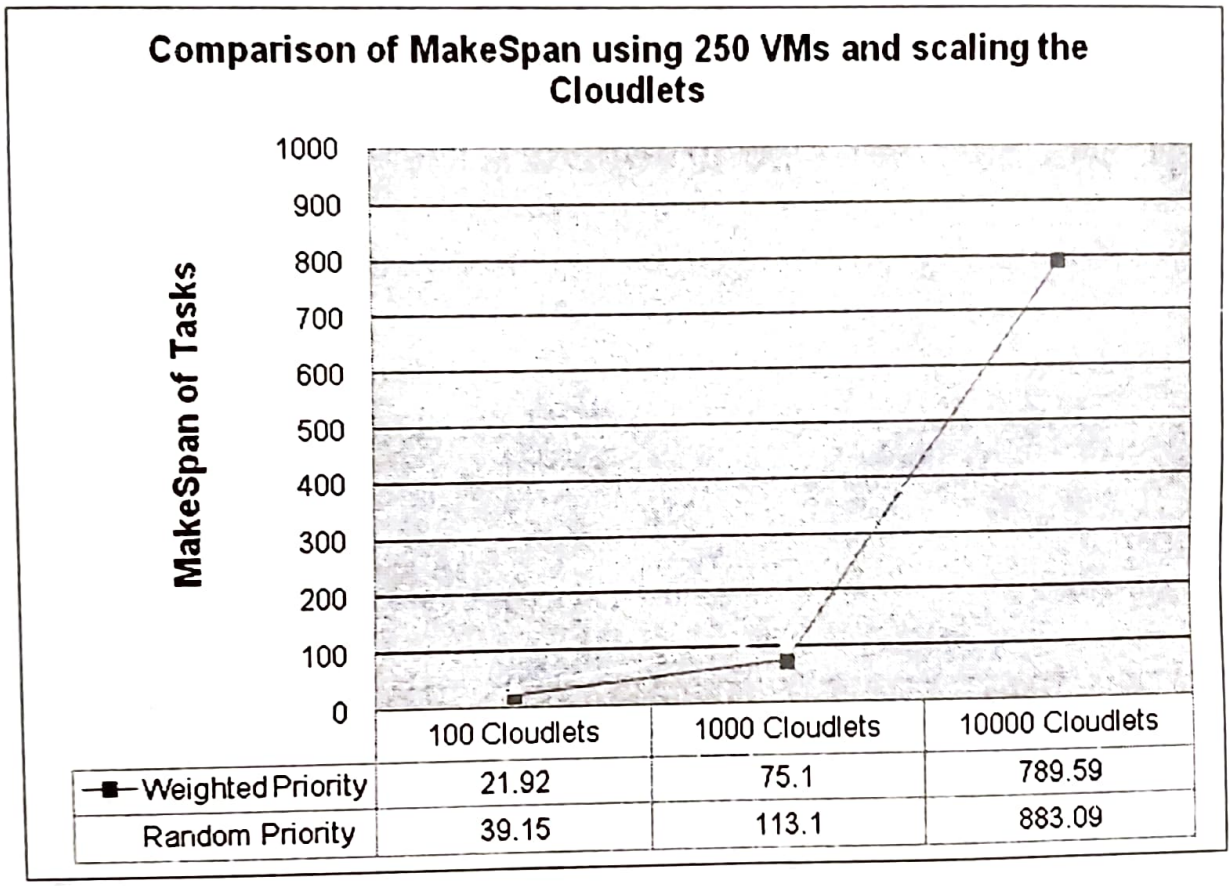
In this workload condition, 25 VMs are created with same randomly selected mips as taken in the first experiment and 100,1000 and 10000 cloudlets respectively with randomly assigned instructions (30000 to 50000 instructions) are taken in submitted tasks list. By executing the algorithm with these specifications, figure 2 is drawn showing the makespan. This figure clearly indicates that as we scaled up the VMs and cloudlets, the makespan increases linearly implying that algorithm is scalable.

Virtual Machine	*Computing Power(MIPS)*	*Bandwidth (Mbps)*	*RAM*	*Virtual Machine*	*Computing Power(MIPS)*	*Bandwidth (Mbps)*	*RAM*
VM#1	2000	4400	2	VM#14	2000	4400	2
VM#2	4800	2400	2	VM#15	2000	4400	2
VM#3	2000	4400	2	VM#16	2000	4400	2
VM#4	4800	2400	2	VM#17	2000	4400	2
VM#5	2000	4400	2	VM#18	2000	4400	2
VM#6	4800	2400	2	VM#19	2000	4400	2
VM#7	2000	4400	2	VM#20	2000	4400	2
VM#8	4800	2400	2	VM#21	2000	4400	2
VM#9	2000	4400	2	VM#22	2000	4400	2
VM#10	4800	2400	2	VM#23	2000	4400	2
VM#11	2000	4400	2	VM#24	2000	4400	2
VM#12	4800	2400	2	VM#25	2000	4400	2
VM#13	2000	4400	2				



Workload Condition 3

In this last workload condition, the VMs and cloudlets are scaled up to make it a more realistic scenario. In this experiment, Cloud based data center is created that has 250 VMs, each with heterogeneous RAM size, bandwidth and randomly selected mips as taken in earlier workload conditions and 100,1000 and 10000 cloudlets of different sizes (randomly selected between 30000 to 50000 instructions) are submitted to Datacenter and scheduled. The resulting figure 3 depicts the makespan. The important observation here is that some VMs may be equivalent, in terms of their computing power and bandwidth.



6. Results and Discussion

In the workload condition 1, the percentage improvement in makespan of weighted priority algorithm over random priority algorithm is 12% for 100 cloudlets, -0.68% for 1000 cloudlets and -0.80% for 10000 cloudlets. This initial jitter performance may be due to low load circumstances. Whereas in workload condition 2 of 25 VMs , the percentage improvement in makespan of weighted priority algorithm over random priority algorithm is 30.67% for 100 cloudlets, 9.52 % for 1000 cloudlets and 4.17% for 10000 cloudlets. In the workload condition 3 of 250 VMs, the percentage improvement in makespan of weighted priority algorithm over random priority algorithm is 44.02% for 100 cloudlets, 33.6 % for 1000 cloudlets and 10.59% for 10000 cloudlets. So, it is observed that the percentage improvement in the performance moves more steeply as we scale up the tasks and resources. This also reveals the important characteristic of proposed algorithm of being scalable.

7. Conclusion and Future Work

In this paper, user tasks were bound to select VMs using a set of computed priority values. These priority values were computed for both tasks and VMs considering their multiple attributes such as length, filesize, outputfilesize for tasks and MIPs, RAM, Bandwidth for VMs. Evaluation of the proposed algorithm was done through the simulation framework CloudSim 3.0.3. Results of this analysis indicate that the proposed algorithm considerably improves the makespan as compared to random priority algorithm.

The proposed algorithm can further be investigated to compute its impact over throughput rate, average resource utilization, average waiting time, average response heterogeneity of interconnection networks and various SLA parameters can also be taken into consideration for evaluation and comparison with other peer algorithms.

REFERENCES

1. S.Ghanbari and M.Othman, "A Priority based Job Scheduling Algorithm in Cloud Computing." Procedia Engineering 50 (2012): 778-785 in International Conference on Advances Science and Contemporary Engineering 2012 (ICASCE 2012).
2. J.Ru and J.Keung, "An Empirical Investigation on the Simulation of Priority and Shortest-Job-First Scheduling for Cloud-Based Software Systems", *Proc. Australian Software Engineering Conf.*, 2013, pp.78-87.
3. H.Chen, *et al.*, "User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing", in *Parallel Computing Technologies (PARCOMPTECH), National Conference, 2013*
4. G.Raj, *et al.*, "Load Balancing for Resource Provisioning Using Batch Mode Heuristic Priority in Round Robin (PBRR) Scheduling", *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)*, pp. 308-314
5. K.Dharmalingam, *et al.*, "A Threshold Based Priority Scheduling in Cloud Computing Environment", *International Journal of Innovative Science, Engineering & Technology*, Vol. 1 Issue 4, June 2014
6. Z. Lee *et al.*, "A dynamic priority scheduling algorithm on service request scheduling in cloud computing", *International Conference on Electronic & Mechanical Engineering and Information Technology*, 2011.
7. Jing Xiao, Zhiyuan Wang, "A Priority based Scheduling Strategy for Virtual Machine Allocation in Cloud Computing Environment", *International Conference on Cloud Computing and Service Computing*, 2012
8. H. Han, "A Qos Guided task Scheduling Model in cloud computing environment", *Fourth International Conference on Emerging Intelligent Data and Web Technologies*, 2013.
9. Swachil Patel, Upendra Bhoi, "Priority Based Job Scheduling Techniques In Cloud Computing: A Systematic Review", *International Journal Of Scientific & Technology Research* Volume 2, Issue 11, 2013 pp.147
10. W.Lin et al., "Bandwidth-aware divisible task scheduling for cloud computing", *Softw. Pract. Exper.* 2014; pp. 163–174
11. R.N.Calheiros, *et al.* "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," arXiv preprint arXiv:0903.2525, 2009.