Achieving Fairness and Efficiency in Cloud based Environment using new 3-Level Architecture

Aswin V
Student Third year
SSN college of engineering
Chennai, Tamil Nadu.

Karthik M
Student Third year
SSN college of engineering
Chennai, Tamil Nadu.

Balaji C
Student Third year
SSN college of engineering
Chennai, Tamil Nadu.

ABSTRACT
Cloud computing has become an integral part in evaluation of jobs over various domains present in different servers. In that case scheduling is very necessary so as to carry out the process. This paper would focus on suggesting an algorithm that would enhance the efficiency and quality of service over the applications in cloud centers. The 3-tier framework is applied over application layer (level-1, level-2) and Network layer (level-3).Thus 3-tier architecture is classified into three stages where in stage 1, every process is segregated and consolidated into separate units based on cost-time factor. In the second level the quantum value is decided according to which the process is executed in circular queues. The third stage aims at balancing the load based on the energy spent at VMdisk which is created using Paas. Hence on applying this framework we are able to achieve fairness, improvement in efficiency in case of deadlock and load balance respectively.

Keywords
Cloud computing, scheduling, cost-time factor, queuing and Energy.

1. INTRODUCTION
Cloud computing brings about connectivity in running applications that utilizes data and sources which are located in various domains. In other words, it's a nebulous assemblage of computers and servers accessed via the internet where source and user need not stay on a same platform. So cloud computing as in whole encompasses multiple jobs, clients and servers. For developers the accessibility over the cloud is vivid. The constraints and limitations are brought by providing unique license to the clients. Thus there are various types of cloud say private, public and community clouds. When the scope of the user is not limited and in case any client can use the resource in cloud then the kind of cloud that is suitable for this scenario is public cloud. There are cases where there are many levels of restrictions in usage; they are known to be private cloud. The cloud platform in which all the requirements and needs are met within an institutional boundary they are known as 'community' clouds.

Cloud as well as offers various services but pertinent to the paper's novelty we consider only Daas, Saas and Paas.

*Daas-Data as a service, the most basic need of any application is data. These data are hosted via the clouds. Thus there is a need of this service to be used so as to execute any process.

*Saas-software as a service, the scheduling algorithms are converted into codes that are kept at servers using Saas. Since the processes in level 2 are executed based on proposed Programs paradigms, there exist an essential need for Saas.

*Paas-Platform as a service, the third level that supports load balancing process is carried out over cross platform. Hence the process that is held in multiple servers must be united to shed down into many layers which can only be done using the Pass architecture.

2. NEED FOR 3-TIER ARCHITECTURE
This algorithm involves a three level approach in scheduling the process. The factors that are considered are time, cost, performance in case of deadlock and Energy. The community cloud nowadays prioritizes the products of its own provider or enhances the priority of any other third party in favor of cloud service provider's choice. This results in users being misguided towards the availability and execution of the products in cloud. Hence this paper formulates an architecture that results in a fair equity. It also helps to improve the Quality and performance of the overall process.

3. STUDY OF EXISTING ALGORITHMS
The literature review of existing cloud scheduling algorithm is thoroughly mentioned so as to prove the uniqueness of the algorithm that is suggested. The existing cloud algorithms are as follows [10]:

3.1. A Compromised-Time Cost Scheduling Algorithm
Methods - Batch mode
Parameters - Cost and time
Factors - An array of workflow instances
Environment - Cloud Computing
Tools - SwinDEW

Functions - minimize the cost under certain user designated deadlines because the corresponding algorithm for minimizing the execution time under certain user designated cost is similar.

The algorithm provides a just-in-time graph of the time-cost relationship during workflow execution in the user interface for users to choose an acceptable compromise before the next round of scheduling begins if they wish. If no user input is detected at the time of the next round of scheduling, the default scheduling strategy will be automatically applied so no delay will be caused.

This algorithm considers sharing, conflicting and competition of services caused by multiple concurrent instances running on the highly dynamic cloud computing platform.

3.2. A Particle Swarm Optimization based Heuristic for Scheduling
Methods - Dependency mode
Parameters - Resource Utilization and time
Factors - Group of Tasks

Environment - Cloud Computing
Tools - Amazon EC2

Functions - A scheduling heuristic is based on Particle Swarm Optimization (PSO). The heuristic is used to minimize the total cost of execution of scientific application work flows on Cloud computing environments. The communication cost is varied between resources; the execution cost of compute resources and compared the results against "Best Resource Selection" (BRS) heuristic. PSO based task-resource mapping can achieve at least three times cost savings as compared to BRS based mapping. PSO balances the load on compute resources by distributing the tasks the available resources.

3.3. Improved Cost-Based Algorithm for Task Scheduling

Methods - Batch mode
Parameters - Cost Performance
Factors - An array of workflow instances
Environment - Cloud Computing
Tools - Cloud Sim

Functions - This algorithm is used for making efficient mapping of tasks to available resources in cloud. The improvisation of traditional activity based costing is proposed by new task scheduling strategy for cloud environment where there may be no relation between the overhead application base and the way that different tasks cause overhead cost of resources in cloud. The algorithm divides all user tasks depending on priority of each task into three different lists. This scheduling algorithm measures both resource cost and computation performance, it also Improves the computation / communication ratio.

3.4. RASA Workflow Scheduling

Methods - Batch mode
Parameters - Make span
Factors - Grouped Tasks
Environment - Grid Computing
Tools - GridSim

Functions - The algorithm builds a matrix C where Cij represents the completion time of the task Ti on the resource Rj. If the number of available resources is odd, the Min-min strategy is applied to assign the first task, otherwise the Max-min strategy is applied. The remaining tasks are assigned to their appropriate resources by one of the two strategies, alternatively.

3.5. Innovative Transaction Intensive Cost Constraint Scheduling Algorithm

Methods - Batch mode
Parameters - Execution Cost and Time
Factors - Work Flow with large number of instances
Environment - Cloud Computing
Tools - SwinDeW-C

Functions: The primary purpose of the algorithm is to minimize the cost under certain use designated deadlines. The algorithm always enables the compromises of execution cost and time.

This algorithm takes cost and time as the main with user input on the fly and incorporates the characteristics of cloud computing.

3.6. SHEFT Workflow Scheduling (Scalable-Heterogeneous-Earliest-Finish-Time algorithm)

Methods - Dependency mode
Parameters - Execution Time and Scalability
Factors - Group of Tasks
Environment - Cloud Computing
Tools - CloudSim

Functions - SHEFT is an improvised version for HEFT algorithm. The algorithm is applied for mapping a workflow application to a bounded number of processors. At the beginning of the scheduling, any of the resources can be assigned to a task but the task with the highest priority is taken from a list where the priority of the tasks is maintained. For each resource the earliest start time and the earliest finish time is made note.

3.7. Multiple QoS Constrained Scheduling Strategy of Multi Workflows

Methods - Batch mode \ Dependency mode
Parameters - Scheduling success rate, make span, cost And time
Factors - Multiple workflows
Environment - Cloud Computing
Tools - CloudSim

Functions Workflow is done dynamically and the system has three major components such as the pre-processor, scheduler and the executer. The preprocessor has attributes related to cost and time. The preprocessor computes the time and cost surplus the workflow. The ready tasks are then sent to the queue of scheduler which re attributes and re tasks the queue. Finally The Executor selects the best service to sequential execute the tasks in the queue. When a task finishes, the Executor notifies the Pre-processor which the task belongs to of the completion status.

4. CONCEPT OF 3-LEVEL SCHEDULING AND POLICIES

The way of execution of processes that are present in different servers is the main focus of this paper. Initially all the processes are pooled into a single unit by the process scheduler. The allocation and de allocation of the process are handled by the process scheduler. The first level of scheduling brings about sorting of the process according to the cost or time factor which is decided by the user. The actually execution is dealt in the second phase where every single transition is maintained in the Vm update manager. The load balancing is done in the third level having threshold energy as the major criterion to enable layering. Consider the process P1, P2, P3, P4, P5 each individual is maintained in a unique cloudlet of its own. The local processor will use the random access memory in serving the requests. The impact over energy is mainly based on the efficiency rate at which every process is processed. The cost value is decided by the service provider.

Table 1.

PROCESS	COST	TIME	ENERGY	DISK(RAM)
P1	100	30	15	128
P2	300	60	28	256
P3	500	75	39	512
P4	450	69	34	1024

P5	600	85	53	2048

5. SCHEDULING AT LEVEL-1

Every single process that is to be scheduled is fed into the process scheduler. Now in order to allow the virtual machine to access the network so that all the process can be unified into a single unit we require Virtual ports. The Every single process in the system can be identified by the unique port hence there is a possibility that there can be many number of ports so they should be constituted into a unit called virtual machine port group. In order to create this Vsphere client is connected to the ESX Host. Thus the processes from various domains are viewed as a one single entity that are randomly connected as shown in the fig.

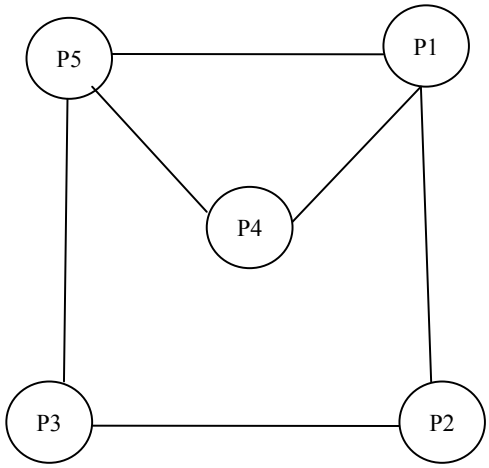


Fig 1.
Process connected randomly over common ESXi host

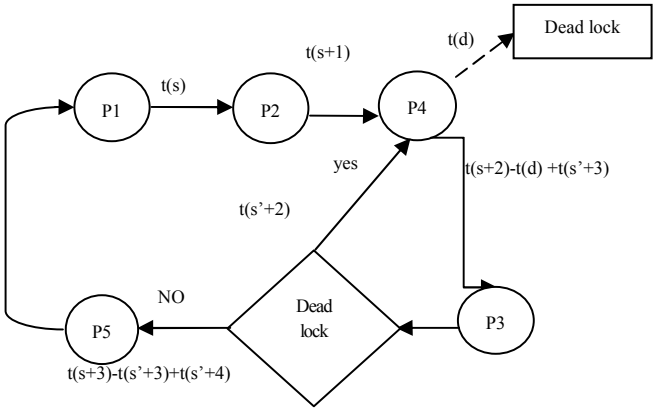
From the table it is evident that the process has its own cost and time values. At this level the choice of scheduling is left to the concern user.[4] The user can select between the cost and time according to which the order of execution is decided. This particular level aims at users being evaded to the choice of the cloud provider rather the system can be cost efficient or time efficient scheduling on the accord of the user. The order of execution:

P1->P2->P4->P3->P5

6. QUEUING AT LEVEL-2

In this level a queue [1],[2] is constructed that follows the same order of execution as that of the order that is recommended in level-1.Thus VM update manager controls the provisioning of the process. The execution of the process is carried out in circular chain. Initially the quantum of execution of every process is decided based on the measure of burst time of all the process. The quantum will be directly determined from the amount of time taken in execution of lowest time process. Thus accordingly the execution is carried out as shown in the fig below.

When there is a deadlock on execution of process the difference of existing time is calculated as nqt which is added to the quantum time on execution of the next process as shown in the algorithm. At some point of time when dead lock is re leaved then control shifts to that process and remaining part is executed totally. The difference time and actual quantum time are separately stored in VM update manager. This level brings about performance improvement on auto scalable application that would tend to spend time on deadlock which in turn affects the performance. The focus is targeted at the process that uses sources connected with other cloud environment. Fig 2.Process execution



6.1. Time Calculation

Every single process present in the cloudlet consumes its own time period for execution. The maximum time that is spent on deadlock is calculated. The deadlock time cannot be practically calculated at the same instant of execution of other event but based on statistical approach the average time that is required can be calculated. In order to generalize all possibilities of a system we assume that the execution time of P5 is less than that of the time calculated based on the algorithm.

t(s),t(s+1),t(s+2),t(s+3) = The time taken for execution of process P1 , P2, P4, P3, P5.
t(d) = deadlock time
t(s') = pending time required to execute
t(s') = t(s)-time spent on execution

When there exist a deadlock on process P4, the time spent when the process is in deadlock is determined such that the next process in the queue will be executed based on the relative time as shown in the fig. The process quantum time is dynamic that depends upon the waiting state of previous process. The process is iterated when there exists pending modules in previous process that are to be executed.

6.2. Birth and death Markov Model

Let the probability of the process to request execution in its own cloudlet be $\lambda dt(s)$. The probability that the request is satisfied be μdt. We assume that every single process uses a separate channel for execution. The deadlock probability [3] be P(d). The transition from P_i to P_j for the above process satisfies: $\lambda dt(s)\, P_{i-1} = \mu dt\, P_i$.

$\lambda t(s)P_1 = \mu t(s+1)P_2$

$\lambda t(s+1)P_2 = \mu(t(s+2) - t(d) + t(s' + 3))P_3$

Since the process P_4 is in waiting state due to deadlock.

$\lambda(t(s+2) - t(d) + t(s' + 3)P_3 = \mu(t(s+3) - t(s'+3) + t(s'+4))P_5$

$\lambda(t(s+3) - t(s' + 3) + t(s' + 4))P_5 = \mu t(s)P_1$

We know that sum of all probabilities of any event.

$$\sum_{i=0}^{n} P_i = 1$$

Such that we need to solve the process state's individual probabilities to obtain the general equation.

$$P_1 = \frac{\mu t(s+1)P_2}{\lambda t(s)}$$

$$P_2 = \frac{\mu(t(s+2) - t(d) + t(s' + 3))P_3}{\lambda t(s+1)}$$

$$P_3 = \frac{\mu(t(s+3) - t'(s+3) + t(s'+4))P_5}{\lambda(t(s+2) - t(d) + t(s'+3))}$$

$$P_5 = \frac{\mu t(s)P_1}{\lambda(t(s+3) - t'(s+3) + t(s'+4))}$$

By basis of probability, for the system considered will satisfy:

$$P_1 + P_2 + P_3 + P_4 + P_5 = 1$$

The probability of the event in deadlock,

$$P_4 = 1 - (P_1 + P_2 + P_3 + P_5)$$

Assume the time factors,

$$K = \frac{\text{time taken in the process}}{\text{time taken in previous process}}$$

$$K' = \frac{\text{algorithmic time calculation}}{\text{time taken by previous process}}$$

$$K'' = \frac{\text{algorithmic time taken by process}}{\text{algorithmic time taken by previous process}}$$

$$P_4 = 1 - (KP_2 + K'P_3 + K''P_5 + \frac{P_1}{K'})$$

Hence the general equation will be,

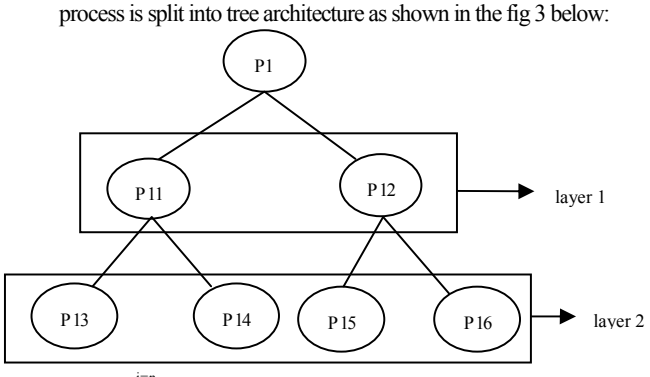
P_d = 1 – (sum of probabilities of other process in terms of algorithmic time factor)

On various case studies, it is evident that the time period taken by process that do not enter waiting will only consume less processor time when compared to deadlock process[7].

They moving that sum of probabilities to be less than one, hence the probability that the event will occur holds true in this algorithm.

7. LOAD SHARING AT LEVEL-3

The major factor that constitutes the energy consumption of the process is bandwidth allocated to the process. The average bandwidth determines the number of bits per second that will be allowed through a port.[6] The usage of resources results in transfer of bits which is directly proportional to the energy consumed by that particular process. This leads to over trafficking in execution of one single process that tend to use too many sources from multiple servers. In order to prevent over work load on one particular VM disk in a vcentre the process is split into tree architecture as shown in the fig 3 below:



$$E(P) = \sum_{i=q}^{j=n} e_{ij}$$

q → denotes the process id

n → denotes the sub layer

Thus the logical tree consists of various layers that are decided upon the usability of resource by a process. Not that every process enters the level 3 but the process that consumes energy more than the V-threshold is fed into this level. The V-threshold is defined as the maximum energy that a Vm disk can uphold in executing a process. Thus according to the above example P11, P12 occupies layer-1 which has common switching path and P13, P14, P15, P16 has got switching path of its own. The criteria in deciding the layer is based on the requirement analysis of the process. Such that, where q denotes the process id and n denotes the sub process id. In the example considered overall energy:

E(p1) = e11+e12+e13+e14+e15+e16

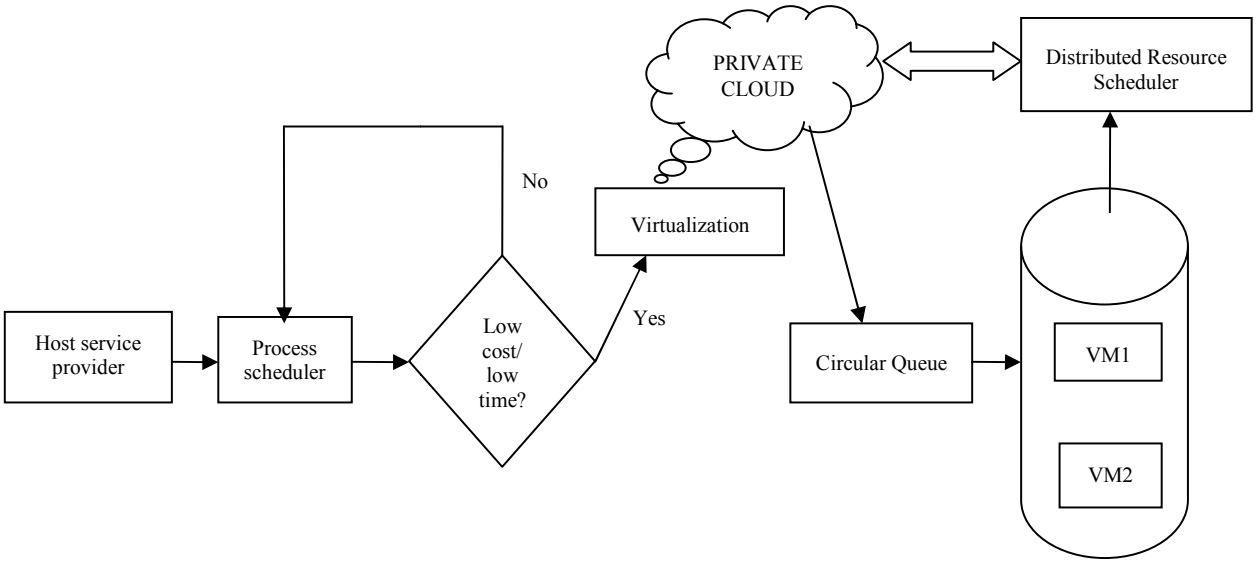
8. ALGORITHM

```
Define cost, time, Energy
Create system tray for every process in host
Input the process states into cloud scheduler
Assign n with number of process
 if(cost factor is considered)
  for i=1 to n
   mincost=cost[i]
  if(cost[i+1]<mincost)
   mincost=cost[i+1]
   end if
  end for
  Deploy process corresponding to mincost into queue
 end if

 if(time factor is considered)
  for i=1 to n
   mintime=time[i]
   if(time[i+1]<mintime)
   mintime=time[i+1]
   end if
  end for
 Deploy process corresponding to mintime into queue
 end if
Define quantum time q for execution of the process
select the process in queue based on fcfs

for i=1 to n

 Execute queue[current] for q time period
 count ET //execution time of current process
  if(deadlock arises)
    nqt=q-ET
    Execute queue[next] for (q+nqt) time period
  end if
  if(deadlock of previous state is re leaved)
   Execute the process for nqt time period
  else
  if(deadlock arises in next)
   Repeat the same steps of level-2 with new nqt
  end if
  end if
 if(process Energy>threshold)
  Construct minimum spanning tree for the process
 end if
  Select every individual sub process in tree
  Define Vm provisioning for sub each process
  Calculate sum which is total of energy of sub process
 if(Process Energy==sum)
  Execute sub process using the level-2
 else
  Assign Vm disks for the sub process remaining
  such that Energy=sum
 end if
end for
```

9. OVER ALL WORKING

The working of this frame work requires a private cloud that links the sorted process from the level1 to the queue architecture and Vm provisioning. Since the process can be

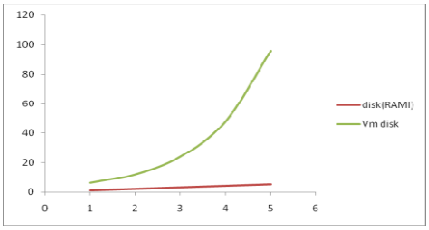
provided from any remote server virtualization is helpful in unifying the overall working. [8] The access control is imposed over the data stored in the private cloud. The steps involved in the process:



1. Every local system consists of process that is to be executed thoroughly according to the needs.
2. Every single process from various servers is fed into a process scheduler.
3. The process is sorted according to the cost or time factor and then virtualization takes place.
4. On virtualization every single process is provided a virtual memory and are taken to a private cloud.
5. Now from the cloud the execution of level-2 takes place where quantum and nqt are fixed.
6. When process energy consumption exceeds that of the threshold then they are fed into distributed resource scheduler where in level-3 execution takes place.

10. GRAPHICAL REPRESENTATION

On layering the process in third level, the process can be executed by more than one Vm drive. The efficiency in execution is directly proportional to the number of layers. Thus the graph is plotted between the rate of execution and process bundles. [11]When every single Vm disk is provided with a RAM of 128 Mb, a single process will be executed by two Vm disks at first level and four in second levels. Layering process results assignment of Vm in geometric progression but when the same process is executed by the local processor only a Single RAM will execute the entire process. It is evident that there is a clear indication of improvement in the slope of Vm curve across the local disk curve proving that efficiency is higher in the case of cloud scheduling. The reason for gradual improvement in the slope is mainly because of the larger number of Vm allocation in successive upcoming layers to the process. Rate of execution Vs process fig 5



11. ACKNOWLEDGEMENT

We extend our gratitude and thanks to our esteemed institution SSN COLLEGE OF ENGINEERING for imparting knowledge with innovation and creativity.

12. CONCLUSION

The main focus of the paper is to achieve fairness in the services that are offered to the users. The performance of the system is the backbone of any technology. So the paper provides ideologies in improving the working in case of deadlock with load balancing mechanisms being followed. The future work would aim at the security aspects in implementing this architecture.

13. REFERENCES

[1] J. Bennett and H. Zhang, Hierarchical Packet Fair Queueing Algorithms, ACM/IEEE Trans. on Networking, 5(5):675-689, Oct. 1997.

[2] L. Lenzini, E. Mingozzi, and G. Stea, Tradeoffs between Low Complexity, Low Latency, and Fairness with Decit Round-Robin Schedulers.? IEEE/ACM Trans. on Networking, 12(4):681-693, April 2004.

[3] Text book: "operating systems" by stuart E. Madnick and John j.Donovan ISBN:0-07-039455-5

[4] "ZXTM for Cloud Hosting Providers," http://www.zeus.comlc1oud

[5] computinglfor_cloud---'providers.html, January 2010.

[6] Achieving Operational Efficiency With Cloud Based Services

[7] Karthik V Bellur, Krupal M, Praveen Jain, Dr.Prakash Raghavendra The 6th International Conference on Computer Science & Education (ICCSE 2011) August 3-5, 2011. SuperStar Virgo, Singapore

[8] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," IEEE Transactions on Services Computing, pp. 266–278, 2010.

[9] M. Cardosa, M. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in Proceedings of IFIP/IEEE Integrated Network Management (IM), 2009.

[10] Sartaj Sahni, "Algorithms analysis and Design", Published by Galgotia Publications Pvt. Ltd., New Delhi, 1996.

[11] A Survey of Various Scheduling Algorithms in Cloud Environment Sujit Tilak 1, Prof. Dipti Patil 2, International Journal of Engineering Inventions, ISSN: 2278-7461, www.ijeijournal.com Volume 1, Issue 2 (September 2012) PP: 36-39.

[12] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS'09). IEEE Press, NY, USA, 2009.

[13] IEEE TPDS, MANY-TASK COMPUTING, NOVEMBER 2010 Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing.

[14] Alexandru Iosup, Member, IEEE, Simon Ostermann, Nezih Yigitbasi, Member, IEEE, Radu Prodan, Member, IEEE, Thomas Fahringer, Member, IEEE, and Dick Epema, Member, IEEE