ISSN: 2321-7782 (Online)
Impact Factor: 6.047
Volume 4, Issue 8, August 2016
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds

S. Maharajothi[1]
Research Scholar,
JJ College of Arts and Sciences (Autonomous),
Pudukkottai – India

Dr. S. Adaekalavan[2]
Assistant Professor Department of computer Applications
JJ College of Arts and Sciences (Autonomous),
Pudukkottai – India

Abstract: Cloud computing is the latest distributed computing paradigm and it offers tremendous opportunities to solve large-scale scientific problems. However, it presents various challenges that need to be addressed in order to be efficiently utilized for workflow applications. Although the workflow scheduling problem has been widely studied, there are very few initiatives tailored for cloud environments. Furthermore, the existing works fail to either meet the user's quality of service (QoS) requirements or to incorporate some basic principles of cloud computing such as the elasticity and heterogeneity of the computing resources. This work proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) clouds. We present an algorithm based on the meta-heuristic optimization technique, particle swarm optimization (PSO), which aims to minimize the overall workflow execution cost while meeting deadline constraints. Our heuristic is evaluated using CloudSim and various well-known scientific workflows of different sizes. The results show that our approach performs better than the current state-of-the-art algorithms.

Keywords: Priority Based Scheduling Algorithm; cloud computing; Scheduling Algorithm; Cloud min-min scheduling; PSO.

I. INTRODUCTION

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information are provided to users over the network. Cloud computing providers deliver application via the Internet, which are accessed from web browser, while the business software and data are stored on servers at a remote location.

we focus on the deployment and step above mentioned dynamic resources in order to achieve the purpose of the SLA agreed to the proposed algorithm now fluctuate in response to dynamic workload future tasks of low priority to high priority task, If preferably it is not possible, because the same priority, then resources to create a new form of virtual machines in the world available.These systems support scalability with respect to high input data rates over static resource deployments, assuming the input rates are stable. When the input rates change, their static resource allocation causes over- or under provisioning, resulting in wasted resources during low data rate periods and high processing latency during high data rate periods. Storm's rebalance function allows users to monitor the incoming data rates and redeploy the application on-demand across a different set of resources, but requires the application to be paused. This can cause message loss or processing delays during the redeployment. As a result, such systems offer limited self-manageability to changing data rates, which we address in this article.

Recent SPSs such as and Stream Cloud have harnessed the cloud's elasticity to dynamically acquire and release resources based on application load. However, several works have shown that the performance of public and private clouds themselves vary for different resources, across the data center, and over time. Such variability impacts low latency streaming applications, and causes adaptation algorithms that assume reliable resource performance to fail. Hence, another gap we address here is autonomic self-optimization to respond to cloud performance variability for streaming applications. In addition, the pay-per-use cost model of commercial clouds requires intelligent resource management to minimize the real cost while satisfying the streaming application's quality of service (QoS) needs.

The dynamic resource allocation based on distributed multiple criteria decisions in computing cloud is tow-fold, first distributed architecture is adopted, in which resource management is divided into independent tasks, each of which is performed by Autonomous Node Agents (NA) in ac cycle of three activities: (1) VMPlacement, in it suitable physical machine (PM) is found which is capable of running given VM and then assigned VM to that PM, (2) Monitoring, in its total resources use by hosted VM are monitored by NA, (3) In VMSelection, if local accommodation is not possible, a VM need to migrate at another PM and process loops back to into placement.

II. Literature Survey

In [9] the author proposed architecture, the use of feedback control theory, adaptive management virtualization Resources, based on the virtual machine. Virtual machine hardware resources based on all of the architecture are brought together in common Hosted applications can access the necessary resources calculated on the basis of shared space cloud infrastructure It is necessary to meet service level objectives (SLO) applications. Adaptive administrator in this framework is Multiple-input multiple-output (MIMO) Explorer, which includes 3: controller CPU, memory controller, and the controller I / S, its purpose is to regulate multi-use virtualized resources, achieved by the application by the CPU entry SLO , memory allocation and VM-control I/O Walsh et al pioneering work.

Alok Kumbhareet al proposes several simpler static scheduling heuristics that operate in the absence of accurate performance prediction model. These static and adaptive heuristics are evaluated through extensive simulations using performance traces obtained from public and private IaaS clouds. Through simulations, we showed that the proposed PLAStiCC heuristic results in a higher overall throughput, up to 20%, as compared to the reactive version of the scheduling heuristic, we proposed earlier and improved here, which performs adaptation actions after observing the effects of these variations on the application.

Thanapal.P et al a new system is proposed to allocate resources dynamically for task scheduling and execution. Virtual machines are introduced in the proposed architecture for efficient parallel data processing in the cloud. Various virtual machines are introduced to automatically instantiate and terminate in execution of job.

III. Problem Statement

Wireless sensor network coverage needs to guarantee that the region is monitored with the required degree of reliability. Locations of sensor nodes constitute the basic input for the algorithm that examines coverage of the network. Coverage problems can be broadly classified as area coverage problem and target coverage problem. Area coverage focuses on monitoring the entire region of interest, whereas target coverage concerns monitoring only certain specific points in a given region.

In recent years ad-hoc parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) clouds. Major Cloud computing companies have started to integrate frameworks for parallel data processing in their product portfolio, making it easy for customers to access these services and to deploy their programs.Expensive .Instead, these companies have popularized an architectural paradigm based on a large number of commodity servers. Problems like processing crawled documents or regenerating a web index are split into several independent subtasks, distributed among the available nodes, and computed in parallel.In recent years a variety of systems to facilitate MTC has been developed. Although these systems typically share common goals (e.g. to hide issues of parallelism or fault tolerance), they aim at different fields of application. MapReduce is designed to run data analysis jobs on a large amount of data, which is expected to be stored across a large set of share-nothing commodity servers.Once a user has fit his program into the required map and reduce pattern, the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce program.

IV. Methodology

A. Adaptive Resource Management

Resource management adaptation is also known as adaptive management. Adaptive management is a systematic process of iteration In the face of uncertainty so powerful process. Its main objective is to monitor the system, to reduce the period of uncertainty. In therefore, the decision in response to one or more resource management objectives and passive or both Active, the information needs to be acquired, terms of future management. Adaptive management is based on the learning process this is why the increase in long-term business results.

B. Service-Level Agreement

Service Level Agreement (SLA) is essentially a service contract when multiple services formally defined. Sometimes the term refers to the time when the contract SLA for service delivery or performance. As an example, an Internet service Vendors typically include the definition of the terms of their contract of service level agreements with customers Service Level (S) to be sold in plain language use. In this case the SLA will typically have a technical definition in terms of mean time between failures (MTBF),mean time to repair or mean time to recovery (MTTR); various data rates; throughput; jitter; or similar measurable details.

C. Scheduling Algorithm

In proposed priority based scheduling algorithm we have modified the scheduling heuristic for executing highest priority task with advance reservation by preempting best-effort task as done. Algorithm shows the pseudo codes of priority based scheduling algorithm (PBSA).

Scheduling is presented which helps in achieving Service Level Agreement with quick response from the service provider. In our proposed approach Quality of Service metric such as response time is achieved by executing the high priority jobs (deadline based jobs) first by estimating job completion time and the priority jobs are spawned from the remaining job with the help of Task Scheduler.

Scheduling and proposed a particle swarm optimization (PSO) algorithm which is based on small position value rule. In order to improve the efficiency the optimizing task scheduling is necessary

D. A Priority based Job Scheduling Algorithm in Cloud Computing

Proposed a new job scheduling algorithm in cloud computing by using mathematical statistics. This algorithm made its foundation on the priority property that why it is known as Priority-Based Algorithm. It is based on multiple criteria decision making model. In 1980 Thomas Saaty was first on to develop a model that make pair wise comparison based on multiple criteria and multiple attributes and named it as Analytical Hierarchy Process (AHP). AHP is purely based on Consistent Comparison Matrix, so by making the use of AHP, comparison matrices are computed according to the attributes and criteria's accessibilities. In this algorithm, each job requests a resource which has a pre-determined priority. So according to resources accessibilities, comparison matrices of each jobs is computed. Author also computes the comparison matrix of resources which will help later for jobs picking. Then author compute priority vectors (vector of weights) for each the comparison matrix and finally a normal matrix of all jobs is computed.

E. *ALGORITHM: Priority Based Scheduling Algorithm (PBSA)*

```
1. Input: UserServiceRequest
2. //call Algorithm 2 to form the list of task based on   priorities
3. get globalAvailableVMList and gloableUsedVMList and also available ResourceList from each cloud schedular
4. // find the appropriate VMList fromeach cloud scheduler
5. if AP(R,AR) != ϕ then
6. // call the algorithm 1 load balancer
7. deployableVm=load-balancer(AP(R,AR))
8. Deploy service on deployableVM
9. deploy=true
10. Else if R has advance reservation and best-effort task is running on any cloud then
11. // Call algorithm 3 CMMS for executing R
with advance reservation
12. Deployed=true
13. Else if globalResourceAbleToHostExtraVM then
14. Start newVMInstance
15. Add VMToAvailbaleVMList
16. Deploy service on newVM
17. Deployed=true
18. Else
19. queue serviceReuest until
20. queueTime > waitingTime
21. Deployed=false
22. End if
23. If deployed then
24. return successful
25. terminate
26. Else
27. return failure
28. terminate
```

F. Cloud min-min scheduling (CMMS)

Min-min scheduling is popular greedy algorithm. The dependences among tasks not careful in original min min algorithm. Thus in the dynamic min-min algorithm used , authors uphold the task dependences by updating the map able task set in every preparation step. The tasks whose precursor tasks are all assigned are placed in the map able task set.

A cloud scheduler record implementation schedule of all resources using a slot. Once an AR task is assigned to a cloud, first reserve availability in this cloud will be checked by cloud scheduler. Then best-effort task can be pre-empted by AR task, the only case once most of resources are earmarked by some other AR task. Later there are not enough resources left for this AR task in the obligatory time slot. If the AR task is not disallowed, which means there are enough resources obtainable for the task, a set of required VMs are selected randomly.

In future online adaptive procedure the remaining static resource distribution will be re-evaluate recurrently with a predefined incidence. In each reevaluation, the schedulers will re-calculate the projected finish time of their tasks. Note that a scheduler of a assumed cloud will on lyre consider the tasks that are in the jobs succumbed to this cloud, not the errands that are assigned to this cloud.

Algorithm 3 Cloud min-min scheduling (CMMS)

Require: A set of tasks, m different clouds ETM matrix

Ensure: A schedule generated by CMMS

```
1. For a mappable task set P
2. While there are tasks not assigned do
3. Update mappable task set P
4. For I = task vi ∈ P do
5. Send task check requests of vi to all other cloud schedulers
6. Receive the earliest resource available time response and And list of task with their priorities form all other cloud scheduler
7. Find the cloud Cmin(vi) giving the earliest finish time of vi, assuming no other task preempts vi
8. End for
9. Find the task-cloud pair (vk, Cmin(vk)) with earliest finish time in the pairs generated in forloop
10. Assign task v k to cloud Dmin(vk)
11. Remove v k form P
12. Update the mappbale task set P
13. End while
```

V. Algorithm Concepts

Algorithm: To compute and assign the priority for each request based on the threshold value and allocate the service to each request‘s

Step 1: [Read the clients request data i.e, time, importance, price, node and requested server name] Insert all values into the linked list

Step 2: [For each request and its tasks find the **time** priority value based on the predefined conditions] Assign priority value to each task for the client‘s request.

t_p[i] = priority value

Step 3: [For each request and its tasks find the **node** priority value based on the predefined conditions]

Assign priority value to each task for the client‘s request.

n_p[i] = priority value;

Step 4: [For each client‘s input data check whether it is within the threshold value or not]

if (input value is within the threshold limit and total node <= available node)

[Add respective computed time and node priority value and other parameters like importance and price]

Sum[k] = t_p[i] + n_p[i] + importance + price

Print ―Ready to execute available node = available node – total node

else if (input value is within the threshold limit)

sum[k] = t_p[i] + n_p[i] + importance + price

```
print ―within the limit but it is in queue
else
print ―Exeed the condition
Step 5: [Sort the sum[k] values]
Step 6: Client‘s request is ready to execute from least values of sum[k]
Stop
```

In order to run particular model huge computational resources such as server, memory in terms of storage disk, processors, software etc are needed. Also some jobs are to be executed in parallel and some others in sequential manner. In that situation job type is also very important parameter.

In a cloud environment type of user that is whether the user is internal to a cloud (in case of private cloud) or he is external to cloud(in case of public cloud) is also another important parameter to be considered during job submission. So the developed priority algorithm discusses in detail how efficiently it will help cloud admin to decide or calculate priority among the user requests.

After the successful execution of resource allocation algorithm, the jobs requested by users needs to be submitted. The main difficulties in the resource allocation in a cloud system are to take proper decision for job scheduling, execution of job, managing the status of job etc. Apart from traditional best fit and bin packing algorithm in this paper an algorithm is developed for the job allocation in the cloud environment to be decided by the cloud administrator. priority based on the client and server requirements and requests by the users. In the present algorithm to decide the resource allocation in a better and impartial way, a technique based on threshold of all the parameters (both client and server side) is considered. For example the requested number of processors cannot be more than 20 etc. (server) and a job maximum run time will be 200 hrs (user).

VI. Implementation

A. Setup

We evaluate our performance based on the priority by simulating scheduling algorithm. One by one Simulation of the working group completed in 10 games. In each execution Simulation, a group of 70 different analog service Applications (ie jobs), and each includes a service request up to 18 subtasks. We believe in Simulation of clouds. All 70 will be subject to random cloud service requests any time soon. In these requests services 70, 15 Application is in AR mode, while the rest is the best way to work with different SLA objectives. That Table 1 Parameter set randomly in simulation According to their maximum and minimum values. Since we only focus on the planning algorithms, we do simulations locally without implementing in any exiting cloud system or using VM interface API.

Table 1: Parameters and its Range

Parameter	**Minimum**	**Maximum**
No. of virtual machine in cloud	23	120
No. of CPU in a VM	1	7
Disk Space	8000	100000
Memory	16	2048
Speed	100	1000

B. Result

In Figure 1 shows the average execution job loose situation. We realized that the algorithm PBSA. The minimum average execution time. Resource Parameters when work occurs AR work best be replaced by. Such as Resource contention at least loosened, it is expected that Target part-time work is nearing completion of the actual time. Therefore Adaptation procedure does not affect the date of execution significant.

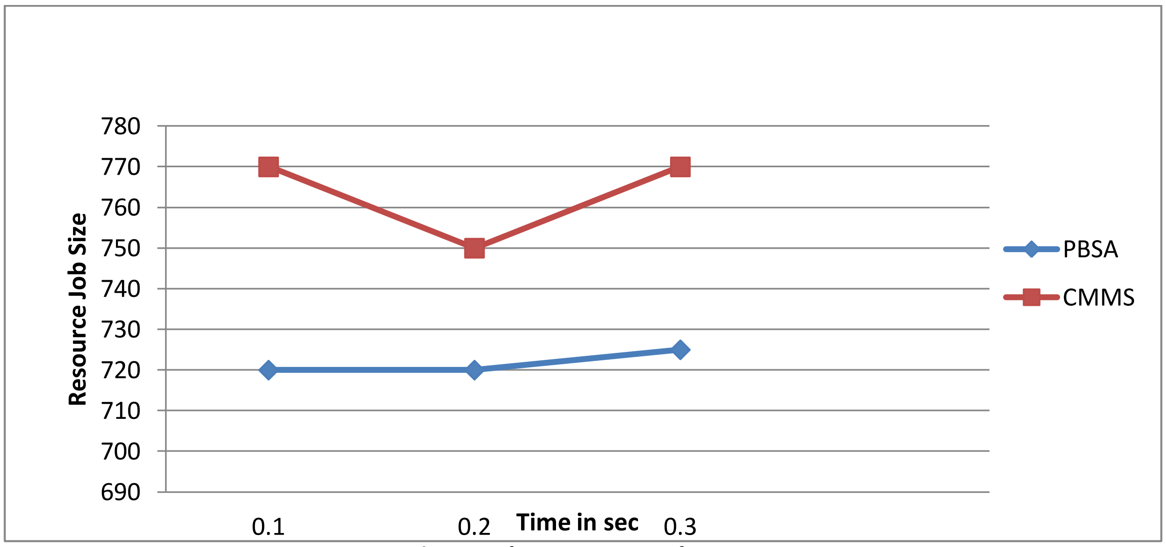
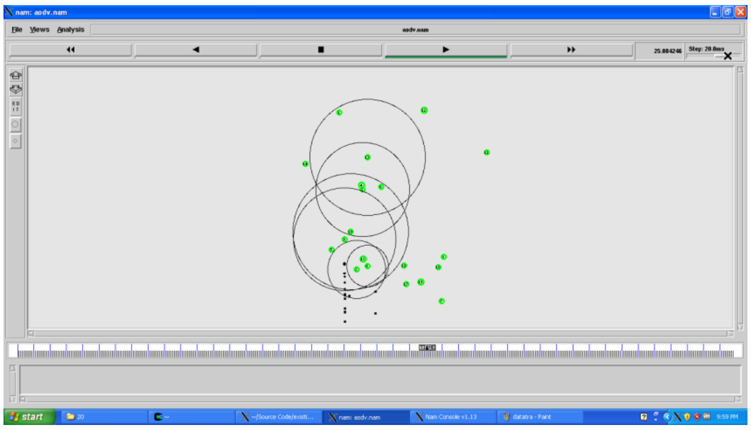


Fig 1. Average job execution time in loose situation



Under prove difficult situation shown in Figure 2 PBSA behavior CMMS better. In stressful situations the scramble for resources more so when the work actually completed it is often later than expected arrival. Because AR preemption works the best, the process of adaptation and upgrade Information more meaningful works in a difficult situation.

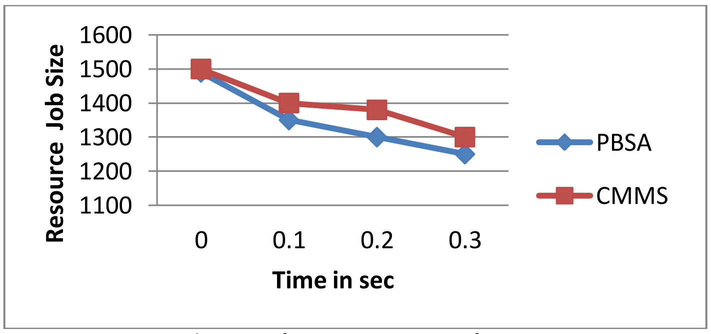
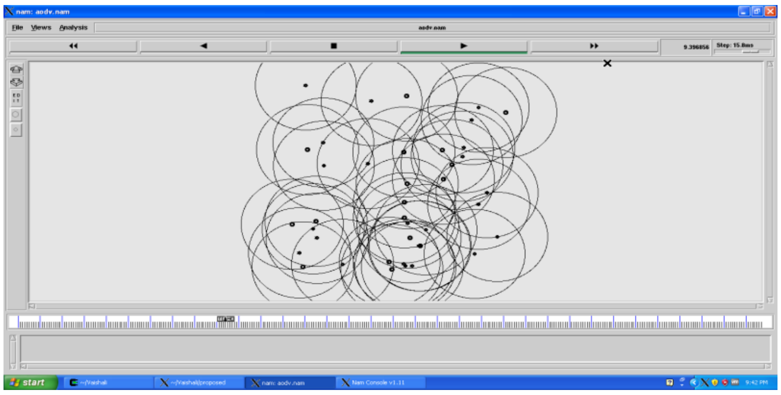


Fig 2 Average job execution time in tight situation

This requires dynamically estimating the energy consumed by methods during execution. This take inspiration from the recent PowerTutor model which accounts for power consumption of CPU, LCD screen, GPS, WiFi, 3G, and audio interfaces on HTC Dream and HTC Magic phones.

Power Tutor indicates that the variation of estimated power on different types of phones is very high, and presents a detailed model for the HTC Dream phone which is used in our experiments. We modify the original PowerTutor model to accommodate the fact that certain components such as GPS and audio have to operate locally and cannot be migrated to the cloud.



By measuring the power consumption of the phone under different cross products of the extreme power states, PowerTutor model further indicates that the maximum error is 6.27% if the individual components are measured independently. This suggests that the sum of independent component-specific power estimates is sufficient to estimate overall system power consumption.

Using this approach we devise a method with only minor deviations from the results obtained by PowerTutor. We implement this energy estimation model inside the ThinkAir Energy Profiler and use it to dynamically estimate the energy consumption of each running method.

VII. Conclusion and Future Work

Cloud computing resources means that in the selection, implementation and management time, management software (e.g., database server, load Balancers, etc.) and hardware resources (for example, CPU Storage, networking, etc.), in order to ensure security application performance. These techniques to improve response time, performance, save Energy, quality of service, SLA. The ultimate goal of resources Configuration is to maximize the benefits of the cloud Prospects for service providers and cloud the user's perspective, in order to reduce costs.

There are many current challenges Strategic resource allocation. A mechanism to overcome the challenges faced by the prior art It must be used. The architecture must be proposed it is suitable for data-intensive applications and high performance computing Also on the actual workload. Mechanism must It recommends that effective use of cloud computing resources to enable QoS and SLA violations in meeting minimization Dynamic Allocation of clouds. Also These mechanisms should also be used to configure SaaS and IaaS users.

References

1. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud computing andemerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility”, Future Generation ComputerSystems 25 (2009) 599–616.
2. S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A particle swarm optimization-based heuristic for scheduling work flow applicationsin cloud computing environments,” in AINA ’10: Proceedings of the 2010, 24th IEEE International Conference
3. onAdvancedInformationNetworking and Applications, pages 400–407, Washington, DC, USA, 2010, IEEE Computer Society.
4. M. Salehi and R. Buyya, “Adapting market-oriented scheduling policies for cloud computing,”I n Algorithms and Architectures for Parallel Processing, volume 6081 of Lecture Notes in Computer Science, pages 351–362.Springer Berlin / Heidelberg, 2010.
5. J. M. Wilson, “An algorithm for the generalized assignment problemwith special ordered sets,” Journal of Heuristics, 11(4):337–350, 2005.
6. M. Qiu and E. Sha, “Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 14, no. 2, pp. 1–30, 2009.
7. M. Qiu, M. Guo, M. Liu, C. J. Xue, and E. H.-M. S. L. T. Yang, “Loop scheduling and bank type assignment for heterogeneous multibank memory,” Journal of Parallel and Distributed Computing(JPDC), vol. 69, no. 6, pp. 546–558, 2009.

8. T. Hagras and J. Janecek, “A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems,” Parallel Computing,vol. 31, no. 7, pp. 653–670, 2005.
9. “Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control,” in First International Conference on Information Science and Engineering, April 2010, pp. 99-102.
10. W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, “Utility Functions in Autonomic Systems,” in ICAC’04: Proceedings of the First International Conference on Autonomic Computing. IEEE Computer Society, pp. 70–77, 2004.
11. M Qiu, M. Guo, M. Liu, C. J. Xue, and E. H.-M. S. L. T. Yang, “Loop scheduling and bank type assignment for heterogeneous multibank memory,” Journal of Parallel and Distributed Computing(JPDC), vol. 69, no. 6, pp. 546–558, 2009.
12. Yazir Y.O., Matthews C., Farahbod R., Neville S., Guitouni A., Ganti S., Coady Y., “Dynamic resourceal location based on distributed multiple criteria decisions in computing cloud,” in 3rd International Conference on Cloud Computing, Aug. 2010, pp. 91-98.
13. Goudarzi H., Pedram M., “Multi-dimensional SLA-based Resource Allocation for Multi-tier Cloud Computing Systems,”in IEEE International Conference on Cloud Computing, Sep. 2011, pp. 324- 331.
14. Shi J.Y., Taifi M., Khreishah A.,“Resource Planning for Parallel Processing in the Cloud,” in IEEE 13th International Conferenceon High Performance and Computing, Nov. 2011, pp. 828-833.