International Journal of Grid and Distributed Computing
Vol. 9, No. 4 (2016), pp.211-222
http://dx.doi.org/10.14257/ijgdc.2016.9.4.19

Scheduling Performance Analysis on Parameter Sweep Applications in Cloud Environments

Ning Han*, Dongbo Liu

Department of Computer Application Technology, Hunan Institute of Engineering
**linghan81@126.com*

Abstract

Recently, cloud computing has becoming a promising networking infrastructure paradigm which enable us to deploy large-scale applications in a cost-effective manner. However, many existing cloud platforms are designed for supporting commercial applications instead of large-scale scientific computing workloads. As a result, the effectiveness of running such kind of applications on cloud platforms is still an opening issue, especially when the performance penalties introduced by virtualization technology is taken into consideration. In this work, we take effects on analyzing the scheduling performance of cloud platforms for Parameter Sweep Applications (PSA), which is one of most used program models in large-scale scientific computing applications. All the experiments are conducted on our integrated performance evaluation middleware. The experimental results indicate that many conventional scheduling algorithms which are effective in classic distributed systems (i.e. grid, cluster) need to take into account the negative effects introduced by virtualization technology. In addition, the experimental results also indicate some useful hints for improving the scheduling performance of PSA workloads in cloud environments.

Keywords*: Cloud Computing; Resource Virtualization; Task Scheduling; Parameter Sweep Application*

1. Introduction

In traditional distributed systems, large-scale scientific computing applications are always the major concern as its ever-increasing resource requirements [1,2,3]. Consequently, plenty of high-performance computing platforms and systems were developed, such as cluster and grid [4,5], which are usually cost-expensive and dedicated for special groups of researchers. In recent years, cloud computing has emerged as a promising paradigm for deploying large-scale application in many commercial areas, which offers an alternative resource provision model that is resources are no longer hosted by the academic facilities but are leased from big data centers only when needed [6,7].

Many researchers have already recognized that cloud computing paradigm holds great promise for the performance-hungry scientific computing community. Firstly, cloud platforms is a cheap alternative to supercomputers and specialized clusters, and more reliable and scalable than high-performance grids [8,9]. Secondly, cloud platforms also promise to scale up instantly and temporarily within the limitations imposed only by the available financial resources [10,11]. Therefore, many scientific communities are deploying more and more their applications onto cloud-based platforms. Even so, there are still many issues and challenges that need to be addressed when using cloud platforms for running those scientific computing applications. Generally speaking, the features and requirements of commercial

Ning Han is the corresponding author.

ISSN: 2005-4262 IJGDC

workloads are quite different from that of scientific computing applications. For example, scientific computing applications generally require fixed number of dependable resources for accomplishing their goals, while commercial cloud users only care about their received results and quality of service (QoS). These differences raise an important research question, that is, can the cloud system provide sufficient efficiency for scientific computing applications just like traditional high-performance systems (*i.e.*, grid and cluster).

In this paper, we many concentrate the scheduling performance of Parameter Sweep Applications (PSA) in cloud platforms. In a typical PSA, there are plenty of loosely coupled tasks which seldom communicate with other tasks during the execution. So, the PSA programming paradigm is often named as Task-Farm or Many-Task Computing. The PSA paradigm has been widely used in modeling various scientific problems, such as Parameter Optimization [12], Engineering Design [13], DNA Analysis [14] and *etc*. The tasks are independent by each other in time and space, the key issues of scheduling them is how to improving the throughput and resource utilization with constrain to the available resources when deploying them on grid or cluster platforms [15,16]. However, the available resources in cloud platforms are elastically provisioned instead of fixed. More importantly, cloud platform often apply virtualization technology to provide underlying IT-infrastructures, which often means significant performance penalties comparing with traditional high-performance platforms. So, in this study we develop an effective performance evaluation platform, by which we can extensively investigate the scheduling performance of cloud with various benchmarks, traces and scheduling algorithms.

The remainder of this paper is organized as following: in Section 2, we summarize the related work; in Section 3, we present the framework and some special extensions of our performance evaluation platform; in Section 4, presents the experimental results and performance evaluation. Finally, Section 5 concludes the paper with a brief discussion of the future work.

2. Related Work

In the past decade, there are plenty of studies on performance evaluation and analysis for high-performance grid and cluster systems. For example, in [17] the authors presented an efficient data transfer system specifically tailored to the needs of data-intensive applications and evaluated the data transferring performance on in. An extensive experimental evaluation that carried out by means of a proof-of-concept implementation of the File Mover shows the ability of the File Mover to outperform alternative data transfer systems. In [18], the authors presented a set of components that enable deployment of overlay networks that use split-TCP connections to improve GridFTP transfer performance. Emulation demonstrates the conditions for which, splitting a TCP connection is most useful and reveals a significant source of overhead. In [19], Sanjay *et al*. developed a comprehensive set of performance modeling strategies for predicting execution times of parallel applications on both dedicated and non-dedicated environments. These strategies adapt to changing network and CPU loads on the grid resources. By extensive evaluation on various scale-size clusters with random loads and load traces from a grid system, these strategies show less than 30% average percentage prediction errors in all cases. In addition, they indicated that grid scheduling using predictions of execution times from performance modeling techniques will lead to perfect mapping of applications to resources in many cases.

With the development of cloud computing, many researchers are beginning to study the performance of real-world platforms. For example, in [20] the authors described an approximate analytical model for performance evaluation of cloud server farms and solve

it to obtain accurate estimation of the complete probability distribution of the request response time and other important performance indicators. The model allows cloud operators to determine the relationship between the number of servers and input buffer size. In [21], the authors studied the performance of a distributed Cloud Computing model, based on the Amazon Elastic Compute Cloud (EC2) architecture that implements a Gang Scheduling scheme. Extensive simulations were conducted to study, analyze, and evaluate both the performance and the overall cost of two major gang scheduling algorithms. In [22], Proposed a novel computing model (namely Cloud-Grid), able to achieve full cloud and grid integration. The security issues on the Cloud-Grid model was analyzed, and a solution based on fine-grained access control mechanisms and identity federation was proposed, which allows cooperation and interoperability among untrusted cloud resources.

3. Cloud Performance Monitor and Evaluator

Cloud Performance Monitor and Evaluator (CP-M&E) developed by our research group is an easy-to-use toolkit to evaluate or monitor the runtime performance of cloud platforms [23]. It also can be used as a cloud test-bed for performance analysis when new resource management policies or scheduling algorithms are proposed. Typically, CP-M&E is a lightweight PaaS middleware that can be easily deployed onto existing systems. Its designing is based on plug-in technology, which means all of key components can be conveniently replaced or extended in the future. In this study, we mainly use it to configure the resource allocation policy and scheduling algorithm in global meta-scheduler. To do this, we extended many components in the original implementation of our CP-M&E, which will be presented in this section.

3.1 Overall Framework of CP-M&E

The overall framework of CP-M&E is depicted in Figure 1, which is consists of four key components including *Service Portal*, *Synthetic Workload Generator*, *VM Scheduler* and *VM Manager*. The descriptions of these components are presented as following.

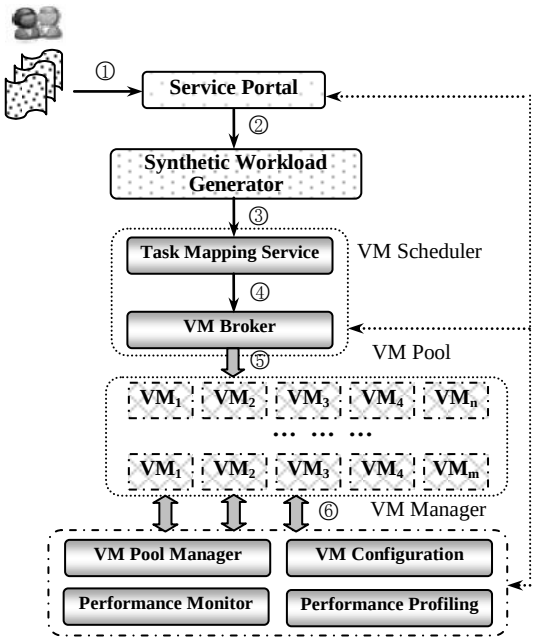


Figure 1. Framework of CP-M&E

(1) *Service Portal* is responsible for accept user's benchmark requirements which is described by a set of XML files.

(2) *Synthetic Workload Generator* is responsible for translating the user's abstract requirements into a set of workloads. The generated synthetic workloads are characterized by several factors, such as task arriving interval, task execution time, task resource requirements, task type and *etc*.

(3) *VM Scheduler* works like conventional meta-scheduler but we separate the task mapping and VM broker by two subcomponents. It is because that we can easily replace the scheduling algorithm or policy when we want to evaluate the performance of different scheduling algorithms.

(4) *VM Manager* consists of four subcomponents, and they are designed for VM resource pool management and VM performance analysis. The provision of VM is controlled by VM pool manager and VM configuration, and the runtime performance and capability of individual VM are monitored and logged through Performance Monitor and Performance Profiling subcomponents.

The CP-M&E is designed and implemented as a portable and extensible framework for generating and submitting both real and synthetic workloads to analyze the performance of cloud computing environments. In CP-M&E, VM management policy and task scheduling algorithm can be flexibly configured at runtime, in this way, researchers and administrators are able to compare the effectiveness and efficiency of different policies and algorithms.

3.2 Working Model of CP-M&E

There are six steps when using CP-M&E to do performance evaluation: 1) The user submits the benchmark files which defines all the requirements of this performance evaluation, including the task type and size, scheduling algorithm, VM configuration and *etc*, to the *Service Portal*; 2) According to the benchmark files, *Synthetic Workload Generator* will use proper random workload model to produce a set of corresponding synthetic workloads. The VM configurations specified by users are applied to corresponding subcomponents in *VM Manager*; 3) The *Task Mapping Service* schedules the synthetic workload through pre-defined task algorithms. It is noteworthy that VM resource allocation and task dispatching are carried by *VM Broker* components. So *Task Mapping Service* only produces a scheduling scheme in this step; 4) *VM Broker* executes the VM allocation and task dispatching. In this step, *VM Broker* maintains several queues of each active VM in the virtual resource pool for monitoring the availability of VMs; 5) When VM Broker obtains enough available VMs for executing current task, it dispatches the task onto one or more VMs to execute; 6) The subcomponents in *VM Manager* is always monitoring and logging the performance statistics of the VM pools. When it listens a new VM configuration requirement, corresponding actions are taken by certain subcomponents.

3.3 Working Model of CP-M&E

The workload generator component is designed to generate workloads based on the user's requirements. In this study, we want to investigate the scheduling performance under various kinds of benchmarks. So, three basic workload models, including Lublin-Feitelson, Cirne-Berman and Tsafrir-Estion are incorporated in the workload generator compoenent in CP-M&E. As cloud systems are normally defined as open and utility-based systems, which mean resource providers might share their resources with different kinds of users, including remote users and local

users. Therefore, we must take this case into consideration when generating synthetic workload so as to probably model the real world cloud environments as much as possible. So, in the implementation of our synthetic workload generator, a remote task is given a lower priority than local jobs so that the remote task is less intrusive. Given the VM is idle when a remote task arrives, the completion time of the remote task can be expressed as

$$T_{exec} = \sum_{i=1}^{n} (T_i + S_i) \tag{1}$$

where T_i is the execution time consumed by the remote task in the i^{th} time unit, S_i is the time consumed by the local task. Let ζ denote the workload of the remote task and η denote the computing capacity of a VM which is defined as the amount of workload that can be finished in a unit time. If we assume that task arrival interval follows the Poisson distribution with parameters λ and μ, by using basic probability theory, we can obtain that

$$\Pr\{T_{exec} \le t\} = \begin{cases} e^{-\frac{\lambda\zeta}{\eta}} + (1 - e^{-\frac{\lambda\zeta}{\eta}}) \Pr\left\{\sum_{i=1}^{n} S_i \le t - \frac{\zeta}{\eta}\right\} \\ 0, \quad otherwise \end{cases} \tag{2}$$

By the *little-equation* in Queue Theory, we can get the expected value and variance of $\sum S_i$ as following

$$Exp\left(\sum_{i=1}^{n} S_i\right) = \frac{1}{1 - e^{-\frac{\lambda\zeta}{\eta}}} \times \frac{\lambda}{\mu - \lambda} \times \frac{\zeta}{\eta} \tag{3}$$

$$Var\left(\sum_{i=1}^{n} S_i\right) = \frac{1 + \mu}{1 - e^{-\frac{\lambda\zeta}{\eta}}} \times \frac{\lambda\mu^2}{(\mu - \lambda)^3} \times \frac{\zeta}{\eta} \tag{4}$$

With (3) and (4), we need a distribution model of $\sum S_i$ to calculate $\Pr\{\sum S_i \le t - \zeta/\eta\}$ in (2). By extensive simulations, we notice that Gamma distribution is the best-fit distribution model to describe the $\Pr\{\sum S_i\}$. In our implementation, we use the Gamma distribution in the calculation of the distribution of the remote task completion time by (2).

4. Experiments and Performance Analysis

4.1 Experimental Setting

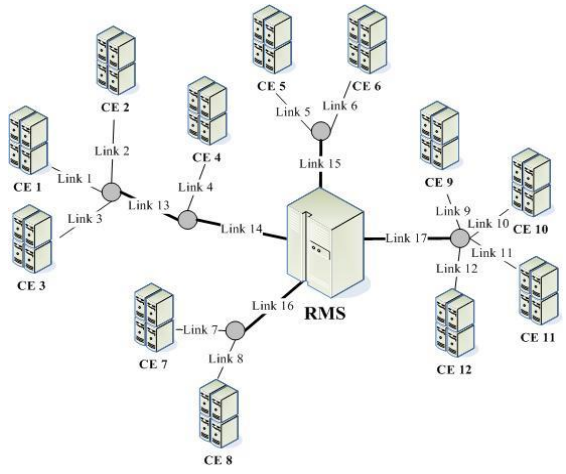


Figure 2. Networking Topology of Underlying IT-Infrastructure

The tested cloud platform is deployed in the HP High-Performance Datacenter in our institute, in which there are eleven high-performance clusters that connected by educational networks. The networking topology of the underlying IT-infrastructure of the cloud platform is shown in Figure 2. There are about 2500 PCs and 200 servers that providing physical resources for constructing the virtual machine pool as shown in Figure 1. The XCP software is used for realizing resource virtualization on each high-performance cluster. Above all the virtualized clusters, the CP-M&E middleware is deployed, which is mainly responsible for dispatching tasks in PSA onto certain VMs. In order to monitor the performance of the tested cloud platform more clearly, we also developed a friendly user client for the CP-M&E middleware, which enables to obtain various performance metrics by different ways. The main window of this user client software is shown in Figure 3.

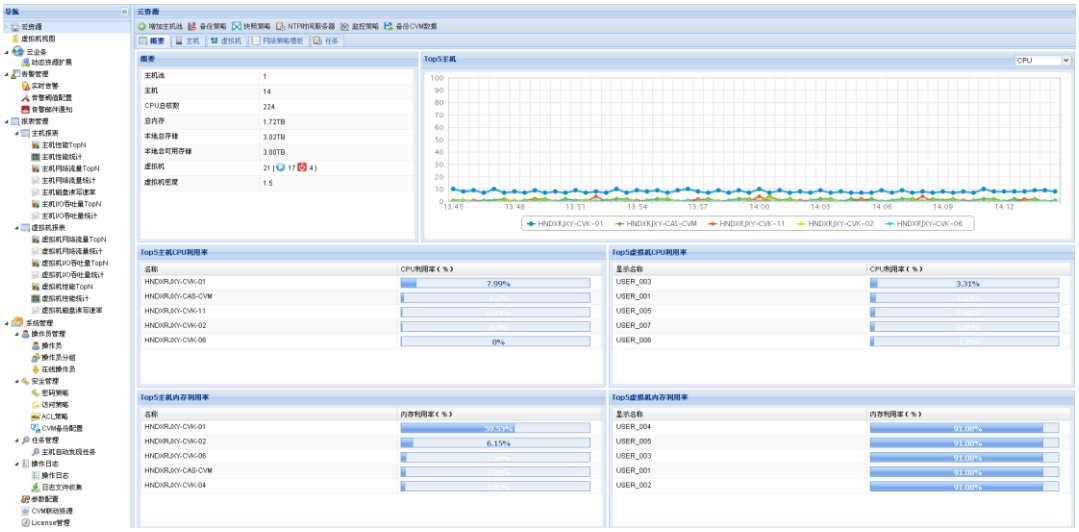


Figure 3. Main Window of the User Client for CP-M&E

4.2 Performance on Deadline-Missing Rate

When running a PSA application, Deadline-Missing Rate (DMR) is an important metric to evaluate the performance and effectiveness of a scheduling algorithm. It is because that the final execution time of a PSA depends on the successful accomplishing of all its subtasks. However, distributed systems are generally failure-prone which means that some tasks may not finish its execution strictly according to its pre-defined deadline. So, some sort of fault-tolerance is allowable when running PSAs on these systems. The DMR metric can be used for measuring that how many subtasks have violated its deadline requirement for a given scheduling policy. So, in the first experiment we evaluate the scheduling performance on the DMR metric.

In order to compare the performance of various scheduling policies, we implement four typical scheduling polices which being widely-applied in traditional grid systems, including Round Robin Policy (RR_P), Capability-based Random Policy (CR_P), Cluster Minimized Policy (CM_P), and Hybrid-Policy Co-allocation Model (HPCM). On the other side, Redundant Scheduling technique (also called K-request scheduling) has also been widely applied to improving the reliability of scheduling PSA applications. As shown in the study of [24], when Redundant redundant scheduling technology is used, K=2 or K=3 is more suitable than other higher value. Because the system's performance become almost the optimal when K=2 or K=3, and higher value of K will increase overload of the

system. So, in our experiments, we only considerate the cases of K=1, K=2 and K=3. The experimental results are shown in Figure 4.

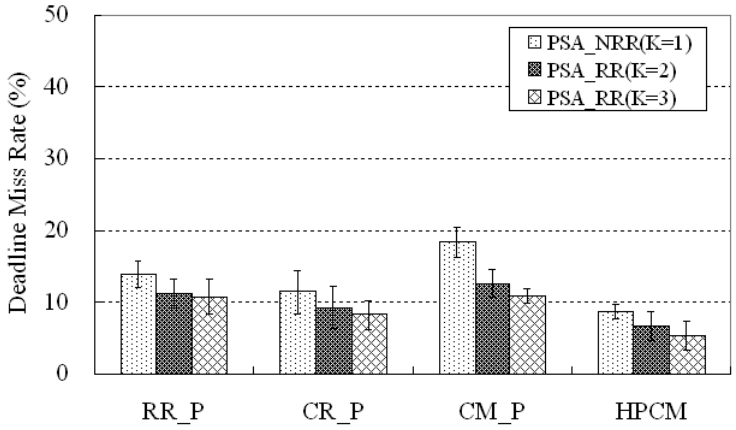


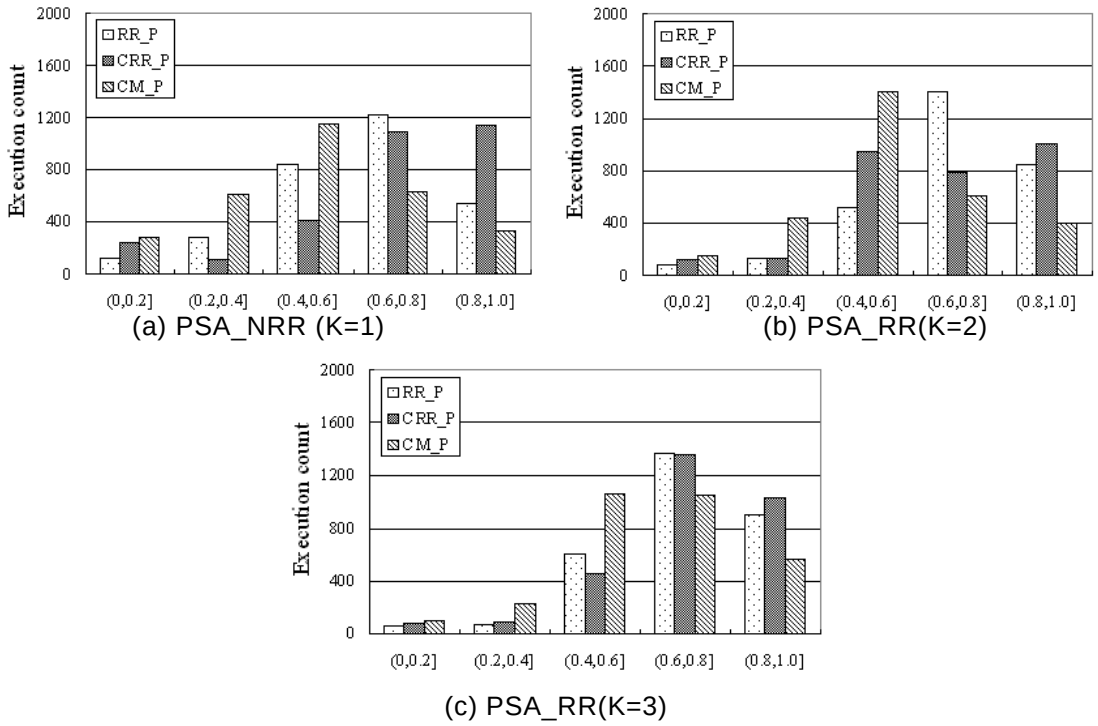
Figure 4. Performance Comparison on DMR with Different Scheduling Policies

As we can see that, CR_P is more effective to reduce DMR comparing with CM_P and RR_P, and CM_P performs worst in all experiments. After analyzing the simulative data in detail, we find that most of the tasks are scheduled on a few of computing sites (CE_3, CE_9, CE_10) when using CM_P. As the tasks in workload are all PSA type, CM_P can not fully exhibit its merits that reducing the communication overhead. It is the reason that CM_P's deadline-miss rate is the highest in all the co-allocation policies. On the other side, RR_P uniformly distributes the workload onto all computing sites, which in turn improve the PSA task's executive efficiency. During the experiment, we notice that most of the deadline-miss occur on these low-capability computing sites (*i.e*. CE_1, CE_6, CE_8) when using RR_P. On the contrary, CR_P gets over the disadvantages of both CM_P and RR_P by distributing the workload according to the capability of the computing sites, which is the main reason that CR_P is more effective than CM_P and RR_P.

As shown in the Figure 4, when redundant scheduling technology is used the deadline-miss rate of the three policies are all reduced. Among them, CM_P's improvement is most significant. It is because that redundant scheduling helps CM_P overcome its shortcoming of unbalancing-workload. As to HPCM, it evaluates the three policies' scheme in term of deadline-guarantee. So, the deadline-miss rate is significantly reduced when HPCM is used. For PSA_NRR, the deadline-miss rate of HPCM is 8.74%, which is reduced about 24% comparing even with CR_P; for PSA_RR(K=2), it is 6.68%; and for PSA_RR(K=3), it is only 5.32%. However, the results in Figure 4 only indicate the overall performance of HPCM.

Table 1. Decision Distribution of HPCM

	Deadline-Miss Counts		
	RR_P	**CR_P**	**CM_P**
PSA_NRR(K=1)	214	357	128
PSA_RR(K=2)	139	332	73
PSA_RR(K=3)	152	204	64



(a) PSA_NRR (K=1)

(b) PSA_RR(K=2)

(c) PSA_RR(K=3)

Figure 5. Distribution of Execution Count in Different DMR Range

In order to examine the detailed working of HPCM, we record all the decisions made by HPCM during the experiment. In Table 1, the distribution of HPCM's deadline-missing counts are listed in details. Also, we record the execution count according to the DMR range as shown in Figure 5(a) ~(c). It can be seen that HPCM selects CR_P as the final co-allocation policy for over 50% tasks. When redundant scheduling is used, the percentage of selecting CM_P is increased quickly, especially when K=3 the percentage of selecting CM_P is about twice as when redundant scheduling is not used. The increasing of selecting CM_P indicates that CM_P's deadline-guarantee increases quickly when redundant scheduling is used, and HPCM is adaptive to meet such situation.

4.3 Performance on Resource Utilization

In this experiment, we take efforts on investigating the resource utilization of different scheduling policies. The experimental results are shown in Figure 6.

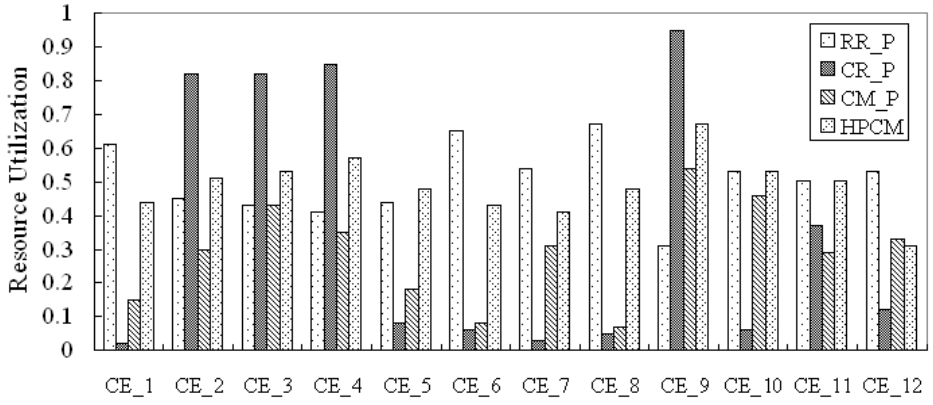


Figure 6. Resource Utilization in Different High-Performance Clusters

By the results, we can see that most of the tasks are scheduled onto a few of high-performance clusters (*i.e*. CE_2, CE_4 and CE_9) when CR_P is used. Such load-imbalance results in that CR_P's resource utilization rate is the lowest (only 12%). Even so, as CR_P takes into account the resources' static capacity while co-allocating resources, its deadline violation rate is about a half of the RR_P's, which makes its resource benefit more than RR_P's (about 20%). Meanwhile, it can be seen that CM_P outperforms HPCM in terms of resource utilization rate for the first 2000 jobs. After that, the resource utilization rate of the HPCM gradually surpasses that of CM_P, and does not fluctuate so dramatically as that of CM_P does. The reason is that HPCM co-allocates resources based on tasks utility function, so a few powerful resource that can better guarantee deadline and budget constraints have been frequently selected, which makes HPCM suffer from load-imbalance of the beginning of simulation just like CR_P. However, HPCM is capable of adjusting its retail prices and resource quantities for optimizing the benefits of both the system and clients. This feedback mechanism helps HPCM find an efficient solution to resources' deployments.

4.4 Performance Analysis with Different VM Provision

In this experiment, we change the underlying VM provision policies so as to evaluate the performance of cloud platform when running PSA applications. Generally speaking, VM provision policy is to decide that how many VMs should be provided so as to satisfy the user demands. We mainly test three policies, including Maximize Throughput Provision Policy (MTPP), Minimize Response-time Provision Policy (MRPP), Maximize Utilization Provision Policy (MUPP). In this experiment, we use HPCM as the up-level global scheduler. We used synthetic workload generator to produce four sets of workloads, and the number of tasks in each workload are 2000, 5000, 10 000 and 20000 respectively. Each task is characterized by its arrival time, resource demands, estimation of execution time and a cost budget. The performance metrics concerned include Mean Execution Time and Mean Response Time, and the results are shown in Figure 7 ~ Figure 9.

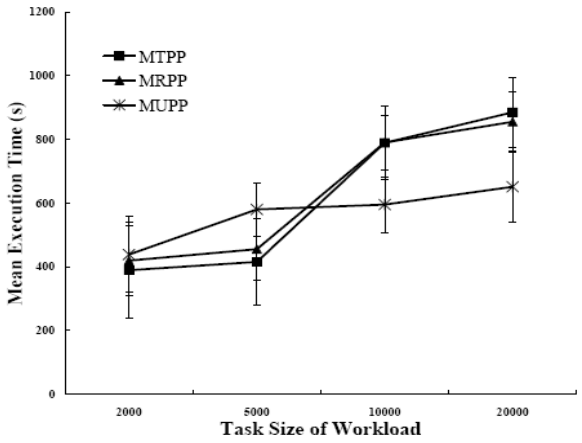


Figure 7. Performance on Mean Execution Time

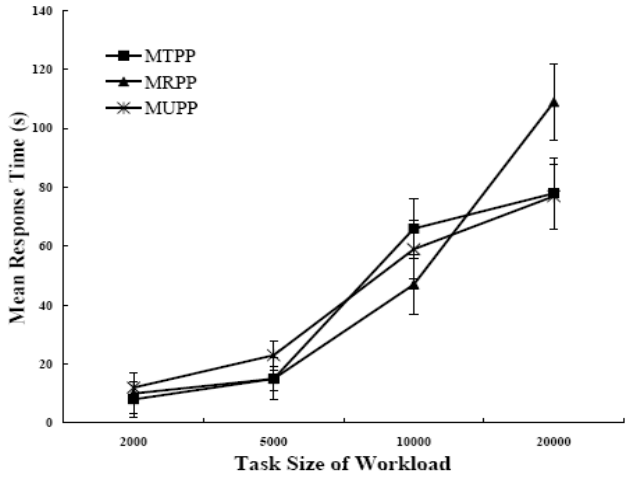


Figure 8. Performance on Mean Response Time

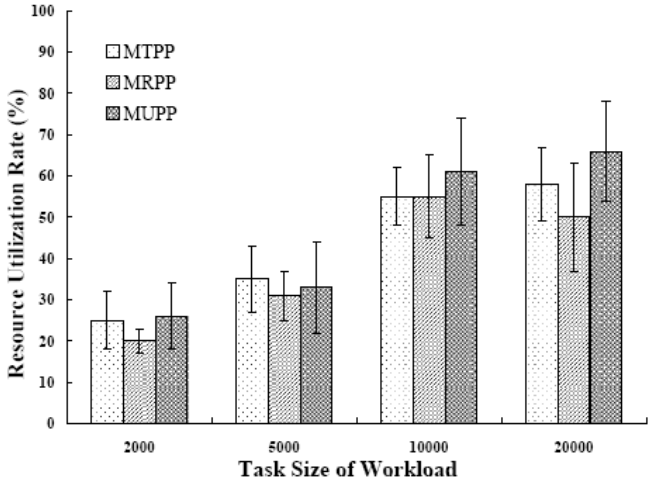


Figure 9. Performance on Resource Utilization Rate

As shown in Figure 7 and Figure 8, the MTPP policy outperforms MRPP and MUPP in term of Mean Execution Time when the size of workload is less than 10000. However, its performance reduces quickly when workload becomes heavy. On the contrary, the performance of MUPP policy seems relative stable when workload increases from 2000 to 20000. By this result, we might draw a conclusion that MUPP is more adaptive in presence of dynamical workload. With respective with Mean Response Time, MUPP also outperforms the other two policies when workload size is less than 10000. However, its performance significantly reduced when workload size increases to 20000. As we know that mean response time is decided by the task waiting time, such a result suggest that MUPP might result in some of task waiting too long in the queues, even it leads them execution more efficiently. By examine the details experimental results (not shown here because of space limitation), we found that MUPP policy tends to lead to small tasks waiting for powerful resources. So, task's execution time is reduced, but task's response time will be increased. As to the resource utilization rate (shown in Figure 9), MUPP also performs better than MTPP and MRPP. However, this performance metric seems to be affected more by the workload size. Its value significantly increases with the increasing of workload size.

6. Conclusion

In this paper, we take effects on analyzing the scheduling performance of cloud platforms for Parameter Sweep Applications (PSA). All the experiments are conducted on our integrated performance evaluation middleware, namely Cloud Performance Monitor and Evaluator (CP-M&E). The experimental results indicate that many conventional scheduling algorithms which are effective in classic distributed systems need to take into account the negative effects introduced by virtualization technology. In the future, we plan to conduct more experiments to investigate other performance metrics, such as energy-efficiency, reliability, price scheme, in cloud platforms. Also, we will investigate the scheduling performance on running scientific workflow applications.

References

[1] J.M. Brooke, M.S. Parkin.Enabling scientific collaboration on the Grid. Future Generation Computer Systems, vol. 26, no. 3, **(2010)**, pp. 521-530.
[2] W. Dou, J.L. Zhao, S. Fan. A collaborative scheduling approach for service-driven scientific workflow execution. Journal of Computer and System Sciences, vol. 76, no. 6, **(2010)**, pp. 416-427.
[3] L. Wang, M. Kunze, J. Tao. Towards building a cloud for scientific applications. Advances in Engineering Software, vol. 42, no. 9, **(2011)**, pp. 714-722.
[4] J. Montagnat, T. Glatard, I. Campos-Plasencia, *et al*. Workflow-based data parallel applications on the EGEE production Grid infrastructure. Journal of Grid Computing, vol. 6, no. 4, **(2008)**, pp. 369-383.
[5] T. Ferrari, L. Gaido. Resources and services of the EGEE production infrastructure. Journal of Grid Computing, vol. 9, no. 2, **(2011)**, pp. 119-133.
[6] W.T. Tsai, W. Wu, M.N. Huhns. Cloud-based software crowdsourcing. IEEE Internet Computing, **(2014)**, 18(3): pp. 78-83.
[7] V. Mauch, M. Kunze, M. Hillenbrand. High performance cloud computing. Future Generation Computer Systems, vol. 29, no. 6, **(2013)**, pp. 1408-1416.
[8] M. Maurer, I. Brandic, R. Sakellariou. Adaptive resource configuration for Cloud infrastructure management. Future Generation Computer Systems, vol. 29, no. 2, **(2013)**, pp. 472-487.
[9] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente. Key Challenges in Cloud Computing Enabling the Future Internet of Services. IEEE Internet Computing, vol. 17, no. 4, **(2013)**, pp. 18-25.
[10] A. Nathani, S. Chaudhary, G. Somani. Policy based resource allocation in IaaS cloud. Future Generation Computer Systems, vol. 28, no. 1, **(2012)**, pp. 94-103.
[11] M. Sedaghat, F. Hernández, E. Elmroth. Unifying cloud management: towards overall governance of business level objectives. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, **(2011)**, pp. 591-597.
[12] W. Tan, I. Foster, R. Madduri. Combining the power of taverna and caGrid scientific workflows that enable web-scale collaboration. IEEE Internet Computing, vol. 12, no. 6, **(2008)**, pp. 61-68.
[13] M. Berger, T. Fahringer. Practical experience from porting and executing the wien2k application on the EGEE production Grid infrastructure. Journal of Grid Computing, vol. 8, no. 2, **(2010)**, pp. 261-279.
[14] Y. Joshi, S. Vadhiyar, Analysis of DNA sequence transformations on grids. Journal of Parallel and Distributed Computing, vol. 69, no. 1, **(2009)**, pp. 80-90.
[15] S. Verboven, P. Hellinckx, F. Arickx, *et al*. Runtime prediction based grid scheduling of parameter sweep jobs. Journal of Internet Technology, vol. 11, no. 1, **(2010)**, pp. 47-54.
[16] J. Yan, G.-M. Tan, N.-H. Sun. Optimizing parallel S(n) sweeps on Unstructured grids for multi-core clusters. Journal of Computer Science and Technology, vol. 28, no. 4, **(2013)**, pp. 657-670.
[17] C. Anglano, M. Canonico. The File Mover: high-performance data transfer for the grid. Concurrency and Computation-Practice & Experience, vol. 20, no. 1, **(2008)**, p. 99-123.
[18] P. Rizk, C. Kiddle, R. Simmonds, *et al*. Performance of a GridFTP overlay network. Future Generation Computer Systems, vol. 24, no. 5, **(2008)**, pp. 442-451.
[19] H.A. Sanjay, S. Vadhiyar. Performance modeling of parallel applications for grid scheduling. Journal of Parallel and Distributed Computing, vol. 68, no. 8, **(2008)**, pp. 1135-1145.
[20] H. Khazaei, J. Misic, V.B. Misic. Performance analysis of cloud computing centers using M/G/m/m plus r queuing systems. IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 5, **(2012)**, pp. 936-943.
[21] I.A. Moschakis, H.D. Karatza. Evaluation of gang scheduling performance and cost in a cloud computing system. Journal of Supercomputing, vol. 59, no. 2, **(2012)**, pp. 975-992.
[22] V. Casola, A. Cuomo, M. Rak, *et al*. The CloudGrid approach: security analysis and performance evaluation. Future Generation Computer Systems, vol. 29, no. 1, **(2012)**, pp. 387-401.
[23] P. Xiao, D. Liu, P. Qu. An integrated performance evaluation middleware for virtual resources in cloud environments. International Review on Computers and Software, vol. 7, no. 2, **(2012)**, pp. 579-585.

[24] B. Nazir, K. Qureshi, P. Manuel. Replication based fault tolerant job scheduling strategy for economy driven grid. Journal of Supercomputing, vol**.** 62, no. 2, **(2012)** pp. 855-873.

Authors

Ning Han received his bachelor degree in Beijing University of Science and Technology. Currently, he works in the HP High Performance Lab of Hunan Institute of Engineering as a senior networking engineer. His research interests include complex networking analysis, distributed computing, information security technology, fault-tolerance in distributed systems.

Dongbo Liu received his doctor degree in Hunan University in 2012. Now he works in Hunan Institute of Engineering as an associate professor. His research interests include distributed system, cloud computing, high-performance application. He is now a member of CCF in China, and worked as Senior Engineer in HP High-performance Lab of Hunan Institute of Engineering.

