Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 928 – 935

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

# Cost-Based Multi-QoS Job Scheduling using Divisible Load Theory in Cloud Computing

Monir Abdullah[a], Mohamed Othman[b,1]

[a]*Department of Information Technology, Faculty of Computer Science and Information Systems, Thamar University, Thamar, Yemen*
[b]*Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor D.E., Malaysia*

**Abstract**

The advent of cloud computing as a new model of service provisioning in distributed systems, encourages researchers to investigate its benefits and drawbacks in executing scientific applications such as workflows. In this research, we attempt to investigate the use of a Divisible Load Theory (DLT) to design efficient strategies to minimize the overall processing time for scheduling jobs in compute cloud environments. We consider homogeneous processors in our analysis and we derive a closed-form solution for the load fractions to be assigned to each processors. Our analysis also attempts to schedule the jobs such a way that cloud provider can gain maximum benefit for his service and Quality of Service (QoS) requirement user's job. Finally, we quantify the performance of the strategies via rigorous simulation studies.

*Keywords:* Cloud computing, Job scheduling, Load balancing, Divisible load theory, Multi Quality of Services.

## 1. Introduction

Cloud computing is a recent trends of technology, where user can rent software, hardware, infrastructure and computational recourse as per user basis [1]. Users can submit their jobs into cloud for computational processing or leave their data in cloud for storage. Different users has different QoS requirement. cloud scheduler must be able to schedule the jobs such a way that cloud provider can gain maximum benefit for his service and QoS requirement user's job is also satisfied.

In DLT in case of clouds, an arbitrarily divisible load without having any previous relations is divided and first distributed among the various processors (for simplicity here the load is divided equally between the master computers), so that the entire load can be processed in shortest possible amount of time. An important reason for using DLT is its flexibility, tractability, data parallelism, computational difficulties [2, 3, 4].

Moreover, in order to realize the full potential of the cloud platform, an architectural framework for efficiently coupling public and private clouds is necessary. As resource failures due to the increasing functionality and complexity of hybrid cloud computing are inevitable, a failure-aware resource provisioning algorithm that is capable of attending to the end-users QoS requirements is paramount [5].

[*]Corresponding author. Tel.: +967-01-505258 ; fax: +967-06-425094 .
[1]The author is also an associate researcher at Computational Science and Mathematical Physics Lab., Mathematical Science Ins., UPM.
*E-mail address:* monir.yem@gmail.com, mothman@fsktm.upm.edu.my.

1877-0509 © 2013 The Authors. Published by Elsevier B.V.
Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.258

In the traditional networked computing systems such as grid environments, by and large, the number of CPUs available is fixed. A compute cloud being an environment that is said to offer an ”elastic service” (automatic scaling of resources as per the demand), motivates us to use a compute cloud system which is ideal for incorporating changes in resource requirements [6].

Our contributions can be summarized as follows. With the principle that all participating processors stopped computing at the same time instant, the closed-form formulas for both processing time and workload fraction for each processor are derived. We also consider cost-based multi-QoS scheduling on a compute cloud environment [7].

The rest of the paper is organized as follow: section 2 gives with related works; in section 3, a cloud scheduling environment and a set of mathematical equations has been developed to formulate the problem; in section 4 we gives the new derived DLT model to address this problem; section 5 gives the experimental results; the last part concludes with future work.

## 2. Related Works

In cloud computing, end users do not own any part of the infrastructure. The end-users simply use the services available through the cloud computing paradigm and pay for the used services. The cloud computing paradigm can offer any conceivable form of services, such as computational resources for high performance computing applications, web services, social networking, and telecommunications services [8].

DLT has proven to be a valuable tool in handling large-scale computational loads on networked systems for various aerospace data and image processing applications [6]. Although DLT uses linear modeling, recent studies also show the use of the DLT paradigm in handling computation that demands a nonlinear style of processing [9]. DLT was successfully applied for Scheduling divisible loads on large scale data Grids and produced competitive results [10, 11, 13].

Recently, DLT paradigm was investigated to design efficient strategies to minimize the overall processing time for performing large-scale polynomial product computations in compute cloud environments. A compute cloud system with the resource allocator distributing the entire load was considered to a set of Virtual CPU Instances (VCI) and the VCIs propagating back the processed results to resource allocator for postprocessing [6]. Furthermore, a programming pattern for programmers was proposed to easily develop high performance applications on dynamic and heterogeneous cloud environments using DLT paradigm [12]. This pattern uses a performance-based approach to distribute workloads within a program to working nodes to reduce scheduling overhead.

Moreover, the scheduling strategy should be developed for multiple workflows with different QoS requirements. In [7], a multiple QoS constrained scheduling strategy of multi-workflows (MQMW) was considered to address this problem. The strategy can schedule multiple workflows which are started at any time and the QoS requirements are taken into account. Here, the indivisible jobs only considered.

To the best of our knowledge, DLT model considering cost-based multi-QoS scheduling has not been applied in compute cloud. In this research, a DLT paradigm will be used to address this problem.

## 3. Scheduling Environments and Cost Models

### *3.1. Scheduling model environment*

The proposed model of scheduling environment mainly consists of five components [14]:

1. A set of users (cloud customers): There are lots of cloud end-users who want to get service from cloud. They are seeking for different kind of service (like computational service, platform service, infrastructure service) having different level of QoS (may have different cost and deadline, possibly different priority). Users can submit their job in preprocessing unit.
2. Preprocessing unit and task classifier: This unit takes user’s job and performs some preprocessing and classification. This unit consists of two sub components:
    (a) Preprocessing unit, which is responsible to perform the attributes of different job and QoS. It also encodes the attributes into Users Job Attribute Vector (UJAV). The UJAV includes Expected Instruction Count (EIC), job deadline and delay cost (Rials/ time unit).

(b) Task classifier, which classifies the task based on attributes, determine by first sub-unit. For example, job can be classified into different type based on service: Software as a Service (SaaS), Platform aaS (PaaS) and Infrastructure aaS (IaaS). Then it sends jobs to scheduler for scheduling into an appropriate Queue.

3. Data center /Executer: This a main component of which is responsible for providing user service. Data center mainly consist of a number of storage resource (storage unit and storage server), a collection of virtualized machine and a collection of computational resources (mainly processing unit). This paper considers only computational resource. Each processing unit takes job from corresponding dispatcher queue and schedule by scheduler.
4. Data center manager: It collects recent Process Unit Attribute Vector (PUAV) from different processing unit. The PUAV includes Million Instructions Per Second (MIPS) that indicate how many instructions can be executed by the machine per second and the cost of execution the instruction.
5. Job scheduler: It takes two inputs, one from preprocessing unit and other from datacenter manager, it gets information what QoS is required by a user from associated UJAV and it can also determine which processing unit can optimally satisfy that QoS from PUAV. Thus main task of the job scheduler is to perform an optimal mapping from job to processing unit as in Fig. 1.

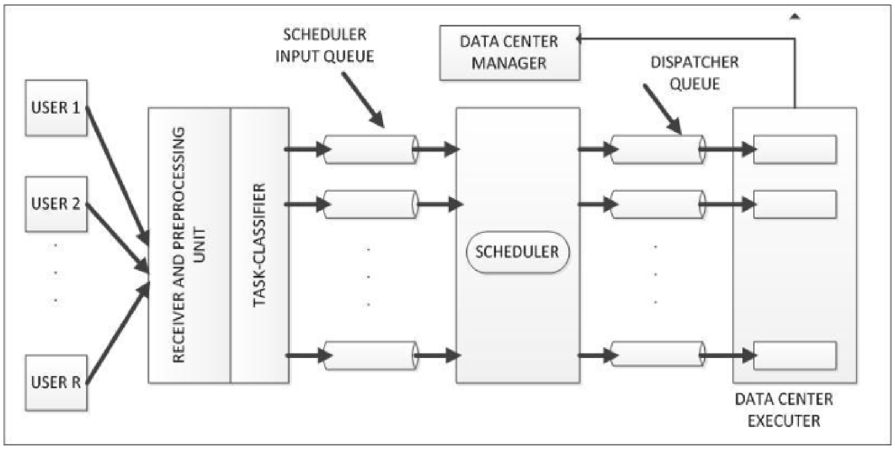

Fig. 1. Cloud Scheduling Environment

## 3.2. Optimality criterion

In all the literature related to the divisible load scheduling domain so far, an optimality criterion [2] is used to derive an optimal solution as follows. It states that in order to obtain an optimal processing time, it is necessary and sufficient that all the sites that participate in the computation must stop at the same time. Otherwise, load could be redistributed to improve the processing time. The timing diagram for this distributed system in the optimal case is depicted in Fig. 2.

## 3.3. Notations and definitions

The notations and definitions that are used throughout this paper are shown in Table 1.

## 3.4. Cost Model

Let consider the following cost factor: $\omega_i$ be the cost per instruction for processing unit $i$ and $\beta_j$ indicates the delay cost of job $j$. Suppose, $M$ machines with $N$ jobs and assign these $N$ jobs into $M$ machines ($N$=$M$), in such an order that following condition can be satisfied:

Form user side, finish time ($Tf$) must be less than the worst case completion time ($Twcc$), scheduling must be done such way to preserve QoS and to avoid possible starvation as:

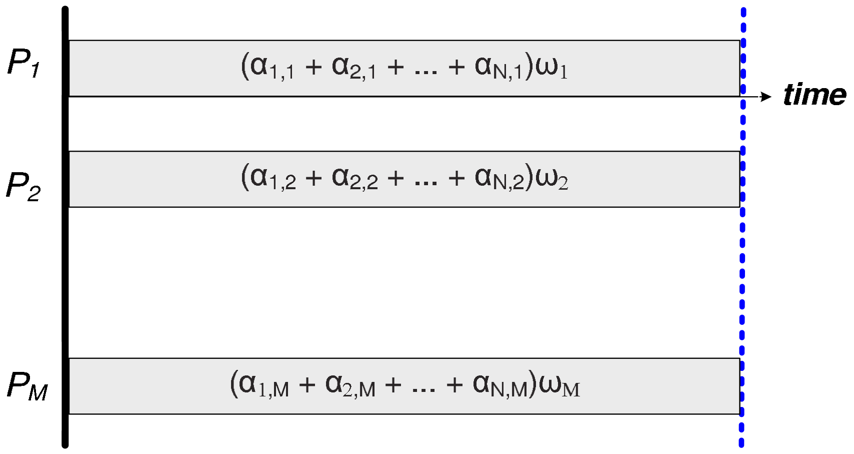
$P_1$
$(\alpha_{1,1} + \alpha_{2,1} + \ldots + \alpha_{N,1})\omega_1$
time
$P_2$
$(\alpha_{1,2} + \alpha_{2,2} + \ldots + \alpha_{N,2})\omega_2$
$P_M$
$(\alpha_{1,M} + \alpha_{2,M} + \ldots + \alpha_{N,M})\omega_M$

Fig. 2. Timing diagram of the distribution strategy with *M* Jobs and *N* Processors

Table 1. **Notations and definitions**

| Notation | Definition |
|---|---|
| $N$ | The total number of jobs in the system |
| $M$ | The Number of the processors in the system |
| $\omega_i$ | The cost per instruction for processing unit $i$ |
| $\beta_j$ | The delay cost of job $j$ |
| $EIC_j$ | Indicates the expected instruction count per job $j$ |
| $\Psi j$ | The estimated delay cost for job $j$ |
| $MIPS_i$ | Indicates how many million instructions can be executed by machine $j$ per second. |

$$Tf \leq Twcc$$

This condition must be satisfied anyhow, otherwise the job is considered as a failure job and the corresponding scheduling is illegal.

From cloud provider side, to minimize the cost spend on the job: Suppose $i^{th}$ machine is assigned to $j^{th}$ job. Then the cost for execution job $j$ is:

$$IC_j * \omega_i$$

where $IC$ is the instruction count. Let $\Psi j$ , estimated delay cost for job $j$, can be defined as:

$$\Psi_{\mathrm{J}} = \begin{cases} 0 & if \quad Td \geq Tf \\ \delta_{\mathrm{J}} * (Tf - Td) & if \quad Td < Tf \end{cases} \tag{1}$$

where, $Td$ is the deadline for job $j$ and $Tf$ is the estimated finish time, when job $j$ is assigned to processing unit $i$. Thus overall cost to execute all $M$ jobs can be given by:

$$\varsigma = \sum_{i=1}^{M}((IC_j * \omega_i) + \Psi_{\mathrm{J}}) \tag{2}$$

Thus, cloud provider's aim is to schedule jobs (i.e find a permutation: $N \rightarrow M$ such a way which minimize the value of:

$$Min(\varsigma) = min[\sum_{i=1}^{M}((IC_j * \omega_i) + \Psi_{\mathrm{J}})]. \tag{3}$$

As there are $M$ number of machines and $N$ number of jobs and assuming that all machines are capable to perform any job, then there are total $M * N$ numbers of way to assignment. And if $M = N$, then it need $M!$ assignments, which has an exponential complexity O ($M!$). Thus this problem is a kind of NP-Complete problem. A probabilistic search algorithm can solve this assignment problem in finite time. Here, we will discuss the cost model for different types of jobs.

## 4. Proposed DLT Cost model

In this section, the proposed DLT model for scheduling divisible load on cloud environment. The closed form solution for the load allocation are presented. Here, we will discuss step by step the derivation of a closed form solution by which one can calculate the optimal fraction of the load that has to be assigned to each processing node to achieve the minimum cost and the optimal data allocation for each processor. The scheduling diagrams for divisible and indivisible jobs are shown in fig. 3.

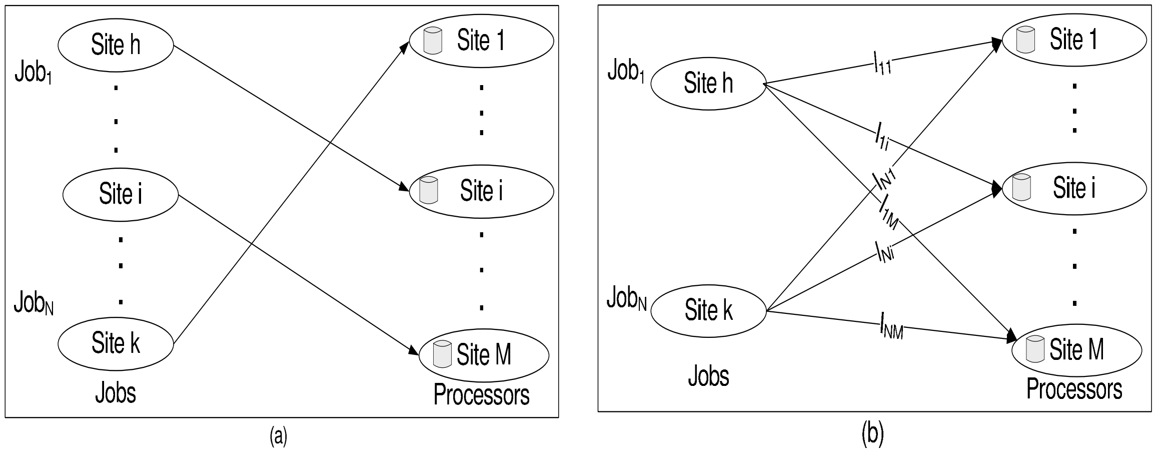

Fig. 3. (a) scheduling indivisible jobs; (b) scheduling divisible jobs.

Based on [4], the fraction of load of each processors is:

$$\alpha_i = \frac{1}{\omega_i \cdot \sum_{j=1}^{M} \frac{1}{\omega j}} \tag{4}$$

The total load that will be executed by each processor is:

$$L_i = \alpha_i \cdot \sum_{j=1}^{M} IC_j \tag{5}$$

The cost of execution load $L_i$ in processor $i$ is:

$$Cost_i = L_i \cdot \omega_i \tag{6}$$

All processing units will finish the processing at the same same based on the DLT model. Because of that, the finish time $Tf_i$ of each job will be:

$$Tf_i = \frac{\sum_{j=1}^{M} IC_j}{\sum_{j=1}^{M} MIPS_j} \tag{7}$$

The estimated delay time for job $i$, can be defined as:

$$DelayTime_i = Tf_i - Deadline_i \tag{8}$$

Here also we will implement same rule of indivisible jobs (see Equation (1)).

$$DelayCost_i = DelayTime_i * \delta_i \tag{9}$$

The total cost $Tc$ for scheduling $N$ jobs on $M$ processors is :

$$Tc = \sum_{i=1}^{M} L_i \cdot \omega_i + \sum_{j=1}^{N} DelayCost_j \tag{10}$$

## 5. Experimental Results and Discussions

To evaluate the performance of the model, it has been simulated to find best schedule for different number of jobs and different number of machines. A number of jobs having different attributes are generated randomly and also a number of processing unit having random attributes are generated randomly. We have examined the overall performance of the model by running it under 100 randomly generated cloud configurations. For instance, $N$ different jobs (20 , 50 and 100) having different characteristics are given generated randomly. Similarly $M$ different process units (10, 20, 30, 40, and 50) attributes with random characteristics are generated. When was applied the cost model, we have varied the job parameters uniformly: job deadline (1 to 10), delay cost (1 to 10), $EIC$ (100 to 1000). Also we uniformly distributed process units as: $MIPS$ (10 to 100) and $\omega$ (1 to 10).

The simulation results proved that the proposed model will give good results in terms of total cost. Thus, we will compare the performance of the model with different random configuration. The performance of the model was compared in Table 2.

Table 2. Total cost vs. no. of processors for different no. of jobs.

| No. of Processors | No. of Jobs | | |
|---|---|---|---|
| | 100 | 50 | 20 |
| 10 | 468795.84 | 127024.04 | 41921.60 |
| 20 | 254998.58 | 108169.19 | 38832.78 |
| 30 | 221457.88 | 95340.75 | 37729.90 |
| 40 | 203984.40 | 94794.23 | 37111.98 |
| 50 | 197773.22 | 92100.83 | 36940.18 |

Fig. 4 clearly demonstrated the performance of the proposed model. We can see that when the the number of nodes increases, the total cost decreases. Under all criteria, we observe that the proposed model yields the highest efficiency for any number of processing nodes.

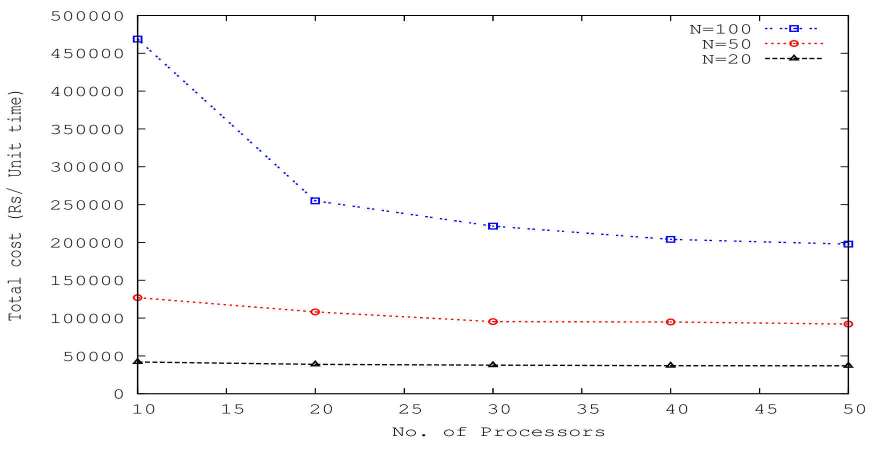

Fig. 4. Total cost versus no. of processors

Furthermore, when delay cost is only considered the proposed model produced good results. The delay cost is decreases as the number of processors increases. When the number of processing nodes approximately is more than 60, the total cost is zero for any number of jobs. The plot shows that a better results is obtained as the number of the processing nodes increases. This is expected as more processing nodes are involved in the computations. Fig. 5 is plotted delay cost against the number of processors.

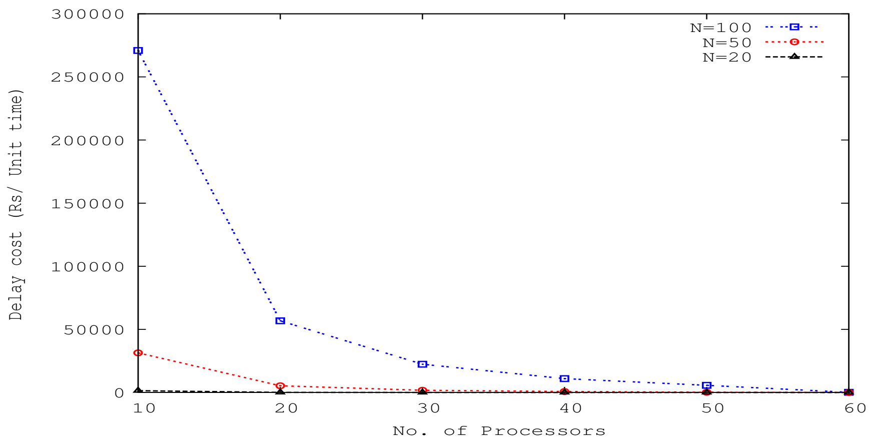

Fig. 5. Delay cost versus no. of processors

## 6. Conclusion

In this paper we have successfully employed the DLT paradigm to handle a job scheduling problem in cloud environment. The use and applicability of the DLT paradigm in a compute cloud environment is demonstrated by developing a distribution strategy that is shown to minimize the overall total cost. To this end, we designed and analyzed a closed form solution for scheduling jobs problem and validated all our findings via rigorous simulation experiments. As an immediate extension to this work one can attempt to consider the the impact of communication overheads and dynamic workload. It also not consider the more real-time job allocation restriction like political concern, machine failure. Further optimization can be done with considering these issues. With such improvements, the proposed model can be integrated in the existing cloud infrastructures in order to improve their performance.

## Acknowledgements

This work has been supported by Universiti Putra Malaysia, Research University Grant Scheme RUGS 05-01-10-0896RU/F1 and Malaysian Ministry of High Education, Fundamental Research Grant Scheme FRGS/1/11/SG/UPM/01/1.

## References

[1] T. Sridhar, Cloud Computing: A Primer, Part 1: Models and Technologies. The Internet Protocol Journal. 12 (3) (2009), 2 – 19.
[2] V. Bharadwaj, D. Ghose, T. Robertazzi, Divisible load theory: a new paradigm for load scheduling in distributed systems, Cluster Computing 6 (1) (2003), 7 – 17.
[3] A. Shokripour, M. Othman, Categorizing DLT researches and its applications, European Journal of Scientific Research 37 (3) (2009) 496 – 515.
[4] H. M. Wong, B. Veeravalli, Dantong Y., and T. G. Robertazzi, Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints, in: Proceeding of the IASTED Conference on Parallel and Distributed Computing and Systems, Marina del Rey USA, (2003), pp. 7 – 11.

[5] B. Javadi, J. Abawajyb, R. Buyya, Failure-aware resource provisioning for hybrid Cloud infrastructure, Journal of Parallel and Distributing Computing, 72(2012), 1318 - 1331.
[6] G.N. Iyer, V. Bharadwaj, S.G Krishnamoorthy. On Handling Large-Scale Polynomial Multiplications in Compute Cloud Environments using Divisible Load Paradigm, IEEE Transactions on Aerospace and Electronic Systems, 48 (1) (2012), 820 – 831.
[7] Xu, M., Cui, L., Wang , H., Bi, Y.. A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing. IEEE International Symposium on Parallel and Distributed Processing with Applications, (2009), pp. 629 – 634.
[8] M. Mezmaza, N. Melabb, Y. Kessaci b, Y.C. Lee, E. G. Talbi, A.Y. Zomayac, D. Tuyttens, A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, Journal of Parallel and Distributing Computing, 71(2011), 1497 – 1508.
[9] J. T. Hung, and T. G. Robertazzi. Scheduling nonlinear computational loads. IEEE Transactions on Aerospace and Electronic Systems, 44(3) (2008), 1169 – 1182.
[10] M. Othman, M. Abdullah, H. Ibrahim, S. Subramaniam. Adaptive Divisible Load Model for Scheduling Data-Intensive Grid Applications. Lecture Notes in Computer Science (LNCS) , Springer, Heidelberg, 4487(2007), 446 - 453. .
[11] Othman, M., Abdullah, M., Ibrahim, H., Subramaniam, S.: A2DLT: Divisible Load Balancing Model for Scheduling Communication-Intensive Grid Applications. Lecture Notes in Computer Science (LNCS) , Springer, Heidelberg, Part I. LNCS, 5101(2008), 246 - 253.
[12] Wen-Chung Shih, Shian-Shyong Tseng, and Chao-Tung Yang. Performance Study of Parallel Programming on Cloud Computing Environments Using MapReduce, International Conference on Information Science and Applications (ICISA), (2010), pp. 1 – 8.
[13] M. Abdullah, M. Othman, H. Ibrahim and S. Subramaniam, New Optimal Load Allocation for Scheduling Divisible Data Grid Applications, Future Generation Computer Systems, 7 (62) (2010), 971 – 978.
[14] D. Dutta, R. C. Joshi, A GeneticAlgorithm Approach to Cost-Based Multi-QoS Job Scheduling in Cloud Computing Environment. International Conference and Workshop on Emerging Trends in Technology (ICWET 2011), Mumbai, India, (2011), pp. 422 – 427.