ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Enhanced Load Balancing Algorithm in Three-Tier Cloud Computing

Suneel K S*, Ravichandra A J. Dr. H S Guruprasad
*PG Scholar, Dept. of CSE, BMSCE, Bangalore, India
PG Scholar, Dept. of CSE, BMSCE, Bangalore, India
Professor and Head, Dept. of CSE, BMSCE, Bangalore, India

Abstract

Load Balancing is a critical operation in cloud Computing. Load Balancing makes the efficient utilization of the resources in the cloud infrastructure resulting in the optimal pricing model for the end user and also maintaining the QoS for the end user. This paper proposes a novel algorithm for three tier cloud architecture, ELBMM which is an enhanced version of the LBMM algorithm. This algorithm reduces the average execution time of the user tasks thereby increasing the machine availability time which finally leads to the uniform distribution of the workload in the cloud infrastructure.

Keywords: Load Balancing, Three tier Cloud, Execution time, Machine availability time.

Introduction

Cloud Computing is the technology that provides services to its clients "on-demand". The basic essence of cloud computing is its capability to service its clients on demand. To a simple level, the cloud is a shared network of computer which is used by the clients of the cloud to store data and run software. It is the improvisation of the distributed computing. The entities who manage the cloud are called the cloud service providers. The cloud providers offer "Services" to their client but not product. So, the client can only use the services provided but they do not own the resources of the cloud.

A Cloud is a datacenter which contains the physical Hosts. Cloud contains the physical computing infrastructure along with the software that is used to manage their physical computing Infrastructure. Client and Cloud are connected via internet. Client of the Cloud generates tasks and sends it via internet to the cloud. Cloud is responsible for executing the task of the user. The complete environment setup for the executing task is the responsibility of cloud provider or cloud service. [1]

In real-time, the cloud can get any number of tasks. Handling these tasks, setting up the environment for the execution of these tasks should be handled by the software associated with the cloud. So, obviously same sort of load balancing algorithm must be implemented at the cloud so that it can help the proper utilization of the resources of the cloud by evenly distributing load on the resources of the cloud.

Load balancing strategy is very important in cloud computing because the arrival of tasks at the cloud is very unpredictable and cloud provides instant scale up or down of resources to its client. So, client of the cloud is not bothered about the load and hence load balancing is an integral part of the software associated with cloud.

Evolution of cloud computing

As many technologies, cloud computing is also a product of evolution. The following phases show the evolution of cloud computing. [2]

Phase 1: Mainframe Computing: This type of computing is also called the centralized computing, where in which a central mainframe is used to execute the tasks submitted by the users, who are connected to the mainframe via dumb terminals. This type of computing was used till 1981 for a time span of half century ahead. But, as the time went on, more applications, which are powerful, are developed and users started using their applications. So, the mainframe computing became obsolete. Another problem was a bug within a mainframe can cause the whole mainframe to shut down.

Phase 2: Distributed Computing: The entry of IBM PC in 1981, gave users the capacity to store their

Data and programs in their local systems and execution was carried out at their own systems. In distributed computing, each user has his own system, where he can store his own data and programs and execute the same. So centralized computing was replaced by distributed computing. This is the time when desktops came into picture.

Phase 3: Cloud Computing: As the internet technology advanced, the user requirements also got elevated. This passed the way for the new technology called Cloud Computing. Cloud Computing provided services to its customer on-demand.
The main advantage of cloud computing is that provides services in three levels, Infrastructure as a service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

There are other few advantages of cloud computing:
- *On demand self-service:* Customers of cloud can decide how much computing capability he is in need.
- *Broad-network access:* Service is offered over a network that the client can access over a network that the client can access through any standard type of devices (mobiles, laptops, PCs, tablets and so on)
- *Resource pooling:* A cloud is able to serve many clients by pooling its resources and assigning/rearranging computing them according to demand.
- *Rapid elasticity:* The cloud services are very flexible in nature, they can provide IT level scale up or down in instant time.
- *Measured Service:* The amount of cloud resources used by the client is measured, allowing user of the cloud to pay-per-process.

Literature survey

Three-Tier Architecture of the Cloud: Fig 1.1 shows the three-tier architecture of cloud.
Service node: It represents the hosts in the data-center. It executes the tasks of the user. A group of service node is maintained by a service manager. The grouping of service nodes can be customised according to the application. [3]
Service manager: Each service manager has a set of service nodes under its control. The main responsibility of the service manager are dividing the incoming tasks some logical subtasks and scheduling the name to the service nodes. [3]
Request manager: A set of service managers are controlled by an entity called request manager. Its responsibilities include the clustering or re-clustering of service nodes in service managers, assigning a cluster of service nodes to a service manager and to schedule the client task to service managers depending on various criteria's. Request manager is placed at the top of the hierarchy and the client task first interacts with the request manager in the clouds.

Multi-level hierarchical topology can decrease the data storage cost but in the meantime it also increases the network management cost [2]. Agent mechanism can be used to collect the information in this type of hierarchical architecture. Its autonomy to traverse the whole hierarchy and it is not mandatory to install it into other nodes, makes it very powerful. Agents can visit any nodes in the cloud and can extract information regarding CPU capability, network usage and so on. So, by modelling agent as a separate entity, the wastage of resources in the cloud can be addressed [3].

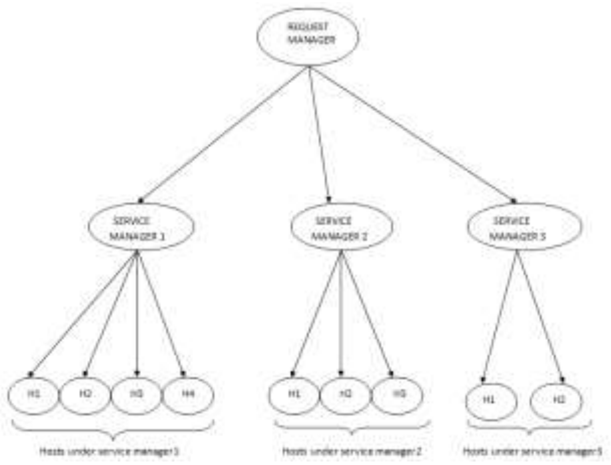
REQUEST MANAGER
SERVICE MANAGER 1
SERVICE MANAGER 2
SERVICE MANAGER 3
H1
H2
H3
H4
H1
H2
H3
H1
H2
Hosts under service manager1
Hosts under service manager2
Hosts under service manager3

Fig 1.1: Three-tier architecture of the cloud

Load balancing in cloud computing
Load balancing in a system is the process of distributing tasks over the processing components of the system such that the resources of the system are efficiently utilized. Without load balancing, as the load on the system increases, the performance of the system decreases [4].

Load balancing in cloud computing is usually implemented at IaaS level of the cloud service model. But it can also be implemented at SaaS and PaaS level also. Implementation of load balancer in cloud computing has two perspectives. First is the technical perspective, which is concerned with the time required for the execution of load balancing algorithm, responsiveness and availability of the service after implementation of load balancer. Second perspective is the Business perspective, which is concerned with the cost to implement the load balancer, additional hardware required for load balancer and so on. [5]

Basically load balancing algorithms can be designed based on two approaches. First is the static approach, which considers the previous state of the system while balancing the load. One good example for this type is the round-robin load balancing approach which divides the resources of the cloud to the incoming task on time slice basis. These type of algorithms are not widely used because it requires the prior knowledge of the system. The second type is the dynamic approach, which only considers the current state of the system for balancing the load. An example of dynamic load balancing algorithm is Artificial Ant Colony Search [5, 6].

Some of the examples of existing load balancers include DNS, Anycast, ZXTM LB [7], Amazon Load Balancing [8]. With respect to cloud, load balancing can happen at two levels. First level load balancing is when the application is loaded to the cloud. Here the load balancer is used to distribute the required number of application instances across the computers. Second level load balancing is when the uploaded application receives too many requests, the load balancer is used to distribute the requests across the application instances. [9]

Load balancing in cloud computing is not only a technical-oriented issue, but also it is an environmental-oriented issue also. As the resources are utilized efficiently, power consumed is less and correspondingly carbon emission is also less leading to the green computing concept. [10]

Heuristics for mapping tasks to the service node: A heuristics is a technique which is employed to solve a problem, when the classical technique to solve the problem is not satisfactory. Designing a heuristic technique to solve a problem requires the experience on the problem [7].

Machine availability time (MAT): It is the time span after which machine will be available for execution. [3] The main goal of load balancing is to reduce the MAT so that the availability of the machine is increased. MAT at time ‘t’ depends on the execution time of tasks before time ‘t’.

Estimated task completion time (ETCT): It is the time required to execute a task Ti on machine Mj. [7]

Completion time (CT): It is the time required to complete a task Ti on machine Mj. [7]

CT (Ti) =MAT (Mj) +ETCT (Ti, Mj).

Some of the heuristics that are used to map tasks to service node are as follows:

Opportunistic load balancing (OLB): OLB assigns each task in arbitrary order to a machine that is expected to be available regardless of the execution time of the task on that machine. [8]

Minimum Execution Time (MET): MET assigns each task to a machine in an arbitrary order such that the execution time of the task own the machine regardless of that machine’s availability [8].

MCT (minimum completion time): MCT assigns task to a machine in an arbitrary order such that the completion time of the task on that machine is less [8].

Min-Min: This technique first calculates minimum completion time of task on the machine and assigns task to the machine which has minimum completion time. [8]

Max-Min: This technique first calculates minimum completion time of task on the machine and assigns task to the machine which has maximum completion time. [8]

Proposed methodology

The proposed methodology is as follows:

Threshold of service node:

Server nodes represent hosts in the cloud. So, they must have name characteristics such as CPU capacity, network bandwidth. A threshold must be defined on these characteristics that is used as a bound to schedule the tasks on them. This is usually done by the agents of the architecture [3].

Threshold of service manager:

At logical level, service manager can be visualized as a composite service node which is made up of many heterogeneous service nodes. So, for requires threshold of service manager, the request manager requires threshold of service manager for scheduling of tasks to the service manager. [3]

Eg: Total CPU capability >=20000MIPS
Total Bandwidth>=20000KBPS
Completion time (CT) is defined as
CT (Ti) =MAT (Mj) +ETC (Ti, Mj) ... (1)
Ti-> task i from a finite set of tasks
Mj->machine j from a finite set of machines.

So from the equation (1) we can see that the estimated time for calculation cannot be reduced, because the task size and machine capacity are constant.

But the machine availability time (MAT) can be reduced by optimally scheduling the tasks to the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

service node. Here lies the criterion for the load balancing. When MAT of machine reduces then the CT of a task on that machine reduces, this in turn reduces the load on the machines, leading the machine to a state of uniform load. [4]

Proposed algorithm

The proposed algorithm is as shown below:

1) Whenever a task arrives at the request manager, get the size of the task and the thresholds of the service managers that come under request manager.
2) Depending on the threshold, the tasks are scheduled to the service manager by the request manager (if the task size is below the threshold of the service manager).
3) At service manager, divide the tasks into subtasks. Number of subtasks equals the total number of service nodes present in the service manager)
4) Create an estimated time for computation matrix for all the sub-tasks of the original task on all the service nodes in a service manager.
    a) Rows in ETC matrix are equal to total number of sub tasks.
    b) Columns in ETC matrix are equal to total number of service nodes in a service manager.
    c) Each element C_{ij} of the matrix represents the execution time of i^{th} sub-task on the j^{th} service node.
5) Now, traverse the matrix from row-wise and select a value such that the value must be minimum when compared with other values in the row. At the same time, it must be maximum value in that column. Let's say C_{ij} the selected value. Now, schedule task i to service node j.
6) Repeat step 5 until all the subtasks are assigned to service nodes.

Simulation results and analysis

CloudSim

CloudSim is a toolkit for modelling the cloud resource management and application provisioning techniques. It was developed by a bunch of cloud research scientists at The clouds lab, MIT, Melbourne, Australia in 2009. Simulators in cloud computing are very important because, it is impossible to set-up the infrastructure just required for the experimentation in cloud computing. [9]
Cloudsim is implemented in java platform. Every entity in cloudsim is modelled as a class. For example Datacenter in cloud computing is modelled as a class named Datacenter. The Cloud based applications and services are modelled in a class called Cloudlet. The main advantage of this is that researcher can run his experiment on cloud by using cloudsim, he can also use the defined classes to create his own class by simply extending the feature of old classes. This provides the flexibility for using CloudSim in different experimentations. [10]

Results

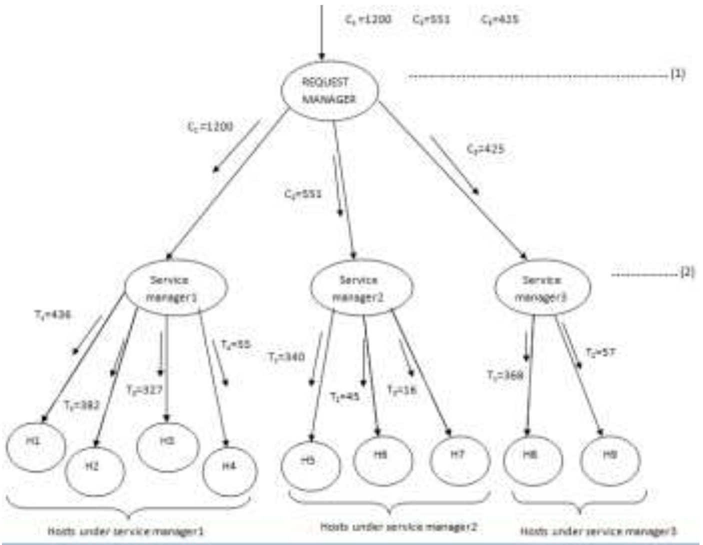

Fig 1.2: Working principle of the proposed algorithm

Fig 1.2 shows the working principle of the proposed algorithm. Tasks C1, C2, C3arrive at the request manager at a time and the request manager sends them to service managers depending on the task's requirement.

At service manager, the task is divided into sub-tasks such that they can be executed in parallel. Total no of sub-tasks generated is equal to the total no of service nodes present in the service manager. After dividing incoming task into sun-tasks, a ETC matrix is created in the service manager. This matrix is used to map sub-tasks to the service nodes. ETC matrix for incoming task C1 looks as follows:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

	Service node H1	Service node H2	Service node H3	Service node H4
Sub-task T1	0.872	1.453	4.36	1.09
Sub-task T2	0.764	1.273	3.82	0.955
Sub-task T3	0.654	1.09	3.27	0.8175
Sub-task T4	0.11	0.183	0.55	0.1375

Now Sub-task T1 is assigned to service node H1 as the execution time of T1 on H1 is minimum in its row and maximum in its column.

	Service node H2	Service node H3	Service node H4
Sub-task T2	1.273	3.82	0.955
Sub-task T3	1.09	3.27	0.8175
Sub-task T4	0.183	0.55	0.1375

Similarly T2 is assigned to service node H4

	Service node H2	Service node H3
Sub-task T3	1.09	3.27
Sub-task T4	0.183	0.55

Finally, T3 is assigned to service node H2 and T4 to H3.This is how the algorithm works. This process is repeated for all the cloudlets at the service manager.

The following graphs show the results of the simulation for the proposed method of load balancing.

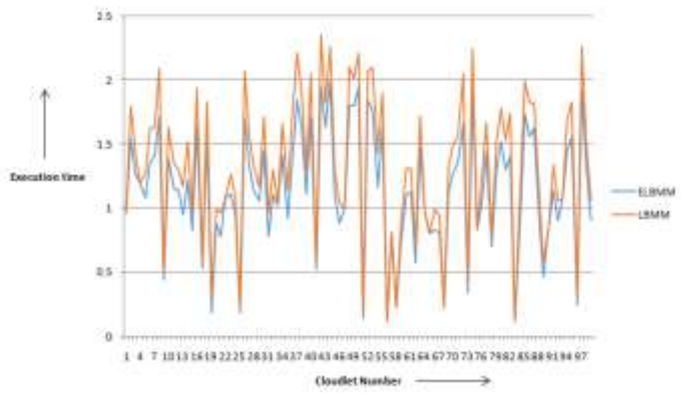

Fig 1.3: Cloudlets Vs execution time

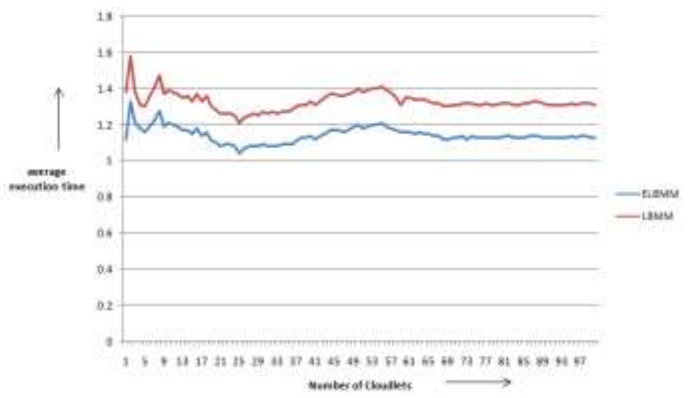

Fig 1.4: Number of Cloudlets Vs their average execution time

As explained in eqn (1), the completion time of a task depends on its own execution time and machine availability time. Machine availability time of a machine is directly proportional to the execution time of the previously assigned tasks to the machine. As and on the execution time of the tasks decreases, the machine availability time also decreases. Decreasing execution time of tasks correspond to the efficient scheduling decision (effective utilization of resources). So from the graph it is evident that the proposed algorithm works better than the algorithm in [3].

The graph in Fig 1.3 shows the execution time of cloudlets. The inference from this graph is that the LBMM algorithm defined in [3] takes more time to execute than the proposed algorithm. The graph defined in Fig 1.4 shows the behaviour of the proposed algorithm as the cloudlets increase in number. The average execution time of the cloudlets is plotted against the number of cloudlets. From this graph also, it is evident that the proposed algorithm has less average execution time than the LBMM algorithm defined in [3].

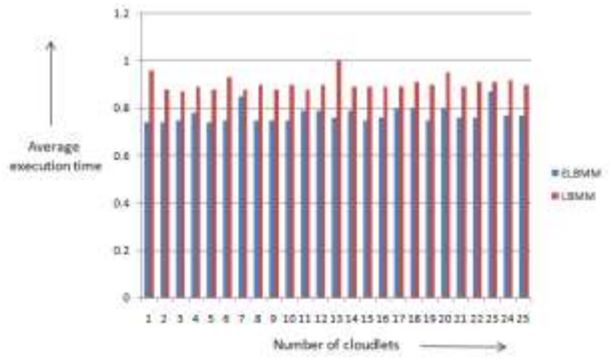

Fig 1.5: Average execution time of Cloudlets at Service manager 1 Vs number of Cloudlets

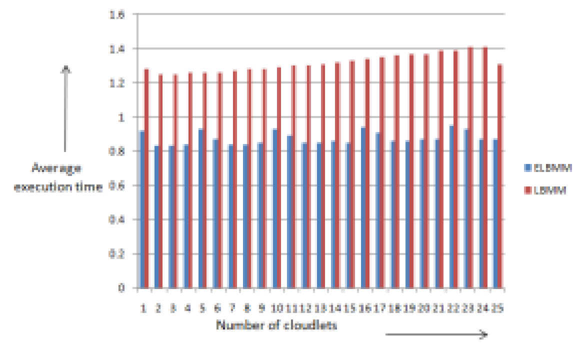

Fig 1.6: Average execution time of Cloudlets at service manager 2 Vs number of Cloudlets

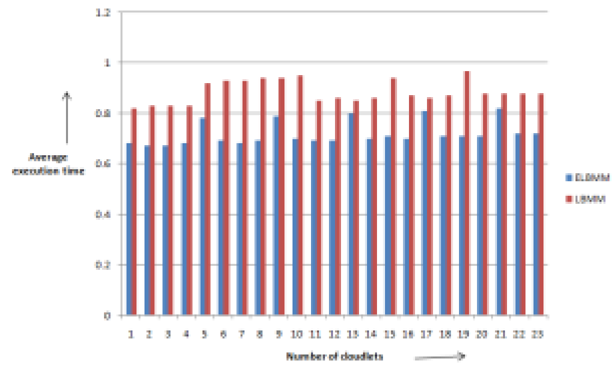

Fig 1.7: Average execution time of Cloudlets at Service manager 3 Vs number of Cloudlets

The graphs at Fig 1.5, 1.6, 1.7 show the average execution time of cloudlets at different service managers. From these graphs it is evident that the scheduling decision that is incorporated in service manager is of very importance in reducing the execution time of the cloudlets.

Conclusion

The proposed methodology and the algorithm ELBMM has shown improved results over the LBMM algorithm [3]. This algorithm also increases the overall efficiency of the utilization of the cloud infrastructure. The simulation results prove that the algorithm works optimally in all the different workload scenarios.

ACKNOWLEDGEMENT

"The authors would like to acknowledge and thank Technical Education Quality Improvement Programme [TEQIP] Phase 2, BMS College of Engineering and SPFU [State Project Facilitation Unit], Karnataka for supporting the research work"

References

1. Dan C Marinescu "Cloud Computing Theory and Practices".
2. Rabi Prasad Padhy, Manas Ranjan Patra,"Evolution of Cloud Computing and Enabling Technologies", International Journal of Cloud Computing and Services Science, Vol.1, No.4, October 2012, ISSN: 2089-3337, pp 182-198.
3. Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", International Conference on Computer Science and Information Technology [ICCSIT], Vol. 1 , 9-11 July 2010, ISBN- 978-1-4244-5537-9, pp 108-113.
4. Amritpal Singh, Rajeev kumar bedi, Sunil kumar gupta, "Design and implementation of an efficient scheduling algorithm for cloud balancing in Cloud Computing", International Journal of Emerging Trends and Technology in computer Science [IJETTCS], Volume 3, Issue 1, Jan-Feb 2014, ISSN- 2278-6856.
5. A Khiyaita, M Zbakh, H El Bakkali, D El Kettani, "Load balancing cloud computing: state of art," 2nd National Days of Network Security and Systems [JNS2], 20-21 April 2012, Morocco, pp 106-109, DOI: 10.1109/JNS2.2012.6249253.
6. Ruixia Tong, Xiongfeng Zhu, "A Load Balancing Strategy Based on the Combination of Static and Dynamic", 2nd International Workshop in Database Technology and Applications (DBTA), 27-28 Nov 2010, Wuhan, pp 1-4, DOI: 10.1109/DBTA.2010.5658951.

7. ZXTM Load Balancer: Features and Specifications, Zeus Technology 2009;
8. http://aws.amazon.com/fr/elasticloadbalancing/, Elastic Load Balancing;
9. Jitendra Bhatia, Tirth Patel, Harshal Trivedi, Vishrut Majmudar, "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud", International Symposium on Cloud and Services Computing, 17-18 Dec 2012, Mangalore, pp 15-20, DOI: 10.1109/ISCOS.2012.25.
10. Lori MacVittie, "Cloud Balancing: The Evolution of Global Server Load Balancing", pp 1-12. www.f5.com, F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447.
11. Yatendra Sahu, R K Pateriya, Rajeev Kumar Gupta, "Cloud Server Optimization with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm", 5th International Conference on Computational Intelligence and Communication Networks, 27-29 Sept 2013, Mathura, pp 527-531, DOI: 10.1109/CICN.2013.114.
12. Nidhi Jain Kansal, Inderveer Chana, "Cloud Load Balancing Techniques: A Step Towards Green Computing", International Journal of Computer Science Issues, Vol 9, Issue 1, No. 1, Jan 2012, ISSN (Online): 1694-0814.
13. Tracy D. Braun, Howard Jay Siegel, Noah Beck, Lasislau L Boloni, Muthucumara Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, Richard F Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, Vol 61, Issue 6, June 2001, pp 810-837, DOI: 10.1006/jpdc.2000.1714.
14. R Armstrong, D Hensgen, T Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions", 7th Heterogeneous Computing Workshop, 30 Mar 1998, Orlando, pp 79-87, DOI: 10.1109/HCW.1998.666547.
15. Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A F De Rose, Rajkumar Buyya, "CloudSim: A Toolkit for the Modeling and Simulation of Cloud Resource Management and Application Provisioning Techniques", Journal of software Practice and Experience, Vol 41, Issue 1, Jan 2011, pp 23-50, DOI: 10.1002/spe.995.
16. Rajkumar Buyya, Rajiv Ranjan, Rodrigo N Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", 7th High Performance Computing and Simulation Conference, Leipzig, Germany, June 21-24, 2009, ISBN: 978-1-4244-4907-1.