AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Priority Based Workflow Task Scheduling In Cloud Computing Environments

[1]Kumar P and [2]Sheila Anand

[1]*Associate Professor, Rajalakshmi Engineering College, Chennai, India*
[2]*Dean (Research) Computer Studies, Rajalakshmi Engineering College, Chennai, India*

ARTICLE INFO

Article history:
Received 25 August 2014
Received in revised form
19 September 2014
Accepted 29 October 2014
Available online 16 November 2014

Keywords:
Cloud Computing, Scheduling, Workflow, Priority, ABC

ABSTRACT

Cloud Computing delivers computing resources as a service over a network (Internet) to the customers. The tasks or jobs of the users would require to be executed in a particular order to complete the whole task. Workflow scheduling manages the execution of the inter-dependent tasks on the distributed resources. Workflow scheduling algorithms are used to allocate the resources to workflow tasks in a manner that preserves the dependency constraints. At the same time, the tasks must be scheduled efficiently in order to minimize the execution time as well as cost incurred in using the heterogeneous resources of the cloud. This paper proposes a multiple criteria decision making model for scheduling tasks based on priority and cost. This paper extends our earlier work on Artificial Bee Colony algorithm to schedule workflow job based on assigned priority while optimizing the cost and execution time.

To Cite This Article: Kumar P and Sheila Anand, Priority Based Workflow Task Scheduling In Cloud Computing Environments. ***Aust. J. Basic & Appl. Sci.,*** *8(17): 532-539, 2014*

INTRODUCTION

Cloud Computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources, such as, networks, servers, storage, application and services (Amandeep Verma, 2014). Cloud services can be rapidly provisioned and released with minimum management effort or service provider interaction. The services are commoditized and delivered in a manner similar to traditional utilities such as water, electricity, gas, and telephony. In such a model, users can access services based on their requirements without regard to where the services are hosted or how they are delivered. Several computing paradigms have promised to deliver this utility computing vision and these include cluster computing, Grid computing, and more recently Cloud computing.

Cloud computing architecture typically consists of a front end and a back end connected by Internet or Intranet networks. The front end comprises of client devices which can be thin client, fat client or mobile devices (Dervis Karaboga, 2007). The clients need some interface and applications for accessing the cloud computing system. The back end consists of the various servers and data storage systems. A central server is typically used for administering the cloud system which includes monitoring the overall traffic and fulfilling the client demands in an efficient manner.

Applications hosted and executed using clouds are often composed from a set of services which form a workflow. Workflow processing requires tasks to be executed based on their control and data dependencies. As workflow scheduling is a well-known (Elzeki, 2012) NP-complete problem, many heuristic and meta-heuristics methods have been proposed for distributed systems like grids. Users may not always need to complete workflows earlier than they require and instead, may prefer to use cheaper services with lower Quality of Service (QoS) that are sufficient to meet their requirements.

Workflow scheduling is one of the key challenging issues in the cloud workflow systems. It allocates suitable resources to workflow tasks such that the execution can be completed to satisfy objective functions imposed by users. The scheduling process maps and manages the execution of the inter-dependent tasks on the distributed resources. Scheduling of workflows is a challenging task especially when many workflows are considered. Scheduling algorithms are required to implement the workflow scheduling strategies and also for automating the process of scheduling. Proper scheduling can have significant impact on the performance of the system and the user's service requests have to be provisioned with minimal management effort or service provider interaction.

Corresponding Author: Kumar P., Associate Professor, Rajalakshmi Engineering College, Chennai, India

This paper proposes a workflow scheduling algorithm that is based on the multiple criteria of job priority and optimized execution time. The remaining paper is organized as follows. The related work in the area of workflow scheduling for distributed computing is presented under Section 2 "Related work". The proposed work is discussed in the Section 3 "Proposed Workflow Scheduling". The proposed algorithm has been implemented and tested and the results has been presented in the Section 4 "Performance Evaluation". Section 5 Conclusion and Future Work concludes the paper.

Related work:

Cloud Computing helps user tasks to dynamically provision computing resources at specified locations. Workflow scheduling plays a vital role in the workflow management. Proper scheduling can have significant impact on the performance and service utilization.

Workflow applications can be commonly represented or modelled as a Directed Acyclic Graph (DAG), defined by a tuple G (T, E), where T is the set of n tasks {t1, t2,.....,tn}, and E is a set of e edges, represent the dependencies. Each t_i ε T, represents a task in the application and each edge (t_i.....t_j) ε E represents a precedence constraint, such that the execution of t_j ε T cannot be started before t_i ε T finishes its execution [19]. If (t_i, t_j) ε T, then ti is the parent of tj, and tj is the child of ti. A task with no parent is known as an entry task and a task with no children is known as exit task. This is illustrated with an example shown in Figure 1.

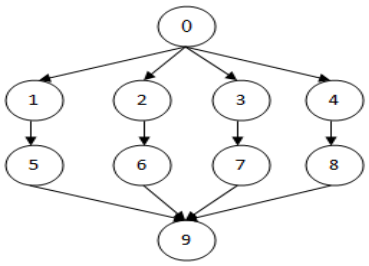

Fig. 1: Workflow Application Example.

Figure 1 shows the dependencies among the different tasks shown as workflow graph G. Task 0 is an entry task as it has no parents. Task 9 is the exit task as it has no children. Task 0 is the parent of child tasks 1, 2, 3 and 4. The parent task has to be executed before the child tasks and the output of parent node acts as an input to child node. Task 9 can be executed only after the completion of tasks 5, 6, 7 and 8. The different tasks have to be allocated resources for their execution.

The workflow scheduling algorithms proposed for Grid Computing have been primarily classified into two basic categories: Best effort based scheduling and QoS based scheduling. In traditional community based computing paradigms, best effort based scheduling strategies are often applied to only minimize the execution time without considering the cost of resources or user task priorities.

Amandeep Verma has proposed a scheduling algorithm to schedule workflow tasks over available cloud resources that minimizes the execution cost for a given deadline and budget constraints. The focus is not on total time taken for execution of the jobs. Fitness of individual is calculated based on the cost. Priority is assigned only for first task, other tasks are scheduled randomly. This work has been extended (Ke, 2010) to schedule workflows so as to optimize the total cost within the user's specified budget. Each workflow's task is assigned priority using bottom level (b-level) and top level (t-level). Computation of b-level and t-level for assigning priority increases the overall execution time of the associated tasks.

Workflow scheduling based on QoS parameters has been proposed. The QoS parameters suggested include Cost, time and resource utilization. It is based on multiple criteria decision making model. Here, each job requests a resource with determined priority. So comparison matrices of each jobs according to resources accessibilities is computed. For each of the comparison matrices priority vectors (vector of weights) are computed. The next step is to compute Priority Vector of S (PVS), where S is set of jobs. PVS is calculated by multiplying priority vectors and comparison matrices. The final step is to choose the job with maximum calculated priority, so a suitable resource is allocated to that job. The list of jobs is updated and the scheduling process continues till all the jobs are scheduled to suitable resource. Experimental results indicate that the suggested process is relatively complex and issues such as consistency and cost have not been addressed. Youchan Zhu, Huili Liang scheduling algorithm is also proposed to schedule tasks according to QoS parameters required by the clients. However, user priority of tasks have not been considered.

Li Yanga *et al* have proposed a novel grid scheduling heuristic that adaptively and dynamically schedules the task without requiring prior information on the workload of incoming tasks. This uses two types of queue namely, waiting queue and execution queue. This approach is based on exploiting information on processing

capability of individual grid resources and applying on tasks assigned to the slowest processors. In this approach the cost are ignored.

Chandrashekhar *et al* have proposed an algorithm by considering multiple SLA parameter such as memory, execution time and bandwidth and resource allocation by prevention mechanism for high priority task execution can improve the resource utilization in Cloud. But it does not focus on server optimization.

Hu baofang *et al* have proposed an improved adaptive genetic algorithm (PAGA) based on priority mechanism. This approach for job scheduling looks at the combination of least execution time together with QoS requirement of customer jobs. An integrated fitness function based on priority is designed to indicate optimized object. It overcomes the disadvantage of traditional AGA algorithm and performance has been compared with other relative genetic algorithms.

Rakesh Kumar Mishra *et al* have proposed a priority based round-robin service broker algorithm which distributes the requests based on the priority of data centers. It provides better performance in comparison with the conventional random job selection algorithm. Jng Xiao and Zhiyuan Wang have proposed a priority based algorithm for scheduling virtual machines on physical hosts in cloud computing environment. The object is to maximize the benefits of the service providers when the current resources are not enough to process all the requests in time. The requests have been ranked according to the profits they can bring. They have compared their performance with the first-come-first-serve strategy.

Proposed workflow scheduling algorithm:

The proposed work looks at the multiple criteria of job priority and optimized cost. The proposed work extends our earlier work on Artificial Bee Colony (ABC) algorithm for scheduling workflows for cloud computing. ABC is an optimization algorithm that is based on the intelligent behavior of honey bee swarm. ABC is known to provide optimum solutions to complex problems and has significant advantages when compared with random search and enumeration techniques.

The proposed algorithm is explained using a sample DAG with 11 tasks shown in Figure 2. Each edge weight of DAG represents the data transmission time between the tasks.

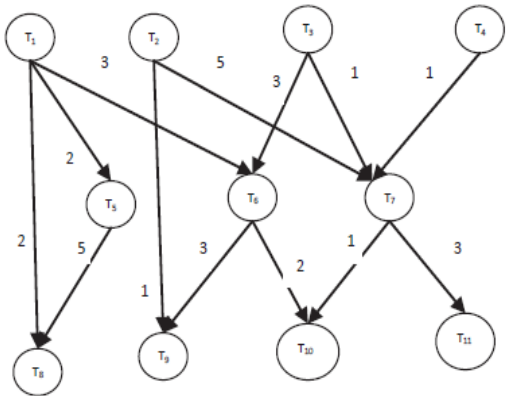

Fig. 2: Sample DAG.

3.1 Estimated Completion Time (ECT) and Cost:

It is computed as an *n* x *m* matrix where *ECTi,j* shows the estimated completion time of a task t_i on the machine m_j. The processor speeds of VM's are selected randomly in the range of 1000-5000 MIPS and cost of using these VM's is set within a range of 2-10 basic units. The fastest VM is roughly five times more expensive than the slowest one. Table 1 shows the expected completion time of various tasks on three different VMs.

Table 1: Expected Completion Time of Tasks on Three VMs.

VMs / Tasks	VM1	VM2	VM3
T1	3	5	1
T2	2	3	1
T3	3	5	1
T4	2	3	1
T5	2	3	1
T6	2	3	1
T7	2	3	1
T8	4	6	2
T9	3	5	1
T10	2	3	1
T11	5	7	3

Table 2: Shows the cost of execution of the tasks in each VM as well as the computed average cost.

VMs / Tasks	VM1	VM2	VM3	Average Cost
T1	8	6	10	8
T2	9	8	10	9
T3	8	6	10	8
T4	9	8	10	9
T5	9	8	10	9
T6	9	8	10	9
T7	9	8	10	9
T8	7	5	9	7
T9	8	6	10	8
T10	9	8	10	9
T11	6	4	8	6

3.2 Priority Assignment:

The priority of the tasks specifies the order of execution of the tasks. In the proposed work, the priorities of all tasks are computed using m-level (major level). m-level of a task of DAG is defined to be the length of the longest path from the task to the entry task without considering the execution time of that task and is given by the Equation (1).

$$\text{m-level}(m_i) = \max\left[\sum_{t_i=1}^{\text{pred}(t_i)} (d_{ij} + \text{mlevel}(t_j) + w_j) + c_j\right] \quad (1)$$

where w_j is the average execution time of the task on the different computing machines. *pred(t_j)* includes all the parent tasks of t_j. d_{ij} is the data transmission time from a task t_i to t_j. c_j is the average cost on different computing machines.

For entry task, that is, a task that has no parents, m-level is taken as zero. The computation of average cost c_{ij} is explained using the example DAG shown in Figure 2. In the example, Task t7 has three parents: t2, t3 and t4. m-level of m-level (t2) is calculated as sum of data transmission from t2 to t7, average execution time(t2) which is 7. Likewise m-level of t3 and t4 are calculated. m-level (t7) is next calculated as maximum of m-level (t2, t3 t4) + average cost (t7). The maximum value 7 is added with the average cost of t(7) (which is 9) to get m-level(t7) as 16.

m-level (t2) = [5+0+2] = 7
m-level t3 = [1+0+3] = 4
m-level t4 = [1+0+2] = 3
m-level(t7) = max(7,4,3) + cost (t7) = 7 +9 = 16

Similarly the m-level of all tasks is calculated. The tasks are then sorted in ascending order of m-level to decide the order of execution (priorities) of all the tasks and the values are shown in Table 3.

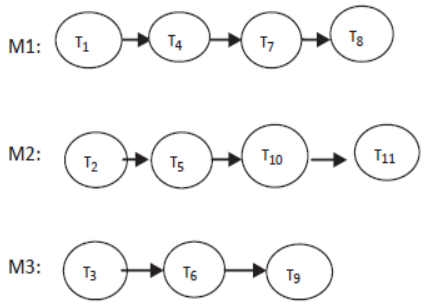

Fig. 3: Schedule according to m-level.

Table 3: Order of execution of tasks for the Sample DAG.

Tasks	Average ECT	m-level	Order of Execution
1	3	0	1
T2	2	0	2
T3	3	0	3
T4	2	0	4
T5	2	14	5
T6	2	15	6
T7	2	16	7
T8	4	21	10
T9	3	20	9
T10	2	19	8
T11	5	21	11

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 532-539

The tasks are assigned to different machines according to their order of execution. Figure 3 shows the schedules generated according to m-level of sample DAG.

Hence VM1 will execute tasks t1, t4, t7 and t8, VM2 will execute t2, t5, t10 and t11, VM3 will execute t3, t6 and t9.

Table 4: Schedule according to m-level (priority).

Tasks /Machines	VM1	VM2	VM3
1	T1	T2	T3
2	T1	T2	NULL
3	T1	T2	NULL
4	T4	T5	T6
5	T4	T5	T9
6	T7	T5	NULL
7	T7	NULL	NULL
8	T8	T10	NULL
9	T8	T10	NULL
10	T8	T10	NULL
11	T8	T11	NULL
12	NULL	T11	NULL
13	NULL	T11	NULL
14	NULL	T11	NULL
15	NULL	T11	NULL
16	NULL	T11	NULL
17	NULL	T11	NULL

The fitness value is computed as number of iterations to complete the jobs and is seen to be 17. However, the service utilization is not optimized as the machines are idle for different periods of time as shown in Table 4.

3.3 ABC Algorithm:

The ABC algorithm is now applied to the obtained task schedule. The optimization is carried out in three phases: Scheduling of tasks with priority, Improving the cyclic process and Least probability phase, which is applied when a best solution is not obtained in the earlier phases.

3.3.1 Scheduling of Tasks with Priority:

The employed bee phase of ABC algorithm is first applied. The proposed approach is a population based approach, which represents a possible solution in the optimization problem and the fitness value corresponds to fitness of the associated solution. A position update process is performed on the m-level task scheduling to improve the chances of getting a better processing order of the tasks to be scheduled. Then the fitness of each service / task is determined by a fitness function. If the fitness of the service has not improved, then new solution is searched for iteratively. The phase is completed when there is no significant improvement in the fitness function. The process is explained in greater detail.

Each task x in R can be defined using coordinates i and j as:

$$x_{ij} = \text{position}(p_i)\ i=1,2,3,\ldots..n;\ j=1,2,\ldots..,m \quad (2)$$

The population obtained using m-level is expressed as:

[1 4 7 8]
[2 5 10 11]
[3 6 9 -]

The position update process is done based on equation (2) to get better processing order of the tasks that improves the service optimization

$$v_{ij} = x_{ij} + q_{ij}(x_{ij} - x_{kj}) \quad (3)$$

where k is a solution in the neighborhood of i, q_{ij} is (-1 to 1) which is a random number generated during program execution and v_{ij} in the neighborhood of x_{ij} for the employed bees.

The updated position that is obtained is given as

Cycle1: [[1 4 7 8], [3 6 9 -], [2 5 10 11]]
Cycle2: [[2 5 10 11], [3 6 9 8], [1 4 7 -]]
Cycle3: [[3 6 9 8], [2 5 10 11], [1 4 7 -]]

The fitness value is once again calculated for the updated neighborhood position and the fitness value obtained was 13. This process is repeated and the fitness value obtained for cycles 2 and 3 were 15 and 16. Since there is no improvement in the fitness value, the scheduling is forwarded to the next phase to check for further improvement.

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 532-539

3.3.2 Improving Cyclic Process:

The tasks are selected based on the probability values of the particular service. The services with in respect to fitness values of the previous phase as: best probability values are updated and the best fitness is selected as the best solution. The probability value specifies the relevance of the particular service for the scheduling process. The probability values are calculated using equation 4.

$$probability(p_i) = \frac{fit(p_i)}{\sum_{i=0}^{d} f(p_i)} \tag{4}$$

where, probability (p_i) represents the probability value of the task p_i and the variable d represent the dimension of the process table. The probability of all the tasks are calculated based on the probability equation 5.

$f(p_i)$ is defined as the difference in computation time of the particular task to total computation time of all the tasks present in that user's request and is given as:

$$f(p_i) = \left[\sum_{j=0}^{n} time(p_j)\right] - time(p_i) \tag{5}$$

The fitness of each service is evaluated by a fitness function.

$$fit(p_i) = \begin{cases} \frac{1}{1+f(p_i)}, if\ \ f(p_i) \geq 0 \\ 1+abs(f(p_i)_i), if\ \ f(p_i) \leq 0 \end{cases} \tag{6}$$

The fitness values of each tasks p_i are calculated using the fitness function. For the given example R, total computation time of all the tasks are calculated. Fitness value calculated for the three machines are given in Table 5.

Table 5: Fitness Values of Services.

Task	Process time	$f(p_i)$	$fit(p_i)$	pi
T1	3	11-3 = 8	0.11	0.013
T4	2	9	0.1	0.011
T7	2	9	0.1	0.011
T8	4	7	0.125	0.017

Hence the population is updated with best probability (greater).

The services with the best probability values are used in the position update phase, which is similar to the previous phase.

$$pos_{new}(O_i) = pos(O_i) + \Phi_{i,j}(pos(O_i) - pos(O_k)) \tag{7}$$

The Initial population [[1 4 7 8], [3 6 9 -], [2 5 10 11]] now becomes
Cycle1: [[2 5 7 11], [3 4 9 -], [1 6 10 8]]
Cycle2: [[1 8 4 7], [3 6 9 -], [5 2 10 11]]
Cycle3: [[1 3 2 -], [4 6 5 11], [7 9 10 8]]

The fitness value is once again calculated for the updated position and the value obtained was once again 15. This process was repeated and the fitness value obtained for cycles 2 and 3 were 15 and 16. It can be noted that again, there is no improvement in the fitness value and hence the scheduling is forwarded to the next phase to check for further improvement.

3.3.3 Random phase:

In this phase, a service request with least probability is calculated and that service is replaced with a randomly generated service request. This process increases the chances of improving the solution. Then fitness of the solution is calculated and position update process is also applied to find out whether there is any improvement in the obtained solution. The process is repeated till the best solution is obtained. If a best solution is not obtainable, then the process is abandoned and the initial population is taken for scheduling.

[1, 4, 7, 8] as [4, 1, 7, 8]
[3, 6, 9, -] as [3, 6, 9, -]
[2, 5, 10, 11] as [2, 5, 10, 11]

Table 6: Fitness Computation for Modified Population.

Tasks /VMs	VM_1	VM_2	VM_3
1	T4	T3	T2
2	T4	T3	NULL
3	T1	T3	NULL
4	T1	T3	NULL
5	T1	T3	NULL
6	T7	T6	T5
7	T7	T6	NULL
8	T8	T6	T10
9	T8	T9	T11
10	T8	T9	T11
11	T8	T9	T11
12	NULL	T9	NULL
13	NULL	T9	NULL

It can be noted that 13 iterations are required to complete all the service requests. Even though the machines are idle for different periods of time, there is still as improvement over the initial process. Hence this solution can be chosen as the order for executing the user requests.

4. Performance Evaluation:

The proposed approach was simulated using *cloudsim* tool which provides a generalized and extensible simulation framework that enables seamless modelling, simulation, and experimentation of emerging cloud computing infrastructures and application services.

The performance of the proposed algorithm was compared with the Genetic Algorithm (GA) based scheduling algorithm given in the related work in section 2. GA starts with an initial set of random solutions called population. Each individual in a population is called a chromosome. The crossover, mutation and selection operations are applied in the procedure till the specified fitness criteria are reached or a specified number of iterations have been completed. The performance was simulated for three different services and service requests arranged randomly for a set of 100 users. The fitness value was computed for GA and the proposed approach and the results obtained is given in Table 7.

Table 7: Comparison of Fitness Value.

Iteration	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
ABC	83	79	71	58	40	29	23	20	19	19
GA	28	25	26	26	30	30	29	29	29	29

The results are also graphically represented in Figure 4.

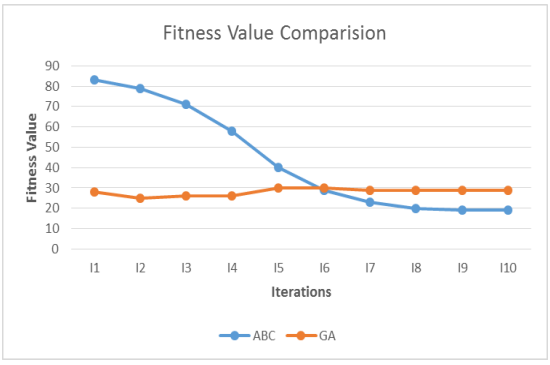

Fig. 4: Fitness Value Comparison.

It can be seen that GA performs better in the initial iterations. However, the proposed approach reaches a better fitness value of 19 as compared to 29 obtained by GA method. Hence the proposed method may be considered more efficient as compared to the GA based scheduling.

Conclusion and future work:

This paper presents an optimized workflow scheduling for cloud computing. The proposed work considers multiple criteria of priority and cost and aims to improve the service utilization while arriving at an efficient schedule of tasks. This paper extends our earlier work on Artificial Bee Colony algorithm to schedule workflow job based on assigned priority and cost while optimizing the server utilization. The proposed work was implemented and performance compared with GA based scheduling. The results showed that the proposed

approach work shows an improvement in performance when compared to GA based scheduling. As future work, it is planned to extend the work to include other criteria such as, execution time, reliability and availability.

REFERENCES

Amandeep Verma, Amandeep Verma, 2014. "Bi-Criteria Priority Based Particle Swarm Optimization Workflow Scheduling Algorithm for Cloud", Proceedings of 2014 RAECS UIET, Punjab University, Chandigarh, pp: 06–08.

Dervis Karaboga and Bahriye Basturk, 2007, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm", Journal of Global Optimization, 39: 459-471.

Elzeki, O.M, M.Z. Rashad, M.A. Elsoud, 2012. "Overview of Scheduling Tasks in Distributed Computing Systems", International Journal of Soft Computing and Engineering, 2(3).

Hung, Q.Y., T.L. Huang, 2010. "An Optimistic Job Scheduling Strategy based on QOS for Cloud Computing", IEEE International Conference in Intelligent Computing and Integrated Systems (ICISS), Guilin, pp: 673-675.

Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang, 2013. "Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm", International Journal of Computer Science.

Ke, L., J. Hai, C. Jinjun, L. Xiao, Y. Dong and Y. Yun, 2010. "A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on Cloud Computing Platform", International Journal of High Performance Computing Applications, pp: 1-16.

Kumar, P., Sheila Anand, 2013. "An approach to optimize workflow scheduling for Cloud Computing Environment", Journal of Theoretical and Applied Information Technology, 57(3): 617-623.

Li Yang, 2012. "A new Class of Priority-based Weighted Fair Scheduling Algorithm", Physics Procedia, pp: 942–948.

Poonam Devi, 2013. "Implementation of Cloud Computing by using Short Job Scheduling", International Journal of Advanced Research in Computer Science and Software Engineering.

Rajkumar Buyyaa, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, 2009. "Cloud Computing and Emerging IT platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," Journal Future Generation Computer Systems, 25(6).

Rodrigo, N. Calheiros, Rajiv Ranjan, Rajakumar Buyya, 2011. "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", in Software: Practice and Experience, 41(1): 23-50.

Saeid Abrishami, Mahmoud Naghibzadeh and Dick H.J. Pema, 2013. "Deadline Constrained Workflow Scheduling Algorithm for Infrastructure as a Service", Journal Future Generation Computer Systems, 29(1): 158-169.

Shamsollah Ghanbari, Mohamed Othman, 2012. "A Priority based Job Scheduling Algorithm in Cloud Computing", International Conference on Advances Science and Contemporary Engineering, 2012(ICASCE 2012), pp: 778 – 785.

Verma, A., S. Kaushal, 2013. "Budget constraint priority based genetic algorithm for workflow scheduling in cloud", In. Proceeding of IET International Conference on Recent Trends in Information, Telecommunication and Computing, India, pp: 8-14.

Youchan Zhu, Huili Liang, 2013. "Research for the virtual machine-oriented cloud resource scheduling algorithm", 6th International Conference on Information Management, Innovation Management and Industrial Engineering, pp: 133-136.

Yu, J. and R. Buyya, 2008. "Workflow Scheduling Algorithms for Grid Computing," Xhafa F, Abraham A (eds), Metaheuristics for scheduling in distributed computing environments, Springer, Berlin.

Zhangjun, W., L. Xiao, N. Zhiwei, Y. Dong and Y. Yun, 2013. "A market-oriented hierarchical scheduling strategy in cloud workflow systems", Journal of Supercomputing, 63(1): 256-293.