GREEDY LOAD BALANCING FOR CLOUD COMPUTING FRAMEWORK

Jayanta Datta
Department of Information Technology, RCC Institute of Information Technology, Kolkata
Kolkata, West Bengal, INDIA
jayanta.datta@rcciit.org.in

Indrajit Pan
Department of Information Technology, RCC Institute of Information Technology, Kolkata
Kolkata, West Bengal, INDIA
indrajit.pan@rcciit.org.in

Abstract

Cloud computing technology helps in resource and application sharing at large scale. Load balancing or resource sharing is one key task which controls quality of services in a cloud computing framework. This proposed work focuses on optimal task management by scheduling through greedy resource allocation strategy. Literature study reveals that greedy resource allocation concept is mostly unexplored in this domain. This work relies on that to propose a task allocation schedule. Acquired schedule is also verified with service level agreement (SLA) protocol for its validity to control quality of services. This new method has been simulated and tested under various load scenarios and also compared with two other widely used models. Experimental study shows promising outcome.

***Keywords*: Cloud computing; greedy method; job scheduling; resource management; service level agreement.**

1. Introduction

Technology enhancements are pushing individuals and organizations towards heavy dependency for computer applications. Recent computer applications are more data centric and they require heavy computational resources and costly peripheral devices. These computational resources and peripherals are not economically feasible to be acquired by every organization or individual. Advent of distributed technology in computer science has convinced for resource sharing concept. Cloud computing framework helps in that line [George and Pramila, (2021)]. Cloud resources primarily include several large data centers, many broad scale applications and hardware infrastructures. All these resources are virtually extended among end users round the clock through the cloud computing technology. Normally there are three models of cloud computing. Those are private cloud, public cloud and hybrid cloud. Hybrid cloud is a mix of private and public cloud [Hu *et al.*, (2012)]. Public cloud model is the most widely used and challenging model among these. Majority of people using computer and mobile devices are largely dependent upon cloud data centers. Cloud service level agreement (SLA) provides an initial understanding between cloud service providers and end users upon different subscription terms.

Resource allocation and scheduling of different incoming job requests to various virtual resource centers are one of the major components of cloud computing [Dutta and Joshi, (2011)]. Different infrastructures, data storage servers, and hardware devices are virtually extended as resources through several virtual machines. Service level agreements monitors this sharing strategy across several cloud service subscribers. This sharing or resource allocation is mostly performed through two mechanisms called static allocation and dynamic allocation [Li, (2009)]. Active resource allocation is needed for dynamic scheduling based on the availability of resources.

Proposed work employs greedy scheduling mechanism for optimal allocation of resources. Sometimes it has been observed in the literature that best fit allocation is simply not the best due to non standard behavior of some virtual units and sudden break down at some sites. Literature study also reveals that performance of most of the virtual machines gradually degrade at higher utilization. Quite often it has been observed that in a public cloud framework, many virtual servers are lying underutilized where as some of the virtual sites are overloaded with the task requests. In this work all such issues have been taken care of. Some of salient features of this proposed model are;

(1) Deploying greedy load balancing mechanism for resource scheduling among different incoming requests which are considered as job here.
(2) Dynamically applying an allocation threshold for each virtual machine (termed as resource here) based on incoming job requests.
(3) Verification and validation of final schedule with service level agreement to ensure that each task (or job) will be completed within a given time frame and that will not violate the commitment of service level agreement.

This greedy method has been simulated and tested under various load scenarios. Also it has been compared with two other widely used models called round robin scheduling and random scheduling. Comparative performance is quite encouraging in terms of task throughput and percentage of successful completion.

Remaining part of this article is organized as in the section 2 various state of the art research works have been discussed which are followed by the extensive discussion on proposed greedy model in section 3. Section 4 presents results and analysis of this new method and finally the conclusion and future scope appears in section 5.
.

2. Review of Literature

Some prominent research works on load balancing and job scheduling in cloud computing environment have been reviewed thoroughly to understand the present state of the research and progress in this domain. This section will illustrate those reviews briefly and finally this will try to draw the motivation for this new proposal made in this article.

A priority based task scheduling algorithm is proposed in the work of [Agarwal and Jain, (2014)]. Initially data center brokers find the availability of requested resource and its present load whenever it receives an incoming request. Priority scheduler decides the importance of the request based on its frequency and span. Thus it assigns a priority to every request and based on the priority the requested resource is assigned to the task. This work also presents a comparative analysis of this priority scheduler with traditional First Come First Serve (FCFS) and Round Robin scheduling mechanisms. Comparative analysis shows superiority of priority scheduler over FCFS and round robin scheduling mechanisms.

Another work published by [Benoit *et al.*, (2008)] has discussed a concept of bag of tasks. In this bag of tasks method, scheduler receives a bucket full of tasks of different nature. Some of the tasks are unique and sequential and some are parallel and uniform in nature. Sequential tasks have to be scheduled one after another following the defined sequence protocol however the parallel tasks can be scheduled in parallel as those are independent and necessarily need not to wait for another task to complete. If requisite resources are available then these tasks can be assigned over same time span. This bag of tasks concept attempts to reduce overall execution completion time.

[Chang *et al.*, (2012)] addresses resource distribution and job scheduling across a collection of heterogeneous systems connected over a grid. An adaptive scoring based mechanism is the underlying principle of this proposed method. The authors have commented that traditional first come first serve methodology is not adequately suitable for heterogeneous collection of resources. It takes a snap of real time resource scenario to make the decision on scheduling and allocation to enhance the throughput of the system. It has considered both computing intensive and data intensive jobs for scheduling. Once a job is submitted it computes scores of each resource cluster based on its transmission efficiency and computing efficiency. Each cluster performs a local update based on the past allocations and ongoing allocations. Additionally, a perception of global cluster update is there to study the mutual compatibility and availability among the clusters. Proposed method has been compared with an ACO based scheduling [Xu *et al.*, (2003)] and most fit fast task scheduling [Wang *et al.*, (2005)].

Another priority based job scheduling concept is proposed by [Ghanbari and Othman, (2012)]. Basically this method has been proposed as multi criteria decision making problem based on analytical hierarchical process. Hierarchical process follows three levels as (i) objective level, (ii) attributes level and (iii) alternative level. Objective level performs scheduling, attribute level holds resources and alternative level manages jobs. Once the resource requests appear, scheduler assesses the priority of jobs and then performs assignment.

[Kaleeswaran *et al.*, (2013)] proposes a dynamic scheduling mechanism for data based on genetic algorithm. This proposed process accepts job requests and passes those requests to scheduler for resource allocation. During resource allocation phase, all possible combinations of solution pairs are generated and the best fit solution is chosen for scheduling.

A multi-objective task scheduling through throughput optimization has been proposed by [Lakra and Yadav, (2015)]. Authors have conceptualized different parameters like user bandwidth, cost of processing and execution time as multiple objective criteria for scheduling to optimize the job completion per unit time. Proposed method uses non-dominated sorting [Zhao *et al.*, (2011)] to solve multi-objective problem. Virtual machines are sorted in the order of high to low quality of service metric based on these multiple objectives. Accordingly all incoming jobs are allocated with resources from different virtual machines.

A survey article by [Mishra *et al.*, (2020)] on load balancing techniques in cloud computing provides an in depth insight to the domain of this discussion. Primarily authors have segregated load balancing algorithms in to two parts as static allocation and dynamic allocation. Static allocation only focuses on two factors, initial task arrival and availability of resources whereas dynamic strategy engulfs the challenge of run time allocation through virtual machine provisioning. These dynamic strategies based on different heuristics are further classified as off-line mode scheduler which is specifically batch allocation and another is online mode scheduler. Popular off line mode scheduler discussed here are sufferage method, max-min method and min-min method where as among the online mode scheduler opportunistic load balancer, minimum execution time (MET) method, minimum compilation time (MCT) method, simulated annealing, genetic algorithm, tabu search and A* search method have been discussed.

A sustainable task scheduling strategy has been presented in the article by [Mukherjee *et al.*, (2021)]. This method reduces makespan time and task completion time of virtual machines through heuristic load balancing algorithm. It selects the best fit virtual machine from the pool to envisage proper resource utilization before scheduling. Authors have used raspberry pi as cloudlet simulator and android application as edge units.

Different particle swarm optimization (PSO) based mechanisms for scheduling has been reported in [Pradhan *et al.*, (2021)]. Different PSO mechanisms discussed in this article are standard PSO, jumping PSO, learning PSO, bi-objective PSO, multi-objective PSO, modified PSO, binary PSO, hybrid PSO and parallel PSO. All these PSO mechanisms for load balancing are compared in terms of virtual machine's makespan, throughput and execution time. Some additional parameters like energy utilization, reliability and scalability have been discussed in applicable cases.

[Sagar and Bhambhu, (2012)] discussed different load balancing algorithms and their performances for cloud computing framework. Different load balancers discussed here are random method, round robin method, weighted round robin method and dynamic round robin method. Another survey appears in [Shafiq *et al.* (2021)]. It discusses throttled algorithm, equally spread current execution method, round robin method, weighted round robin method, honey-bee method and genetic algorithm for load balancing in cloud computing to ensure quality of services (QoS) parameters of different virtual machines in cloud computing framework.

[Tong *et al.*, (2021)] discusses the constraint of service level agreement during dynamic load balancing for cloud computing. Here the upcoming task is first dynamically allocated to an available virtual machine and then that allocation is cross verified with service level agreement for deadline constraint. After proper cross verification and validation the scheduling of a task gets approval.

Another genetic algorithm inspired load balancing mechanism has been explained in the work of [Vanitha and Marikkanu, (2017)]. Authors have implemented one cost function model for each virtual machines based on CPU availability and transmission bandwidth. This cost function model helps to choose best fit virtual machine for allocation of the task. [Varalakshmi *et al.*, (2011)] has rendered a concept of workflow optimized scheduling. They have focused on the nature of tasks like whether the incoming task is independent or dependent. Workflow scheduling is performed first to overcome inter task resource acquisition competition which is finally resolved through a proper task scheduling for resources. [Xu *et al.*, (2011)] proposed a new concept of scheduling with the help of Berger model of justice in social distribution. It performs scheduling in two phases, in the first phase it ensures several quality of service parameters are taken care of during scheduling and second phase ensures fair resource allocation.

Apart from these articles there are many other articles which discuss job scheduling, load balancing and resource allocation. It is understood so far that the key requirements of a good scheduling algorithm are enhanced throughput of the system, reduced resource rejection and a good makespan metric across all virtual machines. In order to design a powerful scheduling algorithm one must not overlook the computational complexity of the proposed method. Above reviewed articles mostly relate to the dynamic scheduling mechanisms. Optimal resource distribution should be a major criterion during dynamic scheduling. Also the scheduling constraint must be verified through service level agreement. Role of service level agreement has been mostly overlooked in literature. [Tong *et al.*, (2021)] has recently kept the scope of service level agreement in their proposal for the validation. Thus there remains a profound need for exploring optimal scheduling strategy under the validation measure of service level agreement. Proposed work has attempted to address optimality issues through the greedy concept which is mostly overlooked in this domain under the adherence of service level agreements.

3. Proposed Greedy Load Balancing Mechanism

Task scheduling is one of the major load balancing aspects in cloud computing framework. It becomes more relevant for the public cloud system. It is important to improve the efficacy of overall infrastructure in terms of throughput. This proposed method utilizes an optimal solution strategy through greedy job scheduling mechanism. Final allocation is verified with service level agreement for deadline validation of each task.

3.1. *Greedy algorithm for load balancing in distributed cloud framework*

This proposed algorithm is intended to balance work load distribution among various application servers under a cloud server network. Here these loads are considered as different jobs (j_n), where j_n denotes nth job. Each job will be associated with a burst time (b_n). Burst time means the amount of CPU access (in terms of unit time cycle) required by the respective job in cloud server. Ideally we will take m number of cloud application servers, where s_m will denote mth server.

Usually when n >> m, then balancing the total load imposed by n becomes a challenge. Optimal allocation of total load imposed by n across m servers is the purpose of this algorithm.

This algorithm will follow a greedy allocation strategy of n jobs across m servers, so that none of the servers are overloaded and at the same time none of them remains under-utilized.

3.2. *Detail Strategy*

Concept of 0/1 knapsack (above threshold) will be observed. Above threshold means, during assignment if assigned load to jth server (s_j) is x, where x < th, and burst of next kth job to be assigned is b_k, then even if $(x + b_k) > th$, kth job will be assigned to jth server. Here threshold load is represented by th. Calculation of threshold load is shown in pseudo code.

Logic behind this 0/1 Knapsack (above threshold) is that none of the initial servers will be left under loaded so as to ensure no over loading at trailing servers. This allocation strategy is stable because actual capacity of each server is much higher than threshold load (th).

3.2.1. Workflow Example of Proposed Method

Step-1: Input is taken in terms of number of jobs, burst time and the number of servers as shown in table 1.

Jobs	Burst time
j_1	3
j_2	5
j_3	5
j_4	3
j_5	4

Table 1. Incoming loads (jobs) with corresponding burst time

Let us assume three servers are allotted for the balancing, e.g. s_1, s_2, s_3

Step-2: Then the threshold value is computed, which is *max (Average burst time, Maximum burst time)*

Average burst time as per table 1 = *Sum of incoming burst times/ Total number of servers* = 20/3
Maximum burst time = max (burst time) = 5
hence, threshold (th) = max (6, 5) = 6

Step-3: Sequentially jobs are taken and compared with remaining threshold capacity of each server. Initially each server is assigned with the threshold computed in step 2. However this threshold of a specific server is updated once a job is assigned to that server.

Step-4: Process of step 3 is repeated until all the jobs are assigned to some server.

Considering the assumption taken under table 1, five jobs are j_1, j_2, j_3, j_4 and j_5 along with three servers s_1, s_2 and s_3.

Iteration	Job no (Burst)	Server threshold before allocation			Server threshold after allocation		
		s_1	s_2	s_3	s_1	s_2	s_3
1	1 (3)	6	6	6	3 (j_1)	6	6
2	2 (5)	3	6	6	0 (j_1, j_2)	6	6
3	3 (5)	0	6	6	0 (j_1, j_2)	1 (j_3)	6
4	4 (3)	0	1	6	0 (j_1, j_2)	0 (j_3, j_4)	6
5	5 (4)	0	0	5	0 (j_1, j_2)	0 (j_3, j_4)	1 (j_5)

Table 2. Workflow example through assignment iteration

Now according to the algorithm, the j_1 and j_2 will be allocated to s_1, j_3 and j_4 will be allocated to s_2, and finally the job j_5 will be allocated to s_3. Step wise progress is shown in table 2.

Step-5: Proposed schedule is verified with service level agreement.

Explanation of above demonstration

1. Initially we will work for the first server which is empty, till it reaches its utmost capacity.
2. In this case burst time of j_1 and j_2 will be allocated to s_1. Now, the server s_1 will be totally occupied it.
3. Now this will work for server s_2, till it reaches its limit. So now the next jobs j_3 and j_4 will be allocated to s_2 and s_2 capacity will be reached.
4. Now, final job j_5 gets allocated s_3.

3.2.2. Pseudo Code of the Proposed Method

Input:

(i) A list of jobs (j_n) where value of n will be user defined
(ii) Each n^{th} job will be associated with a burst time b_n
(iii) Number of application servers (m)

Output:

(i) Server wise job assignment (which job is assigned to which server)
(ii) Total burst assigned to each server
(iii) Number of overloaded servers along with their server index
(iv) Number of underloaded servers along with their server index

Pseudo Code:

```
i.    $t_w = \sum_{i=1}^{n} b_i$ , where $t_w$ is total burst assigned to the system.
ii.   $t_{avg} = \frac{t_w}{m}$ , where $t_{avg}$ is average burst assigned to system
iii.  $th = \max\left[\left(\forall i : b_i\right), t_{avg}\right]$, th is the threshold capacity of every system
iv.   Initialize two m size arrays with 0, (sa) representing server allocation and (w) representing assigned
      weight to that server.

v.    set, k = 1 (which will take count of jobs)
vi.   loop on j: 1 to m
      a.  if (k < n):
            i.  $w_j \leftarrow 0$, where w is assigned weight on that server
           ii.  while ($w_j$ < th):
                  1.  $w_j = w_j + b_k$
                  2.  $sa_j \leftarrow j_k$
                  3.  k = k + 1
```

```
        b. else:
              i. break
  vii.  loop on j: 1 to m
        a. print($sa_j$)
  viii. loop on j: 1 to m
        a. if ($w_j$ > th):
              i. print: Overloaded ($sa_j$)
        b. else:
              i. print: Underloaded ($sa_j$)
  ix.   validate: SLA
```

This pseudo code formally illustrates the method as described in section 3.2.1.

4. Results and Discussion

Experimental simulation of proposed method has been carried out in python 3.6 environment and Cloudsim simulator [Tong *et al.*, (2012)]. Proposed greedy technique has been also compared with two very well known mechanisms. These mechanisms are round robin scheduling and random algorithm [Mishra *et al.*, (2020)].

Round robin process assigns task to each virtual machine following a rotational policy. It simply checks the available resource bandwidth of the assigning virtual machine and assigns. Random algorithm is not quite superlative but in some critical situation where a systematic flow of assignment fails then literature survey shows that random algorithm generates excellent results. Probability of success rate under such random simulation is quite high [Mishra *et al.*, (2020)].

All three algorithms, i.e. proposed greedy scheduling, round robin scheduling and random scheduling algorithms were implemented in python and those were integrated with Cloudsim simulator infrastructure. These experimental trials were performed in two phases.

4.1. *Generic scheduling trials*

Ten randomly generated sequence of tasks (job) and their resource requirements were taken in to the consideration for first phase experimentation. Also the number of virtual machines (VM) was generated randomly for each trial. Considering this set of jobs and VMs as benchmark data, this first phase trial was executed separately for three methods. Table 3, table 4 and table 5 show the records of this analytical study for greedy method, round-robin and random method respectively.

Trial #	Job #	VM #	Greedy Method		
			Allocated #	Failed #	Success %
1	10	4	4	0	100
2	100	25	25	0	100
3	1000	231	231	0	100
4	5000	946	946	0	100
5	10000	1933	1933	0	100
6	20000	3805	3805	0	100
7	40000	5231	5231	0	100
8	80000	10000	10000	0	100
9	100000	14517	14517	0	100
10	200000	29638	29638	0	100

Table 3. Experimental trials on proposed greedy method

Trial #	Job #	VM #	Round-robin Method		
			Allocated #	Failed #	Success %
1	10	4	4	0	100
2	100	25	25	0	100
3	1000	231	231	0	100
4	5000	946	946	0	100
5	10000	1933	1933	0	100
6	20000	3805	3805	0	100
7	40000	5231	5231	0	100
8	80000	10000	9788	212	97.88
9	100000	14517	14190	327	97.74
10	200000	29638	28917	721	97.56

Table 4. Experimental trials on round-robin method

Trial #	**Job #**	**VM #**	**Random Method**		
			Allocated #	**Failed #**	**Success %**
1	10	4	4	0	100
2	100	25	25	0	100
3	1000	231	211	20	91.34
4	5000	946	931	15	98.41
5	10000	1933	1865	68	96.48
6	20000	3805	3732	73	98.08
7	40000	5231	5112	119	97.72
8	80000	10000	9899	101	98.99
9	100000	14517	13491	1026	92.93
10	200000	29638	28999	639	97.84

Table 5. Experimental trials on random method

Observation from above experimentation reveals that the task completeness of proposed method is very high. It has acquired 100% success rate in all randomized trials. That proves high accuracy of the proposed method. On the contrary, success rate of round-robin method drops a little under heavy load as observed in trial 8, 9 and 10. Random method relative displays varying performance at different trials which is not linear in nature.

4.2. *Comparative analysis*

Separate trials performed on random benchmark are recorded in section 4.1 through table 3, 4 and 5. This section gives a single snap shot of the success percentage of each method with respect to each and every trial. This comparative study is given below through table 6.

Trial #	**Success Percentage**		
	Greedy Method	**Round-robin Method**	**Random Method**
1	100	100	100
2	100	100	100
3	100	100	91.34
4	100	100	98.41
5	100	100	96.48
6	100	100	98.08
7	100	100	97.72
8	100	97.88	98.99
9	100	97.74	92.93
10	100	97.56	97.84

Table 6. Comparative analysis on successful scheduling between three methods

5. Conclusion

Suitable task scheduling through load balancing is a major performance constraint in cloud computing. There are many parameters associated with task scheduling like user bandwidth, cost of processing and execution time which mostly control overall performance of the system. However simple and adequate allocation of virtual machines among requesting jobs is the main concern. This work has deployed greedy job scheduling method. Proposed allocation strategy dynamically reads the task requirements and assigns a threshold of allocation on each virtual site. This enables even distribution of jobs across all sites. Derived schedule has been verified and validated with service level agreement protocols to check compatibility with committed task deadlines. Experimental results have shown that percentage of successful job completion is very high and overall through put of the system has got better under proposed greedy method in compare to others. In future this base model technique on greedy job scheduling can be hybridized to meet multiple criteria optimization by considering major associated parameters like user bandwidth, cost of processing and execution time.

References

[1] Agarwal, A.; Jain, S. (2014): Efficient optimal algorithm of task scheduling in cloud computing environment. International Journal of Computer Trends and Technology, 9(7), pp. 344 – 349.
[2] Benoit, A.; Marchal, L.; Pineau, J.; Robert, Y.; Vivien, F. (2008): Offline and online master-worker scheduling of concurrent bags-of-tasks on heterogeneous platforms. Proc. of 2008 IEEE International Symposium on Parallel and Distributed Processing, 2008, pp. 1-8.
[3] Chang, R. S.; Lin, C-Y.; Lin, C-F. (2012): An adaptive scoring job scheduling algorithm for grid computing. Information Sciences, 207, pp. 79 – 89.
[4] Dutta, D.; Joshi, R. C. (2011): A genetic: algorithm approach to cost based multi-QoS job scheduling in cloud computing environment. Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ICWET ,11, pp. 422 – 427.
[5] George, S. S.; Pramila, R. S. (2021): A review of different techniques in cloud computing. Materials Today: Proceedings, Article in Press.
[6] Ghanbari, S.; Othman, M. (2012): A priority based job scheduling algorithm in cloud computing. Procedia Engineering, 50(2012), International Conference on Advances Science and Contemporary Engineering 2012, pp. 778 – 785.
[7] Hu, Z.; Wu, K.; Huang, J. (2012): An utility-based job scheduling algorithm for current computing Cloud considering reliability factor. Proc. of 2012 IEEE International Conference on Computer Science and Automation Engineering, 2012, pp. 296-299.

[8] Kaleeswaran, A.; Ramaswamy, V.; Vivekanandan, P. (2013): Dynamic scheduling of data using genetic algorithm in cloud computing. International Journal of Advances in Engineering & Technology, 5(2), pp. 327 – 334.
[9] Lakra, A. V.; Yadav, D. K. (2015): Multi-objective task scheduling algorithm for cloud computing throughput optimization. Procedia Computer Science, Elsevier, 48, pp. 107 – 113.
[10] Li, L. (2009): An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers. Proc. of 2009 Third International Conference on Multimedia and Ubiquitous Engineering, pp. 295-299.
[11] Mishra, S. K.; Sahoo, B.; Parida, P. P. (2020): Load balancing in cloud computing: a big picture. Journal of King Saud University – Computer and Information Sciences, 32(2020), pp. 149 – 158.
[12] Mukherjee, D.; Nandy, S.; Mohan, S.; Al-Otaibi, Y. D.; Alnumay, W. S. (2021): Sustainable task scheduling strategy in cloudlets. Sustainable Computing: Informatics and Systems, 30 (2021), pp. 100513.
[13] Pradhan, A.; Bisoy, S. K.; Das, A. (2021): A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. Journal of King Saud University – Computer and Information Sciences, Article in Press.
[14] Sagar, J.; Bhambhu, L. (2012): Implementation of load balance algorithm in cloud computing. International Journal of Science and Research, 3 (9), pp. 1684 – 1687.
[15] Shafiq, D. A.; Jhanjhi, N. Z.; Abdullah, A. (2021): Load balancing technique in cloud computing environment: A review. Journal of King Saud University – Computer and Information Sciences, Article in Press.
[16] Tong, Z.; Deng, X.; Chen, H.; Mei, J. (2021): DDMTS: A novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing. Journal of Parallel and Distributed Computing, 149 (2021), pp. 138 – 148.
[17] Vanitha, M.; Marikkannu, P. (2017): Effective resource utilization in cloud environment through a dynamic well organized load balancing algorithm for virtual machines. Computers and Electrical Engineering, 57 (2017), pp. 199 – 208.
[18] Varalakshmi P.; Ramaswamy A.; Balasubramanian A.; Vijaykumar P. (2011): An optimal workflow based scheduling and resource allocation in cloud. Abraham A., Lloret Mauri J., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, 190, pp. 411 – 420.
[19] Wang, S-De.; Hsu, I-T.; Huang, Z-Y. (2005): Dynamic scheduling method for computational grid environment. Proc. of International Conference on Parallel and Distributed Systems, 2005, pp. 22 – 28.
[20] Xu, B.; Zhao, C.; Hu, E.; Hu, B. (2011): Job scheduling algorithm based on Berger model in cloud environment. Advances in Engineering Software, 42(7), pp. 419 – 425.
[21] Xu, Z.; Hou, X.; Sun, J. (2003): Ant-algorithm based task scheduling in grid computing. Proc. of Canadian Conference on Electrical and Computer Engineering, 2, pp. 1107 – 1110.
[22] Zhao, J.; Zeng, W.; Liu, M.; Li, G. (2011): Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution. Proc. of 2011 International Conference on Cloud and Service Computing, 2011, pp. 185-190.

Authors Profile

Jayanta Datta, is an Assistant Professor in the Department of Information Technology at RCC Institute of Information Technology, Kolkata, India since 2018. Prior to this he was associated with Computer Application Department in the same Institute since 2006. Jayanta has received B. Sc. (MTMH) from The University of Burdwan in 2001. He completed MCA in 2004 and M. Tech. in Software Engineering in 2007, both from West Bengal University of Technology, W.B. His research interest includes SLA in Cloud Computing and Smart contract in Blockchain Technology. He has published couple of journal and conference publications in his credit. Currently he is pursuing his Ph. D. with Cloud Computing specialization.

Indrajit Pan, received his B.E. in Computer Science and Engineering with Honors from The University of Burdwan (2005) and M. Tech. in Information Technology from Bengal Engineering and Science University, Shibpur (2009). He was the recipient of University Medal in his Masters. He obtained Ph.D. in Engineering from Indian Institute of Engineering Science and Technology, Shibpur in 2015.
His current research interest includes community detection; cloud computing, influence theory and digital microfluidic biochip. He joined RCC Institute of Information Technology in 2006 and now the Associate Professor and Head of the Information Technology Department.
He has research publications in different International journals, edited books and conference proceedings. He has also coauthored some edited research volumes and international conference proceedings. He served as guest editor in International Journal of Hybrid Intelligence, Inderscience and currently an editorial board member of Elsevier's Applied Soft Computing Journal. He is now a senior member of IEEE, USA and ACM, USA.