ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/272793262

Analytical Study of Load Scheduling Algorithms in Cloud Computing

Conference Paper · December 2014

DOI: 10.1109/PDGC.2014.7030706

CITATIONS
7

READS
349

2 authors:

Divya Chaudhary
Northeastern University
16 PUBLICATIONS **140** CITATIONS

SEE PROFILE

Bijendra Kumar
Netaji Subhas University of Technology
75 PUBLICATIONS **919** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Location-Aware Data Distribution in P2P Mobile Cloud View project

Trend Analysis in Research View project

All content following this page was uploaded by Divya Chaudhary on 26 February 2015.

The user has requested enhancement of the downloaded file.

Analytical Study of Load Scheduling Algorithms in Cloud Computing

Divya Chaudhary, Bijendra Kumar

Department of Computer Engineering
Netaji Subhas Institute of Technology, Sector-3, Dwarka, New Delhi, India - 110078
divyadabas@gmail.com, bizender@hotmail.com

***Abstract*—Scheduling enables the cloud in balancing the large amount of load present in the system for faster computation. It plays a vital and significant part in the execution of the load in the various heterogeneous systems. The scheduling portrays a selection of resources for the tasks for better resource utilization. This paper differentiates the various load scheduling algorithms applied in the various heterogeneous systems in detail. It plays a key role in larger resource utilization and handling. This paper defines the basic cloud computing fundamentals and the concept of load balancing i.e., scheduling of load in cloud. The applied load scheduling algorithms are elaborated and surveyed extensively.**

***Keywords*—Cloud, Scheduling, Min-Min Algorithm, A* Algorithm, MET, MCT, Genetic Algorithm**

I. INTRODUCTION

Cloud Computing is a form of expressing the Internet having similar qualities. It binds a wide range of possibilities and outcomes for the coming generations. The cloud depicts the platform as well as the applications. It helps in hosting a large number of applications accessible with the help of internet and enabling a large amount of products and services. Provisioning of resources plays a key role in their allocation to the servers as well as systems. Distributed computing and Grid computing have paved way for a new virtualized type of computing called cloud computing, which promises to cut the marketing and working costs and enabling on-demand computing.

Figure 1 Cloud Computing.

Load scheduling refers to mechanism of scheduling and balancing the load in the system. The cloud platforms provides ability to dynamically adjust to the quantity of resources allocated to an application for handling the variable demands which are sometimes anticipated occurring due to the day and night observations of the access pattern or unanticipated events occurring due to a understated rise in the status of the applications.

As we know, huge resources need to be allocated to the servers as well as systems. This leads to congestion or imbalanced allocation of the resources in the system [3]. So, to overcome the imbalance in the network we require a load balancing and scheduling approaches. Scheduling is the mechanism used for distributing the load on the system processes or virtual machines for the optimal resource utilization. Load refers to the work done on the system and allocation of resources in the system. [29] It is the handling or the management of the load for better resource utilization. It is implemented to the resources in the system which can be disks, drivers, memory buffers, processors, network simulators, etc. [4]. Load balancing with scheduling helps in achieving maximum response time, maximum utilization of resources and the reduction of the overheads produced by the system. [22]. we detect that on increasing the number of users in the cloud computing system, the tasks to be scheduled increases proportionally. Therefore, there arose a need for better and more efficient algorithms to schedule tasks on these systems for greater resource utilization where results vary in different environments.

The remaining paper is structured as follows: Cloud framework and load scheduling are discussed in Section II. The existing load scheduling algorithmic approaches in various heterogeneous systems like distributed, grid and cluster are showcased in Section III. Finally, the conclusions and the future work are discussed in Section IV.

II. CLOUD FRAMEWORK & LOAD SCHEDULING

Cloud computing is defined as a service to distribute the computing power and storage capacity to the end users of a heterogeneous system. It is an internet based technology utilizing both dominant remote servers and internet network to handle the data, information and their applications. Users can obtain the data, information and files at any computer system location supporting an internet networking connection. It provides much more efficient computing by imparting to the users centralized memory, computation, bandwidth and storage and offering massive benefits of reduced time, unlimited computing power and flexible computing. It comprises a large number of concepts primarily resource allocation [12], scheduling [1] to be handled. It can be defined as a type of parallel and distributed system which constitutes a group of inter-connected and virtualized computers which support dynamic allocation and offers to more than one integrated resources depending on the service level agreements (SLAs) on the negotiation between the consumers as well as

the service suppliers. The NIST (National Institute of Standards and Technology) [27] state that:

"Cloud computing is a model that enables convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, storage devices, servers, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

A cloud is theoretically an elastic and distributed system in which the storage spaces and resources are distributed throughout the networks. It is distinguished as the resources are virtual and infinite and the details of the physical systems on which software runs are hidden from the user. The main motivation behind cloud computing is the handling, management and scheduling computing resources connected by a network and supporting user resources and services evenly according to the needs. It provides the ability to use the resources according to the requirements of the users and payment according to usage and elaborated in Figure 2. Thereby, helping the users explore the vast and dynamic potential of cloud computing. [28] The characteristics are rapid elasticity, massive scalability, virtualization, quality of service, measured on demand self-services, wide network access, resource assembling and resilient computing. The cloud computing resources rapidly match with the increasing capabilities if the demand increases. It enables the measuring [6] of used resources similar to the utility computing with human intervention available on the networks. This resource pool helps in enabling the use of physical and virtual resources by multiple users providing them capability to deliver and uphold a satisfactory level of service in the aspect of liabilities and challenges to regular and standard operation. [27] It is generally divided into two distinct models:

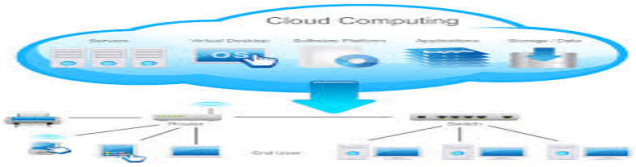

Figure 2: Framework of Cloud

a) Deployment Model: It depicts the management and positioning in the cloud infrastructure namely Public, Private, Hybrid and Community Clouds.

b) Service Model: It refers to various types of services a user can retrieve on the cloud computing platform such as Software as a Service (services as applications by use of standard interfaces to the users), Platform as a Service (services offering operation and development platforms to users), [3] Infrastructure as a Service (infrastructure resources namely fresh data storage, networking capacity and processing power as service).

Load scheduling is a computer networking technique to allocate workload through numerous computers or computer groups, networks, central processing units (CPUs), [3] disk drives, or other resources to attain ideal resource exploitation, higher throughput, lesser response time, and evade overloading of the system. [29] By the usage of many components for load balancing than a single one increases reliability through redundancy. It is a method of reallocating the complete available load on the system to the shared system's individual nodes for effective resource utilization and higher response time of the job and concurrently removing a condition of the nodes over loading and under loading. Aimed at internet services, the scheduler is generally a software package which listens to the port where outside users join to obtain services benefitting applications. The load scheduler passes requests to one of the servers in the backend which replies back to the load scheduler [12]. This helps the scheduler to reply to the users without the users ever knowing the internal configured separation and details of tasks and components [9,24]. It also averts users from interacting with servers openly providing greater security assistances by abstracting the configuration of the core network, dissimilar services on ports and averting outbreaks on the kernel's network stack. The load balancing mechanism in cloud computing environments requires three conditions [28]: Usage of local self-organization to reduce the information exchange when the local load is not very heavy or usage of cloud scheduling mechanism in the heterogeneous environment or cloud computing scheduling mechanism to affect the system average response time and increase the system throughput.

The objective of the load balancing is to enhance the performance comprehensively with effective load sharing and attaining cost effectiveness. Moreover, to have a backup plan ready in case the system fails to enable system stability and accommodate the future modification in the system by users. There are many similarities and differences among traditional scheduling algorithms [20]. Load balancing or scheduling between storage nodes is an important aspect in cloud. The load at a particular node is proportionate to the figure of file portions the node holds. The resources in load balanced cloud are nicely exploited and allocated [17] to maximize the performance of applications running in it.

III. LOAD SCHEDULING ALGORITHMS

The load scheduling algorithms are designed to schedule the load in the system for the accomplishment of the goals and producing the optimized results as a whole. The things that are considered while developing algorithm are load estimation and assessment, system performance and stability, selection and communication among the nodes, work type to be transferred, etc. The load scheduling algorithms depicted in heterogeneous environment are discussed below and tabulated in Table 1[1].

A. Min-Min Scheduling Algorithm

It is the simplest algorithm acting as a base for existing [1,30] cloud scheduling algorithms. It is fast and provides higher and better performance than others by scheduling the best case first. It starts with a set S of all unmapped tasks. Then, the resource R having lowest completion time for all tasks is located and the task T with smaller size is nominated and assigned to the relevant resource R. Lastly, task T is

detached from set S and the same procedure is performed again and again by Min-Min till every tasks is assigned. Assuming [30] we have a set of n tasks (T_n) need to be scheduled onto m available resources (R_m). We denote the expected completion time for task i starting from 1 to n on resources j from 1 to m as Ct_{ij}, where rt_j depicts the ready time of resource R_j and Et_{ij} depicts the execution time of the task T_i on resource R_j. So the completion time is

$$Ct_{ij} = Et_{ij} + rt_j$$

It initializes with U set of all unallocated tasks. The set of minimum completion times for each task is found. Then, the task having the complete minimum completion time from M is selected and allocated to the relevant machine. The newly allocated task is now detached from U, and the process goes on repeating till all tasks are mapped. It is developed keeping the concept of MCT algorithm (minimum completion time) as here all unallocated tasks are considered in a go during each allocating decision whereas in the case of MCT only single task is taken at a time. The tasks are ordered and mapped to change the availability status to the minimum value to tasks of the machine. So the first task allocated to a vacant system be t_i. The machine (m_j) finishes the task (t_i) first and thereby executing it the fastest. The availability status of each and every machine (mj) mapped after tasks is reduced to minimum value. So, on the execution time basis the assigned tasks are allocated to the first value which is relatively higher as compared with other strategies.

B. Max-Min Scheduling Algorithm

In this algorithm firstly M (set of minimum execution time) is computed analogous to the min-min algorithm. Then, the task having total highest completion time amongst M is nominated and allocated to the relevant machine. The recently allocated task is detached from U and this process continues till all tasks are allocated in the system [1,16]. Intuitively, it tries to minimalize the drawbacks encountered for computation of tasks having higher execution time. In the meta task, there are tasks having very small or very large execution time. The task mapped first to the best machine has greater larger execution time as well as it permits its simultaneous execution with the left over lesser execution time tasks [19]. It offers better mapping than the above discussed algorithm in which all the shorter tasks get executed first, followed by longer running tasks execution while several machines remain idle waiting for resources. Thus, it gives a higher makespan (total completion time) and mapped system with higher load balancing in the machines.

C. Segmented Min-Min Scheduling Algorithm

This algorithm arranges the tasks based on ETCs (Estimated Time to Complete) [14]. The tasks are sorted on the basis sorting key using an ordered list by the trade-off factor N in maximum ETC, minimum ETC and average ETC. After this segments with the equal size using a trade-off factor N are created using the task list partitioning scheme. The scheduling of the segment of bigger tasks is performed first thereafter the segment of shorter tasks i.e., in decreasing order. To allocate the various tasks to corresponding machines for each and every segment basic algorithm is used. The task sorting is done before scheduling here so that the larger tasks are scheduled earlier.

D. A Scheduling Algorithm*

It is a binary tree based search technique starting at the root node giving a void solution. The tree propagates as nodes showcase part mappings where the machines are allocated the smaller sets or subsets of tasks [1]. The subpart allocation and obtaining the solution states that child nodes one or task is allocated in comparison to parent node. Before becoming inactive the parent node for the proper allocation of task t_a, the children are generated from the parent node. Pruning is applied to keep track of tree's execution time and to limit the number of maximum active nodes at particular instant of time. A cost function f(n) is selected for partial and best solution representing the makespan's estimated lower bound on each node. Consider a second function g(n) representing the partial solution of the task machine assignment i.e., the set of tasks act as the basis of maximum availability time of machine and another function h(n) be the lower-bound estimate on the partial and complete solution's makespan. The cost function of the node is:

$$f(n)=g(n)+h(n).$$

The function h(n) is stated by two functions having two separate strategies to derive a lower bound estimate h1(n) and h2(n). The h1(n) gives the best possible metatask makespan of conflict less task allocation to machine is evaluated by performing the impractical with minimum completion time. Thus, the lower bound estimate of the left work provides cumulative value of availability time of the machine which could lead to increase in the total makespan. The function h2(n) represents an estimate of the minimal increase in the makespan of the metatask by ideal task allocation which is distributed among the machines. Thus, the root node generates nodes. The children are generated from the node having minimum value of f(n) and till the creation of all the nodes the process is repeated. So the process of pruning the tree starts as soon the node is added by deactivating the largest value f(n) containing leaf node. This process repeats until the complete allocation of the leaf node is reached. Without pruning this strategy is similar to a comprehensive search.

E. Heterogeneous Earliest Finish Time Algorithm (HEFT)

The algorithm computes the average execution time for every task and average network communication time amongst the resources of two consecutive tasks [26]. Then, orders the tasks on the basis of the rank function in the workflow i.e., greater rank amount task is awarded greater priority. The tasks are organized on the basis of their priority rank and resource is allotted to each task to complete it in the minimum time in the allocation phase.

F. Tabu Search Algorithm

It is a solution space search keeping track of the already searched regions to allow no repetitions around areas [7,15,25] and follows the genetic algorithm approach. It starts with a random allocation and mapping of the initial solution

generated with the help of uniform distribution. For the proper functioning of the present solution and allowing it to flow through the solution space and for locating the neighbouring local least solution, a short hop is executed. This short hop is performed by taking into consideration the potential task pairs, for the purpose of performing machine assignments in correspondence to others for all the potential pairs. The machines mi and mj are remapped to tasks ti and tj. Firstly, the boundary values of the different rounds are set. Then, each new solution (allocation) is computed and the new solution is considered for acceptance. If the newly obtained makespan is a progress, the new solution is saved and replaces the existing one.

G. *A Double Min-Min Algorithm for Task Scheduling*

A priori in the heterogeneous computing environment is composed of the amount of machines and metatask size [4,5] These are static experience based technique, therefore the expected precise estimation of the machines task execution time is stored in an expected time to compute matrix where the ith task's probable execution time on jth machine is ETC (t_i, m_j). The scheduling occurs using the min-min scheduling approach. In this, we allocate existing grid resources to independent job sets and tasks are allocated to the machines taking into account the workload and computing capacity of each job and resources (in MIPS) are taken into account.

H. QoS Guided Min-Min Algorithm

This algorithm considers QoS (Quality of Service) constraints like bandwidth, time, memory [2,10] in addition to primitive Min-Min heuristic. Some of the tasks are contented with lower amount of network bandwidth while others need high. The tasks having higher amount of QoS parameter request are allocated first similar to the min-min technique.

I. Minimum Execution Time algorithm

This algorithm allocates each task in random order to the machine with the finest predictable execution time for that task based on machine's accessibility to achieve optimal task machine result [1] thereby causing among the machines a major load imbalance. This approach cannot be applied to heterogeneous computing environments.

J. Resource-Aware-Scheduling algorithm (Duplex)

It is an algorithm based on Max-Min and Min-Min task scheduling algorithms and is called Resource Aware Scheduling Algorithm (RASA) or Duplex Algorithm [8]. It applies the benefits of both the above discussed algorithms and hides their drawbacks by outperforming the predefined scheduling algorithms used in large scale distributed systems [13]. The heuristic used here support both the heuristics of the algorithms and then use the improved solution. It is adjusted to exploit the conditions, with minimal overhead and higher performance either by Min-min or Max-min.

K. Round Robin & Weighted Round Robin Algorithm

It is a static and a decentralized algorithm [28] used on web servers. The processes are divided among the processors in a round robin fashion using a particular time value. The allocation order of the resources to the processors is maintained irrespective of remote ones. They possess equal distribution of workload among the processors and different processing time of job for processors. But here the nodes become overloaded and at times become under loaded [11]. In the weighted round robin algorithm, weights are assigned in a particular order to each and every task in the system to allocate resources for optimal utilization of resources.

L. Genetic Algorithm (GA)

This is a methodology applied for probing large solution locations [1,23]. It operates for a given metatask on a population of 200 chromosomes. Each chromosome acts as a vector with position i representing task t_i, and its entry at particular position in the machine where task allocation is done. It generates the initial and early population by two strategies: out of a uniform distribution 200 randomly generated chromosomes or by seeding (one chromosome from min-min algorithm (used for metatask mapping) as well as 199 random solutions). It executes eight times and the best mapping is considered the concluding solution. Each chromosome is represents a particular fitness value, which depicts the matching of tasks to machines specifying the makespan of the chromosome. So, subsequently at the initial population generation, evaluation of all the chromosomes in the population occurs on the basis of their fitness value. The lower value indicates a good mapping. Then, for purpose selection a rank-based roulette wheel scheme is used. This scheme duplicates few selected chromosomes based on probability of outcomes and deletes the rest. The elitism, which ensures the best solution exists in the population provides a greater probability of good mapping in the future generations and duplication rate. Next, the crossover operator randomly chooses a pair of chromosomes as well as a starting location in the beginning chromosome. For segments of both chromosomes from the predefined location to the last crossover exchange takes place by machine assignments between relative tasks. After crossover, the mutation is performed. Mutation arbitrarily chooses a chromosome and a task from the chromosome, and reallocates it to a naive machine. At last, the chromosomes from the improved population are computed for a second time. This algorithm stops when any one of the criterion is fulfilled: either the total 1000 runs are made, or elite chromosome does not change for 150 runs, or each and every chromosome get compacted to the same mapping.

M. Simulated Annealing (SA) Algorithm

It is a recursive procedure that chooses only one possible solution for each metatask at a particular instant of time. It uses a system allowing the acceptance of the poorer solutions for getting a effective search on the outcome of solution space [1,22]. The outcome is based on the temperature of the system that decreases for the cycle. As the cooling of system temperature occurs, poorer solutions are to be less acknowledged. The system temperature at the beginning is the makespan of the initial allocation at nodes. The first allocation plotting is calculated from a uniform random distribution and

allocation plotting (mapping) is mutated similar to GA generating a new makespan. The decision algorithm is used for accepting/rejecting the new mapping. [1] If new makespan is better, old mapping is replaced by new one. If the new makespan is of poor quality (larger), a uniform random number, z ϵ [0, 1) is chosen and it is checked with y and represented as

$$y = 1/\ 1+e^{((\text{old makespan-new makespan})/\ \text{temperature})}$$

If z<y, the older mapping is accepted else the new mapping is selected by the system. For solutions with similar makespans (very large system temperature), y→0.5 leading to acceptance of poorer solutions with approximately a 50% probability. While, for solutions with very unlike makespans (very small system temperature) then, y→1, elimination of poorer solutions takes place. After performing the mutation after every cycle the 90% system temperature reduction takes place of its current value representing the cooling rate. When there occurs no variation in terms of the makespan for 150 iterations or the temperature of system approaches zero and it stops. For a better comparison, for the every cooling rate the stopping conditions are modified and executed eight times, and then lastly the best solution out of 16 rounds is accepted as the final mapping. The additional execution time and Min-min seedings are used, but found poorer solutions than GA but lesser execution time.

N. Genetic Simulated Annealing Algorithm

This algorithm combines the GA and SA techniques [1,21]. It follows genetic algorithm strategy for the selection process and the simulated annealing cooling schedule and system temperature as well as a streamlined simulated annealing decision making process is used making the newly computed chromosome's acceptance or rejection decision. We set the initial temperature of the system to the average makespan of the initial population. Whenever we have to implement the crossover or mutation operation occurs, the newly generated chromosome is compared with the already existing original chromosome. On comparing the results, if the newly obtained makespan is smaller than the existing makespan plus the system temperature, then the new chromosome is accepted. Otherwise, in the next iteration the original chromosome is survived. Therefore, when there is a decrease in system temperature, the acceptance chances of poor solutions also gets reduced. The finishing criterion that needs to be followed is either no change in the elite chromosome in 150 iterations or 1000 total iterations.

O. Minimum Completion Time (MCT)

This algorithm allocates every task in arbitrary manner to the machines using the expected minimum completion time for task [1] but it makes the assignment of the tasks to the machines that are not having the tasks with minimum execution time. It combines the advantages of OLB and MET to overcome the poor performance circumstances.

Table 1

EXISTING SCHEDULING ALGORITHMS

Algorithm	**Method Used**	**Allocation Strategy**	**Observations**
Min-Min Scheduling Algorithm [1,30]	Many cases used, Minimum completion time	Tasks allocated resources for execution	1. Resource cost and computation performance is calculated. 2. Higher utilization rate of resources
Max-Min Scheduling Algorithm [1, 19]	Multiple cases, Speed, Max. completion time	Resources allocated to tasks	1. Computational performance and resource cost are evaluated. 2. Greater resource utilization.
Segmented Min-Min Scheduling Algorithm *[14]*	Multiple cases, Speed, Resource Utilization	Request provisioning, Segmentat-ion	1. Measures both resource cost and computation performance 2.The utilization rate of resources is high
A* Scheduling Algorithm [1]	Group approach, Resource Utilization	Unscheduled task groups, Pruning is done	1. Complete the task at earliest time. 2. Obtaining lower makespan and cost 3. Better allocation
Heterogeneous Earliest Finish Time algorithm (HEFT) [26]	Dependenc y Method, Execution time, scalability	Group of tasks ordered by rank function	1. It is used for optimizing workflow execution time. 2. It also enables resources to scale elastically during workflow execution.
Tabu Search Algorithm [1, 7]	Reliance method, Resource utilization, time	Genetic Algorithm approach, Solution Space	1. Faster execution rates 2.It is used for good distribution of workloads onto resources and in particular time frames
A Double Min Min Algorithm for Task Scheduling [4]	Various cases, Speed, Utilization	Request allocation similar to Min-Min	1. Calculates the resource cost and computational performance 2.The utilization rate of resources is high
QoS Guided Min-Min Algorithm [2, 10]	Segments are made, Resource Utilization	Quality of Service constraints considered	1. Similar to Min-Min algorithm with sub policy. 2. Resources are scaled elastically in execution with higher utilization rate
Minimum Execution Time [1]	Cases used, Resource usage	Execution time	1. Better results than OLB 2.Faster Execution and high utilization rate
Resource-Aware-Scheduling algorithm (RASA) or Duplex [1, 8]	Many cases utilized, Uses both Min-Min & Max-Min Scheduling	Execution time and completion time of tasks	1.Measures both resource cost and computation performance 2.The utilization rate of resources is high
Round Robin Algorithm, Weighted Round Robin Algorithm [11, 28]	Jobs made, Cost, Weights assigned to each task	Jobs allocated to resources based on time value	1.It is used to reduce cost and time by round robin fashion 2. Reduces the debt or cost 3. Lesser resource utilization
Genetic Scheduling Algorithm [1, 23]	Several cases, Speed, Resource, Population	Demand distribution strategy is followed	1. Speed of the GA is greater than traditional 2. High resource utilization rate

Simulated Annealing Algorithm[10]	Many cases, Cost, Temperature	Iterative, Considers solution spaces	1.Measures both resource cost and computations 2. Similar to GA but poor solution
Genetic Simulated Annealing [1,22]	Numerous cases, Execution cost and time	Combination of GA and SA	1. To minimize the cost for better resource utilization 2. Enables lesser execution cost and time.
Minimum Completion Time [1]	Grouping Approach, Execution cost and time	Request provisioning	1. Primary and basic algorithm way to Min-Min 2. Optimizes both makespan and cost.

IV. CONCLUSION AND FUTURE WORK

This paper elaborates scheduling and dynamic provisioning playing an important role in assigning the resources to tasks in the cloud network. It depicts cloud computing acting as a virtual benchmark in distributed computing. The load is scheduled in such a manner that the total load is reassigned to the distinct nodes of the shared system to make effective resource utilization and to improve the response time, completion time, debt and makespan of the job. This paper discusses the several existing load scheduling algorithms in various heterogeneous networking environments. These algorithms are analyzed on various scheduling parameters and strategies. The analysis is done to obtain greater resource utilization and reduced cost and debts in the system in a tabular form. The future work is intended to suggest a new load scheduling algorithm providing higher resource utilization and reduced cost / debt in a cloud computing environment.

REFERENCES

[1] Braun, Tracy D et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems" In Journal of Parallel and Distributed computing, Volume 61, Issue 6, Pages 810 – 837, 2001.

[2] He, XiaoShan, XianHe Sun and Gregor Von Laszewski, "QoS guided min-min heuristic for grid task scheduling", In Journal of Computer Science and Technology, Volume 18, Number 4, Pages 442 – 451, 2003

[3] Wang, Shu-Ching, et al., "Towards a load balancing in a three-level cloud computing network", In 3rd IEEE International Conference Computer Science and Information Technology (ICCSIT), Vol. 1, Pages 108 - 113, 2010.

[4] DDH Miriam, KS Easwarakumar, "A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems", In International Journal of Computer Science Issues, Vol. 7, Issue 4, Pages 8-18, 2010.

[5] Kong, Xiangzhen et al, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction", In Journal of Network and Computer Applications, Vol. 34, Issue 4, Pages 1068 – 1077, 2011.

[6] Hirales-Carbajal, Adán, et al, "Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid", In Journal of Grid Computing, Volume 10, Number 2, Pages 325 – 346, 2012.

[7] W. Chen, J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements", In IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, Vol. 39, No. 1, January 2009, Pages 29-43.

[8] Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Task Scheduling Algorithm in Grid Environment", In Special Issue of Computer & IT, World Applied Sciences Journal, Vol. 7, Pages152-160, 2009.

[9] J. Yu and R. Buyya, "Workflow Scheduling Algorithms for Grid Computing", In Technical Report, GRIDS-TR-2007-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, May 2007.

[10] Meng Xu, Lizhen Cui et al, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing", In IEEE International Symposium on Parallel and Distributed Processing with Applications, 10-12 August 2009, Pages 629-634.

[11] Bhathiya Wickremasinghe, Rodrigo N. Calheiros and Raj Kumar Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", In 24th IEEE International Conference on Advanced Information Networking and Applications, 20-23 April 2010, Pages 446-452.

[12] A.Khiyaita, El Bakkali, M.Zbakh, Dafir El Kettani, "Load Balancing Cloud Computing: State of Art", In IEEE Transactions on Software Engineering, 978-1-4673-1053-6, 2012, Pages 106-109.

[13] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling", In 3rd IEEE/IFIP International Conference in Central Asia, 26-28 September 2007, Pages 1-7.

[14] M. Wu, Wei Shu and Hong Zhang, "Segmented min-min: A static mapping algorithm for meta tasks on heterogeneous computing systems", In HCW 9th Heterogeneous Computing Workshop, page 375, Washington, DC, USA, 2000, Pages 375-385. IEEE Comp. Society.

[15] Suraj Pandey, Rajkumar Buyya et al, "A Particle Swarm Optimization based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", In 24th IEEE International Conference on Advanced Information Networking and Applications, 20-23 April 2010, Pages 400-407.

[16] M. Hosaagrahara and H. Sethu, "Max-Min Fair Scheduling in Input-Queued Switches", In IEEE Transactions On Parallel And Distributed Systems, Vol. 19, No. 4, Pages 462-475, April 2008

[17] Sen Su, Jian Li, Qingjia Huang et al, "Cost-efficient task scheduling for executing large programs in the cloud", In Journal of Parallel Computing, Elsevier, Vol. 39 Pages 177–188, 2013.

[18] Doulamis et al., "Fair Scheduling Algorithms in Grids", In IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 11, November 2007, Pages 1630-1648

[19] Luiz F. Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca, "Scheduling in Hybrid Clouds", In IEEE Communications Magazine, September 2012, Pages 42-47.

[20] Chun-Wei Tsai and Joel J. P. C. Rodrigues, "Metaheuristic Scheduling for Cloud: A Survey", In IEEE Systems Journal, Vol. 8, No. 1, March 2014, Pages 279-291

[21] H. Chen, N. S. Flann, and D. W. Watson, "Parallel genetic simulated annealing: A massively parallel SIMD approach", In IEEE Transactions in Parallel and Distributed Computing, Vol. 9, No. 2, Feb. 1998, Pages 126-136.

[22] P. Shroff et al, "Genetic simulated annealing for scheduling data dependent tasks in heterogeneous environments", In 5th IEEE Heterogeneous Computing Workshop, Pages 98-104, 1996.

[23] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey", In IEEE Journal of Computing, Vol. 27, No. 6, June 1994, Pages 17-26.

[24] H. Topcuoglu et al, "Task scheduling algorithms for heterogeneous processors", In the 8th IEEE Heterogeneous Computing Workshop, 1999, Pages 3-14.

[25] L. Wang, H. J. Siegel, V. P. Roychowdhury, A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach", In Journal of Parallel and Distributed Computing, Vol. 47, No. 1, Nov. 1997, Pages 1-15.

[26] Haluk Topcuoglu et al., "Performance Effective and Low Complexity Task Scheduling for Heterogeneous Computing", In IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3, Pages 260-274.

[27] http://en.wikipedia.org/wiki/Cloud_computing

[28] Divya Chaudhary and RS Chhillar, "A New Load Balancing Technique for Virtual Machine Cloud Computing Environment" in International Journal of Computer Applications 69(23) Pages 37-40, May 2013.

[29] http://en.wikipedia.org/wiki/Load_balancing_(computing)

[30] Huankai Chen,Wang, F.,Helian, N., Akanmu, G, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing", In 2013 IEEE National Conference on Parallel Computing Technologies, 21-23 Feb. 2013, Pages 1 – 8, ISBN 978-1-4799-1589-7.

View publication stats