IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

Request Scheduling Techniques in Large Scale Computing Environments

R. Arokia Paul Rajan

Professor, Department Master of Computer Applications, Pope John Paul II College of Education, Pondicherry, India

Abstract: Resource management is a crucial area of research since the emergence of large-scale computing environments. The user's satisfaction these computing systems depends on the efficiency of resource provisioning. Request Schedulingis a key component of resource provisioning and management. Continuously improving the efficiency of request schedulingprinciples significantly enhances the performance of these systems. A detailed survey of literature presented in this paper helps to understand the course of research in this area.

Keywords: load balancing, client/server, peer-to-peer, grid, cloud.

I. INTRODUCTION

Various scheduling techniques have been designed and adopted in different domains of interest in distributed systems. The contributions were designed based on popular existing models as well as by making suitable changes to extend the existing ones.

The following literature survey presents the contributions that are influential for the research works based on design, principles, parameters, metrics, and interfaces.

II. SCHEDULING PRINCIPLES IN CLIENT/ SERVER SYSTEMS

In random allocation principle [1], the requests are assigned to any server picked randomly among the group of servers. In such a case, one of the servers may be assigned with more requests while the other servers remain idle. However, on an average, each server gets its share of the load by random selection.

In round-robin principle [2,3], the scheduler assigns the requests to a list of the servers on a circular basis. The first request is allocated to a server picked randomly from the group so that if more than one scheduler arrives simultaneously, not all of these requests go to the same server. For the subsequent requests, the scheduler follows the circular order to redirect the request. Once a server is assigned a request, the server is moved to the end of the list. This keeps the servers equally assigned.

Weighted round-robin principle [4] eliminates the deficiency of the plain round- robin principle. In a weighted round-robin, one can assign a weight to each server in the group so that if one server is capable of handling twice as much load as the other, the powerful server gets a weight of 2.

In such cases, the scheduler will assign two requests to the powerful server for each request assigned to the lower one.

III. SCHEDULING PRINCIPLES IN PEER-TO-PEER SYSTEMS

ID management technique [5] is a greedy distribution principle that directs joining peers to a highly frequented region of the ID space. It is based on the principle that peers responsible for these regions are most likely to be overloaded. To identify these highly-frequented regions, the statistics on the utilization of the peers' overlay links during the regular operation of the Peer-to-Peer network has been collected.

In Intra-cluster principle [5], the cluster leader receives the information periodically regarding the loads and available disk space of the peers. Based on the load, the cluster leader creates a sorted list of the peers such that the first element of the list is the heavily loaded peer. Periodically, the cluster leader checks for any load imbalance due to any peer joining/leaving the system. Inter-cluster principle manages this hotspot imbalance of load by replicating the hot data from the first peer in the list to the last peer and the second peer to the second last peer and so on [6-8]. If the load difference between the peers exceeds a pre-specified threshold, then the data will be replicated.

Dynamic structured P2P Systems with Directories [9] stores the load information of the peer nodes in a number of directories which periodically schedule reassignments of virtual servers to achieve better balance. Each directory has an ID known to all nodes and is stored in a node responsible for that ID. Each directory collects load and capacity information from nodes of its peer. When node's utilization jumps above a parameterized threshold, it immediately reports to the directory which it has contacted recently. It then schedules immediately transferring from the current node to the lightly loaded nodes.

IV. SCHEDULING PRINCIPLES IN GRIDSYSTEMS

Fuzzy based scheduling principle [10] is based on the fuzzy logic, which is a multivalued logic control and the

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

rules are in the form of fuzzy conditional statements. The mapping of input to output is provided by Fuzzy Inference System (FIS). The advantage using fuzzy-based approach is that it detects the imbalance between nodes and avoids the unnecessary load. Fuzzy-based load balancing analyses information passed from the load monitor and then make a decision. It uses a domain expert's knowledge for the creation of rule base.

Genetic Algorithm [11] based approach starts with a randomly generated initial population called a chromosome. Solutions from one population are taken and used to form a new population. After several generations, final solution or optimal solution is generated. Three basic operations used in GA are selection, crossover, and mutation. In Agent-based approach [12], a centralized control mechanism is used by the agent. An agent searches the suitable node for the execution of the job. A system with multiple agents dispatches the agents to multiple nodes to execute the service. All agents have prior knowledge about other agents. Whenever an agent receives a job, it connects with the other agents to determine the job execution time.

The Hybrid approach [13] maintains the status of each node as idle or busy. To effectively utilize the participant node and the overall system, a hybrid approach is beneficial. In the static approach, there is no need for continuous collection of system information. In other hand, dynamic approach assigns a task to the appropriate node based on continuous monitoring of system information. Policy-based approach [14] handles different computation time of a job on various nodes. The initial execution time of a job is set to the mean value. This value is taken by using the different time values on a set of available nodes. When the algorithm changes its scheduling decision mean time, it is updated using iterative scheduling approach. History based approach [15] estimates the start time for the job and then allocates it to the appropriate server. The estimation of the start time is done using execution history. The scheduler contains various modules such as resource select, reservation map, and information service.

V. SCHEDULING PRINCIPLES IN CLOUDSYSTEMS

Randomized algorithm [16] is static in nature. In this algorithm, a request can be handled by a particular server n with a probability p. The process allocation order is maintained for each processor independent of allocation from the remote processor. This algorithm works well if the processes are equally loaded. However, the problem arises when loads are of different computational complexities. The randomized algorithm does not maintain deterministic approach. It works well when the round-robin principle generates overhead for process queue.

In Round-robin principle [17], the processes are divided between all nodes. Each request is assigned to the node in a round-robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the workload distributions between processors are equal, the job processing times for different processes are not same. So at any point of time, some nodes may be heavily loaded and while others remain idle. This principle is used in web server. Round-robin principle is presented as follows:

Procedure Round-Robin(N, R)
/* N – Number of VMs; R – Requests; */
Output: Request assignment
1. repeat
2. maintain an index of VMs and the state of the VMs (busy/available). At the start, all VMs have zero allocation.
3. receive the users' requests.
4. store the arrival time and burst time of the user requests.
5. allocate to VMs on the basis of their states known from the VM queue.
6. allocate the time quantum for user request execution.
7. decide the scheduling order.
8. de-allocate the VMs after the execution of requests.
9. until all the requests are served;

The service scheduling schemes are modelled using a queuing game model [18] which is used in software as a service (SaaS) Cloud model. The objective is to maximize the Cloud Computing Platform's (CCP) payoff by controlling the service requests, whether to join or balk, and controlling the value of the CCP.

In Honey bee behaviour inspired load balancing technique [19], the current workload of the Virtual Machine (VM) is calculated to decide the VM states namely over-loaded, under-loaded or balanced. According to the current load of VM, they are grouped. The priority of the request is taken into consideration only after removing the requests waiting in the overloaded VM. The requests are then scheduled to the lightly loaded VM.

The Ant colony optimization approach [20] is aimed to provide efficient distribution of workload among the nodes. When a request is initialized, the ant starts moving towards the source of food from the head node. The unprocessed request keeps a record of every node it visited and records their data for future decision making. A scheduling principle is then developed based on the sociological justice distribution theory - Berg model [21]. This algorithm adopts the commercialization and virtualization features of Cloud computing differing from the traditional job scheduling algorithm's character by focusing on efficiency and establishes dual fairness constraints under the Cloud environment.

A dynamic priority parallel job scheduler [22] allows users to control their allocated capacity by adjusting their spending over time. This mechanism allows the scheduler to make more efficient decisions about the jobs and users priorities. It gives users the tool to optimize and customize

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

their allocations to fit the importance and requirements of their jobs. The designed principle of priority based job scheduling algorithm [23] is used in Cloud environment. It uses multiple criteria decision making model - Analytical Hierarchy Process.

User-priority guided Min-Min scheduling algorithm [24] accommodates the demands of different users by delivering the services at different levels of quality. Therefore, the user gets guarantee for the service that he sought for. Dynamic balancing algorithm [25] maintains the requests in a queue to a computing node based on the capacity of the machine. Dynamic Round-Robin (DRR) algorithm [26] schedules the energy-aware virtual machines based on the power save strategy which yields better results compared to greedy and round-robin principles.

Throttled load balancing algorithm [27] is implemented with a Throttled Load Balancer (TLB) to monitor the loads on each VM. TLB ensures only a pre-defined number of Internet Cloudlets are allocated to a single VM at any given time. If more request groups are present than the number of available VM's at a data center, some of the requests will have to be queued until the next VM becomes available. Throttled principle is presented as follows:

Procedure Throttled(N, R)
/* N – Number of VMs; R – Requests; */
Output: Request assignment
1. repeat
2. maintain an index table of VMs and the state of the VM (Busy / Available). At the start, all VM's are available;
3. receive a new request;
4. query for the next allocation with resource pool;
5. parse the allocation table from the top until the first available VM is found or the table is parsed completely;
6. if VM found then
7. return the VM ID to the controller;
8. send the request to the VM identified by that ID;
9. notify the new allocation;
10. update the allocation table accordingly;
11. else
12. append the request in the Queue;
13. de-allocate the VM when the VM finishes processing the request;
14. until all the requests are served;

Equally spread current execution principle [28] handles the requests with priorities. It distributes the load randomly by checking the size and transfers the load to those virtual machines which are lightly loaded to maximize throughput. It is spread spectrum technique in which the load balancer spreads the load of the job in hand into multiple virtual machines.

Least connection principle [29] is a dynamic scheduling principle which counts the number of connections for each server dynamically to estimate the load. The load balancer records the connection number for each server. The connection number increases when a new connection is dispatched to it and decreases the number when connection finishes or timeout happens.
Active Monitoring principle [30] manages the load among available VM's in a way to even out the number of active tasks on each VM at any given time. Figure 2.3 shows the processes involved in the active monitoring principle.

Procedure Active_Monitor(N, R)
/* N – Number of VMs; R – Requests; */
Output: Request assignment
1. repeat
2. find the available VM.
3. check for all current allocation count is less than the max length of VM list and allocate the VM.
4. if available VM is not allocated, create a new one.
5. count the active load on each VM.
6. return the ID of those VM which is having least load.
7. allocate the request to one of the VM.
8. if a VM is overloaded then distribute some of its work to the VM having least work so that every VM is equally loaded.
9. receive the response to the request sent and then allocate the waiting requests from the job pool / queue to the available VM & so on.
10. until all the requests are served;

In the task scheduling principle [31], two-level task scheduling mechanism is carried out to meet dynamic requirements of users as well as to obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then the virtual machines to host resources, thereby improving the task response time, resource utilization and overall performance of the Cloud computing environment.

Biased random sampling [32] is a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. A virtual graph is constructed that represents the load on the server. Each server is symbolized as a node in the graph with each in degree directed to the free resources of the server. This principle is fully decentralized thus making it apt for large network systems like Cloud. This work contributed a novel approach using parallelism, shared state, and lock-free optimistic concurrency control.

Min-Min algorithm [33] begins with a set of all unassigned tasks. First, minimum completion time for all tasks is found. Among this set, a minimum value is selected which is the minimum time among all the tasks on any resources. According to that minimum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine. The assigned task is then removed from the list of the tasks that are to be assigned to the machines. The same procedure is

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

followed until all the tasks are assigned to the resources. But this approach has a major drawback. It can lead to starvation.

Max-Min algorithm [34] is also same as the min-min algorithm except the following: after finding out minimum execution times, the maximum value is selected which is the maximum time among all the tasks on any resources. Then, according to that maximum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine. The assigned task is then removed from the list of the tasks that are to be assigned to the machines.

The objective of the branch and bound token routing algorithm [35] is to minimize the system cost by moving the tokens around the system. In a scalable Cloud system, agents cannot have enough information about distributing the workload due to communication bottleneck. So the workload distribution among the agents is not fixed. The drawback of the token routing algorithm can be removed with the help of heuristic approach of token based load balancing. This algorithm provides fast and efficient routing decision.

VI. CONCLUSION

This paper presented the essence of various request scheduling principles from literature and the gaps found in the literature for further researches are summarized as follows:

i. There is a wide scope for designing scheduling techniques suited for large-scale distributed environments.
ii. A technique which is adopted in a particular architecture cannot be used as such for another since the scenario and the parameters are unique.
iii. Based on the objective function of the system, amendments to the existing principles can result in the emergence of new scheduling principles.
iv. Cloud cannot be generalized with a generic scheduling principle. There is a need to design customized principles for its different service models as well as for the varied services.
v. The literature on heterogeneous resources management in large-scale distributed management is scarcely available.
vi. There is a scope for developing weighed nodes scheduling principles based on statistical methods.
vii. There is a need for methods that assigns a weight for each server in a cluster as well as enumerates the number of requests it can process.
viii. There is a need for customer preference analysis methodologies incorporated in customer centered business offerings like a Cloud computing model. The literature survey reveals only a few contributions in that direction.
ix. Setting up and experimenting the researches in real time large-scale computing environments will be a costly affair. Hence, there is a need for simulators.

REFERENCES

[1] A. N. Tantawi and D. Tawsley, "Optimal Static Load Balancing in Distributed Computer Systems," IEEE Transactions of Computers, vol. 41, issue 3, pp. 381-384, 1992.
[2] C. Wang, C. Huang and H. Liang, "ASDF: An Autonomous and Scalable Distributed File System," Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 485-493, 2011.
[3] D. Grosu, A.T. Chronopoulos and M. Leung, "Cooperative load balancing in distributed systems," Practice and Experience in Concurrency and Computation, vol. 20, no. 16, pp. 1953-1976, 2008.
[4] J. Zinke and B. Schnor, "The impact of weights on the performance of Server Load Balancing systems," Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, pp. 30-37, 2013.
[5] I. Hwang, K. Roy, H. Balakrishnan and C. Tomlin, "A distributed multiple-target Identity Management Algorithm in Sensor Networks," Proceedings of the IEEE Conference on Decision and Control, vol. 1, pp. 728-734, 2004.
[6] Z. Xin-lian and X. Jian-bo, "IISA: An Inter-Cluster and Intra-Cluster Scheduling Algorithm Cluster-Based for Wireless Sensor Network," Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1-6, 2008.
[7] R. Ranjan, L. Zhao, X. Wu, A. Liu, A. Quiroz and M. Parashar, "Peer-to-peer cloud provisioning: Service discovery and load-balancing," Principles, Systems and Applications in Cloud Computing, pp. 195-217, 2010.
[8] B. Zhang and S. Wang, "An optimization model of load balancing in Peer to Peer (P2P) Network," Proceedings of the International Conference on Computer Science and Service System, pp. 2064-2067, 2011.
[9] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," Proceedings of the 9th International Conference on Peer-to-Peer, pp. 99–100, 2009.
[10] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, Elsevier, vol. 63, no. 6, pp. 217-240, 2006.
[11] R. P. Prado, S. GarcíaGalán, A. J. Yuste, J. E. Muñoz Expósito, A. J. Sánchez Santiago and S. Bruque, "Evolutionary Fuzzy Scheduler for Grid Computing," Proceedings of the 10th International Work-Conference on Artificial Neural Networks, Part I, pp. 286-293, 2009.
[12] B. S. Mohapatra, S. Kumar and Jena, "A Genetic Algorithm Based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems," Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 2008.
[13] Z. Shi, H. Huang, J. Luo, F. Lin and H. Zhang, "Agent-based grid computing," Applied Mathematical Modelling, Elsevier, vol. 30, issue 7, pp. 629–640, 2006.
[14] S. ThamaraiSelvi, R. K. SathiaBhama, S. Architha, T. Kaarunya and K. Vinothini, "Scheduling in Virtualized Grid Environment using Hybrid Approach," International Journal of Grid Computing and Applications, vol.1, no.1, pp. 1-12, 2010.
[15] E. Magaña and J. Serrat, "QoS Aware Policy-Based Management Architecture for Service Grids," Proceedings of the 14th IEEE International Workshops on Enabling Technologies, pp. 290-291, 2005.
[16] M. Swarna, P. S. SitharamaRaju and N. Vadaparthi, "Memoir: A History based Prediction for Job Scheduling in Grid Computing," International Journal of Computer Applications, vol. 46(10), pp.1-13, 2012.
[17] A. Thomas Henzinger, V. Anmol Singh, Vasu Singh, T. Wies and D. Zufferey, "Static scheduling in Clouds," Proceedings of the 3rd USENIX conference on Hot topics in cloud computing, pp. 1-6, 2011.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

[18] B. Wickremasinghe, R. N. Calheiros and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452, 2010.

[19] F. Lin, X. Zhou, D.o Huang, W. Song and D. Han, "Service Scheduling in Cloud Computing based on Queuing Game Model," KSII Transactions on Internet and Information Systems, Vol.8 (5), pp. 1554-1566, 2014.

[20] L. D. DhineshBabu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing, vol. 13 (5), pp. 2292–2303, 2013.

[21] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," Proceedings of the 14th International Conference on in Computer Modelling and Simulation, pp. 3-8, 2012.

[22] H. Yu, Y. Lan, X. Zhang, Z. Liu, C. Yin and L. Li, "Job Scheduling Algorithm In Cloud Environment," Proceedings of the 5th International Conference on Computational and Information Sciences, pp. 1652-1655, 2011.

[23] S. Ghanbaria and B. Mohamed Othmana, "A Priority based Job Scheduling Algorithm in Cloud Computing," Proceedings of the International Conference on Advances Science and Contemporary Engineering, Elsevier, pp. 778 – 785, 2012.

[24] Z. Lee, Y. Wang and W. Zhou, "A dynamic priority scheduling algorithm on service request scheduling in cloud computing," Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology, vol. 9, pp. 4665-4669, 2011.

[25] G. Liu, J. Li and J. Xu, "An Improved Min-Min Algorithm in Cloud Computing," Proceedings of the 2012 International Conference of Modern Computer Science and Applications, pp. 47-52, 2013.

[26] W. Tian, Y. Zhao, Y. Zhong, M. Xu and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters," Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems, pp. 311-315, 2011.

[27] C. Lin, P. Liu and J. Wu, "Energy-Aware Virtual Machine Dynamic Provision and Scheduling for Cloud Computing," Proceedings of the IEEE International Conference on Cloud Computing, pp. 736-737, 2011.

[28] V. Bagwaiya and S. K. Raghuwanshi, "Hybrid approach using throttled and ESCE load balancing algorithms in cloud computing," Proceedings of the International Conference on Green Computing Communication and Electrical Engineering, pp. 1-6, 2014.

[29] N. Rodrigo Calheiros, Rajiv Ranjan, A. Beloglazov, A. F. Cesar De Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software: Practice and Experience, Wiley Press, vol. 41, no.1, pp. 23-50, 2011.

[30] L. Yang and S. Yu, "A variable weighted least-connection algorithm for multimedia transmission," Journal of Shanghai University, vol. 7, issue 3, pp. 256-260, 2003.

[31] X. Wua, M. Denga, R. Zhanga, B. Zengb and S. Zhoua, "A Task Scheduling Algorithm based on QoS-Driven in Cloud Computing," Proceedings of the 1st International Conference on Information Technology and Quantitative Management, vol. 17, pp. 1162-1169, 2013.

[32] Sheeja and S. Manakattu, "An improved biased random sampling algorithm for load balancing in cloud based systems," Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pp. 459-462, 2012.

[33] H. Chen, F. Wang, N. Helian and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," Proceedings of the National Conference on Parallel Computing Technologies, pp. 1-8, 2013.

[34] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing," International Journal of Computer Applications, vol. 50(12), pp. 22-27, 2012.

[35] C. Jung, H. Kim and T. Lee, "A Branch and Bound Algorithm for Cyclic Scheduling of Timed Petri Nets," IEEE Transactions on Automation Science and Engineering, vol.12, no.1, pp.309-323, 2015.