Novel Adaptive Scheduling Algorithm for Computational Grid

Sunita Bansal, Gowtham K
Computer Science & Information Systems Group
Birla Institute of Technology & Science
Pilani, Rajasthan, 333031, INDIA
sunita_bansal@bits-pilani.ac.in

Chittaranjan Hota
Computer Science & Information Systems Group
Birla Institute of Technology & Science, Pilani
Hyderabad Campus, Hyderabd, AP, INDIA
hota@bits-hyderabad.ac.in

Abstract— **Scheduling is an important issue in computational grid. In computational grid, computing resources are connected through networks. So, if we want to take advantage of computational grid, an efficient scheduling algorithm is necessary to assign jobs to the appropriate nodes. Our adaptive load sharing algorithms uses a timer to find a receiver/ sender. If receiver does not find a sender it broadcasts a message to decrease threshold. Similarly if sender does not find receiver within poll limit it broadcasts a message to increase the threshold. We implemented distributed algorithms using a decentralized approach that improves average response time of jobs. The job arrival process and the CPU service times are modeled using M/M/1 queuing model. We compared the performance of our algorithms with similar algorithms in the literature. We present some results that verify the effectiveness of our scheme.**

Keywords- **Job Scheduling, Computational grid.**

I. INTRODUCTION

The Grid [1] is emerging as a wide-scale, distributed computing infrastructure that promises to support resource sharing and coordinated problem solving in dynamic, multi institutional Virtual Organizations [3]. The idea is similar to the former meta-computing [4] where the focus was limited to computing resources while Grid computing takes a broader approach. On one hand, Grid computing provides the user with access to locally unavailable resource types. On the other hand, there is the expectation that a large number of resources are available. A computational Grid is the cooperation of distributed computer systems where user jobs can be executed either on local or on remote computer systems. With the Grid becoming a viable high performance alternative to the traditional super-computing environment, various aspects of effective Grid resource utilization are gaining significance. With its multitude of heterogeneous resources, a proper scheduling and efficient load balancing across the Grid is required for improving the performance of the system.

In general, any load balancing algorithm consists of two basic policies - transfer policy and location policy. The transfer policy decides if there is a need to initiate load balancing across the system. Using workload information, it determines when a node becomes eligible to act as a sender (transfer job to another node) or as a receiver (retrieve a job from another node). The location policy determines a suitably under-loaded processor. In other words, it locates complementary nodes to/from which a node can send/receive workload to improve overall system performance. Location based policies can be broadly classified as sender-initiated, receiver-initiated or symmetrically-initiated [5] – [8]. Further, while balancing the load, certain type of information such as number of jobs waiting in queue, job arrival rate, CPU processing rate, etc. at each processor as well as at neighboring processors may be exchanged among the processors for improving the overall performance. Based on the information that can be used, load balancing algorithms are classified as static, dynamic or adaptive [7], [9]–[11]. In a static algorithm, the scheduling is carried out according to a predetermined policy. The state of the system at the time of the scheduling is not taken into consideration. On the other hand, a dynamic algorithm adapts its decision to the state of the system. Adaptive algorithms are a special type of dynamic algorithms where the parameters of the algorithm and/or the scheduling policy itself is changed based on the global state of the system. According to another classification, based on the degree of centralization, load scheduling algorithms could be classified as centralized or decentralized [7], [11]. In a centralized system, the load scheduling is done only by a single processor. Such algorithms are bound to be less reliable than decentralized algorithms, where load scheduling is done by many, if not all, processors in the system. However, decentralized algorithms have the problem of communication overheads incurred by frequent information exchange between processors. Load balancing involves assigning to each processor work proportional to its performance, thereby minimizing the response time of a job. But there are a wide variety of issues that need to be considered for heterogeneous Grid environment. For example, capacities (in terms of processor speed) of the machines differ because of processor heterogeneity. Also their usable capacities vary from moment to moment according to load imposed upon them. Further, in Grid computing, as resources are distributed in multiple domains in the Internet, not only the computational nodes but also the underlying network connecting them is heterogeneous. The heterogeneity results in different capabilities for job processing and data access. For instance, typical bandwidths in Local Area Network (LAN) vary from 10Mbps to 1Gbps, whereas it is few kbps to Mbps for Wide Area Network (WAN). Also network bandwidth across resources varies from link to link for large-scale Grid environments. Further, network topology among resources is also not fixed due to dynamic nature of the Grid. Many load balancing algorithms [2], [12],

978-1-4244-4793-0/09/$25.00 ©2009 IEEE

Authorized licensed use limited to: Synopsys. Downloaded on June 28,2023 at 09:24:03 UTC from IEEE Xplore. Restrictions apply.

[13] have been proposed for traditional cluster computing and distributed systems. However, little work has been done to cater to the following unique characteristics for the Grid computing environment: heterogeneous resources, considerable communication delay and dynamic network topology. The present work is targeted to the Grid model where heterogeneous resources are connected through arbitrary topology and network bandwidth also varies from link to link. This aspect precludes the application of traditional load balancing algorithms to Grid Computing. The aim of this paper is to present load balancing algorithms adapted to heterogeneous Grid computing environment.

The rest of paper is organized as follows. Section 2 presents the related work. Section 3, we present the system model. Section 4 describes the design of our algorithms. In Section 5, we describe the implementation of our algorithms and simulation results. Finally, Section 6 concludes our paper.

II. Related Work

Lu and Zomaya [14] introduced a hybrid policy to reduce the average job response time of a system. In their work, clusters are organized into regional grids around a set of well-known broker sites in terms of the transfer delay. When a job arrives at a site it calculates the expected completion time of the job within the region, based on which it transfers the job either to any other site or to a global queue. Each broker gets periodically update information from the remote sites and then it sends information about the lightly loaded nodes to its local site in order to minimize average response time of the system.

Shah et al. [15] proposed dynamic, adaptive, and decentralized load balancing algorithms for computational grids. Several constraints such as communication delay due to underlying network, processing delays at the processors, job migration cost and arbitrary topology for grid systems are considered. They proposed two algorithms, MELISA and LBA wherein MELISA works with large-scale grid environments and LBA works with small-scale grids. These algorithms used metrics like migration cost, resource and network heterogeneity, and expected completion time of jobs for the purpose of balancing the load.

Viswanathan et al. [16] developed incremental balancing and buffer estimation strategies to derive optimal load distribution for non-critical load. The job scheduler will first obtain the information about the available memory capacity and computing speed of the sinks and the size of the jobs from the sources. The scheduler will then calculate the optimum load fractions and notify each sink. Then sinks ask the source to transfer the load.

Han et al. [17] introduced a new type of client scheduling proxy that mediates job scheduling between server and clients. Scheduling proxy is elected amongst the clients clustered using network proximity. This proxy serves as a leader of the clients. Scheduling proxy downloads coarse-grained work unit from the server and distributes fine-grained work units to the clients. In order to prevent the failure of scheduling proxy, workers must preserve result until they receive confirmation from the server.

Zhang et al. [18] described a distributed scheduling service framework for real time CORBA. The global scheduling decisions considered the end system's requirements and the entire system. Their results showed the benefits of their framework when the resource management is necessary.

III. System Model

Our Grid system consists of M heterogeneous processors, P1, P2, ..., PN, connected via communication channels assuming an arbitrary topology. We assume that each processor has jobs. The jobs are executed without further user interaction. This assumption eliminates the possibility of dropping of a job due to unavailability of buffer space. The jobs are assumed to arrive randomly at the processors, the inter-arrival time being exponentially distributed with average $1/\lambda i$. The jobs are assumed to require service time that is exponentially distributed with mean $1/\mu i$. Each processor is modeled as a M|M|1 Markov chain, with the number of jobs queued up for processing at each processor representing the state of the system. Job size is assumed to have normal distribution with a given mean and variance.

IV. Novel Adaptive Load Balancing Algorithm

This algorithm is constructed by combining the transfer and location policies of sender-initiated and receiver- initiated [7] algorithms with timer component.The policies on which the proposed algorithm works has been discussed below:

Information Policy*:* The information policy is demand driven, since polling starts only after a node becomes a receiver or sender.

Transfer Policy: The predefined threshold transfer policy based on CPU queue length considers a node as a sender or receiver. A node is identified as a sender/receiver if a new job originating/departing at the node makes the queue length increase/decrease threshold *T*. A node identifies itself as a suitable sender/receiver for a task transfer if accepting /departing the task will not cause the node's queue length to increase/decrease *T*.

Selection Policy*:* All the newly coming tasks to the sender can be selected to be transferred to another node.

Location Policy*:* The responsibility of the location policy is to find a suitable sender or receiver for a node. A node finds suitable node through polling. If a node finds suitable node it transfers the job. Otherwise the node will broadcast a message to increase the threshold *T* by 1. Using this all other nodes will increase threshold *T*, no more polling takes place in high load.

Authorized licensed use limited to: Synopsys. Downloaded on June 28,2023 at 09:24:03 UTC from IEEE Xplore. Restrictions apply.

It will reduce the polling messages. Similarly when a new node joins the network or it becomes a receiver it will start a timer and wait for a sender. If a node does not find a sender in predefined time period $Time_{th}$ then it will broadcast a message to decrease the threshold *T* by 1. So that load can be balanced.

The workings of sender component of NASA (Novel Adaptive Scheduling Algorithm) algorithm are shown in Figure 2 and 4. Initially, we assume each node has information about other nodes in network known as Source S = $\{S_i, S_{i+1}\ldots\ldots S_n\}$. When a new job arrives at site S_i, based on threshold T_i, it decides whether the job is to be processed locally is to be transferred. To transfer the job the node selects a site S_j randomly and polls. If site s_j, queue length is less than S_{jq} then threshold T_j, transfers the job. Otherwise adds S_j into Poll Set. If Poll Set is equal to Poll Limit means it has polled all the nodes in the network. Now, it will broadcast a message to increase the threshold *T* by 1, so other nodes will not poll to the network.

The working of receiver component has been shown in Figure 1 and 3. When a node becomes a receiver it starts timer Time. If it reaches Threshold time Th_{time} then it broadcasts a message to decrease the threshold.

Discussion: At high system loads, the probability of a node being under loaded is negligible, resulting in unsuccessful polls by the sender-initiated component. Unsuccessful polls result is broadcast a message to increase threshold. This scheme prevents future sender-initiated polls at high system loads. Hence, the sender initiated component is deactivated at high system loads, leaving only receiver- initiated load sharing. At low system loads, receiver-initiated does not poll, it starts a timer and waits to get a job. This waiting has a positive effect, resulting in balanced load between nodes.

Procedure Find_Sender ()

```
//S is all nodes in network
S={Si,Si+1…Sn}
// any node departs a job in S called as Si
 Si  departs a job Ji
 / / β  is decide based on network size
Time = 0 and Thtime = β
While Time != thtime
wait to get a job
end while
If  time == th time
broad cast a message to increase T by 1
end if
```

Figure 1: Algorithm for receiver

Procedure Find_Receiver ()

```
// S is all nodes in network
S={Si,Si+1…Sn}
// any node get a job in S called as Si
Si  get a job Ji
// Sqi is queue length and Ti is threshold  of Si
 If Sqi >= Ti  then
//P Pollset
        P={ϕ}
// Q is PollLimit
        Q={ϕ}
End if
while P != Q
//  Selects a node randomly from Source
    Select  node Sk
 //Add node into Pollset
   P={P ∪ Sk }
Poll Sk
if  Skq < Tk then
  Transfer Ji
End if
   If ( P = Q)
Broadcast a message to all node
increase Threshold T
  Break
End if
End while
```

Figure 2: Algorithm for sender

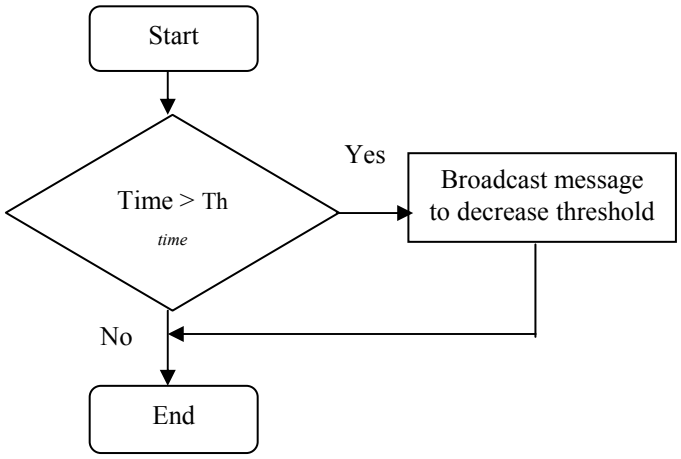

Figure 3: Flow chart for receiver component

Authorized licensed use limited to: Synopsys. Downloaded on June 28,2023 at 09:24:03 UTC from IEEE Xplore. Restrictions apply.

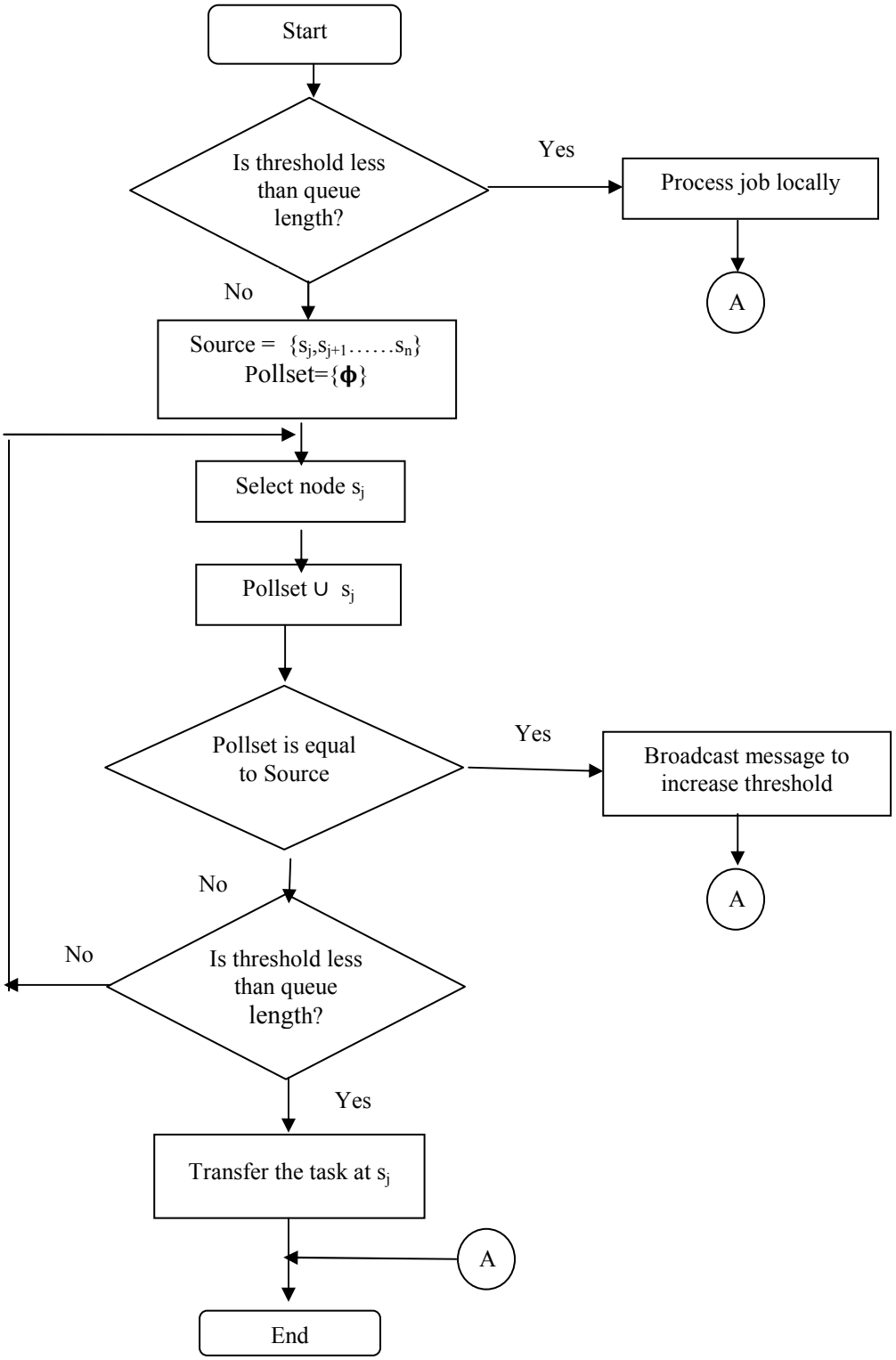

Figure 4: Flow chart for sender component

Authorized licensed use limited to: Synopsys. Downloaded on June 28,2023 at 09:24:03 UTC from IEEE Xplore. Restrictions apply.

I. V. Simulation Results

Experiments were done to prove NASA. It can efficiently assign a newly arrived task on under loaded node. Receiver gets a job from senders after some time. The experiments were implemented on grid simulator. We assume that the simulated grid system includes a fixed 12 number of nodes. A fixed threshold of $T = 1$, Poll limit =2 and $\beta = 10^{-6}$ was used.

In the experiments, the NASA was compared with symmetrically initiate algorithms. For each simulation run, the number of jobs varies. From the graphs we can say that the transfer time of jobs is lesser than symmetrically initiate algorithm because there is balance of load between sender and receiver.

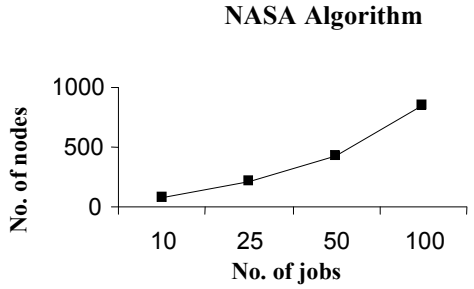

Figure 5: Simulation result of NASA algorithm.

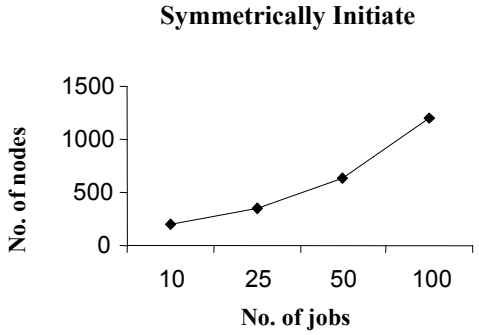

Figure 6: Simulation result of symmetrically initiate algorithm.

VI. Conclusion

Experiments were done to prove NASA. It can efficiently assign a newly arrived task on under loaded node. Receiver can get job from senders after some time. The experiments were implemented on grid simulator. We assume that the simulated grid system includes a fixed 12 number of nodes. In the experiments, the NASA is compared with symmetrically initiate algorithms. For each simulation run, the number of jobs varies. From the graphs we can say that the transfer time of jobs is lesser than symmetrically initiate algorithm because it also balances the load between sender and receiver.

References

[1] I.Foster and C.Kesselman, "The Grid : Blueprint for a future computing infrastructure", Morgan Kaufmann Publishers, USA, 1999.

[2] L.Anand, D.Ghose, and V.Mani, "ELISA: An Estimated Load Information Scheduling Algorithm for distributed computing systems", International Journal on Computers and Mathematics with Applications, Vol.37, Issue 8, pp. 57-85, April 1999.

[3] I.Foster, C.Kesselman, and S.Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations", International Journal of High Performance Computing Applications, Vol. 15, Issue 3, pp. 200-222, 2001.

[4] L.Smarr and C.E.Catlett, "Metacomputing", Communications of the ACM, Vol. 35, Issue 6, pp. 44-52, June 1992.

[5] Y.Feng, D.Li, H.Wu, and Y.Zhang, "A dynamic load balancing algorithm based on distributed database system", Proceedings 4th International Conference on High Performance Computing in the Asia-Pacific Region, China, pp. 949-952, May 2000.

[6] M.Willebeek-LeMair and A.Reeves, "Strategies for dynamic load balancing on highly parallel computers", IEEE Transactions on Parallel and Distributed Systems, Vol. 9, Issue 4, pp. 979-993, September 1993.

[7] N.Shivaratri, P.Krueger, and M.Singhal, "Load distributing for locally distributed systems", IEEE Computer, Vol. 25, Issue 12, pp. 33-44, December 1992.

[8] H.Lin and C.Raghavendra, "A dynamic load-balancing policy with a central job dispatcher (LBC)", IEEE Transactions on Software Engineering, Vol. 18, Issue 2, pp. 148-158, February 1992.

[9] J.Watts and S.Taylor, "A practical approach to dynamic load balancing", IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 3, pp. 235-248, March 1998.

[10] G.Manimaran and C.Siva Ram Murthy, "An efficient dynamic scheduling algorithm for multiprocessor real-time systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 9, Issue 3, pp. 312-319, March 1998.

[11] M.J.Zaki and W.Li.S.Parthasarathy, "Customized dynamic load balancing for a network of workstations", Journal of Parallel and Distributed Computing, Vol. 43, Issue 2, pp. 156-162, June 1997.

[12] J.Krallmann, U.Schwiegelshohn, and R.Yahyapour, "On the design and evaluation of job scheduling algorithms", In 5th Workshop on Job Scheduling Strategies for Parallel Processing, Vol LNCS 1659, pp. 17-42, 1999.

[13] D.G.Feitelson, L.Rudolph, U.Schwiegelshohn, K.C.Sevcik, and P.Wong, "Theory and practice in parallel job scheduling", In 3rd Workshop on Job Scheduling Strategies for parallel processing, Vol LNCS 1291, pp. 1-34, 1997.

[14] Kai L., Albert Y. Z. A Hybrid Policy for Job Scheduling and Load Balancing in Heterogeneous Computational Grids. Sixth International Symposium on IEEE Transactions on Parallel and Distributed System. Hagen berg, Austria, 2007, July 5-8.

[15] Ruchir S., Bhardwaj V., Manoj M. On the Design of Adaptive and De-centralized Load Balancing Algorithms with Load Estimation for Computational Grid Environments.

[16] Sivakumar V., Thomas G. R., Dantong Y., Bharadwaj V. Design and Analysis of a Dynamic Scheduling Strategy with Resource Estimation for Large-Scale Grid Systems. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing. Pittsburgh, USA, 2004 November 8.

[17] Jaesun H., Daeyeon P. Scheduling Proxy: Enabling Adaptive Grained Scheduling for Global Computing System. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing. Pittsburgh, USA, 2004, November 8.

[18] Jiangyin Zhang, Lisa DiPippo, Victor Fay-Wolfe, Kevin Bryan, Matthew Murphy, A Real-Time Distributed Scheduling Service For Middleware Systems, IEEE WORDS 2005.

[19] http://www.gridbus.org/gridsim/

Authorized licensed use limited to: Synopsys. Downloaded on June 28,2023 at 09:24:03 UTC from IEEE Xplore. Restrictions apply.