Optimization and Job Scheduling in Heterogeneous Networks

Abdelrahman Elleithy, Syed S. Rizvi, and Khaled M. Elleithy
Computer Science and Engineering Department University of Bridgeport, Bridgeport, CT USA
{aelleithy, srizvi, elleithy}@bridgeport.edu

***Abstract*— A heterogeneous network is a connected network of different platforms and operating systems. Job scheduling is a problem of selecting a free resource for unexecuted task from a pool of submitted tasks. Furthermore, it is required to find for every resource the best order of the tasks assigned to it. The purpose of this paper is to develop an efficient algorithm for job scheduling in heterogeneous networks. The algorithm should include parameters such as properties of resources and properties of jobs. The algorithm includes a cost function that is required to be optimized which includes parameters such as the total processing time, average waiting time. Our results demonstrate that the proposed algoritghm can be efficiently used to determine the performance of different job scheduling algorithms under different sets of loads.**

I. Introduction

Job scheduling for heterogeneous networks has received significant attention in literature due to its significant effect of the overall performance of such networks [1, 2, 3]. An important component of the management system of a heterogeneous network is an optimal and sub-optimal scheduler. The scheduler should be able to create a schedule after analyzing the pending workload and the free computing resources. The efficiency of a distributed computing system depends on the quality and features of the scheduler. Scheduling in a heterogeneous networked environment involves scheduling over two dimensions, time and space, and on two levels, jobs and computing resources [1].

A. *Problem Identification*

The problem of job scheduling in heterogeneous network is a problem of identifying a resource for every task from the pool of unexecuted tasks. We define the problem using the following three dimensions:

(1) Constraints
There are three types of constraints

(A) Jobs constraints:
- Initial priority
- Time and data dependency
- Preemptability
- Memory size required
- Completion deadline
- Number of processing slots required

(B) Recourses constraints.
- Memory size
- Number of processing slots available
- Processing speed

(C) Scheduling constraints:
- Job advance reservation
- Parallel job partitioning

(2) Load balancing
In order to balance the load among the network we assume that jobs are assigned to processors whenever they are free.

(3) Cost function
It is required to optimize a weighted cost function including with parameters such as total processing time, average waiting time, and average violation of completion deadline.

II. Related Work

There are many approaches reported in literature for dynamic scheduling and load balancing in grid systems. Many of these involve some sort of centralized monitoring system, such as [4, 5, 6, 7], to collect up-to-date information on grid nodes. Such approaches suffer from the fact that the information needs to be kept up-to date as well as additional overhead which impacts negatively the performance. Such a phenomenon is obvious when the system is experiencing a heavy load [2].

Development in computational grid technologies has lead to high scale performances in distributed systems, wherein the grid resources are geographically dispersed and heterogeneous in nature. Nonetheless, a grid site uses a large scale of communication overhead to capture load information. Also, computational grid systems rely on load balancing to enhance the utilization of each node, and minimize the average response time of each jobs. A node in terms of a distributed system has “different processing speed and system resources.” These nodes control the decision making process in load balancing.

Since the load balancing decision is distributed; it is costly to let each node obtain the dynamic state information of the whole system. To address this problem, some algorithm developed a suitable work around; for instance, Mosix which uses a probabilistic approach to choose a random subset of hosted to talk to and cut down communication cost. Diffusion-based approach uses the

Contact author: srizvi@bridgeport.edu

near-neighbor load information to apportion surplus load from heavily loaded areas in the system. [1]

III. Proposed Analytical Model

In this section we discuss how we represent our problem in a three dimensional model. This mathematical representation is a new representation that is not reported in the literature.

A. *Constraints Representation*

There are three types of constraints

(A) Jobs constraints:

1. Initial priority is represented using two dimensional array IP of dimension n*3, IP[i,j], s.t. 1<= i <= n , 1<= j <= 3

   IP [i, 1] represents the priority of the job which is a number between 1 and n.

   IP [i, 2] represents the status of the job. Status equal 0 means the job did not start and it can be assigned to any free processor. Status equal 1 means the job started execution. Status equal 2 means the job is preempted and it can be assigned to any processor. Status equal 3 means the job finished execution.

   IP [i, 3] represents the finished slots if the job is in preempted status.

2. Time and data dependency:

   a. Time dependency is represented using one dimensional array T of dimension n. T[i] = j, means that task number i can not be started before time j.
   b. Data dependency is represented using two dimensional array D of dimension n*n. D[i,j] = 1 means job i can not start before job j is finished , D[i,j] = 0 means job i can start before job j is finished.

   Please note that D[i,i] = 0 for all values 1<= i <= n.

3. Preemptability: is represented using one dimensional array P of dimension n. P[i] = 1, means that task number i can be preempted during execution. P[i] = 0, means that task number i cannot be preempted during execution.

4. Memory size: is represented using one dimensional array M of dimension n. M[i] = j means job i requires memory of size j bytes.

5. Completion deadline: is represented using one dimensional array CD of dimension n. CD[i] = k means job i has to be finished by time k.

6. Number of processing slots required is represented using one dimensional array NPSR of dimension n. NPSR [i] = j means that job i requires j slots.

B. *Recourses Constraints:*

1. Memory size is represented using one dimensional array MP of dimension n. MP[i] = j means processor i has j bytes available for execution of tasks.

2. Number of processing slots is represented using one dimensional array NPS of dimension n. NPS[i] = j means processor i has j slots that can be used for processing tasks.

3. Processing speed is represented using one dimensional array PS of dimension n. PS[i] = j means processor i has a speed of j instructions per slot.

C. *Scheduling Constraints:*

1. Job advance reservation is represented using one dimensional array AR of dimension n. AR[i] = 1 means processor i allows advance reservation. AR[i] = 0 means processor i does not allow advance reservation.

2. Parallel job partitioning is represented using one dimensional array JP of dimension n. JP[i] = 1 means processor i allows partitioning. JP[i] = 0 means processor i does not allow partitioning.

(2) Load balancing

In order to balance the load among all processors, It is required to keep all processors busy. Instead of communicating the status of each processor to all processors, which requires exchanging large amount of data, processors get the next task to execute from the initial priority list (IP).

(3) Cost function

Our cost function will include the following parameters:

P: total processing time
W: average waiting time
V: average violation of completion deadline

The cost function is a weighted function. The following are the weights:

Ψ: weighted cost function
α: weight of total processing time
β: weight of average waiting time

γ: weight of average violation of completion deadline

$$\Psi = \alpha * P + \beta * W + \gamma * V$$

IV. Proposed Parallel Algorithm

The following is the parallel algorithm that will be executed by every processor. Figure (1) shows the initial status of the scheduler:

Select_task ()
{
Repeat for every free processor, p,

Select the highest priority job, k, from IP such that:
IP [k, 2] = 0 did not start, or
IP [k, 2] = 2 job was preempted

Check Time and data dependency:
a. T[k] >= current_clock
b. D [k, i] for all values are satisfied. This condition can be checked using IP
c. M[k] <= MP [p] : satisfy memory constraint

Case
- If all constraints are satisfied, set IP [K ,2] = 1
- If any constraint is violated, select next available task
- If there is no available task, wait for next slot
- If IP [i, 2] = 3 for all values of I then
Finish_simulation_and_Produce_Statistics ();
}

Preemption ()
{
Repeat for busy processors (p) every time slot
- Check for the preemptability of the current task(T)
- If (P [T] = 1) and (current_period = Preemption_period) then
(a) IP [T, 2] = 2
(b) IP [T, 3] = IP [T, 3] + current_period
- Select_task ();
}

Finish_Task_and_Collect)_Statistics ()
{
Repeat for busy processors (p) every time slot and for task (k)

- Check if task(T) has completed NPS(T)
- If task (T) finished execution then
(a) IP [T, 2] = 3

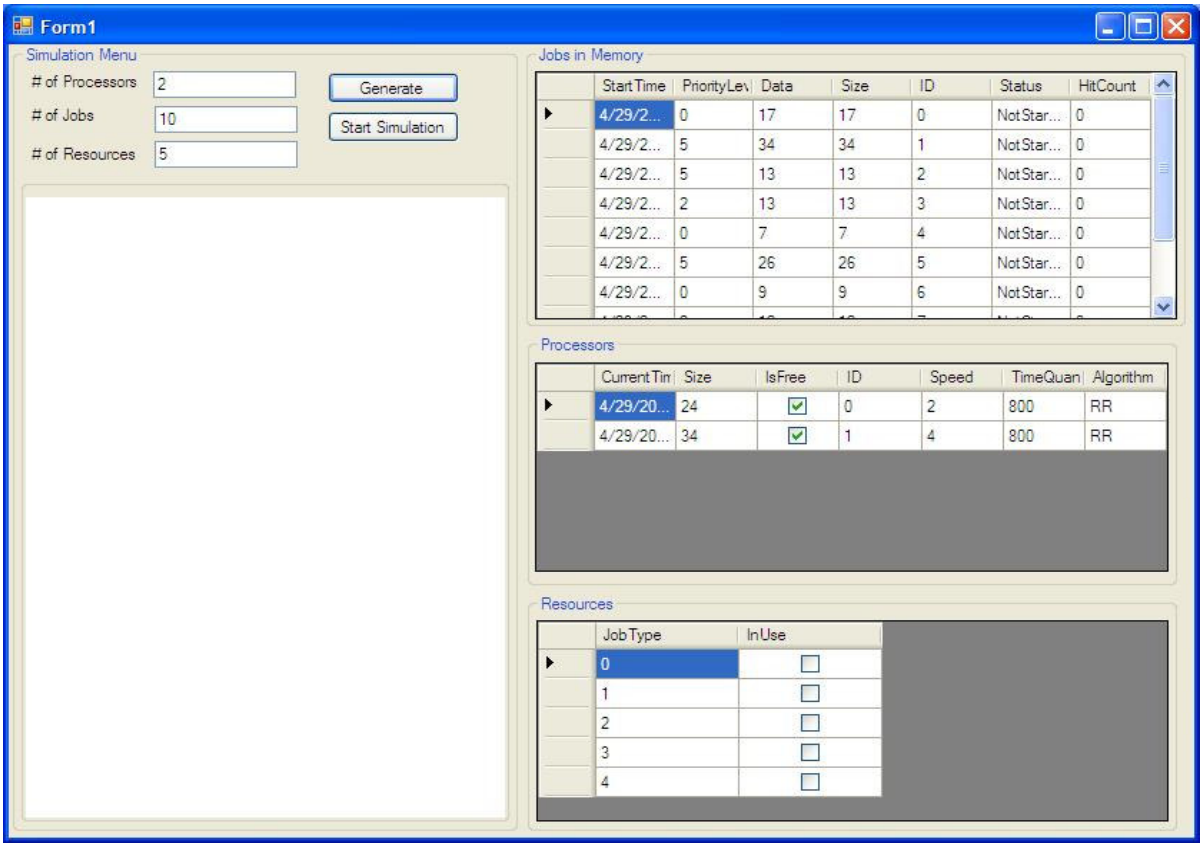

Figure 1: Status of the scheduler before run start for 2 processors, 10 jobs and 5 resources

(b) total processing time = total processing time + current-period

(c) If (current_clock - CD [T])> 0 then
average violation of completion deadline = average violation of completion deadline + (current_clock - CD [T])

}

Finish-simulation-and-Produce-Statistics ()
{
- Update P, V, W
- Calculate $\Psi = \alpha * P + \beta * W + \gamma * V$
- Print Statistics
}

V. Implementation And Experimental Varifications

We have implemented a simplified version of the algorithm using Visual Studio 2005 in C#. The following is a discussion of the implementation of the program. Figure (1) shows the initial status of the scheduler:

Inputs:

The user is allowed to use the visual interface for the following data:
(1) Number of processors
(2) Number of jobs
(3) Number of resources

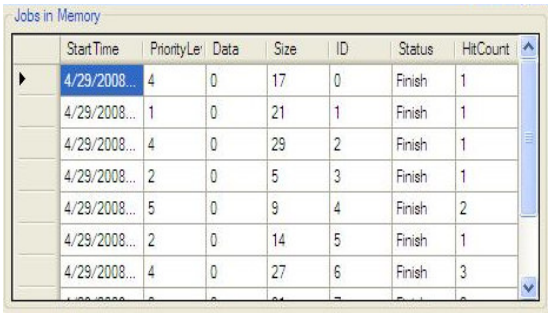

Figure 2: Data about jobs in memory

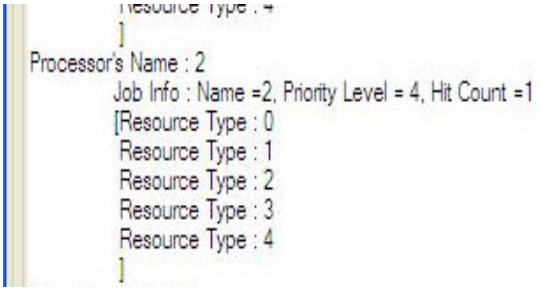

Figure 3: Data of the assigned task.

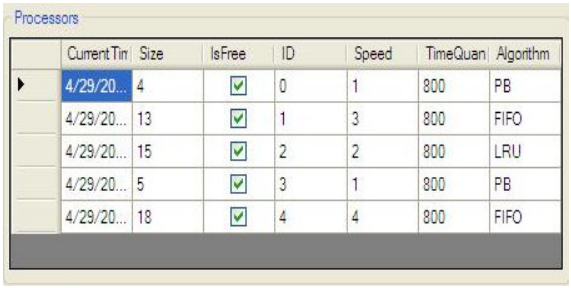

Figure 4: Example of priority algorithms used by the scheduler.

The scheduler generates randomly the following data:
(1) The start time
(2) The priority level
(3) The data size

During the run of the simulation the following data is displayed:

(1) Data about jobs in the memory such as its status, speed and the allocated time. Figure 2 shows the data about jobs in memory.

(2) The detailed status of every processor when ever a job is assigned such as the priority, the resources used for that specific job, and the hit count for that specific processor. Figure 3 shows the status of the assigned task.

(3) The priority algorithm used for that specific processor. The following priority algorithms are supported by the scheduler: PB, FIFO, and LRU.

Figure 4 shows examples of priority algorithms used by the scheduler. Finally, Figure 5 provides the final results based on the proposed algorithm with the comprehensive amount of different statstics.

VI. Conclusion

In this paper we tackled the job scheduling problem in heterogeneous networks by developing a mathematical model and an efficient algorithm that takes into consideration the three types of constraints defined above, balancing load among processors in order to optimize the weighted cost function.

We have implemented a prototype of the scheduler for educational purpose. The implementation can be easily used as an educational tool for teaching concepts of scheduling in heterogeneous networks.

As a continuation of this study in a different course or an independent study, we are planning in the future to do a complete analysis of the algorithm and its performance in terms of different constraints:

- Initial priority
- Time and data dependency

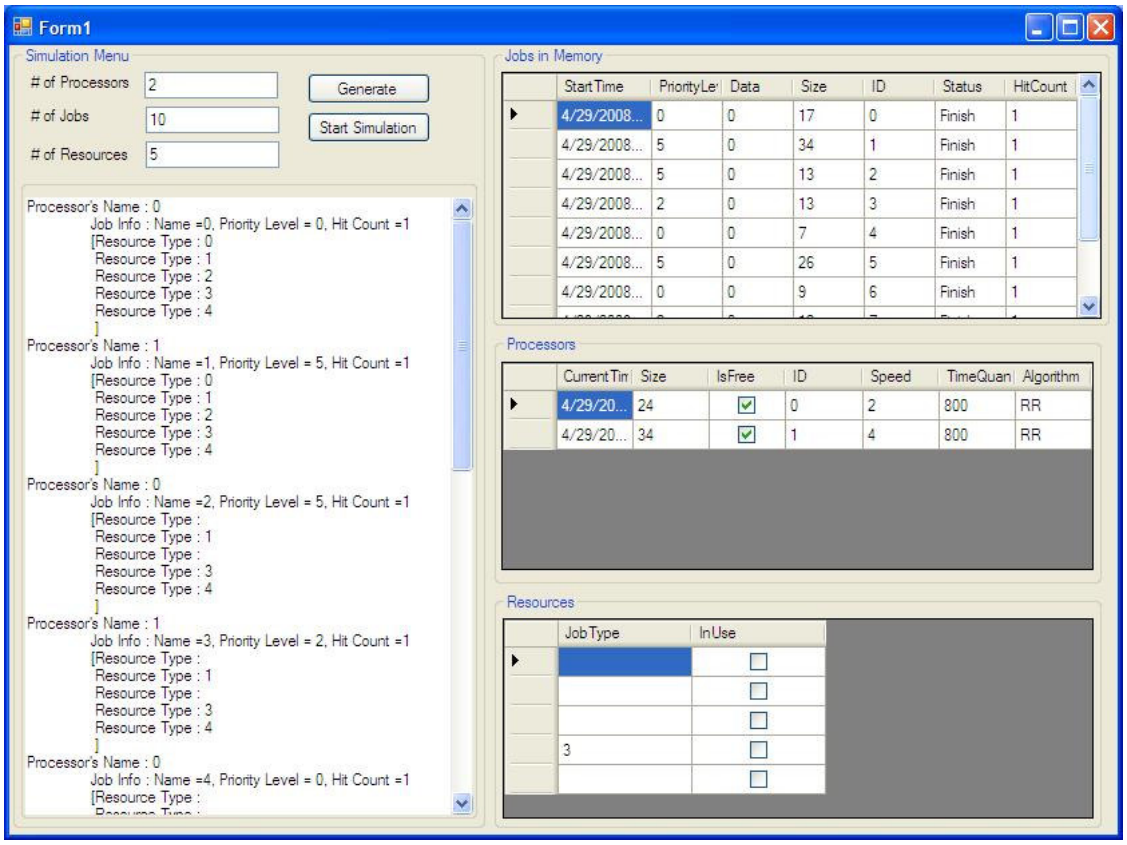

Figure 5: Final results of the simulation for 2 processors, 10 jobs and 5 resources

- Preemptability
- Memory size required
- Completion deadline
- Number of processing slots required
- Memory size
- Number of processing slots available
- Processing speed
- Job advance reservation
- Parallel job partitioning

REFERENCES

[1] K. Lu , Y. Zomaya, "A Hybrid Policy for Job Scheduling and Load Balancing in Heterogeneous Computational Grids," *Sixth International Symposium on Parallel and Distributed Computing* (ISPDC'07), pp. 19-26, 2007.

[2] L. Markov, "Two Stage Optimization of Job Scheduling and Assignment in Heterogeneous Compute Farms," *10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)*, pp. 119-124, 2004

[3] W. Homer, C. Lee, W. Chen, T. Lee, "A Job Schedule Model Based on Grid Environment," *First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07)*, pp. 43-49, 2007.

[4] S. Fitzgerald, I. Foster, C. Kesselman, V. Laszewski, G. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," *Proc of 6th IEEE Symp. on High-Performance Computing*, 1997, pp.365–375, 1997.

[5] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke,"A Computation Management Agent for Multi-Institutional Gridsm" *Proc. 10th IEEE Symp. on High-Performance Computing*, San Francisco, CA, USA, 2001

[6] R. Buyya, J. Abramson, and J. Giddy, J. Nimrod, " Architecture for a Resource Management and Scheduling System in a Global Computational Grid," *4th IEEE Conf. on High-Performance Computing in the Asia-Pacific Region*, China, 2000

[7] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments," *Proceedings of the 9th Heterogeneous Computing workshop* (HCW'2000), pp349-363, 2000.

Authors Biographies

Abdelrahman Elleithy has received his BS in Computer Science in 2007 from the Department of Computer Science and Engineering at the University of Bridgeport, Connecticut, USA . Abdelrahman is currently a MS student and expected to receive his MS in Computer Science in December 2008. Abdelrahman has research interests in wireless communications and parallel

processing where he published his research results papers in national and international conferences.

SYED S. RIZVI is a Ph.D. student of Computer Engineering at University of Bridgeport. He received a B.S. in Computer Engineering from Sir Syed University of Engineering and Technology and an M.S. in Computer Engineering from Old Dominion University in 2001 and 2005 respectively. In the past, he has done research on bioinformatics projects where he investigated the use of Linux based cluster search engines for finding the desired proteins in input and outputs sequences from multiple databases. For last one year, his research focused primarily on the modeling and simulation of wide range parallel/distributed systems and the web based training applications. Syed Rizvi is the author of 45 scholarly publications in various areas. His current research focuses on the design, implementation and comparisons of algorithms in the areas of multiuser communications, multipath signals detection, multi-access interference estimation, computational complexity and combinatorial optimization of multiuser receivers, peer-to-peer networking, and reconfigurable coprocessor and FPGA based architectures.

DR. KHALED ELLEITHY received the B.Sc. degree in computer science and automatic control from Alexandria University in 1983, the MS Degree in computer networks from the same university in 1986, and the MS and Ph.D. degrees in computer science from The Center for Advanced Computer Studies at the University of Louisiana at Lafayette in 1988 and 1990, respectively. From 1983 to 1986, he was with the Computer Science Department, Alexandria University, Egypt, as a lecturer. From September 1990 to May 1995 he worked as an assistant professor at the Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. From May 1995 to December 2000, he has worked as an Associate Professor in the same department. In January 2000, Dr. Elleithy has joined the Department of Computer Science and Engineering in University of Bridgeport as an associate professor. Dr. Elleithy published more than seventy research papers in international journals and conferences. He has research interests are in the areas of computer networks, network security, mobile communications, and formal approaches for design and verification.