Job Scheduling Algorithm based on Dynamic Management of Resources Provided by Grid Computing Systems

I. Ungurean

Department of Computers, Faculty of Electrical Engineering and Computer Science, Stefan cel Mare University of Suceava, Romania, phone: +40-230-320277, e-mail: ioanu@eed.usv.ro

Introduction

The necessity of job scheduling or sharing has occurred within many real life situations [6]. The most common examples include: process scheduling on distributed computing systems, product scheduling on manufacturing systems or scheduling the available resources to many users. The issue of scheduling has generally asked the following question: “Which will be the best method of organizing the workloads, so that it can be ended as fastest as possible?” [1] Within a distributed computing system, requests of processing are randomly received from the system’s users. A good planning of these requests assumes their assigning towards available processors, so that all requests have to be solved as soon as possible.

In the last years, simultaneously with the coming out of distributed and parallel computing systems, which are more and more sophisticated or complex [3, 4], the research in computing jobs scheduling has known a high stimulus. Within this paper, a job scheduling algorithm within a heterogeneous grid system [2] is proposed.

Job scheduling

Drawing up the scheduling issue consists in generally of four main stages:
1. Modeling the system.
2. Defining the jobs processing type.
3. Establishing a cost (or an objective) function.
4. Specifying the constraints.

The model will describe the system, the network type, the number of processors or the network topology, etc. The jobs processing type establishes the scheduling algorithm. In generally, the objective of a scheduling issue (meaning the objective function) is defined by the following: if a set of jobs and a system are known, what mapping of these jobs to processors will be the best, so that the desired cost function is optimized? For instance, the objective might consist of minimizing the next relationship

\[ \text{Total cost} = \text{computing cost} + \text{communication cost}. \]  

Therein, total cost means the time necessary to processing a job or a set of jobs. The total cost includes both the processing time and the communication time. The constraints specific to these issues might refer to limitations concerning the availability of processors and of communication channels within the system.

As regards the scheduling and load balancing, two types of algorithms exist within the specialty literature: centralized and distributed. The centralized approach [1] is not scalable, since when the system’s size increases, the central node will become a bottleneck system. Related to distributed approach, all the nodes of a system participate on taking decisions to load balancing. For these reasons, the load balancing distributed algorithms are more scalable and tolerant to errors [7, 12].

There are many approaches concerning the dynamic scheduling and load balancing specific to grid systems. Most of them base upon a centralized monitoring system, in order to collect data about the nodes of a system [8]. Such approaches are represented by Condor-G [9], Nimrod/G [10] and AppLes [11]. Taking into account that a point where all information stored has to exist within these systems, a reducing of performances will take place simultaneously with the increasing of nodes number.

Modeling the grid system

The first step on carrying out a scheduling algorithm will be represented by the system’s modeling, throughout this algorithm is applied. Therein, a GRID system is assumed to be formed by a collection of nodes nodes \( (\text{node}_1, \, \text{node}_2, \ldots, \, \text{node}_n) \) connected by means of a communication network, where the most used network is the Internet. Each \( \text{node}_i \) participates to grid community by a set of \( r_i \) resources. The resources can be represented by computation ability, storage memory, communication rate, etc. The nodes can be grouped depending on many criteria, such as the type of resources, the physical proximity, communication rate amongst them, etc. Herein, the
distribution on groups is accomplished, so that each group includes nodes relatively near one to each other, as concerns the network transfer delaying point of view. The nodes of a group are usually interconnected by a local network of high speed, a network that might be protected by means of a firewall.

Each group is formed of master node, which is assigned with jobs for the group, and distributes them towards the nodes, by using a scheduling algorithm. The groups of nodes can use different scheduling algorithms. On its turn, the grid disposes of a master node that distributes jobs to groups, by using an algorithm of jobs scheduling. The architecture of the grid modeled within this paragraph is illustrated in Fig. 1.

![Fig. 1. The architecture of a grid system divided in groups, depending upon the communication rate between nodes](image)

Each node is characterized by the computation ability (MIPS), RAM memory, storage memory, etc. Each group might be seen as an equivalent computation node, which comprises the computation ability of all nodes being part of this group.

Herein, the jobs are assumed to be executed by any node of the grid system. Each node is associated to a jobs queue. For each job, the execution time specific to every node of the grid system might be estimated.

Taking into account the situation exposed above, a job is assumed to be executed by any node within the grid system. Each node is associated to a jobs queue. And, each job of the queue is assigned with an estimated execution time.

An algorithm of scheduling and dynamic load balancing

This algorithm carries out two essential issues: scheduling and distribution towards nodes the jobs arriving and the dynamic adjustment of nodes loading into the system, by transferring the jobs from loaded nodes towards the other nodes. Therefore, the distribution of first come, first served (FCFS) with a round robin mechanism of the execution nodes is proposed. The jobs arrive to the master node of the grid system. By using the round robin mechanism, these jobs are distributed towards master nodes associated to each group of nodes. Each group of nodes includes a queue with the jobs waiting to be executed.

A periodical or random information exchange is carried out between the master nodes, as concerns the loading degree of each nodes group. Subsequent to the information exchange, each master node will be able to know the loading degree of the other nodes groups. Afterward this operation, an adjusting algorithm specific to the loading of systems’ nodes is started. The result of executing this algorithm will be represented by continuous reducing of work loading differences within the network, along the system, by means of jobs migration from the nodes highly loaded towards those weekly loaded. Jobs migration from one group to another will be accomplished only if a benefit is achieved, as concerns the execution time of a job. On calculating this benefit, the time necessary for transferring a job from a group of nodes to another will be taken into account.

The same issue is also valid at the level of each group. The master node distributes the jobs received towards the nodes of the group, by using the distribution first come, first served with a round robin mechanism, specific to nodes. Each nod is provided with a local queue, where all jobs expecting to be executed are stored. A periodical or random information exchange is carried out between the nodes of a group; where after the same algorithm of loading the system’s nodes is started.

The adjusting algorithm specific to loading of systems’ nodes within the network will be forwards explained in pseudo-code.

**Exchange information with other nodes from the local group**

*For each job from the queue assigned to local node do*

  *Search node where the execution time + transfer time is minimal*

  *If for the find node the transfer time + execution time is less than the execution time on the current node then*

    *Transfer the job to the node found.*

    *Update the index loading for the local node and the node where the job was transfer.*

  *End if*

*End for*

This algorithm can be run periodically or at each information transfer. When a node receives a new job or finishes executing a job, this will send the updated loading index towards the other nodes.
Experimental results

In order to validate this algorithm, the GSSIM simulator (Grid Scheduling SIMulator) was used. GSSIM represents a simulator created so as to schedule the jobs of grid systems. This was developed by Poznań Supercomputing and Networking Center of Poland and can be used in accordance to Apache Software License. In order to carry out the algorithm’s simulation, an extension (plugin) was created, which implements the proposed algorithm. This extension is created under the form of a java’s class that implements the scheduling interface defined by GSSIM [13].

For instance, GSSIM has implemented three scheduling algorithms[13]: SchedulingFCFS (first come, first served), SchedulingARFCFS (first come, first served with advanced reservation) SchedulingFCFSRR (first come, first served with a round robin mechanism of selecting the resources). Within the present activity, the algorithm described in this paper was implemented, by implementing the interfaces provided by GSSIM. As input data for the algorithm, synthetic data including 10, 25, 50 or 100 jobs were generated; each jobs is provided with the execution time defined and the number of processor necessary so as to be executed. These jobs are distributed towards two types of resources. The first resource includes 6 execution nodes, defined by the following configuration: the first node includes 2 processors of 2000MIPS rate and 2000Mb of node memory, where the other 5 include 4 processors of 2000MIPS rate and 2000MB of node memory. The second resource includes 3 execution nodes, where each node includes 2000MB of memory and 4 processors of 2000MIPS rate (Fig. 2).

![Fig. 2. Gantt chart for 10 jobs distributed to 3 computing nodes](image)

The Gantt chats are achieved after the GSSIM simulation, which result after scheduling the input data from the two types of resources taken into consideration. One might emphasize from these diagrams that the nodes of jobs are loaded by approximately equal jobs. One might also notice that the solution just found is not really the optimal, and the algorithm used can always be improved.

A comparison between the execution times is illustrated in Fig. 3 for a set of 100 jobs, which are distributed by means of the hybrid algorithm and the algorithm first come, first served with a round robin mechanism. The case of execution on 3 nodes is shown at top, while the case of execution on 6 nodes is shown at bottom. The time is represented in minutes on X axis.

![Fig. 3. The execution times (represented in minutes) for the proposed algorithm and the FCFS algorithm, specific to 100 jobs on 3 nodes (at top) and on 6 nodes (at bottom)](image)

Conclusions

An algorithm of job scheduling and dynamic adjustment of nodes loading within a grid system is proposed within the present paper. Considering the logical point of view, the grid system can be seen as a multi-level tree, where the highest level is represented by the grid, the next level is represented by the local groups of nodes, and the lowest level is described by the nodes; the proposed algorithm can be executed at each level of this tree.

Validation of the proposed algorithm was carried out by the help of GSSIM simulator. In order to testing, many jobs packages randomly generated by GSSIM application were used. Two types of computing resources were used: the first was formed of three nodes and the second of six nodes. Subsequent to testing, a comparison between this algorithm and the algorithms already implemented within GSSIM package was attained; the execution time of the jobs will be shorter, in case a mechanism of adjusting the nodes loading is used.
The load balancing algorithm will be continuously developed, if one takes into account the existence of more parameters, such as: the energetic consumption to each node or the memory requirements of each job, and so on.

References


Received 2010 02 15