Different types of systems model for Dynamic Load balancing

Mohammad Haroon
Research Scholar
TMU Moradabad

Prof (Dr) Mohammad Husain
Director (JIT)Barabanki

Abstract: *In this paper presents a number of scalable, extensible system model for load balancing in a computational grid. The system model define the basic work for the load-balancing algorithms. The systems model is include of a grid architecture model, job queue model, communication model, job model, job migration model, and performance objective. The grid architecture model provides a representation and organization of system resources. The job queue model provides two-level architecture for the job-waiting queue at each grid site. The communication model provides an estimate of expected communication costs for message exchange and job transfer among grid sites. The job model provides a representation for jobs, and defines the job information needed by the load-balancing algorithms. The job migration model considers techniques for reducing the opportunities for site thrashing and job starvation. The performance metric for evaluating our load-balancing algorithms is given in the performance objective.*

Architecture model

It is assumed that the grid system consists of a collection of sites *S* connected by a communication network (Figure 1). The set *S* contains *n* sites, labelled as $s_1,\ldots, s_n$. Logically, the architecture is hierarchical and is divided into four levels: the grid, site, cluster and node levels. The capacity of resource management is different at different levels. The node can be a workstation or a processor. The other three levels are now discussed.

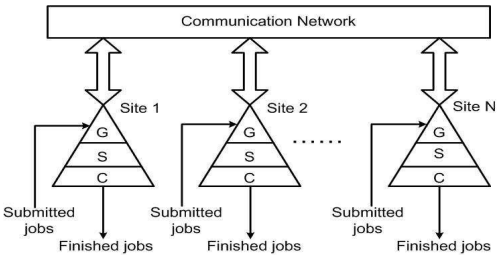


Figure 1 Logical view of the grid architecture; G, S, C are grid, site and cluster levels, respectively.

Cluster level

The cluster level contains a cluster of processors. The processors in a processor cluster share communication bandwidth and are protected by firewalls from the outside world. Processor clusters include tightly coupled multiprocessors such as a Sequent (in which processors communicate via shared-memory), distributed-memory multicomputer such as a Paragon, and loosely coupled workstations such as a Sun 4 cluster (in which processors communicate via message passing).
The management of jobs at cluster level has been addressed by many research and commercial systems, including: Condor ,Load Sharing Facility (LSF) , Portable Batch System (PBS) , Load Leveler , Sun grid Engine/CODEINE , Maui , MOSIX , COSY , A comprehensive review of seven commercial packages and 12 research packages is given in

Site level

The Site is an organizational entity. Each site contains a processor cluster. Each site has a broker denoted by the circle. On the one hand, each site s_i can be regarded as a whole system, and all of its nodes have a common objective. On the other hand, a site s_i can fully centrally control the resources of its nodes, but cannot directly operate the resources of nodes in other sites. In this view, all nodes are cooperative within the same site.
The site model can be extended to support

IJERT

sophisticated architecture. For example, a site may contain multiple administration domains. Each site has the freedom to choose the number of hierarchical levels and of clusters or resources belonging to each level, such that these numbers will best satisfy its management goals.

To clarify the statement and emphasize our main ideas in the dissertation, we will simplify the model of grid site to one computing node with a single processor. Our scheduling can be easily extended to accommodate these complicated cases.

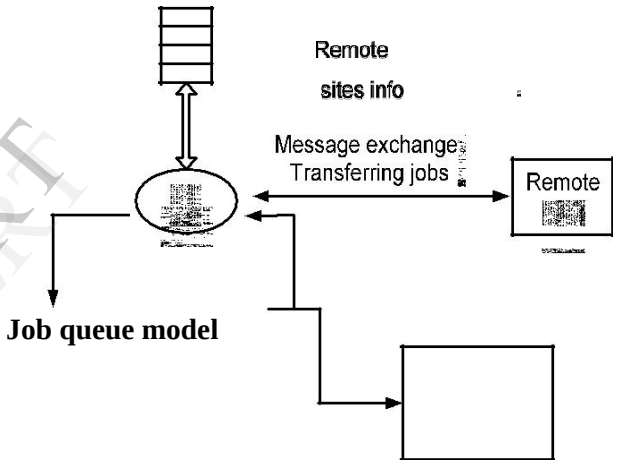
System heterogeneity can be of different kinds—for example, processor speed, memory and disk I/O. A simpler and more practical solution is to use CPU speed alone. It is reasonable to assume that a machine with a powerful CPU will have matching memory and I/O resources. The sites in the grid system may have different processing power. Processing power of a site s_i is denoted as APW_i. For i≠j, APW_i may be different from APW_j. APW_i is presented as the number of computational units that the site can execute per unit of time. The processing power of a grid site s_i is measured by the average processing power across all processors within the grid site s_i if that site has more than one processor. The most common measure of heterogeneity used in literature is the ratio of processing power of the system nodes [54]. APW_i means the ratio of the average processing power of site s_i to the average processing power of the slowest site s_j in the system—in other words, a job that takes one unit of time on the site s_i requires APW_i units of time on the site s_j.

Grid level

All sites at the grid level are organised in a fully distributed way. There is no central broker in the computational grid. The sites themselves are in a completely connected graph (Figure 3.1). The grid sites are mutually independent. Each grid site communicates only with a subset of grid sites while maintaining load information.

Role of site brokers

The site broker handles all communications with other site brokers via core grid middleware on behalf of the local site, and acts as a grid scheduler. It handles all communication with local scheduler on behalf of remote sites. Site brokers are software processes that can run on a computer node in a cluster or on a separate server node. When the node fails, a predetermined backup node becomes the site broker. The focus of this dissertation is on the design of algorithmic mechanisms for grid schedulers.



Job queue model

We assume that there is a global job-waiting queue at each site that holds those jobs waiting to be assigned to local job management system or a remote grid site (Figure 3.2). Jobs that are submitted to the site are first placed in this queue. The site broker will determine that the jobs in the global job-waiting queue are processed at local site or at remote sites. If a job is determined to be processed at the local site, it will be transferred to the underlying job management system at cluster level within the site. We use $GJQ(s_i)$ to denote the global job-waiting queue in site s_i. The jobs in the global

IJERT

job-waiting queue are processed in a "first-come-first-serve order.

The job-waiting queue at site level is different from the job-waiting queue at cluster level. For the following reasons, we used a job-waiting queue at site level.

The implementation complexity of pulling a job from the job-waiting queue managed by cluster-level job management system can be reduced.

Different load-balancing algorithms can be implemented at site level and have not any interference with job management system at cluster level. This incurs no extra work for the underlying job management system.

This approach leads to a flexible and portable solution to the existing grid job management system. It is a compromise between the benefits obtained from load-balancing algorithms applied at site level and the implementation complexity introduced in modifying the job management system running at cluster level. Although a trend is starting to occur as vendors adopt a grid perspective to scheduling, by combining pairs of local and grid schedulers into a single scheduler these systems do not interoperate and are not yet widely used.

Communication model

The sites S are fully interconnected, such that there is at least one communication path between any two sites in S. The only way that inter-site communication can occur is through message passing. There is a non-trivial transfer delay on the communication network between the sites. The transfer delay is different between different pairs of sites. The underlying network protocol guarantees that messages sent across the network are received in the order sent. The sites are interconnected by point-to-point links. There is no efficient broadcasting service available.

In general, the network performance between any site pair(s_i, s_j) is represented as two parameters: a transfer delay TD_{ij} and a data transmission rate BW_{ij}. The communication time for sending a message of Z bytes between these sites is then given

TDij + Z/BWij where Z/BWij is the transmission time. The two parameter abstractly represent the total time for traversing all of the links on the path between s_i and s_j. BW_{ij} is presented as effective data transferring rate in bytes per time of unit, or is characterised in terms of Kb/s. TD_{ij} includes a startup cost and delays incurred by contention at intermediate links on the path between s_i and s_j. TD_{ij} and BW_{ij} can be dynamically forecast by what is known as the Network Weather Service [38]. Other research has been proposed on estimating host distance between any two IP addresses

Job model

For any site, $s_i \varepsilon S$, jobs are arriving at s_i. We assume that the arrival of jobs is a random process with an average delay, λ^{-1}, between two successive arrivals (e.g., the arrivals could be a Poisson process with rate, λ; that is, the delay between two successive arrivals follows an Exponential law with the same rate of change). The jobs are assumed to be computationally intensive, mutually independent, and can be executed at any site. Job execution is not time-shared, but dedicated. As soon as a job arrives, it must be assigned to exactly one site for processing. When a job is completed, the executing site will return the results to the originating site of the job. We use J to denote the set of all jobs generated at S, $J = \{j_1,\ldots, j_r\}$. The following parameters related to the job are created automatically by the system:

bornSite(j_i) denotes the originating site of the job j_i exeSite(j_i) denotes the executing site of the job j_i arrTime(j_i) denotes the arrival time of job j_i, which is the time when the job is generated at bornSite(j_i) endTime(j_i) denotes the finish time of j_i; this includes the job communication time from bornSite(j_i) to exeSite(j_i), waiting time

IJERT

queued at the exeSite(j_i), processing times at the exeSite(j_i), and the communication time it takes to return the processing results from exeSite(j_i) to bornSite(j_i) respTime(j_i) denotes the finish time of j_i. respTime(j_i) endTime(j_i) arrTime(j_i). Each job j_x that arrives at a grid site s_i is represented in two parameters: the amount of computation and the amount of communication. The values for these two parameters may be unknown or can be estimated from prediction techniques. The amount of computation normally has one of the following formats.

An expected execution time $ETC(j_x, s_{std})$, that is, the time that would be taken at a standard platform (with a *APW* equal to 1) for processing that job. On a site s_i with APW_i, the expected execution time of a job $ETC(j_x, s_i)$ will therefore be$ETC(j_x, s_{std})/APW_i$. We assume that the expected execution time $ETC(j_x, s_{std})$ follows a type of probabilistic distribution (for instance, an Exponential, Hyperexponential or Bounded Pareto distribution).

The number of computation unit in a job j_x is denoted as NCU_x. Thus, the
expected execution time for the job *jx* on site *si* is NCUx/APW

In a grid environment, the related file of a job needs to be transferred through much slower internet links if the job is scheduled to run in a remote site. Therefore, the cost of file transfers or the amount of communication must be considered in the scheduling algorithm. The amount of communication is calculated in one of two ways.

A:The file size of a job j_x includes input file size A_{1x} and output file size A_{2x}. Assume that, on average, A_{1x} bytes are required to profile a job and that A_{2x} bytes are required to return a response for the job. A_{1x} and A_{2x} are represented as the number of packets needed to be transferred. Thus, the communication time for job j_x needed for transfer purpose is denoted as follows:

$$\text{commTime}(jx) = \text{t com jx}(si, sj) + \text{Tcomjx}(sj, si)$$

$$\text{Tcomjx}(si, sj) = \frac{\text{TDij}}{\text{BWij}} + \text{A1x}$$

$$\text{Tcomjx}(sj, si) = \frac{\text{TDji}}{\text{BWji}} + \text{A2 x}$$

where Tcomjx (si, sj) denotes the communication time of the job jx from si to sj, Tcomjx (sj, si)denotes the communication it takes to return the processing results from sj to si.

However, due to the changes in the load situations that might occur during the transmission of the job, this job may have to make several moves before it reaches its final destination where it will be processed. Thus, we assume that the job j_x has been
Transfer from Ti to Tj

B. The communication time for running a job in a remote site is set to the computation time divided by *CCR*, where *CCR* is the computation to the communication ratio. By using a range of *CCR* values, different communication time incurred in transit can be accommodated. The computation time is the expected execution time $ETC(j_x, s_{std})$. The communication time means the total of the communication time of transferring a job from its bornsite(j_i) to its final exesite(j_i) and the communication time of sending the execution results from its exesite(j_i) to its bornsite(j_i).

start.

The age of a job is set to 1 when it is moved for the first time, and is incremented by 1 each time the job is moved again.

is a constant that can be adjusted empirically to change the extent to which ageing affects the operation of the scheduler.

The approach promotes the position of transferred job in the global job queue of that site s_x, instead of adding it at the end of the

IJERT

queue. This can considerably reduce the probability that the job will be transferred again, and guarantees the minimization of its response time. We used the approach throughout our simulation to improve the performance of the proposed algorithm.

A more conservative approach was used to reduce the rate at which jobs are moved from one site to another. This can be achieved by restricting the maximum number of jobs transmitted between sites to one job at any given time. This approach is more robust and requires minimal processing time at each site.

Performance objective

Our major objective is to minimize the average (overall) response time for a collection of jobs, here denoted as *ART*. Minimizing the *ART* of the jobs submitted for processing in a parallel/distributed system is a critical performance metric for improving the overall performance of the system. Many load-balancing algorithms have striven to meet this objective of minimising the *ART*

The average response time for a collection of jobs is defined by:

$$ART = \frac{\sum_{i=1}^{u} responseTime(j_i)}{u}$$

where u represents the total number of jobs completed for evaluation purpose.

Note that $u < r$.

To evaluate the performance of our algorithms that developed in Chapter 4–6, we define the improvement factor of algorithm F over another algorithm G as follows in terms of average response time of jobs:

$$\frac{ART(G) - ART(F)}{ART(G)}$$

where *ART(F)* denotes the average response time of jobs using algorithm *F*.

ART(G) denotes the average response time of jobs using algorithm *G*. A positive value of the improvement factor indicates an improvement, while a negative value implies degradation. The value of the improvement factor is presented in terms of percent (%).

Conclusion:

This paper has described both a model for presenting grid resource architecture, and a model for presenting job queue. Then a communication model and job model are also presented. These two models define the information needed to construct cost functions for computation and communication. The migration considerations and major performance objectives were then discussed.

References:

1: Load Sharing Facility. http://www.platform.com/Products/Platform.LSF.Family/.

2 Portable Batch System. http://www.openpbs.org/.

3: Sun Grid Engine / CODEINE. http://www.sun.com/software/gridware/index.xml.

4: Load Leveler. http://www-03.ibm.com/systems/clusters/software/loadleveler.html.

5: COSY.http://www.ccrl-nece.de/~falk/COSY/cosy.shtml.

6: Condor. ://www.cs.wisc.edu/condor/.

IJERT

7: MOSIX. http://www.mosix.org/.

8: A. Barak A. and A. Shiloh, The MOSIX2 management system for linux clusters and organizational Grids, white paper, March 2007.

9: R. Wolski, N. Spring, J. Hayes, The network weather service: A distributed resource performance forecasting service for meta computing, Journal of Future Generation Computing Systems, 15 (5–6) (1999) 757–768.

10: P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, IDMaps: a global internet host distance estimation service, IEEE/ACM Transactions on Networking,

11: A. Agrawal, H. Casanova, Clustering hosts in P2P and global computing platforms, in: Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 12–15 May 2003, pp. 367–373

IJERT