Load Balancing in Distributed System using Genetic Algorithm

Purnima Shah
Government Engineering College,Gandhinagar

S. M. Shah
Government Engineering College, Gandhinagar

ABSTRACT
Distributed systems are characterized by resource multiplicity and system transparency. A variety of widely differing techniques and methodologies for scheduling processes of a distributed system have been proposed. These techniques are broadly classified into three types: task allocation approach, load balancing, load sharing. The main goal of load balancing is to equalize the workload among the nodes by minimizing execution time, minimizing communication delays, maximizing resource utilization and maximizing throughput. The scheduling in distributed system is NP-complete problem even in best conditions, and methods based on heuristic search have been proposed to obtain optimal and suboptimal solutions. This paper presents a new concept for process scheduling in distributed system considering load balancing. In this paper, using the power of genetic algorithms we have shown how to perform load balancing efficiently.

Index Terms
Distributed systems, genetic algorithm, load balancing, scheduling

1. INTRODUCTION
ADVANCEMENT in computer networking technologies have led to increase interest in the use of large-scale parallel and distributed computing systems. Distributed systems are gaining popularity by one of its key feature: resource sharing. A Load balancing algorithm tries to balance the total systems load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance. Effective load balancing algorithms/techniques are used to distribute the processes/load of a parallel program on multiple hosts to achieve goal(s) such as minimizing execution time, minimizing communication delays, maximizing resource utilization and maximizing throughputs. Process scheduling in a distributed system is done in tow phases: in the first phase processes are distributed on computers, and in the second processes execution order on each processor must be determined. Process scheduling in distributed systems has been known to be NP-complete. Genetic algorithm is one of the widely used techniques for constrain optimization. Genetic algorithm is basically search algorithm based on natural selection and natural genetics. Using the power of genetic algorithm process scheduling considering load balancing can be done more efficiently than other conventional methods.

2. LOAD BALANCING ALGORITHMS
Load balancing on multi computers is a challenge due to the autonomy of the processors and the interprocessor communication overhead incurred in the collection of state information, communication delays, redistribution of load etc. Parallel and distributed computing environment is inherently best choice for solving/running distributed and parallel program applications. In such type of applications, a large process/task is divided and then distributed among multiple hosts for parallel computation. In a system of multiple hosts the probability of one of the hosts being idle while other host has multiple jobs queued up can be very high. Here load balancing is likely to improve performance. Such imbalances in system load suggest that performance can be improved by either transferring jobs from the currently heavily loaded hosts to the lightly loaded ones or distributing load evenly/fairly among the hosts .The algorithms known as load balancing algorithms, helps to achieve the above said goal(s).

The processors are categorized according to workload in their CPU queues as heavily loaded (more tasks are waiting to be executed), lightly loaded (less tasks are waiting to be executed in CPU queue) and idle processors/hosts (having no pending work for execution). Here CPU queue length is used as an indicator of workload at a particular processor. Some others workload indicators (also known as load index) have been suggested by researchers. But none of them is found to be an idle one.

Substantial work has been done on load balancing in the past years, taking on a variety of forms. The general problem may be studied in different types of computing environments, using different strategies and at different levels. The system may be loosely coupled, with a number of functionally complete computers connected by one or more networks through which messages may be transmitted and remote resources accessed, or may be a tightly coupled, with several CPU-memory combinations connected by a bus to shared memory and secondary storage. The resources to be shared may be of the same type and capacity (homogeneous system), or of different type and/or capacities (heterogeneous system). The algorithms used for load balancing may require no information, or only information about individual jobs (static algorithm) or may make decisions based on the current load situation (dynamic algorithm). The transfer of a job may be initiated by the originating host (source-initiative algorithm), or by the target host (server-initiative algorithm). The unit of execution that is to transferred/redistributed may range from complete jobs submitted by the users, or individual processes, or even smaller program modules. The units may also be the components of

parallel computations with specific communication requirements. Finally, the transfer of a job may be restricted to be done prior to the start of its execution (initial job placement), or may also be allowed during its execution (process migration).

The algorithm adopted for load balancing is closely related to the type and amount of load and job information assumed to be known to the decision-making modules. Accordingly these algorithms come into two basic categories - static and dynamic.

2.1 Static load balancing algorithms

In these types of algorithms, no dynamic load information is used, the assignments of the jobs to the processing hosts are made a priori using job information (e.g. arrival time, average execution time, amount of resources needed and their intercrosses communication requirements), or probabilistically. However in reality this whole information may not be known a priori (at compile time) and things will get worse if we work in heterogeneous systems where job execution times on processing nodes will vary according to capacity of hosts. System administrators have attempted static balancing of workloads with user accounts for a long time. User accounts workload generated by the users is balanced in the long run. Such a method is simple and potentially effective, but is severely limited by administrative considerations (e.g., students in one class need to be assigned to the same machine) and often in situations in which some of the machines is heavily congested, while others stay almost idle. Periodic reassignment of the user accounts may also be necessary as the user demand change.

Static load balancing can be classified into two categories – optimal and sub-optimal.

2.1.1 Optimal SLB

When all the information regarding the state of the system as well as the resource needs is known an optimal assignment can be made based on some criterion function. Examples of optimization measures are minimizing total process completion time, maximizing utilization of resources in the system, or maximizing system throughput. For example simulated Annealing (SA) and genetic algorithms (GA's) are optimization techniques.

2.1.2 Sub-Optimal SLB

When for some of computations, optimal solution does not exist then sub-optimal methods can be applied. These methods rely on the rules-of-thumb and heuristics to guide a scheduling process. List scheduling is the most popular technique despite of poor performance in high communication delay situations. Lot of static algorithms, taking into account their optimal and sub-optimal nature, has been suggested by researchers so far. This includes approximate algorithms like Solution space enumeration and search, Graph theoretic approach, Mathematical programming and queuing theoretic. Some other are round-robin algorithm, recursive-bisection algorithm, heuristic algorithms and randomized algorithms.

2.2 Dynamic Load Balancing Algorithms (DLB)

The main problem with SLB algorithms was that they assume too much job information which may not be known in advance even if it is available, intensive computation may be involved in obtaining the optimal schedule. Because of this drawback much of the interest in load balancing research has shifted to DLB algorithms that consider the current load conditions (i.e. at execution time) in making job transfer decisions. So here the workload is not assigned statically to the processing hosts as was being done in SLB but instead of this workload can be redistributed among hosts at the runtime as the circumstances changes i.e. transferring the tasks from heavily loaded processors to the lightly loaded ones. Dynamic load balancers continually monitor the load on all the processors, and when the load imbalance reaches some predefined level, this redistribution of work takes place. But as this monitoring steals CPU cycles so care must taken as when it should be invoked. This redistribution does incur extra overhead at execution time.

A DLB algorithm considers following issues:

(1) Load estimation policy

Which determines how to estimate the workload of a particular node of the system.

(2) Process transfer policy

Which determines whether to execute a process locally or remotely.

(3) State information exchange policy

Which determines how to exchange the system load information among the nodes.

(4) Priority assignment policy

Which determines the priority of execution of local and remote processes at a particular node.

(5) *Migration limiting policy,*

Which determines the total number of times a process, can migrate from one node to another.

A large number of algorithms have been proposed, mostly heuristic in nature, as the optimal solution often requires future knowledge and is computationally intensive. The most widely approach for studying DLB algorithms is analytic modeling and simulation. For analytic modeling, the computer system is modeled as a queuing network with job arrivals and their resource consumptions following certain probabilistic patterns.

3. GENETIC ALGORITHM

A Genetic Algorithm (GA) is a search algorithm based on the principles of evolution and natural genetics. GAs combine the exploitation of past results with the exploration of new areas of the search space. By using survival of the fittest techniques

combined with a structured yet randomized information exchange, a GA can mimic some of the innovative flair of a human search. A generation is a collection of artificial creatures (strings). In every new generation, a set of strings is created using information from the previous ones. Occasionally, a new part is tried for good measure. GAs are randomized, but they are not simple random walks. They efficiently exploit historical information to speculate on new search points with expected improvement. The majority of optimization methods move from a single point in the decision space to the next using some transition rule to determine the next point. This point-to point method is dangerous as it can locate false peaks in multimodal (many-peaked) search spaces. By contrast, GAs work from a database of points simultaneously (a population of strings), climbing many peaks in parallel. The probability of finding a false peak is reduced compared to methods that go point to point.The mechanics of a simple GA are surprisingly simple, involving nothing more complex than copying strings and swapping partial strings. Simplicity of operation and power of effect are two main attractions of the GA approach. The effectiveness of the GA depends upon an appropriate mix of exploration and exploitation. Three operators to achieve this are: selection, crossover, and mutation.Selection according to fitness is the source of exploitation. The mutation and crossover operators are the sources of exploration. In order to explore, they must disrupt some of the strings on which they operate. The trade-off of exploration and exploitation is clearest with mutation. As the mutation rate increases, mutation becomes more disruptive until the exploitative effects of selection are completely overwhelmed. More information is provided on these operators in [2].

4. THE GA-BASED ALGORITHM

Genetic algorithms, as powerful and broadly applicable stochastic search and optimization techniques, are the most widely known types of evolutionary computation methods today. In general, a genetic algorithm has five basic components as follows:

1.An encoding method that is a genetic representation (genotype) of solutions to the program.

2.A way to create an initial population of individuals (chromosomes).

3.An evaluation function, rating solutions in terms of their fitness, and a selection mechanism.

4.The genetic operators (crossover and mutation) that alter the genetic composition of offspring during reproduction.

5.Values for the parameters of genetic algorithm.

4.1 Genotype

In the GA-Based algorithms each chromosome corresponds to a solution to the problem. The genetic representation of individuals is called Genotype.

4.2 Initial Population

A genetic algorithm starts with a set of individuals called initial population. Most GA-Based algorithms generate initial population randomly.

4.3 Fitness Function

As discussed before, the main objective of GA is to find a schedule with optimal cost while load-balancing; processors utilization and cost of communication are considered. We take into account all objectives in following equation. The fitness function of a Schedule T should be such that a fitter solution (Schedule) has less execution time, less communication cost, higher processor utilization and maximum system throughput.

4.4 Selection

The selection process used here is based on spinning the roulette wheel, which each chromosome in the population has a slot sized in proportion to its fitness. Each time we require an offspring, a simple spin of the weighted roulette wheel gives a parent chromosome.

4.5 Crossover

Crossover is generally used to exchange portions between strings. Crossover is not always affected, the invocation of the crossover depends on the probability of the crossover Pc. Two crossover operators are given. The GA uses one of them, which is decided randomly.

4.5.1 Single-Point Crossover

This operator randomly selects a point, called Crossover point, on the selected chromosomes, then swaps the bottom halves after crossover point, including the gene at the crossover point and generate two new chromosomes called children.

4.5.2 Proposed Crossover

This operator randomly selects points on the selected chromosomes, then for each child non-selected genes are taken from one parent and selected genes from the other.

4.6 Mutation

Mutation is used to change the genes in a chromosome. Mutation replaces the value of a gene with a new value from defined domain for that gene. Mutation is not always affected, the invocation of the Mutation depend on the probability of the Mutation Pm. There are different mutation operators. The GA uses one of them, which is decided randomly.

5. CONCLUSION

Scheduling in distributed operating systems has a significant role in overall system performance and throughput. The scheduling in distributed systems is known as an NP-complete problem even in the best conditions. We have presented a new concept using GA-Based method to solve this problem

Genetic algorithms can be best to solve NP-complete problem. GA can be easily parallelized. This algorithm considers multi objectives in its solution evaluation and solves the scheduling problem in a way that simultaneously minimizes execution time

and communication cost, and maximizes average processor utilization and system throughput.

6. REFERENCES

[1] M. Deriche, M.K. Huang, and Q.T. Tsai, "Dynamic Load-Balancing in Distributed Heterogeneous Systems under Stationary and Bursty Traffics," Proc. 32nd Midwest Symp. Circuits and Systems, vol. 1, pp. 669-672, 1990.

[2] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass.: Addison-Wesley, 1989.M.

[3] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma "Performance Analysis of Load Balancing Algorithms" World Academy of Science, Engineering and Technology 38 2008.

[4] Melanie Mitchell, An introduction to genetic algorithms: Easterm Economy Edition, 2004.

[5] C. Lu and S.-M. Lau. A performance study on load balancing algorithms with process migration. In Proceedings, IEEE TENCON 1994, pages 357-64, Aug.1994.

[6] Y. T. Wang and R. J. T. Morris Load sharing in distributed systems. IEEE Trans. Comput., C-34(3), Mar. 1985.

[7] Gilbert Syswerda, Jeff Palmucci, "The application of Genetic Algorithms to Resource Scheduling," Proc. Fourth International Conference on Genetic Algorithms, pp.502-508, 1991.