International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-2 Number-4 Issue-6 December-2012

An Efficient load balancing using Genetic algorithm in Hierarchical structured distributed system

Priyanka Gonnade[1], Sonali Bodkhe[2]
Mtech Student Dept. of CSE, Priyadarshini Instiute of Engineering and Technology, Nagpur, India[1]
Faculty Dept. of CSE, Priyadarshini Instiute of Engineering and Technology, Nagpur, India[2]

Abstract

In this paper, a genetic algorithm based approach for job scheduling in distributed system considering dynamic load balancing is discussed. The underlying distributed system has hierarchical structure and job scheduling is done in two levels: group level and node level. Scheduling in distributed system involves deciding not only when to execute a process, but also where to execute it. A proper job scheduling will enhance the processor utilization, reduces execution time and increases system throughput. A power of Genetic algorithm will give the optimal solution for scheduling of job. The job scheduling is centralized at each node in a hierarchy and genetic algorithm is applied to each central node. This centralized job scheduling policy considers load balancing to prevent the node connected in the system from getting overloaded or become idle ever(if possible).

Keywords

Heterogeneous distributed computing system (HDCS), Genetic Algorithm, Global Load Balancer (GLB), Local Load Balancer (LLB), and Designated Representative (DR).

1. Introduction

Distributed heterogeneous computing is being widely applied to a variety of large size computational problems. These computational environments are consists of multiple heterogeneous computing modules, these modules interact with each other to solve the problem [2,3]. In a Heterogeneous distributed computing system (HDCS), processing loads arrive from many users at random time instants. A proper scheduling policy attempts to assign these loads to available computing nodes so as to complete the processing of all loads in the shortest possible time. There are number of techniques and methodologies for scheduling processes of a distributed system. These are task assignment, load-balancing, load-sharing approaches [4,5]. Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors.
In task assignment approach, each process submitted by a user for processing is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance. In load sharing approach simply attempts to conserve the ability of the system to perform work by assuring that no node is idle while processes wait for being processed. In load balancing approach, processes submitted by the users are distributed among the nodes of the system so as to equalize the workload among the nodes at any point of time.

A Genetic Algorithm (GA) is a search algorithm based on the principles of evolution and natural genetics [1, 2, 7, 8, 9]. GA combines 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.

The organization of this paper is as follows. After the introduction in section 1, different load balancing approaches has been elaborated in section 2. Problem definition and system model is given in section 3. Load balancing using GA has been proposed in section 4. Finally, concluding remarks appear in section 5.

2. Organization of Load Balancing schemes

The load balancing algorithms in general purpose distributed computing systems is presented and the organization of the different load balancing schemes is shown in Figure 2.1[4, 5, 7, 8, 9]

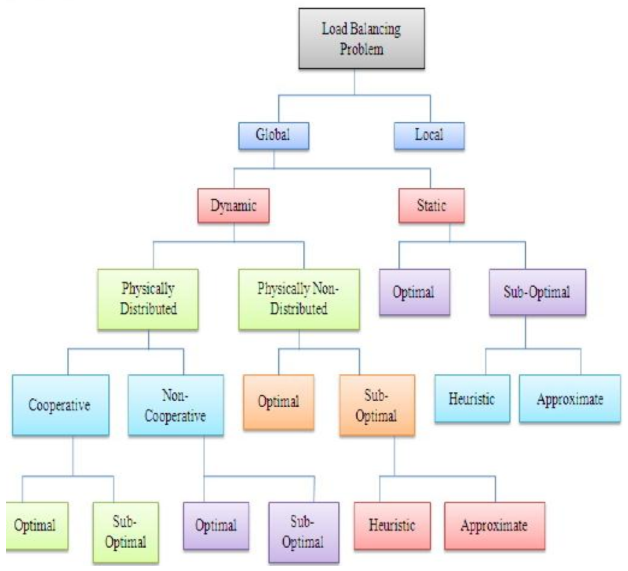

Figure 2.1: Load balancing schemes

2.1 Local vs. Global[4]

A distinction is drawn between local and global scheduling at the top level. The local scheduling discipline determines how the processes resident on a single CPU is residing and executed. A global scheduling policy uses information about the system to allocate processes to multiple processors to optimize a system-wide performance. Grid scheduling falls into the global scheduling branch.

2.2 Static versus Dynamic[4,5,7]

The Static load balancing, also known as deterministic distribution, assigns a given job to a fixed resource. Every time the system is restarted, the same binding task resource is used without considering changes that may occur during the system lifetime. In this approach, every task comprising the application is assigned once to a resource. So, the placement of an application is static, and a firm estimate of the computation cost can be made in advance of the actual execution.

The Dynamic load balancing takes into account the fact that the system parameters may not be known beforehand. That's why we don't use a fixed or static scheme will eventually produce poor results .A dynamic strategy gives good results rather then the static. A dynamic strategy is usually executed several times and may reassign a previously scheduled task to a new resource based on the current state of system environment.

The benefits of dynamic over static load balancing are that the system needs not be aware of the run-time behaviour of the application before execution. For dynamic strategies, the main problem is how to characterize the exact workload of a system, while it changes in a continuous way. Dynamic strategies can be applied both for homogeneous or heterogeneous platforms with different degree of performances.

2.3 Optimal vs. Suboptimal[4,8,9]

All information regarding the state of resources and the jobs is known, an optimal assignment could be made based on some criterion function, such as minimum makespan and maximum resource utilization. Due to the NP-Complete nature of scheduling algorithms and the difficulty in Grid scenarios to make reasonable assumptions which are usually required to prove the optimality of an algorithm, current research tries to find suboptimal solutions, which can be further divided into the following two general categories[4].

1. Approximate and
2. Heuristic.

2.4 Approximate vs. Heuristic[7,8]

The approximate algorithms used in formal computational models, but instead of searching the entire solution space for an optimal solution, they are satisfied when a solution that is sufficiently good is found. In the case where a metric is available for evaluating a solution, this technique can be used to reduce the time taken to find an acceptable schedule.

2.5 Distributed vs. Centralized[4,9]

The responsibility of dynamic load balancing is for making global decisions may lie with one centralized location, or be shared by multiple distributed locations. The centralized strategy has the advantage for the implementation, but suffers from the lack of scalability, fault tolerance and the possibility of becoming a performance bottleneck. In distributed strategy, the state of resources is distributed among the nodes that are responsible for managing their own resources or allocating tasks residing in their queues to other nodes.

2.6 Cooperative vs. Non-cooperative[7,9]

If a distributed load balancing mode is adopted then the next issue that should be considered is whether the nodes involved in job balancing are working independently or cooperatively. In the noncooperative case, an individual system load balancing acts as alone as autonomous entities and make the decisions regarding their own objectives independently of these decisions effects about the rest of the system.

3. Problem definition and system model

3.1 Heterogeneous distributed computing system

Heterogeneous distributed computing system (HDCS)[10,11,13] utilizes a distributed suite of different high-performance machines, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements. Distributed computing provides the capability for the utilization of remote computing resources and allows for increased levels of flexibility, reliability, and modularity. In heterogeneous distributed computing system the computational power of the computing entities are possibly different for each processor as shown in figure 3.1.

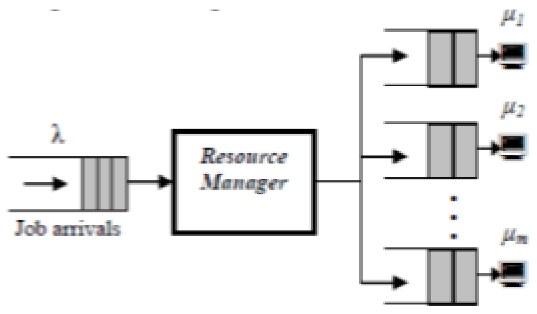

Figure 3.1: Heterogeneous distributed system

A large heterogeneous distributed computing system (HDCS) consists of potentially millions of heterogeneous computing nodes connected by the global Internet. The applicability and strength of HDCS are derived from their ability to meet computing needs to appropriate resources. Resource management sub systems of the HDCS are designated to schedule the execution of the tasks that arrive for the service. HDCS environments are well suited to meet the computational demands of large, diverse groups of tasks. The problem of optimally mapping also defined as matching and scheduling.

3.2 Hierarchical structure

In this paper, it is supposed that underlying distributed system has hierarchical structure. Fig.3.2 shows this structure[6,10]. In the designed structure, load balancing is done in two levels: group level, which is done by **Global Load Balancer (GLB)** and node level, that is done by **Local Load Balancer(LLB).** This case cause significantly lower communication overheads that aroused from collecting of the state information. Several nodes make one group. In any group, there is one node as designated representative (DR). DRs do local load balancing and communicate with the GLB. Nodes of each group are connected only to their group's DR. Task that is allocated to any group and node is processed in that group and node, and there is no other replacement. There is no connection between groups. GLB is connected to each group's DR. Load balancing mechanism is centralized in both levels.

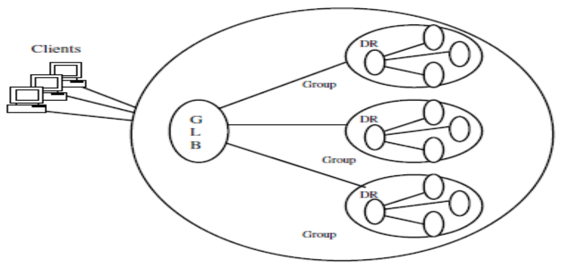

Figure 3.2: **the underlying system**

3.3 Dynamic load distribution algorithms

A dynamic load distribution algorithm must be general, adaptive, stable, fault tolerant and transparent to applications. Load balancing algorithms can be classified as (i) global vs. local, (ii) centralized vs. decentralized, (iii) Non-cooperative vs. cooperative, and (iv) adaptive vs. non adaptive [7,13]. In this paper we have used centralized load balancing algorithm, a central node collects the load information from the other computing nodes in HDCS. Central node communicates the assimilated information to all individual computing nodes, so that the nodes get updated about the system state. This updated information enables the nodes to decide whether to migrate their process or accept new process for computation. The computing nodes may depend upon the information available with central node for all allocation decision.

Scheduling of tasks in a load balancing distributed system involves deciding not only when to execute a process, but also where to execute it. Accordingly, scheduling in a distributed system is accomplished by two components [4]: the *allocator* and the *scheduler*. The allocator decides where a job will execute and the scheduler decides when a job gets its share of the computing resource at the node.

4. Load balancing using Genetic algorithm

In this section, an algorithm which utilizes GA for load balancing in HDCS is given. Genetic algorithms work with a population of the potential solutions of the candidate problem represented in the form of chromosomes. Each chromosome is composed of variables called genes. Each chromosome (genotype) maps to a fitness value (phenotype) on the basis of the objective function of the candidate problem. Jobs arrive at unknown intervals for processing and are placed in the Global central scheduler queue of unscheduled tasks from which tasks are assigned to processors. Each task is having a task number and a size. GA follows the concept of solution evolution by stochastically developing generations of solution populations using a given fitness statistic. They are particularly applicable to problems which are large, nonlinear and possibly discrete in nature, features that traditionally add to the degree of complexity of solution. Due to the probabilistic development of the solution, GA does not guarantee optimality even when it may be reached. However, they are likely to be close to the global optimum. This probabilistic nature of the solution is also the reason they are not contained by local optima. The proposed algorithm for load balancing is presented in figure 4.1.

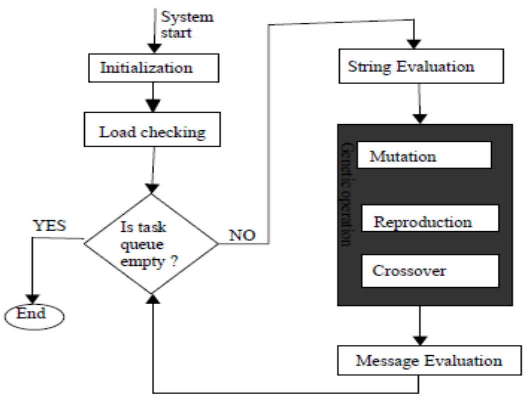

Figure 4.1: Load balancing algorithm

4.1 Genotype

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

4.2 Initial Population

This will initialize a population of possible solutions. This can be achieved using the *sliding window* technique. The window size is fixed, with the number of elements in each string equal to the size of the window.

4.3 Fitness Function

The main objective of GA is to find a node with optimal cost for load-balancing. The fitness function will have a fitter node where the task should be transferred such that it has less execution time, less communication cost, higher processor utilization and maximum system throughput.

The objective is derived on the basis of the load deviation that occurs in each node at each level. In ideal case i.e. when the load is evenly distributed among the nodes, load deviation is zero. But often it may not be possible. Therefore, load deviation is minimized to its fullest possible extent. In order to calculate the load deviation amongst the nodes, we have to calculate the total load of the node connected in the system. The mean can be computed using total load and number of nodes (m) in the network.

The calculations are as follows:

At any node i, load at a node is obtained by using the equation.

$$L_{i=} \sum_{for\ all\ r}[n^p + n^q + n^r] \qquad (1)$$

where r is the numbers of jobs allocated to the node i and p,q,r are the number of pending, queued or running jobs.

$$L_{total=} \sum_{i=1}^{m}[L_i] \qquad (2)$$

Mean load of a node is obtained as-

$$L_{mean=} \frac{L_{total}}{m} \qquad (3)$$

Thus, objective function for the load deviation of a node is given as –

$$min\left(\sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(L_j - L_{mean}\right)^2}\right) \qquad (4)$$

where Lj is the load at the node j, Lmean is the average load and m is the number of nodes Equation (4) depicts the overall load deviation at each level in a distributed system. In this, objective is to minimize this value to get the optimal load distribution.

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. A *Single-Point Crossover* 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 generates two new chromosomes called children.

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.

5. Conclusion

This paper deals with the load balancing in hierarchical structured distributed system with emphasis on the observation of load deviation and load distribution among the nodes. Load balancing 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 and evaluated a GA-Based method to solve this problem. This algorithm considers multiobjectives in its solution evaluation and provides a node with a load that protects node from being overloaded or idle which will simultaneously minimizes maxspan, communication cost and maximizes average processor utilization.

References

[1] Kashani, M.H.; Jamei, M.; Akbari, M.; Tayebi, R.M. Utilizing Bee Colony to Solve Task Scheduling Problem in Distributed Systems Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2011.

[2] Page, A.J., Keane, T.M. and Naughton, T.J. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. Journal of Parallel and Distributed Computing, 2010 Vol. 70, 758-766.

[3] Ahwaz, Iran Mortazavi, S.S.; Rahmani, A.M. Two Hierarchical Dynamic Load Balancing Algorithms in Distributed Systems, 2009. AH-ICI 2009.

[4] Bibhudatta Sahoo, Sudipta Mohapatra, and Sanjay Kumar Jena. A Genetic Algorithm Based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2008, Las Vegas, Nevada, USA, July 14-17, 2008, 2 Volumes. CSREA Press 2008, ISBN 1-60132-084-1.

[5] Nikravan, M. and Kashani, M.H. A Genetic Algorithm For Process Scheduling In Distributed Operating Systems Considering Load balancing in Proceedings of the 21th European Conference on Modeling and Simulation, 645-650, 2007.

[6] Gamal Attiya & Yskandar Hamam. Two phase algorithm for load balancing in heterogeneous distributed systems. Proc. 12th IEEE EUROMICRO conference on Parallel, Distributed and Network-based processing, Coruna, Spain 2004, 434-439.

[7] A. Y. Zomaya, & Y. H. The. Observations on using genetic algorithms for dynamic loadbalancing. IEEE Transactions on Parallel and Distributed Systems, 12(9), 2001, 899-911.

[8] A. Y. Zomaya, C. Ward, & B. Macey. Genetic Scheduling for Parallel Processor Systems. Comparative Studies and Performance Issues, IEEE Transaction Parallel and Distributed Systems, 10(8), 1999, 795-812.

[9] Seong-hoon Lee, Tae-won Kang, Myung-sook KO, Gwang-sik Chung, Joon-min Gil and Chong-sun Hwang. A Genetic Algorithm Me hod for Sender-based Dynamic Load Balancing Algorithm in Distributed Systems. First Intemational Conference on Knowledge-Based Intelligent, 1997.

[10] Iman Barazandeh,S.S. Mortazavi,A.M. Rahmani. Two New Biasing Load Balancing Algorithms in Distributed Systems .IEEE conference on distributed computing, 2009.

[11] F. Bonomi, & A. Kumar. Adaptive Optimal Load-Balancing in a Heterogeneous Multiserver System with a Central Job Scheduler. IEEE Trans. on Computers, 39(10) 1990, 1232-1250.

[12] E.S.H.Hou, N.Ansari, & H.Ren. A Genetic Algorithm for Multiprocessor Scheduling. IEEE Trans. On Parallel and Distributed Systems, 5(2), 1994, 113-120.