Study of Genetic Algorithm for Process Scheduling in Distributed Systems

Usha Barad

1,PG Student of Computer Engineering, Merchant Engineering College, Gujarat, India

Abstract : This paper presents and evaluates a new method for process scheduling in distributed systems. The problem of process scheduling in distributed system is one of the important and challenging area of research in computer engineering. Scheduling in distributed operating system has an important role in overall system performance. Process scheduling in distributed system can be defined as allocating processes to processor so that total execution time will be minimized, utilization of processors will be maximized and load balancing will be maximized. Genetic algorithm is one of the widely used techniques for constrain optimization. The scheduling in distributed systems is known as an NP-complete problem even in the best conditions, and methods based on heuristic search have been proposed to obtain optimal and suboptimal solutions. In this, paper using the power of genetic algorithms. We solve this problem considering load balancing efficiently. We evaluate the performance and efficiency of the proposed algorithm using simulation result.

Keyword: Distributed system, scheduling, Genetic algorithm, load balancing

I. Introduction

Scheduling in distributed operating systems is a critical factor in overall system efficiency. A Distributed Computing system (DCS) is comprised of a set of Computers (Processors) connected to each other by communication networks. Process scheduling in a distributed operating system can be stated as allocating processes to processors so that total execution time will be minimized, utilization of processors will be maximized, and load balancing will be maximized. The computational complicated process cannot be executed on the computing machine in an accepted interval time. Therefore, they must be divided into small sub-process. Process scheduling in distributed system is done in two phases: in first phase processes are distributed on computers and in second processes execution order on each processor must be determined [1]. Several methods have been proposed to solve scheduling problem in DCS. The methods used to solve scheduling problem in distributed computing system can be classified into three categories graph theory based approaches , mathematical models based methods and heuristic techniques [1]. Heuristic algorithm can be classified into three categories iterative improvement algorithms [2], the probabilistic optimization algorithms and constructive heuristics. Heuristic can obtain sub optimal solution in ordinary situations and optimal solution in particulars. The first phase of process scheduling in a distributed system is process distribution on computer. The critical aspects of this phase are load balancing. Recently created processes may be overloaded heavily while the others are under loaded or idle. The main objectives of load balancing are to speared load on processors equally, maximizing processors utilization and minimizing total execution time [4]. The second phase of process scheduling in distributed computing system is process execution ordering on each processor. Genetic algorithm used for this phase. Genetic algorithm is guided random search method which mimics the principles of evolution and natural genetics . Genetic algorithms search optimal solution from entire solution space. In dynamic load balancing, processes must be dynamically allocated to processors in arrival time and obtain a near optimal schedule, therefore the execution of the dynamic load balancing algorithm should not take long to arrive at a decision to make rapid task assignments.

A GA starts with a generation of individuals, which are encoded as strings known as chromosomes. A chromosome corresponds to a solution to the problem. A certain fitness function is used to evaluate the fitness of each individual. Good individuals survive after selection according to the fitness of individuals. Then the survived individuals reproduce offspring through crossover and mutation operators. GA-based algorithms have emerged as powerful tools to solve NP-complete constrained optimization problems, such as traveling salesman problem, job-shop scheduling and flow-shop scheduling, machine learning, VLSI technology, genetic synthesis and etc.In this paper using the power of genetic algorithms we solve this problem considering load balancing efficiently. The proposed algorithm maps each schedule with a chromosome that shows the execution order of all existing processes on processors. The fittest chromosomes are selected to reproduce offspring; chromosomes which their corresponding schedules have less total execution time, better load-balance and processor utilization. We assume that the distributed system is non-uniform and non-preemptive, that is, the processors may be different, and a processor completes current process before executing a new one. The load-balancing mechanism used in this paper only schedule processes without process migration and is centralized.

2. Model and Problem Definition

2.1. System Model

The system used for simulation is loosely coupled non-uniform system, all task are non-pre-emptive and no process migration are assumed. The process scheduling problem considered in this paper is based on the deterministic model. The most important purposes of distributed system are assigned as providing an appropriate and efficient environment for sharing resource, having an acceptable speed and high reliability and availability.Network topology, processors speed, communication channels speed and so on. Since we study a deterministic model, a distributed system with *m* processors, *m* > 1 should be modeled as follows:P= {p1, p2, p3,..., pm} is the set of processors in the distributed system. Each processor can only execute one process at each moment, a processor completes current process before executing a new one, and a process cannot be moved to another processor during execution. G is an m × m matrix, where the element guv 1≤ u,v ≤ m of G, is the communication delay rate between Pu and Pv. H is an m × m matrix, where the element huv 1≤ u, v ≤m of H, is the time required to transmit a unit of data from Pu and Pv. It is obvious that huu=0 and ruu=0.T = {t1, t2, t3,...,tn} is the set of processes to execution. E is an n × m matrix, where the element eij 1 ≤ i ≤ n, 1 ≤ j ≤ m of E, is the execution time of process ti on processor pj. In distributed systems the execution time of an individual process ti on all processors is equal.F is a linear matrix, where the element fi 1≤ i ≤ n of F, is the target processor that is selected for process ti to be executed on.C is a linear matrix, where the element ci i ≤ i ≤ n of C, is the processor that the process ti is presented on just now.

The problem of process scheduling is to assign for each process ti Є T a processor fi Є P so that total execution time will be minimized, utilization of processors will be maximized, and load balancing will be maximized.

The processor load for each processor is the sum of process execution times allocated to that process [1].

$$Load\ (p_i) = \sum_{j=1}^{\substack{No.of\ allocated\\ processes\ on\\ processor\ i}} a_{j,i} + \sum_{k=1}^{\substack{No.of\ new\ assigned\\ processes\ on\\ processor\ i}} a_{k,i} \quad \ldots\ldots (1)$$

The length or *max span* of schedule T is the maximal finishing time of all the processes or maximum load. Also, communication cost (CC) to spread recently created processes on processors must be computed:

Maxspan (T) = max (Load (p_i)) for each 1≤i≤ Number of processors … (2)

$$CC(T) = \sum_{i=1}^{nNo.of\ new\ processes} (r_{c_i f_i} + h_{c_i f_i} X d_i) \quad \ldots\ldots (3)$$

The processor utilization for each processor is obtained by dividing the sum of processing times by scheduling length. The average of process utilization is obtained by dividing the sum of all utilizations by number of processors [2].

$$U(p_i) = {Load\ (p_i)}/{maxspan} \quad \ldots\ldots\ldots (4)$$

$$AvgU = {(\sum_{i=1}^{No.of\ processors} U(p_i))}/{No.of\ processors} \quad \ldots\ldots (5)$$

2.2. Genetic algorithm

Genetic algorithm is guided random search algorithm based on the principles of evolution and natural genetics. It combines the exploitation of the past results with the exploration of new areas of the search space. 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 [3]:

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.

Genetic algorithm maintains a population of candidate solutions that evolves over time and ultimately converges. Individuals in the population are represented with chromosomes. Each individual is numeric fitness value that measures how well this solution solves the problem. Genetic algorithm contains three operators. The selection operator selects the fittest individuals of the current population to serve as parents of the next generation. The crossover operation chooses randomly a pair of individuals and exchanges some part of the information. The mutation operator takes an individual randomly and alters it. As natural genetics, the probability of crossover is usually high, the population evolves iteratively (in the genetic algorithm terminology, through generation) in order to improve the fitness of its individuals The structure of genetic algorithm is a loop composed of a selection, followed by a sequence of crossovers and mutations. Probabilities of crossover and mutation are constants and fixed in the beginning. Finally, genetic algorithm is executed until some termination condition achieved, such as the number of iterations, execution time, execution time, result stability, etc.

3. FRAMEWORK FOR GENETIC ALGORITHM

3.1 String representation

The genetic representation of individuals is called genotype. The main criteria in selecting the string representation for the search node is that the new string generated from the application of genetic operator must represents legal search node for the problem. Every process is present and appears only once in the schedule. The string representation used in this paper is an array of n × m digits, where n is the number of processes and m shows the processor that the process is assigned to. Process index shows the order of execution on that processor.

3.2 Initial population

A genetic algorithm starts with a set of individuals called initial population. Most GA-Based algorithms generate initial population randomly. Here, each solution *i* is generated as follows: one of the unscheduled processes is randomly selected, and then assigned to one of the processors. The important point is the processors are selected circularly, it means that they are selected respectively form first to last and then come back to first. This operation is repeated until all of processes have been assigned. An initial population with size of *POPSIZE* is generated by repeating this method.

3.3 Fitness function

The fitness function is essentially the objective function for the problem. It provides a mean to evaluate the search nodes and also controls the reproduction process. For the process scheduling in distributed system problem, we can consider factor such as load-balancing, processors utilization etc. We take into account this objective in following equation. Fitness function of schedule T is

$$f(T) = \frac{Average\ utilization}{Scheduling\ length}$$

This equation shows that a fitter solution has less scheduling length and higher processor utilization.

3.4 Reproduction

The reproduction process is typically based on the fitness values of the strings. The principal is that string with higher fitness value will have higher chance of surviving to the next generation. We calculate fitness of the entire individual in population, picking the best individual of them and then form a group. We can make a slight modification to basic reproduction operation by always passing the best string in the current generation to the next generation. This modification will increase the performance of genetic algorithm.

3.5 Crossover

Crossover is generally used to exchange portions between strings. The crossover operation picks top two strings from best group. Random task Ti from one of these two strings picks and put on the new string with care of precedence relation. If Ti task at same position then Ti is coping on the same place for new string. If we randomly choose two same parents (A=B) and if one of parents e.g. A the best string we use operator mutation on the second B string. Otherwise we mutate the first set and child generate randomly.

3.6 Mutation

Crossover operation of genetic algorithms (GAs) cannot generate quite different offspring from their parents because the acquired information is used to crossover the chromosomes. An alternate operator, mutation, can search new areas in contrast to the crossover.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.

3.7 Termination condition

We can apply multiple choices for termination condition: maximum number of generation, equal fitness for fittest selected chromosomes in respective iterations.

```
Genetic Algorithm
{
Randomly create an initial population
Assign a fitness value to each individual
Form Group of best individual
WHILE NOT termination criteria DO
{
Assign a priority value to the individual in group
Choose two best individual from the group;
Crossover surviving individuals;
Mutation child;
Recorded best individual in group and eliminate worst one
in group.
}
}
```

Genetic Algorithm Steps

4 . Conclusion

The paper makes an analysis of existing genetic algorithm and their parameters.Scheduling in distributed operating systems has a significant role in overall system performance and throughput. This algorithm considers multi objectives in its solution evaluation and solves the scheduling problem in a way that simultaneously minimizes scheduling length and communication cost, and maximizes average processor utilization and load-balance. Most of existing approaches tend to focus on one of the objectives. Experimental results prove that our proposed algorithm tend to focus on more objective simultaneously and optimize them.

References

[1] Dr Apurva Shah And Vinay Hansora "A Modified Genetic Algorithm For Process Scheduling In Distributed System" IJCA Special Issue On "Artificial Intelligence Techniques - Novel Approaches & Practical Applications"AIT, 2011

[2] M. Nikravan And M.H.Kashani " A Genetic Algorithm For Process Scheduling In Distributed Operating Systems Considering Load Balancing", European Conference On Modelling And Simulation.

[3] C.C.Shen, & W.H.Tsai, "A Graph Matching Approach To Optimal Task Assignment In Distributed Computing Using A Minimax Criterion", IEEE Trans. On Computers, 34(3), 1985, 197-203.

[4] G.L.Park, "Performance Evaluation Of A List Scheduling Algorithm In Distributed Memory Multiprocessor Systems", International Journal Of Future Generation Computer Systems 20, 2004, 249-256

[5] L.M.Schmitt, "Fundamental Study Theory Of Genetic Algorithms" , International Journal Of Modelling And Simulation Theoretical Computer Science 259, 2001, 1 – 61.

[6] W.Yao, J.Yao, & B.Li, "Main Sequences Genetic Scheduling For Multiprocessor Systems Using Task Duplication", International Journal Of Microprocessors And Microsystems, 28, 2004, 85-94.

[7] A.T. Haghighat, K. Faez, M. Dehghan, A. Mowlaei, & Y. Ghahremani, "Multicast Routing With Multiple Constraints In High-Speed networks based on genetic algorithms" , *In ICCC 2002 Conf.*, India, 2002, 243–249.