International Journal of Mechatronics, Electrical and Computer Technology

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

A Genetic Algorithm-Based Approach for Process Scheduling In Distributed Operating Systems

Mohammad Nikravan

Department of Computer Engineering, Islamic Azad University, Shahr-e-Qods Branch, Tehran, Iran

*Corresponding author's E-mail: *Nikravan@Qodsiau.ac.ir*

Abstract

A Distributed Computing System comprising networked heterogeneous processors requires efficient process allocation algorithms to achieve minimum turnaround time and highest possible throughput. To efficiently execute processes on a distributed system, processes must be correctly assigned to processors and determine the execution order of processes so that the overall execution time is minimized. Even when target processors are fully connected and the communication among processors is fast and no dependencies exist among processes the scheduling problem is NP-complete. Complexity of scheduling problem dependent of number of processors, process execution time and the processor network topology. As distributed systems exist in kinds of homogeneous and heterogeneous, in heterogeneous systems the difference between processors leads to different execution time for an individual process on different processors and makes scheduling problem more complex. Our proposed genetic algorithm is applicable for both homogeneous and heterogeneous kinds.

Keywords: Distributed systems, homogeneous, heterogeneous, scheduling and genetic algorithm.

1. Introduction

Scheduling plays a significant role in overall efficiency in a Distributed Computing System (DCS) .In such systems each process has its own characteristics like execution time. Scheduling in distributed systems has known as a NP-complete problem .So solving

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

this problem using usual algorithms seems difficult. GAs have been applied to the optimization of many kind of problems , such as traveling salesman problem , job-shop scheduling and flow-shop scheduling , machine learning , VLSI technology , genetic synthesis and etc[11].

A genetic algorithm comprises these parts: encoding problem, genetic operators such as Crossover and Mutation, Fitness function and selection mechanisms .The proposed algorithm in this paper is applicable for both homogeneous and heterogeneous distributed systems. The reminder of this paper is organized as follows: In the next section the general model of a distributed system is defined. In the section 3 first we have an overview on GA and then we design a GA for the process scheduling problem .In section 4 experiment results are shown.

2. System Model and Definitions

In order to schedule the processes in a distributed system, we should know the information about the input processes and distributed system itself such: Network topology, processors speed, communication channels speed and so on .Since obtaining such information is beyond the scope of this paper, so we suppose that the required information is available and a deterministic model is assumed. In a distributed system there are *m* processors, *m* > 1. The followings are defined for the model.

P = $\{p_1, p_2, p_3, ..., p_m\}$ is the set of processors in the distributed system. R is a $m \times m$ matrix, where the element $r_{u,u}$ $1 \le u, u \le m$ of R, is the communication delay rate between p_u and p_v. A is a $n \times m$ matrix, where the element a_{ij} $1 \le i \le n$, $1 \le j \le m$ of A, is the execution time of process *i* on processor p_j. L is an $n \times m$ matrix, where the element l_{ij} $1 \le i \le n$, $1 \le j \le m$ of L, is the latency time for process *i* to start execution on processor p_j . D is linear matrix,

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

where the element $d_i \quad 1 \le i \le n$ of D, is the data volume for process *i* to be transmitted, when process *i* is to be executed on a remote processor.

S is an $m \times m$ matrix, where the element $S_{u,v} \quad 1 \le u,v \le m$ of S, is the time required to transmit a unit of data from p_u to p_v. A chromosome consists of *n* digits, where *n* is the number of processes. A digit can take any one of the *1..m* values , which shows the processor that the process is assigned to . The *fitness* function defined as inverse of Total Run Time (TRT) $fitness = \frac{1}{Total\ \ Run\ \ Time} = \frac{1}{TRT}$.

The total run time for *n* processes is obtained by equation:

$TRT = \max(r_{c[i]f} + S_{c[i]f} * d_i + l_{if} + a_{if})$, $1 \le i \le n$ that $c[i]$ is the processor that the process i is present on it just now and f is the target processor that is selected for process i to be executed on .

C is linear matrix, where the element $c_i \quad 1 \le i \le n$ of C, is the processor that the process i is present on it just now. In homogeneous distributed systems the execution time of an individual process *i* on all processors is equal each other, that means:

$1 \le i \le n \quad a_{i1} = a_{i2} = ... = a_{im}$

A data structure STATUS associated with every processor, which has two fields, showing the maximum no. of the processes that can be allocated to this processor and the memory capacity of that machine.

3. Genetic Algorithms

A genetic algorithm initiate biological evolutionary theories to solve optimization problems. A GA includes a set of individual elements (the population) and a set of biological imitated operators defined over the population itself. According to the evolutionary theories, only the most fitted elements in a population are probable to survive and generate offspring, thus transmitting their biological heredity to new

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

generations [14]. In computing terms, a genetic algorithm map a problem onto a set of (typically binary) strings i.e. 0s and 1s. When a problem is to be solved by a genetic algorithm, solution must be encoded to a chromosome .A chromosome comprises a set of values called gens, which each gen shows one characteristic of solution. Each chromosome is an individual of population, so to solve any problem with GA, the problem should easily map to a population of binary strings.

Each solution is associated with a fitness value that reflects how good it is. The survival of the individual depends on its fitness value. The higher the fitness value, the more is the chance to survive. Genetic operators such as crossover and mutation are used to create the subsequent generation from the strings of the current population [12].

The fitness function in a genetic algorithm is the objective function that is to be optimized. The simplest form of GA involves three types of operators: selection, crossover and mutation [15]. 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 is achieved, such as the number of iterations, execution time, results stability, etc.

Selection: The operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more time it is likely to be selected to reproduce.

Crossover: It is generally used to exchange portions between strings. The operator randomly chooses a locus and exchanges the subsequences before and after that locus between strings. Crossover is not always effected .The invocation of the crossover depends on the probability of the crossover.

Mutation: It is used to change the gens in a chromosome. Mutation replaces value of a gen with a new value from defined domain for that gen . Mutation can occur in each digit position in a string with some very small probability.

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

3.1. A Genetic Algorithm for distribute processes on processors

The proposed algorithm is configured as follows:

1) Randomly Generate a population of few chromosomes (about 10 chromosomes); verify STATUS for all processors, calculate value of fitness for generated chromosomes and take the one with highest value , save the highest calculated fitness .
from now on , the highest fitness calculated in (#1) considered as threshold limit and any chromosome below the threshold will be rejected and not included in population.

2) Generate the initial population of chromosomes; All chromosomes that generated in this step must have fitness value greater than the threshold limit calculated in (#1). In this step size of population is between 50...100 chromosomes.

3) SELECT: probability of selection of parent is linearly dependent on the fitness value.

- i.e. $ax + b$, that x is fitness value of chromosome x ,and a,b are arbitrary values.

4) Perform Crossover with probability of crossover (Pc) , and Mutation with probability of mutation (Pm) ; that these operation are performed on randomly chosen positions of selected chromosomes .

5) If number chromosomes is less than 1000 go to (#3).

6) Pick up a subpopulation of chromosomes randomly, using the probability of selection as in SELECT. The size of this population can be between 10 and 20.

7) The chromosome with highest fines value represents the allocation and is our answer.

3.1.1. Description of SELECT: To make SELECT operator as efficient as possible ,the probability of selection must be as accurate as possible .One way is to make p_i copy from chromosome i and include in population , which p is proportional to the probability

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

of selection of ith chromosome. Thus, the total number of chromosomes (copies included) will be:

$$T = \sum_{i=1}^{n} p_i \qquad (1)$$

As we see this method requires much memory to store chromosomes.
Another way, to save memory, is to attach a field with each new chromosome generated. We store in this extra field an integer value proportional to fitness of chromosome. So when value *x* has been stored in this field, it means that chromosome represents `X' copies of the chromosome.

When we want to randomly select a chromosome, we generate a random number from *1* to (X1 +X2 + … +Xn), where *Xi* is the number in the field associated with the ith chromosome. Let us say, the number generated is Y and X1 + X2+… + Xk < Y< X1 + X2 + … + Xk+1.
Thus the chromosome selected is Xk+1th chromosome.

4. Experiment Results

The GA presented in this paper was implemented and tested in a DHOS as well as in a DHES. The experiments were performed on a Sun-space workstation. The control parameters used in our experiments are as follows:

- crossover probability : 0.9
- mutation probability : 0.3
- initial population size : 75
- number of processes : 28
- number of processors : 5

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

Results

(1) On running the above example, we obtained the following results:

The selected chromosome is 3544555514422442513151111144

/* i.e. process 1 is allocated to processor 3, process 2 to processor 5, process 3 to processor 4 and so on */

Time required by the program is 3 seconds.

(2) In another example, considering 6(six) processors and 39 processes
And their corresponding matrices, we obtained the following results:
Selected chromosomes is

5462256553343233363145266112111153666624

Time required by the program is 3 seconds.

(3) In the third example, considering 8(eight) processors and 50 processes
and corresponding matrices, we obtained the following results:

Selected chromosomes is 67245582315555458321671771564 4

Time required by the program is 4 seconds.

Conclusion

To schedule processes in a distributed system, first processes must be assigned to processors; after that any common non distributed scheduling algorithm such as *Round Robin, Sjf, HRRN,...* can be applied on each individual processor. The problem discussed in this paper is how to distribute processes on processors. As distributing processes on processors in a distributed system is a NP-complete problem, using usual methods in not suitable .So to solve this problem a genetic algorithm is proposed. The execution and communication costs have been taken into account, which play a vital role for a process allocator in a distributed operating system.

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

References

[1] *Y. Lee, C. Chen,* A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems, Department of Computer Science and Information engineering National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

[2] M. GOLUB ,Scheduling Multiprocessor Task with Genetic Algorithm,Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing, University of Zagreb.

SUAD KASAPOVIC, Department of Telecommunications, faculty of Electrical Engineering and Computing University of Zagreb.

[3] J. Nowacki, MultiProcessor Scheduling with support by Genetic Algorithms-based Learning Classifier Systems, Franciszek Sered, Polish -Japanese Institute of Information Technologies , Polish Telecom ,Institute of Computer Science, Polish academy of Sciences

[4] M. Grajcar, Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System, University of Passau Chair of Computer Architectures.

[5] J. Koronacki , and C. Janikow, Distributed Scheduling with Decomposed Optimization Criterion: Genetic Programming Approach, Franciszek, Department of Mathematics and Computer Science University of Missouri USA ,Institute of Computer Science, Polish Academy of Sciences.

[6] S. Thilakawardana and R. Tafazolli, Use of Genetic Algorithms in Efficient Scheduling for Multi Service Classes, Centre for Communications Systems Research (CCSR), University of Surrey, uildford, GU27XH, UK.

[7] A. KUMAR TRIPATHI, B. KUMER SARKER and N. KUMAR, A Genetic Algorithm- Based Multiple Task Allocation Considering Load, Department Of Computer Engineering, I.T. Banaras Hindu University, DEO PRAKASH VIDYARTHI , Department of Computer Science, Faculty of Science, Banaras Hindu University.

[8] A. Kumar, B. Sarker and N. Kumar, Global Scheduling in Distributed Real-Times Computer Systems, Department of machine Design, Royal Inst. Of technology stockholm.

[9] D. Montana, Introduction to the Special Issue Evolutionary Algorithms for Scheduling, BBN Technologies (GTE Internetworking), Cambridge.

[10] K. Yue, D. Lilja, Designing Multiprocessor Scheduling Algorithms Using a Distributed Genetic Algorithm System Department of Electrical Engineering.

[11] S. Woo, S. Yang , S. Kim , and T. Han, Task Scheduling In Distributed Computing Systems with a Genetic Algorithm, Parallel Processing Laboratory , Dept. Of Computer Science, Yonsei University , Seoul.

[12] M. Sriniwas and L. M. Patnaik, \Genetic Algorithm: A survey," IEEE Computer (June 1994), pp. 44.

[13] J. L. R. Filho, P. C. Treleaven, and C. Alippi, \Genetic Algorithm Programming Environments", IEEE Computer (June 1994), pp. 29{42.

Vol. 2(2), Jan, 2012, pp 23-31, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

[14] A. K. Tripathi, B. K. Sarker, N. Kumar and D. P. Vidyarthi, \Multiple Task Alloca-tion with Load Considerations", International Journal of Information and Com- puting Science 3 (June 2000) 36{44.

[15] A. K. Tripathi, D. P. Vidyarthi, and A. N. Mantri, \A Genetic Task Allocation Algo-rithm for Distributed Computing System Incorporating Problem Speci_c Knowl-edge", International Journal of High Speed Computing 8 (1996) 363{370.