International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-11, September 2019

Dynamic Load Balancing Based on Genetic Algorithm

S. Sandhya, N.K. Cauvery

Abstract: ***Load balancing has been the focus of research over the current days in many domains but more importantly they are crucial for distributed computing. The research mainly focuses towards distributing load based on the current usage of nodes to facilitate effective resource utilization and obtain better performance from the system. Balancing load is to distribute the tasks on to the available or idle nodes so that resources are utilized fairly in a distributed environment. By developing strategies to assign the processes on a heavily loaded processor to an idle/under loaded processor in a way that balances out the load, the total processing time can be reduced hence achieving improved processor utilization. Genetic Algorithm(GA) is a search based approach that is robust and that can adapt to the search space for optimizing the solution are gaining immense popularity. GA in the proposed work considers the load as a parameter to evaluate fitness of the strings. The strings are also generated based on the load information of the nodes. The fitness evaluates the strings to identify only the underutilized or idle nodes which can take the transmitted load. Hence the work proposed explores and illustrates how GA could be employed to solve the problem of dynamic load-balancing.***

Keywords: Distributed Computing, Dyanmic Load Balancing, Genetic Algorithm, Resource utilization.

I. INTRODUCTION

A load balancing algorithm attempts to balance overall system load by transmitting the load from heavily loaded nodes to lightly loaded nodes in an attempt to ensure overall good performance relative to some specific metrics of the system. 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 a search algorithm based on natural selection and natural genetics. Using the power of genetic algorithm process scheduling can be done more efficiently to balance load than other conventional methods.

II. STATE OF ART DEVELOPMENT

A huge amount of research has been done in the field of load balancing and improving the performance of the all nodes in the distributed system. The application of the genetic algorithm in load balancing has improved the performance of the distributed system when compared to the other conventional algorithm. Load balancing on multi computers is a challenge due to the autonomy of the processors and the inter-processor communication overheads incurred in the collection of state information, communication delays, redistribution of load etc. Load balancing works mainly by distributing the tasks running on a node to others which are idle. If there are multiple nodes in a distributed system, deciding the destination node which can readily accept the offloaded load adds overhead to the execution time. This overhead includes time spent to send request message to all the nodes and the response from them to the requesting processor. The requesting processor should then pick one of the underloaded node as destination node to offload its task. Load balancing usually improves system performance by offloading the tasks to the lightly loaded nodes. There are wide variety of algorithms employed to balance load, they are classified as centralized Vs decentralized, dynamic Vs static, periodic Vs non-periodic, and those with thresholds Vs without thresholds. The chosen approach here assigns the process as a whole to the identified destination node decided/picked as an outcome of GA. The idea is to determine optimal solution to the problem of load balancing.

III. PROPOSED APPROACH

The general outline of genetic algorithm execution could have thought of comprising the following five phases as listed below:

Initialization Phase: The initial population is created by randomly selecting the processes/requests from the request queue, in FCFS order and then randomly assigned to nodes (servers). The initial population is obtained by swapping the orders of assignment of processes, for a fixed number of times.

Evaluation Phase: The fitness of each schedule evaluated by executing the fitness function defined.

Selection Phase: Selection phase provides to get more copies of the solutions with higher fitness value and hence survival-of–the-fittest mechanism can be implemented on the candidate solution.

Genetic Operation Phase: This phase provides to create new population with better off springs. It has three stages (a) Mutation (b) Reproduction and (c) Crossover

Termination Phase: Stopping conditions are defined as when the programme encounters these situations, it will get terminated.

The termination condition will be the maximum number of generations reached.

After the string is generated as an outcome of the genetic algorithm the victim nodes are selected and the load is migrated.

Manuscript published on 30 September 2019.
***Correspondence Author(s)**
Sandhya S*, Computer Science and Engineering, R V College of Engineering, Bangalore, India.
Dr. N. K. Cauvery, Information Science and Engineering, R V College of Engineering, Bangalore, India.

IV. METHODOLOGY

The proposed work starts with an initially generated population either seeded or by a random approach. The chosen population strings which are referred as parent strings undergo series of genetic operation to produce the new off-springs. These are evaluated for fitness and only the fitter strings move on to the next iteration which could be the candidate solution for the identified problem. The process repeats till termination condition is met. The overview of the flow of GA is shown in Fig 1.

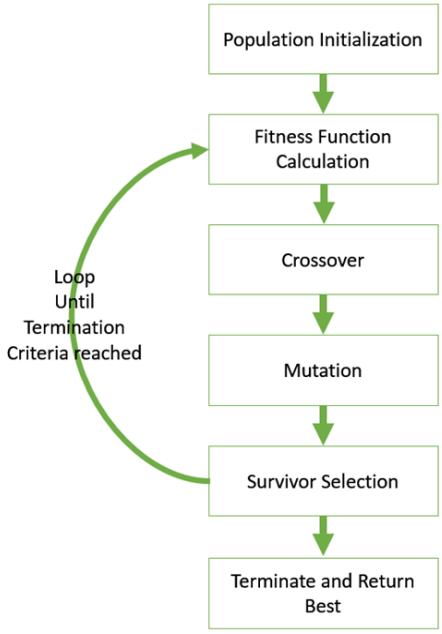


Fig 1 Flow diagram of a genetic algorithm evaluation

Proposed Genetic Algorithm:

Step 1: Initialization– Initial population is generated such that the population-level center of mass is high.

Step 2: Check_load– The load estimates of the nodes are computed and over loaded nodes are identified.

Step 3: Perform the following steps if load state is Overloaded.

Step 4: Genetic-operation

Step 4.1: Selection – The fitness value of the strings in the population is computed.

Step 4.2: Crossover – One point crossover operation is done on the chromosomes.

Step 4.3: Mutation – Swap mutation is used to add mutation to the population.

(Genetic operation is done for multiple generations to find a maximal fitness chromosome and requests messages are sent to the receivers in the maximal fitness chromosome)

Step 5: Message_evaluation – Is used by each processor to accept or reject a request from another processor.

Step 6: Repeat Step 2 to Step 5 if tasks in the distributed system need to be offloaded.

A. *Encoding Method*

In GAs, the chromosome is used to represent a solution to the problem and the encoding method provides the representation of the genes in the chromosome. Some of the widely known encoding methods are character encoding, binary encoding and real-valued encoding.

The proposed method takes the load estimate for a node that is formulated as the weighted sum of the CPU, memory and network loads of the node.

$$L_n = (W_{cpu} \times CPU_{Load\_Avg}) + (W_{mem} \times MEM_{load}) + (W_{NET} \times NET_{load})$$

The weights are used to vary the importance of the different types of loads, but they are normalized equally. Binary encoding is used to represent the set of processors to which the request messages from the sender are sent to the receiver processor.

E.g. Consider set of 10 receiver processors. The chromosome is represented as:

0	1	1	0	0	1	1	1	0	0

Here bit 1 at position *i* represents processor P_i is selected for sending requests and bit 0 at position *i* represents processor P_i is discarded.

B. *Initial Population*

The first phase in the functioning of a genetic algorithm is to find an appropriate initial population. Traditionally pseudo random numbers and quasi random numbers are used to generate initial population. Pseudo random numbers produce random numbers which are genuine while quasi random numbers are used when we need to distribute the numbers evenly without any gaps and cluster formation over the entire region.

Some of the factors which need to be taken care while generating initial population are the fitness function. In the work, the search space, difficulty of the problem, the diversity in the population, the size of the population and the selection pressure are used.

C. *Fitness Function*

The fitness function is used to determine how fit an individual is to progress to the next generation. Using fitness function formula, a fitness score is given to each individual. Based on the fitness score the probability of selecting that individual is decided. In general, more the fitness value, better is the fitness of the chromosome.

The main concern of the work is to offload the task such that all the processors tend to move towards the ideal load i.e., all processors are loaded equally. This can be done by minimizing the deviation of all the loads from the average load. Hence, in the approach used to formulate the fitness function the deviation from mean between the nodes are considered.

Consider the k^{th} processor P_k which is in overloaded state, whose load is L_k. The difference of the load and the load average is calculated as Δ:

$$\Delta = L_k - L_{avg}$$

This signifies the amount of load to be offloaded from P_k as part of the load balancing. This is done so that after offload, value reaches the average load, which is the desired ideal condition. The objective function is formulated as follows:

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$OF = \frac{1}{m}\sum_{i=1}^{m}\{[(L_i + \Delta) - L_{avg}]^2 + \propto * C(k,i)\}$$

- Here, $C(k,i)$ is the cost of communication from sender P_k to receiver P_i.
- $\propto$ is the smoothening factor, used to vary the importance of the communication cost.
- m is the number of processors chosen in the chromosome (bits set to 1)

This objective function has to be minimized. Hence, fitness function is taken as the inverse of the objective function.

$$f = \frac{1}{OF}$$

For example:

Consider a set of 8 receiver processors P with following loads:

2	4	5	4	3	10	12	8

Let 4 and 8 be the limits for Underloaded and Overloaded states respectively.

So, processors P6, P7, P8 are overloaded.

L_{avg} = (2 + 4 + 5 + 4 + 3 + 10 + 12 + 8) / 8

L_{avg} = 6

Let P_6 be the sender node in which GA is executed.

$\Delta = L_6 - L_{avg} = 10 - 6 = 4$

Consider 3 sample chromosomes on which the fitness values are computed:

1) **Chromosome 1**: Here processors P_1 and P_2 are selected.

1	1	0	0	0	0	0	0

m = 2, loads = 2, 4

$OF = \frac{1}{2}\{[6 - (2+4)]^2 + [(6 - (4+4)]^2\} = 2$

f = ½ = 0.5

2) **Chromosome 2**: Here processors P_3 and P_8 are selected.

0	0	1	0	0	0	0	1

m = 2, loads = 5, 8

$OF = \frac{1}{2}\{[6 - (5+4)]^2 + [(6 - (8+4)]^2\} = 22.5$

f = 1/22.5 = 0.0444

3) **Chromosome 3:** Here processors P_1 and P_5 are selected.

1	0	0	0	1	0	0	0

m = 2, loads = 2, 3

$OF = \frac{1}{2}\{[6 - (2+4)]^2 + [(6 - (3+4)]^2\} = 0.5$

f = 1/0.5 = 2

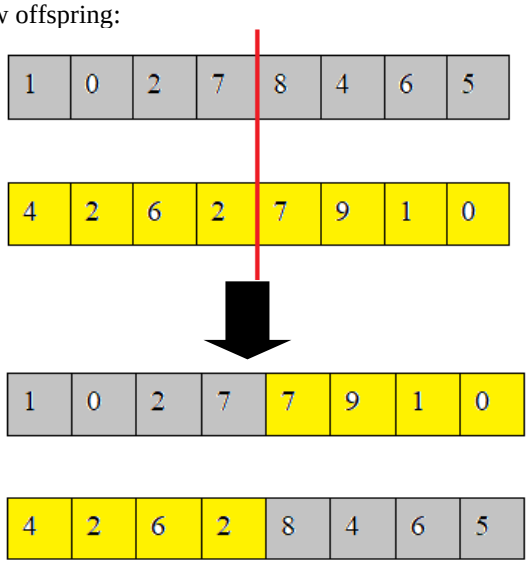
As it is obvious the most efficient chromosome, **Chromosome 3** has the highest fitness value and the least efficient chromosome **Chromosome 2** has the least fitness value.

D. *Crossover*

Crossover is one of the important phase in genetic algorithm, where genetic information of two parents are combined to generate a new offspring. This is also called recombination as the offspring is created by combining the genes of the parents. One Point Crossover is employed for better offspring production.

One point Crossover

A random crossover point is picked from the parent chromosome and genes to the right of the selected crossover points are swapped between the parent chromosomes to get new offspring:

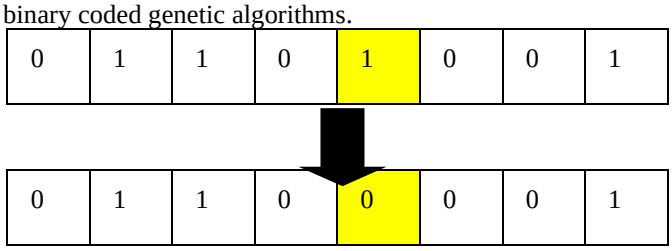


E. *Mutation*

Mutation happens to maintain genetic diversity of the population and to prevent the premature convergence. Mutation alters one or more bits in a chromosome from its initial state. Mutation probability is defined according to the user. High probability leads to primitive random search, hence mutation probability should to be set to low.

Bit Flip Mutation

One or more bits are selected randomly and are flipped to get a new offspring in bit flip mutation. This is used in binary coded genetic algorithms.



V. RESEARCH CHALLENGES

Challenges faced during the research of GA based load balancing models:

- Formulating the load estimate for a node in the system which gives the best efficiency for the model. A node has multiple parameters which would contribute towards the overall load. Deciding on the particular parameters along with smoothening factors for the load estimate is a challenge.
- Setting load thresholds which are optimal for load balance with process migration.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Initial generation of chromosome population for faster fitness value convergence.
- Formulating the fitness function for the GA model which best evaluates the fitness of the chromosome with respect to the load balance problem.

VI. CONCLUSION

The work implements GA for load balancing in distributed systems. The algorithms and parameters used and results obtained are as described above. The proposed new fitness function for GA model, based on the analysis of the state of art is observed to reduce the overhead to a greater extent since the fitter strings contain the most suitable nodes for load offloading. The proposed approach considers the load deviation from the mean load and the communication cost involved in the exchanging messages as parameters for the fitness function. The proposed fitness function minimizes the overheads, maximize the performance and provides better response times. The model has been validated using sample chromosomes as illustrated in the discussion above.

REFERENCES

1. Seong-hoon Lee, Chong-sun Hwang, *A Dynamic Load Balancing Approach using Genetic Algorithm in Distributed Systems*, IEEE International Conference on Evolutionary Computation, 1998, DOI:10.1109/ICEC.1998.700103, pp 639 – 644.
2. Priyanka Gonnade, SonaliBodkhe, *An Efficient load balancing using Genetic algorithm in Hierarchical structured distributed system*, International Journal of Advanced Computer Research, Volume-2, Number-4 Issue-6, ISSN (print): 2249-7277 ISSN (online): 2277-7970, December-2012, pp 69-73.
3. Nashat Mansour, Geoffrey C. Fox, *An Evolutionary Approach to Load Balancing Parallel Computations*, Syracuse University, Electrical Engineering and Computer Science Technical Reports, April 1991, pp 1-11.
4. Argha Sen, *Improving Performance of Clusters using Load Balancing Algorithms*, National Institute of Technology Rourkela,2011.
5. Pedro A. Diaz-Gomez, Dean F. Hougen, *Initial Population for Genetic Algorithms: A Metric Approach*, International Conference on Genetic and Evolutionary Methods, June 2007.
6. Albert Y. Zomaya, Yee-HweiTeh, *"Observations on Using Genetic Algorithms for Dynamic Load-Balancing"*, IEEE Transactions on Parallel and Distributed Systems, Volume 12, No. 9, September 2001, pp 899 – 911.
7. M. Hakim, J. Jais, S. Salwa, *MOSIX: Implementation, trend and benchmark in Malaysia*, IEEE International Symposium on Information Technology , vol. 3, August 2008,pp. 1-6.

AUTHORS PROFILE

Sandhya . S, BE, M.Tech, (Ph.D) working as Assistant Professor in the Department of Computer Science an Engineering at R. V. College of Engineering. Her areas of interest for research include Networking, Distributed Systems, Genetic Algorithm and Virtualization.

Dr. N. K. Cauvery, . BE, M.Tech, Ph.D working as Professor in the Department of Information Science an Engineering at R. V. College of Engineering. Her areas of specialization include Computer Networks, Compiler Design, Genetic Algorithms.