International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 8 | ISSN : 2456-3307

A Centralized Load balancing policy for Distributed System using Genetic Algorithm

Priyanka Gonnade
Department of CSE, G. H. Raisoni Academy of Engineering and Technology, Nagpur, Maharashtra, India

ABSTRACT

In this paper, a genetic algorithm based approach for scheduling of task in distributed system considering dynamic load balancing is used. The underlying distributed system has hierarchical structure and load balancing is done in two levels: group level and node level. In general, in a distributed system the allocator and the scheduler schedule the task. The allocator determine job's execution on machine and the scheduler determines when to execute the job by using available resources. Using Genetic algorithm to get the optimal solution for scheduling of task this centralized load balancing policy prevents the node connected in the system from being overloaded or become idle ever.

Keywords: Distributed System, Genetic Algorithm, Load Balancing, Scheduling, Processes

I. INTRODUCTION

Even a small group of fastest computers available cannot meet the computational need in areas like cosmology, molecular biology, nonmaterial, etc. However, with the availability of high-speed networks, a large number of geographically distributed computers can be interconnected and effectively utilized in order to achieve a performance, which is not ordinarily attainable on a single computer. The distributed nature of this type of computing environment calls for consideration of heterogeneities in computational and communication resources. A common architecture is the cluster of independent computers communicating through a shared network. Incoming workload has to be efficiently allocated to these computers so that no single computer is overburdened where one or more other computers remain idle. Further, tasks migration from high to low traffic area in a network alleviates the network-traffic congestion problem up to some extent.

Some of these existing approaches consider constant performance of the network while others consider deterministic communication and transfer delay. The load balancing schemes designed under this conviction undermines the randomness in delay. To adequately model load balancing problems, several features of the parallel computation environment is captured: These include-

1. The number of processes in queue;
2. The relative performances of the individual unit;
3. The processor execution time;
4. The maximum utilization of processor;
5. The speed of the processor for the workload;
6. The capability of the processor for load sharing;
7. The transfer of load from one to another.

In this paper, a centralized dynamic approach for load balancing is used. M heterogeneous nodes which are connected with high-speed network are arranged in a hierarchical structure with two level hierarchies. Each level consists of the designated node. The second level is again divided into independent clusters and each cluster consists of the designated node. The second level node communicates with the global designated nodes and collects all the information about the other nodes.

Load distribution is accomplished by using a genetic algorithm which selects an optimal node from among the nodes and load balancing is achieved by proper load distribution of the load. A genetic algorithm always prevents the nodes from getting overloaded or idle. The proposed load balancing algorithm has following objectives: Preventing nodes from getting overloaded or under loaded or idle; Improvement in the job execution time and system performance; No task migration to reduce message passing; To make system stable so that system performance does not decline; and to reduce communication overhead.

II. OBJECTIVE

The main goal of the project is to perform a load balancing over all the nodes connected in the network which will upgrade the performance. The purpose is oriented to come up with a better decision making policy to tackle the randomness in load. This work does not address issues like divisibility of a job, network architecture, operating system, memory size, fault-tolerance and fault-recovery. In the existing literature, the balancing policies have been developed where various above aspects is considered. In this project, we are trying to utilize a node as much as possible. The utilization of node will increase when we have a load balancing policy which will identify which the suitable node for a particular task.

In this project, a centralized dynamic approach for load balancing is used. M heterogeneous nodes which are connected with high-speed network are arranged in a hierarchical structure with two level hierarchies. Each level consists of the designated node. The second level is again divided into independent clusters and each cluster consists of the designated node. The second level node communicates with the global designated nodes and collects all the information about the other nodes. Load distribution is accomplished by using a genetic algorithm which selects an optimal node from among the nodes and load balancing is achieved by proper load distribution of the load. A genetic algorithm always prevents the nodes from getting overloaded or idle.

The proposed load balancing algorithm has following objectives:

1. Preventing nodes from getting overloaded or under loaded or idle;
2. Improvement in the job execution time and system performance;
3. No task migration to reduce message passing;
4. To make system stable so that system performance does not decline;
5. To reduce communication overhead.

III.DESIGN AND IMPLEMENTATION

In the proposed work, the load is the main factor to be consider for load distribution and the objective function is designed for the nodes accordingly.

A. Objective function

The objective function is derive based on 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. However, often it may not be possible. Therefore, load deviation is minimized fully. 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.

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

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.

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

Mean load of a node is obtained as-

$$L_{mean} = \frac{L_{total}}{m} \tag{3}$$

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

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

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.

B. The Model

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 do 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. In this proposed model Genetic Algorithm is used to minimize the load deviation that occurs in the node to get a balanced nodes. Genetic Algorithm has modules which is problem specific. The details are given below:

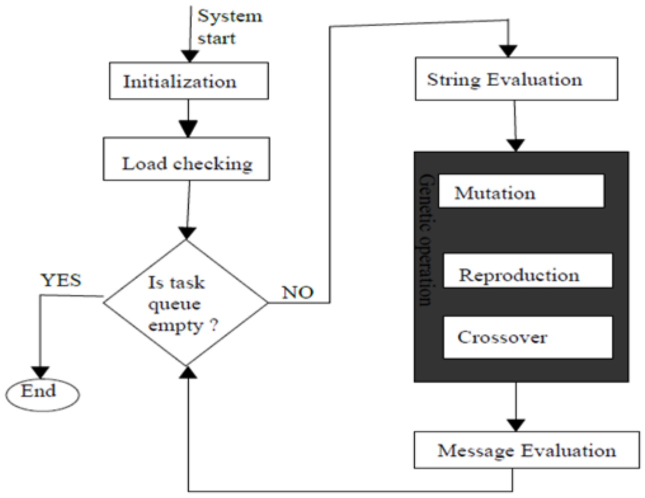
i. ***Chromosome Structure***-In GA, chromosome is the solution of the problem which varies from system to system. The chromosome is made up of genes. The gene structure in this model is the load assigned to the nodes.

ii. ***Initial Population***-Number of chromosome in the genes is the population for GA. In this model, the population is generated randomly. The random population is generated by using random function.

iii. ***Crossover & Mutation***-The children for next generation can be generated by crossover operation. The proposed model uses. 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. 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.

iv. ***Selection***-The selection process used here is based on spinning the roulette wheel, in which each chromosome will get a space according to its fitness value. Each time we require an offspring, a simple spin of the weighted roulette wheel gives a parent chromosome.

C. The Algorithm



The proposed model distributes the tasks on the nodes in such a way that the load deviation is minimum. The load distribution should be the best one to have minimum load deviation. In this section, scheduling algorithm which utilizes GA for load balancing in HDCS is given. The inputs to the algorithm are number of nodes, number of tasks, number of generation and the population size.

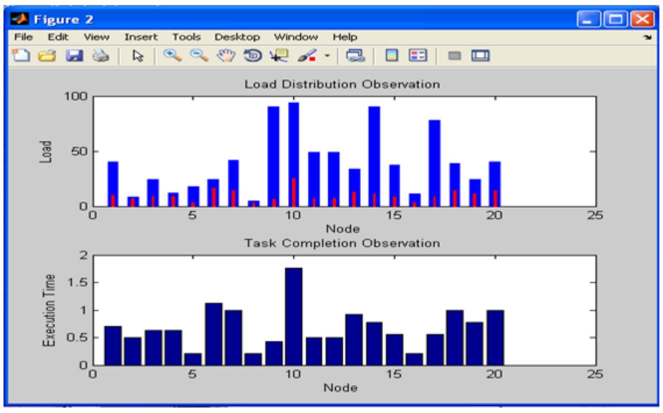


Figure 7. Flow graph for load balancing algorithm

The algorithm consist of various modules- Initialization, Load-checking, String-evaluation, Genetic-operation and Message-evaluation. This algorithm is executed only on the designated node that is GLB and DR .

i. ***Initialization module-*** It will collect all the information of the nodes. This module iteratively gets executed to get the updated information of the nodes. The main feature of this module is to allocate the tasks to the nodes. The allocation is done by using capacity index. The capacity index is calculated by using the formula-

cap_index = cap_node / total_capacity;

Then, the number of tasks to be allocated is calculated by using formula-

*task_alloted = cap_index *no_of_tasks;*

An Initialization module is executed in each processor. A population of strings is randomly generated without duplication.

ii. ***Load checking module*** -It will check load at all the nodes in the system and continuously is updated with the load information of the nodes connected in the system. This module will check its own processor's load by checking the CPU queue length, whenever a task is arrived in a processor. If the observed load is heavy, the load-balancing algorithm performs the following modules.

iii. ***String Evaluation module***-It will calculate the fitness function by using the equation given in equation. Then according to the fitness value calculated for a node the next genetic operation will be performed.

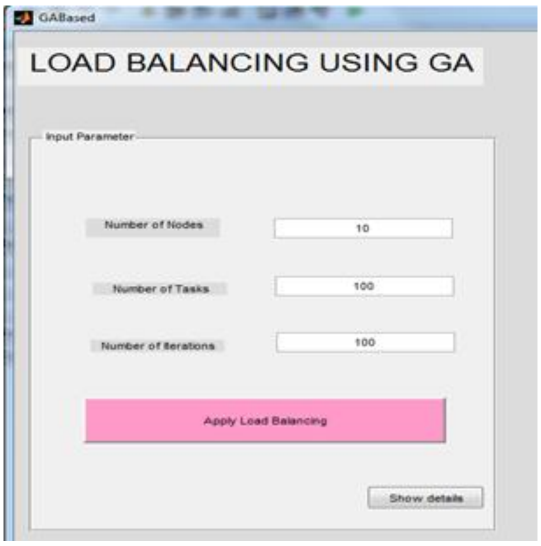
iv. ***Genetic Algorithm module*** -It consists of sub modules such as selection, crossover and mutation. Selection will be on the basis of fitness value. It uses spinning the roulette wheel method. A simple one-point crossover is used. Finally, we get the optimal node selected.

v. ***Message Evaluation module***-It will send the information to the node and assign job to the node.

IV. RESULTS AND DISCUSSION

A. GUI Interface

The GUI interface for the load balancing in distributed system is shown in figure A. Input to the system is the number of nodes, number of task to be allocated and the number of generations.



GABased

LOAD BALANCING USING GA

Input Parameter

Number of Nodes: 10

Number of Tasks: 100

Number of Iterations: 100

Apply Load Balancing

Show details

Figure A. GUI interface

B. Simulation Results

Performance of the proposed model is evaluated by its simulation. Simulation program is done in Matlab. In most of the experiment, it has been observed that the solution converges by 200 generations. Other considered parameters for experiment are as follows. Simulation Parameters: The input parameters, used in the experiment, are as follows. Range of number of machines are 20 to 100 in every experiment, Population Size is 50, Number of jobs are 100 to 1000. Generation to run the experiment is 200.

By varying number of nodes and for 200 tasks

In this experiment, we have considered fixed range of the load with number of task is 200. The workload assigned to each node is shown in figure below:

For 20 nodes:

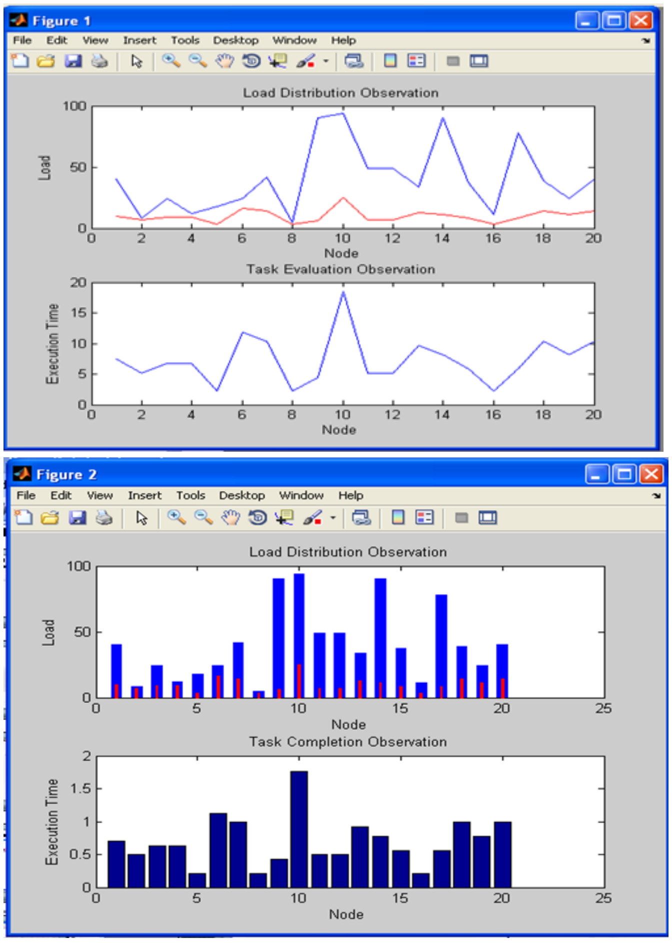


Figure 1. Result for 20 nodes and 200 tasks

For 40 nodes:

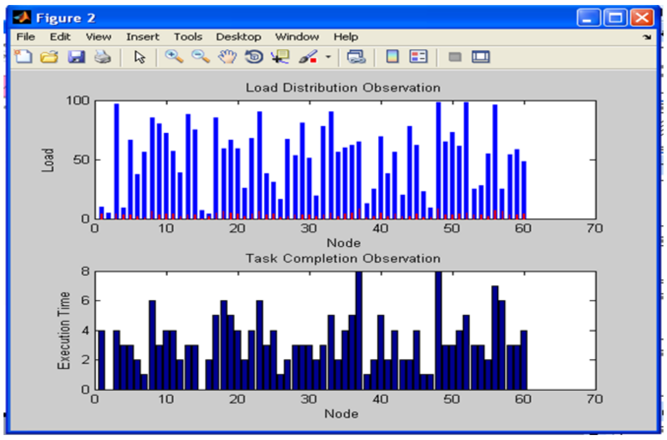


Figure 2. Result for 40 nodes and 200 tasks

For 60 nodes:

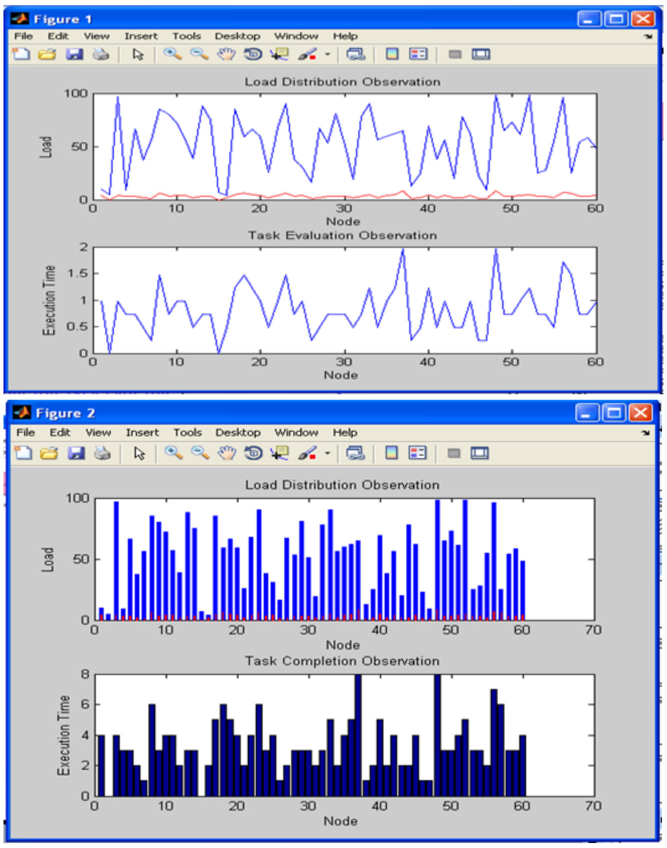
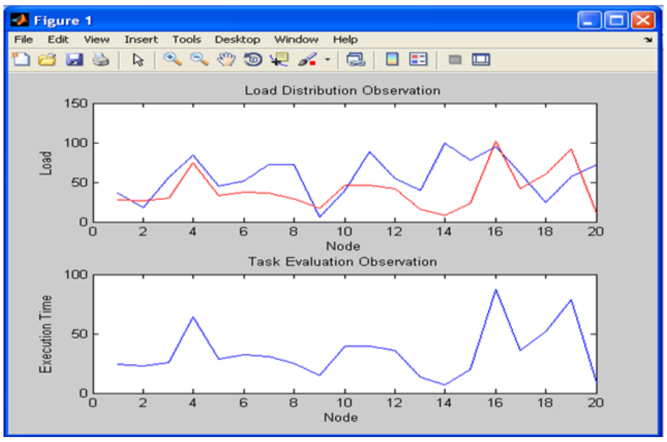


Figure 3. Result for 60 nodes and 200 tasks

By varying number of nodes and for 800 tasks

In this experiment, we have considered fixed range of the load with number of task is 800. The workload assigned to each node is shown in figure below.

For 20 nodes:



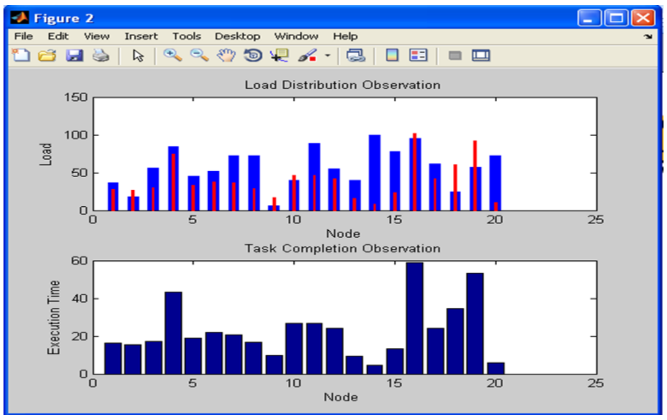


Figure 4. Result for 20 nodes and 800 tasks

For 60 nodes:

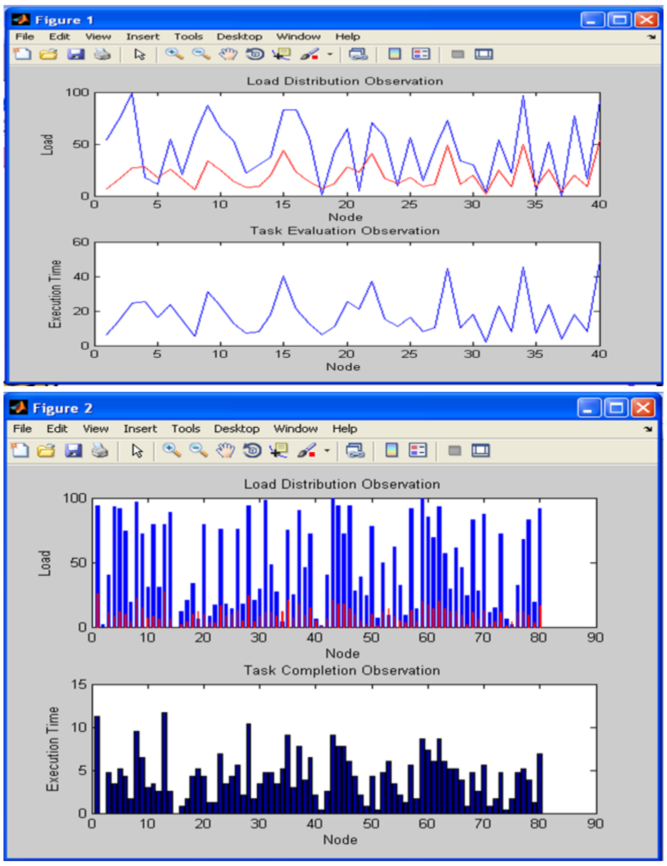
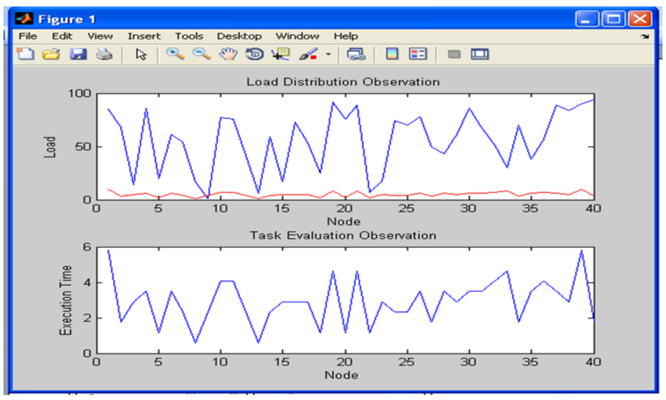


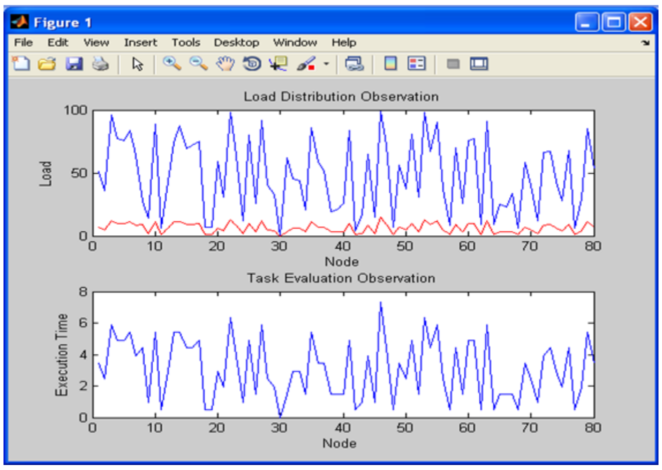
Figure 5. Result for 60 nodes and 800 tasks

By varying number of generations for varying number of nodes

In this experiment, we have considered fixed range of the load with number of task is 200. The workload assigned to each node is shown in figure below.

For 60 nodes:





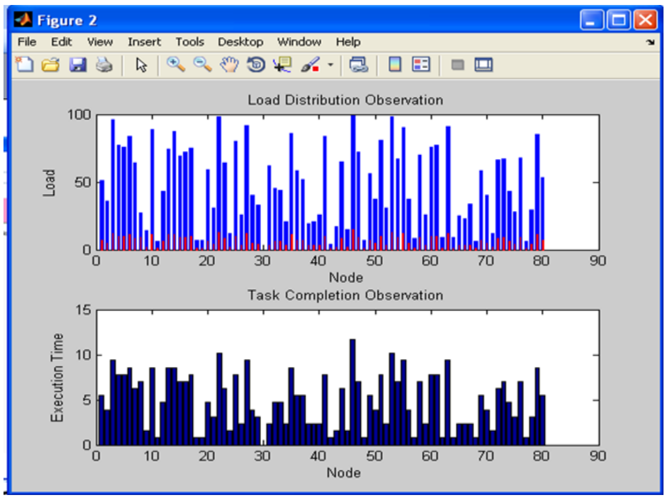


Figure 6. Result for 60 nodes and 200 tasks

For 80 nodes:

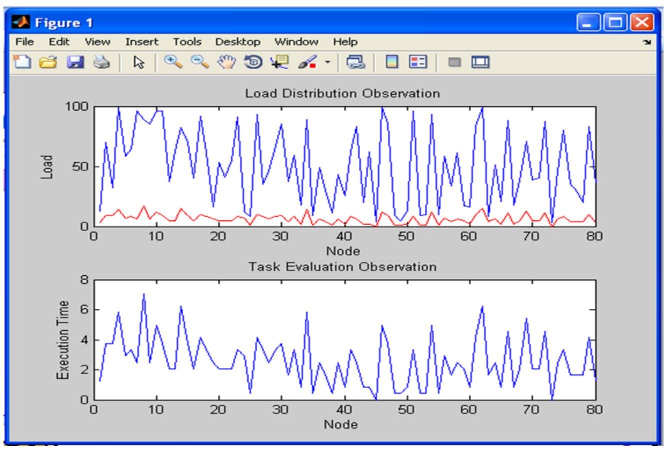


Figure 7. Result for 80 nodes and 200 tasks

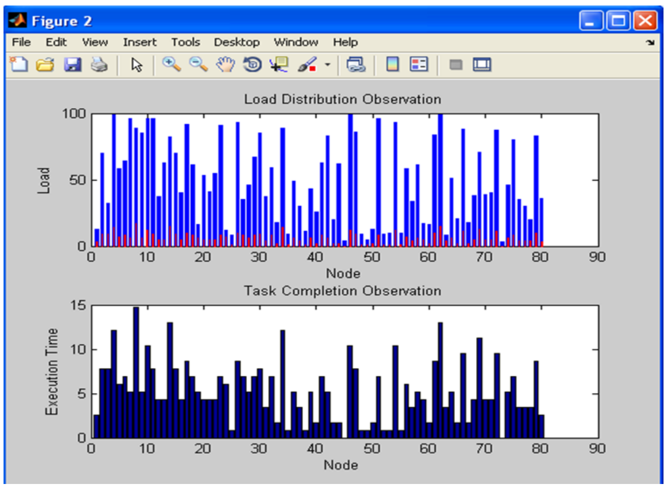
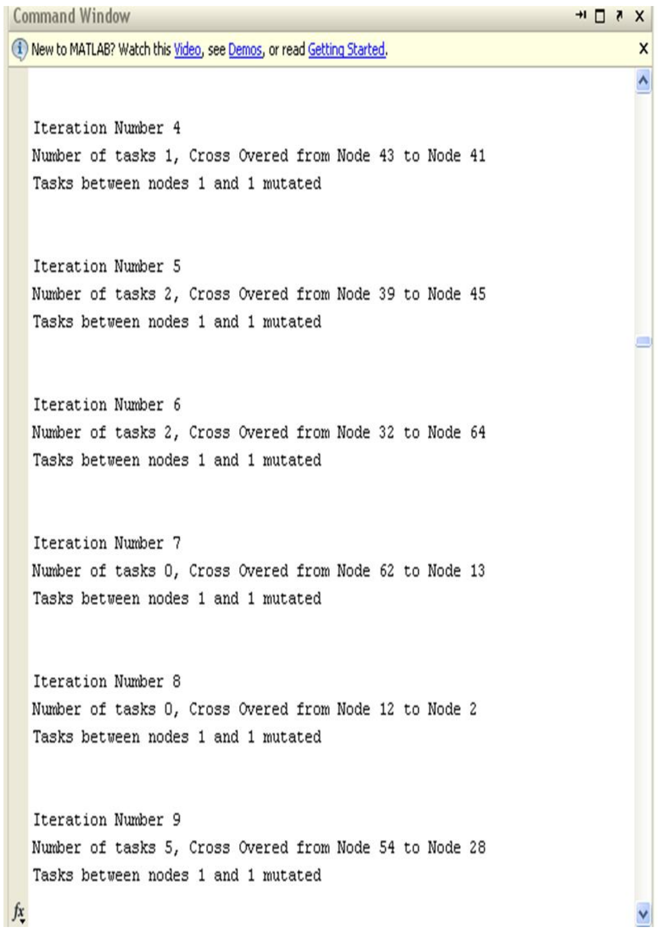


Figure 8 Results of Genetic parameters


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
Iteration Number 4
Number of tasks 1, Cross Overed from Node 43 to Node 41
Tasks between nodes 1 and 1 mutated


Iteration Number 5
Number of tasks 2, Cross Overed from Node 39 to Node 45
Tasks between nodes 1 and 1 mutated


Iteration Number 6
Number of tasks 2, Cross Overed from Node 32 to Node 64
Tasks between nodes 1 and 1 mutated


Iteration Number 7
Number of tasks 0, Cross Overed from Node 62 to Node 13
Tasks between nodes 1 and 1 mutated


Iteration Number 8
Number of tasks 0, Cross Overed from Node 12 to Node 2
Tasks between nodes 1 and 1 mutated


Iteration Number 9
Number of tasks 5, Cross Overed from Node 54 to Node 28
Tasks between nodes 1 and 1 mutated
```

V. CONCLUSION AND FUTURE SCOPE

Scheduling in distributed systems is known as an NP-complete problem even in the best conditions. In this paper, GA-Based method is used to solve the 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.

This work deals with the load balancing in distributed system with emphasis on the observation of load variation and load distribution among the nodes in the distributed environment. Overall, proposed model achieves better load distribution for all the test cases.

Future Scope

a. The work can be extended to the real-life problems faced in balancing the load of a particular system. The algorithm can be applied to the applications where there is continuous variation in the system load. The hierarchical system can be extended to more than two levels or the architecture can be extended where two or more nodes can be used as a designation node in the system.
b. Another extension of the system can be applied to the parallel systems where workload is executed in parallel with the system. The same load balancing algorithm can be applied to the cloud systems.
c. In the present system, the load is considered as a number of tasks running in the system but the file system is one of the factors that can be considered in the system. The file system is the complex system where the fault tolerance property can be improved and can have a proper file distribution property.

VI. 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, 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*", 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. 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",* 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, 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 1994, 113-120.

[13] Masaharu Munetomo, Yoshiaki Takai, Yoshiharu SatoA,"Stochastic *Genetic Algorithm for Dynamic Load Balancing in Distributed Systems"* , ISBN 0-7803-259-1 IEEE Transactions,1995

[14] Seong-hoon Lee and Chong-sun Hwang "A Dynamic Load Balancing Approach using Genetic Algorithm in Distributed Systems", IEEE 1998

[15] Mayuri A. Mehta Devesh C. Jinwala,"Analysis of *Significant Components for Designing an Effective Dynamic Load Balancing Algorithm in Distributed Systems*", Third International Conference on Intelligent Systems Modelling and Simulation 2012

[16] K. Hemant Kumar Reddy Diptendu Shina Roy,"*A Hierarchical Load Balancing Algorithm for Efficient Job Scheduling in a Computational Grid Testbed*", IEEE Transaction on 1st International conference on Recent Advances in Information Technology, 2012