Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing System

Azita Jooyayeshendi, Abbas Akkasi

Abstract—Load balancing problem on Heterogeneous Distributed Computing System (HDCs) deals with allocation of tasks to computing nodes, so that computing nodes are evenly loaded. Due the complexity of dynamic load balancing problem majority of researchers uses heuristic algorithm to obtain near optimal solutions.We have used consistent ETC (Expected Time to Compute) matrix in to study the performance of Genetic algorithm to minimize the makespan
Genetic algorithm has been developed to dynamically schedule heterogeneous tasks on heterogeneous processors in a distributed system. The scheduler operates in an environment with dynamically changing resources and adapts to variable system resources. It operates in a batch fashion and utilises a genetic algorithm to minimise the total execution time and compare with simulated annealing (SA) algorithm.

Keywords—Dynamic load balancing, Genetic algorithm, heterogeneous distributed system, makespan.

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. 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. Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors. [4]

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. Several methods have been proposed to solve scheduling problem in DCS. The proposed methods can be generally classified into three categories: Graph-theory-based approaches, mathematical models-based methods and heuristic Techniques .

Heuristics can obtain suboptimal solution in ordinary situations and optimal solution in particulars. Since the scheduling problem has been known to be NP-complete, using heuristic Techniques can solve this problem more efficiently. Three most well-known heuristics are the iterative improvement algorithms [7],the probabilistic optimization algorithms, and the constructive heuristics. In the probabilistic optimization group, GA-based methods and simulated annealing are considerable which extensively have been proposed in the literature.

One of the crucial aspects of the scheduling problem is load balancing. While recently created processes randomly arrive into the system, some processors may be overloaded heavily while the others are under-loaded or idle. The main objectives of load balancing are to spread load on processors equally, maximizing processors utilization and minimizing total execution time.

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 have proposed scheduling algorithms considering load balancing .[4]

In Section II we review related work and give an overview of how a GA operates. In Section III we describe our scheduling algorithm. In Section IV we present the results of our performance experiments. In Section V we give our conclusions and suggest future directions for our work .

2 RELATED WORKS

- *Azita Jooyayeshendi, Islamic Azad University Bandar Abbas Branch, Iran. E-mail: Azita.Jooya1@gmail.com*
- *Abbas Akkasi, Computer Engineering Department of Islamic Azad University Science and Research Branch, Tehran, Iran. E-mail: Abbas.Akkasi@gmail.com*

Load balancing for distributed computing system is a problem that has been deeply studied for a long time. Different heuristic algorithms are used by researcher to find suboptimal solutions for homogeneous and heterogeneous distributed system. Dandamudi addressed dynamic load sharing in distributed systems and established that load sharing improves performance by moving work from heavily loaded nodes to lightly loaded nodes. A general model for heterogeneous distributed/parallel computer system proposed by Li and Kameda [1] and used to formulate the multiclass job load balancing problem as a nonlinear optimization problem. An algorithmic approach to load balancing problem is presented in [10]. Different form of linear programming formulation of the load balancing problem has been discussed along with greedy, randomized and approximation algorithm to produce sub-optimal solutions to the problem. The solution to this intractable problem was discussed under different algorithm paradigm. Modeling of optimal load balancing strategy using queuing theory was proposed by Francois Spies (1996). This is one of the pioneer works reported in the literature that presents an analytical model of dynamic load balancing techniques as M/M/k queue and simulate with fundamental parameters like load, number of nodes, transfer speed and overload rate. Most appropriate queuing model for homogeneous distributed system can be M/M/m/n, has been analyzed in [1].

Queuing-Theoretic models for parallel and distributed system can be found in [6]. General Job scheduling problem of n tasks with m machines, is presented as an optimization problem in to minimize the makespan. Jong-Chen Chen and et al. investigated the contribution made by evolutionary learning on dynamic load balancing problems in distributed computing system. Bora Ucar and et al. have considered the assignment of communicating tasks to heterogeneous processors, that uses a task clustering method based upon execution time to allocate the task though the heuristic techniques.

A classification of iterative dynamic load balancing technique is discussed in . SA is a heuristic method that has been implemented to obtain good solutions of an objective function defined on a number of discrete optimization problems. Simulated Annealing (SA), proposed by Kirkpatrick et al, has been used as a popular heuristic to solve several optimization problems to obtain sub-optimal solution.

A heuristic algorithm based on simulated annealing is discussed , which guarantees good load balancing on grid environment. A comparative study of the three algorithms (Hill-climbing, simulated annealing and genetic algorithms) is then carried out in considering performance criteria as the amount of search time. Makespan minimization of scheduling problem on identical parallel machines using simulated annealing has been presented by Lee and et al. in

Grid Computing is one of heterogeneous distributed computing system geographically dispersed among several entities. Fidanova used simulated annealing to obtain near optimal solutions for scheduling problem in large grid . Researchers have examined, different heuristics(Opportunistic Load Balancing, Minimum Execution Time, Minimum Completion Time, Min–min, Max–min, Duplex, Genetic Algorithm, Simulated Annealing, Genetic Simulated Annealing, Tabu, and A*) on Mixed-machine heterogeneous computing (HC) environments to minimize the total execution time of the metatask.

Rahmani and Rezvani presented a genetic algorithm for static scheduling, which is again improved by simulated annealing to obtain an improvised solution. They have also established that running time depends on the number of task. Several researchers used SA and GA for load balancing on distributed computing system; however majority of the papers have no specific representation for Genetic algorithms for load balancing.[1]

3 HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM MODEL

3.1 HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM

Heterogeneous distributed computing system (HDCS) utilizes a distributed suite of different high-performance nodes, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements [11,12]. 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 figure1 [10].

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 [11].

Heterogeneity in DCS can be expressed by considering three systems attributes (i) Processor with computing node, (ii) memory, and (iii) networking . The metrics used to quantify the processor or node processing power by means of processing speed and represented with FLOPS (Floating point Operations per Second) and can be measured through LINPACK. Memory attributes are measured as the available memory capacity to support the process. The

networking attributes are the link capacity associated with transmission medium, propagation delay and available communication resources [1].

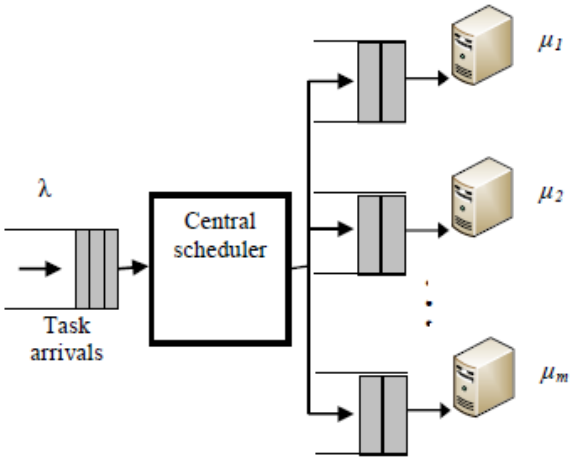

Fig.1. Heterogeneous Distributed Computing System with central Scheduler

In paper we have carried out simulation only considering processing power of the node, which can be represented as Markovian service time distribution [1]. In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds. We have used a centralized load-balancing algorithm framework as it imposes fewer overheads on the system than the decentralized algorithm .Centralized load balancing algorithms requires the global information on computing nodes at a single location and the load balancing policy is initiated from the central location. Heterogeneity of architecture and configuration complicates the load balancing problem [11].

Heterogeneity can arise due to the difference in task arrival rate at homogeneous processors or processors having different task processing rates. We have assumed that all computational tasks are capable of executed on any computing nodes of DCS. A single computing node that acts as a central scheduler or resource manager of the DCS collects the global load information of other computing nodes.

Resource management sub systems of the HDCS are designated to schedule the execution of the tasks dynamically as that arrives 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. A basic assumption is that all computing nodes are always available for processing[1].

3.2 Load balancing problem in Heterogeneous distributedcomputing system

We have used the characterization model proposed by Shoukat Ali and et al as the basic framework to study the impact of system heterogeneity against different heuristic resource allocation algorithms [1]. We consider a heterogeneous distributed computing system (HDCS) consists of a set of M = {M1, M2, … Mm}, m independent heterogeneous, uniquely addressable computing entity (computing nodes). Let there are T = {t1, t2, …, tn} n number of tasks with each task ti has an expected time to compute tij on node M_j. The entire task has expected time to compute on m nodes of HDCS. Hence the generalized load-balancing problem is to assign each task to one of the node M_j so that the loads placed on all nodes are as "balanced" as possible [10] Let A(j) be the set of jobs assigned to node M_j; and Tj be the total time machine M_j have to work to finish all the task in A(j).

Hence $T_j = \sum t_i \in x_j\, t_{ij}$; for all task in A(j). This is otherwise denoted as Lj and defined as load on node Mj. The basic objective of load balancing is to minimize make span, which is defined as maximum loads on any node (T = maxj:1:m (Tj)) . Let x_{ij} correspond to each pair (i,j) of node $M_j \in M$ and task $t_i \in T$.

$x_{ij} = 0$; implies that task i not assign to node j.
$x_{ij} = t_{ij}$; will indicate load of task i on node j.
For each task t_i we need $\sum_{j=1}^{m} x_{ij} = t_{ij}$; for all task $t_i \in T$.

The load on node M_j can be represented as $L_j = \sum_{i=1}^{n} x_{ij}$, where $x_{ij} = 0$ whenever task $t_i \notin (A_j)$. The load balancing problem aims to find an assignment that minimizes the maximum load. Let L be the load of a HDCS with m nodes. Hence the generalized load balancing problem on HDCS can be formulated as Minimize L :

$$\sum_{j=1}^{m} x_{ij} = t_{ij} \quad \text{, for all } \; t_i \in T \quad (1)$$

$$\sum_{i=1}^{n} x_{ij} \leq L \quad \text{, for all } \; M_j \in M \quad (2)$$

$$x_{ij} \in \{0, t_{ij}\} \text{ , for all } t_i \in T, M_j \in M$$

$$x_{ij} = 0 \quad \text{, for all } \; t_i \notin A(j)$$

Feasible assignments are one-to-one correspondence with x satisfying the above constraints [1]. Hence an optimal solution to this problem is the load L_i on a machine (corresponding assignment).

The problem of finding an assignment of minimum makespan is NP-hard . The problem is therefore untractable with number tasks or computing nodes

(processors) exceeds a few units. The solutions to load balancing problem can be obtained using a dynamic programming algorithm with time complexity O(nLm), where L is the minimum makespan[10] The load balancing problem has been evenly treated, in both the fields of computer science and operation research. The algorithm approaches used for load balancing problem are roughly classified as (i) exact algorithms and (ii) heuristic algorithms.

Queuing models are used as the key model for performance analysis and optimization of parallel and distributed system. The HDCS can be modeled as M/M/m/n (Markovian arrivals, Markovian distributed service times, m computing nodes as server, and space for n ≥ m tasks in the system) multi-server queuing system with m servers as computing nodes. However, the heterogeneous multi-server queuing systems are not adequately addressed in research with respect to certain quality of service .The HDCS is modeled as M/M/m/n queuing system with node M1 is the fastest computing node and Mm is the slowest computing node. Assume that service time follow exponential distribution with service rate so that $\mu_1 > \mu_2 > \cdots \mu_m$, where μ_1 is the service rate of node Mi. The arrivals of the tasks at the central server or resource manager are modeled as Poisson with arrival rate .

The tasks that are to be executed at a node are under the control of local scheduler and the scheduling policy of the node is responsible for the execution of the assigned task. We have assumed FCFS policy is being used at computing nodes, which can be modeled as M/M/1 queuing system [1].

4 TASK MODEL AND ITERATIVE LOAD BALANCING TECHNIQUES

4.1 TASK MODEL ON HDCS

In literature of distributed computing researchers have used two different task models as (i) Task graph(TG) or Task

interaction graph(TIG)[1], (ii) expected time to compute(ETC) matrix[6].

The task graphs are both directed and undirected weighted graph that represents process or task to be executed, however majority of the models are not representing any mathematical model for quantifying task heterogeneity. In this paper we have use ETC matrix representation of task that represents task heterogeneity and machine heterogeneity. The tasks are arriving from the different users or nodes to the central scheduler or or serial scheduler have the probability to be allocated to any of the m computing nodes. Hence the tasks are characterized by expected time to compute (ETC) on all m computing nodes, can be represented as follows, In ETC matrix, the elements along a row indicate the execution time of a given task on different nodes[1], in particular tij represent expected time to compute ith task on machine Mj .

TABLE 1
EXPECTED TIME TO COMPUTE (ETC) MATRIX

	M_1	M_2	⋯	M_J	⋯	M_m
T_1	t_{11}	t_{12}	⋯	t_{1j}	⋯	t_{1m}
T_2	t_{21}	t_{22}	…	t_{2j}	…	t_{2m}
T_i	t_{i1}	t_{i2}		t_{ij}		t_{im}
T_n	t_{n1}	t_{n2}		t_{nj}		t_{nm}

The ETC model presented are characterized by three parameters (i) machine heterogeneity, (ii)task heterogeneity and (iii)consistency. The task heterogeneity can be represented with two categories (i) consistent and (ii) inconsistent, here a consistent ETC matrix the computing nodes are arranged in the order of their processing capability or may be arranged as decreasing order of FLOPS. In particular a node Mi has a lower execution time than node Mj for task tk , then tki< tkj . Inconsistent ETC matrix is resulted in practice, when HDCS includes different type of machine architectures.(HPC clusters, Multi-core processor based workstations, parallel computers, work station with GPU units). In literature most of the task execution times are uniformly distributed. A consistent ETC matrix for ten tasks on five machines is shown on table II, which is taken from . To generate ETC matrix, we have used range base ETC generation technique discussed and added one component as arrival time of task. The arrival pattern of the task is based on Poisson distribution. For the analysis of the simulation results through the graph we have used expected

completion time of task uniformly distributed {1, 500} time unit or seconds.[1]

TABLE 2
EXAMPLE OF CONSISTENT ETC MATRIX FOR 10 TASKS ON FIVE MACHINES

Node→ Task ↓	M1	M2	M3	M4	M5
t1	22	21	6	16	15
t2	7	46	5	28	45
t3	64	83	45	23	58
t4	53	56	26	42	53
t5	11	12	14	7	8
t6	33	31	46	25	23
t7	24	11	17	14	25
t8	20	17	23	4	3
t9	13	28	14	7	34
t10	2	5	7	7	6

2.2 ITERATIVE CENTRALIZED ALGORITHMS

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 send the task to other nodes or accept new task for computation. The computing nodes depend on the information available with central node for all allocation decision .

GA based load balancing algorithm uses an iterative structure with stopping criteria as maximum number of iteration. We have also assumed that tasks are independent and can be processed by any computing node in distributed environment. For stability it is also assumed that tasks must not be generated faster than the HDCS can process as shown in equation 3[1].

$$\sum_{i=1}^{n} \lambda_i \leq \sum_{j=1}^{m} \mu_j \quad (3)$$

4.3 CODING SCHEME FOR THE SOLUTION

Simulated annealing algorithms require a suitable representation and evaluation mechanism. In this case we have use a window structure of fixed length say k, with integer value assigned to individual element of the array of size k. That on each step k no of task to be allocated to the computing node through simulated annealing with a minimized value of makespan. Task is assigned dynamically to the computing nodes on the fly. At the time of allocation there may be a large number of tasks are with central scheduler. A sliding window technique is used to select those tasks only that are in the window. The number of elements in the window is fixed is equal to the size of window.

Figure 2. represents 10 tasks and their respective allocation to five computing node. Figure 3 shows the structure of allocation list, indicates the computing node. We have assumed that, current work load as dedicated tasks for each own node, so that the calculation of makespan is carried out from the time point when sliding window is selected[1].

t1	***t2***	***t3***	***t4***	***t5***	***t6***	***t7***	***t8***	***t9***	***t10***
M5	***M3***	***M5***	***M3***	***M2***	***M2***	***M4***	***M4***	***M1***	***M1***

Fig.2. allocation list of task to computing node

5	***3***	***5***	***3***	***2***	***2***	***4***	***4***	***1***	***1***

Fig.3. Allocation list

Figure 4. shows the makespan=73 for the chromosome in figure 3. with corresponding average utilization (AU) of fivecomputing nodes..

Node	A(i)		L_i	AU
1	t(9,1)=13	t(10,1)=2	15	0.2054
2	t(5,2)=12	t(6,2)=31	43	0.5890
3	t(2,3)=5	t(4,3)=26	31	0.4246
4	t(7,4)=14	t(8,4)=4	28	0.3835
5	t(1,5)=15	t(3,5)=58	73	1.0000

Fig.4. Makespan of the system

4.4 GENETIC ALGORITHM

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics. GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. To evolve good solutions and to implement natural selection, we need a measure for distinguishing good solutions from bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness

measure must

3	3	4	5	2	2	4	1	1	4

```
initialise population
do{
crossover
random mutation
selection
}while(stopping conditions not met)
return best individual
```

determine a candidate solution's relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions [4].

The proposed algorithm for task scheduling considering load balancing is presented in figure 5.

Fig.5. Pseudo code for genetic algorithm

5.4 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 [8].

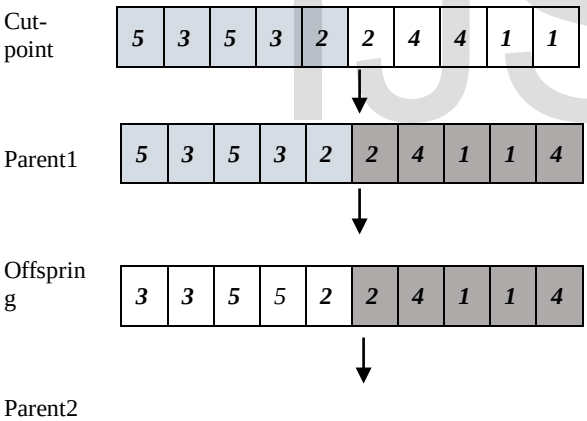

Fig.6. (a) Crossover

6.4 MUTATION

While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution [4].

3	3	5	5	2	2	4	1	1	4

Before

After

Fig.7. (b) Mutation

7.4 FITNESS FUNCTION

After generating the population we have to perform the selection operation. This operation can be performed by using fitness function as shown below.

```
fitness
%makespan calculation
for k=1:size(ts,2)
  ind = find(ts_assign == k);
  if(length(ind)>0)
     L(k) = sum(ts(ind,k));
  else
     L(k) = 0;
  end
end
AU = L / max(L);
ms = max(L);
```

8.4 GENETIC FUNCTION

In our model, Genetic algorithm starts with generating initial schedule TS randomly for 100 tasks. A final allocation list for the tasks is obtained after 80 iteration. Tasks are allocated to the nodes and average utilization is calculated for those 100 tasks before selecting a next 100 tasks from the set of waiting tasks. The algorithm GA_Function called for maximum (n/ WIN_SIZE) times to allocate n tasks to the computing nodes.

```
GA_function ()
WIN_SIZE = 100; %tasks which process at each step
m = size(Tasks,2); %cpu size
n = size(Tasks,1); %tasks amount
makspan(1) = 0;
pop_size = 100; %population size
max_itr = 80;
for i=1:n/WIN_SIZE %execute GA for each Win_size tasks
    ts = Tasks((i-1)*WIN_SIZE+1:(i)*WIN_SIZE,:); %seperate each WIN_SIZE amount of tasks
    %initialize population
    for j=1:pop_size
      pops(j,:) = randi([1 m],WIN_SIZE,1); % first assign randomly
    end
for itr=1:max_itr
```

```
%cross over
N = pop_size + 1;
for j=1:round(pop_size*.8/2)
   %select two parents
   n1 = randi([1 pop_size]);
   n2 = randi([1 pop_size]);
   %select cross-over point
   cp = randi([1 size(pops,2)]);
   %first child
   pops(N,1:cp) = pops(n1,1:cp);
   pops(N,cp+1:end) = pops(n2,cp+1:end);
   %first child
   pops(N+1,1:cp) = pops(n2,1:cp);
   pops(N+1,cp+1:end) = pops(n1,cp+1:end);
   N = N + 2;
end
%mutation
for j=pop_size+1:size(pops,1)
  for f=1:(size(pops,2)*.2)

      pops(j,f) = randi([1 m]);
  end
end
%makspan calculation(ms)
for j=1:pop_size
   ts_assign = pops(j,:);
   fitness;
   MS(j) = ms;
   au(j) = mean(AU);
end
%selection
[v,ind] = sort(MS);
temp = pops(ind(1:pop_size),:);
clear pops
pops = temp;
clear temp
end
%final makspan calculation(ms) for this part
ts_assign = pops(1,:);
fitness;
if(i==1)
   makspan(i) = ms;
else
   makspan(i) = makspan(i-1) + ms;
end
Au(i) = mean(AU);
```

Common approaches used as the stopping criteria in Genetic algorithm (GA) are, (i) one may use a given number of iteration, or (ii) a time limit, or (iii) a given number of iteration without an improvement of the objective function value, (iv) value of the objective function limit as set by the user[1]. We have used a fixed number of iteration proportional to number of task to be schedule on computing nodes. We have use Matlab to design our simulation programs. The experiment was conducted with n=1000 tasks on m=50 computing nodes. The simulation results are compared with simulated annealing algorithm.

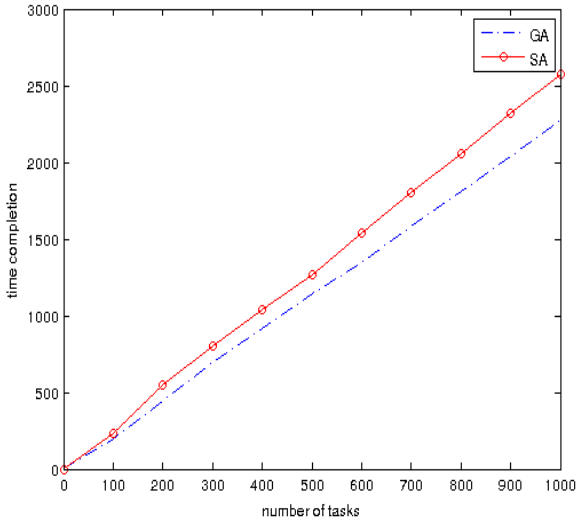

Fig.8. Completion Time of 1000 tasks on

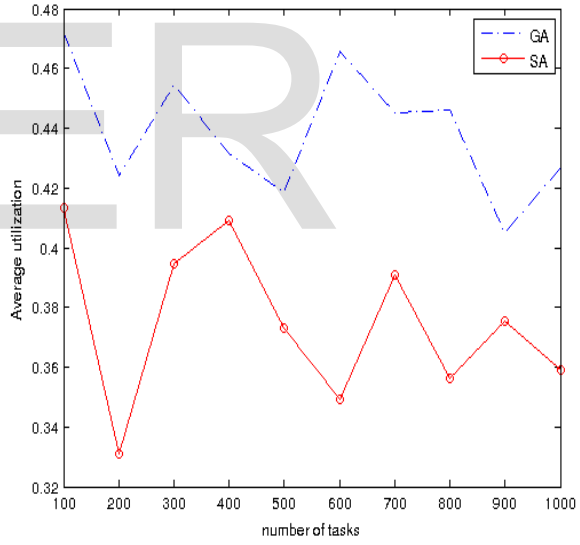

Fig.9. Average Processor Utilization

5 CONCLUSION AND FUTURE WORK

Scheduling 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 new GA-Based method to solve this problem. This algorithm considers multi objectives in its solution evaluation and solves the scheduling problem in a way that simultaneously minimizes maxspan and communication cost, and maximizes average processor utilization and load-balance. Most existing approaches

tend to focus on one of the objectives.

The Figure: 4.2 show that the total time taken for two

algorithms increased linearly as the number of tasks was increased. It was also noted that the GA performed better than the SA algorithm. When comparing the results of the GA and the SA algorithm, one can observe that the gap between these two curves was widening as the number of tasks was increased. This shows that the GA actually reduced the total completion time by a considerable amount (greater speedup) in comparison to the SA algorithm as the number of tasks increased. This also indicates reliable performance of the GA_loadbalancing when the number of tasks increases.

The GA-based load balancing algorithm shows better performance to that of SA in both average processor utilization and completion time or makespan .

Ant Colony algorithm have been proposed over the years for solving static and dynamic load balancing problems on

distributed system. The coding method introduced in this paper can be used to design a Ant Colony algorithm for dynamic loadbalancing in HDCS.

REFERENCES

[1] Sahoo, B., Jena S.K., and Mahapatra, SX, "Simulated Annealing based Heuristic Approach for Dynamic Load Balancing Problem on Heterogeneous Distributed Computing System", (2012).

[2] Page, A. J., Naughton T. J, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing", (2005).

[3] Daoud, M. I., Kharma, N, "An Efficient Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing Systems", (2006).

[4] Gonnade(Kohale), P. G., Bodkhe, S, "Genetic Algorithm for Task Scheduling in Distributed Heterogeneous System", (2012).

[5] Sahoo, B., Jena S.K., and Mahapatra, S, "A Genetic Algorithm Based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems", (2008).

[6] Boxma, O., Koole, G., and Liu, Z, "Queueing-theoretic solution methods for models of parallel and distributed systems", (1994).

[7] Mortazavi, S.S., Rahmani, A.M, "Two new biasing load balancing algorithms in distributed systems", (2009).

[8] Nikravan, M, "A Genetic Algorithm-Based Approach for Process Scheduling In Distributed Operating Systems", (2012).

[9] Ghosh, S, "Distributed systems: an algorithmic approach", (2006).

[10] Kleinberg, J., Tardos, E, "Algorithm Design", (2006).

[11] Wu, J, "Distributed system design", (1999).

[12] Hwang, k.k., Fox,G.C., and Dongarra, J.J, "Distributed and Cloud Computing: From Parallel Processing to the Internet of Things", (2012).