ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

A Survey on Scheduling in Heterogeneous Distributed Systems for Higher Reliability

Sonia I. Fernandes[1], Kranti Wagle[2]

PG Student, Dept. of Electronics, Fr. Conceicao Rodrigues College of Engineering, Bandra, Maharashtra, India[1]

Assistant Professor, Dept. of Electronics, Fr. Conceicao Rodrigues College of Engineering, Bandra, Maharashtra, India[2]

ABSTRACT:Heterogeneous computing systems create unlimited opportunities and challenges in the fields of distributed processing, design of algorithms. Almost all the devices humans use in their day-to-day lives incorporates a heterogeneous distributed system. Active research is going on in field of grid computing which incorporates a heterogeneous distributed system. Scheduling the tasks, overcoming from failures and energy consumption are the major challenging issues in heterogeneous distributed systems. In completion of a particular application, scheduling of the tasks plays an important role. The goal of scheduling is to utilize all the processors with minimum execution time by proper allocation of tasks to the processors. Appropriate task scheduling helps the heterogeneous distributed systems to attain high performance. In order to achieve higher throughput, total completion time of the application should be as small as possible. Failures of machine (processor or link) causes a delay in completion of the tasksThis paper presents a survey of methods used for scheduling a big computing problem into a grid with higher efficiency

KEYWORDS:task graph, precedence constraints, reliability model,grid

I.INTRODUCTION

In today's highly advanced world, the need of computing is increasing vastly. For big applications, the computations are too vast. It might take a huge amount of time for the code to compute on a single processor. A solution to this is parallel computing wherein the entire program is divided into various and each part is computed on a separate processor. Due to this the time required is quite less. There are two subtypes of this viz., shared memory and distributed memory. In shared memory, many processors share a single memory. But in this type of parallel computing, there is lack of scalability viz., increase in the number of processors doesn't increase the performance. In distributed memory, each processor has its own local memory as well as global memory. Due to this, the memory access time will be less as less time will be required for local memory. Adding extra processor won't add to the network traffic but increase the performance.

II.GRID COMPUTING

A. INTRODUCTION

The ubiquity of the internet as well as the availability of powerful computers and high-speed network technologies as low-cost commodity components is rapidly changing the computing landscape and society. These technologies opportunities have led to the possibility of using wide-area distributed computers for solving large-scale problems, leading to what is popularly known as Grid computing[6]. The term Grid is chosen as an analogy to the electric power grid that provides consistent, persuasive, dependable, transparent access to electricity, irrespective of its source. Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering and commerce.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

B. APPLICATION OF GRID COMPUTING

For very high end application, huge codes are written. This code if implemented using serial computing take many years to compute. Thus, a solution for this is supercomputing where in many processors with huge memory are placed in a huge room. The advantage is the code is executed at a faster rate. But this costs a huge amount of money and dedicated huge space. The solution to this is grid computing. Due to this the huge problem are solved easily with low cost and no need of dedicated space.

III.LITERATURE SURVEY

While designing a grid computing environment, reliability and energy conservation is quite necessary. A lot of research is carried out is these areas. The algorithms proposed for heterogeneous scheduling are mapping heuristics (MH) algorithm[9], the dynamic level scheduling(DLS) algorithm[10], the levelized min-time (LMT) algorithm, the critical path on machine (CPOP)[2] algorithm, the heterogeneous earliest finish time (HEFT)[2]algorithm,MLST[7] and dominant sequence clustering algorithm[8].

IV.TASK AND PROCESSOR MODEL

In order to schedule tasks on heterogeneous distributed systems, a task model and processor model needs to be designed. A task model is the arrangement of tasks in terms of the precedence constraints. The task graph has one entry node and one exit node. Other tasks are arranged in level. This arrangement of tasks is called as Directed Acyclic Graph as all task are directed to other task and all in one direction. The example of DAG is given in Fig. (1).

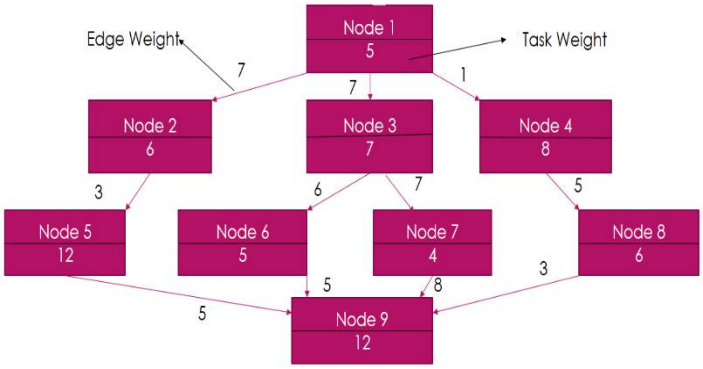

Fig. 1 Task Model

Here the nodes are the parts of the actual task to be executed. The task weight symbolizes the number of instructions in the given tasks and edge weight written in the figure symbolizes the amount of data the task requires from its predecessors for its execution.

In order to form such a task graph, the inputs required are:

- **The number of the tasks** = *N*
- **Mean Weight of tasks:** *W*
  It tells approximately how many operations are to be performed in one tasks.
- **Communication to Computation Cost Ratio =** *CCR*
  Depending on the value of *CCR*, the application can be divided as computation intensive task if *CCR* is low and the tasks having high *CCR* are communication intensive tasks
- **Shape Factor** = α
  This value tells how the graph will look visually. High value of α imply a wider graph and high level of parallelism and low value of α gives a graph with many levels and low level of parallelism.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

In order to execute these tasks, a heterogenous distributed model of processors is required to be formed. The Fig. 2 shows how processors model is formed.

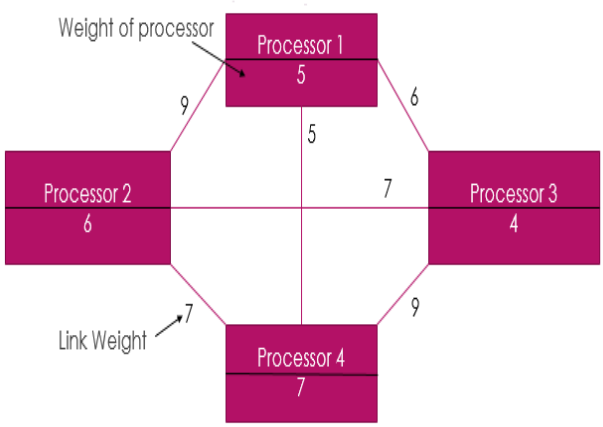

Fig. 2 Processor Model

The weight of the processor implies the number of operations the processor completes in unit time and link weight implies how much data can be transferred in unit time.

In order to form such a processor model, the input required are:

- **The number of processors**: *M*
- **Heterogeneity Factor**: *h*
  This value shows the degree of variance in the computational cost of the processors.

V. RELIABILITY MODEL

The reliability model is needed to be included in the scheduling if the reliability of task is major concern. The reliability model of *Shatz and Wang*[3]is mostly used model. According to their model, the failure of the processor or links follows a Poisson's distribution and the failure rate is constant
for a given processor or link. Thus, it can be also stated that the reliability of a processor/link during interval *d* is given $e^{-\lambda d}$
where λ is failure rate of processor or links.
The reliability and failure reliability of task v_i on processor p_x is given respectively by

$$R\left[E_{v_i,p_x}\right] = e^{-\lambda_{p_x} t_p(v_i,p_x)} \quad (1)$$

$$F\left[E_{v_i,p_x}\right] = 1 - R\left[E_{v_i,p_x}\right] \quad (2)$$

Where, λ_{p_x} is the failure rate of processor and $t_p(v_i,p_x)$ the processing time of task v_i on processor p_x.
The reliability and failure reliability of communication between tasks *i andj* where i^{th}task is scheduled on x^{th} processor and j^{th}task on the y^{th}processor is given respectively by

$$R\left[E_{v_{i,j},p_{x,y}}\right] = e^{-\lambda_{x,y} t_c(v_{i,j},p_{x,y})} \quad (3)$$

$$F\left[E_{v_{i,j},p_{x,y}}\right] = 1 - R\left[E_{v_{i,j},p_{x,y}}\right] \quad (4)$$

Where, $\lambda_{x,y}$ is failure rate of link between processors *x* and *y* and $t_c(v_{i,j},p_{x,y})$ is the time required by j^{th}task to communicate with i^{th}task.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

VI. SCHEDULING ALGORITHM

A. HEFT Algorithm

The Heterogeneous Earliest Finish Time Algorithm aims at scheduling tasks on processors which has lowest completion time.

The HEFT algorithm is given below[2]

1. Take the input parameters and form the task graph and heterogeneous distributed processor model.
2. Calculate the various reliability related values and normalize them
3. Compute the upward rank by traversing the graph from exit node.
   $Rank_u(n_i) = W_i + \max_{n_j \in succ(n_i)}(c_{ij} + Rank_u(n_i))$ (5)
   While calculating rank, its value for the exit task is taken $W_{exit\ task}$
4. Arrange the tasks in decreasing order of their rank.
5. While unscheduled task in the queue do,
   a. Assign task to the processor with minimum EFT (Earliest Finish Time).
   b. End

In order to calculate the Earliest Finish Time for task n_i on processor p_j following steps are followed:

- Initialize $EST(n_{entry}, p_j)$ as zero
- Compute EST (Earliest Start Time) for all other task on the same processor using this formula
  $EST(n_i, p_j) = max\left\{avail[j], \max_{n_m \in pred(n_j)}(AFT(n_m) + C_{m,j}\right\}$ (6)
  Where, $avail[j]$ is the time after which the j^{th}processor is available for execution and $AFT(n_m)$ is the actual finish time of the task. It is calculated as the EFT of that task on the processor on which it is scheduled.
- $EFT(n_i, p_j) = w_{ij} + EST(n_i, p_j)$
  Where, w_{ij} is the time required by the task *i*on processor *j*.

B. HAAS Algorithm

The Heterogeneous Allotment -Aware Scheduling Algorithm is three step processviz., allotment parameter, rank calculation and makespan allotment.

Abbrevation	Meaning
$\bar{t_p}(v_i)$	Mean processing time of task v_i
$\bar{t_c}(e_{i,j})$	Mean communication time between task v_i and v_j
θ_p	Recovery time of processor
α_i	Number of allocated processors to task v_i
F_i	Mean failure probability of task v_i
$T_p(v_i, \alpha_i)$	Total processing time required by task v_i on α_i allocated processor
β	Reliability overhead
$\bar{R}\left[E_{e_{i,j}}\right]$	Mean reliability of communication between task v_i and v_j
$\bar{R}\left[E_{v_i}\right]$	Mean reliability of computation of task v_i

Table 1. Commonly used Abbreviations[1]

1) **Allotment Parameter[1]:** Usually, each task is being allotted a single processor to execute on. But if this processor fails, the task has to wait until the processor recovers itself from failure. Thus, the makespan increases if large number of failures takes place, hence reliability is low. In order to increase the reliability of the application,

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

the number of allotted processor should be increased. Increasing the allotted processor, decreases the efficiency of the application. Hence, the allotment of processors should be done appropriately.
In order to find the appropriate number of processor to be allotted, two parameters viz., Allotment Lower Bound and Allotment Parameter are calculated.
Allotment Lower Bound is usually set to 1 viz., at least one processor must be allocated to each task.

Allotment parameter (μ) is given by[4]

$$\mu = \frac{(2+\rho)m - \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m}}{2} \quad (7)$$

The table of ρ for various values of m is given below

m	ρ	m	ρ
2	0	8	0.5921
3	0.1198	9	0.6703
4	0.2290	10	0.7450
5	0.3293	11	0.8164
6	0.4225	12	0.8851
7	0.5098	13	0.9513

For $m \geq 14$, $\rho = 0.9999$

Table 2. Values of ρ for corresponding values of m[1]

The value of α_i is chosen from the maximum of both the parameters.
The total processing time required by task v_i on α_i allocated processor is given by

$$T_p(v_i,\alpha_i) = \bar{t_p}(v_i) + \left(\frac{\bar{t_p}(v_i)}{2} + \theta_p\right)\left(\frac{F_i{}^{\alpha_i}}{1-F_i{}^{\alpha_i}}\right) \quad (8)$$

2) **Rank Calculation[1]:** The rank of the set of tasks is related the priority of the tasks. The priority of the tasks is usually the total time required for the calculation as well as considering the reliability consideration. Rank of the task is calculated from the exit task to the entry tasks.
The rank of the tasks are calculated by the

$$RRank = T_p(v_i,\alpha_i) + T_p(v_i,\alpha_i)\beta(1-\bar{R_i})^{\alpha_i} + max_{v_j \epsilon succ(v_i)}\left(\bar{t_p}(e_{ij}) + RRank(v_j)\right) \quad (9)$$

Where $\bar{R_i}$ is given by

$$\bar{R_i} = P(no\ failures\ occurs) + \sum_{k=0}^{\infty} P(k\ recoverable\ failures\ occuring) * \prod_{v_k \epsilon pred(v_i)} \left(1-\left(1-\bar{R_i}(e_{k,i})^{\alpha_k}\right)\right)$$

$$= \frac{\bar{R}[E_{v_i}]}{1-\left(1-\bar{R}[E_{v_i}]\right)\theta_p} \prod_{v_k \epsilon pred(v_i)} \left(1-\left(1-\bar{R_i}(e_{k,i})^{\alpha_k}\right)\right)$$

3) **MakespanMinimization[1]:** The need of this process is to schedule the tasks on the processors such that the makespan is reduced and the reliability is high[5]. The algorithm is

a. while scheduling list is not empty do
b. Remove the first task v_i form the scheduling list and set $S_i = \emptyset$ and $CT = \infty$
c. while $\alpha_i > |S_i|$ do
i. Set $L_{min} = \infty$ and $pc = null$
ii. for each $p_k \in P\ but\ not\ in\ S_i$
1.if $L_{min} > L(v_i,p_k,S_i)$ & $CT > EFT(v_i,p_k,S_i)$ then
Set $pc = p_k$ and $L_{min} = L(v_i,p_k,S_i)$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

End

2. If pc=null then

$$S_i = S_i \cup \{p_k\}$$
$$CT = \max_{p_x \in S_i} EFT\{v_i, p_k, S_i\}$$

Else
Break the while loop

VII. RESULTS AND COMPARISION

In order to compare these two algorithms, various comparison metrics are used. The most widely used metrics are schedule length ratio, speedup and reliability probability.

i. Schedule length ratio[1]: The task set are with different properties, hence it's a necessity that each graph is normalized to its schedule length. It is calculated by

$$SLR = \frac{makespan}{\sum_{v_i \in CP_{min}} \min_{p_j \in P} t_c(v_i, p_j)}$$

Now, this value cannot be negative, as denominator is the lower bound. Lower value of *SLR* implies the better algorithms.

ii. Speedup[1]: It is the ratio of sequential execution to the parallel execution. A higher value of speedup implies more amount of code is parallelized and vice versa.

$$Speedup = \frac{min_{p_j \in P}\{\sum_{v_i \in V} T(v_i, p_j)\}}{makespan}$$

Where,

$$T(v_i, p_j) = t_p(v_i, p_j) + \frac{1 - e^{-\lambda_j t_p(v_i, p_j)}}{e^{-\lambda_j t_p(v_i, p_j)}} \left(\frac{t_p(v_i, p_j)}{2} + \theta_p \right)$$

iii. Reliability Probability[1]: It is the probability that the task scheduled using a specific algorithm is reliable. A better reliable code has a higher reliability probability.

In order to test the two algorithms mentioned above, 100 random task graphs were created and were tested, so as to get more accurate results.

The fig shows a comparison between two algorithms for varying number of tasks. In order to obtain the graphs of the comparison ratios the number of tasks are varied as 40, 41and 42, shape factor is kept as 1, CCR value is kept as 0.1 and heterogeneity factor as 0.5. Using these values, following graphs are obtained

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

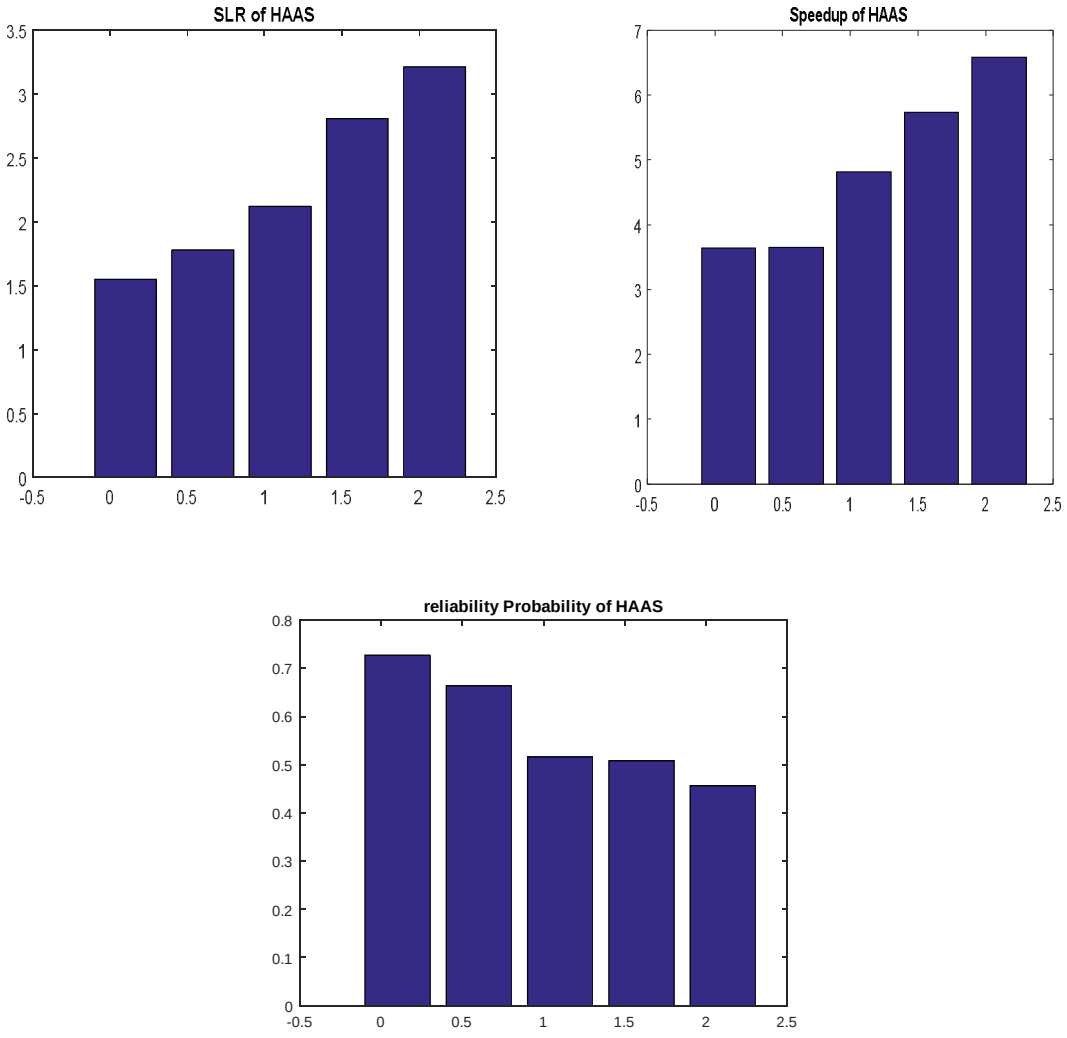

Fig 3 Results of varying tasks for HAAS

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

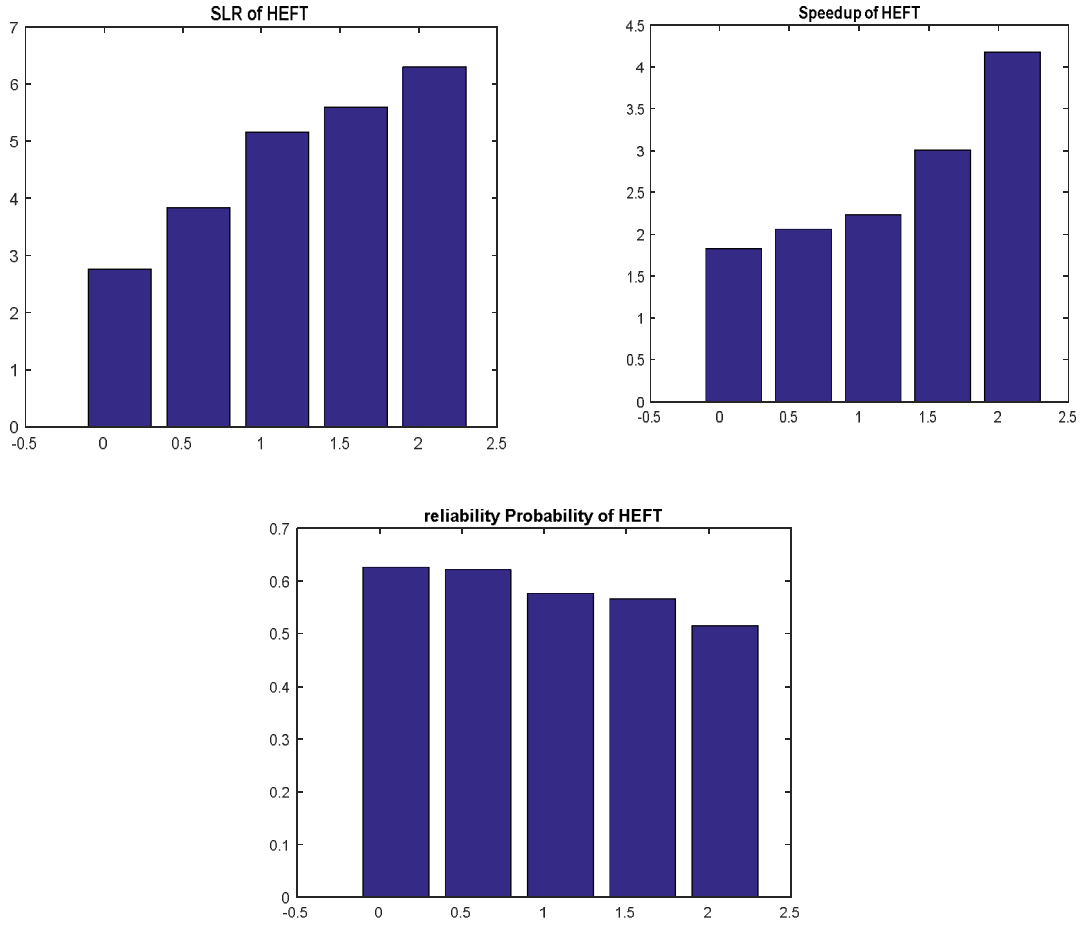

Fig 4 Results of varying tasks for HEFT

From the graph it is evident that the code is more parallelized in HEFT as compared to HAAS. This happens as the tasks in HEFT are not duplication giving way to more parallelism whereas in HAAS algorithm, in order to increase the reliability of the tasks the tasks are duplicated thus the makespan of the tasks increases and hence low parallelism. The reliability probability of tasks is more for HAAS algorithm than HEFT algorithm. As the number of tasks increases, the reliability probability decreases. This happens because the processors will be active for more amount of time if the task are increases thus increasing the probability of failure of tasks and hence the reliability probability decreases with the increase of tasks.

In order to obtain the graphs of the comparison ratios the value of CCR is varied as 0.1 to 2.1 in factor of 0.5, shape factor is kept as 1, heterogeneity factor as 0.5 and number of tasks is set as 40. Using these values, following graphs are obtained.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

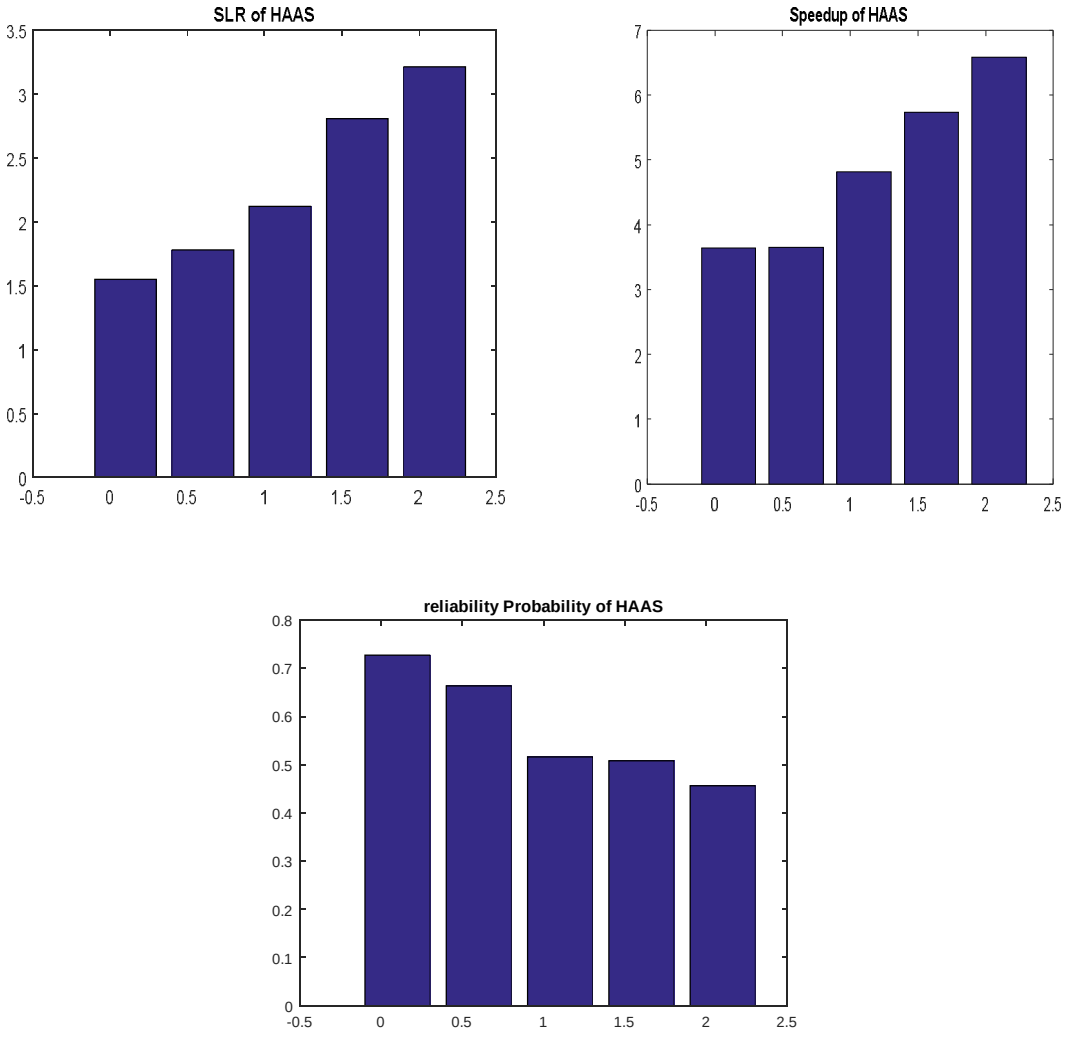

Fig 5. Results of varying CCR for HAAS

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

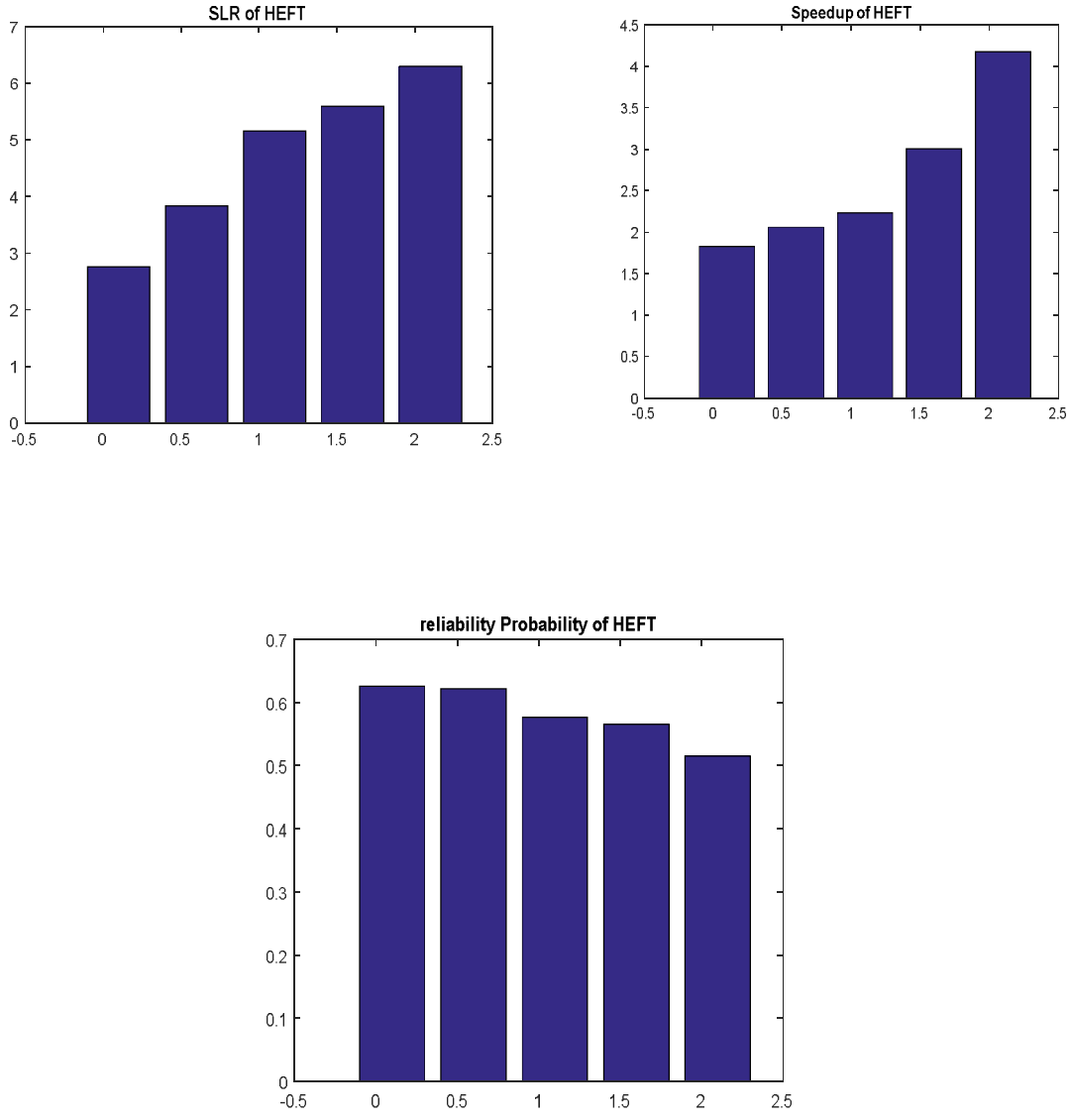

Fig6. Results of varying CCR for HEFT

The value of CCR is the communication cost divided by the computation cost which implies that it the ratio of the data required for communicating to the instructions to be executed. Increasing the CCR means the communication cost is increased and computation cost is decreased thus making the task as communication intensive. As the CCR value increases the code become more parallelized as less time is utilized in implementing the task as compared to communicating and hence the speedup increases. Also as the CCR increases the reliability probability also keeps on decreasing as many processors are running parallel thereby increasing the chances of processor failure.

VII.CONCLUSION

The HAAS Algorithm is highly efficient in the terms of reliability but however increases the makespan i.e. the completion time. On the contrary, the HEFT algorithm has low reliability and very less makespan. Thus a set of tasks

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijareeie.com

Vol. 6, Issue 11, November 2017

scheduled using HEFT will complete faster but there is high chance that the processor or link might fail thus leading to all the remaining tasks waiting until the failed processor or link recovers.

REFERENCES

[1] Chi-Yeh Chen,"Task Scheduling for Maximizing Performance and Reliability Considering Fault Recovery in Heterogeneous Distributed Systems,"IEEE Trans. on Parallel and Distributed systems, Vol.27, No.2,February 2016.
[2] Topcuoglu,H.Hariri, S.Wu, M.y,"Performance-Effecti've and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel and Distributed System, 13(3), pp: 260-274,2002.
[3] S. Shatz, J.P.Wang, "Models and Algorithms for reliability-oriented Task-allocation in redundant distributed-computer system" IEEE Trans. Rel., vol. 38, no. 1, pp. 16–27, Apr. 1989.
[4] Klaus Jansen,Hu Zhang," Scheduling Malleable Task with Precedence Constraints," SPAA'2005, July 18-20, Las Vegas, Nevada , USA.
[5] I. Assayad, A. Girault, and H. Kalla,"A bi-criteria scheduling heuristic for distributed embedded systems under reliability and realtime constrainst," in Proc. Int. Conf Dependable Syst. Netw. June 2004, pp.347-356
[6] S. Guo, H.-Z. Huang, Z. Wang, and M. Xie,"Grid service reliabilty modeling and optimal task scheduling considering fault recovery,"IEEE Trans. Rel., vol.60, no.1, pp. 263-274, Mar 2011
[7] R. Eswari and S. Nickolas," A Level-Wise Priority based Task scheduling for Heterogeneous Systems," International Journal of Information and Education Technolagy," Vol. 1, No. 5, Dec.2011.
[8]] T. Yang and A. Gerasoulis, "DSC: Scheduling parallel tasks on an unbounded number of processors," IEEE Trans. Parallel Distrib.Syst., vol.5, no 9, pp. 951–967, Sep. 1994.
[9] H. El-Rewini and T. Lewis, "Scheduling parallel program tasks onto arbitrary target machines," J. Parallel Distrib. Comput., vol. 9,no. 2, pp. 138–153, 1990
[10] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristi for interconnection-constrained heterogeneous processor architectures," IEEE Trans.Parallel Distrib. Syst., vol. 4, no. 2, pp. 175–187, Feb. 1993.