A PROJECT REPORT
ON

RELIABLE AND ENERGY EFFICIENT TASK SCHEDULING IN CLOUD COMPUTING

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Information Technology

BY

Saurabh Nath Pandey 20178044

UNDER THE GUIDANCE OF
Dr. ASHISH KUMAR MAURYA

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY ALLAHABAD
PRAYAGRAJ, 211004

Undertaking

I declare that the work presented in this report titled ”Reliable and Energy Efficient Task Scheduling in Cloud Computing”, submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, for the award of the Bachelor of Technology degree in Information Technology, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

May, 2021
Prayagraj

(Saurabh Nath Pandey 20178044)

Certificate

Certified that the work contained in the report titled ”Reliable and Energy Efficient Task Scheduling in Cloud Computing”, by Saurabh Nath Pandey has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Ashish Kumar Maurya
Computer Science and Engineering Dept.
M.N.N.I.T Allahabad, Prayagraj

May 2021,
Prayagraj

Preface

As a part of the course curriculum and to deepen and widen practical knowledge in the concept of Task Scheduling in Cloud Computing students are required to make project report on Task Scheduling in Cloud Computing.

Completing the project helped us know more about the concept Task Scheduling in Cloud Computing. Working with other team members taught us the importance of cooperation, coordination, and synergy. I hope you will find our project report interesting. All constructive criticism and feedback is cordially invited.

Saurabh Nath Pandey

Acknowledgements

We would like to express our special thanks of gratitude to **Dr. Ashish Kumar Maurya** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion and his invaluable guidance that supported us in completing this project..

We must express our sincere heartfelt gratitude to all the staff members of Computer Science and Engineering Department who helped us directly or indirectly during this course of work.

Saurabh Nath Pandey

Contents

List of Figures

Chapter 1

Introduction

Cloud computing is Internet-connected mode of super computing. It is a type of shared infrastructure, which simply puts the huge system pools together by using various means. It gives users a variety of storage, networking and computing resources in the cloud computing environment via Internet, users put a lot of information and access a lot of computing power on their own computer with the help of cloud computing. Cloud consists of numerous resources which are heterogeneous in nature i.e. different from each other in one or more aspects.This makes scheduling of task in cloud computing different and more challenging from traditional scheduling methods.

Although the cloud makes processing quick, simple, and dynamic, still it is confronting various testing issues which should be examined. Cloud server needs a lot of energy. As indicated by Moore's law,the quantity of semiconductors in a thick coordinated circuit expands twofold in 2 year intervals. This expanding number of semiconductors brings about an increment in power utilization which creates a lot of heat in servers.This leads to an increment in the cooling costs.Different other problems due to high energy utilization cause failure in system's execution, dependability issues, and deplorable effects on the climate. In this way, decreasing power utilization is consistently one major area on which research focuses at this point of

time. Cloud computing is also employed to perform several operations which are time critical in nature such as banking , IoT and in upcoming future it might also be used in self driving vehicles. All such things are very time critical and the reliability required to perform them needs to be at a very significant level. So , while keeping in mind the reduction in energy consumption , we also need to focus on maximizing the reliability of our system. But, both of these factors i.e. energy minimization and reliability maximization are conflicting in nature which has been explained in the later part of this paper. So, it becomes quite challenging to achieve both of these goals simultaneously.

In past few years many scheduling algorithms have been proposed for the effective management and execution of a business applications in the form of a workflow which is a NP Hard problem. To do it in a polynomial time several algorithms like HEFT, r-HEFT, min–min, critical-path-on-a-processor(CPOP) have been proposed which aim for reducing the schedule length(execution time).Dynamic Voltage Frequency Scaling (DVFS) is also considered as one of the most effective ways to for minimization of energy consumption , specially when we are dealing with the workflows. DVFS approach minimizes energy consumption by reducing the voltage and frequency but it works only for the homogeneous clusters.But even after all this there are still a lot of things which have been left unexplored for making our system more energy efficient.Dogan and Ozguner work helped in enhancing the system's reliability and minmimizing the schedule length simultaneously.They proposed a bi-norms heuristic which is called RDLS[11] to achive this . Tang et al. proposed the HRDS heuristic in a grid type environment. Additional, checkpointing and replication are urgent strategies to improve the system's availibility and its reliability.

Nonetheless, the algorithm brings about a growth in the response repetition where excess resources are utilized to store back ups (if there should be an occurrence of check-pointing) or to run copies (in the event of replication). As an outcome, energy utilization is sped up too. As we have seen in DVFS we were reducing the frequency to achieve the energy efficieny but we should also consider that by reducing the frequency we are reducing our system's reliability as well which we don't want to do. The trade-off between these two factors brings new difficulties in cloud designs, and there's a solid need for research in the area of streamlining the association among energy utilisation and reliability. Some research work [5,6] focussed on both these isssues concurrently but they are limited to the embedded systems only. The work mentioned in [5] provides a reactive fault tolerance to our system by using an adaptive check-pointing but it also leads to increase in the overall energy utilisation of the system.Thereafter, Lee and Zomaya [7] outfitted two energy efficient work-flow scheduling techniques by utilizing DVFS to offer the tradeoff between the energy utilisation and scheduling quality. But, none of them contemplated the reliability and energy efficiency together. So,our top concern is to do a research that considers both of these two factors simultaneously

So,a new scheduling algorithm named REEWS(reliability and energy-efficient workflow scheduling) was proposed for heterogeneous clouds with DVFS. REEWS intended to preserve dependency of workflow, along with reliability maximization and reduction in power utilisation while not doing any harm to the user specified deadline or Quality-of-Service (Qos). Dynamic and static energy are the two components which constitute the the total power consumed in the given system.Dynamic power is frequency dependent and the static one remains constant when device is in running mode. For the sake of measuring

reliability we use the concept of MTTF (mean time to failure). As we know that cloud infrastructure is itself very vast and complex.So there will definitely be some faults or issues will be there in it. Some of them could be permanent while others could be transient.Since we are living in the age of advanced hardwares so for us temporary or transient faults are the major challenges and therefore REEWS focuses on such kind of faults.

In this work I have tried to do a certain amount of improvement to the existing REEWS algorithm to make it some more energy efficient.

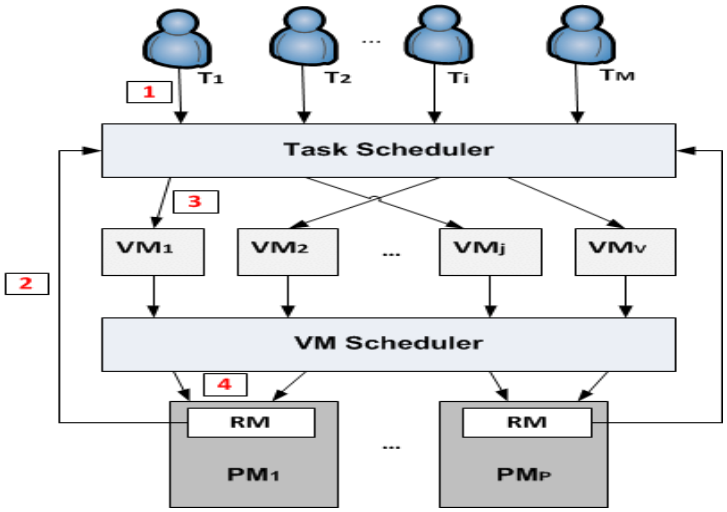

Figure 1.1: General View of Task Scheduling

Chapter 2

Related Work

Hitherto, various researches have been conducted to fathom the issue related with the scheduling of workflows thinking about the streamlining targets of deadline, reliability and energy utilization. One of the fundamental methodologies for example dynamic segment deactivation[8] was given to lessen the energy utilization in enormous grids. Thought was to process the general heap of system and turning off inactive assets. Then, to maintain a strategic distance from on/off cycles, assets are held ahead of time for upcoming needs. Wang et al.[10] gave a methodology to schedule priority constraint/ interdependent processes in a homogeneous cluster to reduce power utilization using the dynamic voltage frequency scaling strategy. Two algorithms were being given by them: the one takes into account the slack circulation of non-basic tasks along with not expanding the makespan, next one takes into consideration the green help level consent in order to lessen power utilization with increment in the scheduled length inside breaking point.

Another group of research works thought about the improvement of systems reliability for the issues related to scheduling. Faragardi et al[3]. proposed a method for analysing the reliability using an analytical model that focuses on the system's resource allocation. This approach was having two constraints. Application constraints were

dealt with Quality-of-Service and through errand priority structure . One more constraint is the requirement of resources which restricts the capacity utilization along-with the memory. Also, the impact of organization's topology on the system reliability was thought of. An alternate methodology utilizing clusters was given where jobs are being first admitted after which they go through the famous Poisson's distribution of failure. Here, tasks are acknowledged by system at the moment when cutoff time is ensured;otherwise ongoing tasks are dismissed. The acknowledged tasks go through a dynamic booking heuristic where a proportion of reliability is taken into the account.

Even as of now there aren't many research works which targets both i.e reliability maximization and reduction in energy utilisation simultaneously. Because of the developing interest, it is advantageous to investigate the reasonable methodologies for this. So, the REEWS was proposed. It reduces the communication time with the help of clustering, then it considers the used specified deadline and distributes it among all the tasks by assigning them their target time for completion, thereafter assigning each cluster to that PM which ensures highest reliability and least energy utilization.

In this work an improvement is being made in the clustering approach of REEWS that ensures more savings in the communication cost than the clustering which was used previously.All this is done without putting the reliability of the system at stake and thereby maintaining the system's reliability as well.

Chapter 3

REEWS

3.1 Mathematical Models

Here,we broadly discuss four cloud models viz. power,system,application and reliability models.The processing elements are represented by a model.The process applications must be optimised, as well as the goals to be achieved.

3.1.1 System Model

The current study uses a cloud computing system model that consists of m physical machines which are heterogeneous . Also, we assume k number of frequency levels (f1, f2,..., fk) and voltages corresponding to them for each processing element(i.e. PMs), which means that it works at various voltages and frquencies.Furthermore, there is a cost associated with switching.The frequency of transitions between levels is believed to be insignificant (10–150 μ m).

A communication link connects each processing element to the others in order to keep the problem from becoming too complicated. The inter-processor communication takes via. a reliable communication sub-system, and communication between processors is believed to be the same.

3.1.2 Application Model

For representing a parallel application with is bounded by priority that means a worksflow, we take the help of an acyclic graph with directed edges (DAG) denoted by symbol G(T, E). where set of n tasks are being represented by T and edges between the tasks are being represented by E between pair of tasks τ_i to τ_j where τ_i is immediate predecessor of τ_j or we can say it is the τ_j's parent and τ_j is immediate successor of τ_i or we can say it as τ_i's child. Edge weight w(eij) between tasks denotes the information (in MegaBytes) being communicated from τ_i to τ_j when they executed on two distinct PMs. The computation cost of each task is represented through weight w(τ_i), i.e. the total amount of instruction (in millions) that needs to be performed for that individual task.

A task with no parent is called an entry task and the one having no child is called as exit task. The execution time (et_i) can be known from the computation cost (w(τ_i)) i.e. $et_i = (w(\tau_i) * CPI)/f_{r,max}$ where CPI stands for cycles per instruction and $f_{r,max}$ denotes maximum frequency of a PE during execution of τ_i. Among two precedence constraints, the communication time $(ct_{i,j})$ is the most important and it is computed as $ct_{i,j} = w(i,j)/bw(pm_r, pm_s)$ where bw(pm_r, pm_s) denotes the communication channel's bandwidth (in Megabits per second). If they run on the same physical machine, communication time is ignored .The other attributes are :

1.Estimated Start Time(est_i) :

$$est_i = \{ 0 \qquad if \tau_i \varepsilon entrytask$$
$$= max_{\tau_p \varepsilon predecessor of \tau_i}(est_p + et_p + ct_{pi}) Otherwise.....(3.1)$$

2.Estimated Finish Time($\mathbf{eft}_i$) :

$\boldsymbol{eft}_i = est_i + et_i$.....(3.2)

3.1.3 Power Model

Memory , networking interface , CPU and disk storage are major contributors for the power consumption in cloud data-center. As compared to different other sources, CPU utilises the primary part of the power.Thus in order to cover all the various sources of energy consumption in system the subsequent power model is being adopted and the equation governing the power consumption is given by,

$$P = P_s + h(P_{ind} + C_{eff} * V^2 * f) \tag{3.3}$$

where dynamic power which depends on frequency is denoted by P_d,static power consumption is denoted by P_s , is and frequency independent power is denoted by P_{ind} .h decides the system's state. If h = 1, the system is active; else it is in sleep mode.

The static power P_s is an inevitable element, that's utilised while the time device is on. It is used for holding the memory ,running the clock and for the maintenance of the basic circuitry while the device is in sleep mode.P_{ind} is the regular power consumption by the machine however it may be decreased to a minimum cost by retaining the machine in standby mode.P_{ind} indicates that it doesn't depends on the frequency and is utilised for accessing the peripheral machines such as input output machines and different outside machines (primary memory, disk storage etc.). Also it remains fix for the jobs that require a lot of computing. Therefore, we have to deal only with the most presid-

ing element which is the dynamic power (P_d) or the power consumed by the processor , that's dynamically modified in step with working frequencies,

$$P_d = C_{eff} * V^2 * f) = C_{eff} * f^3 \tag{3.4}$$

where C_{eff} represents effective load capacitance, V is the applied voltage and f represents clock frequency. The f is directly proportional to voltage.So from eq. 3.4 we conclude that dynamic power is frequency dependent and is minimum when frequency is equal to $f_{ee} = 3\sqrt{P_{ind}/C_{eff}}$.Thus for a task τ_i we compute energy using the following equation :

$$E_i(f_{r,op}) = (P_{ind} + C_{eff} * (f_{r,op})^3) * \frac{et_i}{f_{r,op}} \tag{3.5}$$

et_i is the execution time of the task τ_i at when the device is running at ite peak voltage/frequency.To compute the total energy consumed we sum up all the energy that is being utilised by each individual task:

$$E_{total} = \sum_{i=1}^{n} E_i(f_{r,op}) \tag{3.6}$$

3.1.4 Reliability Model

While executing an application, each permanent and temporary faults might also additionally arise because of software flaws,hardware crash, excessive heat generated by device, etc. However the chances a getting a permanent fault is very less than compared to the temporary one . Therefore, in REEWS temporary faults are taken care of.So in later section of this paper if we use the word fault then we will be referring to the temporary one. Basically, temporary faults comply with

the Poisson's distribution. The factor that needs to be centered on is that the rate by which the faults occur(λ).It relies upon on the running frequency of CPU.
The fault rate is given by:

$$\lambda(f_{r,op}) = \lambda_o * 10^{\frac{d(1-f_{r,op})}{1-f_{r,min}}} \tag{3.7}$$

where $f_{r,op}$ denotes the frequency on which device is currently operating, λ_o is initial fault rate and d is a constant. From the equations 3.5 and 3.7 we can infer that as frequency decreases energy consumption decreases and fault rate increases and vice versa.But, as the fault rate increases reliability decreases.So, reducing energy consumption and increasing reliability are clashing with each other. We define the systen's reliability as the probability of execution of a task without failure and is given by:

$$Rel_{\tau_i}(f_{r,op}) = e^{-\lambda(f_{r,op}*\frac{et_i}{f_{r,op}}} \tag{3.8}$$

for a group of n tasks the overall reliability is the product of all reliabilities of each task in the group.

$$Rel_G = \prod_{i=1}^{n} Rel_{\tau_i}(f_{r,op}) \tag{3.9}$$

3.2 Algorithm

REEWS is one of the latest algorithm for the workflows. Calculation for priority bound applications in the cloud atmosphere which boosts the system's reliability and limits power utilization. It comprises of four primary stages:

1. Priority Calculation : To locate a legitimate topological ordering of the work process so priority imperatives are fulfilled.Priority of

a task τ_i is given by :

$$pr_i = et_i + max_{\tau_{ji}}(ct_{i,j} + pr_j) \tag{3.10}$$

Algorithm1 : CalculatePriority (T)
1. Initialize priority of exit task i.e.
   pr[exit]$\leftarrow$ et[exit]
2. **for** each task τ_i in T in reverse order
3. Calculate the priority pr_i for each task as per equation (3.10)
4. **end for**
5. **endprocedure**

2.Clustering of Tasks : We perform a grouping of tasks in order to reduce the overall communication time which leads to the reduction in energy consumption. This happens because if we group the inter-dependent tasks together and execute them on same physical machine then the time of communication greatly reduces.This happens because there is a very low latency and high bandwidth in inter-processor communication as compared to the communication between the two distinct machines.

Algorithm2 : Clustering(T)
1. **Add** all tasks form T to the task list
2. k=0
3. **for** each task τ_i in task list
4. **if** τ_i has not been assigned to any cluster **then**
5. k=k+1;Make a new cluster cl_k
6. **label**:
7. Add task τ_i to the cluster cl_k
8. Sort the children of task τ_i
9. **for** each child τ_j of τ_i

10. **if** τ_j is unassigned and parent(τ_j) is assigned
11. $\tau_i \leftarrow \tau_j$ and **goto** label
12. **break**
13. **End if**
14. **End for**
15. **End if**
16. **End for**

3.Deadline Distribution : Distribution of user defined deadline as per the agreement of Quality of Service. If the deadline for a given task is more than the estimated finish time then in such cases we can extend our finish time.By doing this our processors get more time and so they can reduce their operating frequency which will eventually lead to decrease in energy consumption.In Algorithm 3 we achieve this.Algorithm assign the exit task a target-time which is equal to the deadline.To achieve this goal the concept of deciding path and deciding parent is used as follows:

Deciding Path:The path from τ_{entry} to the task τ_i for which the sum of edge weights is maximum is the deciding path of τ_i.

Deciding Parent:The parent τ_p of a task τ_i which is not assigned and which gives the maximum value for est(τ_p) + et_p + $ct_{p,i}$, , is the deciding parent of a task τ_j.
We use the concept of normalizing factor (N) in for the sake of uniform distribution of sub target time among the tasks on the path. Then we go for final execution time thereafter sub target time.Below given equations are used to calculate them:

Normalization Factor(N) :

$$N = \frac{(lft(\tau_{end}) - est(\tau_{start}) - pl(Path)}{\sum_{i \varepsilon Path} et_i} \tag{3.11}$$

Final Execution Time(fet) :

$$fet(\tau_i) = et_i + et_i * N \tag{3.12}$$

Sub Target Time

$$SubTarget(\tau_i) = est(\tau_i) + fet(\tau_i) \tag{3.13}$$

Algorithm 3 : Distribute_TargetTime(τ)

1. **while** τ has unallocated parent **do**
2. Path $\leftarrow$ null, $\tau_k \leftarrow \tau$
3. **while** τ_k has unallocated parent **do**
4. $Path \leftarrow deciding\_parentof\tau_k + Path$
5. $\tau_k \leftarrow deciding\_parentof\tau_k$
6. **end while**
7. **call** AllocateSubTargetTime(Path)
8. **for** each task τ_i in Path
9. $Allocate(\tau_i) \leftarrow true$
10. **end for**
11. **for** each task τ_i in Path
12. Update est_i where $\tau_j \varepsilon child(\tau_i)$
13. Update lft_j where $\tau_j \varepsilon parent(\tau_i)$
14. call DistributeTargetTime(n)
15. **end for**
16. **end while**
17. **end** Procedure

Algorithm 4 : Allocate_SubTargetTime(Path)

1. $\tau_{start} \leftarrow$ start task of Path
2. $\tau_{end} \leftarrow$ end task of Path
3. $pl \leftarrow 0$
4. **for** each task τ_i in Path
5. $\quad pl = p1 + et_i + ct_{i,j}$ where τ_j is successor of τ_i in Path
6. $\quad \tau_i \leftarrow \tau_j$
7. **end for**
8. Calculate normalizing factor according to equation (3.11)
9. **for** each task in Path
10. $\quad$ Calculate fet according to equation (3.12)
11. $\quad$ Calculate SubTarget according to equation (3.13)
12. **end for**
13. **end Procedure**

4.Assignment of Tasks : For the sake of minimizing the energy consumption and maximizing the relability we dool out the task to the heterogeneous processors at appropriate frequency and voltage levels so the general reliability of the system is greatest alongside minimization of energy utilization.

3.3 Scheduling Algorithm

Algorithm 5 : Scheduling

1. **for** each task τ_i in T
2. $\quad$ Calculate et_i and $ct_{i,j}$ on all processors
3. $\quad$ Calculate est_i, and eft_i according to equations in (3.1-3.2)
4. **end for**
5. **call** Calculate Priority(τ_{entry})

6. cl ← call Clustering(T)
7. SubTargetTime(τ_{exit}) ← D
8. Allocate(τ_{exit}) := true
9. **call** Distribute_TargetTime(τ_{exit})
10. **for** each cluster cl_i
11. **for** each processor/physical machine pm_r in PM
12. Calculate energy efficient frequency f_k on pm_r for τ_c (where τ_c ε cl_i) according to equation (3.5) such that it meets the TargetTime of cl_i
13. Calculate reliability of τ_c on pm_r (where τ_c ε cl_i) according to equation (3.8)
14. **end for**
15. Assign cl_i to processing element providing maximum reliability
16. **end for**

Chapter 4

Proposed Work

4.1 Improved Clustering

In IREEWS clustering is made more efficient in comparison to REEWS. Earlier in REEWS the child (immediate successor) was being added to the cluster of that parent which was its last parent that got allocated.But our aim was to assign it in such a way that we attain minimum communication cost which leads to less waiting time and thus gives us a reduction in overall energy consumption.So to do this we allocate child to the cluster of that parent which has maximum communication cost with child task.

Algorithm 7 : Cluster(T)
1. Add all entry tasks τ_i from T into list L1
2. **for** each task τ_i in taskList
3. assign[Task]←**false**
4. **end for**
5. k←0
6. **for** each task τ_i in L1
7. create a new cluster c
8. put τ_i in c
9. clusterId[c]← k++
10. assign[τ_i]← true

11.**end for**
12.**for** each cluster cl_i
13. x=first task of cl_i
14. RecurClust(x)
15.**end for**
26.**end procedure**

Algorithm 8 : RecurClust(Task x)
1. assign[x]$\leftarrow$ true
2. Obtain all child of x in TaskList L1
3. Sort L1 according to the priority of tasks
4. **for** each task τ_i in L1
5. **if** all parents of τ_i are assigned
6. $\tau_j \leftarrow$ parent of τ_i having maximum communication cost with τ_i
7. Put τ_i within the cluster of τ_j
8. clusterId[τ_i]$\leftarrow$ clusterId[τ_j]
9. **call** RecurClust(τ_i)
10. **end if**
11.**end for**
12.**end procedure**

In this clustering algorithm we recusively iterate through all the succerors of a task and thereafter we allocate those tasks to the cluster of their parent which is having maximum communication cost with it.

Consider below example :

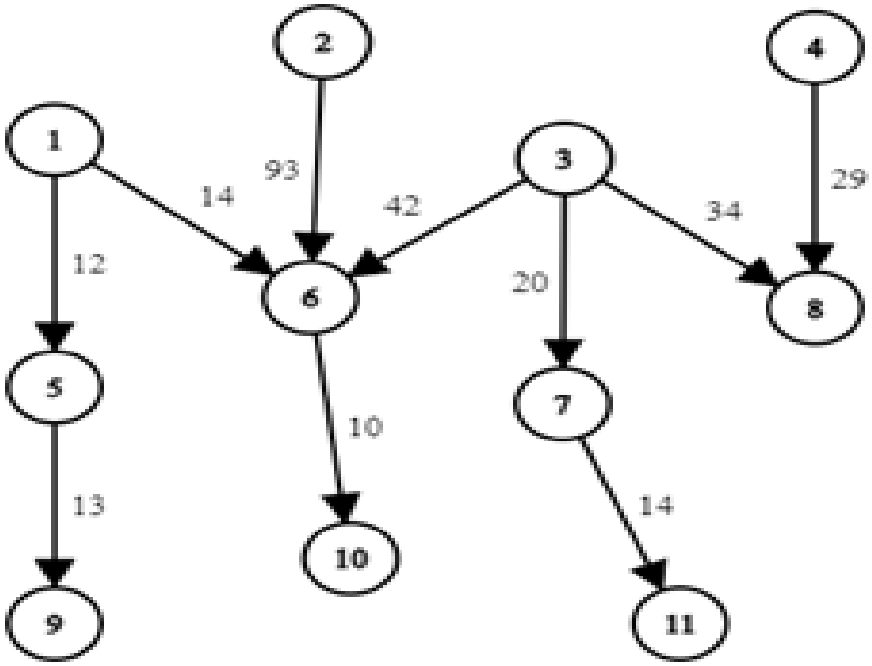

Figure 4.1: Task Graph

On using clustering approach given in REEWS algorithm we get the following clusers:

Cl_1	Cl_2	Cl_3	Cl_4
1	2	3	4
5		6	8
9		7	
		11	

Figure 4.2: REEWS Cluetering output

But,If we put task 2 and task 6 in same cluster it will be more efficient. If we put task 8 and task 3 in same cluster it will be more efficient. This is because 2 transfers more data to 6 as compared to

3 hence if we club 1 and 6 together it will reduce the communication time more significantly and thus reducing the overall execution time which reduces energy consumption. Same applies for next case too. Thus in order to achieve this we use improved clustering algorithm defined in IREEWS.On using that we get following clusters:

Cl_1	Cl_2	Cl_3	Cl_4
1	2	3	4
5	6	7	
9	10	11	
		8	

Figure 4.3: IREEWS Cluetering output

which solves the above problem.

Chapter 5

Experimental Setup

Since for testing our algorithm we need lot of workflows . So it becomes very difficult to generate these workflows manually . So , we have used workflows from peagasus workflow gallery which includes different types of workflows.

5.1 Types of workflows used :

We have used following types of workflows for our algorithm:
1.**Cybershake** :

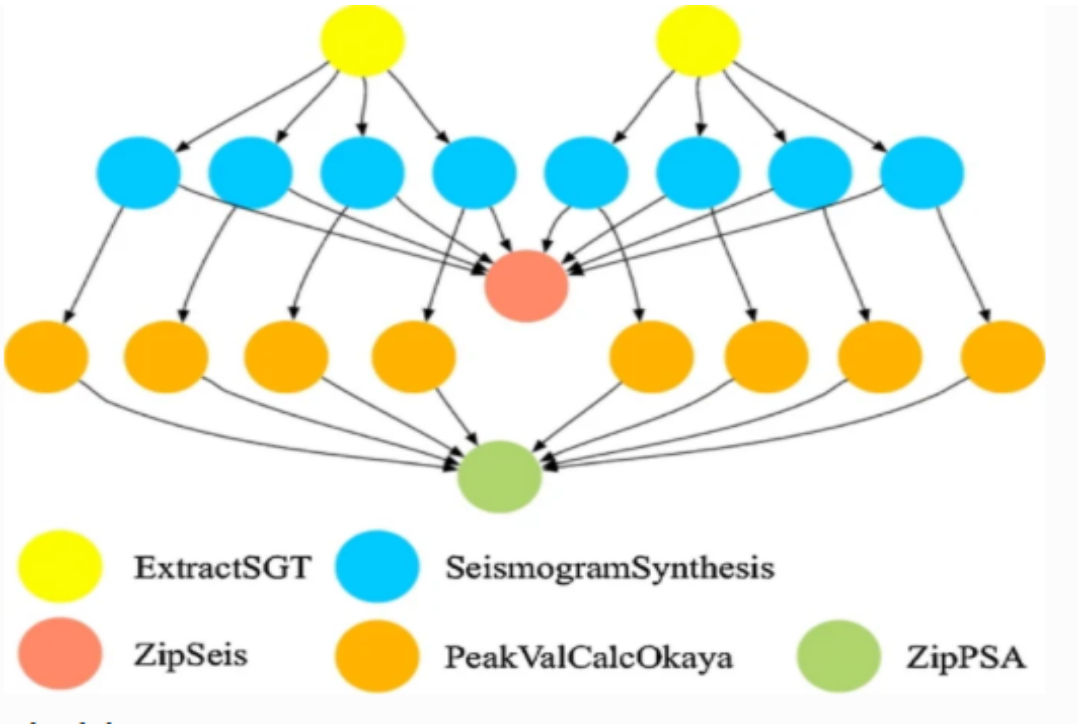

Figure 5.1: Cybershake Structure

2.**Montage**:

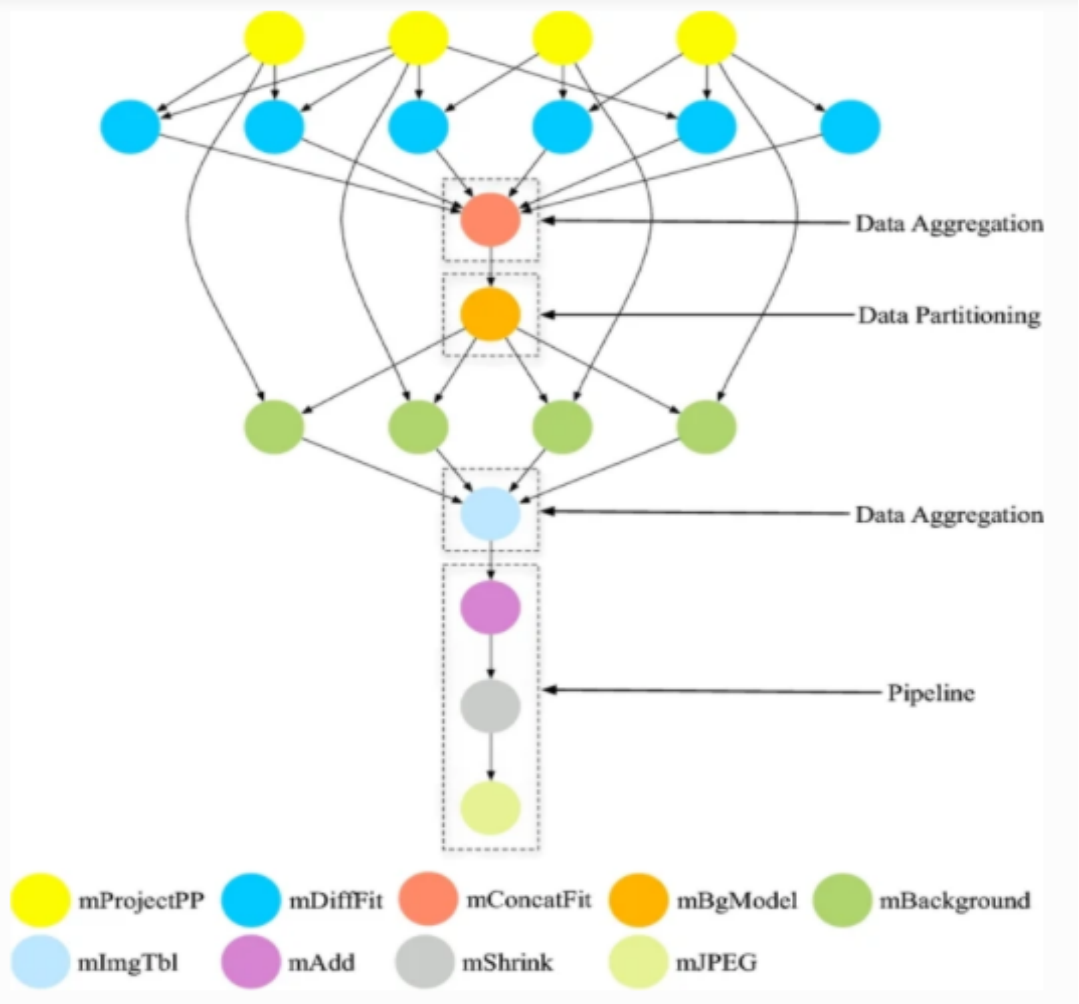

Figure 5.2: Montage Structure

3.**Genome**:

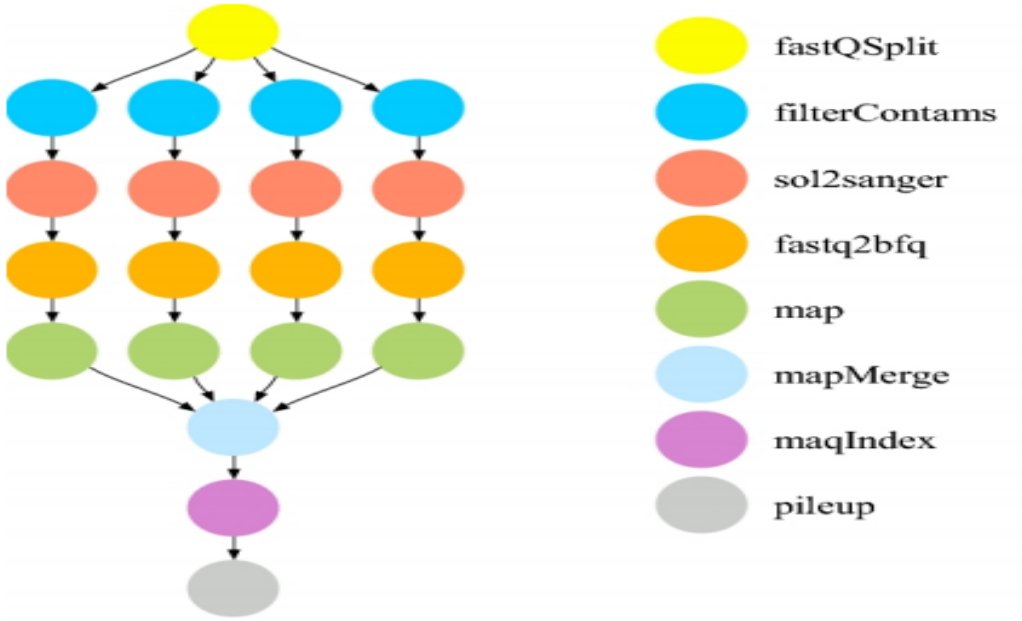

Figure 5.3: Genome Structure

4.**Sipht**:

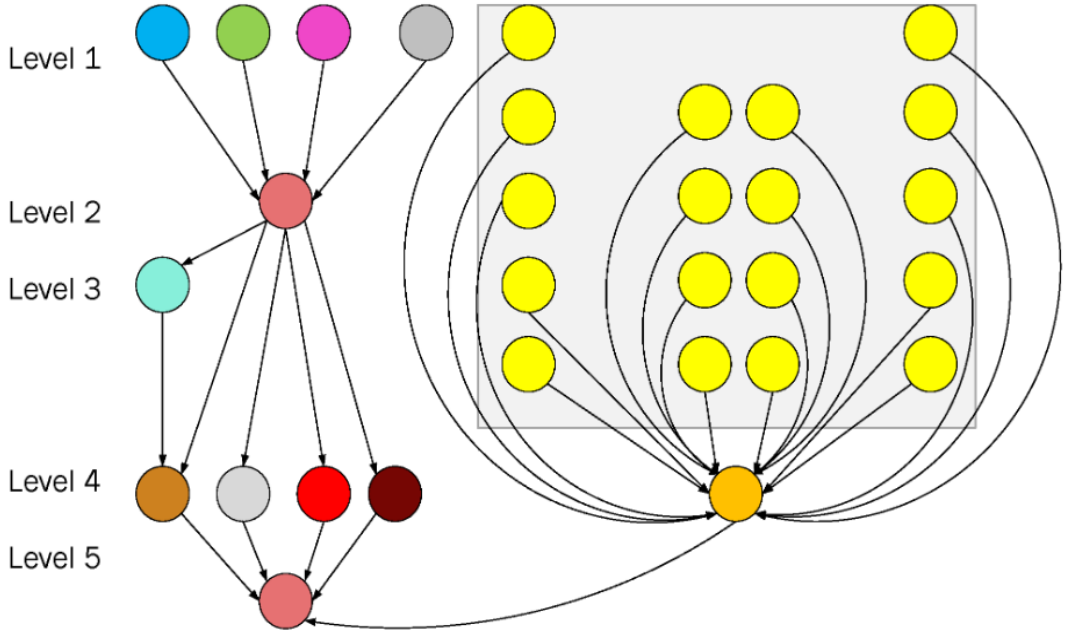

Figure 5.4: Sipht Structure

5.**Ligo**:

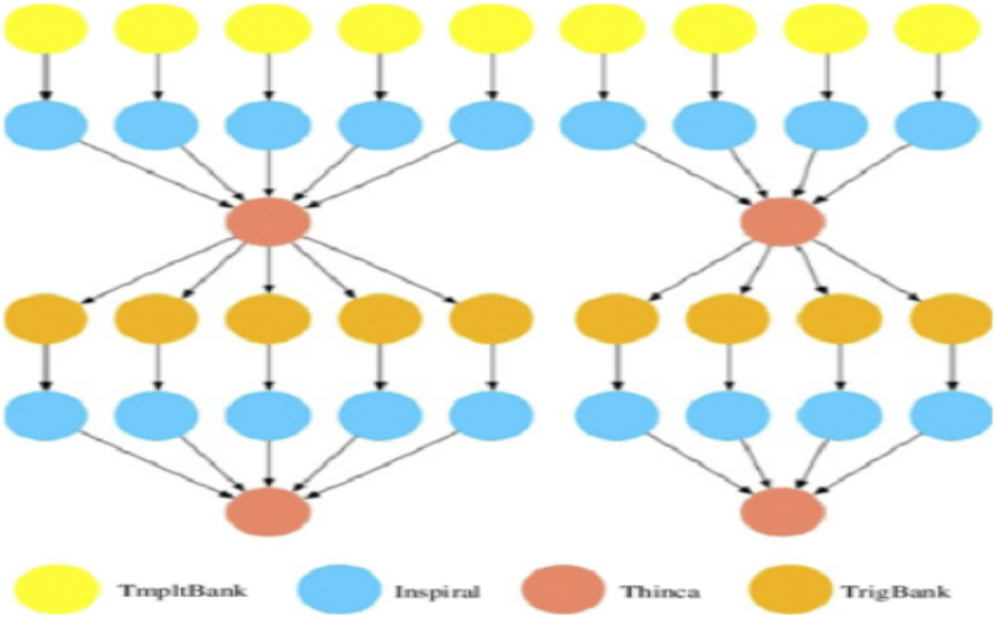

Figure 5.5: Ligo Structure

Chapter 6

Result Analysis

We get the following results :

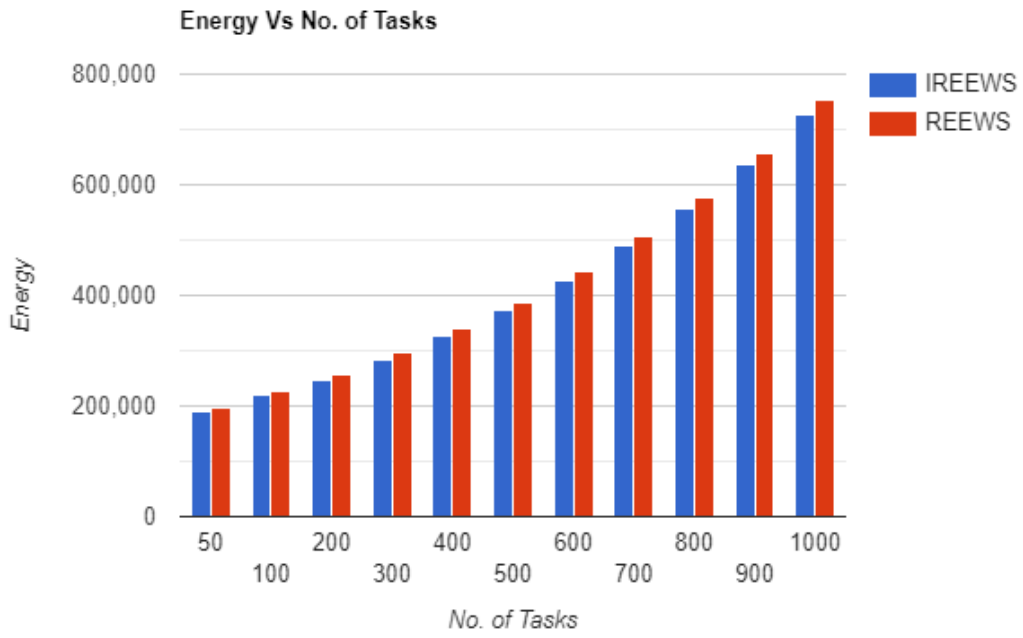

Figure 6.1: Energy Consumption Vs Number of Tasks in CYBERSHAKE

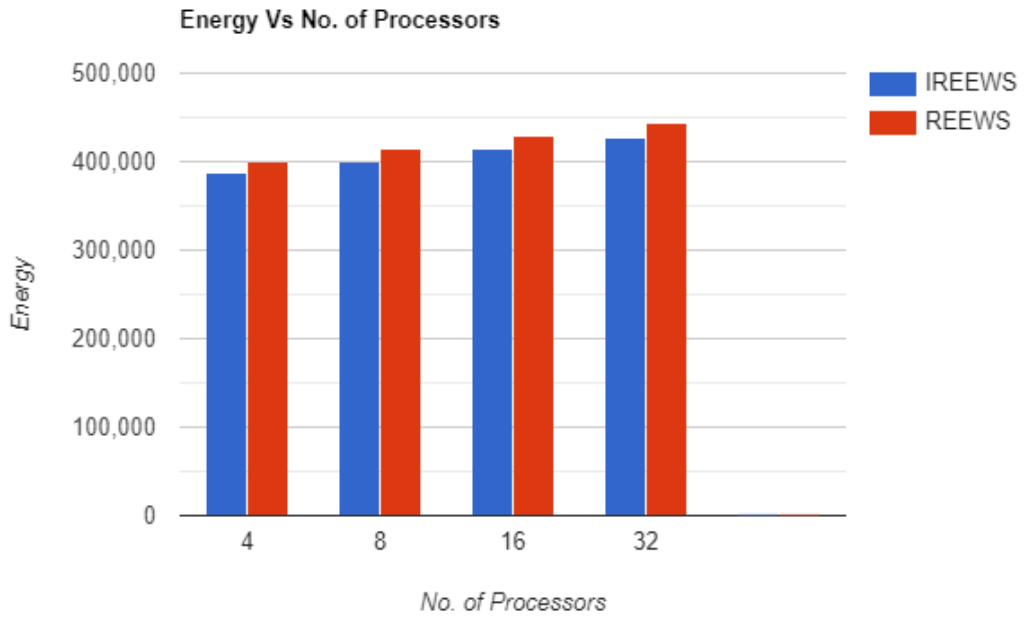

Figure 6.2: Energy Consumption Vs Number of Processors in CYBERSHAKE

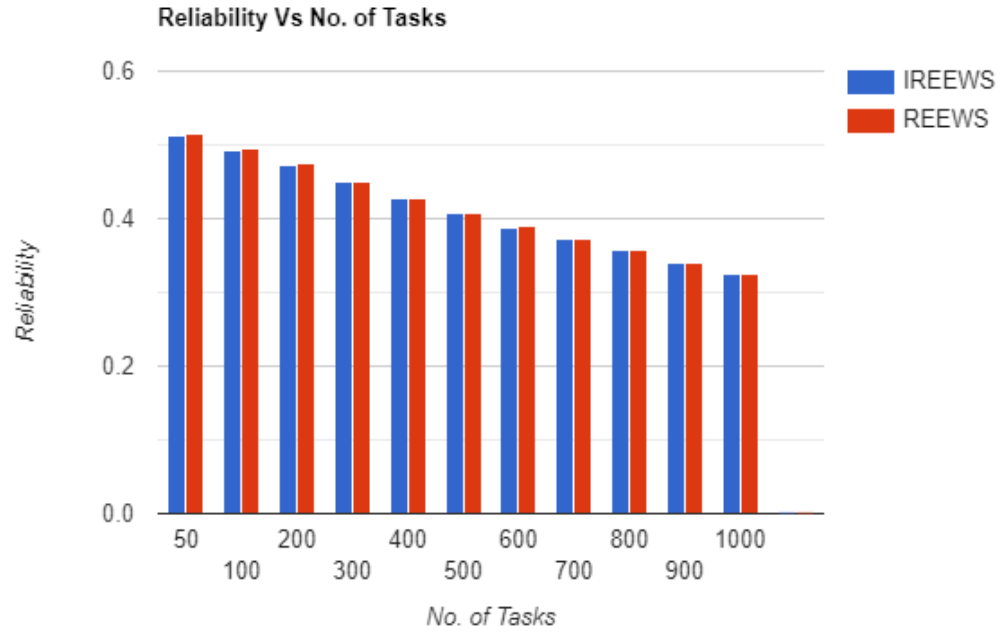

Figure 6.3: Reliability Vs No. of Tasks in CYBERSHAKE

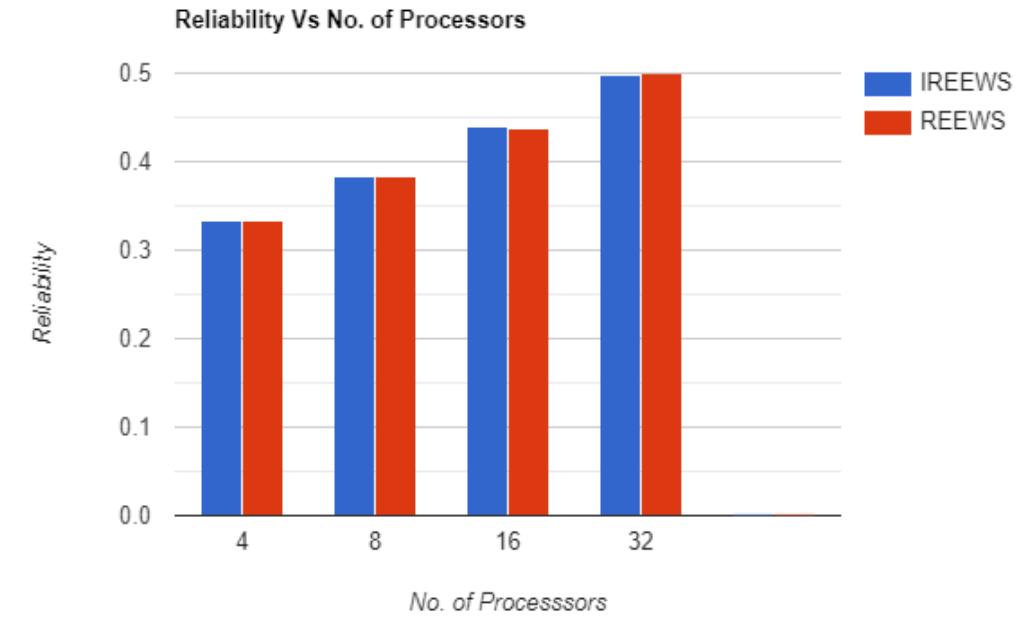

Figure 6.4: Reliability Vs Number of Processors in CYBERSHAKE

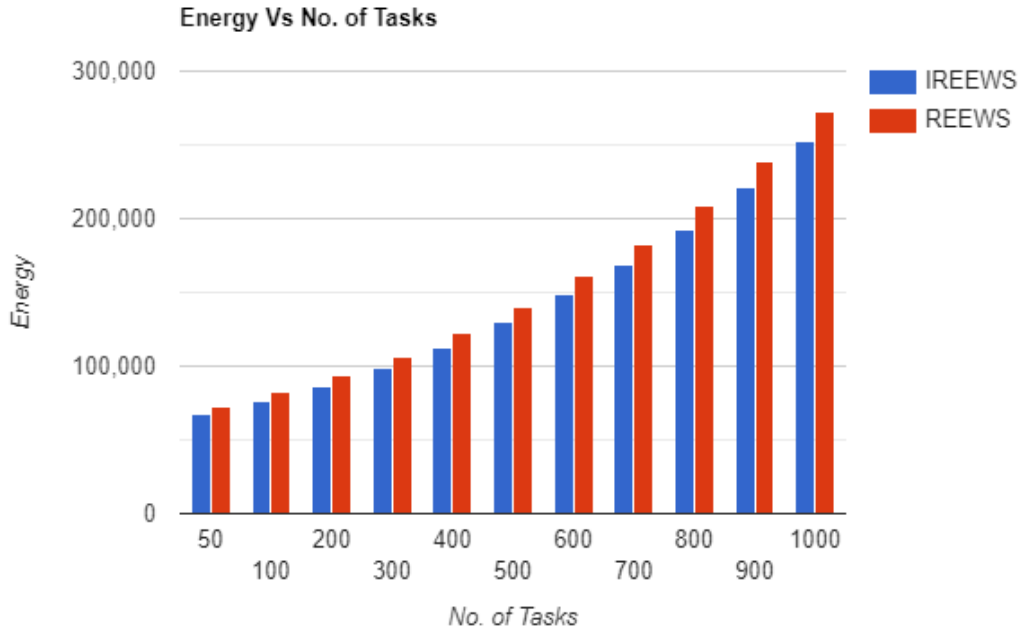

Figure 6.5: Energy Consumption Vs Number of Tasks in MONTAGE

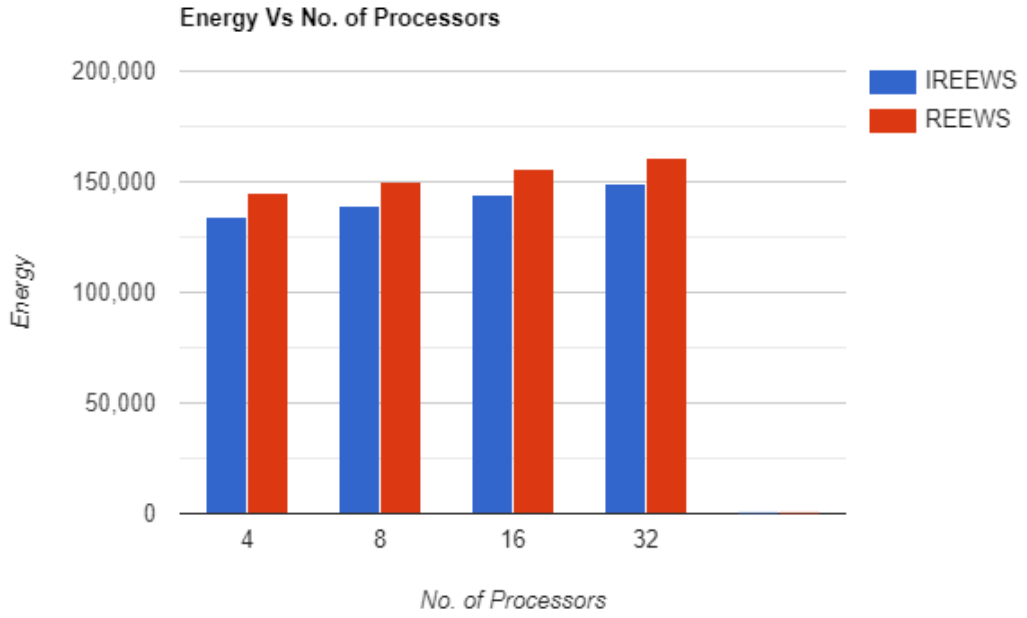

Figure 6.6: Energy Consumption Vs Number of Processorin MONTAGEs

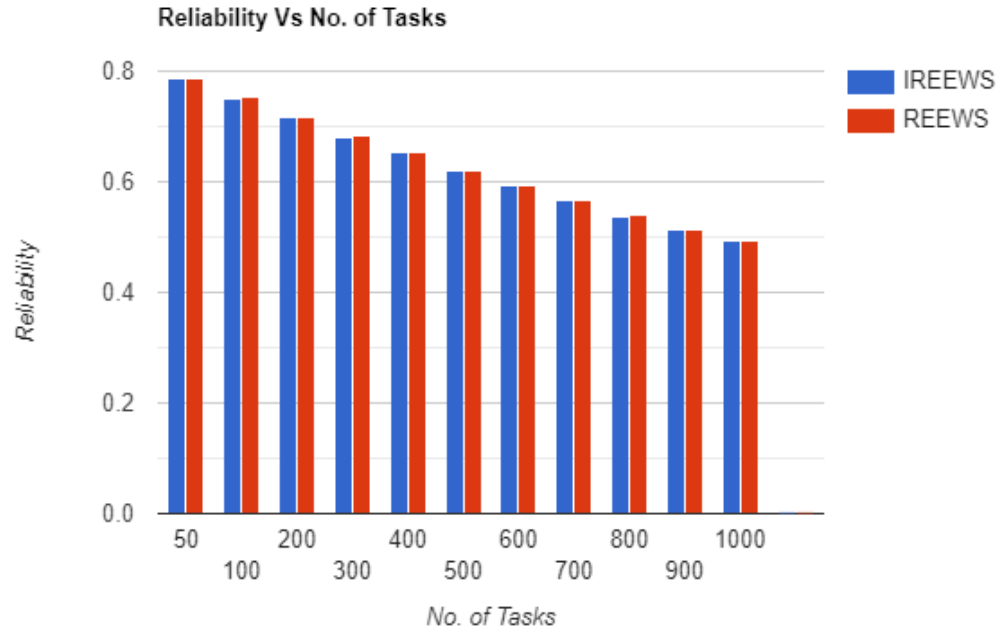

Figure 6.7: Reliability Vs No. of Tasksin MONTAGE

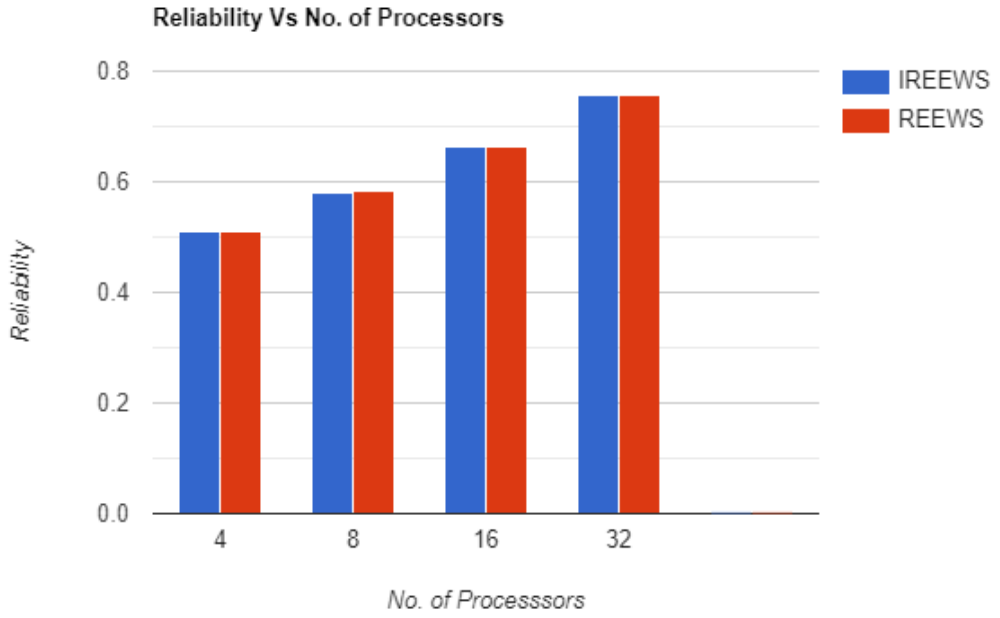

Figure 6.8: Reliability Vs Number of Processorsin MONTAGE

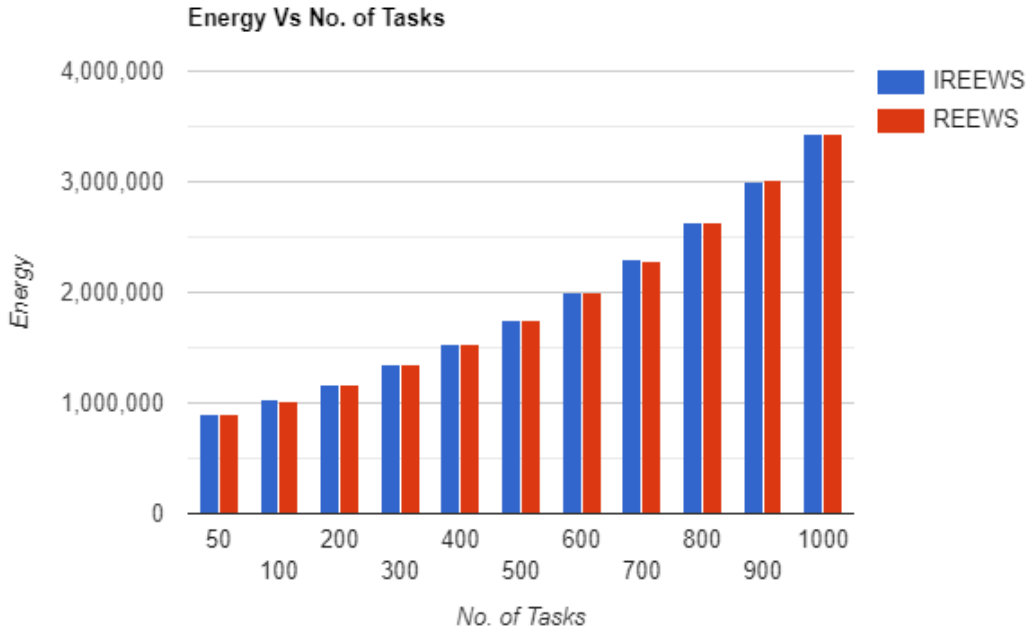

Figure 6.9: Energy Consumption Vs Number of Tasks in GENOME

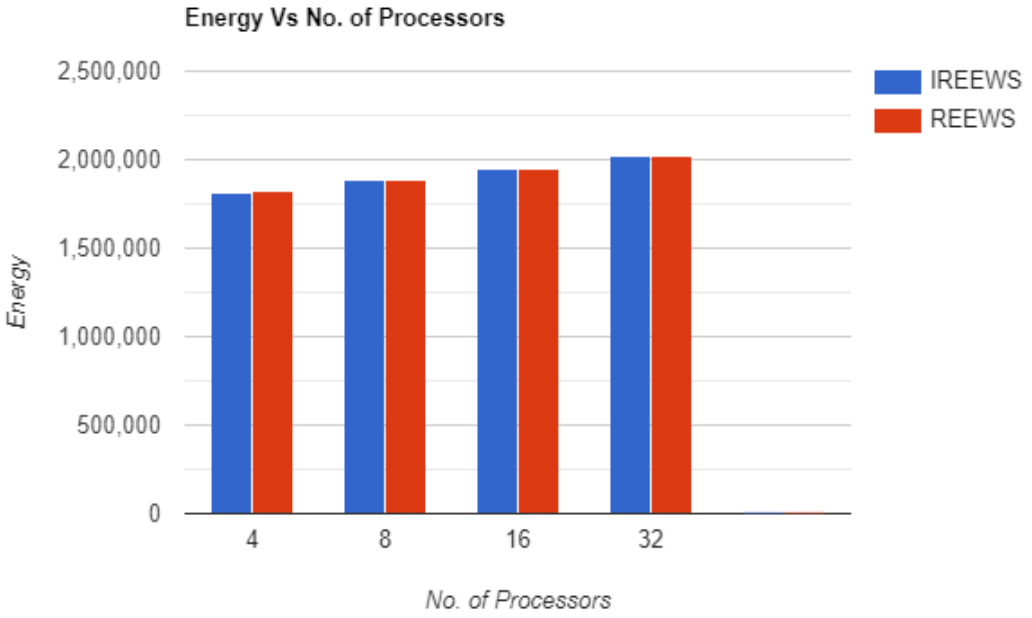

Figure 6.10: Energy Consumption Vs Number of Processors in GENOME

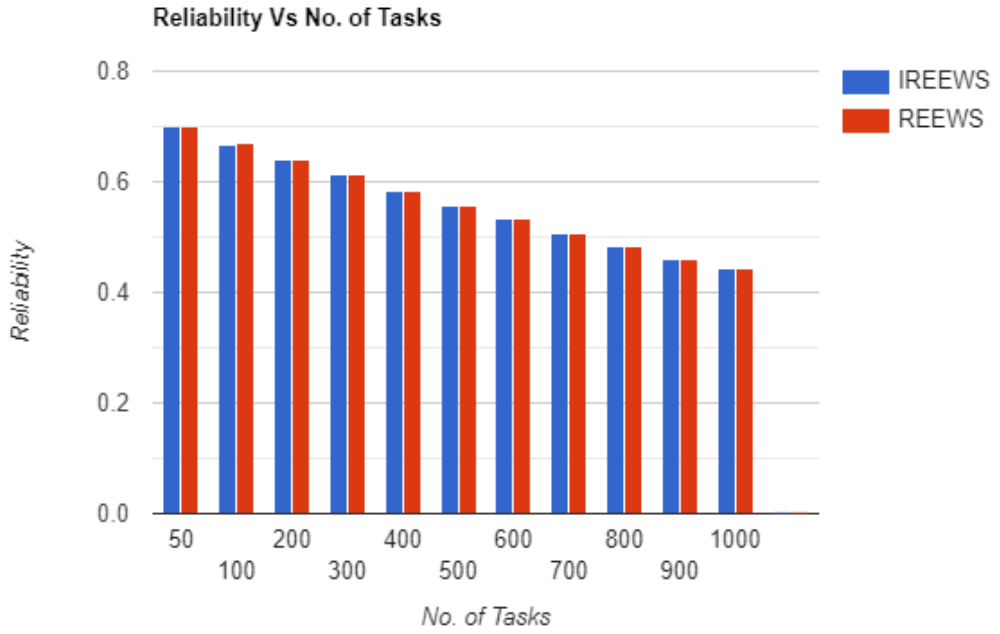

Figure 6.11: Reliability Vs No. of Tasks in GENOME

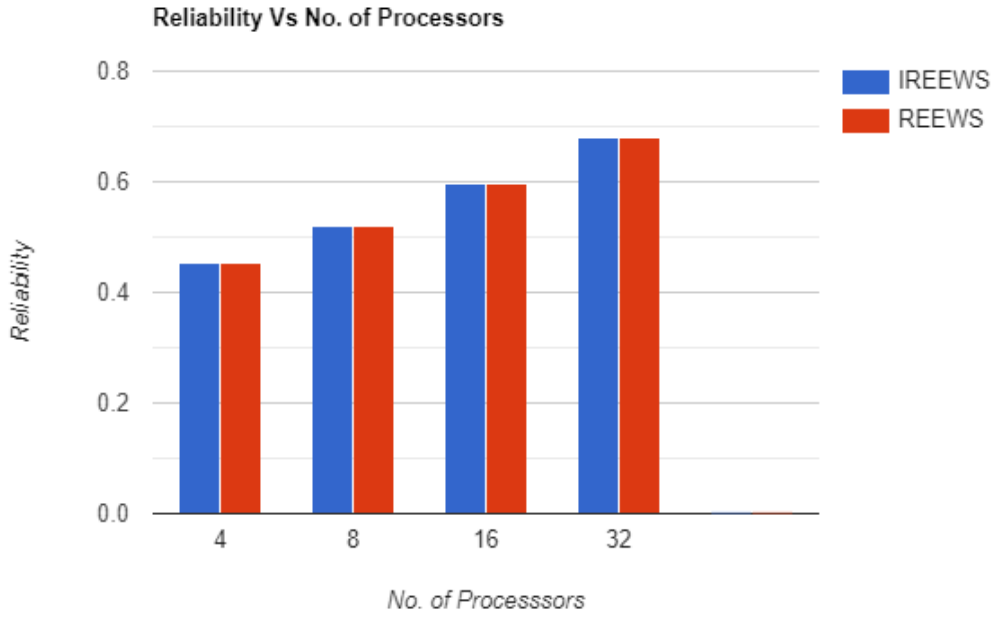

Figure 6.12: Reliability Vs Number of Processors in GENOME

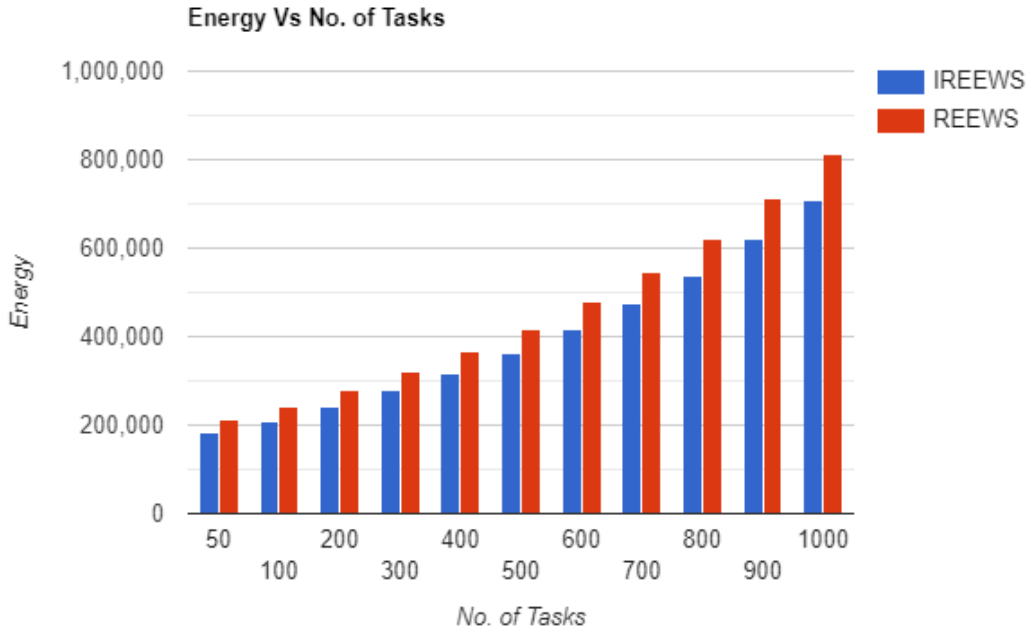

Figure 6.13: Energy Consumption Vs Number of Tasks in SIPHT

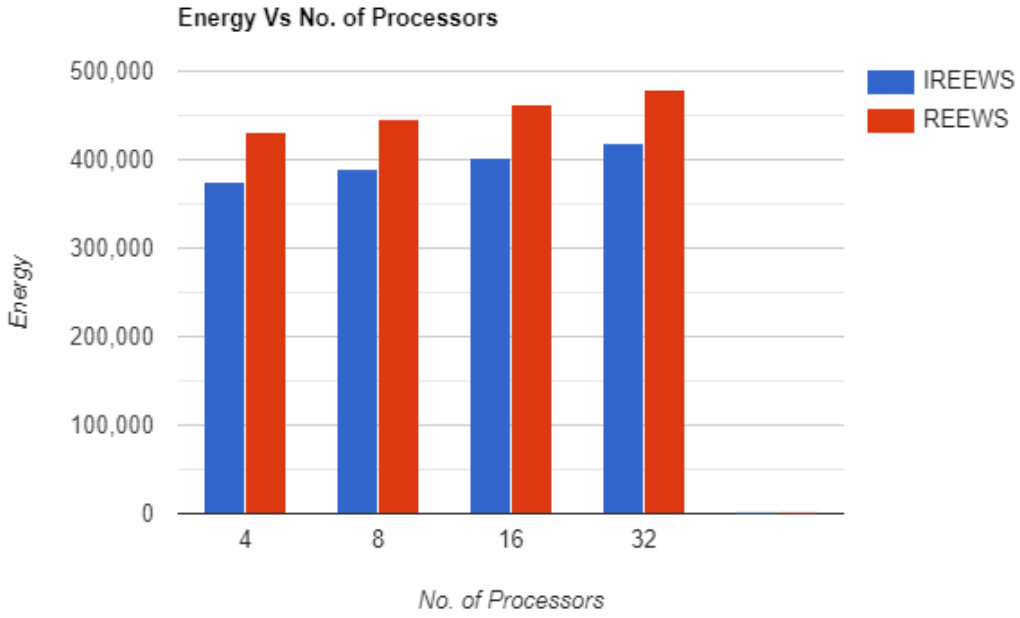

Figure 6.14: Energy Consumption Vs Number of Processors in SIPHT

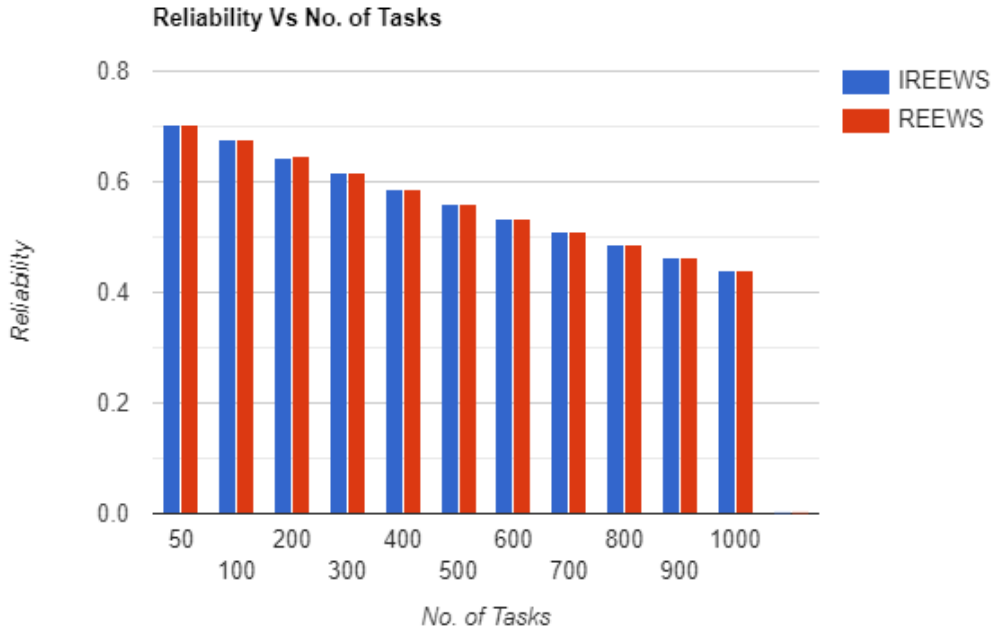

Figure 6.15: Reliability Vs No. of Tasks in SIPHT

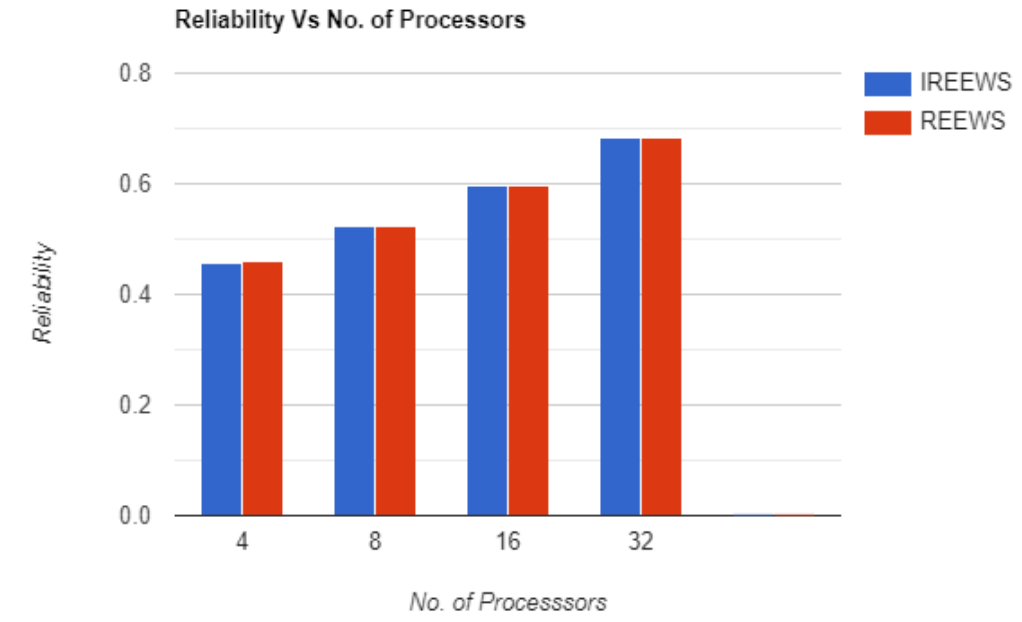

Figure 6.16: Reliability Vs Number of Processors in SIPHT

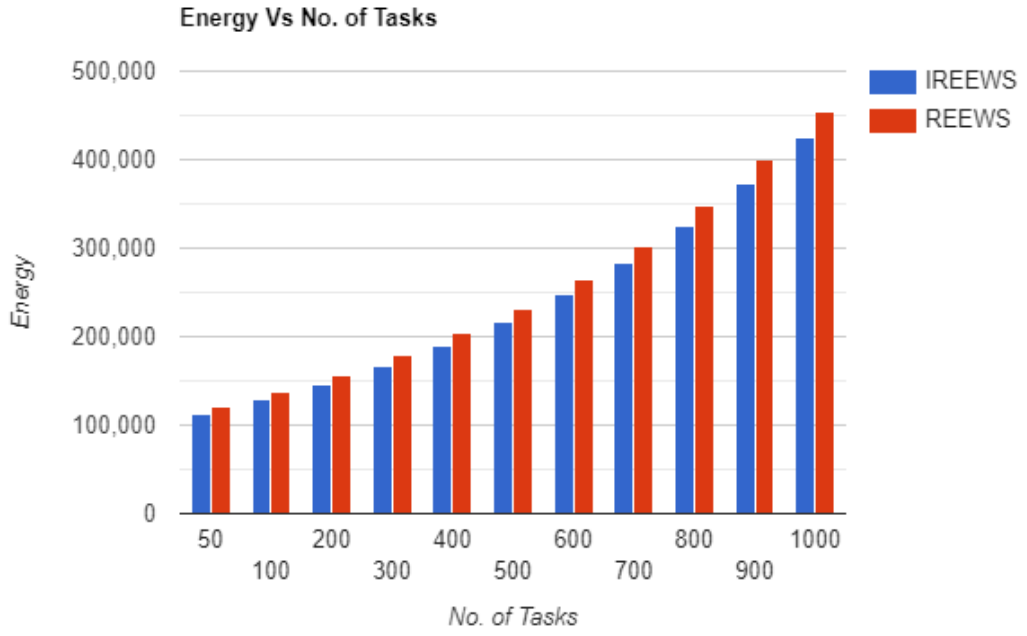

Figure 6.17: Energy Consumption Vs Number of Tasks in LIGO

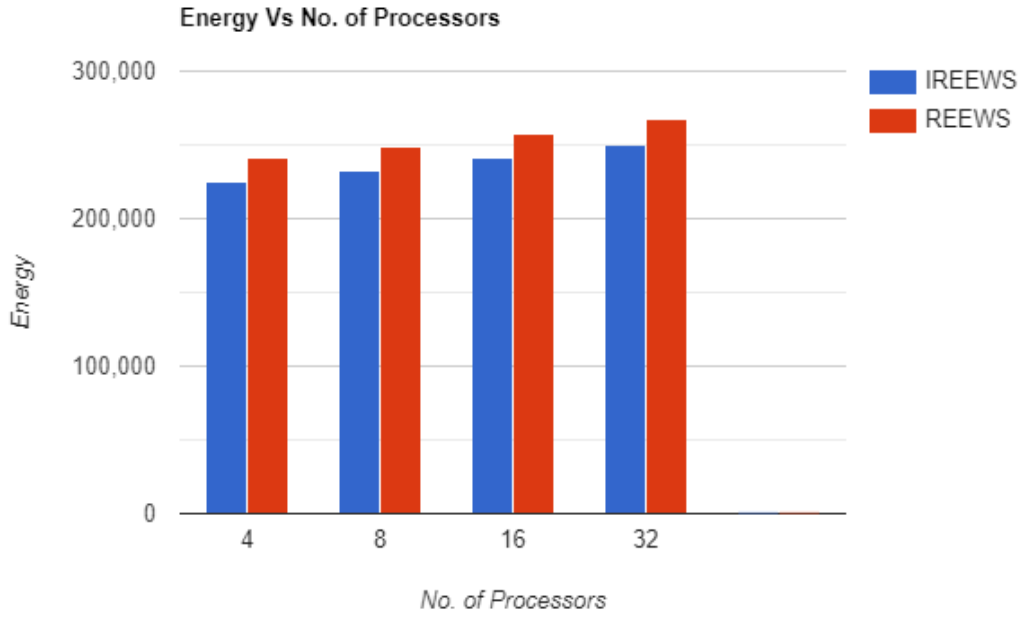

Figure 6.18: Energy Consumption Vs Number of Processors in LIGO

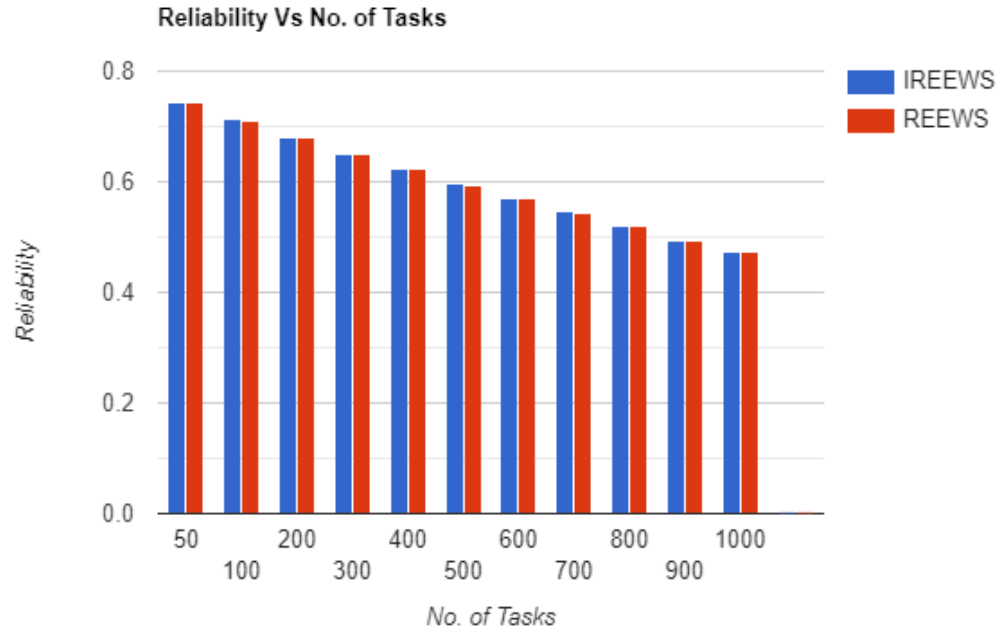

Figure 6.19: Reliability Vs No. of Tasks in LIGO

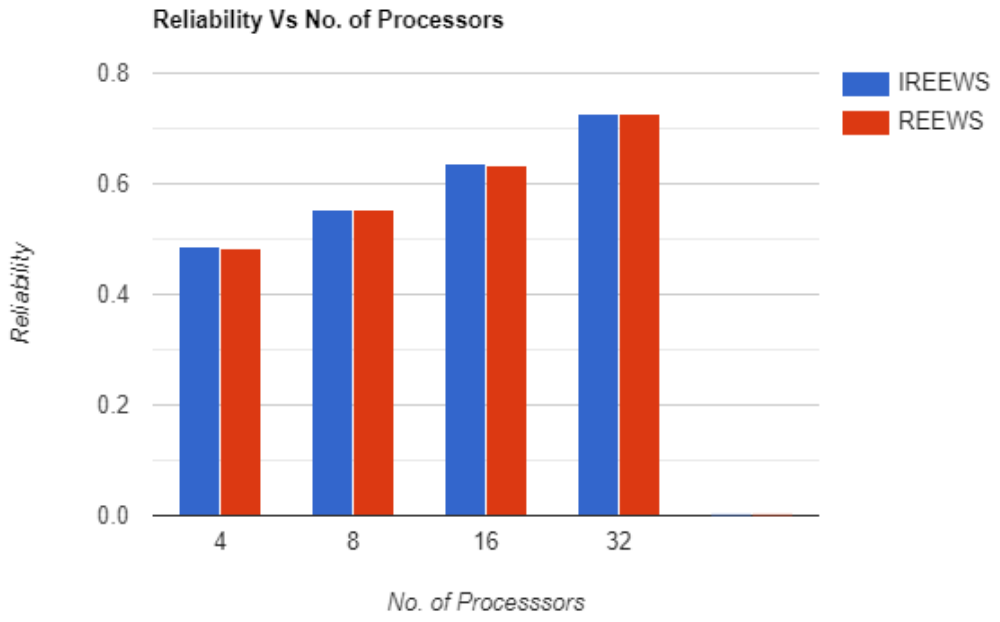

Figure 6.20: Reliability Vs Number of Processors in LIGO

We performed our experiment by varying the total number of processors as 4,8,16,32 and total number of tasks as 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000.In total it was performed on nearly 1100 task graphs which were collected from peagasus workflow gallery. Thus there were total 4400 results were being generated and the graphs are being plotted based on those 4400 results.

In these results we can see that the improved IREEWS algorithms is more energy efficient than actual REEWS algorithm.This happened because of the clustering approach that is being used in IREEWS.The clustering in IREEWS saves much communication time as compared to REEWS algorithm.However in certain special type of graphs like genome it can be observed that there is not much significant difference between the energy consumption while using any of the two algorithms. This can be attributed to the fact that since in Genome there is one task which is connected to all others via some path (in figure 5.3

we can see fastQSplit denoted in yellow color is one such task) due to this when we do clustering then both the algorithms give very similar results and thus there is not so much significant difference in their energy utilisation.IREEWS gives almost similar results than REEWS when it comes to the reliability of the tasks and thus it can be seen that it doesn't do any compromise with the reliability of the tasks while making it more energy efficient.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this work we have tried to create more effective clusters. This helps us in getting an edge in the reducing the total energy consumption. We did our simulation using workflowsim using various types of workflows which we have collected from peagasus workflow gallery.

7.2 Future work

We can make this algorithm more sensitive towards the deadline to fulfill more stricter deadlines. Because it may so happen that after doing clustering of tasks the total execution period of that cluster might extend our expected deadline. In such cases we need to perform declustering by splitting that cluster into smaller clusters and therafter executing them on different VM(s).So, in future we expect to make an efficient way and think of some effective modifications to the algorithm to achive this without doing any harm to energy utilization and reliability.

References

[1] *Ritu Garg, Mamta Mittal , Le Hoang Son "Reliability and energy efficient workflow scheduling in cloud environment" (2019)*

[2] *Wang, L., Khan, S.U., Chen, D., KołOdziej, J., Ranjan, R., Xu, C.Z., Zomaya, A.: Energy-aware parallel task scheduling in a cluster. Fut. Gener. Comput. Syst. 29(7), 1661–1670 (2013)*

[3] *Faragardi, H.R., et al.: An analytical model to evaluate reliability of cloud computing systems in the presence of QoS requirements. In: 2013 IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS). IEEE (2013)*

[4] *Zhu, D., Melhem, R., Mosse, D.: The effects of energy management on reliability in real-time embedded systems. In: IEEE/ ACM International Conference on Computer Aided Design (ICCAD'04), pp. 35–40 (2004)*

[5] *Zhang, Y., Chakrabarty, K.: Energy-aware adaptive checkpointing in embedded real-time systems. In: Proceedings of the Design, Automation Test in Europe Conference, pp. 918–923 (2003)*

[6] *Zhu, D., Melhem, R., Mosse, D.: The effects of energy management on reliability in real-time embedded systems. In: IEEE/ ACM International Conference on Computer Aided Design (ICCAD'04), pp. 35–40 (2004)*

[7] *Lee, Y.C., Zomaya, A.Y.: Energy conscious scheduling for distributed computing systems under different operating conditions.*

[8] IEEE Trans. Parallel Distrib. Syst. 22(8), 1374–1381 (2011) *Orgerie, A.C., Lefe'vre, L., Gelas, J.P.: Save watts in your grid: green strategies for energy-aware framework in large scale distributed systems. In: 2008 14th IEEE International Conference on Parallel and Distributed Systems (pp. 171–178). IEEE (2008)*

[9] *Pruhs, K., Van Stee, R., Uthaisombut, P.: Speed scaling of tasks with precedence constraints. Theory Comput. Syst. 43(1), 67–80 (2008)*

[10] *Wang, L., Khan, S.U., Chen, D., KołOdziej, J., Ranjan, R., Xu, C.Z., Zomaya, A.: Energy-aware parallel task scheduling in a cluster. Fut. Gener. Comput. Syst. 29(7), 1661–1670 (2013)*

[11] *Dogan, A., Ozguner, F.: Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. 13(3), 308–323 (2002)*