OPTIMAL TASK PARTITIONING MODEL IN DISTRIBUTED HETEROGENEOUS PARALLEL COMPUTING ENVIRONMENT

Javed Ali[1] (Research Scholar), Rafiqul Zaman Khan[2](Associate Professor)

Department of Computer Science, Aligarh Muslim University, Aligarh, India.
[1]javedaligs@gmail.com,[2]rzk23@yahoo.co.in

ABSTRACT

Parallel computing systems compose task partitioning strategies in a true multiprocessing manner. Such systems share the algorithm and processing unit as computing resources which leads to highly inter process communications capabilities. We focus on real-time and non preemptive systems. A large variety of experiments have been conducted on the proposed algorithm. Goal of computation model is to provide a realistic representation of the costs of programming.

The paper represents the optimal iterative task partitioning scheduling in the distributed heterogeneous environment. Main goal of the algorithm is to improve the performance of the schedule in the form of iteration using results from previous iterations. The algorithm first uses the b-level computation to calculate the initial schedule and then improve it iteratively. The results show the benefit of the task partitioning. The main characteristics of our method are optimal scheduling and strong link between partitioning, scheduling and communication. Some important models for task partitioning are also discussed in the paper. We target the algorithm for task partitioning which improve the inter process communication between the tasks and use the recourses of the system in the efficient manner. The proposed algorithm contributes the inter-process communication cost minimization amongst the executing processes. This paper is the extended version of [15].

KEYWORDS

Criteria, Communication, Partitioning, Computation, Cluster, Speedup, Serial Exection

1. INTRODUCTION

Parallel computing is used to solve the large problems in the efficient manner. The scheduling techniques we discuss might be used by an algorithm to optimize the code that comes out of parallelizing algorithms. Thread can be used for task migration dynamically [1].The algorithm would produce fragments of sequential code, and the optimizer would schedule these specks such that the program runs in the shortest time. Another use of these techniques is in the design of high-performance computing systems. A researcher might want to construct a parallel algorithm that runs in the shortest time possible on some arbitrary computing system which is used to increase the efficiency and decreases the turnaround time. Parallel computing systems are

implemented upon platform comprise of the heterogeneous platforms comprise the different kinds of units, such as CPUs, graphics co-processors, etc. An algorithm is constructed to solve the problem according to the processing capability of the machines used on the cluster and mode of communication amongst the processing tasks [10]. The communication factor is the highly important feature to solve the problem of task partitioning in the distributed systems. A computer cluster is a group of computers working together closely in such a manner that it's treated as a single computer. Cluster is always used to improve the performance and availability over that of a single computer. Task partitioning is achieved by linking the computers closely to each other as a single implicit computer. The large tasks partitioned in the various tasks by the algorithms to improve the productivity and adaptability of the systems. A cluster is used to improve the scientific calculation capabilities of the distributed system [2]. The process division is a function that divides the process into the number of processes or threads. Thread distribution distributes threads proportionally according to the need, among the several machines in the cluster network [chandu10].Thread is a function which execute on the different nodes independently so communication cost problem is not considerable[3]. Some important model [4] for task partitioning in parallel computing system are: PRAM ,BSP etc.DAG is a well known representation of parallel applications in which nodes represents the tasks and edges represent the communication overhead. ANP (Arbitrary Network Topology) strategy. So the key factor to achieve the desired results upon the dynamically changed hardware constraints.

2. NOTATIN TABLE:

Total(t)	Total Task
DAG(H)	DAG Height
P	Number of Processors
MinCT	Minimum Computational Time
MaxCT	Maximum Computational Time
MinCR	Minimum Computational Rate
MaxCR	Maximum Computational Time
Ψ	Speed Up
b-level	Bottom Level of DAG
e	Serial execution portion of the algorithm

3.1 Priority Assigning and Start Time Computing Phase

Computation of the b-level of DAG is used for the initial scheduling[17]. The following instructions are used to compute the initial scheduling cost of the task graph:

1. Construct a list of nodes in reverse order(L_i)
2. ***for*** each node $a_i \varepsilon L_i$ ***do***
3. max=0
4. ***for*** each child a_c of a_i ***do***
5. if $c(a_i,a_c)$+b-level(a_c)> M then
6. M= $c(a_i,a_c)$+b-level(a_c)
7. ***endif***
8. ***endfor***

9. b-level(a_i)=weight(a_i)+M
10. ***endfor***

In the scheduling process b-level is usually constant until the node has been scheduled. Procedure computes b-level and schedules a list in the descending order. The quantitative behavior of the proposed strategy is depending upon the topology used on the target system. This observation might lead to the conclusion that b-level perform best results for all experiments. Algorithm employ the attribute ALAP (As Late As Possible) start time which measure that how far the node's start time can be delayed without increasing the schedule length. The procedure for computing the ALAP is as follows:

1. construct the ready list in reverse topological order (M_i)
2. ***for*** each node $a_i \varepsilon M_i$ ***do***
3. min=k // where k is call procedure(C.P.) length
4. ***for*** each predecessor a_c of a_i ***do***
5. ***if*** alap(a_c)-c(a_c, a_i)<k then
6. k= alap(a_c)-c(a_c, a_i)
7. ***endif***
8. ***endfor***
9. alap(a_i)=k-wgt (a_i)
10. ***endfor***

According to the priority of the nodes the tasks allocated on the processors in the distributed computing environment. The ALAP time is computing and then constructs a list of tasks in the ascending order of the ALAP time. Ties are broken by considering the ALAP time of the predecessors of the tasks.

4. PRAM MODEL

It is a robust design paradigm provider. PRAM composed of P processors, each with its own un-modifiable program. A single shared memory composed of a sequence of words, each capable of containing an arbitrary integer [5]. PRAM model is an extension of the familiar RAM model of sequential computation that is used in algorithm analysis. It consists of a read-only input tape and a write-only output tape. Each instruction in the instruction stream is carried out by all processors simultaneously and requires unit time, reckless of the number of processors. Parallel Random Access Machine (pram) model of computation consists of a number of processors operating in lock-step and communicating by reading and writing locations in a shared memory in efficient and systematic manner[13].In its model each processor has a flag that controls whether it is active in the execution of an instruction or not. Inactive processors do not participate in the execution of instructions.

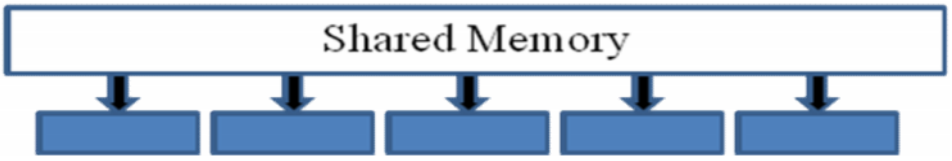


Figure 1.PRAM Model Shared Memory

The processor id can be used to distinguish processor behavior while executing the common program. The operation of a synchronous PRAM can result in simultaneous access by multiple processors to the same location in shared memory. The highest processing power of this model can be used by using Concurrent Read Concurrent Write (CRCW) operation. It's a baseline model of concurrency and explicit model which specify operations at each step[11]. It allows both concurrent reads and concurrent writes to shared memory locations. Many algorithms for other models (such as the network model) can be derived directly from PRAM algorithms[12]. Classification of the PRAM model:

1. In the Common CRCW PRAM, all the processors must write the same value.
2. In the Arbitrary CRCW PRAM, one of the processors arbitrarily succeeds in writing.
3. In the Priority CRCW PRAM, processors have priorities associated with them and the highest priority processor succeeds in writing.

5. PROPOSED MODEL FOR TASK PARTITIONING IN DISTRIBUTED ENVIRONMENT SCHEDULING:

Task partitioning strategy in parallel computing system is the key factor to decide the efficiency, speedup of the parallel computing systems. The process is partitioned into the subtasks where the size of the task is determined by the run time performance of the each server [9]. In this way assign no. of tasks will be proportional to the performance of the server participate the distributed computing system. The inter process communication cost amongst the task is very important factor which is used to improve the performance of the system [6]. The scheduler schedules the tasks and analyzes the performance of the system. The inter processes communication cost estimation criteria in the proposed model is the key factor for the enhancement of the speed up and turnaround time [8]. The C.P.(Call Procedure) is used to dispatching the task according to the capability of the machines. In this model server machine is assume to make up of n heterogeneous processing *elements using the cluster. Every processing element can run one task at a time and all tasks can be assigning to any node. In the proposed model subtasks communicate to each other to share the data, so execution time is reduced due to the sharing of the data. These subtasks assign to the server which dispatch the tasks to the different nodes. The scheduling algorithm is used to compute the execution cost and communication cost.So the server is assumed by a system(P,[Pij],[Si],[Ti],[Gi],[Kij]) as follows:*

a) $P=\{Pi,\ldots.,Pn\}$// **where Pi denotes the processing elements on cluster.**
b) $[Pij]$,**where nxn is processor topology.**
c) Si ,**1<=i<=n, specify the speed of processors *Pi*.**
d) Ti, **1<=i<=n, specify the startup cost of initiating a message on *Pi*.**
e) Gi, **1<=i<=n, specify the startup cost of initiating a process on *Pi*.**
f) Kij, ***is the transmission rate over the link connecting two adjacent processors Pi and Pj.***

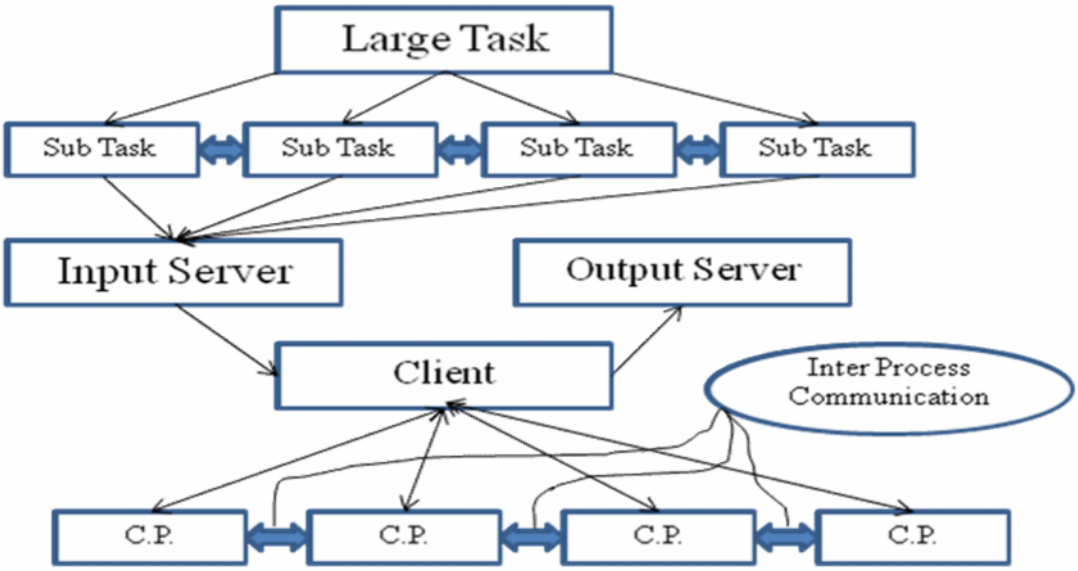


Figure 2: Proposed Dynamic Task Partitioning Model

In the designing of the parallel algorithm, the main goal is to achieve a much as parallelism as possible. Partitioning is the process of dividing the computation and the data into different computational parts.

Nowadays, most research on the integrated circuit or logic optimization are based on single PC, so this paper will add C.P.(Call Procedure) in optimization to improve the speed of logic optimization.

This model splits both computation and data into small tasks [14]. The following basic requirement of partitioning is satisfied by the proposed model:

- There are at least one order of magnitude more primitive tasks than processors upon the target machine to avoid later design options may be too constraints.
- Redundant data structure storage and redundant computations are minimized which cause to achieve large scalability for high performance computations.
- Primitive partition able tasks are roughly of the same size to maintain the balance work among the processors.
- Number of tasks is increasing function of the problem size which avoid the constraints that its impossible to se more processors to solve large problem instances.

The model comprises the existence of an I/O element associated with each processor in the system. The processing time may be executed with help of the Gantt Chart. The connectivity of the processing element can be represented using an undirected graph called the scheduler machine graph [7]. The C.P.(Call Procedure) are used to assign the task dynamically. Task can be assign to a processing element for execution while this processing element is communicating with another processing element. Program completion cost can be computed as:

Total Cost=communication cost +execution cost

Where:

Execution cost=Schedule length
Communication cost=the number of node pairs (w,μ) such that (w, μ)▫A and proc(w)=proc(μ).

Algorithm used for the proposed model: An optimal algorithm for scheduling interval ordered tasks on m processor. A task graph G=(V,A) and m processors, the algorithm generates a schedule f that maps each task v▫V, to a processor Pv and a starting time tv. The communication time between the processor Pi and Pj may be defined as:

comm.(i,j)={0 for i=j, otherwise 1}

task-ready(μ,i,f):the time when all the messages from all task in N(v) have been received by processor Pi in schedule f.

start time(μ,i,f):the earliest time at which task v can start execution on processor Pi in schedule f.
proc(μ,f):the processor assign to task μ in schedule f.

start(μ,f):the time in which task μ begins its actual execution in schedule f.

task(i, τ,f):the task schedule on processor Pi at time τ in schedule f .If there is no task schedule on processor Pi at time t in schedule f ,then task(i, τ,f) returns the empty task Φ. Its assume that n_2 (Φ)< n_2 (μ) .

5.1 Proposed Algorithm for Inter-Process Communication Amongst the Tasks:

In this algorithm the task graph generated and the edge cut gain parameter is considered to calculate the communication cost amongst the tasks[9].

$$gain(i,j) = €.gainedgecut+(1-€)$$
$$gainedgecut=edgecutfactor/oldedgecut$$
$$edgecutfactor= oldedgecut-new\_edgecut$$

Where € is used to set the percentage of gains from edge-cut and workload balance to the total gain.

The bigger €, the higher percentage of edge-cut gain contribute to the total gain of the communication cost.

5.2 Pseudo Code for the Proposed Algorithm:

1.start

2. $task(i,\tau,f) \leftarrow \Phi, for\ all\ positive\ integer\ i, where\ 1 \leq i \leq P\ and\ \tau \geq 0$

3.Repeat

4.Let μ be the unmark task with the highest priority

5. ***for*** **i=1 to P** ***do***

6.compute b-level for all tasks

7.schedule all tasks into non-increasing order of b-level

8.compute alap constructs a list of tasks in the ascending order of the alap time

9. $task\_ready(\mu,i,f) \leftarrow \max(start(\mu,f) + comm(poc(\mu,f),i) + 1) + gain(i,j) for\ each\ \mu$

10. **Start_time(**μ,i,f**)**$\leftarrow \min \tau$**,where**

$task(i,\tau,f) \leftarrow \Phi and\ t \geq task\_ready(\mu,i,f)$

11. ***endfor***

12. **f(μ)**$\leftarrow (start\_time(\mu,i,f)\ if$

13. $start\_time(\mu,i,f) < (start\_time(\mu,i,f),\ 1 \leq j \leq P, i \neq j\ or$

14. $start\_time(\mu,i,f) = (start\_time(\mu,i,f)$ **and**

15. $n_2\ (task(i,(start\_time(\mu,i,f) - 1),$**f)** $\leq n_2\ (task(j,(start\_time(\mu,j,f) - 1),$**f)**

$1 \leq j \leq P, i \neq j$

16. $mark\ task$ **μ until all tasks marked**

17. ***endif***

5.3 Phase (A) Low Communication Overhead: Optimality of the algorithm over the target machine can be achieved due to the following reasons:

Fact(1) **:** **comm(i,** τ_1**,j,** τ_2**) where 1≤,i,j≤P**

Swapping of the task by the task schedule on processor node n_i at τ_1 with the task schedule on n_j at time τ_2.When the swapping of the task amongst the different processor then

Fact(2) : **total comm(i,j,**τ**) where 1≤,i,j≤P**

The effect of the above operation is to swap all the task schedule on node (n_i) at time τ_1 with the task schedule n node n_j at time $\tau_2 \forall\ \tau_1, \tau_2 \geq \tau$.

Fact(3) **:** The following operation is equivalent to the more than one swap operations**:**

total comm(i,j,τ**) ~ comm(i,** τ_1**,j,** τ_2**)** $\forall\ \tau_1, \tau_2 \geq \tau$

The following results from the above facts prove the optimality of the proposed model:

1. The operation comm(i, τ_1,j, τ_2) on the schedule f of the tasks preserves the feasibility of the schedule of any task (w):

$$\mathbf{f(w)=(p, \tau_1) \text{ where } p \in \{i,j\} \text{and } \tau_1 = \tau\text{-}1}$$

2. The feasibility of the schedule f in the proposed model increased for any task schedule

$$\mathbf{f(w)=(p, \tau_1) \text{ where } p \in \{i,j\} \text{ and } \forall\ \tau_1}$$

3. The operation comm(i, τ_1,j, τ_2) and n_2(total comm(i,j, τ) $\geq n_2$(task(i, τ_1,f)) shows the optimality on the schedule of any task(w)

$$\mathbf{f(w)=(p, \tau_3) \text{ where } p \notin \{i,j\}) \text{ and } \forall\ \tau_3}$$

4. The operation comm (i,j,τ) preserves the feasibility of the schedule of any task(w)

$$\mathbf{f(w)=(p, \tau_1) \text{ where } p \in \{i,j\} \text{ and } \tau_1 \leq \tau\text{-}1}$$

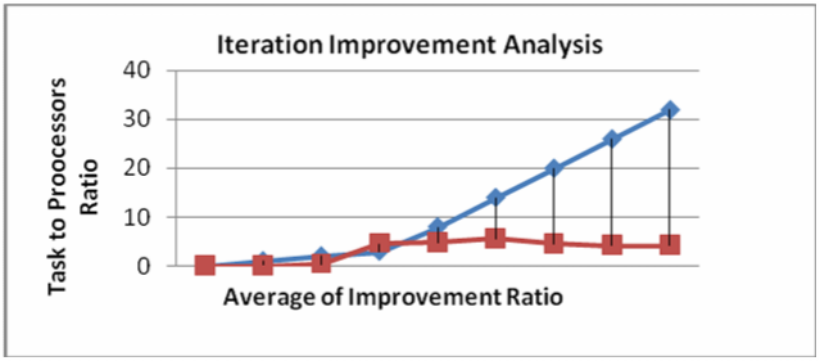
5. The operation comm(i,j,τ) also shows the optimality of the schedule of any task(w)

$$\mathbf{f(w)=(p, \tau_1) \text{ where } p \notin \{i,j\} \text{ and } \forall\ \tau_1}$$

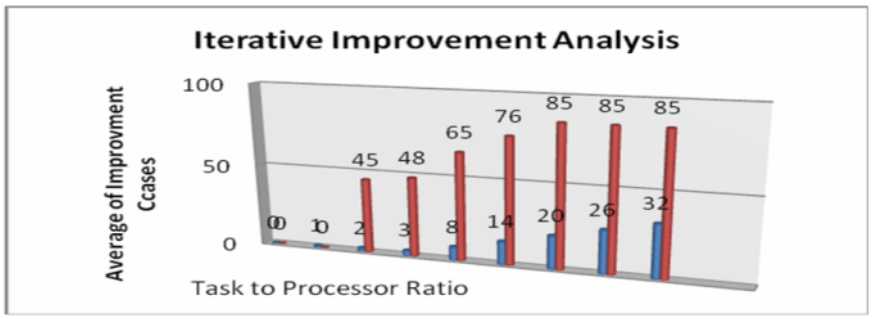
6. EXPRIMENTAL PHASE

In the experiment the CCR is increased as 0.1,0.5,1.0,1.5, 1.75, 2.0 , 5,5.5,7,12 and the load factor varies from 0 to 12 with increment of 1.5 and then increment up to 500.The task number ranging from 5,10,15,40,70,100,130, and 150. In [16] the percentage of the improvement cases is ranging from 70-75%,in the proposed model the percentage of improved cases is upto 85% with the increment of the iterations. Performance analysis of the algorithm based upon the DAG example discussed in [16] with the assuming parameters:

No. of Iteration	MaxCR	MinCR	MaxCT	MinCT	P	DAG(H)	Total(T)
7	6	1	150	12	5	13	150



Fig(A):Iteration Improvement Analysis



Fig(B):Iteration Improvement Analysis

The results shows in figure (A) and fig(B) in which the simulation is run 100 times under each case. The experimental computation predicted that when α small then percentage of improved cases is also small. This simulation result predict that initial schedule iteration contribute very low improvement in the computation of the algorithms. When α is high then the percentage of improved cases reached at critical point, after that it become constant even if α increased .Figure (B) shows that average improvement ratio increased up to the fix limit and its decreases even if the ratio is increases.

In the speedup is the ratio of the serial execution of the program to the parallel execution. In our experiment the result s estimated upon the heterogeneous around 30 processors successively. When the number of the nodes are increased then the speedup is increased up to a improvement level after this level speedup factor is not increased even if the number of processors increase.

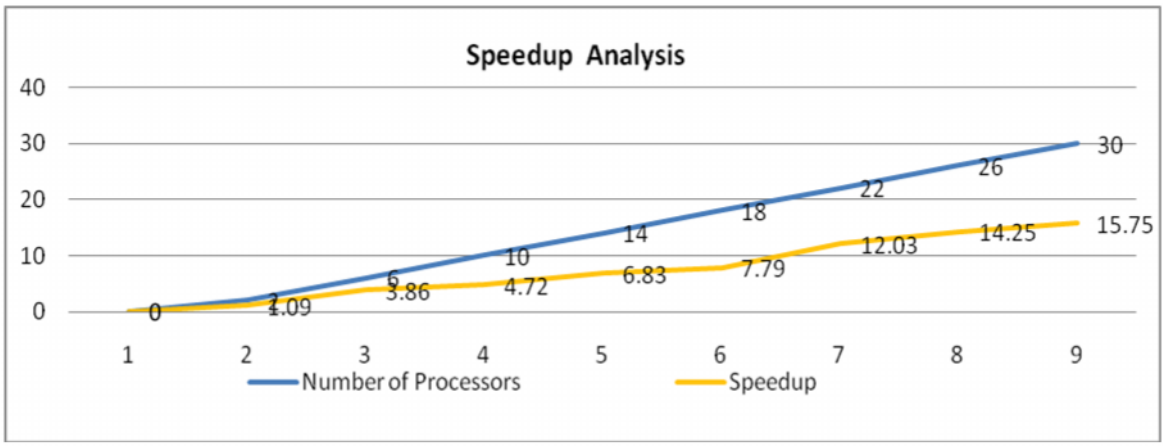


Fig 1. Speedup Analysis of the Algorithm upon the Number of Processors

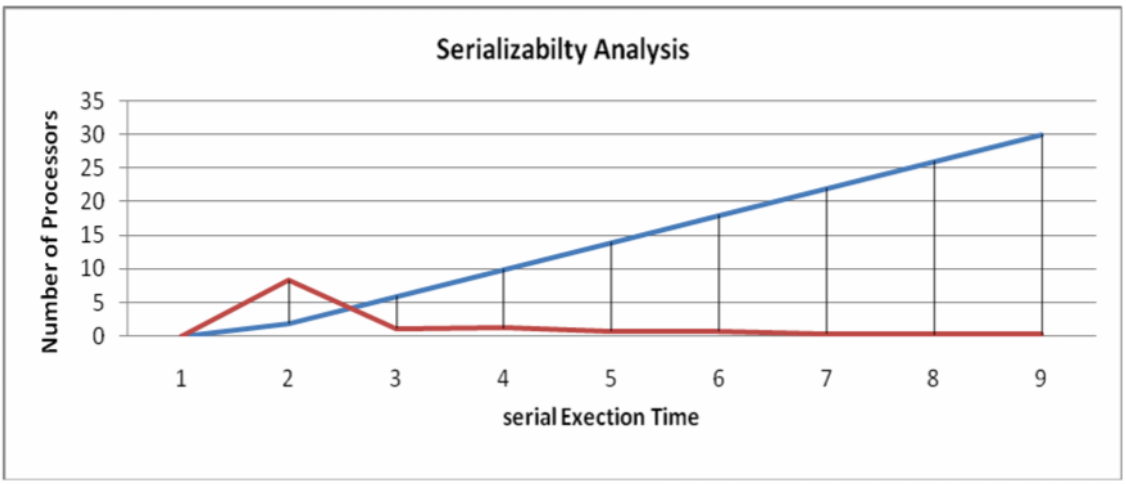


Fig 2. Serializabiliity Analysis of the Algorithm upon the Number of Processors

Amdhal's Law and Gustafson Law ignore communication overhead comm.(i,j), so they can over estimate speedup or scaled speedup. Serial fraction of the parallel computing algorithm can be determined by the Karp-Flatt Metric which is defined as:

$$e = \left(\frac{1}{\psi} - \frac{1}{p}\right) / \left(1 - \frac{1}{p}\right)$$

Serial fraction (e) is useful for the computation of the parallel overhead generated by the execution of the algorithm over the distributed computing environment. Where ( ) is the speedup factor and P are the number of processors using for the parallel execution in distributed heterogeneous environment.

7. CONCLUSION AND FUTURE WORK:

In this paper, we proposed a new model for estimating the cost of communication amongst the various nodes at the time of the execution. The Improvement ratio of the iterations is also discussed in the paper. Our contribution gives cut edge inter-process communication factor which is highly important factor to assign the task to the heterogeneous systems according to the processing capabilities of the processors on the network. The model can also adapt the changing hardware constraints. The researchers can improve the gain percentage for the inter process communication.

8.0 REFERENCES:

[1] N. Islam and A. Prodromidis and M. S. Squillante, ''Dynamic Partitioning in Different Distributed-Memory Environments'', Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, pages 155-170,April 1996.

[2] David J. Lilja, ‘‘Experiments with a Task Partitioning Model for Heterogeneous Computing,’’ University of Minnesota AHPCRC Preprint no. 92-142, Minneapolis, MN, December 1992.
[3] L. G. Valiant. ‘‘A bridging model for parallel computation’’. Communications of the ACM, 33(8):103-111, August 1990.
[4] B. H. H. Juurlink and H. A. G. Wijshoff. ‘‘Communication primitives for BSP Computers’’ Information Processing Letters, 58:303-310, 1996.
[5] H. EI-Rewini and H.Ali, ‘‘The Scheduling Problem with Communication’’ ,Technical Report ,University Of Nebraska at Omaha,pp 78-89,1993.
[6] D. Menasce and V. Almeida, ‘‘Cost-Performance Analysis of Heterogeneity in Supercomputer Architectures ’’, Proc. Supercomputing ’90, pp. 169-177, 1990.
[7] T.L. Adam, K.M. Chandy, and J.R. Dickson, “A Comparison of List Schedules for Parallel Processing Systems,” Comm. ACM, vol. 17, pp. 685-689, 1974.
[8] L. G. Valiant. ‘‘A bridging model for parallel computation’’. Communications of the ACM, 33(8):103-111, August 1990.
[9] H. El-Rewini,T. G. Lewis, Hesham H. Ali , ‘‘ Task Scheduling in Parallel and Distributed Systems”,Prentice Hall Series in Innovative Technology,pp 48-50.1994.
[10] M. D. Ercegovac, ‘‘Heterogeneity in Supercomputer Architectures,’’ Parallel Computing, No. 7, pp.367-372, 1988.
[11] P.B. Gibbons. A more practical pram model. In Pro-ceedings of the i989 Symposium on Parallel Algorithms and Architectures, pages 158-168, Santa Fe, NM, June 1989.
[12] Y. Aumann and M. O. Rabin. ‘‘Clock construction in fully asynchronous parallel systems and PRAM simulation”. In Proc. 33rd IEEE Symp. on Foundations of Computer Science, pages 147-156, October 1992.
[13] R. M. Karp and V. Ramachandran. ,‘‘ Parallel algorithms for shared-memory machines”. In J. van Leeuwen, editor, Handbook of Theoretical Computer Science, Volume A, pages 869-941. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.
[14] H.Topcuoglu, S. Hariri, and M.Y. Wu, “Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing,”IEEE Trans. Parallel and Distributed Systems, Vol. 13, No.3, pp. 250-271, March 2002.
[15] Javed A.,Rafiqul Z. K., “Dynamic Task Partitioning Model in Parallel Computing Systems”, Proceeding First International conference on Advanced Information Technology (ICAIT-2012),Coimbatore, Tamil Nadu,
[16] G. Liu, K. Poh, M. Xie, “Iterative list scheduling for heterogeneous computing, J. Parallel Distrib. Comput”. 65 (5) (200 Manik Sharma, Smriti, “Static and Dynamic BNP
[17] Parallel Scheduling Algorithms For Distributed Database”, IJCT, Vol 1, No.1, 2011. 5) 658–663.

BIOGRAPHY AUTHORS:

(1). Dr. Rafiqul Zaman Khan

Dr. Rafiqul Zaman Khan, is presently working as a Associate Professor in the Department of Computer Science at Aligarh Muslim University, Aligarh, India.

He, is presently working as a Associate Professor in the Department of Computer Science at Aligarh Muslim University, Aligarh, India. He received his B.Sc Degree from M.J.P Rohilkhand University, Bareilly, M.Sc and M.C.A from A.M.U. and Ph.D (Computer Science) from Jamia Hamdard University.

He has 18 years of Teaching Experience of various reputed International and National Universities viz

King Fahad University of Petroleum & Minerals (**KFUPM**), K.S.A, Ittihad University, U.A.E, Pune University, Jamia Hamdard University and AMU, Aligarh. He worked as a **Head** of the Department of Computer Science at Poona College, University of Pune. He also worked as a **Chairman** of the Department of Computer Science, AMU, Aligarh.

His Research Interest includes Parallel & Distributed Computing, Gesture Recognition, Expert Systems and Artificial Intelligence. Presently **04** students are doing PhD under his supervision.

He has published about **25** research papers in International Journals/Conferences. Names of some Journals of repute in which recently his articles have been published are International Journal of Computer Applications (ISSN: 0975-8887), **U.S.A**, Journal of Computer and Information Science (ISSN: 1913-8989), **Canada**, International Journal of Human Computer Interaction (ISSN: 2180-1347), **Malaysia,** and Malaysian Journal of Computer Science(ISSN: 0127-9084), **Malaysia**. He is the Member of Advisory Board of International Journal of Emerging Technology and Advanced Engineering (IJETAE), Editorial Board of International Journal of Advances in **Engineering** & Technology (IJAET), International Journal of Computer Science Engineering and Technology (IJCSET), International Journal in Foundations of Computer Science & technology (IJFCST) and Journal of Information Technology, and Organizations (JITO).

(2)Javed Ali:

Javed Ali is a research scholar in the department of computer science. His field of interest is parallel computing, He published four research paper in international journals. He attended two international conferences. State Scientist award is given to him for making the refrigerator running without electricity.He did Bsc(Hons) and MCA from Aligarh Muslim University,Aligarh.