Optimum Job Segmentation Example in Broadcast Heterogenous Parallel Computing Surroundings

[1]Akhil Kumar, [2] Prof D.P Singh

[1]*Research Scholar Pacific University, Udaipur,* [2] *Prof in RBCET Bareilly,*

ABSTRACT

Parallel computing systems write job breakdown schemes in a true parallel processing manner. Such arrangements apportion the algorithmic program and auctioning unit as computing imaginations which leads to highly inter process communications theory capacities. We concentrate on real-time and non preemptive arrangements. A large assortment of experiments has been carried on the advised algorithmic program. Goal of calculation example is to allow a realistic histrionics of the costs of programming.

The research paper constitutes the optimum iterative aspect job division programming in the broadcast heterogeneous surroundings. Main goal of the algorithm is to amend the performance of the schedule in the form of iteration using results from previous looping. The algorithmic program first applies the b-level calculation to compute the initial schedule and then amend it iteratively. The consequences demonstrate the gain of the job breakdown. The main features of our method are optimum programming and strong associate between breakdown, programming and communication. Some significant examples for job breakdown are also talked about in the paper. We aim the algorithmic program for job breakdown which amend the inter action communication among the jobs and use the appeals of the arrangement in the effective manner. The proposed algorithmic program conduces the inter-process communicating cost reduction between the accomplishing processes. This paper is the broadened version of [1].

KEYWORDS
Standards, Connection, Breakdown, Calculation, Clump, Accelerate, Serial Assassination

1. INTRODUCTION

Parallel computing is used to figure out the large troubles in the efficient way. The programming proficiencies we talk about might be employed by an algorithmic program to optimize the code that appears of parallelizing algorithmic program. Thread can be employed for task movement dynamically [15].The algorithmic program would acquire fragmentizes of sequent code, and the optimizer would agenda these corpuscles such that the curriculum runs in the shortest time. Another use of these proficiencies is in the aim of high-performance computing systems. An investigator might want to conception a parallel algorithmic program that runs in the shortest time possible on some arbitrary computing system which is used to increase the efficiency apartment and decreases the turnaround time. Parallel computer system are enforced upon platform constitute of the heterogeneous platforms constitute the different kinds of units, such as CPUs, graphics co-processors, etc. An algorithm is constructed to solve the problem allowing to the processing capableness of the automobiles used on the cluster and mode of communication amongst the processing tasks [10]. The communicating factor is the highly significant feature to solve the problem of task breakdown in the distributed systems. A cluster is a group of computers working united closely in such a manner that it's treated as a single computer. Cluster is always wont to amend the performance and availability over that of a single computing machine. 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 algorithmic program to amend the productiveness and adaptability of the systems. A cluster is wont to amend the scientific calculation capabilities of the distributed system [2]. The process division is a function that separates the process into the number of processes or threads. Thread distribution distributes threads proportionately according to the need, among the several automobiles in the cluster network [chandu10].Thread is a function which execute on the unlike nodes independently so communication cost problem is not appreciable[3]. Some important model [4] for task partitioning in parallel automatic data processing system are: PRAM ,BSP etc.DAG is a well known internal representation of parallel application program in which nodes represents the jobs and edges represent the communication overhead. ANP (Arbitrary Network Topology) strategy**.** So the key factor to achieve the hoped results upon the dynamically altered 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

2.1 Priority Allotting and Start Time Calculating Phase

Calculation of the b-level of DAG is employed for the initial programming [17]. The following contents are used to calculate the initial programming cost of the job graph:

```
1. Construct a list of nodes in reverse order (Li)
2. for each node aie Li do
3. max=0
4. for each child ac of ai do
5. if c(ai,ac)+b-level(ac)> M then
6. M= c(ai,ac)+b-level(ac)
7. endif
8. endfor
9. b-level(ai)=weight(ai)+M
10. endfor
```

In the programming process b-level is usually constant until the node has been scheduled. Process computes b-level and programs a list in the descending order. The denary behavior of the declared strategy is depending upon the topology used on the aim system. This reflection might lead to the conclusion that b-level perform best results for all tries out. Algorithmic program apply the assign ALAP (As Late As Possible) start time which assess that how far the node's start time can be detained without increasing the schedule length. The procedure for calculating the ALAP is as follows:

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

Accordant to the priority of the clients the tasks allocated on the processors in the apportioned computing environment. The ALAP time is calculating and then constructs a list of tasks in the ascending order of the ALAP time. Ties are bankrupted by believing the ALAP time of the forefathers of the jobs.

3. PRAM MODEL

It is a robust design paradigm provider. PRAM composed of P processors, each with its own unmodifiable 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 calculation that is used in algorithm analysis. It comprises of a read-only input tape and a write-only output tape. Each education in the instruction stream is carried out by all processors at the same time and requires unit time, reckless of the number of processors. Parallel Random Access Machine (pram) model of calculation consists of a number of processors operating in lock-step and communication by reading and writing locations in a shared memory in efficient and systematic manner[13].In its example each CPU has a flag that controls whether it is active in the execution of an command or not. Inactive processors do not take part in the carrying out of educations.

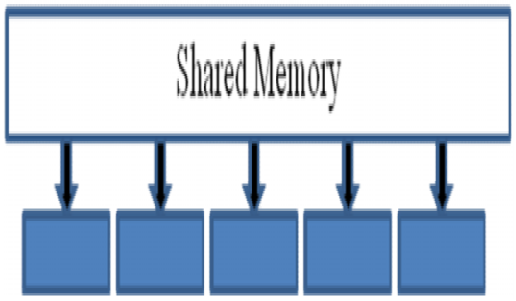


Figure 1.PRAM Model Shared Memory

The CPU id can be used to describe processor conduct while accomplishing the coarse program. The cognitive operation of a synchronous PRAM cans consequence in coincident access by bigeminal C.P.U.s to the same location in shared computer 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 Democratic CRCW PRAM, all the C.P.U.s must compose the same value.
2. In the Capricious CRCW PRAM, one of the processors arbitrarily comes through in writing.
3. In the Antecedence CRCW PRAM, C.P.U.s has priorities affiliated with them and the most eminent priority processor comes after in writing.

4. ADVISED EXAMPLE FOR TASK BREAKDOWN IN ADMINISTERED SURROUNDINGS SCHEDULING:

Job partitioning scheme in parallel computing system is the key component to decide the efficiency, speedup of the parallel automatic data processing system. The process is partitioned off into the subjobs where the size of the task is checked by the run time functioning of the each server [9]. In this way allot no. of jobs will be relative to the carrying out of the server participate the administered computing system. The inter process communicating cost amongst the job is very significant factor which is wont to amend the functioning of the system [6]. The computer hardware agendas the jobs and analyzes the performance of the system. The inter processes communication cost estimation standards in the intended example is the key factor for the enhancement of the speed up and turnaround time [8]. The C.P.(Call Procedure) is used to bumping off 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 marching element can run one task at a time and all jobs can be allotting to any node. In the proposed model subjobs convey to each other to share the data, so execution time is abridged due to the sharing of the data. These subjobs assign to the server which dispatch the jobs to the different nodes. The scheduling algorithm is wont 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,... .,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.*

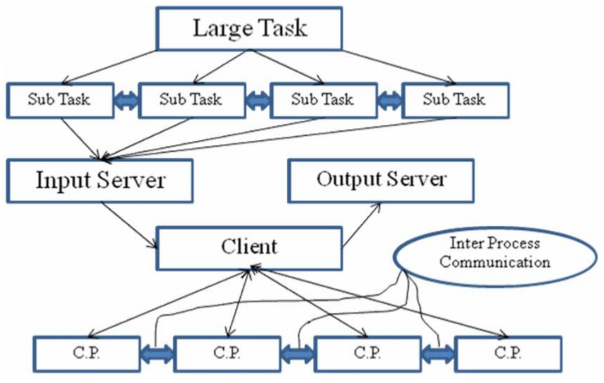


Figure 2: Proposed Active Job Breakdown Example

In the planning of the parallel algorithmic program, the main aim is to accomplish a much as similarity as possible. Breakdown is the action of separating the calculation and the data into dissimilar computational components.
Nowadays, most explore on the integrated circuit or logic optimization are based on single PC, so this research paper will add C.P.(Call Procedure) in optimization to improve the speed of logic optimization.
This example bursts both calculation and data into small activities [14]. The following basic demand of partitioning is met by the aimed example:

- There are at least one order of magnitude more primitive jobs 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 jobs are roughly of the same size to maintain the balance work among the processors.
- Number of jobs is increasing function of the problem size which avoids the constraints that it's impossible to see more processors to solve large problem instances.

The example constitutes the universe of an Input/output element assorted with each CPU in the arrangement. The command processing overhead may be accomplished with help of the Gantt chart. The connectivity of the Agenizing element can be constituted using an aimless graph called the computer hardware machine graph [7]. The C.P. (Call Procedure) is wont to assign the task dynamically. Job can be assign to a marching element for data marching while this processing element is communicating with another processing element. Program completion cost can be calculated 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 programming interval ordered jobs 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 central processor Pv and a starting time tv. The communicating time between the CPU 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 contents 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.

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

start(μ,f):the time in which task μ commences its actual capital punishment in schedule f.

4.1 Aimed Algorithmic program for Inter-Process communicating Between the Jobs:

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

4.2 Pseudo Code for the Proposed Algorithmic program:

```
1.start
2.$task(i,\tau,f) \leftarrow \Phi$, for all positive integer i, where $1 \le i \le P$ and $\tau \ge 0$
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
    $task\_ready(\mu,i,f) \leftarrow$
9.$\max(start(\mu,f)+comm(poc(\mu,f),i)+1)+gain(i,j)$ for each μ
10.    Start_time($\mu,i,f$)← min τ,where
    $task(i,\tau,f) \leftarrow \Phi$ and $t \ge task\_ready(\mu,i,f)$
11.      endfor
12. f(μ)← $(start\_time(\mu,i,f)$ if
13.            $start\_time(\mu,i,f) < (start\_time(\mu,i,f)$ , $1 \le j \le P, i \ne j$ or
14.            $start\_time(\mu,i,f) = (start\_time(\mu,i,f)$ and
15. $n_2\ (task(i,(start\_time(\mu,i,f)-1),f) \le n_2\ (task(j,(start\_time(\mu,j,f)-1),f)$
    $1 \le j \le P, i \ne j$
16. mark task μ until  all tasks marked
17.   endif
```

4.3 Phase (A) Low Communicating Budget items:

Optimum of the algorithmic program over the target automobile can be accomplished due to the following concludes:

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

Exchanging of the job by the task agenda on CPU node ni at τ 1 with the job agenda on nj at time τ 2.When the exchanging of the task amongst the dissimilar processor then

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

The effect of the above cognitive operation is to switch all the task schedule on node (ni) at time τ 1 with the job program n node nj at time τ 2" τ 1, τ 2≥ τ.

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

$$\text{total comm}(i,j,\tau) \sim \text{comm}(i,\tau_1,j,\tau_2)\ \forall\ \tau_1,\tau_2 \ge \tau$$

The following results from the above facts prove the optimum of the proposed example:

1. The operation comm (i, τ 1,j, τ 2) on the program f of the jobs continues the feasibility of the agenda of any job (w):

f(w)=(p, τ 1) where p□{i,j}and τ 1= -1

2. The feasibleness of the agenda f in the aimed model heightened for any task schedule

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

3. The operation comm.(i, τ 1,j, τ 2) and n2(total comm(i,j,) ≥n2(task(i, τ 1,f)) demonstrates the optimality on the schedule of any task(w)

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

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

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

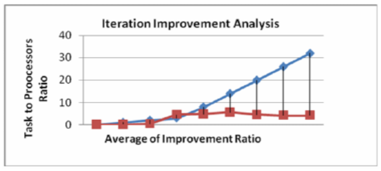
5. The operation comm(i,j, τ) also shows the optimality of the schedule of any task(w)

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

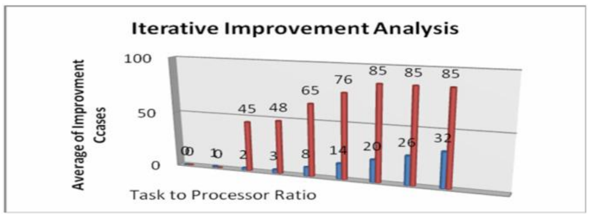
5. EXPRIMENTAL PHASE

In the experimentation the CCR is increased as 0.1,0.5,1.0,1.5, 1.75, 2.0 , 5,5.5,7,12 and the ratio varies from 0 to 12 with growth of 1.5 and then increase up to 500.The task number ranging from 5,10,15,40,70,100,130, and 150. In [16] the part of the amendment cases is ranging from 70-75%,in the proposed model the percentage of amendable cases is upto 85% with the increase of the iterations. Carrying out analytic thinking of the algorithmic program based upon the DAG case discussed in [16] with the accepting parameters:

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



Fig(A):Loop Affirmation Analytic thinking



Fig(B):Loop Statement Analytic thinking

The consequences shows in figure (A) and fig(B) in which the computer simulation is run 100 times under each argumentation. The observational calculation augured that when a small then percent of amendd cases is also small. This computer simulation result indicate that initial agenda iteration encourage very low rectification in the calculation of the algorithmic program. When a is high then the percentage of amendable cases accomplished at critical point, after that it become constant even if a enhanced .Figure (B) depicts that average rectification ratio increased up to the fix limit and its diminishes even if the ratio is gains.

In the accelerate is the proportion of the serial execution of the program to the parallel assassination. In our experiment the result s estimated upon the heterogeneous around 30 C.P.U.s successively. When the count of the nodes are increased then the accelerate is increased up to a amendment level after this level accelerate factor is not increased even if the number of processors increase.

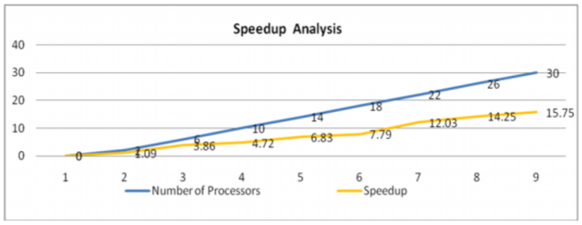


Fig 3. Speedup Analysis of the Algorithm upon the Number of Processors

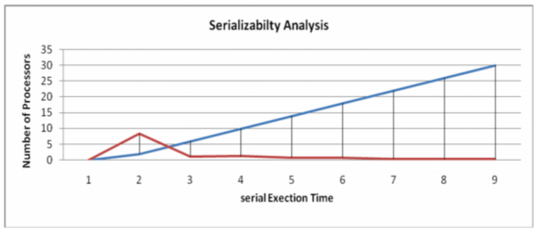


Fig 4. Serializabiliity Analysis of the Algorithm upon the Number of Processors

Amdhal's Law and Gustafson Law ignore communicating overhead comm.(i,j), so they can over appraisal 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 practicable 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.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new model for figuring the cost of communicating between the several nodes at the time of the execution. The Amendment ratio of the iterations is also discussed in the paper. Our contribution gives cut edge inter-action communication factor which is highly important factor to assign the task to the heterogeneous arrangements according to the auctioning capableness's of the C.P.U.s on the network. The model can also adapt the changing hardware constraints. The researchers can amend the gain percentage for the inter process communication.

7.0 REFERENCES:

[1] 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,

[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 Programming 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 Programming 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 Programming for Heterogeneous Computing,"IEEE Trans. Parallel and Distributed Systems, Vol. 13, No.3, pp. 250-271, March 2002.

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

[16] G. Liu, K. Poh, M. Xie, "Iterative list programming for heterogeneous computing, J. Parallel Distrib. Comput". 65 (5) (200 Manik Sharma, Smriti, "Static and Dynamic BNP

[17] Parallel Programming Algorithms For Distributed Database", IJCT, Vol 1, No.1, 2011. 5) 658–