# iit_bhu.jpg

**Implementation and Performance Comparison of Task Scheduling Algorithms**

B.Tech Project Part I

By

Avinash Kumar

(Roll No.:14075011)

Under the guidance of

Prof. A.K. Tripathi

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY (BHU), VARANASI**

**Declaration**

I hereby declare that

1. The work contained in this report is original and has been done by me and under the general guidance of our supervisor Prof A.K Tripathi.
2. The work has not been submitted for any other project.
3. Whenever I have used materials from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

Place: IIT (BHU) Varanasi

Date:

Avinash Kumar

B.Tech Part (III)

Department of Computer Science && Engineering,

Indian Institute of Technology (BHU) Varanasi,

Varanasi, UP 221005.

**Certificate**

This is to certify that the work contained in this report entitled “Task Scheduling Algorithms" being submitted by Avinash Kumar (Roll No 14075011), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi is a bona fide work under my supervision.

Place: IIT (BHU) Varanasi

Date:

Prof. A.K. Tripathi

Department of Computer Science && Engineering,

Indian Institute of Technology (BHU) Varanasi,

Varanasi, UP 221005.

**Acknowledgements**

I would like to express our sincere gratitude to Prof. A.K. Tripathi for giving me the opportunity to explore the field of Distributed and Parallel Computing, by encouraging me to work on this project titled "Task Scheduling", wherein I carried out the study of existing work in the field of task scheduling and also implemented the famous PATS algorithm along with another algorithm given in recent years and also analyzed their performance on various metrics.

Special thanks to Mr. Ashish Kumar Maurya, who dedicated his valuable time and provided me a roadmap for carrying out my project work and was always willing to help me when I was stuck in some problem.

I will also like to thank my branch mates for their valuable suggestions.

**Table of Contents**

Declaration

Certificate………………………………………………………………………….1

Acknowledgement…………………………………………………………………2

Abstract…………………………………………………………………………….3

1. Introduction…………………………………………………………………………4 -5
2. Problem Formulation……………………………………………………………….6 -8
3. Existing Work………………………………………………………………………9-10
4. Algorithms Implemented…………………………………………………………..11-12
   1. PATS…………………………………………………………………………..13-14
   2. PETS…………………………………………………………………………….15-16
5. Illustrative Example…………………………………………………………………17
6. Experimental Results………………………………………………………………18
7. Conclusions and Next Phase Work………………………………………………..18-19

References

**Abstract:**

A heterogeneous task scheduling algorithm called Predict and Arrange Task Scheduling (PATS) algorithm was proposed to achieve a lower bound time complexity with minimum schedule length. Two major steps were introduced, i.e. earliest finish time with level-based task scheduling and idle slot reduction. In the first step, tasks are scheduled according to their predicted earliest finish time from the candidate task list and their dependencies.

Finding an optimal solution to the problem of scheduling an application modeled by a Directed Acyclic Graph (DAG) onto a distributed system is known to be NP-complete. The complexity of the problem increases when task scheduling is to be done in a heterogeneous computing system, where the processors in the network may not be identical and take different amounts of time to execute the same task. This paper introduces a Performance Effective Task Scheduling (PETS) Algorithm for Network of Heterogeneous system, with complexity O (v+e) (p+ log v), which provides optimal results for applications represented by DAGs.

A mapping algorithm for heterogeneous computing systems is proposed in this paper. This algorithm utilizes a new indicator— the relative cost — to obtain optimal mapping. Theexisting Min-min algorithm can be well explained under synergy of this new indicator.

1. **Introduction**

Their aim is to speed up the execution of an application through the collaboration of the processing units. With the introduction of dual-core and multicore processors by IBM, AMD, Intel, and others, even mainstream PCs have become parallel systems. Even though the area of parallel computing has existed for many decades, programming a parallel system for the execution of a single application is still a challenging problem, profoundly more challenging than programming a single processor, or sequential, system. Generally, there are dependences between the tasks that impose a partial order on their execution. Adhering to this order is essential for the correct execution of the application. A crucial step of parallel programming is the allocation of the tasks to the processors and the deﬁnition of their execution order. This step, which is referred to as scheduling, fundamentally determines the efficiency of the application’s parallelization, that is, the speedup of its execution in comparison to a single processor system.

Distributed computing is a model in which components of a software system are shared among multiple computers to improve efficiency and performance. Distributed computing is a field of [computer science](https://en.wikipedia.org/wiki/Computer_science) that studies distributed systems. A distributed system is a model in which components located on [networked computers](https://en.wikipedia.org/wiki/Computer_network) communicate and coordinate their actions by [passing messages](https://en.wikipedia.org/wiki/Message_passing). In the broadest sense of the term, [distributed](http://whatis.techtarget.com/definition/distributed) computing just means that something is shared among multiple systems which may also be in different locations.

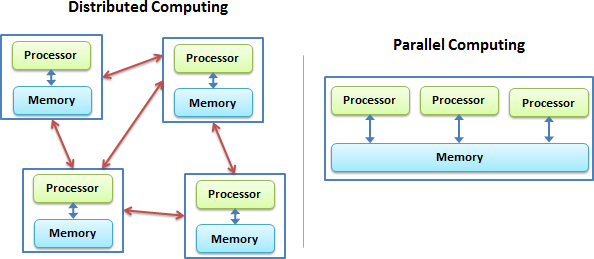


Figure 1:

Heterogeneous computing refers to systems that use more than one kind of processor or cores. These systems have gained performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar [coprocessors](https://en.wikipedia.org/wiki/Coprocessors), usually incorporating specialized processing capabilities to handle particular tasks. Heterogeneous Computing (HC) system is a suite of distributed processors interconnected by high-speed networks, thereby promising high speed processing of computationally intensive applications with diverse computing needs.

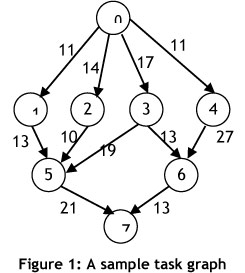
Task scheduling in parallel processing is a technique in which processes are assigned to different processors. Task scheduling in parallel processing uses different types of algorithms and techniques which are used to reduce the number of delayed jobs. Nowadays there are different kind of scheduling algorithms and techniques used to reduce the execution time of tasks. As task scheduling is an NP-hard problem and no one can say that about the best algorithm proposed, we will review some of the task scheduling algorithms and other techniques. Parallel processing is dividing the process into multiple processes and executes them concurrently by the use of more than one CPU or processor.

**Processor Allocation:** A processor allocation of the task graph G= (V,E,w,c) on a ﬁnite set P of processors is the processor allocation function proc: V→P of the nodes of G to the processors of P.

**Schedule:** A schedule S of the task graph G= (V,E,w,c) on a ﬁnite set P of processors is the function pair (ts,proc), where

• ts: V→ Q+ 0 is the start time function of the nodes of G.

• proc: V→P is the processor allocation function of the nodes of G to the processors of P.



1. **Problem Formulation**

This section describes the non-preemptive task scheduling problem for heterogeneous computing environment with the objective of minimizing the overall execution time. An application program consists of a set of tasks and is modeled by a directed acyclic graph (DAG). The graph consists of n tasks and there exit a partial order that is if t(i) < t(j) then t(j) cannot be scheduled until t(i) has completed its execution. E is the set of directed edges. A weight w(i,j) associated with each edge represents the amount of data to be transferred from ith task to jth task. A heterogeneous computing environment consists of a set of processors interconnected by a fully connected topology.

1. **Actual Finish Time (AFT):** *Actual finish time*of task*vj*, denoted as*AFT*(*vj*), is the point of time where the execution of task*vj*is finished bysome assigned processing unit. Note that this is not the estimated finishing time determined during the scheduling process.
2. **Available Time (AVT):** *Available time*of an available processing unit*pa*, denoted as*available*\_*time(**pa**),*is the point of time whereprocessing unit *pa* is neither executing any task nor waiting for the resulting data of predecessor tasks from other processing units.
3. **Schedule Length Ratio (SLR):** It is the ratio of the parallel time to the sum of weights of the critical path tasks on the fastest processor**.** The SLR value is computed by the ratio of makespan and summation of the minimum calculation time of all tasks on the critical path (CP).

**SLR = makespan /{ *vi* *CP* min*pa* P*vi* *(tpa* *(vi* *))}***

1. ***Eﬃciency***: This is another performance evaluation metric defined as the ratio between the lower bound of schedule lengthand the makespan. The lower bound is defined as the summation of the minimum time of all tasks divided by the number of processing units used. The closer is the eﬃciency value to unity, the higher is the performance.

***Efficiency = Lower bound/ makespan;***

1. ***Algorithmic running time*:** The algorithmic running time (*ART*) of an algorithm is the amount of time spent by the scheduling algorithm. It could be viewed as an overhead of determining the schedule length.
2. **Schedule Length:** The schedule length or *makespan* denotes the point of time where execution of the latesttask *vf* finishes. It is the summation of *AFT*(*vf*) from all processing units. This schedule length is the output of all task schedules.

**Scheduling length = Max{ (AFT(V**exit**) )}**

1. **Existing Task Scheduling Approach**

**Algorithm1: Understanding Fundamentals of Fuzzy Rule based Approach Algorithm.**

* This rule-based method proposes an adaptive framework that extends the original model with parameters of Task Scheduling.
* This Approach improves the performance of the model and make it more responsive of the processes.
* This rule-based model has parameters Queue Manager, Process selector.

**Algorithm2: Fundamentals of Mapping Algorithm.**

* This mapping algorithm is described to Min\_min algorithm. This algorithm is used for load balancing and ability to good mapping for task scheduling in parallel computing.
* Min\_min algorithm is also reduced the time complexity of process in task scheduling.
* In a mapping algorithm also added a new type of algorithm which improved to the efficiency and time complexity, load balancing of Min\_min algorithm which called Relative Cost algorithm.

**Algorithm3**: **Predict and Arrange Task Scheduling (PATS).**

* This algorithm wasproposed to achieve a lower bound time complexity with minimum schedule length.
* This algorithm is also proposed to minimize well energy consumption and minimize respond time.
* This also proposed scheduling graph transformation in different category in the Task scheduling.
* This algorithm are work in two phase first one is arrange the all the task to attain the shortest schedule length and other one is reduce the idle slot produce from first phase.

**Algorithm4: Performance Effective Task Scheduling (PETS) Algorithm**.

* This algorithm is used for Network of Heterogeneous system for Directed Acyclic Graph in task scheduling.
* The performance of the algorithm is to explain by comparing the schedule length, speedup, efficiency and the scheduling time with existing algorithms such as, Heterogeneous Earliest Finish Time (HEFT) and Critical-Path On a processor (CPOP) and Levelized Min Time (LMT).
* The proposed algorithm consists of three phases, viz., level sorting, task prioritization, and processor selection.

**Algorithm5: Fundamentals of Lookahead Algorithm.**

* This algorithm is used to improve the efficiency of the Heterogeneous Earliest Finish Time (HEFT).
* After some time found the results indicate that the lookahead variation of HEFT can effectively reduce the makespan of the schedule in most cases without making the algorithm’s execution time prohibitively high in DAG.

**List Scheduling Heuristics:** A list-scheduling heuristic maintains a list of a given graph according to their priorities. It has two phases: task prioritizing phase for selecting the highest priority ready task and processor selection phase for selecting a suitable processor that minimizes a predefined cost function.

**Task Duplication Heuristics:**  The idea behind task duplication based scheduling algorithm is to schedule a task graph by mapping some of its tasks redundantly to reduce interprocessor communication overhead.

**Randomly Generated Application Graphs:** As part of this work we have implemented a random task graph generator that allows the user to generate a variety of test DAGs with various characteristics that depends on several input parameters and they are number of tasks in the graph (v), out degree (β), in degree (γ), shape parameter of a graph (α) Communication to Computation Ratio (CCR).

**4. Algorithms implemented in the Project**

In this paper I have implement the famous PATS algorithm and PETS algorithm and the algorithms are compared on the basis of two metric parameters makespan and SLR values.

**4.1 Predict and Arrange Task Scheduling (PATS).**

The PATS algorithm is explained as under:

* For each task v the prime candidate p and G find from the given parameters.
* In given graph searches v and p into each levels L.
* If found a set which contains all vertex v then worked on this vertex.
* If also found p then compute earliest finish time for each vertex.
* Marked all vertex while all vertex assigned in each level of graph. And further compute max of EST.
* If not found any vertex in graph leaved this graph.
* If remaining task on same level are updated and queued until all the task on the same level.
* This process repeats for all other remaining level of the graph. .

**Processor Selection Phase:** In the processor selection phase that processor is selected for the task that minimizes EFT.

**Time Complexity:** This algorithm has *O*(*v*2 ***×*** *p*) time complexity for *v* vertices or number of tasks and *p* processing units.

**Earliest Start Time (EST):** *Earliest start time*of task*vi*at processing unit*pa*, denoted as*EST**pa**,*is the earliest point of time to start theexecution of task *vi* at assigned processing unit *pa.*

**EST(vi, pa) = Max(max(AFT(Vj), Wpa(Vi)), available\_time(pa))**

where *wpa* *(v* *j* *)* is the amount of time to transmit the resulting data of task *vj* after executing on other processing unit to processing unit *pa*. If a descendant task *vi* which is waiting for the resulting data from predecessor task *vj* is on the same processing unit as *vj*, then the communication cost is assumed to be zero.

**Earliest Finish Time (EFT):** *Earliest finish time*of task*vi*executed by processing unit*pa*, denoted as*EFTvi,**pa**,*is the point of time wherethe execution of task *vi* is finished.

***EFTvi , pa = ESTvi , pa + tpa (vi )***

where *tpa* *(vi* *)* is the amount of time used to execute task *vi* on processing unit *pa*. The second scenario occurs when the length of available time slot is shorter than the actual execution time of *vi*. This implies that any assigned tasks *vk* in the same processing unit *pa* must be shifted their starting times to new starting times. Let *V* be the set of all shifted tasks. The value of *EFTvi,* *pa* in this scenario becomes.

***EFTvi , pa = ESTvi , pa + tpa (vi ) +* *tpa (vk )***

4.2 **Performance Effective Task Scheduling (PETS) Algorithm.**

Algorithm:

* Read the DAG, associated attributes values, and the number of processor P;
* For all tasks at each level Li do
* Begin
* Compute DRC, DTC and ACC.
* Compute rank(vk) = DTC(vk)+DRC(vk)+ACC(vk)
* Construct a priority queue using ranks;
* While there are unscheduled tasks in - > the queue do
* Begin
* Select the first task, vk from the queue for scheduling;
* For each processor pk in the processor set P do
* Begin
* Compute EFT (vk,pk) value

using insertion based scheduling policy;

* Assign the task vk to processor pk,

which minimizes the EFT;

* End;
* End;

**Earliest Finish Time (EFT):** *Earliest finish time*of task*vi*executed by processing unit*pj*, denoted as*EFTvi,**pj**,*is the point of time wherethe execution of task *vi* is finished.

***EFT(vi, pj) = Wij + EST(vi, pj);***

**Earliest Finish Time (EFT):** *Earliest finish time*of task*vi*executed by processing unit*pj*, denoted as*EFTvi,**pj**,*is the point of time wherethe execution of task *vi* is finished.

***EFT(vi, pj) = Wij + EST(vi, pj);***

**DTC:** Data transfer cost

**DTC(Vi) = Summation{Ci, j}/m from j=1 to x,**

x is the number of immediate successors of vi

**DRC:** Data Receiving Cost (DRC)

**DRC(Vj) = Max {rank(vj)}**

**ACC:** The Average Computation Cost (ACC) of a task is the average computation cost on all the available m processors and it is computed by using Eqn.

**ACC(Vi) = Summation{Wi, j}/m from j=1 to m**

**Rank Of Each Task:** Rank of a task vi (rank(vi)) is the sum of its DTC, DRC and ACC values of that task. For the every task at each level i, rank(vi) is computed by using Eqn.

**rank( Vi ) = DTC( Vi ) + DRC( Vi ) + ACC( Vi )**

**Comparison Metrics:** We have used the following metrics to evaluate the performance of the algorithm and the metrics are.

* **Schedule Length Ratio (SLR):** It is the ratio of the parallel time to the sum of weights of the critical path tasks on the fastest processor**.**
* **Speedup:** The speedup is the ratio of the sequential execution time to the parallel execution time.
* **Efficiency:** The efficiency is the ratio of the speedup value to the number of processor used to schedule the graph.
* **Frequency of better quality of schedules:** The number of times that each algorithm produced better, worse and equal quality of schedules compared to every other algorithm is counted in the experiments.

**Time Complexity**: The time complexity of the above algorithm is O (v+e) (p+ log v) where v is the number of vertex and p is the number of processors and e is number of edge of graph.

1. **Illustrative example**

For an illustration consider the application directed acyclic graph given below.

**8**

**6**

**2**

**100**

|  |  |
| --- | --- |
|  |  |

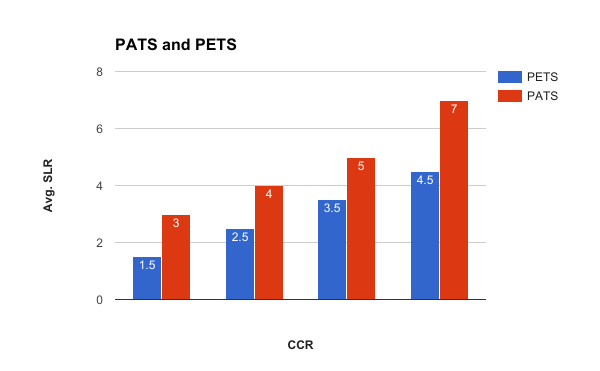
|  |  |  |
| --- | --- | --- |
| Node | Rank |  |
| 0 | 25.4 |  |
| 1 | 12.7 |  |
| 2 | 18.7 |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Node | Processor | Start Time | Finish Time |
| 0 | P0 | 0.000 | 2.000 |
| 1 | P0 | 2.100 | 9.100 |
| 2 | P1 | 5.100 | 11.100 |
| 3 | P1 | 11.400 | 14.400 |

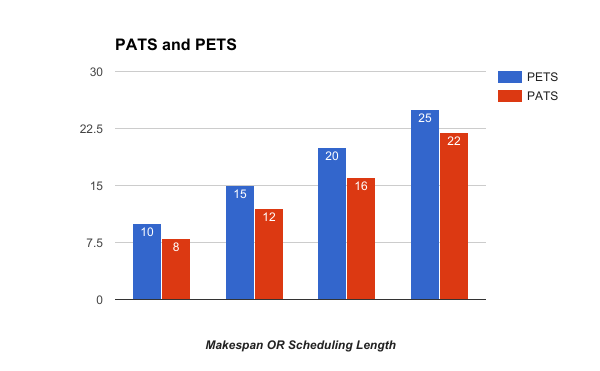
1. **Results**

* The proposed PATS distributed assignment algorithm focuses on low time complexity having shortest schedule length or makespan and idle reduction.
* The task scheduling algorithm PETS proposed here has been proven to be better for scheduling DAG structured applications onto heterogeneous computing system in terms of performance matrices (average schedule length ratio, speedup, efficiency, frequency of best results) and scheduling time.

**Experimantal Results**

****

**Performance of Makespan Between PATS && PETS**

****

**Conclusions and Next Phase Work**

In the next phase of my project I intend to implement two more algorithms Lookahead and MSL. The present algorithms were implemented on randomly generated graphs with varying number of nodes and processors .In order to study the correct behavior of these algorithms huge variety of graphs needs to be generated. So generation of all types of graph that will help in properly understanding the effectiveness of the algorithms needs to be done too. After successful generation of all types of graphs, these implemented algorithms will be run on these graphs and various parameters like makespan and SLR will be calculated and their performance will be measured. Different types of graphs (pictorial ones) between numbers of nodes vs makespan, number of nodes vs SLR, CCR vs makespan will be presented. After the detailed study of these algorithms and their performance, if possible some improvement will be suggested.

**References**

1. [J. Khan, R. Vemuri, An iterative algorithm for battery-aware task scheduling on portable computing platforms, Proceedings of the Design, Automation](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0001) [and Test in Europe (DATE), IEEE Computer Society, Munich, Germany, 2005, pp. 622–627.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0001)
2. [V. Rao, N. Navet, G. Singhal, A. Kumar, G.S. Visweswaran, Battery aware dynamic scheduling for periodic task graphs, Proceedings of the 20th Interna-tional Parallel and Distributed Processing Symposium (IPDPS), IEEE, Rhodes Island, Greece, 2006.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0002)
3. [S. Zhang, K.S. Chatha, Automated techniques for energy eﬃcient scheduling on homogeneous and heterogeneous chip multi-processor architectures,](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0003) [Proceedings of the 13th Asia and South Pacific Design Automation Conference (ASPDAC), IEEE, Seoul, Korea, 2008, pp. 61–66.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0003)
4. [C. Yang, J. Chen, T. Kuo, L. Thiele, An approximation scheme for energy-eﬃcient scheduling of real-time tasks in heterogeneous multiprocessor systems,](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0004) [Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), IEEE, Nice, France, 2009, pp. 694–699.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0004)
5. [S. Zikos, H.D. Karatza, Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times, Simul. Model.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0005) [Pract. Theory 19 (1) (2011) 239–250.](http://refhub.elsevier.com/S1569-190X(15)30094-0/sbref0005)
6. R.L. Graham, L.E. Lawler, J.K. Lenstra, and A.H. Kan, “Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey”, Annals of Discrete Mathematics, pp. 287-326, 1979.
7. T. Cassavant and J.A. Kuhl, “Taxonomy of Scheduling in General Purpose Distributed Memory Systems”, IEEE Trans. Software Engineering, vol. 14, no. 2, pp. 141-154, 1988.
8. [3] C.C. Hui and S.T. Chanson, “Allocating Task Interaction Graphs to Processors in Heterogeneous Networks”, IEEE Trans. Parallel and Distributed Systems, vol. 8, no. 9, pp. 908-926, Sept. 1997.