CHAPTER 2

*Literature Survey*
## CHAPTER 2

### Chapter 2: Literature Survey

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.</td>
<td>GENERAL</td>
<td>20</td>
</tr>
<tr>
<td>2.2.</td>
<td>DISTRIBUTED COMPUTING TO GRID COMPUTING</td>
<td>23</td>
</tr>
<tr>
<td>2.3.</td>
<td>REVIEW OF GRID SCHEDULING ALGORITHM</td>
<td>26</td>
</tr>
<tr>
<td>2.4.</td>
<td>RELATED WORK ON SCHEDULING WORKFLOWS</td>
<td>36</td>
</tr>
<tr>
<td>2.5.</td>
<td>RELATED WORK ON SCHEDULING HEURISTICS</td>
<td>40</td>
</tr>
<tr>
<td>2.6.</td>
<td>TIME COST OPTIMIZATION</td>
<td>45</td>
</tr>
<tr>
<td>2.7.</td>
<td>POWER MODELS</td>
<td>47</td>
</tr>
<tr>
<td>2.8.</td>
<td>POWER MANAGEMENT TECHNIQUES</td>
<td>54</td>
</tr>
</tbody>
</table>
2. LITERATURE SURVEY

2.1. GENERAL

The quality of service must be ensured for the execution of the task by the grid. It is important to have a typical grid framework to build the computing environment of grid (i.e., a grid middleware). The Globus is the grid world’s de facto standard and has been effective in creating open and standard conventions and interfaces. PKI based grid testament is its significant support to solve the grid security issue, which empowers cross-hierarchical asset access control. Likewise, it gives tool box and instruments to VO-degree work accommodation (GRAM) and resource detection (GRIS).

Cluster computing system is referred as a “local scheduler” instead of “global grid scheduler”. The scheduling of applications over the resources of the grid is termed as grid scheduling. The local scheduler controls only one single cluster and most of the times it possesses its own resource. The job of a grid scheduler differs from the local scheduler where it is responsible in performing resource detection, allocation of resources, scheduling of tasks and managing the execution of tasks on different organizational fields.

Schedulers: It is an application where the management of tasks is its responsibility. Some of the management issues are allocation of resources that are required by a certain task, scheduling the parallel task execution by dividing the task accordingly, event correlation, management of information, and service level management
capabilities. Later, the tree structure of these schedulers is developed with the root of the tree as meta-scheduler and leaves as the other low level schedulers. The service level requirements are used to assess the task given to the schedulers of the grid computing environments and the execution is carried out after the allocation of the particular resources.

**Resource Broker:** The service requester and the service provider are paired using the services of the resource broker. The particular task execution is carried out by choosing the most potential resources from the paired service provider. The pairing process is made more efficient by using the details collected from the corresponding resources by this resource broker.

The significant motivation behind a grid is to virtualize the resources to tackle issues. The key resources for which the Grid computing is designed to provide the access include but not restricted are:

- Computing/processing power
- Data storage/networked file systems
- Communications and bandwidth
- Application software

All the elements in the grid are equally significant enough and hence no single element can establish the grid on its own.
Having the capacity to bond together a few million workstations - groups, computers, supercomputers, desktop PCs - with information stockpiling, tools, graphical devices is the Fantasy grid processing attempts to accomplish and the truth of grid computing could be progressive in science. With a typical Grid framework, Grid scheduling is to respond to the query: in what way the given set of Grid applications (typically calculating escalated) is scheduled over various distributed resources?

There will be numerous tasks in every application. Each application has a number of tasks. A few essential queries need to be managed while mapping the tasks to the resources.

- How do the relations among different jobs influence scheduling resolutions?
- How does the variation of resources influence the execution of a schedule?
- What execution models are required to be utilized by a scheduler utilization to measure the excellence of a schedule?

Scheduling becomes more complex as the tasks are relying on one another. Actually, most of the research is carried out on the tasks that are not dependent on each other or task that can be divided into sub tasks or tasks that are not completely depending on other tasks. These methodologies commonly utilize either systematic routines to
create deterministic schedules (generally an ideal schedule) or exact information investigation and heuristic quest systems to search for a better result.

2.2 DISTRIBUTED COMPUTING TO GRID COMPUTING

During 1970s, when workstations were initially interfaced by systems, the thought of saddling distributed computing power was conceived. A couple of schedule tries different things with distributed computing, including a couple of projects called Creeper and Reaper, ran on the Internet’s antecedent, the ARPANET. In 1973, the Xerox Palo Alto Research Centre introduced the first Ethernet system and the first undeniable distributed computing exertion was in progress. Researchers J. F. Shoch and J. A. Hupp [18] made a project called "worm," which consistently travelled around 100 Ethernet-associated machines. These worms can travel from computer to computer utilizing static resources for favorable needs. Every worm utilized static resources to make a calculation and had the capability to replicate and communicate clones to different nodes of the system. Shoch and Hupp circulated realistic pictures using these worms and imparted calculations for interpreting accurate computer graphics.

In an alternate exertion, R. Crandall began giving static, organized NeXT machines to do some tasks. Crandall mounted software that permitted the computers, while not being used, to do processing and to integrate Endeavour’s with different machines on the system. His product, named “Zilla,” initially concentrated around
discovering, figuring, and validating more numbers, and afterward proceeds with test encryption [19].

Further investigation of distributed computing focused on utilizing a network connected cluster. In 1988, a software system named Condor is developed by M. Livny at the University of Wisconsin, Madison. Condor made the University’s idle machines to work to give High Throughput Computing (HTC) [20, 21, 22]. Likewise, in 1988, a software system is developed by A. Lenstra and M. Manesse of the DEC Systems Research Centre that distributed the jobs to workstations inside and outside their Palo Alto, California lab through email [23].

Distributed computing mounted to a worldwide level with the development of the Internet in the 1990s. In 1996, the Great Internet Messene Prime Search (GIMPS) [24] venture utilized distributed computing to scan for tremendous Messene prime numbers. C. Percival dispatched PiHex [25], an effective distributed computing exertion to compute the digits of π. Among the most of the newly introduced distributed computing ventures, two activities specifically have demonstrated that the idea works greatly well, far superior to numerous masters had expected. The primary of these progressive undertakings is distributed to utilize a large number of autonomously owned workstations over the Internet to crash encryption codes.

The Seti@home venture is the second, the best and mainstream of distributed computing ventures ever. In excess of two million
individuals, the biggest number of volunteers for any Internet
distributed computing venture to date, have mounted the Seti@home
software from the time when the venture began in May 1999. This
venture decisively demonstrated that distributed computing could
quicken figure the results of venture while managing the budget of the
venture.

Further, many more projects in numerous distinctive fields used
and exhibited the usefulness of distributed computing. In 2000,
Stanford researchers propelled Folding@home for protein portable
implementation. NASA propelled click workers, a venture to hunt
down hallows on Mars. Intel-United Device's Cancer Research Project
was propelled for hunting medications in cancer treatment. All these
activities utilize the Internet distributed computing standard.

As more applications utilize the universally distributed
computing method, numerous issues and difficulties also emerge.
Most importantly, how to proficiently accomplish and use the
boundless and broadly distributed dynamic resources turns into an
obligatory assignment to each internationally distributed computing
venture. Secondly, the distributed computing must stress over the
reliability of these calculations executed on a probable "untrusty"
machine. For instance, familiarity with Seti@home has demonstrated
that clients may forge the calculations and return wrong or mistaken
output to acquire more satisfied profits.
Thirdly, processing distributed computing environment needs to experience the issues that emerge from the conceivable unexpected inaccessibility of specific workstations or portions of the system. Consequently, execution, dependability, reliability, adaptability, and security get to be discriminating problems in applications of distributed computing. In huge scale distributed computing environments, these issues get to be more unexpected.

To address these issues, most of the later exertions in distributed computing are intended for creating a universally useful distributed computing framework, giving coordinated security, accessibility, versatility, unwavering quality, and reasonability for common applications of distributed computing. In the mid of 1990s, the word "grid" was devised by I. Foster and C. Kesselman in their book, "The Grid: Blueprint for a New Computing Infrastructure," to indicate a projected distributed computing framework for cutting edge of science and technology.

The growth of grid computing is very fast and is globally acknowledged in famous insight representing a system for “flexible, safe, synchronized resource distribution among active groups of individual perceptions and resources.

2.3 REVIEW OF GRID SCHEDULING ALGORITHM

Grid scheduling is characteristically more confused than neighborhood resource scheduling as it must control expansive scale
resources crosswise over administration limits. In such a vibrant distributed computing environment, resource accessibility differs significantly. So scheduling turns out to be truly difficult. A wide ranging research is carried out related to issues of scheduling in distributed systems. The literature survey presented in this chapter concentrates in 4 ways: static task scheduling, application-level scheduling, resource availability prediction, and economic methods in decentralized task scheduling system.

2.3.1 Static Task Scheduling:

Given a set of assignments and resources, a fixed scheduler processes the implementation strategy in advance of the run time. In static task scheduling [58], details about the resources and execution parameters are thought to be known. Considering upon how a task could be made into sub tasks, significant exploration might be sorted into two separate zones: divisible task scheduling, where the task is partitioned into subjective measured parts ("chunks"), and scheduling of the tasks that are fixed sized and are independent of each other.

- **Divisible Workload Scheduling:** Divisible workload scheduling [59] is depending on the Divisible Load Theory (DLT). By description, a separable task is persistent, which is a robust hypothesis in real-world scenarios. The objective of divisible workload scheduling is to reduce the overall execution time (makespan) by determining an ideal solution for partitioning the task into various true number-sized parts.
Independent Task Scheduling: Given a set of autonomous workloads and a set of accessible resources, independent task scheduling [60] endeavors to reduce the aggregate execution time of the workload set by discovering an ideal mapping of jobs to resources. The parameter used to discover such a mapping is the approximation of "turnaround time" or execution time (machine accessible time + anticipated time to process). Discovering the pre- eminent mapping is really a combinatorial optimization issue.

2.3.2 Application-level Scheduling:

Application-level scheduling [61] investigates the impact of distinctive application prerequisites on scheduling and recommends versatile methods of scheduling depending on runtime resource accessibility. The notion of application level scheduling is to implant scheduling into the application itself to create application aware resource varieties with the goal that it could choose the size of the task for every sub task depending on dynamic resource accessibility. Application-level scheduling aid applications change their schedule in two methods:

- An application can alterably deteriorate its workload and produce a schedule without any intervention to improve the general execution provided that the run-time resource accessibility data is known.
• For looping or approximately coupled parallel applications, given past workload execution data, an application can alter the schedule for the present workload.

2.3.3 Prediction Techniques:

The most generally demanded resources by a Grid application appeals as resisted, information, and system assets [63]. For a Grid resource, it is not difficult to get a machine's fixed resource data, for example, the frequency of the CPU, the size of memory, system transmission capacity, database and so on. At the same time runtime resources of the application, for example, the workload of CPU, memory that is available, and existing volume of the network are flexible in light of the fact that Grid computing environment is a dynamic resource sharing environment where numerous applications are being executed.

At time of scheduling choice making, exact resource expectation on run-time asset estimation parameters is precarious to enhance the performance of the application. In predication examine, the investigation is focused around recorded information on past resource accessibility data and records related to the performance of the task. Different factual procedures can put on the off chance that resource behavior shadows or is expected to take after certain appropriation; stochastic methodology investigation might be utilized to foresee forthcoming resource estimations on a static time point or in the course of a certain temporary of time. The Regression procedure might
be utilized for estimating the performance within the area of an execution model.

There are two approaches to foresee the estimated execution time of an application, in the event that there exists an execution model or not. On the off chance that no execution model is accessible, exact information exploration could be utilized. Utilizing this strategy, first discover past execution time records for comparable applications utilizing some searching technique focused around parameter classification principles, for example, permissible range of execution time.

At that point the mean execution time of past applications within an average failure rate is calculated and utilizes this assessment as the estimation of present application's execution time. The failure rate is processed by means of some factual estimation, for example, unqualified error, or minimum square root error. On the off chance that the application has an execution model, in the same way as seen inseparable task scheduling, run-time particular resource parameters can be estimated.

2.3.4 Economic Methods in Scheduling: The Grid environment as an extensive scale distributed system obscures the scheduling procedures and techniques [62]. However the expense of planning itself must be sensible to make it productive. Some cost-effective models that could be petitioned for supervising the nature of Grid resources are:
• **Commodity market model.** In this model, the cost is utilized as a solitary estimation to trade or purchase/offer resource [64]. Evaluating techniques are the significant concern of this model. Distinctive evaluating routines could be utilized, for example, level rate, user based, membership based (settled rate for a time to time), and interest supply-based. Naturally, cost is controlled by, however not restricted to the subsequent parameters: resource capacity, expense, request, client desire, and client inclination.

• **Posted price model.** This model is similar to a sales poster for the announcement of rebate or campaign [65]. In a business sector, customers find distributed sales data and contact makers straightforwardly to confirm and apply rebate or arrange campaigns. Sales promotion can happen, in an environment of Grid computing, when new services are offered by Grid and needs to fascinate clients, or when a Grid needs to amplify resource use amid off-crest time.

• **Bargaining model.** The Customer may discuss with producers for an appropriate cost [66]. In business, this frequently happens when the purchaser discovers a more reasonable cost from different makers (value match). In the environment of the grid, negotiating depends on various impartial functions of resource management and client of the resource. For instance, the owner of the resource may diminish cost for the assets with lower use or
poor execution. The resource can foresee a lower cost with guarantee to utilize more resource from this holder later on.

• **Tender/contract-net model.** In this model, an offering procedure is launched by customer [65]. Every producer reacts with their available items and proposed costs. The Customer thinks about each producer’s offer and picks leader. The ultimate output is an agreement.

• **Auction model.** This model is very famous for customers to offer on a thing promoted by a producer [67]. The practice is started by producer. There are numerous customary auction techniques, for example, English auction, first-value fixed offer auction, Vickrey auction, Dutch auction, and twofold bartering.

The task scheduling is one of the fundamental components of parallel and distributed computing. With the development of grid computing, The new scheduling algorithm is required for grid environment. The diminishment of turnaround time is a specific end goal in scheduling the task in parallel. The primary disadvantage of parallel scheduling is disregarding the particular shared nature of the resource.

In a computational grid, scheduling issue is improved by amplifying framework usage and satisfies cost-effective framework and client stipulations. Job scheduling in grid is an incorporated part of computing and deals with fixed or vibrant strategies, application
prototypes and QoS imperatives like transmission capacity, goal capacities and resource vibrant behavior. There are many procedures in the literature based on job scheduling to decrease the execution time.

With the bang of the Grid, there are two notions that are required to be taken into point in a scheduling algorithm, for example, non-committed system and Quality of Service. In a non-faithful system, clients have their own particular local tasks and can’t give elite administrations to remote tasks. Anticipation is required in the case of calculation time for non-faithful systems. In a computational network, it is attractive to seek the best QoS for distant resources to satisfy application demands. He Xiao shan, Xian-He Sun., [68] have expressed that the scheduler in the nature of Grid requires application and QoS imperatives to be considered to improve mapping in between resources and applications. Forecast model might be implemented to address the difficulties of the non-faithful system.

2.3.5 Work flow Scheduling

The Work process is a methodology in which different archives, tasks, data or information is passed starting with one candidate, then onto the next for activity as per a set of procedural principles. The capacity to process occasional serious work processes decides the accomplishment of e-business. The most conspicuous feature for these work processes is the immense amount of simultaneous basic work process examples and their serious competition of resources.
Work processes are perplexing applications which might be deteriorated into reliance tasks such that fulfillment of one task takes after other. At the end, the entire work process could be executed by finishing all the task execution orderly. An assignment is little measure of work which obtains information from the execution of past assignment and may give the output to successor assignments.

The issue of planning assignments that are dependent could be planned as $Rvm|prec|pmtn$ or $Rvm|prec|npmtn$. Schedule the assignments on irrelevant analogous virtual machines. It is denoted as 'Rvm'. The assignments are related to one another. It is signified with antecedence relation 'prec'. 'pmtn' demonstrates the jobs distributed to a machine might be seized and may be allocated to different machines also. "nptmn" demonstrates that the jobs distributed to a machine can't be appropriated relegate to different machines unless a break down happens.

![Diagram of Scheduling Model](image)

Figure 2.1 Scheduling Model
Figure 2.1 demonstrates the scheduling prototype improved from traditional frameworks to cloud computing (Manimaran, G and Murthy, C. S. R., [26]). VM1, VM2, VM3 are the virtual machines. The highly-heuristic methods for productive scheduling of resources in grid can additionally be reached out to clouds (Bhanu, S.m.s and Gopalan, N. P., [27]). The stages of scheduling are given below:

1. Resource Discovery: detecting the presently idle resources (without being distributed to any task) in the framework.
2. Resource Selection: Selecting one of the idle resources depending on certain fundamental procedure to schedule it to a task on the queue which is active.
3. Job Execution: Assigning the selected resource to a task and executing the task

A Directed Acyclic Graph is used to denote the tasks that are dependent. A DAG G= (V, E) where V is the set of nodes n, of the graph, signifying the jobs to be prepared, and E is the set of edges, signifying the reliance among jobs. A flow of work is a DAG which comprises of jobs as nodes and the trust as edges.

- The dependency of a task can be on more than one task (Predecessor) i.e., the task that is dependent on the other can start its execution only when its predecessor has completed its execution successfully.
- The scheduling of the predecessor needs to be done before scheduling the tasks that are dependent on it.
• As the workflow scheduling is complicated, the jobs might be arranged in a simpler approach to make this scheduling process consistent and financially suitable.

• This arrangement serves to verify that an assignment needs to be extracted after the submission of its antecedent job.

• The assignments are arranged progressively on the basis of height.
  o (height of a specific assignment must be more than all its antecedents. i.e., the height 0 assignment is antecedent of height 1)

While task scheduling is being performed, height turns into a key variable. The scheduling happen based on length, the task will be viewed just when all its antecedents have been executed. In this way, at a specific height all the tasks that are available are not reliant on one another. Henceforth at every level, the task scheduling is done irrespective of the dependency. The scheduling of workflow at any specific height might be dealt as autonomous task scheduling.

2.4 RELATED WORK ON SCHEDULING WORKFLOWS

A wide range of scheduling algorithms for cloud, grid and distributed computing settings has been exhibited in the previous research. Each of these systems has its own benefits and weaknesses.
2.4.1 Adaptive Scheduling Algorithm

The Adaptive Scheduling Algorithm (ASA) was proposed to discover a suitable order for implementing the work process exercises. This algorithm reflected the work process for catastrophe situations. The ASA classifies the whole scheduling procedure into 2 stages: (1) a logical partition step and (2) a physical distribution step. The key inspiration driving this multistage methodology is to diminish the intricacy of scheduling issues. The logical partition step relegates these segments by utilizing stipulation programming. In the meantime system progressions are measured in ASA to make it a vigorous and versatile scheduling of single complex DAG; they are not appropriate for scheduling illustration exhaustive work processes empowered on a distributed computing environment. The fundamental feature of point concentrated work processes is that it holds an immense number of conceivable relative basic simultaneous occurrences.

2.4.2 Adaptive Rescheduling Algorithm

The versatile scheduling algorithm was proposed by Yu and Shi [28]. This algorithm adjusts heterogeneous fastest completion time, to actualize the rescheduling method in the work processes. It takes after the static methodology to schedule the applications of work process and states that the static scheduling is best for the work process planning. They present the organizer which listens for the incident of attention. The events are two kinds: alteration of resource pool and variations in the resource performance. Taking into account
the data it gets, the organizer reschedules the job once more depending on the information it obtained. Till all jobs complete its execution the organizer continues tuning in. The disadvantages here are the rescheduling of assignments over and over. The methodology takes after just static system which does not consider changing nature of the processor’s ability.

### 2.4.3 Resource Allocation for Workflows

Guo, W et al [29] projected two algorithms for optical network work processes. They are job based assignment and work process based assignment. They related both algorithms and infer that the job based methodology is appropriate for processing concentrated jobs and work process based methodology is appropriate for information escalated assignments. The methodology for assignment based portion is Min-Min and it is local. It won't utilize resources completely at every neighborhood state. The work process based approach thoroughly thinks about the numerous options for work process scheduling and oblige more number of cycles.

### 2.4.4 Static Scheduling of Department Parallel Tasks

This strategy was proposed as a procedure to statically plan the similar assignments that are dependent on assorted processing users. Both the Job parallelism and information parallelism are considered by the procedure.
The following are the steps of the scheduling algorithm a) identify the tasks that are available in the schedule. b) The priority is given to the task identified and c) chooses the task that has high preference and assign to one or more processors. d) The step ‘c’ is repeated till all the tasks are allocated. These steps are focused around the list scheduling method.

2.4.5 Non-Evolutionary Random Scheduling Algorithm

Boyer, W.f and Hura, G. S., [30] proposed a two different algorithms as randomized search algorithm and dependent task scheduler. These algorithms were projected for relating and scheduling of the tasks that are interdependent on one another in an appropriated heterogeneous computing system. The tasks are mapped to the schedule by the dependent task scheduler. It is building piece of the arbitrary search algorithm. The Min-span algorithm is generalized by this algorithm. This system needs a smaller amount memory than non-search algorithm. But the computation time of a schedule is more and the average performance also is less.

2.4.6 Myopic Algorithm

In Myopic algorithm [69], the task that is prepared is scheduled provided that its parent tasks have completed their scheduling and about to complete the task sooner. This process is repeated throughout the procedure. This algorithm is not appropriate for the scheduling of group of tasks as it is concentrates on the single task.
2.4.7 Ant Colony Optimization Algorithm

The various QoS parameters like time, reliability and cost are considered in ACO algorithm. Clients are permitted to characterize QoS stipulations to ensure the eminence of the schedule. This chapter presents 7 heuristics. Also, the aim of the procedure is depending on the client characterized QoS inclination (Chen, W.n and Zhang, J., [31]). Similarly, the client needs to set the lowest threshold value for the desirable QoS. The looping phases of the ACO require considerable amount of time and hence not appropriate for scheduling instance-intensive tasks as they comprise a number of simultaneous instances.

2.4.8 Dynamic Level Scheduling

This is a compile time scheduling heuristic (Sih, G. C and Lee, E. A.,[32]). During the scheduling process of tasks that are dependent to the heterogeneous processors, the inter-process communication is taken into consideration. Routing and scheduling are carried out at the same time and hence it is a time taking process. The mapping of the task to a processor requires the re-routing of data transmission.

2.5 RELATED WORK ON SCHEDULING HEURISTICS

In general, there are four classes of scheduling heuristics for workflow applications, namely individual task scheduling, list scheduling, cluster and duplication based scheduling. The detailed analysis of workflow scheduling heuristics is shown in figure 2.2.
Figure 2.2: A taxonomy of grid workflow scheduling heuristics [89]

*Individual task scheduling:* The individual task scheduling is the simplest scheduling method for scheduling workflow applications and it makes scheduling decision based only on one individual task. The Myopic algorithm [74] has been implemented in some Grid systems such as Condor DAGMan [75].

*List scheduling:* A list scheduling heuristic prioritizes workflow tasks and schedules the tasks based on their priorities. Min-Min, Max-Min, Sufferage proposed by Maheswaran et al. [76] are three major heuristics which have been employed for scheduling workflow tasks in vGrADS [77] and Pegasus [78]. The heuristics are based on the performance estimation for task execution and I/O data transmission.
Cluster based and Duplication based scheduling: Both cluster based scheduling and duplication based scheduling are designed to avoid the transmission time of results between data interdependent tasks, such that it is able to reduce the overall execution time. Sources Bajai and Agrawal [79] proposed a task duplication based scheduling algorithm for network of heterogeneous systems(TANH). The algorithm combines cluster based scheduling and duplication based scheduling

Meta-heuristics: Meta-heuristics provide both a general structure and strategy guidelines for developing a heuristic for solving computational problems. They are generally applied to a large and complicated problem. They provide an efficient way of moving quickly toward a very good solution. Many meta-heuristics have been applied for solving workflow scheduling problems, including GRASP (Greedy Randomized Adaptive Search Procedure) [80], Genetic Algorithms [81] and Simulated Annealing [82].

Important heuristics for scheduling workflows are explained below:

Latest completion time (LCT): LCT is the time on which the critical task finishes its execution. A non-critical task can extend its execution till local critical completion time (LCCT) but it should not exceed this time. LCCT is the completion time of a critical task of particular height (local).

\[ LCT(T_{mc}) = LCCT \]
Where $T_{nc}$ is non critical task

**Latest start time (LST):** LST is the time when the non-critical task starts its execution, so that it completes execution on or before the completion of critical tasks. LST [83] is the maximum time a task can delay its execution on that particular machine. The latest start time varies between the machines for the same task.

$$\text{LST}(T_{nc}, VM_i) = \text{LCT}(T_{nc}) - C(T_{nc}, VM_j)$$

This value may be negative when the task completion time is higher than the completion time of critical tasks. Where $C(T_{nc}, VM_j)$ is the completion time of non-critical task at VM $VM_j$.

**Earliest start time (EST):** EST is the time when execution of all predecessors of a task is complete so that the task under consideration is ready to execute.

$$\text{EST}(T_h) = \max \{ C(T_{h-1}, VM_j) \mid T_{h-1} \text{ is the predecessor of task } t_h \}$$

**Earliest completion time (ECT):** The earliest possible time of a task can be executed given that all machines are ready to execute.

$$\text{ECT}(T_{nc}) = \min \{ C(T_{nc}, VM_j) \}$$

**Opportunistic Load Balancing:** Opportunistic Load Balancing (OLB) assigns each task, in arbitrary order, to the next machine that is expected to be available, regardless of the task’s expected execution time on that machine. The intuition behind OLB is to keep all machines as busy as possible [84].
**MCT:** Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task. This causes some tasks to be assigned to machines that do not have the minimum execution time for them [85].

**Min-Min:** Min-min heuristic uses minimum completion time (MCT) as a metric, meaning that the task which can be completed the earliest is given priority. This heuristic begins with the set $U$ of all unmapped tasks. Then the set of minimum completion times ($M$), is found.

$$M = \left\{ \min(\text{completion time} (T_i, M_j)) | T_i \in U \right\}$$

$M$ consists of one entry for each unmapped task. Next, the task with the overall minimum completion time from $M$ is selected and assigned to the corresponding machine and the workload of the selected machine will be updated. And finally the newly mapped task is removed from $U$ and the process repeats until all tasks are mapped [86].

**Max-min:** The Max-min heuristic is very similar to min-min and its metric is MCT too. It begins with the set $U$ of all unmapped tasks. Then, the set of minimum completion times ($M$) is found as mentioned in min-min. Next, the task with the overall maximum completion time from $M$ is selected and assigned to the corresponding machine, and the workload of the selected machine will be updated. And finally the
newly mapped task is removed from \( U \) and the process repeats until all tasks are mapped [87].

**LJFR-SJFR:** Longest Job to Fastest Resource-Shortest Job to Fastest Resource (LJFR-SJFR) heuristic begins with the set \( U \) of all unmapped tasks. Then the set of minimum completion times is found the same as Min-min. Next, the task with the overall minimum completion time from \( M \) is considered as the shortest job in the fastest resource (SJFR). Also the task with the overall maximum completion time from \( M \) is considered as the longest job in the fastest resource (LJFR). At the beginning, this method assigns the \( m \) longest tasks to the \( m \) available fastest resources (LJFR). Then this method assigns the shortest task to the fastest resource and the longest task to the fastest resource alternatively. After each allocation, the workload of each machine will be updated [88].

### 2.6 TIME COST OPTIMIZATION

The set of resources is considered by the task scheduling algorithm with the objective of reducing the execution time as much as possible. Nonetheless, the quantity of VMs being used for distribution is minimized using the proposed algorithm. A Balanced Time Scheduling (BTS) is a heuristic algorithm presented by Byun. The given task is to complete its process within a specified duration. In this regard, BTS approximates the least required number of computing resources. The BTS is economical, standard and scalable. As all the resources are not being utilized during the period of hiring
by the BTS, there is possible wastage of resources in the static allocation procedure. The critical path analysis is used for approximating the required resource quantity in a fundamental procedure proposed by Sudarsanam. Upgraded fractional basic path is the basis of this scheduling algorithm. The optimal one is determined by computing the execution time and resource utilizations iteratively. A common method of scheduling the bi-criteria task depending on the dynamic programming is proposed in Wieczorek. This procedure is depending on sliding obligation. The best among the candidate schedules is determined by producing and verifying the schedules repeatedly. The least possible quantity of resources required to complete the task before the least potential make span is determined by the procedure projected by Huang.

The number of resources required is computed by exact information accumulated from numerous example work processes, by changing such parameters as the size of the DAG, the correspondence and computation ratio, etc.

(Sulistio, A and Buyya, R [33]) projected a vibrant auction based scheduling methodology to reduce the time to execute the client applications receiving the corresponding portion distribution. Regardless of the possibility that this methodology goes for reducing the expense, it doesn't guarantee the application accomplishment within a fixed financial plan, besides it can cause in more overhead due to the constant intercession of the agent to select the least
expensive resources. Static performance driven planning of tasks portrayed by a series of parameter sweep jobs in a heterogeneous surroundings had suggested. An adaptable time-cost procedure for scheduling non-preemptive tasks that are dependent on one another in distributed computing work streams is proposed.

2.7 POWER MODELS

The particulars of the power models are presented in this section.

2.7.1 Introduction

The approximation of the power consumed by the system in progress is the initial step in the design of the system which reduces the power utilization. The initial stages of the system design should carry out the process of analysis where worthy thoughts on elevating power degeneracy can motivate the selection among various frameworks.

The accuracy of the power analysis performed at lower levels is not satisfactory when compared to the analysis at the system level as the information related to the actual realization of the process are not completely defined at lower levels. Figure 2.3 precise these ideas or notions. There are two methods to approximate the power dissipation at various levels of perception. They are methods based on the simulation and the probabilistic methods.
Figure 2.3: Power Analysis and Optimization at different levels of the design [70].

- **Simulation-Based Methods**: The power dissemination is attained by applying particular data examples to the project. Therefore; the approximation is not just influenced by the precision of the prototype depiction, yet on the input designs as well. The data examples ought to be strictly identified with the genuine application in which the system is connected. Simulation based techniques are broadly utilized, as they are strictly identified with the timing and efficient implementation and verification of the framework.

- **Probabilistic Methods**: The techniques need the particulars of the average conduct of the input designs from their probabilities. Thus, it is conceivable to protect an extensive number of examples with inadequate computational exertion. The exchanging movement, important to perform power estimation, is figured from the signal probabilities of the circuit.
nodes. Methodologies to such strategies are characterized by probabilistic implementation, figurative model, and implementation of fluctuating concreteness.

### 2.7.2 Power Estimation Models

Numerous merged and correct devices, evaluate the power dissemination from RTL to circuit level. However, at more elevated amounts an extensive exploration is still to be carried out. Power models are categorized on the base of the deliberation level of the representation of the framework and are explored below:

- **TRANSISTOR LEVEL POWER ESTIMATION**
  A faultless approximation of power utilization might be completed at the transistor level, implementing the continuous behavior of the system and examining the source power by utilizing SPICE-like test systems. The CPU time required for the implementation is amazingly high, making the implementation conceivable just for the systems with hundreds of transistors and less input designs.

- **GATE LEVEL POWER ESTIMATION**
  It is conceivable to investigate the behavior of the system with the use of digital simulators at gate level, if one has points of interest of the single logic gate. The approximation of power utilization is attained by utilizing exchanging movement and single node limit utilizing the relationship reported as a part of the Equation (2.1). At this level the consequences of the power
estimation determinedly relies upon the deferral model used, that may effectively evaluate the vicinity or nonattendance of glitches. In a "zero delay" model all movements happen at the same time. Glitches are not considered, so the power estimation is exceptionally hopeful.

- **RT LEVEL POWER ESTIMATION**

  Multipliers, adders and registers can be used to approximate the power at register transfer level (RTL). The basis of imprecision at this level relies on the poor displaying of vibrant impact (eg. glitches), causing a wrong estimation of the exchanging action, and on the poor depiction information of the practical pieces and interconnections with a resulting imprecision approximation of the capacitances.

  Some scientific routines at RTL use multifaceted nature, or a comparable gate count, as a capacitance approximation. Thus the power disseminated by a block might be approximately predicted as the product of gates and the power utilization of a solitary reference gate; a fixed activity factor is expected.

  A few strategies are focused around analytical macro models (straight, piecewise direct, spine) of the power dissemination of each one piece. The model fits the test information acquired from mathematical simulations at lower levels or exploratory information. The model is influenced by a fault inherent in the model, by an approximation faults because of the restricted amount of trials and by a failure because of the reliance of the estimations on the input
designs. The model could be characterized by the Eq. 2.1 or as a multi-dimensional look-up table (LUT).

- **SYSTEM LEVEL POWER ESTIMATION**

  System level power estimation depends upon the power investigation of the equipment and programming parts of the system. The segments in a framework level depiction are chip, DSPs, buses, peripherals, whose interior design is, generally not characterized. Battery, power dissemination and cooling framework demonstration ought to be considered at this level.

  Power approximation is profoundly imprecise in light of the fact that the complete structural planning of the framework is not characterized. Alternately, outline investigation opportunity is high and hence is the optimization of power. At this level of deliberation power approximation generally is performed for the assessment of distinct architectures of the framework, so as to select the optimal one as far as power utilization as well.

  To facilitate power approximation, a prototype of the power disseminated by every single block is made and the coefficients of the prototype are evaluated from the data determined from the lower levels. The system level power model could be inferred from the power dispersal of the single CMOS device [71], as given in Equation (2.1), and might be characterized by the relationship given below

\[
E = N (CV_{DD}^2 + Q_{SC}V_{DD}D + I_{\text{leak}}V_{DD}T)
\]  

(2.1)
Where

\( V_{DD} \) – The supply voltage,

\( D \) – The average number of commutations of the gates of the block,

\( N \) – The number of gates,

\( C \) – The average capacitance of the gates,

\( Q_{SC} \) – The average charge lost due to short-circuit current during commutation,

\( I_{\text{leak}} \) – The average leakage current of the block,

\( T \) – The mean energy dissipated during a time period.

The mean amount of replacements ‘\( D \)’ must be computed while the framework level implementation and thus relies on the particular application and investigation path. The coefficients ‘\( C \)’, ‘\( Q_{SC} \)’ and ‘\( I_{\text{leak}} \)’ are associated with the particular technology selected. ‘\( N \)’ is the number of proportionate gates important to execute the block portrayed at the framework level,

On the off chance that the block has not yet been executed, the intricacy of the block, that is, an approximation of the number of gates needed for its execution, ought to be given. Obviously, if the comprehensive design of the framework is not yet characterized, just an approximate valuation might be given. A sample of this method is given in Figure 2.4. From the "SystemC" code of every module the
amount of proportionate gates needed for the execution of the module is evaluated

Figure 2.4: Complexity estimation from ‘SystemC’ source code

The mathematical operations on different SystemC types (scint, scuint, scbigint, scbiguint, sc fixed, scufixed, sc fix, scufix), the bitwise and comparison operators and assignments and the c++ control instructions (if else, switch case, for and while) is documented and a segment from a library of a reference technology is connected to each operator. Programming procedures have been established to provide these consequences in a spontaneous way. The system presented a power analysis based on the instructions and made functional in numerous additional tasks. Instruction- based power analysis associates a vitality model to every direction, for instance, the one stated in Equation (2.1). The power model ought to be parametric so as to permit the reuse of the IP practical depiction, as well as of the power prototype.

Power structural designs with a specific end goal to achieve the anticipated trade-offs regarding distinctive parameters like velocity,
throughput and power utilization. The comprehensive stages of instruction based power demonstration and investigation are stated in Figure 2.5.

![Diagram of system level power modeling and analysis](image)

Figure 2.5: System level power modeling and analysis

### 2.8. POWER MANAGEMENT TECHNIQUES

Different power management methods are described in this section.
2.8.1 Clock Gating

Clock gating is an ordinarily applied method used to decrease the power by gating off clock signals to registers, latches, etc. Gating may be carried out when there is no obliged action to be executed by logic whose inputs are determined from group of memory components. Meanwhile, new yield values from the logic are neglected; the memory components serving the logic could be hindered from overhauling to avoid insignificant exchanging actions in the rationale. Clock gating may be used at the function unit level for governing exchanging action by restraining input upgrades to function units. It may not be possible to apply clock gating to single memory components because of overhead in producing the enable sign, although toward self-gating memory components have been suggested that compare the present and next state values to permit local clocking. The thought of crippling the clocks to unused units to lessen power dissemination in microchips has been examined in (William J. Dawdle And Charles L. Seitz, [34]). In the CAD group, comparable strategies have been exhibited at the regional level of configuration. Protected assessment tries to alertly recognize which parts of a logic circuit are constantly utilized and which are not. Logic pre-computation looks to infer a pre-computation circuit that under unique conditions does the calculation for the rest of the circuit. Both these systems are similar to contingent clocking, which might be utilized at the structural level to diminish the power by impairing unused units. (William J. Falter., [35])
demonstrated that clock gating can essentially diminish power utilization by impairing definite practical units if instruction decode showed that they won't be utilized.

2.8.2 Power Gating

Traditionally, the essential basis of power dissemination in CMOS transistor devices has been the vibrant exchange because of charging/ discharging load capacitances. In any case, poorer supply voltages must be coupled with poorer transistor threshold voltages to keep up high exchanging rates needed for complex implanted applications. The International Technology Roadmap for semiconductors forecasts an enduring scaling of supply voltage with an analogous decline in transistor threshold voltage to keep up a 30% enhancement in the execution of each era.

The disadvantage of edge scaling is a growth in spillage power dissemination because of an exponential expands in sub-limit spillage current actually when there is no switching in the transistor. The evaluating factor of 7.5 expands in spillage current and a five- fold growth in the aggregate spillage power dissemination in every chip era; thus, the requirement for power gating to spare spillage power.

(G. Palermo and C. Silvano, [36]) proposed a novel power gating instrument for instruction caches, which approximates vibrantly and adjusts to the desired size of instruction cache, and switch off the power supply to the unused SRAM cells of the cache. Likewise, power
gating may be used to any core which is idle, or bank of cache, or practical units in a processor with multiple cores. However the rise and fall of power supply as a phase of power gating is commonly done over hundreds and a large number of clock cycles evade unexpected growth or reduction of power when gates change on or off correspondingly.

2.8.3 Dynamic Voltage and Frequency Scaling

Dynamic Voltage and Frequency Scaling (DVFS) was presented in the 1990s (S. Gupta and F. N. Najm [37]), providing an extraordinary guarantee to significantly lessen the power utilization in huge computerized frameworks (containing cores of the processor, memory banks, buses, and so forth) by adjusting both power and recurrence of the framework concerning altering loads. DVFS control procedures could be executed at diverse levels, for example, in the processor micro-construction modeling. Regrettably, the full guarantee of DVFS has been impeded by show off-chip voltage controllers that fail to offer the capability to adapt to distinctive voltages at small time scales. Cutting edge executions are constrained to transiently coarse-grained adaptations administered by the operating system.

Later, there was a flow of enthusiasm toward on-chip switching power controllers. These controllers offer the possibility to give various on-chip power areas in future multi-core embedded processors. An on-chip controller, working with high exchanging frequencies, can
deter massive channel inductors and capacitors, permit the channel capacitor to be incorporated altogether on the chip, place more tiny inductors on the package, and empower quick voltage transitions at nanosecond time scales. Besides, an on-chip controller can undoubtedly be separated into different parallel duplicates with minimal overhead to give numerous on-chip power domains.

In (Tony Givargis et al., [38]), the authors depict and model these expenses, carry out an exhaustive investigation of a CMP framework with on-chip included controller and project an offline integer linear programming based DVFS procedure utilizing the processor with multiple core test system. They infer that on-chip controller can altogether enhance DVFS viability and lead to complete framework power savings in a CMP. However, designers should precisely represent overheads and expenses when planning the cutting edge DVFS frameworks and procedures.

2.8.4 Smart Caching

Cache memories in installed processors assume a noteworthy part in deciding the performance metric of the power. In this section, the two techniques for sparing power in embedded smart caches are presented: cache set estimation and low power cache coherence protocols. In (P. Kermani and L. Kleinrock., [39]) the authors utilize two systems proposed before, path prediction and specific direct mapping, to diminish L1 vibrant cache power while conserving the performance. Path prediction and selective direct mapping foresee the
corresponding path number and give the estimation before the access of cache, as opposed to attend to the tag array to give the path number as in the case of sequential access.

(Andrew Laffely et al., [40]) presented the cache coherence which is power productive. Snoop based cache coherence executions utilize different types of hypotheses to lessen cache miss delay and enhance performance. They stated just a 6.25 percent increment for mean cache miss delay for SPLASH2 benchmark while accomplishing significant decreases in snoop-related action and power dispersal.

2.8.5 Scheduling

DVS and dynamic voltage and frequency scaling (DVFS) procedures have prompted radical decreases in power utilization. Nonetheless, power supply has immediate effect on processor speed, and thus, on the constant execution of an embedded system. Therefore, classic scheduling of tasks, frequency scaling and preferred power supply must be tended together.

Scheduling deals an alternate level of conceivable outcomes for accomplishing vitality and power effective frameworks, particularly when the framework structure is static or the framework displays the radical behavior. For such dynamic frameworks, different power administration strategies are present and are studied for instance. However, these predominantly objective soft real-time frameworks, where the lost in QoS is to be maintained.
Task level voltage scheduling choices can further diminish the power utilization. Few of these intra task scheduling systems utilize many rescheduling facts inside a task, and are normally compiler aided.

Then again, setting the schedule before the jobs begin executing as in (NaguDhanwada et al., [41]) eradicates the internal scheduling overhead, yet with conceivable loss of power effectiveness. Facts might be utilized to exploit the vibrant behavior of the framework, both at job level and at job-group level utilizes stochastic information to infer effective power plants without the overhead of intra-task rescheduling for hard real-time scheduling systems, where each due date must be met.

2.8.6 Comparisons of Power Management Methods

Clock Gating

Pros: Simple additional gating logic
Cons: Leakage power dissipation, noise

Power Gating

Pros: No leaking
Cons: Complex additional on-chip p/g switching logic,
High power/ground, noise

DVFS

Pros: Good controllability between power and performance,
Low p/g, noise
Cons: Complex additional on-chip p/g voltage regulators required

**Smart Caching**

Pros: Software controlled, some level of optimization possible between power and performance

Cons: Cache logic increases Verification of coherence protocols

**Scheduling**

Pros: Global power optimization possibly unlike all other methods. Good control over p/g, noise

Cons: Kernel or user code has to be changed