Chapter 1

Introduction

Over the decades, distributing computing plays a vital role to solve the large commercial, scientific, and mathematical applications. In distributed computing, an application is divided into various tasks and executed on appropriate processors which are distributed over a network. The tasks of the application may be independent or dependent. The independent tasks execute separately on the processors and do not communicate with each other. The dependent tasks are interrelated and may execute when their precedence constraints are satisfied, and any task of the application may communicate with other tasks during execution. The application having dependent tasks is represented by a Directed Acyclic Graph (DAG) [1] where tasks of an application are denoted by nodes and interrelation among the tasks is indicated by directed edges. The competence of computing parallel and distributed applications is extremely dependent on their features, for example, size of data to transfer between tasks, computation time of tasks, precedence constraints among tasks, etc. and platform features such as number of processor, execution power of the processors, link capacity, etc. The processors involved in distributed computing may be homogeneous in which all processors have same processing capabilities, or heterogeneous in which each processor has different processing capabilities. The system performing distributed computing may include multiprocessors, GPUs, multicore processors or a combination of these and involves potentially a great deal of communication overhead which restricts the performance of applications if tasks are not scheduled efficiently. An efficient schedule for a parallel and distributed application significantly relies on the techniques employed to schedule the tasks on to

processors of such systems with the aim of minimizing schedule length or makespan of the application. Another issue of interest may be minimization of energy while scheduling tasks. The minimization of schedule length helps to enhance the processor utilization and system throughput. The problem of obtaining schedules with minimum length has been shown to be NP-complete [2–6]. In an attempt to give polynomial time solutions, the known algorithms end up providing at times suboptimal solutions.

To solve the problem of task scheduling, the known algorithms are grouped into static and dynamic scheduling algorithms [7, 8]. The static scheduling algorithms use all information regarding tasks in advance, such as computation time of tasks on processing units and communication time between tasks, etc. before starting the execution of the application while dynamic scheduling algorithms utilize the required information for scheduling decisions at run-time. There are many methods to assess such information [9]. This thesis focuses on static scheduling algorithms as it generates optimal schedules without considering run-time overheads. Static scheduling algorithms may be grouped into guided heuristic-based algorithms or random search-based algorithms. The second group of the static algorithm takes more time to find the required solution even though the makespan is minimized at the cost of spare time. It gives different makespan for the same problem size and the same inputs based on the various scheduling techniques are used to allocate parallel tasks onto the suitable processors. Alternatively, the first group of static algorithm focuses on generating schedules with the minimum scheduling overhead, but the obtained makespan is not necessarily the shortest. Hence, both heuristic-based and guided random search-based algorithms can be used as per circumstances [10]. The heuristic-based algorithms are further divided into three subgroups that are list scheduling, duplication-based scheduling, and clustering-based scheduling. The list scheduling algorithms [7, 9–16], typically schedule tasks in 2 phases: the primary phase is task prioritization in which tasks are assigned priorities based on their associated execution and communication times; processor selection being the second phase in which suitable processors are selected and task assignment to processors is done. The duplication-based algorithms [2, 17–22] attempt to minimize the communication time between tasks through duplication of tasks onto different processors. The clustering-based algorithms [3, 11, 14, 23–27] are mainly applicable for an unbounded number of processors and generate schedules by grouping heavily

communicating tasks of a given application into clusters and assigning these clusters onto appropriate processors so that the schedule length of an application can be minimized. In this thesis, we focus on clustering-based algorithms.

This thesis attempts to address the problem of scheduling tasks having precedence-constraints for distributed computing on multiprocessors. It performs benchmarking of some well-known task scheduling algorithms and explores the possibility of a framework for this. The thesis proposes interesting intuitive ideas that lead to two scheduling algorithms for minimizing schedule length of applications and another energy-aware scheduling algorithm for minimizing the energy consumption by processors in the considered system. We present experimental results for the proposed algorithms and compared them with the existing algorithms for different types of applications. The results illustrate that the proposed algorithms perform better than some representative known algorithms.

1.1 Background

For sake of completeness, we are here giving a brief description regarding distributed system, that is required for distributed computing. We are also discussing about task scheduling in distributed computing.

1.1.1 Distributed Systems

Distributed computing is computing performed in a distributed system [28]. According to Tanenbaum and Steen [29], a distributed system is defined as "A distributed system is a collection of independent computers that appears to the user as a single coherent system" and according to Colouris et al. [30] it is defined as "A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages".

In distributed systems, various nodes act autonomously and cooperate with each other, which can achieve the purposes of resource sharing, openness, concurrency, scalability, fault-tolerance, and transparency [30].

In summary, distributed systems have the following general characteristics:

- Concurrency of components
- Lack of Global clock
- Independent failures of components

Issues in Distributed Systems: Distributed systems are more complex than systems that run on a single processor. Complexity arises because different parts of the system are independently managed as is the network. There is no single authority in charge of the system so top-down control is impossible [30].

- Transparency: To what extent should the distributed system appear to the user as a single system?
- Openness: Should a system be designed using standard protocols that support interoperability?
- Scalability: How can the system be constructed so that it is scalable?
- Security: How can usable security policies be defined and implemented?
- Quality of service: How should the quality of service be specified?
- Failure management: How can system failures be detected, contained and repaired?

The benefits of distributed systems are that they can be scaled to cope with increasing demand, can continue to provide user services if parts of the system fail, and they enable resources to be shared.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other via message passing.

1.1.2 Distributed Computing versus Parallel Computing

In distributed computing, each processor has its own private memory and information is exchanged by passing messages between the processors while in parallel computing, all processors may have access to a shared memory to exchange information between processors [28, 31].

1.1.3 Task Scheduling in Distributed Computing

Based on the certain parameters, task scheduling in distributed computing can be divided into the following [7, 32–38]:

- **Static versus Dynamic Scheduling:** Based on the knowledge about the task characteristics, task scheduling in distributed computing can be divided into static and dynamic scheduling. In static scheduling, all information regarding tasks are known in advance, such as execution time of tasks on processors, communication time between tasks etc. before starting the execution of the given application while on the other hand, dynamic scheduling uses the required information for scheduling decisions at run-time. The dynamic scheduling incur more overhead than the static scheduling as it has to collect, store and analyze state information. The static scheduling is also known as compile-time scheduling whereas dynamic scheduling scheduling is known as run-time scheduling.

- **Uniprocessor versus Multiprocessor Scheduling:** Based on the number of processors, task scheduling in distributed computing is divided into uniprocessor and multiprocessor scheduling. In uniprocessor scheduling, the system consists of a single processor and all the tasks are scheduled on this processor. On the other hand, a multiprocessor consists of a number of processors on which the tasks may be scheduled. Multiprocessor systems offer the advantage of task parallelism between the tasks. When two tasks assigned to different processors communicate with each other, then inter-processor-communication (IPC) comes into the picture.

- **Preemptive versus Non-preemptive Scheduling:** Based on whether preemption is allowed or not, task scheduling in distributed computing can be divided into preemptive and non-preemptive scheduling. In preemptive scheduling algorithms, once a task has started execution on one of the system processors, it can be interrupted in order to execute another task. However, a task that has started execution has to run till completion in the non-preemptive scheduling model. The task cannot be stopped in between to execute another task.

- **On-line mode versus Batch mode Scheduling:** Based on whether task is considered alone or in group for scheduling, task scheduling in distributed computing can be divide into on-line mode and batch mode scheduling. In on-line mode, a task is scheduled on to a processor as soon as the task arrives at the scheduler. On the other hand, in the batch mode, tasks are not scheduled on to the processors when they arrive; they are collected as a batch and then this batch of tasks is considered for allocation. on-line mode scheduling is thus quick whereas batch mode scheduling can take better advantage of task and resource characteristics in deciding which task to map to which resources.

- **Single-criterion versus Multi-criteria Scheduling:** Based on the number of scheduling criteria, task scheduling in distributed computing can be divided into single-criterion versus multi-criteria scheduling. The scheduling problems dealing with only one criterion to optimize are said to be models with a single criterion while if several criteria are considered the scheduling models are known as multi-criteria scheduling models. Multi-criteria scheduling problems are usually more difficult than single criterion ones.

- **Bag-of-Tasks versus Workflow Scheduling:** Based on whether dependency between tasks is considered or not, task scheduling in distributed computing can be divided into Bag-of-Tasks and Workflow scheduling. In Bag-of-Tasks scheduling, the focus is on scheduling various independent tasks which do not require any communication among themselves over resources. On the other hand, in the Workflow scheduling, the focus is on considering the dependencies between various tasks in the application while making scheduling decisions.

1.2 Motivation

Distributed computing is being carried out by exploiting various forms of computing and communication resources. For the purpose of distributed computing, an application is partitioned into task graph and represented by a DAG for showing precedence constraints, computation and communication costs of the tasks of an application. Many-many algorithms are kept coming during the last 2-3 decades. People have been interested in computing every new algorithm with the other available algorithms especially with the same type of algorithms. This motivates us to engage in research-oriented research for the following three purposes:

- Comparing the task scheduling algorithms for distributed computing on multiprocessors and attempting to obtain some systematic way of benchmarking the task scheduling algorithms.
- An exercise of comparing algorithms for standard task graphs and randomly generated task graphs is likely to provide a comparison that is considered to be acceptable to workers in the field. Further, while trying to compare and understand algorithms, there is always a likelihood for developing some better or new algorithms for the task scheduling on multiprocessors.
- With the advent of the possibility of reduction of energy consumption by processors that can control voltage and frequency dynamically. This motivates to obtain enhancement of developed algorithms to become energy aware by exploiting the capabilities of the said algorithms.

1.3 Objective

"Comparative study of performance of some existing task scheduling algorithms and identifying ways of developing some task scheduling algorithms for distributed computing on multiprocessors, that optimize the schedule length and energy consumption."

1.4 Thesis Contributions

The following contributions working on issues and challenges related to scheduling for distributed computing:

- A survey and detailed discussion of the existing scheduling algorithms within the scope of this thesis;
- Benchmarking some well-known task scheduling algorithms that belong to the group of list-based scheduling for heterogeneous computing systems and exploring the possibility of a framework for benchmarking of scheduling algorithms;
- A clustering-based task scheduling algorithm that proposes and uses the idea of edge prioritization to obtain meaningful clustering of the tasks and improves makespan of the application;
- A clustering-based task scheduling algorithm that makes use of edge zeroing concept on the critical path to reduce the communication cost among the tasks of an application and improves makespan of the application;
- An energy-aware task scheduling algorithm that aims to reduce energy consumption by exploiting dynamic voltage and frequency scaling technique.

1.5 Thesis Organization

The chapters of this thesis are structured as shown in Fig. 1.1. The remainder of the thesis is organized as follows:

- Chapter 2 presents the preliminary concepts and literature review of the task scheduling algorithms. In this chapter, existing algorithms related to list scheduling, duplication-based scheduling and clustering-based scheduling are discussed and analyzed.

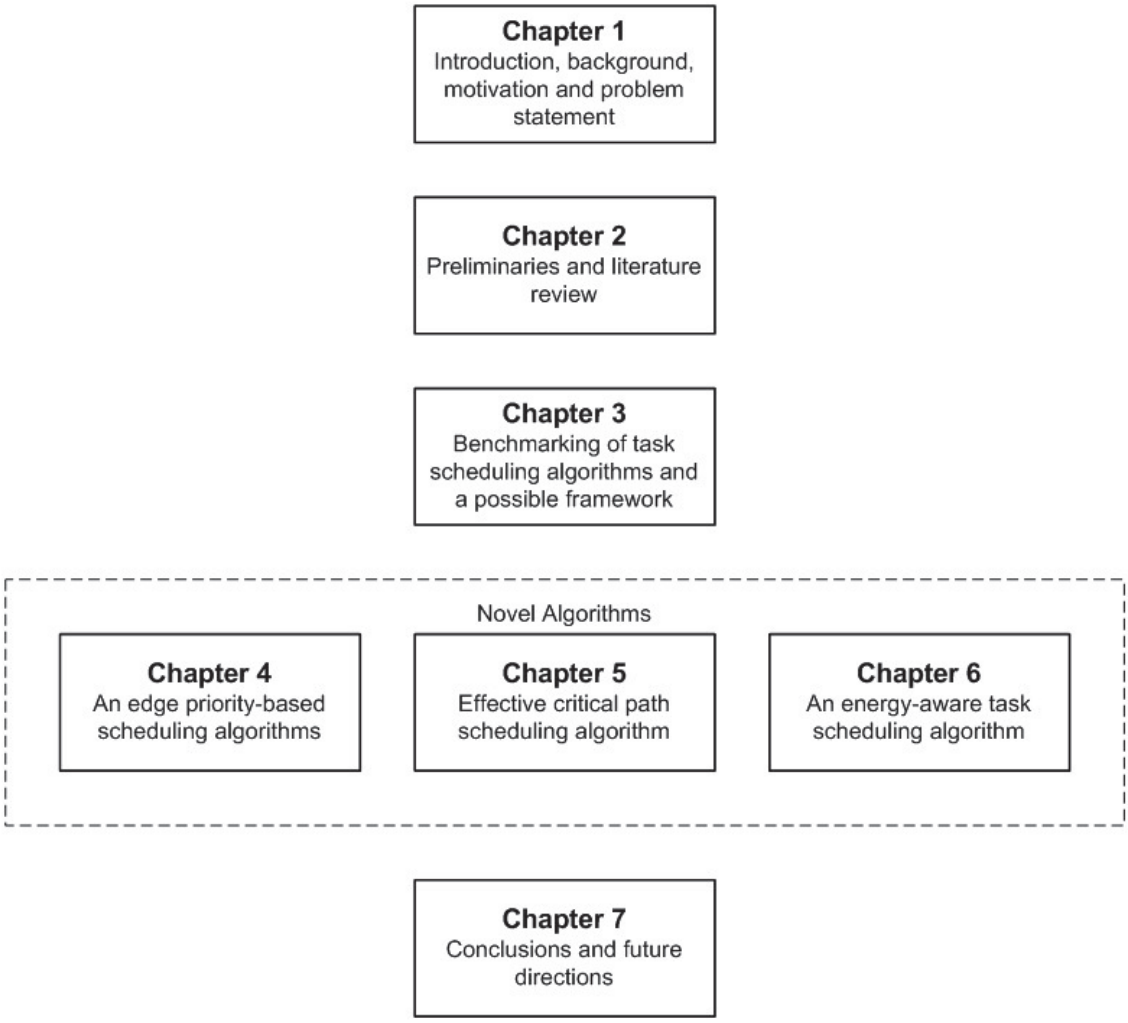

FIGURE 1.1: Thesis organization.

- Chapter 3 provides benchmarking of some well-known task scheduling algorithms that belong to the group of list-based scheduling for heterogeneous computing systems. This chapter discusses the comparison metrics and the algorithms considered for benchmarking. After that, the performance of algorithms are evaluated and compared for randomly generated graphs and the graphs generated from real-world applications such as fast Fourier transform (FFT), Gaussian Elimination, Montage and Epigenomics workflows. Further, this chapter explores the possibility of a framework for benchmarking of scheduling algorithms. This chapter is derived from:
  - **Ashish Kumar Maurya** and Anil Kumar Tripathi, "On Benchmarking

Task Scheduling Algorithms for Heterogeneous Computing Systems," *The Journal of Supercomputing*, Springer, vol. 74, no. 7, pp. 3039-3070, 2018 [**SCI**].

– **Ashish Kumar Maurya** and Anil Kumar Tripathi, "Performance Comparison of HEFT, Lookahead, CEFT and PEFT Scheduling Algorithms for Heterogeneous Computing Systems," *in Proceedings of the* 7^{th} *International Conference on Computer and Communication Technology (ICCCT-2017)*, ACM, pp. 128–132, 2017, India.

- Chapter 4 describes the proposed clustering-based task scheduling algorithm called EPS. This chapter discusses the concept of edge prioritization and clustering in context to the proposed algorithm. The time complexity of EPS is analyzed and an illustrative example demonstrating the working of the EPS algorithm is given. Finally, the experimental results, with a statistical analysis, are presented for randomly generated graphs and the graphs generated from real-world applications such as Gaussian Elimination and FFT. This chapter is derived from:

  – **Ashish Kumar Maurya** and Anil Kumar Tripathi, "An Edge Priority-based Clustering Algorithm for Multiprocessor Environments," *Concurrency and Computation: Practice and Experience*, Wiley, 2018 (in press) [**SCIE**].

- Chapter 5 describes the proposed clustering-based task scheduling algorithm called ECP. This chapter discusses the concept of critical path computation and edge selection regarding ECP. This chapter also presents the complexity analysis of algorithms presented in this chapter and provides an illustrative example demonstrating the working of the ECP. Finally, the experimental results, with a statistical analysis, are presented for randomly generated graphs and the graphs generated from real-world applications such as Gaussian Elimination, FFT and systolic array. This chapter is derived from:

  – **Ashish Kumar Maurya** and Anil Kumar Tripathi, "ECP: A Novel Clustering-based Technique to Schedule Precedence Constrained Tasks on Multiprocessor Computing Systems," *Computing*, Springer, pp. 1-25, 2018 (available as Online First article), [**SCI**].

- Chapter 6 presents an energy aware task scheduling algorithm called EAEPS that aims at reduces energy consumption by exploiting dynamic voltage and frequency scaling technique. This chapter discusses the system models used in this work and formalizes the problem. Finally, the experimental results are compared with the existing algorithms and reported. This chapter is derived from:

  - **Ashish Kumar Maurya** and Anil Kumar Tripathi, "An Energy Aware Edge Priority-based Scheduling Algorithm for Multiprocessor Environments," *in Proceedings of the* 24^{th} *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'18)*, pp. 42-46, 2018, USA.

- Chapter 7 concludes the thesis, summarizes its findings, and provides directions for future work.