Chapter 1

Introduction

1.1 Parallel Computing

1936 was a key year for computer science since the development of modern computer began in this year. A computing system can be divided into two different parts: *serial computing* and *parallel computing*. Both of them were started with the development of architecture, system software, application software, and finally they reached to the problem solving environment. To overcome performance bottlenecks in serial computation is the main reason in the development of parallel computing. Parallelism has been applied for many years, mainly in high performance computing (HPC) systems. HPC is the use of parallel processing, which aggregate the computational power to solve science, engineering and business problems. It delivers higher performance than a desktop computer or workstation to solve advanced application programs quickly, efficiently and reliably. Occasionally HPC is used as a synonym for supercomputing, although a supercomputer system performs at the currently highest operational rate for computers. Some supercomputers work at more than a *petaflop* or 10^{15} floating-point operations per second. HPC technology focuses on developing parallel processing algorithms and systems by incorporating both administration and parallel computational techniques. An HPC system is essentially a network of nodes, each of which contains one or more processing chips, as well as its own memory. The term applies especially to systems that function above a *teraflop* or 10^{12} floating-point operations per second at this time. However a desktop computer generally has a single processing chip, commonly called CPU and a workstation is basically designed for a single user and has

advanced graphics capabilities, large storage capacity, and a powerful processor. [130].

As the size of the system (Here size tells about number of cores/CPU/nodes into a single system for solving large problems. It will increase the power of the system but also increases the complexity of system since they have more nodes.) and complexity of the problem (Now a days we have very large and complex problems such as disaster management, weather forecasting, DNA sequencing, transaction processing and data warehouses. Solving these types of problems, we need HPC systems.) increased in recent years, parallel computing has become the prevalent prototype in computer architecture in the form of multi-core processors. Now, HPC systems are used for emerging applications like weather forecasting [109], DNA sequencing [117], Google search [11], molecular dynamics simulations [61], world-wide banking transactions [134], and many more. For the specific task, sometime specialized parallel computers are used.

Parallel computing is an important research area with a long development history in computer science. A parallel computer is a collection of processing elements (PEs). These PEs cooperate and communicate to solve large problems rapidly. In parallel computing, a problem decomposed in multiple parts and these parts can be solved concurrently by using multiple compute resources which runs on multiple processors. For this process, an overall control/coordination mechanism is applied. The multiple instructions of decomposed computational problem parts should be executed in less response time. The computer resources might be a single computer with many processors, or many computers connected by a network, or a combination of both. In a serial computing, a program runs on a single computer with a single Central Processing Unit (CPU) and instructions are executed one by one. At any moment of time only one instruction executes in serial computation. Parallel computing is parallel with respect to time and space [24].

There are a number of classical problems that can be decomposed into smaller tasks and solved efficiently in parallel manner such as image and signal processing, traveling salesman problem, modeling, and simulation. Figure 1.1 depicts the general process of solving a problem using parallel computing. Therefore, one can observe that parallel computing consists the following phases [118]:

- ✓ Parallel computers (hardware platforms),
- ✓ Parallel algorithm (theoretical basis),

✓ Parallel programming (software supports), and

✓ Parallel applications (large problems).

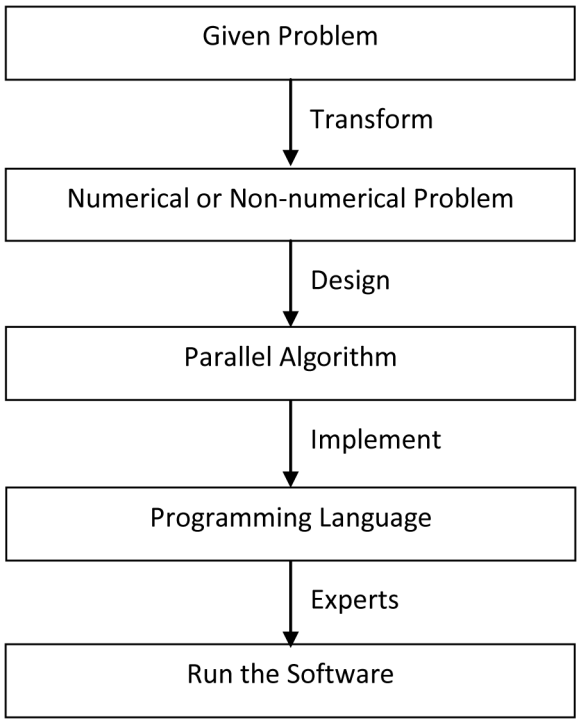

Figure 1.1: Steps of solving a problem using parallel computing.

Parallel computing enables to utilize additional processing resources to accelerate computations. The introduction of next generation multi-core processors like Intel Core i5, Core i7, AMD Phenom II 1055T and 1090T as commodity hardware has led to need for writing algorithms which can efficiently make use of all cores provided by these processors. Thus, the performance may be increased by reducing time that a process takes for completing its task. A massively parallel computer could solve scientific problems faster than a single processor. An exponential speedup is possible if sufficiently many processors are available. HPC systems comprise of thousands of components such as processing nodes, memory, disks, and other peripherals devices. In this context, the interconnection network is responsible for the fast and reliable communication among the processing nodes in a parallel computer. Indeed, the interconnection network play a vital role for linking the communication element in parallel computing environment. The demands on the network depend on the parallel computer architecture in which

the network is used.

Interconnection network is the topology of parallel computer and route messages from processors to memories (concurrent access to a shared memory structure), or from one PE (processor + memory) to another (to provide a message-passing facility). Today's hardware technology uses interconnection networks to design topology of parallel processors, because topologies can significantly affect the performance of the system [42]. These systems can be expressed on the graph to achieve goals of increasing the performance and minimizing the cost.

Today, large applications need to execute in real time and should provide the correct solutions to the critical problems in the presence of multiple failures. In this scenario, interconnection network needs to provide an uninterrupted services to the computing systems. But increase in number of components in the interconnection networks, their complexity leads to higher failure rates. Due to long execution times of larger applications, many interconnection networks show Mean Time Between Failures (MTBF) smaller than the execution time of other applications. So, it is highly important to provide the uninterrupted services and guarantee the correct results of applications even in the presence of failures.

Obviously, the performance of the current systems is nearly related to the dependability concept of fault tolerance. As stated above, increase in number of components and complexity of interconnection network increases failure rates. After analysing this situation, some questions arose such as:

✠ How does a failure affect the computing system ?

✠ How many types of failures may occur on real systems ?

✠ Can those systems tolerate the failures occurred ?

✠ Is the system able to maintain the performance after the occurrence of failures ?

✠ What will be the best option to continue the system services to achieve the fault tolerance ?

These questions address the importance of fault tolerance for the current HPC systems.

This thesis is concentrated to answering these questions with the objective of providing fault tolerance to hypercube interconnection networks in the presence of components (nodes and/or links) failures. We will now give a brief introduction of interconnection networks and network fault tolerance.

1.1.1 Interconnection Network

Interconnection networks play a major role to differentiate modern multiprocessor architectures. They can be categorized according to their topologies, routing strategies and switching techniques. Interconnection networks are built up of processing units, memories, I/O devices, and switching elements. Topology is the pattern in which the individual switches are connected to other elements, like processors, memories and other switches [36]. It connects the processing elements (PEs) of computing systems through links and switches. Switch connects fixed number of input channels to fixed number of output channels. So, the performance of a computing systems is highly dependent on interconnection networks, which is the key factor of success of future computing systems.

There are a number of criteria of selecting a right interconnection network such as bisection width, diameter, routing distance etc. In this thesis, we have considered hypercube interconnection network as a low diameter, low latency, and high speed network. It is used to send the messages form one node to another node through the bidirectional communication links.

1.1.2 Fault Tolerance in Network

Fault tolerance in network is a very important need when some characteristics of the network such as routing may provide only a single path between a source node and a destination node. In this case, if fault occurs at a node or a link along the path then the path will disconnect the communication. One way to achieve fault tolerance in networks is the use of multiple paths between the source-destination paths [70]. Network fault tolerance may be divided into two classes:

▶ Static fault tolerance: They require information in advance to perform some reconfiguration process, and

▶ Dynamic fault tolerance: They do not require information in advance about network faults.

In this thesis, we have chosen adaptive routing protocol for hypercube interconnection networks because it adapts network conditions such as communication bottlenecks. Thus, it provides the wide-range solutions to the problems of network fault tolerance

and taking advantage of multipath networks to development of massively parallel systems. Adaptive routing protocols can use alternative paths between source-destination paths [48].

1.2 Problem Statement

Nowadays, the central issue in the field of distributed system is how to provide reliable routing in the presence of components failures because it can lead to throughput degradation and losses of data. Since the numbers of processors inside supercomputers are vast and continuously growing and any type of node/link failures can stop the transmission of message, hence the data transmission is at the center of these Supercomputers. For such a situation we have to designed fault-tolerant routing algorithms for hypercube.

1.3 Motivation

Major applications of the HPC systems are used to solve advanced computation problems such as data storage and analysis, data mining, modelling, simulations, visualisation of complex data, and rapid mathematical calculations. So, HPC systems can be used to optimise production and delivery processes, analyse or develop large datasets, store large amounts of data for future analysis, create computer visualisations that explain research results, and carry out simulations and/or modelling of complex processes. HPC systems have the capacity to handle and analyse massive amounts of data at high speed. Tasks that can take months using normal computers can be done in days or even minutes by HPC systems. HPC systems can be used to model and solve highly complex problems across a range of high value sectors. As we know, communications are based on prominent and composite high-speed interconnection networks, this situation leads to increase network failure rate.

HPC systems requires a large amount of data communications among the nodes for running the major applications. The probability of the occurrence of node and/or link failures becomes high and it has severe impacts on the system. Since a single node/link failure has the terrible impacts on the system, it halts the entire computing system. In recent decades, many solutions have been proposed for tolerating faults for

computing system, but most of them are handling static faults. They are still valid but have the high cost in terms of time and resources. In the comparison of static fault tolerance, dynamic fault tolerance has better performance results. Dynamic fault tolerance improves the performance of real problems, which is beneficial to the end users and owners of HPC systems.

Hardware, software and networks can not be free from failures. In the presence of failures, a computing system can degrade the performance. So, we need to develop a robust mechanism for handling them. Fault tolerance is the ability to handle the system in the presence of faults. It is a non-functional requirement that requires the system to continue to operate in the faulty condition but with some degraded performance. Fault tolerance overhead and performance degradation are the issues that can affect the computing system's performance. The study of such factors justify the research when these issues are taken into consideration.

1.4 Objectives

The goal of this thesis is to study, implement and evaluate the fault-tolerant routing algorithms for the hypercube interconnection network. We address and design fault-tolerant routing algorithms in the presence of high number of dynamic node and/or link faults. We address this issue by designing adaptive routing protocols for hypercube interconnection networks. This technique addresses network latency and bandwidth utilization for parallel architectures. Adaptive routing algorithms exploits gains of path redundancy in n-cube. We can itemize the goals of our thesis as follows:

(i) First we study the fault tolerance theory with respect to interconnection networks to analyse the possible applications of these concepts.

(ii) Study the properties of hypercube interconnection networks and their previous results and find out the possible solutions to the problem fault tolerance in hypercube interconnection networks.

(iii) Design and implement adaptive fault-tolerant routing for hypercube networks to tolerate a high number of node and/or link failures.

(iv) Analyse the problems induced by failures and evaluate the performance of proposed solutions to the said problems.

(v) Evaluate problems again and again (if necessary) to avoid failures and suggest possible solutions.

(vi) Analyse the behaviour of solutions by means of simulation based experimental study and confirm that the proposed algorithm working properly.

1.5 Contributions

We have designed, implemented and evaluated adaptive fault-tolerant routing protocols for hypercube interconnection networks. We present the following contributions in respect to design and evaluation of the fault-tolerant routing algorithms on hypercube topology:

- Different fault-tolerant routing algorithms have been proposed for huge number of node and/or link failures in HPC systems. These algorithms execute complex applications by providing redundant multiple paths available in a hypercube topology. At the same time, this algorithm handles the congestion problems induced by failures. This work is further elaborated by giving three different fault-tolerant algorithms.

    1. The first algorithm routes the message from the source node to the destination node with the help of node-disjoint paths. It has three phases. In the first phase, a fault diagnosis and physical level monitoring are available along the source-destination path. In the second phase, if a faulty node and/or faulty link occur in the path during transmission of messages, then the algorithm reroutes the message through the alternative path. In the third phase, source node disables the faulty path after noticing the faulty node and/or faulty link in that path and establishes a new alternative path to send the same message to that destination node. This work has an incremental development and enhancement of the original approaches which has been published in [52, 56, 59, 112].
    2. The second algorithm routes the messages from one common source node to multiple distinct destination nodes with the use of subcubes of hypercubes. The proposed algorithm uses the important recursive structure property of

hypercubes. This work has followed an extension and enhancement of the original approaches which has been published in [15, 16, 73, 127].

3. The third algorithm broadcasts the message by the use of more than one independent spanning trees (ISTs). A n-dimensional hypercubes broadcast the reliable message from common root r through n-ISTs which provides n-degree fault tolerance. This work has followed an extension and enhancement of the original approaches which has been published in [115, 121, 132].

- A complete design specifications for each algorithm is given for dealing huge number of dynamic faults. All algorithms have been implemented successfully on Coloured Petri nets (CPN or CP-nets) and all algorithms are well explained with the help of examples. CPNs are a class of high level nets that extend ordinary petri nets. Using CPN tools, it is possible to investigate the behaviour of the modeled system using a simulation, to verify properties by means of state space methods and model checking. The aim of all algorithms is providing reliable message routing on parallel computers in order to avoid fault occurrences by means of node-disjoint paths.

1.6 Research Methodologies

In this thesis, we have designed, implemented and evaluated the different algorithms for fault-tolerant routing. The methodologies of all the algorithms are based on existing theories, knowledge and observations. This thesis is developed on theoretical basis and practically implemented. A survey on routing technique is given by Ni and McKinley [92] and Bertsekas et al. [14]. All methods are included with the theory of fault tolerance, which is the main objective of this thesis. First, we have conducted the literature study of hypercube networks [25, 107] and then fault-tolerance [34, 98]. With the help of some books and related research papers, we focus on the design, implementation, and evaluation of similar algorithms with efficient complexity. We have analysed the effectiveness of all the proposed fault-tolerant routing algorithms through simulation. For this, we have developed simulation model for experimental evaluation of our proposals.

1.7 Scope and Organization of the Thesis

The goal of this thesis is to further study parallel computing, interconnection networks, routing, fault tolerance and node-disjoint paths. The scope of research given in this thesis is to design, implement and evaluate fault-tolerant routing strategies for the hypercube topology. It can be utilized to frame the supercomputers. According to the objectives and research methodologies of this thesis, the scope and organization of the remaining five chapters of the thesis are as follows:

Chapter 2: This chapter explains the basic terminologies of fault tolerance, then background on interconnection networks for HPC systems including topologies and routing. Then the concepts about hypercube interconnection networks and their routing methods and previous related works that explain routing strategy for both cases– routing without failures and routing in the presence of failures.

Chapter 3: This chapter describes in detail the first adaptive fault-tolerant node-to-node routing algorithm over all shortest node-disjoint paths in n-dimensional hypercube interconnection networks. It is designed in such a way that it can handle large number of node and link failures, while delivering all n messages over disjoint-paths in the presence of maximum permissible node/link failures.

Chapter 4: This chapter presents a node-to-set node-disjoint fault-tolerant routing algorithm based on subcubes of the hypercube networks. The n-dimensional hypercube can tolerate maximum $n-1$ faulty nodes. The proposed algorithm generates node disjoint-paths which maximise the probability of setting up non-faulty path in a faulty environment.

Chapter 5: In this chapter, we present data broadcasting on parallel computers through multiple independent spanning trees (ISTs). The n-IST-based broadcasting from common root r on the hypercube network can provide n-degree fault tolerance. The designed fault-tolerant broadcasting algorithm using ISTs may increases message security in hypercube network.

Chapter 6: This chapter concludes the thesis and presents future directions in the research. This chapter also gives initiate to a broad range of open lines for fault-tolerant routing and further work.

The list of references completes the document of this thesis.