Chapter 1

Introduction

High-performance computing (HPC) is crucial in fostering economic and technological advancements. Additionally, it serves as a key metric for measuring the computational prowess of organizations. So, improving the performance and universality of HPC becomes necessary and meaningful. Among technologies and systems available in the HPC environment, multi-core systems are more flexible and best suited to increase computational power.

In contrast to alternative systems, multi-core systems are enabled with shared memory approaches for executing applications. In this paradigm, concurrently running applications on a multi-core system contend for access to shared resources via interconnected links, such as main memory and a shared cache. Due to the limited bandwidth of the main memory, the system's core count becomes constrained. Inadequate management of available shared cache capacity and memory bandwidth can result in harmful interference between different applications. This interference can lead to a significant decline in both overall system performance and the performance of individual applications.

In addition, the performance degradation an application suffers due to inter-application interference at shared resources is influenced by the other applications running simultaneously, as well as by the available memory bandwidth and shared cache capacity. Consequently, various applications encounter different and unpredictable levels of slowdown. This slowdown can be assessed using standard benchmarks and measurement tools.

To measure the performance of computation for a particular system, Standard Performance Evaluation Corporation (SPEC) provides a benchmark suite for all CPUs for the clustering environment. Such standard CPUs run with two different applications, *GCC* [1] and *mcf* [2]. These applications run on a simulated multi-core cluster where all the cores share a main memory channel. Both of the applications are responsible for running the commands internally. All the internal commands on a multi-core system use a general shared resource interface, so the performance decreases unwillingly. But, in today's era, the cluster's scaling increases daily, which increases the probability of memory congestion. This scenario requires some effective scheme to reduce memory congestion, resource allocation and load balancing to improve the performance of multi-core clusters.

The preceding introduction clarifies that an application faces different slowdowns when executing multiple concurrent applications that share some resources. Motivated by these observations, this thesis articulates several insights informed by a comprehensive review of critical scholarly works [3, 4] and attempts to improve on some of them.

1.1 Motivation and Challenges

Data mining plays a vital role in cluster computing in the new era. Most cluster computing techniques depend on scientific methods such as memory congestion reduction, load balancing, resource allocation, and power consumption & reduction techniques. Although many significant trends and technologies have already been discovered that enhance the performance of such scientific methods, the gap still needs to be improved. Therefore, it is necessary to understand such processes and strengthen their uses for multi-core cluster solutions and tools. Several researchers [5, 6, 7] have found that many commonly available desktop devices, including multi-core systems, may not use parallel processing. According to Moore's law, multi-core processors require symmetry between different cores, which is a more significant challenge. Also, the uniform data distribution in multi-core systems is complex. This topic has been working very rapidly over the last several years, a testament to the need to increase day-to-day multi-core operations and needs to be completed. According to Xiaohong et al. [8], Jack et al. [9], and Schadt et al. [10], claim that most of the work refers by using sequential programming paradigm to sync the multi-core desktops. This motivates us to work on a parallel programming paradigm to make multi-core desktops

more efficient.

In the context of multi-core systems, it becomes imperative to consider the arrangement of different cores engaged in resource sharing, load balancing, cache monitoring, etc., to enhance performance. The escalating number of cores in processors has prompted a paradigm shift in HPC, introducing challenges associated with non-uniform core connections and resulting in memory congestion as a critical bottleneck. This bottleneck is particularly exacerbated in scenarios involving the scaling of multi-core systems, especially in weak scaling scenarios. The primary challenge is to mitigate memory congestion to ensure optimal performance in such systems. This research is motivated by the urgent need to address memory congestion issues and unlock the full potential of large-scale computing environments. Additionally, the complex architecture of processor cores, arranged in a parallel configuration, poses analogous challenges related to shared memory and communication patterns. The motivation behind this research lies in optimizing the performance of multi-core systems by alleviating memory congestion issues. As these systems scale up, the impact of memory congestion on overall efficiency becomes more pronounced. Through this work, we seek to unveil innovative process placement and resource allocation strategies to minimize memory latency and enhance data movement.

Furthermore, some significant challenges are classified based on their resource distribution [11, 12]. As we know, the computational power of supercomputers brings a different perspective to research and overcomes many traditional computing limitations. Such limitations include non-linear relationship modelling, solving random problems, solving long-iterative social evolution problems and algorithms, etc. Nowadays, the scale of such simulation applications with these traditional limitations is also becoming complex, so parallel discrete event simulation, which provides high-performance solutions for such problems, is in demand. All the simulation applications vary in their simulation environment and scenarios. The role of HPC is to provide a platform to support and control the scale of such simulations. For such simulations, the execution environment for all scientific tasks is usually well-defined before execution. The dependencies include libraries, applications, job schedulers, and the operating system. Those using such services are bound to use the limited implementation of particular applications. In this context, only the running time set by management can be optimized or controlled. They can also maintain system collaboration, usually available at the system level. In this way, access to

HPC resources becomes more restricted, and all jobs have to wait for execution in a queue based on priority.

These limitations overcome a high-demanding HPC clusters (HPCC) model only when the HPCC tool applies the cluster computing concept to the existing resources. The result will be the base for the Infrastructure as a Service [13, 14, 15] for its delivery model. The cluster computing approach promises to increase flexibility and efficiency in the distribution of resources, utilization, memory management, and memory congestion reduction. Therefore, this motivates us to focus on large-scale, High-performance computing simulation based on parallel discrete event simulation on multi-core systems.

As a solution, a lot of different methods and models have been introduced by several researchers. All these works are based on the decomposition of tasks (DT) and historical tracings (HT). The DT model decomposes the assigned tasks into small individual tasks and then predicts the required resources. For prediction, it measures the requirements of each lesson. At the same time, the historical trace (HT) model predicts the required resources based on the nature and execution patterns of the jobs. However, some researchers have claimed that one cannot accurately predict the resources needed for running applications in an HPC environment [16]. These claims motivate us to work on dependency prediction of long-term resource utilization.

Finally, we can say that the advances in science and technology will demand more processing power to run the models. Hence, HPC must overcome the main challenges to get timely results regarding hardware for its deployment and the application areas with parallel programming paradigms. This motivates us to go with in-depth research to obtain my PhD. So, in this thesis, we have presented a list of challenges in detail in Chapter 2. Further, this thesis also discusses the proposed and published solutions to overcome the discussed issues.

1.2 Research Objectives

The objective of the research is to explore and enhance the performance of multi-core clusters within the realm of High-Performance Computing (HPC). This research is centered on three primary domains: viability, performance optimization, and usability.

In the first domain, viability, the research delves into the essential components that determine the effectiveness and efficiency of HPC systems. This includes the metrics for evaluating system performance, the resources such as CPU, memory, and network capabilities, and the infrastructure options, whether cloud-based or on-premise. Additionally, the software aspect is considered, focusing on both benchmark-based and user application-based performance assessments.

The second domain, performance optimization, investigates methods to improve the efficiency of HPC systems. This involves exploring elasticity, predictors, schedulers, platform selectors, and spot instance handlers to enhance resource management and system adaptability to varying workloads.

The third domain, usability, aims at making HPC systems more accessible and user-friendly. The research covers web portals, legacy to SaaS transitions, execution steering, workflow management, and application deployers, all of which contribute to streamlining the use of HPC resources and applications.

This classification framework provides a structured approach to addressing the challenges in HPC research, focusing on improving the viability, performance, and usability of multi-core clusters.

1.3 Research Methodology

The research methodology is structured into several key phases, each of which contributes to achieving the research objectives outlined above: The research methodology is structured into several key phases, each of which contributes to achieving the research objectives outlined above:

1. **Literature Review:** A comprehensive review of existing techniques relevant to improving the performance of multi-core cluster environments will be conducted. This phase will identify the strengths and weaknesses of current approaches, providing a foundation for the development of new methods that aim to enhance system efficiency and resource utilization.

2. **Methodology Development:** Based on the insights gained from the literature review, new methodologies will be designed with a focus on adaptability and efficiency. The development process will ensure that these methods can respond effectively to varying conditions and challenges within multi-core clusters.

3. **Simulation and Testing:** The proposed methodologies will be implemented and tested using a range of HPC benchmarks and real-world applications. This phase will involve simulating different scenarios to evaluate the methodologies' effectiveness in improving performance, resource management, and overall system behavior.

4. **Performance Evaluation:** The results from the simulation and testing phase will be analyzed to assess the effectiveness of the proposed methodologies. The performance of these methods will be compared with existing techniques, highlighting their strengths and identifying potential areas for further improvement.

1.4 Thesis Overview

This thesis aims to enhance the performance of multi-core clusters through three interdependent qualitative works.

1. The first work focuses on predicting multi-core CPU performance through parallel data mining on public datasets.

2. The second work addresses an effective scheme for reducing memory congestion in a multi-core environment.

3. The third work deals with the dependency prediction of long-time resource uses in an HPC environment.

The contributions of these works include the proposal of a virtual cluster designing framework encompassing simulation modelling, resource computation, and job queue prediction. A feature-based capability prediction with accuracy metrics (FBCA) is presented to validate the model's performance. The results presented in this thesis show a significant improvement in computation cost compared to traditional methods. It also introduces a parallel algorithm for multi-core synchronization on shared memory. Furthermore, it implements a parallel EM algorithm with its observations in terms of speed and errors. It also validates the proposed work with various workflow algorithms and demonstrates effectiveness through statistical analysis using some statistical tests.

Based on the landscape of multicore clusters can be divided into two primary classes: capability-based and performance-based. Each of these classes addresses distinct aspects of multicore cluster architecture, providing a comprehensive framework for understanding, optimizing, and applying these systems in various computing environments. This introduction will explore the detailed classification of multicore clusters, delving into the subcategories, associated challenges, and relevant related work to lay the foundation for a thorough analysis in this thesis.

1.4.1 Capability-Based Classification

Capability-based classification of multicore clusters focuses on the inherent features and resources available within the clusters. This classification is subdivided into three main areas: resource-based, software-based, and infrastructure-based clusters.

1.4.1.1 Resource-Based Clusters

Resource-based clusters are primarily concerned with the hardware components and their capabilities. This area is further divided into two subcategories: CPU-based and memory-based clusters.

- CPU-Based Clusters: These clusters are characterized by the number and type of central processing units (CPUs) they contain. Challenges in this category often involve optimizing CPU utilization, managing heat dissipation, and ensuring

efficient inter-CPU communication. Research has shown that heterogeneous CPU configurations can significantly impact performance, necessitating sophisticated scheduling algorithms and load balancing techniques.

- Memory-Based Clusters: Memory-based clusters focus on the capacity and performance of the memory resources. Key challenges include memory bandwidth, latency, and the management of distributed memory systems. Effective memory management strategies are critical in preventing bottlenecks and ensuring that data-intensive applications can run efficiently.

1.4.1.2 Software-Based Clusters

Software-based clusters derive their classification from benchmark-based assessments. These benchmarks evaluate the performance of clusters based on various software workloads and applications.

Benchmark-Based Classification: This approach uses standardized benchmarks to assess cluster performance across different applications. Challenges include the development of accurate and representative benchmarks that can capture the diverse workload characteristics. Additionally, maintaining consistency and comparability across different benchmarking tools is crucial for reliable evaluation.

1.4.1.3 Infrastructure-Based Clusters

Infrastructure-based clusters are defined by their physical and logical properties. These properties include network topology, storage systems, and the overall design of the cluster infrastructure.

- Properties-Based Clusters: Clusters in this category are classified based on specific properties such as network architecture (e.g., high-speed interconnects, topology design), storage solutions (e.g., distributed file systems, SSD arrays), and the overall robustness of the infrastructure. Challenges include designing scalable and resilient network architectures, optimizing data storage and retrieval mechanisms, and ensuring high availability and fault tolerance.

1.4.2 Performance-Based Classification

Performance-based classification emphasizes the operational efficiency and effectiveness of multicore clusters, primarily focusing on scheduling algorithms. These algorithms are further divided into selector-based and predictor-based scheduling.

1.4.2.1 Selector-Based Scheduling

Selector-based scheduling algorithms choose the optimal resources for tasks based on predefined criteria. These algorithms aim to maximize resource utilization and minimize execution time.

- Selector-Based Scheduling Algorithms: These algorithms select resources based on factors such as current load, resource availability, and specific task requirements. The primary challenge is developing algorithms that can make real-time decisions with minimal overhead, ensuring that the selection process does not become a bottleneck itself. Research in this area often explores heuristic and machine learning approaches to improve the efficiency of resource selection.

1.4.2.2 Predictor-Based Scheduling

Predictor-based scheduling algorithms use predictive models to anticipate future resource needs and allocate resources accordingly. These models are built using historical data and trends.

- Predictor-Based Scheduling Algorithms: These algorithms rely on predictive analytics to forecast resource demand and adjust scheduling decisions proactively. Challenges include building accurate predictive models that can adapt to changing workload patterns and integrating these models into the cluster's scheduling framework. Ensuring that the predictions remain accurate over time and do not degrade due to changing conditions is a significant research focus.

1.5 Challenges of multicore clustering research Work

Multicore cluster classification and optimization present several challenges that have been the focus of extensive research. Key challenges include:

Scalability: As cluster sizes grow, ensuring that classification methods and scheduling algorithms scale effectively is crucial. This includes managing increased communication overhead and maintaining performance consistency.

Resource Management: Efficiently managing diverse resources such as CPUs, memory, and storage while minimizing contention and maximizing throughput is a complex task. Research has explored various dynamic resource management techniques and adaptive scheduling algorithms.

Predictive Modeling: Developing accurate and adaptive predictive models for scheduler algorithms remains an ongoing challenge. Techniques such as machine learning and deep learning have been investigated to improve prediction accuracy and robustness.

Benchmarking: Establishing standardized benchmarks that can accurately reflect real-world workloads and provide meaningful comparisons across different cluster configurations is critical. This includes creating benchmarks that can adapt to evolving software and hardware trends.

1.6 Classification of Related Work

HPC Systems: Research on high-performance computing (HPC) systems has significantly contributed to our understanding of multicore clusters. Studies have focused on optimizing resource utilization, developing scalable scheduling algorithms, and improving fault tolerance.

Cluster Computing: The principles of multicore cluster classification and scheduling are highly relevant to such computing environments. Research in this domain has explored to improve the resource utilization, load balancing, and efficient scheduling.

Machine Learning Integration: Integrating machine learning techniques into cluster management has been a burgeoning area of research. Machine learning models are used for predictive scheduling, anomaly detection, and adaptive resource management.

Finally, the classification of multicore clusters into capability-based and performance-based categories provides a structured framework for analyzing and optimizing these systems. Addressing the associated challenges and building on related work will enable more efficient and effective use of multicore clusters in various computational domains. This thesis aims to delve deeper into these classifications, explore innovative solutions to the outlined challenges, and contribute to the advancement of multicore cluster technology.

1.7 Contribution and Impact

This thesis makes several fundamental contributions toward improving the performance of multi-core clusters. Our contribution advances the state-of-the-art by supporting improved throughput by analyzing CPU components, resource availability by reducing memory congestion and supporting efficiency by predicting long-term resource dependencies in multi-core clusters. . computing system. We have also included extensive state-of-the-art literature for all works. Section 1.4 itemizes these contributions. Now, we highlight these contributions and their impact, which are as follows:

- This thesis discusses the importance of multi-core computing utilization in high-performance computing scenarios.
- This emphasizes the need to carefully consider hardware factors such as processing speed, cache bandwidth, and minimum memory requirements when designing a CPU.
- The significant contribution of the thesis lies in presenting a parallel strategy to enhance multi-core performance by using a parallel data mining approach, which focuses on the Expectation Maximization (EM) algorithm.
- This thesis also evaluates the impact of parallel execution on multi-core CPUs. All the CPU components were identified through data mining approaches considering a detailed evaluation of 32 different CPU families in a virtual environment.

- The findings demonstrate a significant improvement in performance and also present ideas related to multi-core clusters, cache mapping techniques, and ranking of nodes through parallel programming paradigms.
- This thesis has presented a Reducing Congestion with Load Balancing (RCLB) algorithm to reduce memory congestion in a shared memory environment.
- The proposed method reduces memory congestion by optimizing communication locality within shared caches and memory controllers.
- Experimental results are validated on the NAS Parallel Benchmark (NPB) kernels, demonstrating the RCLB algorithm's effectiveness.
- The experimental results show a significant bandwidth improvement from 12.65% to 23% in memory-based tests.
- This thesis also addresses the challenge of unbalanced load distribution in multi-core clustering environments, which negatively impacts computational performance and requires efficient resource allocation.
- One more significant contribution is also presented in this thesis, which includes a feature-based capacity prediction algorithm (FBCA) and an accuracy and relative runtime error (ARRE) prediction algorithm.
- All the proposed algorithms and frameworks are incorporated with a respective simulation environment.
- The FBCA algorithm minimizes the redundancies in the available features, while the ARRE algorithm ensures the effectiveness of the combination technique.
- A comparative performance analysis against some general existing methods has also been carried out, achieving better accuracy from 8 % to 18 %.

These contributions are essential because they address the critical issue of multi-core clusters. They offer a consolidated approach with improved accuracy and resource utilization during computational simulations.

Finally, this thesis aims to understand the effectiveness of combining different techniques to improve the performance of multi-core clusters and reduce memory congestion with better resource allocation.

1.8 Chapter Organization

The remaining chapters are organized as follows: Chapter 2 presents detailed related work on multi-core clustering environments. This chapter also discusses the specific classification of HPC based on multi-core clusters and the issues and challenges that arise. Chapter 3 Discusses performance improvements for multi-core CPUs. Chapter 4 Explains techniques for reducing memory congestion. Chapter 5 discusses some works to improve the computational speed of multi-core CPUs by identifying and allocating long-term resources on dedicated processors.

Finally, Chapter 6 summarizes all the work and discusses future directions.