

# Preface

Computers play an indispensable role in almost all scientific fields and disciplines such as biomedicine, physical simulations, computational chemistry, and astronautics. In order to fulfill the urgent requirement for high computational capacity (for example, huge amount of computational capacity is required in big data analysis), technologies of parallel computing and resource management increasingly grow. Nowadays, multi-core processors have become mainstream in both research and real-world settings, from warehouse-scale datacenter, personal desktops, laptops, to smartphones, since they demonstrate the superior performance per watt and the larger computational capacity compared to the traditional single-core processors. For these widely used parallel systems, a key problem is how to schedule the tasks to efficiently utilize the hardware, improve the performance, and guarantee the Quality-of-Service. Because different types of architectures have totally different features, there is not universal scheduling technique that can work the best across all these architectures and different applications. Scheduling techniques often need to be altered to match the various architectures (e.g., multi-core, datacenter, and distributed system) accordingly. For example, for a big data processing system that runs on large-scale distributed system, task scheduling should focus on load balancing and data locality; for a datacenter, the scheduling should aim to guarantee the quality-of-service of the customer-facing applications (e.g., web search).

However, a book that elaborates task scheduling techniques for emerging complex parallel architectures (e.g., multi-socket architecture, heterogeneous multi-core architecture, and cloud/big data processing platform) is missing. Previous published works mainly discuss traditional task scheduling techniques for generalized parallel systems mathematically. But these techniques suffer from low performance on the emerging complex parallel architectures. To this end, this book discusses the state-of-the-art task scheduling techniques that are optimized against different architectures, and these techniques can be applied in real parallel systems directly.

In this book, we will mainly introduce these task scheduling techniques in the scenarios of emerging parallel architectures, including the multi-core architecture,

cloud platform, and accelerator. The book will examine the current challenges in this topic, and present detailed solutions including algorithms, methods, and perspectives. It is well noticed that parallel architectures are becoming more and more complex. Instances are multi-socket multi-core architecture, asymmetric multi-core architecture, various parallel accelerators, distributed parallel architecture, etc. For different types of parallel architectures, in order to utilize the hardware efficiently and maximize the performance, different techniques have been introduced and integrated into the traditional task scheduling policy. To this end, we elaborate these techniques and demonstrate that they can be implemented efficiently for emerging parallel architectures. This book consists of three main parts: *background*, *task scheduling techniques*, and *perspectives*.

## Part I: Background

In this part, we mainly introduce the background of this book, including the emerging widely used parallel architectures and classic task scheduling techniques. This part includes two chapters.

In the first chapter, we do a survey on the emerging parallel architectures including multi-core architectures, NUMA-enabled multi-core architectures, asymmetric multi-core architectures, accelerators, cloud platform, and so on. Besides these parallel architectures, vendors are now producing other architectures. For instance, Google releases Tensor Processing Unit (TPU), which is a parallel architecture as well recently.

In the second chapter, we introduce the classic task scheduling policies and parallel programming environments. Work-sharing and work-stealing are the two most classic task scheduling policies. In addition, we introduce many parallel programming environments, such as Apache Hadoop, Spark, MIT Cilk, TBB, X10, and so on. These task scheduling policies can be applied in various parallel architectures but may suffer from low performance. In the next part, we introduce the techniques that optimize the parallel applications on various parallel architectures.

## Part II: Task Scheduling for Various Parallel Architectures

In this part, we introduce techniques that can be used to improve the performance of applications on the emerging widely used parallel architectures.

In the third chapter, targeting the multi-socket architecture, we will introduce cache-aware task scheduling policy, which can improve shared cache utilization in different sockets.

In the fourth chapter, on the NUMA (Non-Uniform Memory Access)-enabled architecture, the NUMA-aware task scheduling policy will be introduced, which

can reduce remote memory accesses and improve the performance of applications in consequence.

In the fifth chapter, we introduce workload-aware task scheduling policy, which is proposed for asymmetric multi-core architecture. On this kind of architecture, where different cores operate at different speeds, partitioning by the workload can truly improve the performance.

In the sixth chapter, we introduce asymptotic technique to allocate workload between CPU and GPU for CPU+GPU heterogeneous parallel architecture. On this kind of architecture, different applications have different speedup ratios on the GPU compared with CPU, because the applications have various characteristics. It is not trivial to find an optimal workload partition between CPU and GPU.

Nowadays, big data analysis requires tremendous amount of computers to process the data in parallel. In the seventh chapter, we will introduce several featured dynamic task scheduling policies that can significantly improve the performance of big data processing on heterogeneous cloud platforms.

The eighth chapter contains the quality-of-service aware task scheduling policy for accelerators. Using this policy, it can improve the accelerator utilization. Moreover, it also guarantees the quality-of-service of latency-critical applications.

### **Part III: Summary and Perspectives**

In this part, we summarize all the previously introduced task scheduling solutions for parallel architectures, provide our perspectives, and discuss the possibilities of designing new dynamic task scheduling policies for more other future parallel architectures. Especially, we give several guidelines of designing new efficient and effective task scheduling techniques for those newly released parallel architectures.

After reading this book, we expect the readers will have an overview on the recent progress of task scheduling policies in parallel architectures. And we also hope the book can help the readers to quickly master the focused issues and opening problems if they tend to work in this field. In order to understand this book, the readers are suggested to have some basic knowledge on computer architecture, multi-core, and parallel processing.

Shanghai, China  
September 2017

Quan Chen  
Minyi Guo

<http://www.springer.com/978-981-10-6237-7>

Task Scheduling for Multi-core and Parallel  
Architectures

Challenges, Solutions and Perspectives

Chen, Q.; Guo, M.

2017, XVIII, 243 p. 107 illus., 73 illus. in color.,

Hardcover

ISBN: 978-981-10-6237-7