Parallel Algorithms

Professor
Dr. Masoumeh Damroudi

Author
Mohammad Khorshidi

Course Overview

In this lecture note, I have documented my personal interpretations and understanding of the material. Please note that there may be inaccuracies, and it is recommended to consult primary and authoritative sources for complete and accurate information. Additionally, I welcome any contributions to improve and refine this note. If you are reading this and would like to offer suggestions, feel free to do so via pull requests on my **GitHub** or by emailing me at **mohammad_khorshidi@outlook.fr**.

0.1 Course Structure

0.2 Instructor's Approach

0.3 Grading and Evaluation

The grading structure for this course is as follows, with students expected to earn their grades based on the criteria outlined below:

- **Presentation:** 4 points will be awarded for presenting and explaining a paper related to one aspect of parallelization, accompanied by its relevant implementation.
- **Midterm Exam:** 4 points will be given for the midterm exam.
- **Final Exam:** 12 points will be awarded for the final exam.

Additionally, students are expected to attend all classes and give due attention to the assignments that will be provided throughout the semester.

0.4 Course Materials and Resources

Contents

Chapter 1

Introduction

In today's world, with the increasing volume of data and the need for faster computations, parallel algorithms have emerged as an effective method to enhance the performance of computing systems. Parallel algorithms allow us to divide a problem into smaller sub-problems, each of which can be processed simultaneously by multiple processors. This not only speeds up processing but also optimizes the use of hardware resources.

In this lecture note, we will explore fundamental concepts of parallel processing, computational models such as the PRAM model, and interconnection structures like network links. We will also delve into the principles of designing and analyzing parallel algorithms, covering key metrics such as performance, speedup, and scalability. Our goal is to provide a comprehensive understanding of the challenges and opportunities in the field of parallel algorithms.

Parallel algorithms play a crucial role in improving the performance of complex and resource-intensive computations. The primary goal of parallel algorithms is to divide a large problem into smaller, independent sub-problems, which can be solved simultaneously by multiple processors. This technique is essential in multi-core and distributed computing environments such as supercomputers and large computing clusters. Below are some key concepts and approaches that are foundational in the field of parallel algorithms:

1. **Parallel Computational Models**

   To design and analyze parallel algorithms, various computational models are employed, each defining how processors interact and how instructions are executed in parallel.

   - **PRAM Model (Parallel Random Access Machine):** One of the most widely used models for parallel computing. In PRAM, all processors have simultaneous access to a shared memory. This model is further divided into several subclasses based on the rules of access to shared memory:
     - **EREW (Exclusive Read, Exclusive Write):** Only one processor can read from or write to a memory cell at a time.
     - **CREW (Concurrent Read, Exclusive Write):** Multiple processors can read from the same memory location, but only one can write at a time.
     - **ERCW (Exclusive Read, Concurrent Write):** Multiple processors can write to the same memory location, but reading is exclusive.
     - **CRCW (Concurrent Read, Concurrent Write):** Multiple processors can both read from and write to the same memory location simultaneously.
   - **Interconnection Networks:** For systems where processors have independent memory and communicate via a network, interconnection structures like Mesh, Hypercube, and Ring are used. These networks define how data is transmitted between processors in parallel systems.

2. **Parallel Algorithm Design and Optimization**

   The design of parallel algorithms involves breaking down a problem into sub-problems that can be executed independently. Some key concepts related to the efficiency and performance of parallel algorithms include:

- **Speedup:** This is a measure of how much faster a parallel algorithm runs compared to its sequential counterpart. Ideally, the speedup increases proportionally with the number of processors used. However, due to factors such as communication overhead and non-parallelizable sections of code, perfect speedup is often unattainable.
- **Efficiency:** Efficiency is a measure of how well a parallel system utilizes its resources. It is the ratio of speedup to the number of processors. A parallel algorithm is considered efficient if it achieves high speedup with a minimal number of processors.
- **Scalability:** Scalability refers to the ability of a parallel algorithm to maintain its performance improvement as more processors are added. A scalable algorithm should continue to perform efficiently as the problem size and number of processors increase.

3. **Challenges in Parallelism**

   One of the main challenges in parallel computing is managing dependencies between tasks, which can limit the degree of parallelism:

   - **Data Dependency:** This occurs when one task relies on the output of another. There are three types:
     - **Flow Dependency (RAW - Read After Write):** One task needs to read a variable after another task has written to it.
     - **Anti Dependency (WAR - Write After Read):** A task must write to a variable only after another task has read it.
     - **Output Dependency (WAW - Write After Write):** Two tasks try to write to the same variable simultaneously.
   - **Control Dependency:** This occurs when the execution of a task depends on a control flow decision, such as an if-then-else structure or a loop.
   - **Resource Dependency:** Occurs when multiple tasks require access to the same resource, such as memory or a computational unit, simultaneously.

4. **Applications of Parallel Algorithms**

   Parallel algorithms have broad applications across numerous fields, including:

   - **Scientific and Engineering Computations:** These include solving complex mathematical equations, simulations of physical processes, and large-scale statistical computations.
   - **Big Data Processing:** Parallelism is essential for handling vast amounts of data in distributed systems, where tasks such as sorting, filtering, and data analysis are executed concurrently.
   - **Image and Signal Processing:** Parallel algorithms are often employed in matrix operations, digital filters, and image manipulation tasks.
   - **Artificial Intelligence and Deep Learning:** Many AI algorithms, particularly in neural networks and machine learning, leverage parallelism to process large datasets and improve training efficiency.