Introduction to Parallel Computing

Chieh-Sen (Jason) Huang

Department of Applied Mathematics

National Sun Yat-sen University

Thank Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar for providing slides.

Topic Overview

- Motivating Parallelism
- Scope of Parallel Computing Applications
- Organization and Contents of the Course

Motivating Parallelism

- The role of parallelism in accelerating computing speeds has been recognized for several decades.

- Its role in providing multiplicity of datapaths and increased access to storage elements has been significant in commercial applications.

- The scalable performance and lower cost of parallel platforms is reflected in the wide variety of applications.

- Processor (CPU) is the active part of the computer, which does all the work of data manipulation and decision making.
- Datapath is the hardware that performs all the required data processing operations, for example, ALU, registers, and internal buses.
- Control is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc.

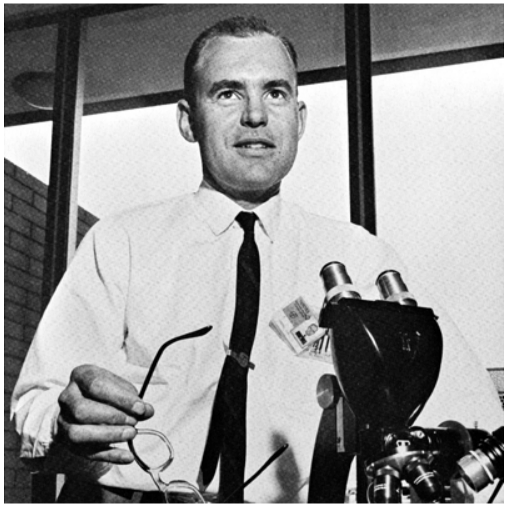
Motivating Parallelism

- Developing parallel hardware and software has traditionally been time and effort intensive.
- If one is to view this in the context of rapidly improving uniprocessor speeds, one is tempted to question the need for parallel computing.
- There are some unmistakable trends in hardware design, which indicate that uniprocessor (or implicitly parallel) architectures may not be able to sustain the rate of *realizable* performance increments in the future.
- This is the result of a number of fundamental physical and computational limitations.
- The emergence of standardized parallel programming environments, libraries, and hardware have significantly reduced time to (parallel) solution.

The Computational Power Argument

Moore's law states (1965):

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000."



Gordon Moore at Fairchild R & D in 1962.

The Computational Power Argument

Moore attributed this doubling rate to exponential behavior of die sizes, finer minimum dimensions, and “circuit and device cleverness”.

In 1975, he revised this law as follows:

“There is no room left to squeeze anything out by being clever. Going forward from here we have to depend on the two size factors - bigger dies and finer dimensions.”

He revised his rate of *circuit complexity* doubling to 18 months and projected from 1975 onwards at this reduced rate.

A die in the context of integrated circuits is a small block of semiconducting material, on which a given functional circuit is fabricated.

By 2004, clock frequencies had gotten fast enough-around 3 GHz that any further increases would have caused the chips to melt from the heat they generated. So while the manufacturers continued to increase the number of transistors per chip, they no longer increased the clock frequencies. Instead, they started putting multiple processor cores on the chip.

The Computational Power Argument

- If one is to buy into Moore's law, the question still remains – how does one translate transistors into useful OPS (operations per second)?
- The logical recourse is to rely on parallelism, both implicit and explicit.
- Most serial (or seemingly serial) processors rely extensively on implicit parallelism.
- We focus in this class, for the most part, on explicit parallelism.

Implicit parallelism is a characteristic of a programming language that allows a compiler or interpreter to automatically exploit the parallelism inherent to the computations expressed by some of the language's constructs. A pure implicitly parallel language does not need special directives, operators or functions to enable parallel execution.

The Memory/Disk Speed Argument

- While clock rates of high-end processors have increased at roughly 40% per year over the past decade, DRAM access times have only improved at the rate of roughly 10% per year over this interval.
- This mismatch in speeds causes significant performance bottlenecks.
- Parallel platforms provide increased bandwidth to the memory system.
- Parallel platforms also provide higher aggregate caches.
- Principles of locality of data reference and bulk access, which guide parallel algorithm design also apply to memory optimization.
- Some of the fastest growing applications of parallel computing utilize not their raw computational speed, rather their ability to pump data to memory and disk faster.

The Data Communication Argument

- As the network evolves, the vision of the Internet as one large computing platform has emerged.
- This view is exploited by applications such as SETI@home and Folding@home.
- In many other applications (typically databases and data mining) the volume of data is such that they cannot be moved.
- Any analyses on this data must be performed over the network using parallel techniques.

The search for extraterrestrial intelligence (SETI) is the collective name for a number of activities undertaken to search for intelligent extraterrestrial life

Folding@home is a distributed computing project for disease research that simulates protein folding, computational drug design, and other types of molecular dynamics. The project uses the idle processing resources of thousands of personal computers owned by volunteers who have installed the software on their systems.

Scope of Parallel Computing Applications

- Parallelism finds applications in very diverse application domains for different motivating reasons.
- These range from improved application performance to cost considerations.

Applications in Engineering and Design

- Design of airfoils (optimizing lift, drag, stability), internal combustion engines (optimizing charge distribution, burn), high-speed circuits (layouts for delays and capacitive and inductive effects), and structures (optimizing structural integrity, design parameters, cost, etc.).
- Design and simulation of micro- and nano-scale systems.
- Process optimization, operations research.

Scientific Applications

- Functional and structural characterization of genes and proteins.
- Advances in computational physics and chemistry have explored new materials, understanding of chemical pathways, and more efficient processes.
- Applications in astrophysics have explored the evolution of galaxies, thermonuclear processes, and the analysis of extremely large datasets from telescopes.
- Weather modeling, mineral prospecting, flood prediction, etc., are other important applications.
- Bioinformatics and astrophysics also present some of the most challenging problems with respect to analyzing extremely large datasets.

Commercial Applications

- Some of the largest parallel computers power the wall street!
- Data mining and analysis for optimizing business and marketing decisions.
- Large scale servers (mail and web servers) are often implemented using parallel platforms.
- Applications such as information retrieval and search are typically powered by large clusters.

Applications in Computer Systems

- Network intrusion detection, cryptography, multiparty computations are some of the core users of parallel computing techniques.
- Embedded systems increasingly rely on distributed control algorithms.
- A modern automobile consists of tens of processors communicating to perform complex tasks for optimizing handling and performance.
- Conventional structured peer-to-peer networks impose overlay networks and utilize algorithms directly from parallel computing.

Applications in Data Science

- Data is too big to be processed and analyzed in one single machine.
- Parallel computing in distributed file systems: Googles distributed file systems and programming model, Google File System (GFS, 2003) and the MapReduce, have addressed the problems of distributed computations and processing failure recovery.
- In memory parallel computing: New large-scale data processing frameworks, such as Apache Spark & H20, and their new DataFrame & Datasets APIs have been proposed to deal with such distributed big data analytics tasks.
- Database parallel computing: Greenplum Database is an advanced open source data platform. It provides powerful and rapid analytics on PETABYTE SCALE data volumes.

Organization and Contents of this Course

- Fundamentals: This part of the class covers basic parallel platforms, principles of algorithm design, group communication primitives, and analytical modeling techniques.
- Parallel Programming using MPI: This part of the class deals with programming using message passing libraries.
- Parallel Data Scicence Computing: This part of the class deals with parallel computing tools in data science, including RHadoop, SparkR. RPostgreSQL,
- Thread Computing: This part of the class deals with thread computing using Pthread, OpenMP, and Cuda.
- Some Applications to Scientific Computing: This part of class deals with programming with MPI to some scientific computing problems, such as, solving solutions for PDEs, imaging processing, finding solutions for linear systems and etc.