

CS 292

Introduction to Parallel Computing

Spring 2008

VANDERBILT



School of Engineering

Announcements

- Read Chapter 1 in class text
- Obtain an account on ACCRE cluster
- Sign up for ACCRE training classes (as needed)

What is Parallelism?

- A strategy for performing large, complex tasks faster.
- A large task can either be performed serially, one step following another, or can be decomposed into smaller tasks to be performed simultaneously, i.e., in parallel.
- Parallelism is done by:
 - Breaking up the task into smaller tasks
 - Assigning the smaller tasks to multiple workers to work on simultaneously
 - Coordinating the workers

What is Parallelism?

- Consider your favorite computational problem...
 - One processor can give you a result in N hours.
 - Why not use N processors and get the result in just one hour?
- The concept is simple:
Parallelism = applying multiple processors to a single problem
- Parallel problem solving is common. Examples: building construction; operating a large organization; automobile manufacturing plant

Sequential Programming

Traditionally, programs have been written for serial computers:

- One instruction executed at a time
- Using one processor
- Processing speed dependent on how fast data can move through hardware; besides CPU speed this includes
 - Disk to memory speed
 - Memory to CPU speed
- Fastest machines execute approximately 1 instruction in 9-12 billionths of a second

The Need for Speed!!

- You might think that one instruction executed in 9 billionths of a second would be fast enough. You'd be wrong.
- There are several classes of problems that require faster processing...
- Simulation and Modeling problems:
 - Based on successive approximations: more calculations means more precision
- Problems dependent on computations / manipulations of large amounts of data
 - Image and Signal Processing
 - Entertainment (Image Rendering)
 - Seismic
- Grand Challenge Problems:

Climate Modeling	Semiconductor Modeling
Fluid Turbulence	Superconductor Modeling
Human Genome	Vision & Cognition
Quantum Chromodynamics	Ocean Circulation

It's more than just speed

- Parallel computing allows us to...
 - Solve large problems that won't fit on a single CPU
 - Run more cases than otherwise possible
 - Run simulations at finer resolutions
 - Model physical phenomena more realistically
- And yes, it is about speed too. We want our answers in a reasonable time (that's a polite way of saying ASAP!)

Weather Forecasting

- Atmosphere is modeled by dividing it into 3-dimensional regions or cells, 1 mile x 1 mile x 1 mile (10 cells high)
 - About 500×10^6 cells
- The calculations of each cell are repeated many times to model the passage of time
- About 200 floating point operations per cell per time step or 10^{11} floating point operations necessary per time step
- A 10 day forecast with 10 minute resolution → 1.5×10^{14} flop
 - At 100Mflops it would take about 17 days
 - At 10Gflops it would take 4 hours
 - At 1.7Tflops it would take 2 minutes

(This assumes peak CPU performance and ignores memory issues)

It's more than CPU Power

- Overall system performance (system throughput) is not just determined by CPU speed.
- Performance is highly dependent upon the ability of the memory system to supply data to the CPU.
- Parallel platforms provide:
 1. Larger aggregate memory size
 2. Higher aggregate memory bandwidth
 3. Larger aggregate cache size

Scope of Parallel Computing

- Use of parallel computing is standard practice in many fields and is making inroads into new fields.
- Engineering:
 - High-end supercomputers are depended upon to design a wide range of devices and machines (airfoils, engines, circuitry, etc.)
 - Also used to solve challenging process optimization problems.
 - View “[Will Pringles Fly?](#)“ at Rice University
- Sciences:
 - Use of parallel computers is allowing us to tackle problems never before imagined
 - See 2001 NY Times article “[All Science Is Computer Science](#)”
 - Note how many fields now include “computational” as a part of their title (e.g., Computational Chemistry)

Scope of Parallel Computing

- Entertainment:
 - Image rendering (from *The Last Starfighter* to *Shrek*)
 - *The Last Starfighter* was one of the first movies to have large segments that were solely computer generated: used a Cray XM-P
 - Gaming consoles
- Business:
 - “Follow the money”
 - Many big financial firms rely upon the most powerful “private” supercomputers to make decisions ahead of their competitors
- Data Mining:
 - Databases are becoming too large; requires parallel processing to return results in a timely manner

Announcements

- Read Sections 2.1-2.3

Topic Overview

- Implicit Parallelism: Trends in Microprocessor Architectures
- Limitations of Memory System Performance
- Dichotomy of Parallel Computing Platforms
- Communication Model of Parallel Platforms
- Physical Organization of Parallel Platforms
- Communication Costs in Parallel Machines
- Messaging Cost Models and Routing Mechanisms
- Mapping Techniques
- Case Studies

Scope of Parallelism

- Conventional architectures coarsely comprise of a processor, memory system, and the datapath.
- Each of these components present significant performance bottlenecks.
- Parallelism addresses each of these components in significant ways.
- Different applications utilize different aspects of parallelism - e.g., data intensive applications utilize high aggregate throughput, server applications utilize high aggregate network bandwidth, and scientific applications typically utilize high processing and memory system performance.
- It is important to understand each of these performance bottlenecks.

Implicit Parallelism: Trends in Microprocessor Architectures

- Microprocessor clock speeds have posted impressive gains over the past two decades (two to three orders of magnitude).
- Higher levels of device integration have made available a large number of transistors.
- The question of how best to utilize these resources is an important one.
- Current processors use these resources in multiple functional units and execute multiple instructions in the same cycle.
- The precise manner in which these instructions are selected and executed provides impressive diversity in architectures.
- We will only introduce some of these issues in lecture – the text has a more complete discussion

Pipelining and Superscalar Execution

- Pipelining overlaps various stages of instruction execution to achieve performance.
- At a high level of abstraction, an instruction can be executed while the next one is being decoded and the subsequent one is being fetched.
- This is akin to an assembly line for manufacture of cars.

Pipelining and Superscalar Execution

- Pipelining, however, has several limitations.
- The speed of a pipeline is eventually limited by the slowest stage.
- For this reason, conventional processors rely on very deep pipelines (20 stage pipelines in state-of-the-art Pentium processors).
- However, in typical program traces, every 5-6th instruction is a conditional jump! This requires very accurate branch prediction.
- The penalty of a misprediction grows with the depth of the pipeline, since a larger number of instructions will have to be flushed.

Superscalar Execution

- Scheduling of instructions is determined by a number of factors:
 - True Data Dependency: The result of one operation is an input to the next.
 - Resource Dependency: Two operations require the same resource.
 - Branch Dependency: Scheduling instructions across conditional branch statements cannot be done deterministically a-priori.
 - The scheduler, a piece of hardware looks at a large number of instructions in an instruction queue and selects appropriate number of instructions to execute concurrently based on these factors.
 - The complexity of this hardware is an important constraint on superscalar processors.

Superscalar Execution: Issue Mechanisms

- In the simpler model, instructions can be issued only in the order in which they are encountered. That is, if the second instruction cannot be issued because it has a data dependency with the first, only one instruction is issued in the cycle. This is called *in-order* issue.
- In a more aggressive model, instructions can be issued out of order. In this case, if the second instruction has data dependencies with the first, but the third instruction does not, the first and third instructions can be co-scheduled. This is also called dynamic issue.
- Performance of in-order issue is generally limited.

Superscalar Execution: Efficiency Considerations

- Not all functional units can be kept busy at all times.
- If during a cycle, no functional units are utilized, this is referred to as vertical waste.
- If during a cycle, only some of the functional units are utilized, this is referred to as horizontal waste.
- Due to limited parallelism in typical instruction traces, dependencies, or the inability of the scheduler to extract parallelism, the performance of superscalar processors is eventually limited.
- Conventional microprocessors typically support four-way superscalar execution.