Computer Organization

Douglas Comer

Computer Science Department
Purdue University
250 N. University Street
West Lafayette, IN 47907-2066

http://www.cs.purdue.edu/people/comer

© Copyright 2006. All rights reserved. This document may not be reproduced by any means without written consent of the author.
XVII

Parallelism
Two Fundamental Hardware Techniques Used To Increase Performance

- Parallelism
- Pipelining
Parallelism

- Multiple copies of hardware unit used
- All copies can operate simultaneously
- Occurs at many levels of architecture
- Term *parallel computer* applied when parallelism dominates entire architecture
Characterizations Of Parallelism

- Microscopic vs. macroscopic
- Symmetric vs. asymmetric
- Fine-grain vs. coarse-grain
- Explicit vs. implicit
Parallelism is so fundamental that virtually all computer systems contain some form of parallel hardware. We use the term microscopic parallelism to characterize parallel facilities that are present, but not especially visible.
Examples Of Microscopic Parallelism

- Parallel operations in an ALU
- Parallel access to general-purpose registers
- Parallel data transfer to/from physical memory
- Parallel transfer across an I/O bus
Examples Of Macroscopic Parallelism

• Symmetric parallelism
  – Refers to multiple, identical processors
  – Example: dual processor PC

• Asymmetric parallelism
  – Refers to multiple, dissimilar processors
  – Example: PC with a graphics processor
Level Of Parallelism

- Fine-grain
  - Parallelism among individual instructions or data elements
- Coarse-grain parallelism
  - Parallelism among programs or large blocks of data
Explicit And Implicit Parallelism

- **Explicit**
  - Visible to programmer
  - Requires programmer to initiate and control parallel activities

- **Implicit**
  - Invisible to programmer
  - Hardware runs multiple copies of program automatically
Parallel Architectures

- Design in which computer has reasonably large number of processors
- Intended for \textit{scaling}
- Example: computer with thirty-two processors
- Not generally classified as parallel computer
  - Dual processor computer
  - Quad processor computer
## Types Of Parallel Architectures

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISD</td>
<td>Single Instruction Single Data stream</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data streams</td>
</tr>
<tr>
<td>MIMD</td>
<td>Multiple Instructions Multiple Data streams</td>
</tr>
</tbody>
</table>

- Known as *Flynn classification*
Conventional (Nonparallel) Architecture

- Known as *Single Instruction Single Data*
- Other terms include
  - *Sequential architecture*
  - *Uniprocessor*
Single Instruction Multiple Data (SIMD)

- Each instruction specifies a single operation
- Hardware applies operation to multiple data items
Vector Processor

- Uses SIMD architecture
- Applies a single floating point operation to an entire array of values
- Example use: normalize values in a set
Normalization On A Conventional Computer

for i from 1 to N {
    V[i] ← V[i] × Q;
}

{Normalization On A Conventional Computer}
Normalization On A Vector Processor

\[ V \leftarrow V \times Q; \]

- Trivial amount of code
- Special instruction called \textit{vector instruction}
- If vector \( V \) larger than hardware capacity, multiple steps are required
Graphics Processors

- Graphics hardware uses sequential bytes in memory to store pixels
- To move a window, software copies bytes
- SIMD architecture allows copies in parallel
Multiple Instructions Multiple Data (MIMD)

- Parallel architecture with separate processors
- Each processor runs independent program
- Processors visible to programmer
Two Popular Categories Of Multiprocessors

- Symmetric
- Asymmetric
Symmetric Multiprocessor (SMP)

- Most well-known MIMD architecture
- Set of $N$ identical processors
- Examples of groups that built SMP computers
  - Carnegie Mellon University (C.mmp)
  - Sequent Corporation (Balance 8000 and 21000)
  - Encore Corporation (Multimax)
Illustration Of A Symmetric Multiprocessor
Asymmetric Multiprocessor (AMP)

- Set of $N$ processors
- Multiple types of processors
- Processors optimized for specific tasks
- Often use master-slave paradigm
Example AMP Architectures

- Math (or graphics) coprocessor
  - Special-purpose processor
  - Handles floating point (or graphics) operations
  - Called by main processor as needed

- I/O Processor
  - Optimized for handling interrupts
  - Programmable
Examples Of Programmable I/O Processors

- Channel (IBM mainframe)
- Peripheral Processor (CDC mainframe)
Multiprocessor Overhead

- Having many processors is not always a clear win
- Overhead arises from
  - Communication
  - Coordination
  - Contention
Communication

• Needed
  – Among processors
  – Between processors and I/O devices

• Can become a bottleneck
Coordination

- Needed when processors work together
- May require one processor to coordinate others
Contestation

- Processors contend for resources
  - Memory
  - I/O devices
- Speed of resources can limit overall performance
  - Example: $N-1$ processors wait while one processor accesses memory
Performance Of Multiprocessors

- Disappointing
- Bottlenecks
  - Contention for operating system (only one copy of OS can run)
  - Contention for memory and I/O
- Another problem: either need
  - One centralized cache (contention problems)
  - Coordinated caches (complex interaction)
- Many applications are I/O bound
According To John Harper

“Building multiprocessor systems that scale while correctly synchronising the use of shared resources is very tricky, whence the principle: with careful design and attention to detail, an N-processor system can be made to perform nearly as well as a single-processor system. (Not nearly N times better, nearly as good in total performance as you were getting from a single processor). You have to be very good — and have the right problem with the right decomposability — to do better than this.’’

http://www.john-a-harper.com/principles.htm
Definition Of Speedup

- Defined relative to single processor

\[ \text{Speedup} = \frac{\tau_1}{\tau_N} \]

- \( \tau_1 \) denotes the execution time on a single processor
- \( \tau_N \) denotes the execution time on a multiprocessor
- Goal: speedup is linear in number of processors
Ideal And Typical Speedup

Number of processors (N) | Speedup
---|---
1 | ideal
4 | actual
8 | 12 | 16 | 32 | 2006
Speedup For \( N >> 1 \) Processors

![Graph showing speedup for 1 to 32 processors. The ideal line and actual line are depicted.]
Summary Of Speedup

When used for general-purpose computing, a multiprocessor may not perform well. In some cases, added overhead means performance decreases as more processors are added.
Consequences For Programmers

• Writing code for multiprocessors is difficult
  – Need to handle mutual exclusion for shared items
  – Typical mechanism: locks
The Need For Locking

- Consider an assignment

\[ x = x + 1; \]

- Typical code is

```
load  x, R5
incr  R5
store R5, x
```
Example Of Problem With Parallel Access

• Consider two processors incrementing item $x$
  – Processor 1 loads $x$ into its register 5
  – Processor 1 increments its register 5
  – Processor 2 loads $x$ into its register 5
  – Processor 1 stores its register 5 into $x$
  – Processor 2 increments its register 5
  – Processor 2 stores its register 5 into $x$
Hardware Locks

- Prevent simultaneous access
- Separate lock assigned to each item
- Code is

  lock 17
  load x, R5
  incr R5
  store R5, x
  release 17
Programming Parallel Computers

- Implicit parallelism
  - Programmer writes sequential code
  - Hardware runs many copies automatically

- Explicit parallelism
  - Programmer writes code for parallel architecture
  - Code must use locks to prevent interference
The Point About Parallel Programming

From a programmer’s point of view, a system that uses explicit parallelism is significantly more complex to program than a system that uses implicit parallelism.
Programming Symmetric And Asymmetric Multiprocessors

- Both types can be difficult to program
- Symmetric has two advantages
  - One instruction set
  - Programmer does not need to choose processor type for each task
Redundant Parallel Architectures

- Used to increase reliability
- Do not improve performance
- Multiple copies of hardware perform same function
- Can be used to
  - Test whether hardware is performing correctly
  - Serve as backup in case of hardware failure
Loose And Tight Coupling

- **Tightly coupled multiprocessor**
  - Multiple processors in single computer
  - Buses or switching fabrics used to interconnect processors, memory, and I/O
  - Usually one operating system

- **Loosely coupled multiprocessor**
  - Multiple, independent computer systems
  - Computer networks used to interconnect systems
  - Each computer runs its own operating system
  - Known as *distributed computing*
Cluster Computer

- Distributed computer system
- All computers work on a single problem
- Works best if problem can be partitioned into pieces
Grid Computing

- Form of loosely-coupled distributed computing
- Uses computers on the Internet
- Popular for large, scientific computations
Summary

- Parallelism is a fundamental optimization
- Computers classified as
  - SISD (e.g., conventional uniprocessor)
  - SIMD (e.g., vector computer)
  - MIMD (e.g., multiprocessor)
- Multiprocessor speedup usually less than linear
Summary (continued)

- Multiprocessors can be
  - Symmetric or asymmetric
  - Explicitly or implicitly parallel

- Programming multiprocessors is usually difficult
  - Locks needed for shared items

- Parallel systems can be
  - Tightly-coupled (single computer)
  - Loosely-coupled (computers connected by a network)
Questions?