CS4961 Parallel Programming

Lecture 4: Memory Systems and Interconnects

Mary Hall
September 1, 2011

Homework 2: Due Before Class, Thursday, Sept. 8

'handin cs4961 hw2 <file>'

Problem 1: (Coherence) #2.15 in textbook
(a) Suppose a shared-memory system uses snooping cache coherence and write-back caches. Also suppose that core 0 has the variable x in its cache, and it executes the assignment x=5. Finally, suppose that core 1 doesn’t have x in its cache, and after core 0’s update to x, core 1 tries to execute y=x. What value will be assigned to y? Why?
(b) Suppose that the shared-memory system in the previous part uses a directory-based protocol. What value will be assigned to y? Why?
(c) Can you suggest how any problems you found in the first two parts might be solved?

Homework 2, cont.

Problem 2: (Bisection width/bandwidth)
(a) What is the bisection width and bisection bandwidth of a 3-d toroidal mesh?
(b) A planar mesh is just like a toroidal mesh, except that it doesn’t have the wraparound links. What is the bisection width and bisection bandwidth of a square planar mesh?

Problem 3 (in general, not specific to any algorithm): How is algorithm selection impacted by the value of \( \lambda \)?

Administrative

• Nikhil office hours:
   - Monday, 2-3PM
   - Lab hours on Tuesday afternoons during programming assignments
• First homework will be treated as extra credit
   - If you turned it in late, or up until tomorrow, turn in for half credit
Homework 2, cont.

Problem 4: (A concept) #2.10 in textbook

Suppose a program must execute $10^{12}$ instructions in order to solve a particular problem. Suppose further that a single processor system can solve the problem in $10^6$ seconds (about 11.6 days). So, on average, the single processor system executes $10^9$ or a million instructions per second. Now suppose that the program has been parallelized for execution on a distributed-memory system. Suppose also that if the parallel program uses $p$ processors, each processor will execute $10^{12}/p$ instructions, and each processor must send $10^9(p-1)$ messages. Finally, suppose that there is no additional overhead in executing the parallel program. That is, the program will complete after each processor has executed all of its instructions and sent all its messages, and there won’t be any delays due to things such as waiting for messages.

(a) Suppose it takes $10^{-9}$ seconds to send a message. How long will it take the program to run with 1000 processors, if each processor is as fast as the single processor on which the serial program was run?

(b) Suppose it takes $10^{-2}$ seconds to send a message. How long will it take the program to run with 1000 processors?

Today’s Lecture

• Key architectural features affecting parallel performance
  - Memory systems
    - Memory access time
    - Cache coherence
    - Interconnect
  - A few more parallel architectures
    - Sunfire SMP
    - B6/L supercomputer
  - An abstract architecture for parallel algorithms
  - Sources for this lecture:
    - Textbook
    - Larry Snyder, "http://www.cs.washington.edu/education/courses/524/08wi/"

An Abstract Parallel Architecture

- How is parallelism managed?
- Where is the memory physically located?
- Is it connected directly to processors?
- What is the connectivity of the network?

Memory Systems

• There are three features of memory systems that affect parallel performance
  - Latency: Time between initiating a memory request and it being serviced by the memory (in micro-seconds).
  - Bandwidth: The rate at which the memory system can service memory requests (in GB/s).
  - Coherence: Since all processors can view all memory locations, coherence is the property that all processors see the memory image in the same state.
Uniform Memory Access (UMA) multicore system

Time to access all the memory locations will be the same for all the cores.

Non-uniform Memory Access (NUMA) multicore system

A memory location a core is directly connected to can be accessed faster than a memory location that must be accessed through another chip.

Cache coherence

Programmers have no control over caches and when they get updated.

A shared memory system with two cores and two caches

x = 2; /* shared variable */
y0 privately owned by Core 0
y1 and z1 privately owned by Core 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Core 0</th>
<th>Core 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>y0 = x;</td>
<td>y1 = 3 * x;</td>
</tr>
<tr>
<td>1</td>
<td>z = y1;</td>
<td>Statement(s) not involving x</td>
</tr>
<tr>
<td>2</td>
<td>Statement(s) not involving x</td>
<td>z1 = 4 * y1;</td>
</tr>
</tbody>
</table>

y0 eventually ends up = 2
y1 eventually ends up = 6
z1 = ???
**Snooping Cache Coherence**

- The cores share a bus.
- Any signal transmitted on the bus can be "seen" by all cores connected to the bus.
- When core 0 updates the copy of x stored in its cache it also broadcasts this information across the bus.
- If core 1 is "snooping" the bus, it will see that x has been updated and it can mark its copy of x as invalid.

**Directory Based Cache Coherence**

- Uses a data structure called a directory that stores the status of each cache line.
- When a variable is updated, the directory is consulted, and the cache controllers of the cores that have that variable's cache line in their caches are invalidated.

**Definitions for homework Problem #1**

- **Write through cache**
  - A cache line is written to main memory when it is written to the cache.
- **Write back cache**
  - The data is marked as dirty in the cache, and written back to memory when it is replaced by a new cache line.

**False Sharing**

- A cache line contains more than one machine word.
- When multiple processors access the same cache line, it may look like a potential race condition, even if they access different elements.
- Can cause coherence traffic.
Shared memory interconnects

- **Bus interconnect**
  - A collection of parallel communication wires together with some hardware that controls access to the bus.
  - Communication wires are shared by the devices that are connected to it.
  - As the number of devices connected to the bus increases, contention for use of the bus increases, and performance decreases.

**Crossbar**
- Allows simultaneous communication among different devices.
- Faster than buses.
- But the cost of the switches and links is relatively high.
- Crossbars grow as \( n^2 \) making them impractical for large \( n \).

---

**SunFire E25K**

- **4 UltraSparcs**
- Dotted lines represent snooping
- **18 boards connected with crossbars**
  - Basically the limit
  - Increasing processors per node will, on average, increase congestion
Distributed memory interconnects

- Two groups
  - Direct interconnect
    - Each switch is directly connected to a processor memory pair, and the switches are connected to each other.
  - Indirect interconnect
    - Switches may not be directly connected to a processor.
- The cost of an interconnect can be a big part of a machine, governed by
  - Number of links
  - Number of switches
  - Available bandwidth (highly optimized links are expensive)

Direct interconnect

- Figure 2.8

Message transmission time = l + n / b

Definitions

- Bisection width
  - A measure of "number of simultaneous communications" or "connectivity".
  - How many simultaneous communications can take place "across the divide" between the halves?
- Bisection bandwidth
  - A measure of network quality.
  - Instead of counting the number of links joining the halves, it sums the bandwidth of the links.
Fully connected network

- Each switch is directly connected to every other switch.

bisection width = $p^2/4$

Indirect interconnects

- Simple examples of indirect networks:
  - Crossbar
  - Omega network

- Often shown with unidirectional links and a collection of processors, each of which has an outgoing and an incoming link, and a switching network.
### A generic indirect network

**Figure 2.13**

### Supercomputer: BG/L Node

- Actually has multiple interconnects

**Figure 2.14**

### BG/L Interconnect

- Separate networks for control and data
- Can then specialize network implementation for type of message
- Also reduces congestion

### Blue Gene/L Specs

- BG/L was the fastest computer in the world (#1 on the Top500 List) when the textbook was published
- A 64x32x32 torus = 65K 2-core processors
- Cut-through routing gives a worst-case latency of 6.4 µs
- Processor nodes are dual PPC-440 with "double hummer" FPUs
- Collective network performs global reduce for the "usual" functions
Summary of Architectures

Two main classes

- Complete connection: CMPs, SMPs, X-bar
  - Preserve single memory image
  - Complete connection limits scaling to...
  - Available to everyone (multi-core)
- Sparse connection: Clusters, Supercomputers, Networked computers used for parallelism (Grid)
  - Separate memory images
  - Can grow "arbitrarily" large
  - Available to everyone with LOTS of air conditioning

Programming differences are significant.

Parallel Architecture Model

- How to develop portable parallel algorithms for current and future parallel architectures, a moving target?
- Strategy:
  - Adopt an abstract parallel machine model for use in thinking about algorithms
  1. Review how we compare algorithms on sequential architectures
  2. Introduce the CTA model (Candidate Type Architecture)
  3. Discuss how it relates to today's set of machines

How did we do it for sequential architectures?

- Sequential Model: Random Access Machine
  - Control, ALU, (Unlimited) Memory, [Input, Output]
  - Fetch/execute cycle runs 1 inst. pointed at by PC
  - Memory references are "unit time" independent of location
    - Gives RAM it's name in preference to von Neumann
    - "Unit time" is not literally true, but caches provide that illusion when effective
    - Executes "3-address" instructions
- Focus in developing sequential algorithms, at least in courses, is on reducing amount of computation (useful even if imprecise)
- Treat memory time as negligible
- Ignore overheads

Interesting Historical Parallel Architecture Model, PRAM

- Parallel Random Access Machine (PRAM)
  - Unlimited number of processors
  - Processors are standard RAM machines, executing synchronously
  - Memory reference is "unit time"
  - Outcome of collisions at memory specified
    - EREW, CREW, CRCW...
- Model fails to capture true performance behavior
  - Synchronous execution w/ unit cost memory reference does not scale
  - Therefore, parallel hardware typically implements non-uniform cost memory reference
Candidate Type Architecture (CTA Model)

- A model with \( P \) standard processors, degree, \( \lambda \) latency.
- Node == processor + memory + NIC
- Key Property: Local memory ref is 1, global memory is \( \lambda \)

Estimated Values for Lambda

- Captures inherent property that data locality is important.
- But different values of Lambda can lead to different algorithm strategies

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP AMD</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>SMP Sun Fire E25K</td>
<td>400-660</td>
<td>=&gt; cannot be ignored</td>
</tr>
<tr>
<td>Cluster Ilamion + Myiret</td>
<td>4100-5100</td>
<td></td>
</tr>
<tr>
<td>Super BlueGenEli</td>
<td>5000</td>
<td></td>
</tr>
</tbody>
</table>

Key Lesson from CTA

- Locality Rule:
  - Fast programs tend to maximize the number of local memory references and minimize the number of non-local memory references.
  - This is the most important thing you will learn in this class!

Summary of Lecture

- Memory Systems
  - UMA vs. NUMA
  - Cache coherence
- Interconnects
  - Bisection bandwidth
- Models for parallel architectures
  - CTA model