CS4961 Parallel Programming

Lecture 3: Introduction to Parallel Architectures

Mary Hall
August 31, 2010

Administrative

- Sriram office hours:
  - Monday and Wednesday, 3:30 – 4:30 PM
  - Lab hours on Tuesday afternoons during programming assignments

Homework 1 – Due 10:00 PM, Wed., Sept. 1

- To submit your homework:
  - Submit a PDF file
  - Use the "handin" program on the CADE machines
  - Use the following command:
    "handin cs4961 hw1 <prob1file>"

Problem 1:

- What are your goals after this year and how do you anticipate this class is going to help you with that? Some possible answers, but please feel free to add to them. Also, please write at least one sentence of explanation.
  - A job in the computing industry
  - A job in some other industry where computing is applied to real-world problems
  - As preparation for graduate studies
  - Intellectual curiosity about what is happening in the computing field
  - Other

Homework 1

Problem 2:

(a) Provide pseudocode (as in the book and class notes) for a correct and efficient parallel implementation in C of the parallel prefix computation (see Fig. 1.4 on page 14). Assume your input is a vector of n integers, and there are n/2 processors. Each processor is executing the same thread code, and the thread index is used to determine which portion of the vector the thread operates upon and the control. For now, you can assume that at each step in the tree, the threads are synchronized.

(b) The structure of this and the tree-based parallel sums from Figure 1.3 are similar to the parallel sort we did with the playing cards on Aug. 24. In words, describe how you would modify your solution of (a) above to derive the parallel sorting implementation.

Homework 1, cont.

Problem 3 (see supplemental notes for definitions):

Loop reversal is a transformation that reverses the order of the iterations of a loop. In other words, loop reversal transforms a loop with header

```
for (i=0; i<N; i++)
```

into a loop with the same body but header

```
for (i=N-1; i>=0; i--)
```

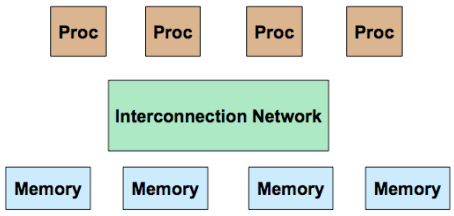
Is loop reversal a valid reordering transformation on the "i" loop in the following loop nest? Why or why not?

```
for (j=0; j<N; j++)
    for (i=0; i<N; i++)
        a[i+1][j+1] = a[i][j] + c;
```


Today's Lecture

- Some types of parallel architectures
  - SIMD vs. MIMD
  - multi-cores from Intel and AMD
  - Sunfire SMP
  - BG/L supercomputer
  - Clusters
  - Later in the semester we'll look at NVIDIA GPUs
- An abstract architecture for parallel algorithms
- Discussion
- Sources for this lecture:
  - 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?

Why are we looking at a bunch of architectures

- There is no canonical parallel computer – a diversity of parallel architectures
  - Hence, there is no canonical parallel programming language
- Architecture has an enormous impact on performance
  - And we wouldn't write parallel code if we didn't care about performance
- Many parallel architectures fail to succeed commercially
  - Can't always tell what is going to be around in N years

Challenge is to write parallel code that abstracts away architectural features, focuses on their commonality, and is therefore easily ported from one platform to another.

Retrospective (Jim Demmel)

- Historically, each parallel machine was unique, along with its programming model and programming language.
- It was necessary to throw away software and start over with each new kind of machine.
- Now we distinguish the programming model from the underlying machine, so we can write portably correct codes that run on many machines.
  - MPI now the most portable option, but can be tedious.
  - Writing portable fast code requires tuning for the architecture.
- Parallel algorithm design challenge is to make this process easy.
  - Example: picking a blocksize, not rewriting whole algorithm.

Six Parallel Architectures (from text)

Shared Memory

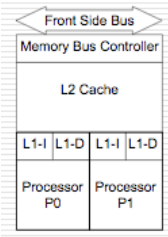
- Multi-cores
  - Intel Core2Duo
  - AMD Opteron
- Symmetric Multiprocessor (SMP)
  - SunFire E25K

Distributed Memory

- Heterogeneous Architecture (ignore for now)
  - IBM Cell B/E
- Clusters (ignore for now)
  - Commodity desktops (typically PCs) with high-speed interconnect
- Supercomputers
  - IBM BG/L

Multi-core 1: Intel Core2Duo

- 2 32-bit Pentiums
- Private 32K L1s
- Shared 2M-4M L2
- MESI cc-protocol
- Shared bus control and memory

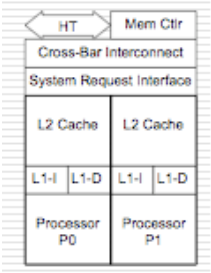


MESI Cache Coherence

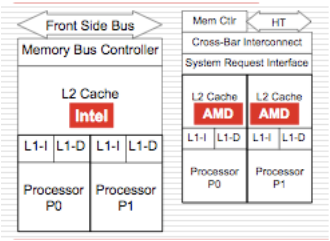
- States: M=modified; E=exclusive; S=shared; I=invalid;
- Think of it as a finite state machine
  - Upon loading, a line is marked E, subsequent reads are OK; write marks M
  - Seeing another load, mark as S
  - A write to an S, sends I to all, marks as M
  - Another's read to an M line, writes it back, marks it S
- Read/write to an I misses
- Related scheme: MOESI (O = owned; used by AMD)

AMD Dual-Core Opteron (in CHPC systems)

- 2 64-bit Opterons
- 64K private L1s
- 1 MB private L2s
- MOESI cc-protocol
- Direct connect shared memory



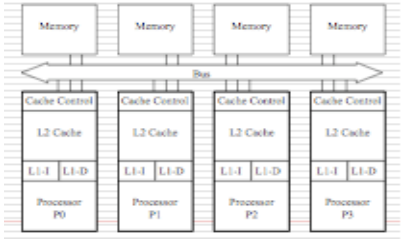
Comparison



- Fundamental difference in memory hierarchy structure

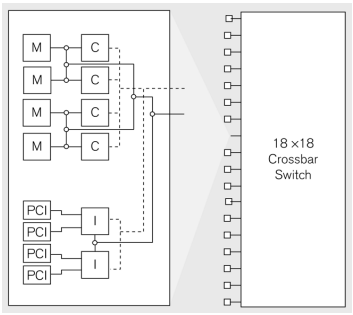
Classical Shared-Memory, Symmetric Multiprocessor (SMP)

- All processors connected by a bus
- Cache coherence maintained by "snooping" on the bus
- Serializes communication



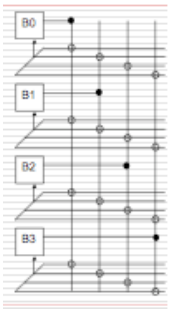
SunFire E25K

- 4 UltraSparcs
- Dotted lines represent snooping
- 18 boards connected with crossbars



Crossbar Switch

- A crossbar is a network connecting each processor to every other processor
- Crossbars grow as n^2 making them impractical for large n

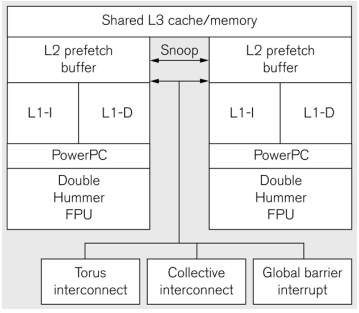


Crossbar in SunFire E25K

- X-bar gives low latency for snoops allowing for shared memory
- 18 x 18 X-bar is basically the limit
- Raising the number of processors per node will, on average, increase congestion
- How could we make a larger machine?

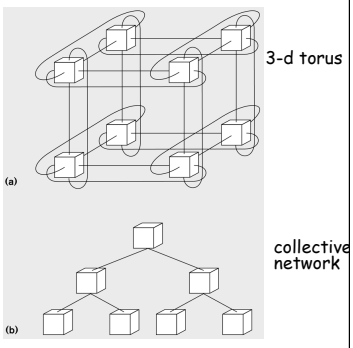
Supercomputer: BG/L Node

- How is this different from multi-cores?



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?

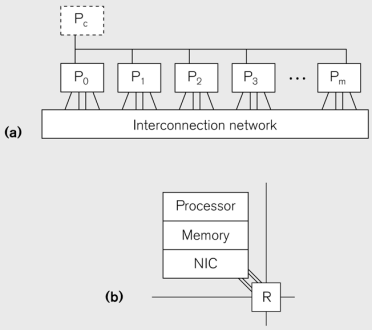
- Sequential Model: Random Access Machine
  - Control, ALU, (Unlimited) Memory, [Input, Output]
  - Fetch/execute cycle runs 1 inst. pointed at by PC
  - Memory references are "unit time" independent of location
    - Gives RAM it's name in preference to von Neumann
    - "Unit time" is not literally 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, d degree, λ latency*
- Node == processor + memory + NIC
- Key Property: Local memory ref is 1, global memory is λ



Estimated Values for Lambda

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

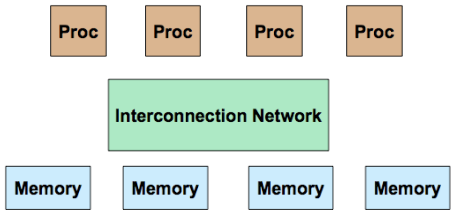
CMP	AMD	100
SMP	Sun Fire E25K	400-660
Cluster	Itanium + Myrinet	4100-5100
Super	BlueGene/L	5000

Lg λ range => cannot be ignored

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!

Back to Abstract Parallel Architecture



- Key feature is then how memory is connected processors!

Memory Reference Mechanisms

- Shared Memory
  - All processors have access to a global address space
  - Refer to remote data or local data in the same way, through normal loads and stores
  - Usually, caches must be kept *coherent* with global store
- Message Passing & Distributed Memory
  - Memory is partitioned and a partition is associated with an individual processor
  - Remote data access through explicit communication (sends and receives)
  - Two-sided (both a send and receive are needed)
- One-Sided Communication (a hybrid mechanism)
  - Supports a global shared address space but no coherence guarantees
  - Access to remote data through gets and puts

Brief Discussion

- Why is it good to have different parallel architectures?
  - Some may be better suited for specific application domains
  - Some may be better suited for a particular community
  - Cost
  - Explore new ideas
- And different programming models/ languages?
  - Relate to architectural features
  - Application domains, user community, cost, exploring new ideas

Summary of Lecture

- Exploration of different kinds of parallel architectures
- Key features
  - How processors are connected?
  - How memory is connected to processors?
  - How parallelism is represented/managed?
- Models for parallel architectures