

CS4961 Parallel Programming

Lecture 3: Introduction to Parallel Architectures

Mary Hall
September 1, 2009

09/01/2009

CS4961

1

Administrative

- Homework 1 posted, due September 3 before class
- Use the "handin" program on the CADE machines
- Use the following command:
"handin cs4961 hw1 <prob1file>"
- Waiving CS4400 prerequisite, replacing with CS3810
- Textbooks are in bookstore! (as of Friday)
- What to call me

09/01/2009

CS4961

2



Homework 1 - Due 9:10 AM, Thursday, Sept. 3

- 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

09/01/2009

CS4961

3



Homework 1

- Problem 2:
 - Provide pseudocode (as in the book and class notes) for a correct and efficient parallel implementation in C of the parallel sums code, based on the tree-based concept in slides 26 and 27 of Lecture 2. Assume that you have an array of 128 elements and you are using 8 processors.
 - Hints:
 - Use an iterative algorithm similar to count3s, but use the tree structure to accumulate the final result.
 - Use the book to show you how to add threads to what we derived for count3s.
- Problem 3:
 - Now show how the same algorithm can be modified to find the maximum element of an array. (problem 2 in text). Is this also a reduction computation? If so, why?

09/01/2009

CS4961

4



Questions on Homework?

- Hint: Barrier synchronization (see pgs. 91-92)

09/01/2009

CS4961

5



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/>"

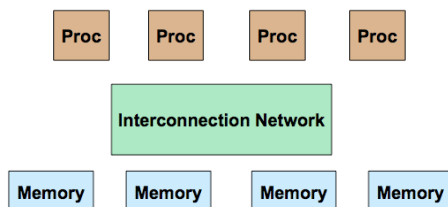
09/01/2009

CS4961

6



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?

09/01/2009

CS4961

7



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.

09/01/2009

CS4961

8



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.

09/01/2009

CS4961

9



Predominant Parallel Control Mechanisms

Name	Meaning	Examples
Single Instruction, Multiple Data (SIMD)	A single thread of control, same computation applied across "vector" elts	Array notation as in Fortran 90: $A[1:n] = A[1:n] + B[1:n]$
Multiple Instruction, Multiple Data (MIMD)	Multiple threads of control, processors periodically synch	Parallel loop: <code>forall (i=0; i<n; i++)</code>
Single Program, Multiple Data (SPMD)	Multiple threads of control, but each processor executes same code	Processor-specific code: <code>if (\$myid == 0) {</code> <code>}</code>

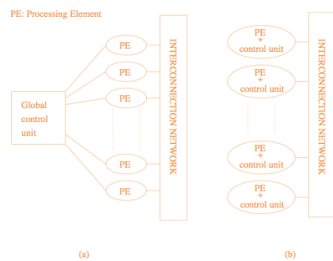
09/01/2009

CS4961

10



SIMD and MIMD Architectures: What's the Difference?



A typical SIMD architecture (a) and a typical MIMD architecture (b).

Slide source: Grama et al., Introduction to Parallel Computing,
<http://www.users.cs.umn.edu/~karypis/parbook>

09/01/2009

CS4961

11



A Few More Words About SIMD (more soon)

- Historically, SIMD architectures at the high end included large vector machines, which are not widely used
- Today SIMD architectures are very popular at a smaller scale
 - Multimedia extensions such as Intel SSE-3
 - GPUs such as Nvidia Tesla
 - Coming: Intel Larrabee
- Interestingly, these architectures use fundamentally different programming models for expressing parallelism

09/01/2009

CS4961

12



Six Parallel Architectures (from text)

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

Shared Memory

Distributed Memory

09/01/2009

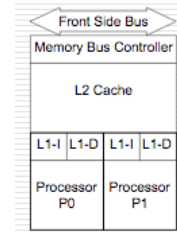
CS4961

13



Multi-core 1: Intel Core2Duo

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



09/01/2009

CS4961

14



MESI Cache Coherence

- States: M=modified; E=exclusive; S=shared; I=invalid;
- 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)

09/01/2009

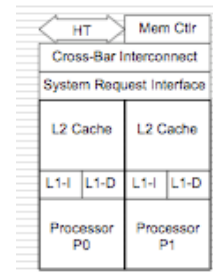
CS4961

15



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



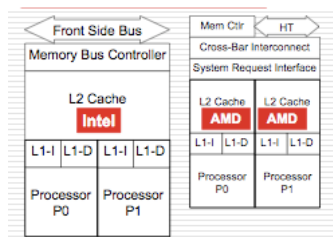
09/01/2009

CS4961

16



Comparison



- Fundamental difference in memory hierarchy structure

09/01/2009

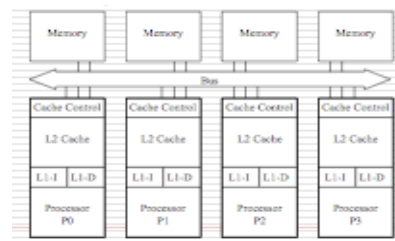
CS4961

17



Classical Shared-Memory, Symmetric Multiprocessor (SMP)

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



09/01/2009

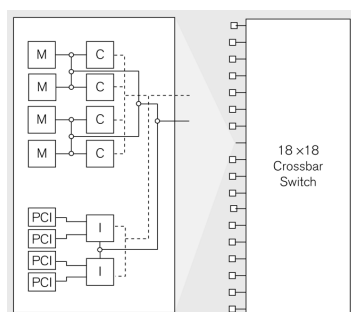
CS4961

18



SunFire E25K

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



09/01/2009

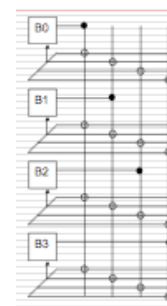
CS4961

19



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



09/01/2009

CS4961

20



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?

09/01/2009

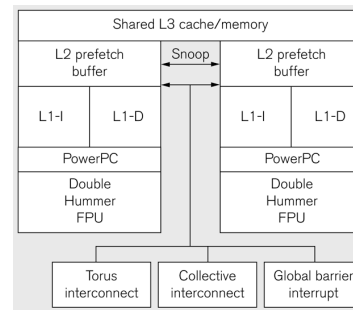
CS4961

21



Supercomputer: BG/L Node

- How is this different from multi-cores?



09/01/2009

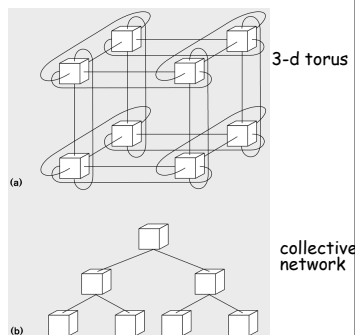
CS4961

22



BG/L Interconnect

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



09/01/2009

CS4961

23



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 μ s
- Processor nodes are dual PPC-440 with "double hummer" FPU's
- Collective network performs global reduce for the "usual" functions

09/01/2009

CS4961

24



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

09/01/2009

CS4961

25



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
- First step: Review how we use von Neumann
- Second step: Introduce the CTA
- Third step: Discuss how it relates to today's set of machines

09/01/2009

CS4961

26



How did we do it for sequential architectures?

- The 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
 - Ignores overheads

09/01/2009

CS4961

27



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 bc synchronous execution w/ unit cost memory reference does not scale

09/01/2009

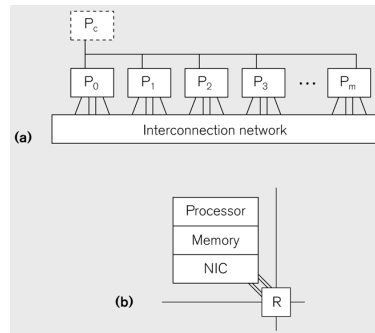
CS4961

28



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



09/01/2009

CS4961

29



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	} Lg λ range => cannot be ignored
SMP	Sun Fire E25K	400-660	
Cluster	Itanium + Myrinet	4100-5100	
Super	BlueGene/L	5000	

09/01/2009

CS4961

30



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

09/01/2009

CS4961

31



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

09/01/2009

CS4961

32

