COMP 605: Introduction to Parallel Computing
(Part 2)
Lecture 07: Parallel Hardware Architectures
Part 2

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Presented: 02/03/15
Updated: 02/16/15
Table of Contents

1 Misc Information
2 Parallel Hardware Arch: Part 2
   - Flynn’s Taxonomy: SIMD
     - Vector Processors
     - GPUs
   - Flynn’s Taxonomy: MIMD
     - Shared Memory
     - Distributed Memory
   - Interconnection Networks
   - Cache Coherence
3 Next Time
HW 1 due Friday, 02/05/16

Student accounts on tuckoo should be assigned.

Check that you are getting emails from the Blackboard (BB) mailing list.

Check that you are getting emails from the Google Group (GG) mailing list.
Flynn’s Taxonomy: SIMD

Source:

http://en.wikipedia.org/wiki/Flynn’s_taxonomy
Single Instruction Single Data
Parallelism achieved by dividing data among the processors.

- Applies the same instruction to multiple data items.
- Called data parallelism
SIMD example

for (i = 0; i < n; i++)
    x[i] += y[i];
SIMD Data Distribution

- What if we don't have as many ALUs as data items?
- Divide the work and process iteratively.
- Ex. \( m = 4 \) ALUs and \( n = 14 \) data items.

<table>
<thead>
<tr>
<th>Round</th>
<th>ALU1</th>
<th>ALU2</th>
<th>ALU3</th>
<th>ALU4</th>
</tr>
</thead>
</table>
SIMD drawbacks

- All ALUs are required to execute the same instruction, or remain idle.
- In classic design, they must also operate synchronously.
- The ALUs have no instruction storage.
- Efficient for large data parallel problems, but not other types of more complex parallel problems.
Vector processors (1)

- Operate on arrays or vectors of data while conventional CPU’s operate on individual data elements or scalars.

- Vector registers.
  - Capable of storing a vector of operands and operating simultaneously on their contents.
Vector processors (2)

- Vectorized and pipelined functional units.
  - The same operation is applied to each element in the vector (or pairs of elements).

- Vector instructions.
  - Operate on vectors rather than scalars.
Vector processors (3)

- Interleaved memory.
  - Multiple “banks” of memory, which can be accessed more or less independently.
  - Distribute elements of a vector across multiple banks, so reduce or eliminate delay in loading/storing successive elements.
- Strided memory access and hardware scatter/gather.
  - The program accesses elements of a vector located at fixed intervals.
Vector processors - Pros

- Fast.
- Easy to use.
- Vectorizing compilers are good at identifying code to exploit.
- Compilers also can provide information about code that cannot be vectorized.
  - Helps the programmer re-evaluate code.
- High memory bandwidth.
- Uses every item in a cache line.
Vector processors - Cons

- They don’t handle irregular data structures as well as other parallel architectures.

- A very finite limit to their ability to handle ever larger problems. (scalability)
The Cray-1 Vector Computer:

- First vector machine (1975)
- $8.86$ million
- appx $140$ MFlops, for weather calculation!!
- load a lot of data into memory, perform a lot of ops on that data
- Freon liquid cooling
- $12$ functional units (address, scalar, vector, and floating point)
Graphics Processing Units (GPU)

- Real time graphics application programming interfaces or API’s use points, lines, and triangles to internally represent the surface of an object.
GPUs

- A graphics processing pipeline converts the internal representation into an array of pixels that can be sent to a computer screen.

- Several stages of this pipeline (called shader functions) are programmable.
  - Typically just a few lines of C code.
GPUs

- Shader functions are also implicitly parallel, since they can be applied to multiple elements in the graphics stream.

- GPU’s can often optimize performance by using SIMD parallelism.

- The current generation of GPU’s use SIMD parallelism.
  - Although they are not pure SIMD systems.
**NVIDIA GPU GF100**

High-Level Block Diagram (2010)

- CPU is called the host and the cores in the GPU are called devices
- 4 "GPC" clusters
- Many SM (stream multiprocessors) each with SPs
- 512 CUDA stream processors (SPs) or cores
- SIMT (single instr. multiple thread)

Source: [http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview](http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview)
NVIDIA GPU

- Each SM core in each GPC is comprised of 32 CUDA cores
- 48/16KB of shared memory (3x that of GT200),
- 16/48KB of L1 (there is no L1 cache on GT200),

Source: http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview
MIMD

- Supports multiple simultaneous instruction streams operating on multiple data streams.

- Typically consist of a collection of fully independent processing units or cores, each of which has its own control unit and its own ALU.
Parallel Hardware Arch: Part 2
Flynn's Taxonomy: MIMD

Multiple Instruction Multiple Data
Shared Memory System (1)

- A collection of autonomous processors is connected to a memory system via an interconnection network.
- Each processor can access each memory location.
- The processors usually communicate implicitly by accessing shared data structures.
Shared Memory System (2)

- Most widely available shared memory systems use one or more multicore processors.
- (multiple CPU’s or cores on a single chip)
Shared Memory System

CPU

CPU

CPU

...

Interconnect

Memory

Figure 2.3
**UMA multicore system**

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

![Diagram of UMA multicore system](image)

**Figure 2.5**
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.

Figure 2.6
Distributed Memory System

- **Clusters** (most popular)
  - A collection of commodity systems.
  - Connected by a commodity interconnection network.

- **Nodes** of a cluster are individual computations units joined by a communication network.

  *a.k.a. hybrid systems*
Distributed Memory System

![Distributed Memory System Diagram]

Figure 2.4
Mac Book Pro - Intel Core

- Intel Core i7, Z77 chipset
- 4 cores, 8 hyperthreads
- $32 + 32$ KB L1 cache for data and instructions (per core)
- $256$ KByte L2 cache (per core)
- $8$ MB L3 cache (split up between cores and GPU)

Source: http://http://www.notebookcheck.net/Review-Intel-Ivy-Bridge-Quad-Core-Processors.73624.0.html
Interconnection networks

- Affects performance of both distributed and shared memory systems.

- Two categories:
  - Shared memory interconnects
  - Distributed memory interconnects
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.
Shared memory interconnects

- Switched interconnect
  - Uses switches to control the routing of data among the connected devices.

- Crossbar –
  - Allows simultaneous communication among different devices.
  - Faster than buses.
  - But the cost of the switches and links is relatively high.
Figure 2.7

(a) A crossbar switch connecting 4 processors (P_i) and 4 memory modules (M_j)

(b) Configuration of internal switches in a crossbar

(c) Simultaneous memory accesses by the processors
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.
Direct interconnect

Figure 2.8

(a) ring

(b) toroidal mesh
Bisection width

- A measure of “number of simultaneous communications” or “connectivity”.

- How many simultaneous communications can take place “across the divide” between the halves?
Two bisections of a ring

Figure 2.9
A bisection of a toroidal mesh

Figure 2.10
Definitions

- **Bandwidth**
  - The rate at which a link can transmit data.
  - Usually given in megabits or megabytes per second.

- **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 = $p^2/4$

Figure 2.11
Hypercube

- Highly connected direct interconnect.
- Built inductively:
  - A one-dimensional hypercube is a fully-connected system with two processors.
  - A two-dimensional hypercube is built from two one-dimensional hypercubes by joining “corresponding” switches.
  - Similarly a three-dimensional hypercube is built from two two-dimensional hypercubes.
Hypercubes

Figure 2.12

(a) one-

(b) two-

(c) three-dimensional
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
Crossbar interconnect for distributed memory

Figure 2.14
An omega network

Figure 2.15
A switch in an omega network

Figure 2.16
More definitions

- Any time data is transmitted, we’re interested in how long it will take for the data to reach its destination.

- Latency
  - The time that elapses between the source’s beginning to transmit the data and the destination’s starting to receive the first byte.

- Bandwidth
  - The rate at which the destination receives data after it has started to receive the first byte.
Message transmission time = \( l + \frac{n}{b} \)

- \( l \): latency (seconds)
- \( n \): length of message (bytes)
- \( b \): bandwidth (bytes per second)
ORNL Titan Supercomputer - Jaguar upgrade

- 38,400-processors, 307,200 CPU cores
- 20-petaflop [ AMD Opteron 6200 ]
- Cray Gemini Interconnect
  - processor-to-processor
  - Optical network of 2x2 switches
  - Banyan: $O(N \log N)$

Source: http://www.extremetech.com/extreme/99413-titan-supercomputer-38400-processor-20-petaflop-successor-to-jaguar
Cache coherence

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

Figure 2.17

A shared memory system with two cores and two caches
Cache coherence

- \( y_0 \) privately owned by Core 0
- \( y_1 \) and \( z_1 \) privately owned by Core 1

\[ x = 2; /* \text{shared variable} */ \]

<table>
<thead>
<tr>
<th>Time</th>
<th>Core 0</th>
<th>Core 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( y_0 = x; )</td>
<td>( y_1 = 3 \times x; )</td>
</tr>
<tr>
<td>1</td>
<td>( x = 7; )</td>
<td>Statement(s) not involving ( x )</td>
</tr>
<tr>
<td>2</td>
<td>Statement(s) not involving ( x )</td>
<td>( z_1 = 4 \times x; )</td>
</tr>
</tbody>
</table>

- \( y_0 \) eventually ends up = 2
- \( y_1 \) eventually ends up = 6
- \( z_1 \) = ???
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.
Next Time

- Next class: 02/05/16
- HW 1 due Friday, 02/05/16. Turn in printed copy at start of class
- Read: Ch2, Pacheco 2011, "An Introduction to Parallel Programming."