Parallel Processing: The Holy Grail

- Use multiple processors to improve runtime of a single task
  - Available technology limits speed of uniprocessor
  - Economic advantages to using replicated processing units
- Preferably programmed using a portable high-level language
Flynn’s Classification (1966)

Broad classification of parallel computing systems based on number of instruction and data streams

• SISD: Single Instruction, Single Data
  – conventional uniprocessor

• SIMD: Single Instruction, Multiple Data
  – one instruction stream, multiple data paths
  – distributed memory SIMD (MPP, DAP, CM-1&2, Maspar)
  – shared memory SIMD (STARAN, vector computers)

• MIMD: Multiple Instruction, Multiple Data
  – message passing machines (Transputers, nCube, CM-5)
  – non-cache-coherent shared memory machines (BBN Butterfly, T3D)
  – cache-coherent shared memory machines (Sequent, Sun Starfire, SGI Origin)

• MISD: Multiple Instruction, Single Data
  – Not a practical configuration

SIMD Architecture

• Central controller broadcasts instructions to multiple processing elements (PEs)

  - Only requires one controller for whole array
  - Only requires storage for one copy of program
  - All computations fully synchronized
SIMD Machines

• Illiac IV (1972)
  – 64 64-bit PEs, 16KB/PE, 2D network
• Goodyear STARAN (1972)
  – 256 bit-serial associative PEs, 32B/PE, multistage network
• ICL DAP (Distributed Array Processor) (1980)
  – 4K bit-serial PEs, 512B/PE, 2D network
• Goodyear MPP (Massively Parallel Processor) (1982)
  – 16K bit-serial PEs, 128B/PE, 2D network
• Thinking Machines Connection Machine CM-1 (1985)
  – 64K bit-serial PEs, 512B/PE, 2D + hypercube router
  – CM-2: 2048B/PE, plus 2,048 32-bit floating-point units
• Maspar MP-1 (1989)
  – 16K 4-bit processors, 16-64KB/PE, 2D + Xnet router
  – MP-2: 16K 32-bit processors, 64KB/PE

(Also shared memory SIMD vector supercomputers
  TI ASC ('71), CDC Star-100 ('73), Cray-1 ('76))

SIMD Machines Today

• Distributed-memory SIMD failed as large-scale general-purpose computer platform
  – required huge quantities of data parallelism (>10,000 elements)
  – required programmer-controlled distributed data layout
• Vector supercomputers (shared-memory SIMD) still successful in high-end supercomputing
  – reasonable efficiency on short vector lengths (10-100 elements)
  – single memory space
• Distributed-memory SIMD popular for special-purpose accelerators
  – image and graphics processing
• Renewed interest for Processor-in-Memory (PIM)
  – memory bottlenecks => put some simple logic close to memory
  – viewed as enhanced memory for conventional system
  – technology push from new merged DRAM + logic processes
  – commercial examples, e.g., graphics in Sony Playstation-2/3
MIMD Machines

Multiple independent instruction streams, two main kinds:

- Message passing
- Shared memory
  - no hardware global cache coherence
  - hardware global cache coherence

Message Passing MPPs
*(Massively Parallel Processors)*

- Initial Research Projects
  - Caltech Cosmic Cube (early 1980s) using custom Mosaic processors
- Commercial Microprocessors including MPP Support
  - Transputer (1985)
- Standard Microprocessors + Network Interfaces
  - Intel Paragon/i860 (1991)
  - TMC CM-5/SPARC (1992)
  - Meiko CS-2/SPARC (1993)
  - IBM SP-1/POWER (1993)
- MPP Vector Supers
  - Fujitsu VPP500 (1994)

*Designs scale to 100s-10,000s of nodes*
Message Passing MPP Problems

- All data layout must be handled by software
  - cannot retrieve remote data except with message request/reply
- Message passing has high software overhead
  - early machines had to invoke OS on each message (100μs-1ms/message)
  - even user level access to network interface has dozens of cycles overhead (NI might be on I/O bus)
  - sending messages can be cheap (just like stores)
  - receiving messages is expensive, need to poll or interrupt

The Earth Simulator (2002)

8 Processors/Node

NEC SX-6
Vector Microprocessor
500MHz / 1GHz
8 lanes
8 GFLOPS

256 GB/s Shared Memory BW
16GB in 2048 Memory Banks

640x640 Node Full Crossbar Interconnect

12 GB/s Each Way

83,200 cables to connect crossbar!

Was World's fastest supercomputer, >35 TFLOPS on LINPACK (June 2002)

(87% of peak performance)
The Earth Simulator (2002)

[Earth Simulator Center]

IBM Blue Gene/L Processor

11/27/2007 11

11/27/2007 12
**BG/L 64K Processor System**

- Peak Performance 360 TFLOPS
- Power Consumption 1.4 MW

**Shared Memory Machines**

- Two main categories
  - non cache coherent
  - hardware cache coherent
- Will work with any data placement (but might be slow)
  - can choose to optimize only critical portions of code
- Load and store instructions used to communicate data between processes
  - no OS involvement
  - low software overhead
- Usually some special synchronization primitives
  - fetch&op
  - load linked/store conditional
- In large scale systems, the logically shared memory is implemented as physically distributed memory modules
Cray T3E (1996)
follow-on to earlier T3D (1993) using 21064’s

Up to 2,048 675MHz Alpha 21164 processors connected in 3D torus

- Each node has 256MB-2GB local DRAM memory
- Load and stores access global memory over network
- Only local memory cached by on-chip caches
- Alpha microprocessor surrounded by custom “shell” circuitry to make it into effective MPP node. Shell provides:
  - multiple stream buffers instead of board-level (L3) cache
  - external copy of on-chip cache tags to check against remote writes to local memory, generates on-chip invalidates on match
  - 512 external E registers (asynchronous vector load/store engine)
  - address management to allow all of external physical memory to be addressed
  - atomic memory operations (fetch&op)
  - support for hardware barriers/eureka to synchronize parallel tasks

Cray XT5 (2007)

Basic Compute Node, with 2 AMD x86 Opterons

Cray SeaStar2+ Architecture

- 6-Port Router
- Bridge Control Processor Interface
- Memory
- PowerPC 440 Processor
- DMA Engine
- HyperTransport Interface

Vector Node
4-way SMP of SX2 Vector CPUs (8 lanes each)

Reconfigurable Logic Node
2 FPGAs + Opteron

Also, XMT Multithreaded Nodes based on MTA design (128 threads per processor)
Processor plugs into Opteron socket
Bus-Based Cache-Coherent SMPs

- Small scale (<= 4 processors) bus-based SMPs by far the most common parallel processing platform today
- Bus provides broadcast and serialization point for simple snooping cache coherence protocol
- Modern microprocessors integrate support for this protocol

Sun Starfire UE10000

(1997)

Up to 64-way SMP using bus-based snooping protocol

- Uses 4 interleaved address busses to scale snooping protocol
- Separate data transfer over high bandwidth crossbar

4 processors + memory module per system board
SGI Origin 2000
(1996)

- Large scale distributed directory SMP
- Scales from 2 processor workstation to 512 processor supercomputer

**Node contains:**
- Two MIPS R10000 processors plus caches
- Memory module including directory
- Connection to global network
- Connection to I/O

**Scalable hypercube switching network**
supports up to 64 two-processor nodes (128 processors total)
(Some installations up to 512 processors)

---

**Origin Directory Representation**
(based on Stanford DASH)

<table>
<thead>
<tr>
<th>H</th>
<th>S</th>
<th>bit-vector C</th>
<th>Memory Block</th>
</tr>
</thead>
</table>

Bit-vector is a representation of which children caches have copies of this memory block

**At home**
(H=1, S= _ ) : no cached copy exists  (R[e])

**Read Only Copies**
(H=0, S=1) : for all C_i=1, i^{th} child has a copy  (R[Dir])

**Writable Copy at C_i**
(H=0, S=0) : for C_i=1, i^{th} child has the Ex copy  (W[id])

size?
Directory Size

Directory size = (M / B) \cdot [s+N] / 8\text{ Bytes}

where

\begin{align*}
M &= \text{Memory size} \\
B &= \text{Block size} \\
N &= \text{number of children} \\
s &= \text{no. of bits to represent the state}
\end{align*}

For M= 2^{32}\text{ Bytes}, B=64\text{ Bytes}, s = 2\text{ bits}

\begin{align*}
\text{Directory size} &= (2^{(32-6)}) \cdot (2+N) / 8\text{ Bytes} \\
&= 2^{23} \cdot (2+N) \text{ Bytes}
\end{align*}

N=16 \Rightarrow \text{directory} \approx 2^{27}\text{ Bytes or } \approx 4\% \text{ overhead}
N=256 \Rightarrow \text{directory} \approx 2^{31}\text{ Bytes or } \approx 50\% \text{ overhead}

This directory data structure is practical for small N but does not scale well!
(Origin shares 1 bit per 2 processors for \leq 64 processors, 1 bit per 8 processors in 512 processor version)

Reducing the Directory Size

Limitless directories- Alewife, MIT

Instead of a N-bit-vector, keep n (lg N- bit) pointers; if more than n children request a copy, handle the overflow in software

effective for large N and low degree of sharing
Reducing the Directory Size

linked-list - SCI (Scaleable Coherent Interface)

- Part of the directory is attached to each cache
- Home and each cache block keep two \((\log N)\)-bit pointers per memory block
- A **doubly linked-list** of the cache blocks holding the same memory block is maintained, with the root at the home site

\[\Rightarrow \text{less storage but bad performance for many readers}\]

---

SGI Altix

*(evolution of Origin systems)*

- Intel Itanium-based large-scale SMP systems
- Up to 512 sockets (1024 cores) in single directory-based NUMA cached shared-memory system
- Runs single instance of Linux over 1024 cores, with up to 128TB of DRAM
- Also supports FPGA reconfigurable nodes attached to shared memory system
Diseconomies of Scale

- Few customers require the largest machines
  - much smaller volumes sold
  - have to amortize development costs over smaller number of machines
- Different hardware required to support largest machines
  - dedicated interprocessor networks for message passing MPPs
  - T3E shell circuitry
  - large backplane for Starfire
  - directory storage and routers in SGI Origin

⇒ Large machines cost more per processor than small machines!

Clusters and Networks of Workstations

Connect multiple complete machines together using standard fast interconnects
  - Little or no hardware development cost
  - Each node can boot separately and operate independently
  - Interconnect can be attached at I/O bus (most common) or on memory bus (higher speed but more difficult)

Clustering initially used to provide fault tolerance

Clusters of SMPs (CluMPs)
  - Connect multiple n-way SMPs using a cache-coherent memory bus, fast message passing network or non cache-coherent interconnect

Build message passing MPP by connecting multiple workstations using fast interconnect connected to I/O Bus. *Main advantage?*
Taxonomy of Large Multiprocessors

- Larger multiprocessors
  - Shared address space
  - Distributed address space
    - Symmetric shared memory (SMP)
      - Examples: IBM eServer, SUN SunFire
    - Distributed shared memory (DSM)
    - Commodity clusters: Raowulf and others
    - Custom cluster
      - Cache coherent: ccNUMA: SGI Origin/Altix
      - Noncache coherent: Cray T3E, X1
      - Uniform cluster: IBM BlueGene
      - Constellation cluster of DSMs or SMPs: SGI Altix, ASC Purple

Portable Parallel Programming?

- Most large scale commercial installations emphasize throughput
  - database servers, web servers, file servers
  - independent transactions
- Wide variety of parallel systems
  - message passing
  - shared memory
  - shared memory within node, message passing between nodes

⇒ *Little commercial software support for portable parallel programming*

Message Passing Interface (MPI) standard widely used for portability
- lowest common denominator
- “assembly” language level of parallel programming
CS252 Administrivia

- Midterm results

CS252 Midterm 2 Problem 1 Distribution
18 points total

Scores

% of Students

0-3  4-7  8-11  12-15  16-18
CS252 Midterm 2 Problem 4 Distribution
20 points total

Scores

% of students

0-5
6-10
11-15
16-20

CS252 Midterm 2 Distribution
80 points total
Ave: 67.3  Median: 71

Scores

% of students

0-5 6-10 11-15 16-20 17-20 21-25 26-30 31-35 36-40 41-45 46-50 51-55 56-60 61-65 66-70 71-75 76-80
CS252 Administrivia

• Presentations, Thursday December 6th, 203 McLaughlin
  – 20 minute slots: 16 minute talk + 4 minute questions/changeover
  – Practice your timing and make sure to focus on getting your message over. I will be ruthless with time for presentations, & will cut you off if you try to go over (practice for the real world of giving conference talks).
  – If your groups has more than one speaker, make sure all slides are on one file on one laptop
  – Make sure to bring any video dongles needed for your laptop

• Presentation sessions:
  9:40am-11:00am Zou, Liu, Killebrew, Kin
  11:10am-12:30pm Huang, Beamer, Lee, Antonelli
  (12:30pm- 2:00pm lunch break)
  2:00pm- 3:30pm Sturton, Hindman, Limaye, Bird

• I would like as many students as possible to attend the sessions (mandatory for your session)
• Must also mail presentation slides by December 10th

CS252 Administrivia

• Final project reports
  – 10 page, ACM-conference style papers (double column format)
  – Must be in PDF format (no .doc, or .docx)
  – Email PDF file to Krste and Rose by 11:59:59pm on Monday December 10, NO EXTENSIONS
  – Give your PDF attachment a distinctive name (e.g., <first-author-surname>.pdf)

• Send presentation slides also
Parallel Chip-Scale Processors

- Multicore processors emerging in general-purpose market due to power limitations in single-core performance scaling
  - 2-8 cores in 2007, connected as cache-coherent SMP
- Also, many embedded applications require large amounts of computation
  - Recent trend to build “extreme” parallel processors with dozens to hundreds of parallel processing elements on one die
  - Often connected via on-chip networks, with no cache coherence
- Fusion of two streams likely to form dominant type of chip architecture in future
  - Parallel processing entering the mainstream now

T1 (“Niagara”)

- Target: Commercial server applications
  - High thread level parallelism (TLP)
    » Large numbers of parallel client requests
  - Low instruction level parallelism (ILP)
    » High cache miss rates
    » Many unpredictable branches
    » Frequent load-load dependencies
- Power, cooling, and space are major concerns for data centers
- Approach: Multicore, Fine-grain multithreading, Simple pipeline, Small L1 caches, Shared L2
**T1 Architecture**

- Also ships with 6 or 4 processors

![Diagram of T1 Architecture]

**T1 pipeline**

- Single issue, in-order, 6-deep pipeline: F, S, D, E, M, W
- 3 clock delays for loads & branches.
- Shared units:
  - L1 $, L2 $
  - TLB
  - X units
  - pipe registers

- Hazards:
  - Data
  - Structural

11/27/2007
T1 Fine-Grained Multithreading

- Each core supports four threads and has its own level one caches (16KB for instructions and 8 KB for data)
- Switching to a new thread on each clock cycle
- Idle threads are bypassed in the scheduling
  - Waiting due to a pipeline delay or cache miss
  - Processor is idle only when all 4 threads are idle or stalled
- Both loads and branches incur a 3 cycle delay that can only be hidden by other threads
- A single set of floating-point functional units is shared by all 8 cores
  - floating-point performance was not a focus for T1
  - (New T2 design has FPU per core)

Memory, Clock, Power

- 16 KB 4 way set assoc. I$/ core
- 8 KB 4 way set assoc. D$/ core
- 3MB 12 way set assoc. L2 $ shared
  - 4 x 750KB independent banks
  - crossbar switch to connect
  - 2 cycle throughput, 8 cycle latency
  - Direct link to DRAM & Jbus
  - Manages cache coherence for the 8 cores
  - CAM-based directory
- Write through
  - allocate LD
  - no-allocate ST
- Coherency is enforced among the L1 caches by a directory associated with each L2 cache block
- Used to track which L1 caches have copies of an L2 block
- By associating each L2 with a particular memory bank and enforcing the subset property, T1 can place the directory at L2 rather than at the memory, which reduces the directory overhead
- L1 data cache is write-through, only invalidation messages are required; the data can always be retrieved from the L2 cache
- 1.2 GHz at ~72W typical, 79W peak power consumption
Embedded Parallel Processors

- Often embody a mixture of old architectural styles and ideas
- Exposed memory hierarchies and interconnection networks
  - Programmers code to the “metal” to get best cost/power/performance
  - Portability across platforms less important
- Customized synchronization mechanisms
  - Interlocked communication channels (processor blocks on read if data not ready)
  - Barrier signals
  - Specialized atomic operation units
- Many more, simpler cores


- Target market is wireless basestations
- 430 cores on one die in 130nm
- Each core is a 3-issue VLIW
Cisco CSR-1 Metro Chip

188 usable RISC-like cores (out of 192 on die) in 130nm

IBM Cell Processor (Playstation-3)

One 2-way threaded PowerPC core (PPE), plus eight specialized short-SIMD cores (SPE)
Nvidia G8800 Graphics Processor

• Each of 16 cores similar to a vector processor with 8 lanes (128 stream processors total)
  – Processes threads in SIMD groups of 32 (a "warp")
  – Some stripmining done in hardware
• Threads can branch, but loses performance compared to when all threads are running same code
• Only attains high efficiency on very data-parallel code (10,000s operations)

If and how will these converge?

• General-purpose multicores organized as traditional SMPs
• Embedded manycores with exposed and customized memory hierarchies

• Biggest current issue in computer architecture - will mainly be decided by applications and programming models
End of CS252 Lectures

Thanks!

Feedback, anonymous or not, welcome