

Convergence of Parallel Architectures

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

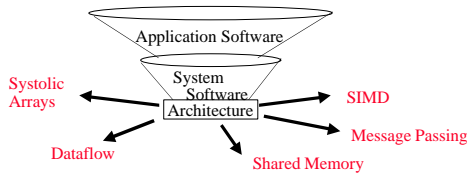
Recap of Lecture 1

- Parallel Comp. Architecture driven by familiar technological and economic forces
 - application/platform cycle, but focused on the most demanding applications
 - hardware/software learning curve
 - More attractive than ever because 'best' building - the microprocessor - is also the fastest BB.
 - History of microprocessor architecture is
 - translates area and density into performance
 - The Future is higher levels of parallelism
 - Parallel Architecture concepts apply at many levels
 - Communication also on exponential curve
- ⇒ Quantitative Engineering approach



History

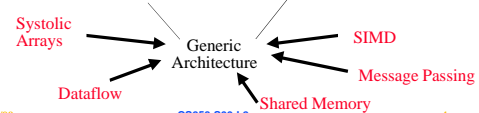
- Parallel architectures tied closely to programming models
 - Divergent architectures, with no predictable pattern of
 - Mid 80s renaissance



1/26/99 CS258 S99 L2 3

Plan for Today

- Look at major programming models
 - where did they come from?
 - The 80s architectural renaissance!
 - What do they provide?
 - How have they converged?
- Extract general structure and fundamental
- Reexamine traditional camps from new perspective (next week)



1/26/99 CS258 S99 L2 4

Administrivia

- Mix of HW, Exam, Project load
- HW 1 due date moved out to Fri 1/29
 - added 1.18
- Hands-on session with parallel machines in

1/26/99 CS258 S99 L2 5

Programming Model

- Conceptualization of the machine that programmer uses in coding applications**
 - How parts cooperate and coordinate their activities
 - Specifies communication and synchronization operations
- Multiprogramming**
 - no communication or synch. at program level
- Shared address space**
 - like bulletin board
- Message passing**
 - like letters or phone calls, explicit point to point
- Data parallel:**
 - more regimented, global actions on data
 - Implemented with shared address space or message passing

1/26/99 CS258 S99 L2 6

Shared Memory => Shared Addr. Space

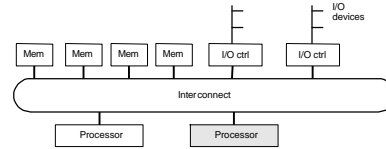
- Bottom-up engineering factors
- Programming concepts
- Why its attractive.

1/26/99

CS258 S99 L2

7

Adding Processing Capacity



- Memory capacity increased by adding modules
- I/O by controllers and devices
- **Add processors for processing!**
 - For higher-throughput multiprogramming, or parallel

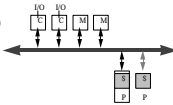
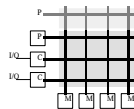
1/26/99

CS258 S99 L2

8

Historical Development

- "Mainframe" approach
 - Motivated by multiprogramming
 - Extends crossbar used for Mem and I/O
 - Processor cost-limited => crossbar
 - Bandwidth scales with p
 - High incremental cost
 - » use multistage instead
- "Minicomputer" approach
 - Almost all microprocessor systems have bus
 - Motivated by multiprogramming, TP
 - Used heavily for parallel computing
 - Called symmetric multiprocessor (SMP)
 - Latency larger than for uniprocessor
 - Bus is bandwidth bottleneck
 - » caching is key: coherence problem
 - Low incremental cost



1/26/99

CS258 S99 L2

9

Shared Physical Memory

- Any processor can directly reference any
- Any I/O controller - any memory
- Operating system can run on any processor, or
 - OS uses shared memory to coordinate
- Communication occurs implicitly as result of
- What about application processes?

1/26/99

CS258 S99 L2

10

Shared Virtual Address Space

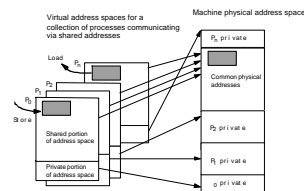
- **Process = address space plus thread of control**
- Virtual-to-physical mapping can be established so that processes shared portions of address
 - User-kernel or multiple processes
- Multiple threads of control on one address
 - Popular approach to structuring
 - Now standard application capability (ex: POSIX threads)
- Writes to shared address visible to other threads
 - Natural extension of uniprocessors model
 - conventional memory operations for communication
 - special atomic operations for synchronization
 - » also load/stores

1/26/99

CS258 S99 L2

11

Structured Shared Address Space



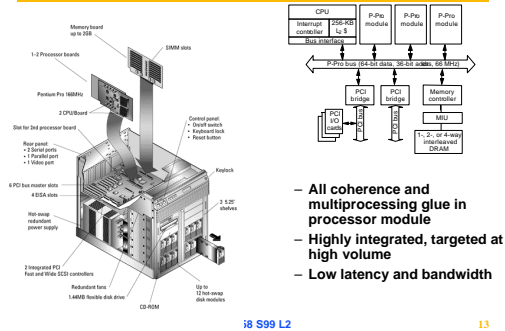
- Add hoc parallelism used in system code
- Most parallel applications have structured SAS
- Same program on each processor
 - shared variable X means the same thing to each thread

1/26/99

CS258 S99 L2

12

Engineering: Intel Pentium Pro Quad

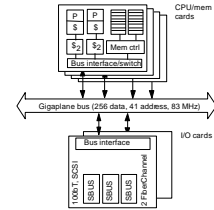


- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth

18 S99 L2

13

Engineering: SUN Enterprise



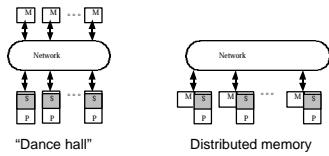
- Proc + mem card - I/O card
 - 16 cards of either type
 - All memory accessed over bus, so symmetric
 - Higher bandwidth, higher latency bus

1/26/99

CS258 S99 L2

14

Scaling Up



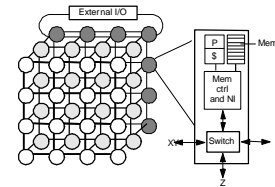
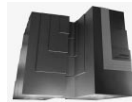
- Problem is interconnect: cost (crossbar) or bandwidth (bus)
- Dance-hall: bandwidth still scalable, but lower cost than crossbar
 - » latencies to memory uniform, but uniformly large
- Distributed memory or non-uniform memory access (NUMA)
 - » Construct shared address space out of simple message transactions across a general-purpose network (e.g. read-request, read-response)
- Caching shared (particularly nonlocal) data?

1/26/99

CS258 S99 L2

15

Engineering: Cray T3E



- Scale up to 1024 processors, 480MB/s links
- Memory controller generates request message for non-local references
- No hardware mechanism for coherence
 - » SGI Origin etc. provide this

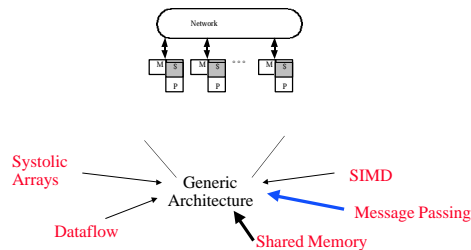
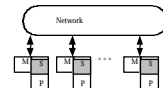
1/26/99

CS258 S99 L2

16

Message Passing Architectures

- Complete computer as building block, including I/O
 - Communication via explicit I/O operations
- Programming model
 - direct access only to private address space (local memory),
 - communication via explicit messages (send/receive)
- High-level block diagram
 - Communication integration?
 - » Mem, I/O, LAN, Cluster
 - Easier to build and scale than SAS
- Programming model more removed from basic hardware operations
 - Library or OS intervention



1/26/99

CS258 S99 L2

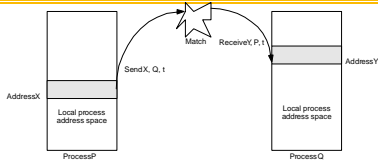
17

1/26/99

CS258 S99 L2

18

Message-Passing Abstraction



- Send specifies buffer to be transmitted and receiving process
- Recv specifies sending process and application storage to receive into
- Memory to memory copy, but need to name processes
- Optional tag on send and matching rule on receive
- User process names local data and entities in process/tag space too
- In simplest form, the send/recv match achieves pairwise synch event
 - » Other variants too
- Many overheads: copying, buffer management, protection

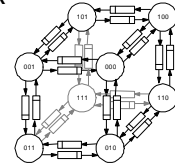
1/26/99

CS258 S99 L2

19

Evolution of Message-Passing Machines

- Early machines: FIFO on each link
 - HW close to prog. Model;
 - synchronous ops
 - topology central (hypercube algorithms)



1/26/99

CS258 S99 L2

20

Diminishing Role of Topology

- Shift to general links
 - DMA, enabling non-blocking ops
 - » Buffered by system at destination until recv
 - Store&forward routing
- Diminishing role of topology
 - Any-to-any pipelined routing
 - node-network interface dominates communication time

$$\begin{matrix} H \times (T_p + n/B) \\ \text{vs} \\ T_0 + H\Delta + n/B \end{matrix}$$

- Simplifies programming
- Allows richer design space
 - » grids vs hypercubes

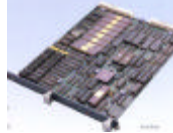
1/26/99

CS258 S99 L2

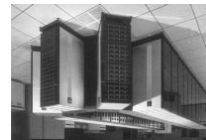
21



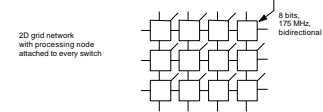
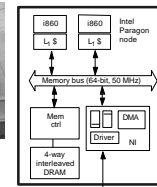
Intel IPSC/1 -> IPSC/2 -> IPSC/860



Example Intel Paragon



Sandia's Intel Paragon XPS-based Supercomputer



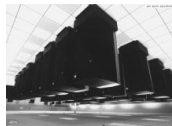
1/26/99

CS258 S99 L2

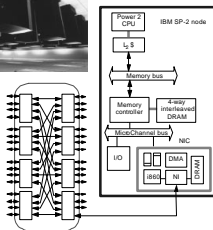
22

Building on the mainstream: IBM SP-2

- Made out of essentially complete RS6000 workstations
- Network interface integrated in I/O bus (bw limited by I/O bus)



General interaction network formed from 8-port switches



1/26/99

CS258 S99 L2

23

Berkeley NOW



- 100 Sun Ultra2 workstations
- Intelligent network interface
 - proc + mem
- Myrinet Network
 - 160 MB/s per link
 - 300 ns per hop

1/26/99

CS258 S99 L2

24

Toward Architectural Convergence

- **Evolution and role of software have blurred boundary**
 - Send/recv supported on SAS machines via buffers
 - Can construct global address space on MP (GA -> P | LA)
 - Page-based (or finer-grained) shared virtual memory
- **Hardware organization converging too**
 - Tighter NI integration even for MP (low-latency, high-bandwidth)
 - Hardware SAS passes messages
- **Even clusters of workstations/SMPs are parallel systems**
 - Emergence of fast system area networks (SAN)
- **Programming models distinct, but organizations**
 - Nodes connected by general network and communication assists
 - Implementations also converging, at least in high-end machines

1/26/99

CS258 S99 L2

25