ECE7660 Parallel Computer Architecture

Convergence of Parallel Architectures
Recap of Lecture 1

- Parallel Comp. Architecture driven by familiar technological and economic forces
  - application/platform cycle, but focused on the most demanding applications

- More attractive than ever because ‘best’ building block - *the microprocessor* - is also the fastest BB.

- History of microprocessor architecture is parallelism
  - 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 growth.
  - Mid 80s renaissance
Today

• Extension of “computer architecture” to support communication and cooperation
  – Instruction Set Architecture plus Communication Architecture

• Defines
  – Critical abstractions, boundaries, and primitives (interfaces)
  – Organizational structures that implement interfaces (hw or sw)

• Compilers, libraries and OS are important bridges today
Modern Layered Framework

- CAD
- Multiprogramming
- Shared address
- Message passing
- Data parallel
- Compilation or library
- Operating systems support
- Communication hardware
- Physical communication medium
- Database
- Scientific modeling
- Parallel applications
- Programming models
- Communication abstraction
- User/system boundary
- Hardware/software boundary
We will discuss …

• 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 issues

• Reexamine traditional camps from new perspective
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
Shared Memory => Shared Addr. Space

- Bottom-up engineering factors
- Programming concepts
- Why it’s attractive
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 programs
Historical Development

- “Mainframe” approach
  - Motivated by multiprogramming
  - Extends crossbar used for Mem and I/O
  - Processor cost-limited => crossbar affordable
  - Bandwidth scales with $p$
  - High incremental cost
    - use multistage instead

- “Minicomputer” approach
  - Almost all microprocessor systems have bus
  - Motivated by multiprogramming.
  - 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
Shared Physical Memory

• Any processor can directly reference any memory location
• Any I/O controller - any memory

• Operating system can run on any processor, or all.
  − OS uses shared memory to coordinate
• Communication occurs implicitly as result of loads and stores

• What about application processes?
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 space.
  - User-kernel or multiple processes
- **Multiple threads of control on one address space.**
  - Popular approach to structuring OS’s
  - 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
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
Engineering: Intel Pentium Pro Quad

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

Dataflow

Generic Architecture

SIMD

Message Passing

Shared Memory
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
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**
- **In simplest form, the send/recv match achieves pairwise synch event**
  - Other variants too
- **Many overheads**: copying, buffer management, protection
Evolution of Message-Passing Machines

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

CalTech Cosmic Cube (Seitz, CACM Jan 95)
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

\[
H \times (T_0 + \frac{n}{B})
\]

vs

\[
T_0 + H\Delta + \frac{n}{B}
\]

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

Intel iPSC/1 -> iPSC/2 -> iPSC/860
Example Intel Paragon

Sandia’s Intel Paragon XP/S-based Supercomputer

2D grid network with processing node attached to every switch

8 bits, 175 MHz, bidirectional
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)
Berkeley NOW

- 100 Sun Ultra2 workstations
- Intelligent network interface
  - proc + mem
- Myrinet Network
  - 160 MB/s per link
  - 300 ns per hop
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 converging
  – Nodes connected by general network and communication assists
  – Implementations also converging, at least in high-end machines