Module 6: "Fundamentals of Parallel Computers"

Lecture 10: "Communication Architecture"

Fundamentals of Parallel Computers

- Agenda
- Communication architecture
- Layered architecture
- Shared address
- Message passing
- Convergence
- Data parallel arch.

[From Chapter 1 of Culler, Singh, Gupta]

Previous Next

Module 6: "Fundamentals of Parallel Computers"
Lecture 10: "Communication Architecture"

Agenda

- Convergence of parallel architectures
- Fundamental design issues
- ILP vs. TLP

Communication architecture

- Historically, parallel architectures are tied to programming models
  - Diverse designs made it impossible to write portable parallel software
  - But the driving force was the same: need for fast processing
- Today parallel architecture is seen as an extension of microprocessor architecture with a communication architecture
  - Defines the basic communication and synchronization operations and provides hw/sw implementation of those

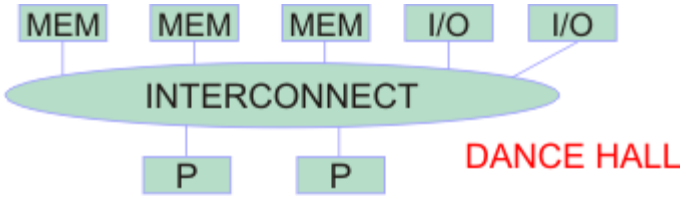
Layered architecture

- A parallel architecture can be divided into several layers
  - Parallel applications
  - Programming models: shared address, message passing, multiprogramming, data parallel, dataflow etc
  - Compiler + libraries
  - Operating systems support
  - Communication hardware
  - Physical communication medium
- Communication architecture = user/system interface + hw implementation (roughly defined by the last four layers)
  - Compiler and OS provide the user interface to communicate between and synchronize threads

Previous Next

Shared address

- Communication takes place through a logically shared portion of memory
  - User interface is normal load/store instructions
  - Load/store instructions generate virtual addresses
  - The VAs are translated to PAs by TLB or page table
  - The memory controller then decides where to find this PA
  - Actual communication is hidden from the programmer
- The general communication hw consists of multiple processors connected over some medium so that they can talk to memory banks and I/O devices
  - The architecture of the interconnect may vary depending on projected cost and target performance
- Communication medium



  - Interconnect could be a crossbar switch so that any processor can talk to any memory bank in one “hop” (provides latency and bandwidth advantages)
  - Scaling a crossbar becomes a problem: cost is proportional to square of the size
  - Instead, could use a scalable switch-based network; latency increases and bandwidth decreases because now multiple processors contend for switch ports
- Communication medium
  - From mid 80s shared bus became popular leading to the design of SMPs
  - Pentium Pro Quad was the first commodity SMP
  - Sun Enterprise server provided a highly pipelined wide shared bus for scalability reasons; it also distributed the memory to each processor, but there was no local bus on the boards i.e. the memory was still “symmetric” (must use the shared bus)
  - NUMA or DSM architectures provide a better solution to the scalability problem; the symmetric view is replaced by local and remote memory and each node (containing processor(s) with caches, memory controller and router) gets connected via a scalable network (mesh, ring etc.); Examples include Cray/SGI T3E, SGI Origin 2000, Alpha GS320, Alpha/HP GS1280 etc.

Previous Next

Message passing

- Very popular for large-scale computing
- The system architecture looks exactly same as DSM, but there is no shared memory
- The user interface is via send/receive calls to the message layer
- The message layer is integrated to the I/O system instead of the memory system
- Send specifies a local data buffer that needs to be transmitted; send also specifies a tag
- A matching receive at dest. node with the same tag reads in the data from kernel space buffer to user memory
- Effectively, provides a memory-to-memory copy
- Actual implementation of message layer
  - Initially it was very topology dependent
  - A node could talk only to its neighbors through FIFO buffers
  - These buffers were small in size and therefore while sending a message send would occasionally block waiting for the receive to start reading the buffer (synchronous message passing)
  - Soon the FIFO buffers got replaced by DMA (direct memory access) transfers so that a send can initiate a transfer from memory to I/O buffers and finish immediately (DMA happens in background); same applies to the receiving end also
  - The parallel algorithms were designed specifically for certain topologies: a big problem
- To improve usability of machines, the message layer started providing support for arbitrary source and destination (not just nearest neighbors)
  - Essentially involved storing a message in intermediate “hops” and forwarding it to the next node on the route
  - Later this store-and-forward routing got moved to hardware where a switch could handle all the routing activities
  - Further improved to do pipelined wormhole routing so that the time taken to traverse the intermediate hops became small compared to the time it takes to push the message from processor to network (limited by node-to-network bandwidth)
  - Examples include IBM SP2, Intel Paragon
  - Each node of Paragon had two i860 processors, one of which was dedicated to servicing the network (send/recv. etc.)

Previous Next

Convergence

- Shared address and message passing are two distinct programming models, but the architectures look very similar
  - Both have a communication assist or network interface to initiate messages or transactions
  - In shared memory this assist is integrated with the memory controller
  - In message passing this assist normally used to be integrated with the I/O, but the trend is changing
  - There are message passing machines where the assist sits on the memory bus or machines where DMA over network is supported (direct transfer from source memory to destination memory)
  - Finally, it is possible to emulate send/recv. on shared memory through shared buffers and flags
  - Possible to emulate a shared virtual mem. on message passing machines through modified page fault handlers

Data parallel arch.

- Array of processing elements (PEs)
- Each PE operates on a data element within a large matrix
- The operation is normally specified by a control processor
- Essentially, single-instruction-multiple-data (SIMD) architectures
- So the parallelism is exposed at the data level
- Processor arrays were outplayed by vector processors in mid-70s
  - Vector processors provide a more general framework to operate on large matrices in a controlled fashion
  - No need to design a specialized processor array in a certain topology
- Advances in VLSI circuits in mid-80s led to design of large arrays of single-bit PEs
- Also, arbitrary communication (rather than just nearest neighbor) was made possible
- Gradually, this architecture evolved into SPMD (single-program-multiple-data)
  - All processors execute the same copy of a program in a more controlled fashion
  - But parallelism is expressed by partitioning the data
  - Essentially, the same as the way shared memory or message passing machines are used for running parallel applications

Previous Next