Message Passing Computers

Mark Greenstreet

CpSc 418 – Jan. 27, 2017

Outline:

- Network Topologies
- Performance Considerations
- Examples

Objectives
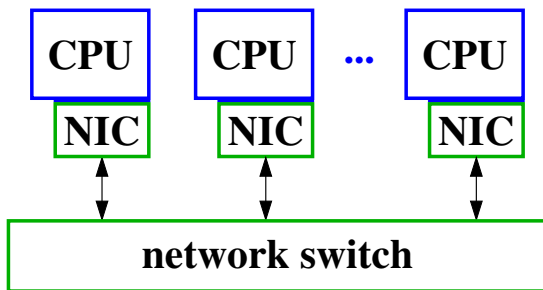
- Familiar with typical network topologies:
  rings, meshes, crossbars, tori, hypercubes, trees, fat-trees.
- Understand implications for programming
  - bandwidth bottlenecks
  - latency considerations
  - location matters
  - heterogeneous computers.

Message Passing Computers

- Multiple CPU's
- Communication through a network:
  - ▶ Commodity networks for small clusters.
  - ▶ Special high-performance networks for super-computers
- Programming model:
  - ▶ Explicit message passing between processes (like Erlang)
  - ▶ No shared memory or variables.

Some simple message-passing clusters

- 25 linux workstations (e.g. lin01 . . . lin25.ugrad.cs.ubc.ca) and standard network routers.
  - ▶ A good platform for learning to use a message-passing cluster.
  - ▶ But, we'll figure out that network bandwidth and latency are key bottlenecks.
- A "blade" based cluster, for example:
  - ▶ 16 "blades" each with 4 6-core CPU chips, and 32G of DRAM.
  - ▶ An "infiniband" or similar router for about 10-100 times the bandwidth of typical ethernet.
  - ▶ The price tag is ∼$300K.
    - ★ Great if you need the compute power.
    - ★ But, we won't be using one in this class.

The Sunway TaihuLight

- The world's fastest (Linpack) super-computer (as of June 2016)
- 40,960 multicore CPUs
  - 256 cores per CPU chip.
  - 1.45GHz clock frequency, 8 flops/core/cycle.
- Total of 10,485,760 cores
- LINPACK performance: 93 PFlops
- Power consumption 15MW (computer) + cooling (unspecified)
- Tree-like
  - Five levels of hierarchy.
  - Each level has a high-bandwidth switch.
  - Some levels (all?) are fully-connected for that level.
- Programming model: A version linux with MPI tuned for this machine.
- For more information, see
  Report on the Sunway TaihuLight System, J. Dongarra, June 2016.

The Westgrid Clusters

- Clusters at various Western Canadian Universities (including UBC).
- Up to 9600 cores.
- Available for research use.

Network Topologies

- Network topologies are to the message-passing community what cache-coherence protocols are to the shared-memory people:
  - **Lots** of papers have been published.
  - Machine designers are always looking for better networks.
  - Network topology has a strong impact on performance, the programming model, and the cost of building the machine.
- A message-passing machine may have multiple networks:
  - A general purpose network for sending messages between machines.
  - Dedicated networks for reduce, scan, and synchronization:
    - ★ The reduce and scan networks can include ALUs (integer and/or floating point) to perform common operations such as sums, max, product, all, any, etc. in the networking hardware.
    - ★ A synchronization network only needs to carry a few bits and can be designed to minimize latency.
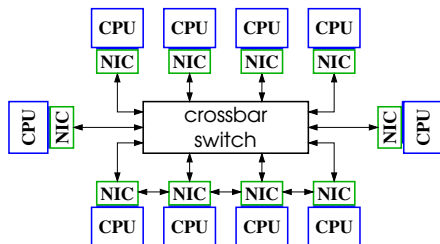
Ring-Networks
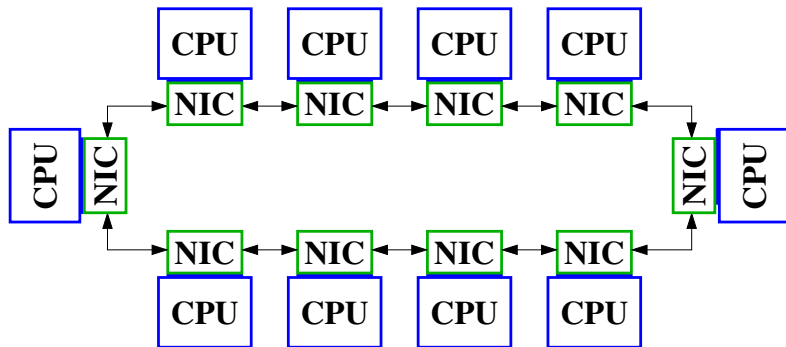
- Advantages: simple.
- Disadvantages:
    - Worst-case latency grows as $O(P)$ where P is the number of processors.
    - Easily congested – limited bandwidth.

Star Networks

- Advantages:
  - ► Low-latency – single hop between any two nodes
  - ► High-bandwidth – no contention for connections with different sources and destinations.
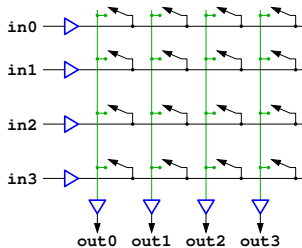- Disadvantages:
  - ► Amount of routing hardware grows as $O(P^2)$.
  - ► Requires lots of wires, to and from switch –
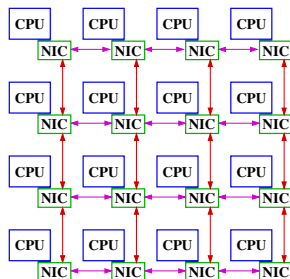   Imagine trying to build a switch that connects to 1000 nodes!
- Summary
  - ► Surprisingly practical for 10-50 ports.
  - ► Hierarchies of cross-bars are often used for larger networks.
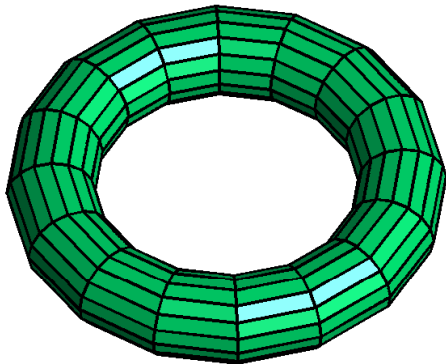
A crossbar switch

Meshes

- Advantages:
  - Easy to implement: chips and circuit boards are effectively two-dimensional.
  - Cross-section bandwidth grow with number of processors – more specifically, bandwidth grows as $\sqrt{P}$.
- Disadvantages:
  - Worst-case latency grows as $\sqrt{P}$.
  - Edges of mesh are "special cases."

Tori

- Advantages:
  - Has the good features of a mesh, and
  - No special cases at the edges.
- Disadvantages:
  - Worst-case latency grows as $\sqrt{P}$.
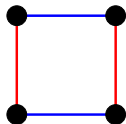
Hypercubes

A 0−dimensional (1 node), radix−2 hypercube

●

Hypercubes

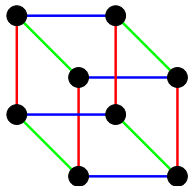A 1−dimensional (2 node), radix−2 hypercube

Hypercubes

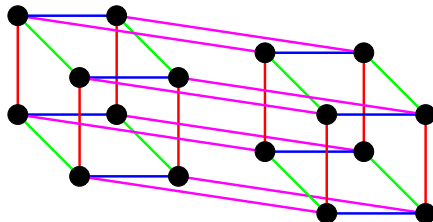A 2−dimensional (4 node), radix−2 hypercube

Hypercubes

A 3−dimensional (8 node), radix−2 hypercube

Hypercubes

A 4−dimensional (16 node), radix−2 hypercube

Hypercubes

A 5−dimensional (32 node), radix−2 hypercube
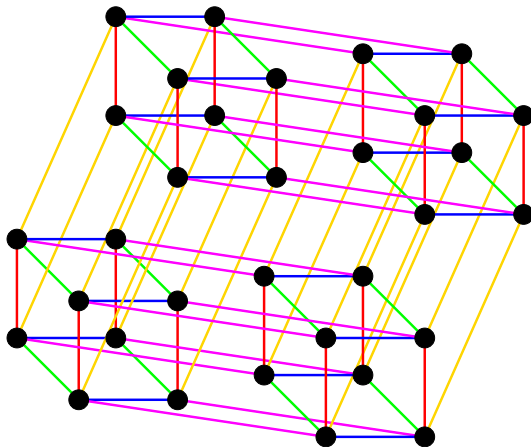
Hypercubes

A 5−dimensional (32 node), radix−2 hypercube
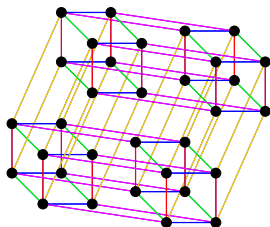
- Advantages
  - Small diameter (log *N*)
  - Lots of bandwidth
  - Easy to partition.
  - Simple model for algorithm design.
- Disadvantages
  - Needs to be squeezed into a three-dimensional universe.
  - Lots of long wires to connect nodes.
  - Design of a node depends on the size of the machine.

Dimension Routing

```
% Send a message, msg, from node src to node dst
for i = 1:d                          % d is dimension of the hypercube
  if(bit(i, src) != bit(i, dst))     % if different for dimension i
    send(msg, link[i]);              % then send msg to our i-neighbour
```

Trees
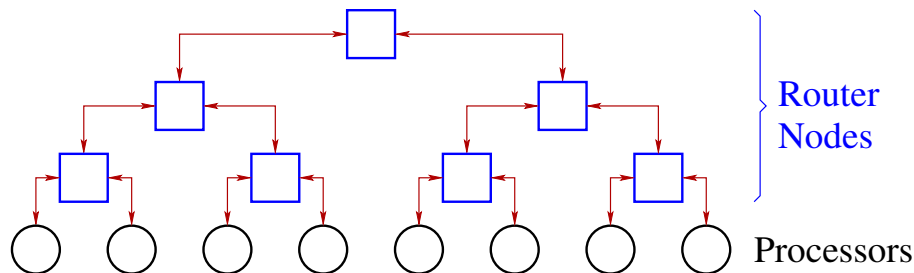
- Simple network: number of routing nodes = number of processors − 1.
- Wiring: $O(\log N)$ extra height ($O(N \log N)$) extra area.
  - Wiring: $O(\sqrt{N} \log N)$ extra area for H-tree.
- Low-latency: $O(\log N) +$ wire delay.
- Low-bandwidth: bottleneck at root.

Fat-Trees
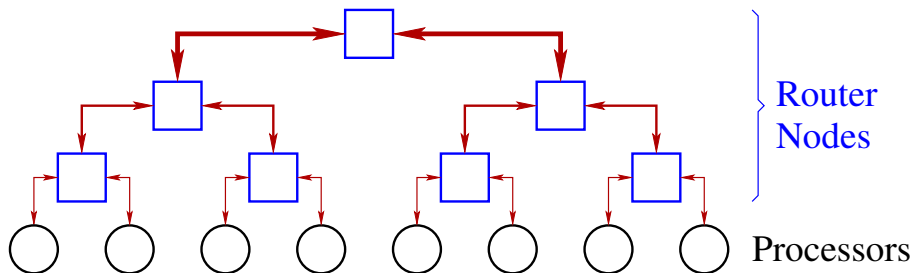
- Use M^α parallel links to connect subtrees with *M* leaves.
- $0 \leq \alpha \leq 1$
  - $\alpha = 0$: simple tree
  - $\alpha = 1$: strange crossbar
- Fat-trees are "universal"
  - For $\frac{2}{3} < \alpha < 1$ a fat-tree interconnect with volume *V* can simulate any interconnect that occupies the same volume with a time overhead that is poly-log factor of *N*.

Performance Considerations

- Bandwidth
  - How many bytes per-second can we send between two processors?
    - May depend on which two processors: neighbours may have faster links than spanning the whole machine.
  - Bisection bandwidth: find the worst way to divide the processors into to sets of $P/2$ processors each.
    - How many bytes per-second can we send between the two partitions?
    - If we divide this by the number of processors, we typically get a much smaller value that the peak between two processors.
- Latency
  - How long does it take to send a message from one processor to another?
    - Typically matters the most for short messages.
    - Round-trip time is often a good way to measure latency.
- Cost
  - How expensive is the interconnect – it may dominate the total machine cost.
    - Cost of the network interface hardware.
    - Cost of the cables.

Real-life networks

- InfiniBand is becoming increasingly prevalent
- Peak bandwidths $\geq$ 6GBytes/sec.
  - achieved bandwidths of 2–3GB/s.
- Support for RDMA and "one-sided" communication
  - CPU A can read or write a block of memory residing with CPU B.
- Often, networks include trees for synchronization (e.g. barriers), and common reduce and scan operations.
- The MPI (message-passing interface) evolves to track the capabilities of the hardware.

Bandwidth Matters
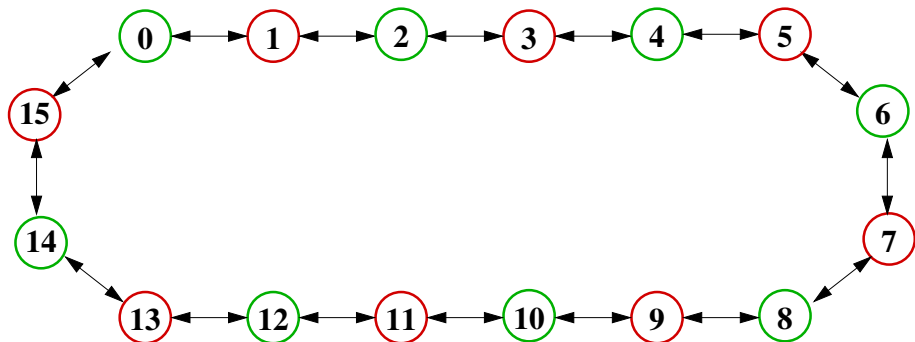
- Assume each link has a bandwidth in each direction of 1Gbyte/sec.
- Each node, *i*, sends an 8Kbyte message to node $(i + 1) \bmod P$, where *P* is the number of processors?
- How long does this take?
- What if each node, *i*, sends an 8Kbyte message to node $(i + P/2) \bmod P$?

What this means for programmers

- Location matters.
  - ▶ The meaning of location depends on the machine.
  - ▶ Getting a good programming model is hard.
  - ▶ Challenges of heterogeneous machines.
- What it means for different kinds of computers
  - ▶ Supercomputers
  - ▶ Clouds
  - ▶ PCs of the future(?)

Summary

- Message passing machines have an architecture that corresponds to the message-passing programming paradigm.
- Message passing machines can range from
  - Clusters of PC's with a commodity switch.
  - Clouds: lots of computers with a general purpose network.
  - Super-computers: lots of compute nodes tightly connected with high-performance interconnect.
- Many network topologies have been proposed:
  - Performance and cost are often dominated by network bandwidth and latency.
  - The network can be more expensive than the CPUs.
  - Peta-flops or other instruction counting measures are an indirect measure of performance.
- Implications for programmers
  - Location matters
  - Communication costs of algorithms is very important
  - Heterogeneous computing is likely in your future.

Preview

January 30: Parallel Performance: Speed-up	
Reading:	Pacheco, Chapter 2, Section 2.6.
Homework:	HW 2 earlybird (11:59pm). HW 3 goes out.
February 1: Parallel Performance: Overheads	
Homework:	HW 2 due (11:59pm).
February 3: Parallel Performance: Models	
Mini Assignments	Mini 4 due (10am)
February 6: Parallel Performance: Wrap Up	
January 8–February 15: Parallel Sorting	
Homework (Feb. 15):	HW 3 earlybird (11:59pm), HW 4 goes out.
February 17: Map-Reduce	
Homework:	HW 3 due (11:59pm).
February 27: TBD	
March 1: Midterm	

- There will be readings assigned from
  Programming Massively Parallel Processors starting after the
  midterm.

- Make sure you have a copy. Note: this year, we'll make sure the
 course works with either the 2nd or 3rd edition.

Review

- Consider a machine with 4096 processors.
- What is the maximum latency for sending a message between two processors (measured in network hops) if the network is
  - A ring?
  - A crossbar?
  - A 2-D mesh?
  - A 3-D mesh?
  - A hypercube?
  - A binary tree?
 - A radix-4 tree?

Supplementary Material

- Message-passing origami: how to fold a mesh into a torus.
- How big is a hypercube: it's all about the wires.

From a mesh to a torus (1/2)
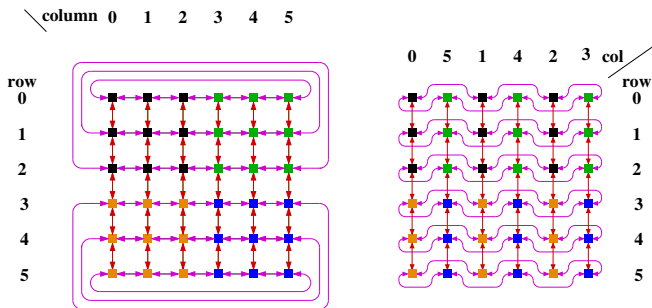
- Fold left-to-right, and make connections where the left and right edges meet.
- Now, we've got a cylinder.
- Note that there are no "long" horizontal wires: the longest wires jump across one processor.

From a mesh to a torus (2/2)
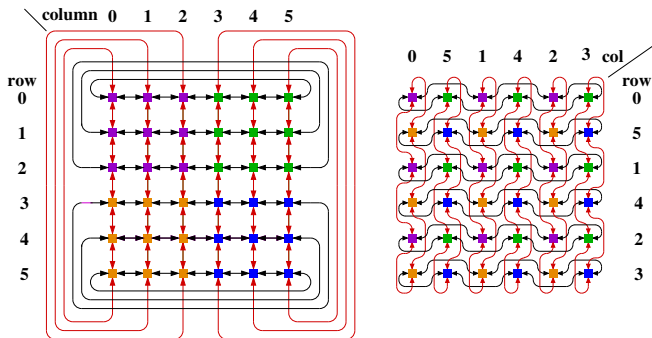
- Fold top-to-bottom, and make connections where the top and bottom edges meet.
- Now, we've got a torus.
- Again there are no "long" wires.

How big is a hypercube?

- Consider a hypercube with $N = 2^d$ nodes.
- Assume each link can transfer one message in each direction in one time unit. The analysis here easily generalizes for links of higher or lower bandwidths.
- Let each node send a message to each of the other nodes.
- Using dimension routing,
  - Each node will send $N/2$ messages for each of the d dimensions.
  - This takes time $N/2$.
  - As soon as one batch of messages finishes the dimension-0 route, that batch can continue with the dimension-1 route, and the next batch can start the dimension 0 route.
  - So, we can route with a throughput of $\begin{pmatrix} N \\ 2 \end{pmatrix}$ messages per $N/2$ time.

How big is a hypercube?

- Consider a hypercube with $N = 2^d$ nodes.
- Assume each link can transfer one message in each direction in one time unit. The analysis here easily generalizes for links of higher or lower bandwidths.
- Let each node send a message to each of the other nodes.
- Using dimension routing,

  we can route with a throughput of $\begin{pmatrix} N \\ 2 \end{pmatrix}$ messages per $N/2$ time.

- Consider any plane such that $N/2$ nodes are on each side of the plane.

  ▸ $\frac{1}{2} \begin{pmatrix} N \\ 2 \end{pmatrix}$ messages must cross this plane in $N/2$ time.

  ▸ This means that at least $N - 1$ links must cross the plane.

  ▸ The plane has area $O(N)$.

How big is a hypercube?

- Consider a hypercube with $N = 2^d$ nodes.
- Assume each link can transfer one message in each direction in one time unit. The analysis here easily generalizes for links of higher or lower bandwidths.
- Let each node send a message to each of the other nodes.
- Using dimension routing,

  we can route with a throughput of $\begin{pmatrix} N \\ 2 \end{pmatrix}$ messages per $N/2$ time.
- Consider any plane such that $N/2$ nodes are on each side of the plane.
  - The plane has area $O(N)$.
- Because the argument applies for *any* plane, we conclude that the hypercube has diameter $O(\sqrt{N})$ and thus volume $O(N^{\frac{3}{2}})$.
- Asymptotically, the hypercube is all wire.