Message-Passing Parallel Computers

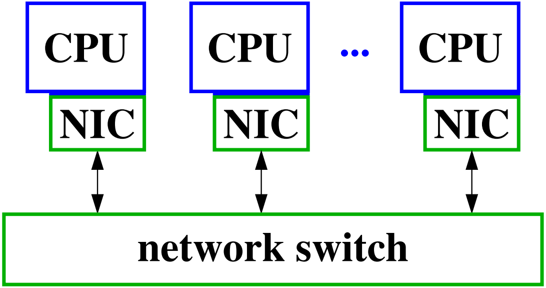
Mark Greenstreet

CpSc 448B – Oct. 6, 2011

Outline:

- Message Passing Computers
- Examples

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 exampe:
  - 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 $\sim$\$300K.
    - Great if you need the compute power.
    - But, we won't be using one in this class.

The K Machine

- The world's second fastest super-computer
- 88,128 8-core SPARC processors.
- LINPACK performance: 10.51 PFlops
- Power consumption 9.89 MW
  - Roughly 10,000 homes.
  - Operating costs estimated at $10M/year.
  - But, it's one of the best supercomputers for PFlops/Watt
- Interconnect:
  - "6D" torus network (called "Tofu").
  - 10 Gbytes/sec, bi-directional, for each link.
  - Hardware support for reduce operations and synchronization.
- Programming model:
  - A version of MPI tuned for this machine.
  - Supports topology-aware programs.
  - The interconnect is designed to make it easy to partition the machine so different jobs can run on different partitions.

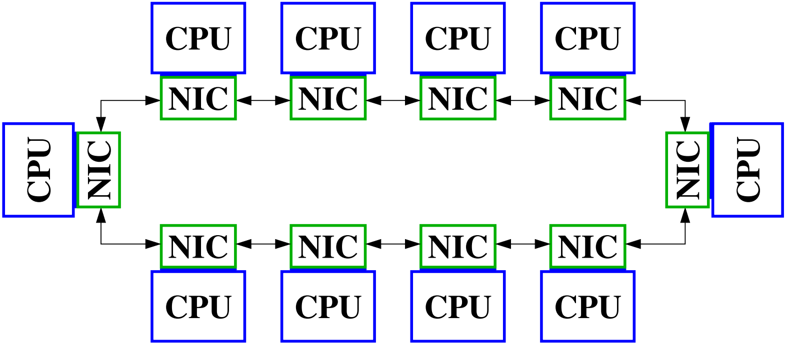
The Westgrid Clusters

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

Network Topologies

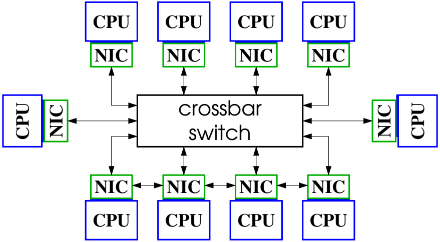
- Network topologies are to the message-passing community what cache-coherence protocols are to the shared-memory people:
  - **Lots** of papers have been published.
  - Machine designers are always looking for better networks.
  - Network topology has a strong impact on performance, the programming 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



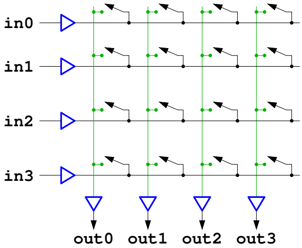
- Advantages: simple.
- Disadvantages:
  - Worst-case latency grows as $O(P)$ where P is the number of processors.
  - Easily congested – limited bandwidth.

Star Networks

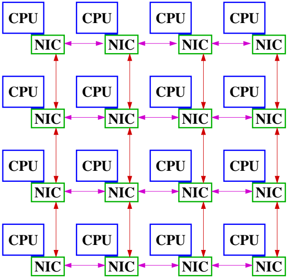


- Advantages:
  - Low-latency – single hop betweeen any two nodes
  - High-bandwidth – no contention for connections with different sources and destinations.
- Disadvantages:
  - Amount of routing hardware grows as $O(P^2)$.
  - Requires lots of wires, to and from switch –
    Imagaine trying to build 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

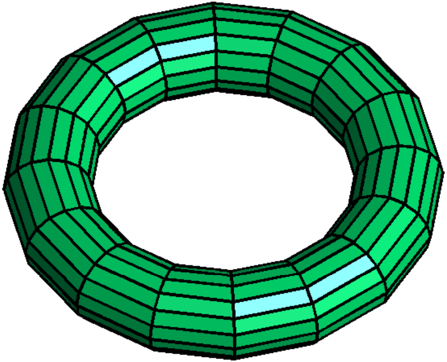


Meshes



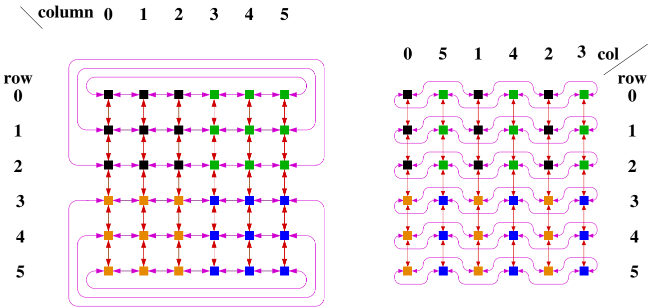
- Advantages:
  - Easy to implement: chips and circuit boards are effectively two-dimensional.
  - Cross-section bandwidth grow with number of processors – more specifically, bandwidth grows as $\sqrt{P}$.
- Disadvantages:
  - Worst-case latency grows as $\sqrt{P}$.
  - Edges of mesh are “special cases.”

Tori



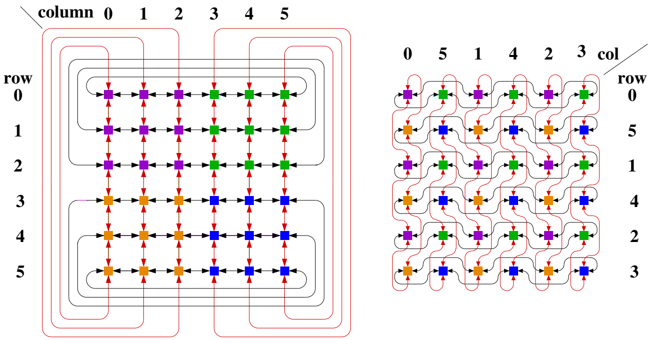
- Advantages:
  - Has the good features of a mesh, and
  - No special cases at the edges.
- Disadvantages:
  - Worst-case latency grows as $\sqrt{P}$.

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.

Hypercubes

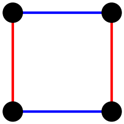
A 0–dimensional (1 node), radix–2 hypercube

Hypercubes

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

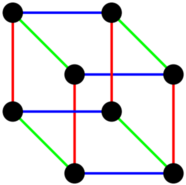
Hypercubes

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



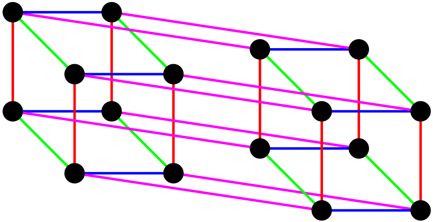
Hypercubes

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



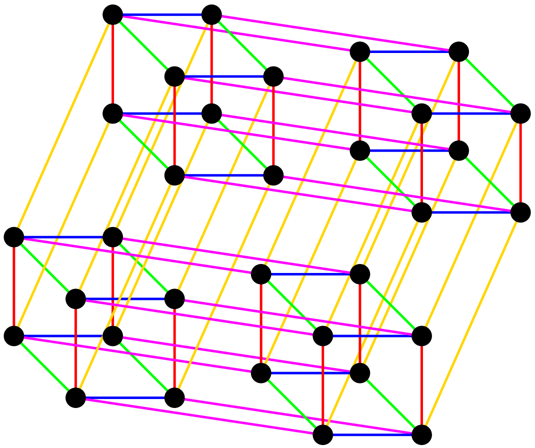
Hypercubes

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



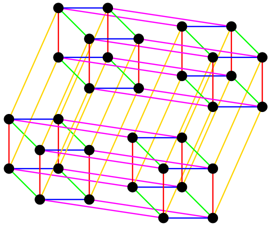
Hypercubes

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
```

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 $\binom{N}{2}$ 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 $\binom{N}{2}$ messages per $N/2$ time.
- Consider any plane such that $N/2$ nodes are on each side of the plane.
  - $\frac{1}{2}\binom{N}{2}$ 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 $\binom{N}{2}$ 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.

Real-life networks

- 3D Tori.
- Trees and fat-trees.
- 5 and 6D tori.

What this means for programmers

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