

CS395T: Introduction to Scientific and Technical Computing

Instructors:

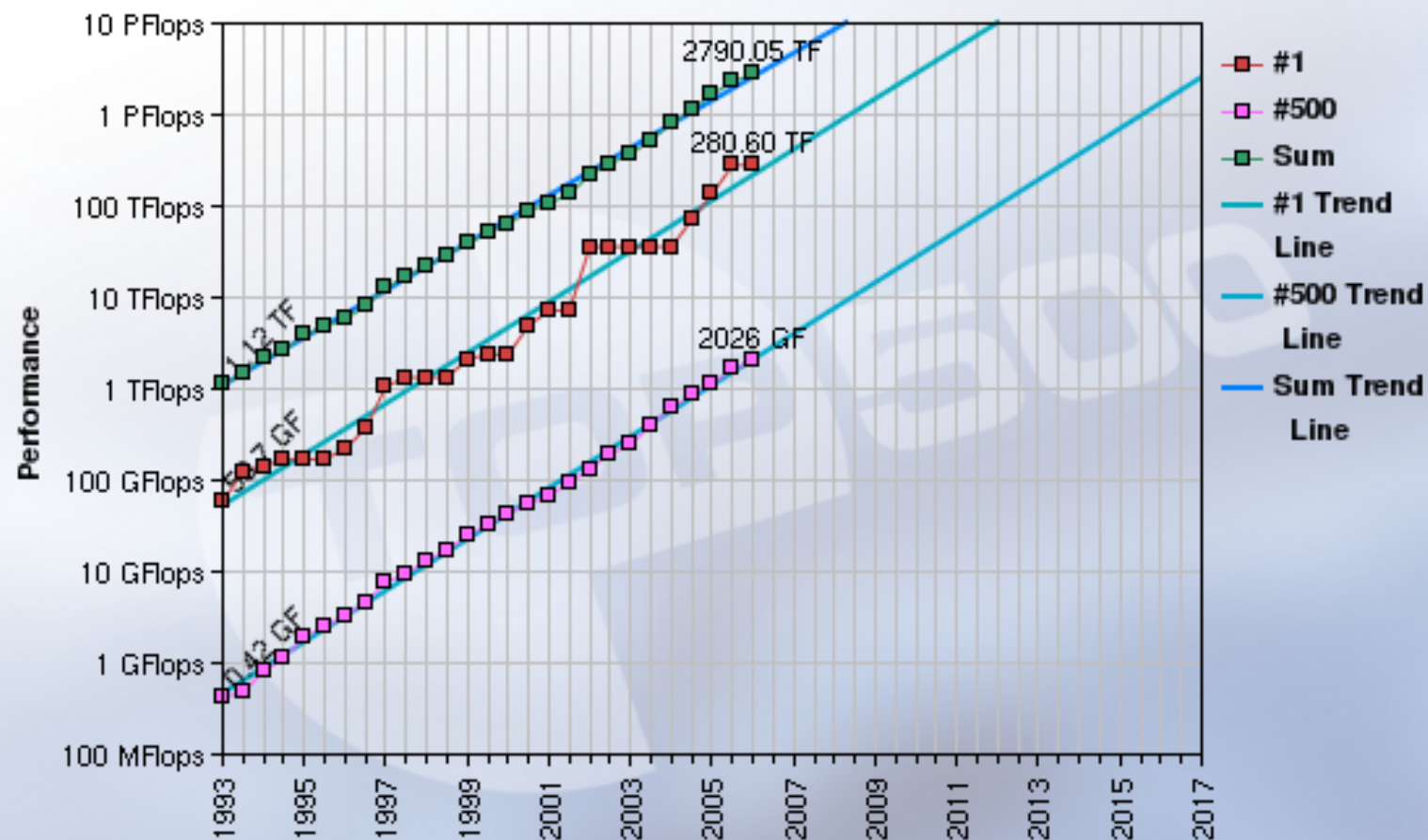
Dr. Karl W. Schulz, Research Associate, TACC

Dr. Victor Eijkhout, Research Scientist, TACC

Outline

- Parallel Computer Architectures
- Components of a Cluster
- Basic Anatomy of a Server/Desktop/Laptop/Cluster-node
 - Memory Hierarchy
 - Structure, Size, Speed, Line Size, Associativity
 - Latencies and Bandwidths
 - Intel vs. AMD platforms
 - Memory Architecture
 - Point-to-Point Communications between platforms.
- Node Communication in Clusters
 - Interconnects

Projected Performance Development



28/06/2006

<http://www.top500.org/>

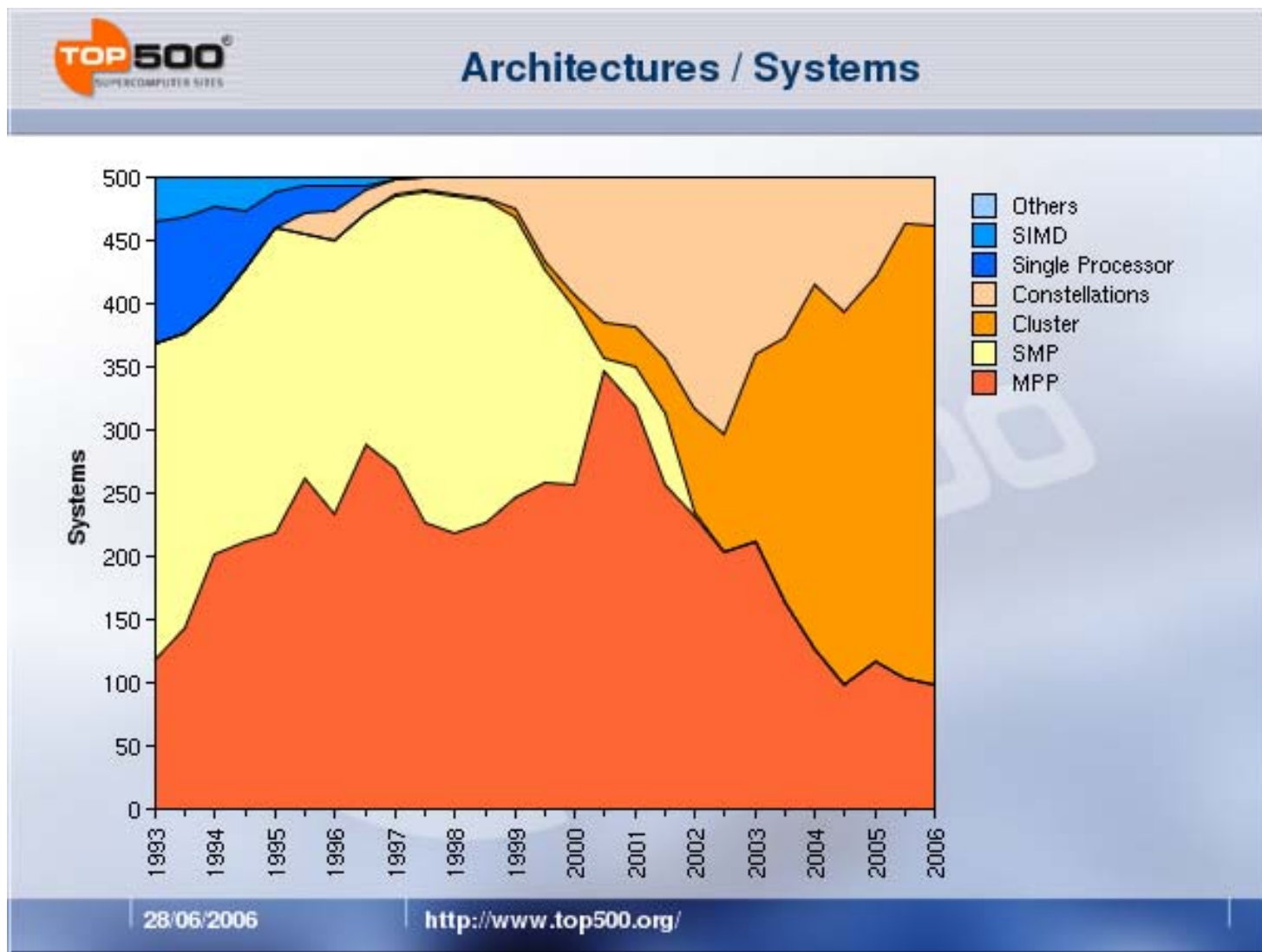
The Top500 List

- High-Performance LINPACK
 - Dense linear system solve with LU factorization
 - $\frac{2}{3} n^3 + O(n^2)$
 - Measure: MFlops
 - <http://www.netlib.org/benchmark/hpl/>
- The problem size can be chosen
 - fiddle with it until you find n to get the best performance
 - report n , maximum performance, and theoretical peak performance
- <http://www.top500.org/>

Parallel Computers Architectures

- **Parallel computing** means using multiple processors, possibly comprising multiple computers
- Until recently, Flynn's (1966) taxonomy was commonly used to classify parallel computers into one of four types:
 - (SISD) Single instruction, single data
 - Your desktop (unless you have a newer multiprocessor one)
 - (SIMD) Single instruction, multiple data:
 - Thinking machines CM-2
 - Cray 1, and other vector machines (there's some controversy here)
 - Parts of modern GPUs
 - (MISD) Multiple instruction, single data
 - Special purpose machines
 - No commercial, general purpose machines
 - (MIMD) Multiple instruction, multiple data
 - Nearly all of today's parallel machines

Top500 by Overall Architecture



SIMD

- Based on regularity of computation: all processors often doing the same operation
- Big advantage: processor do not need separate ALU
- ==> lots of small processors packed together
- Ex: Goodyear MPP: 64k processors in 1983
- Use masks to let processors differentiate

Vector Machines

- Based on a single processor with:
 - Segmented (pipeline) functional units
 - Needs sequence of the same operation
- Dominated early parallel market
 - overtaken in the 90s by clusters, et al.
- Making a comeback (sort of)
 - clusters/constellations of vector machines:
 - Earth Simulator (NEC SX6) and Cray X1/X1E
 - modern micros have explicit vector instructions
 - MMX, SSE, etc.
 - GPUs
 - (But ordinary CPUs are also pipelined)

Pipeline

- Assembly line model (body on frame, attach wheels, doors, handles on doors)
- Floating point multiply: exponent align, multiply, exponent normalize
- Separate hardware for each stage: pipeline processor
- Complexity model: asymptotic rate, $n_{1/2}$
- Multi-vectors, parallel pipes (demands on code)
- Is like SIMD

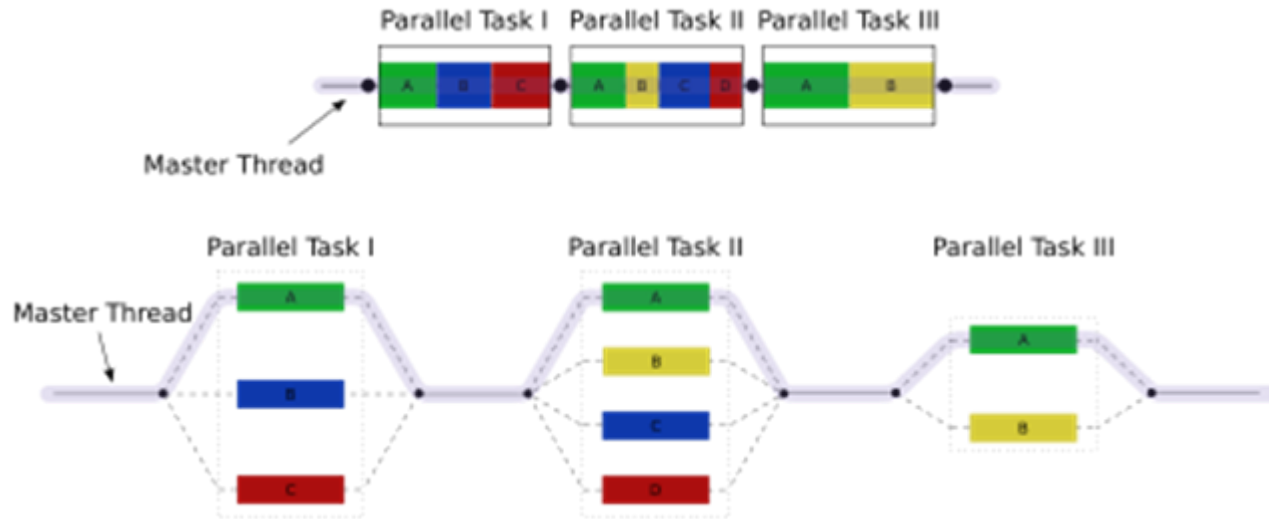
Parallel Computer Architectures

- Top500 List now dominated by MPPs, Constellations and Clusters
- The MIMD model “won”.

Parallel programming models

- Loop-based parallelism: OpenMP
- Message passing: MPI
- Compiler & OS-based approaches (parallel languages)

OpenMP



- Loop-based parallelism: iterations spread over threads.
- Shared memory.
- Various issues: critical regions, binding, thread overhead

OpenMP programming

- “pragma”-based: directives to the compiler

```
#pragma omp parallel default(none) \  
    shared(n,x,y) private(i)  
{  
    #pragma omp for  
    for (i=0; i<n; i++)  
        x[i] += y[i];  
} /*-- End of parallel region --*/
```

clauses

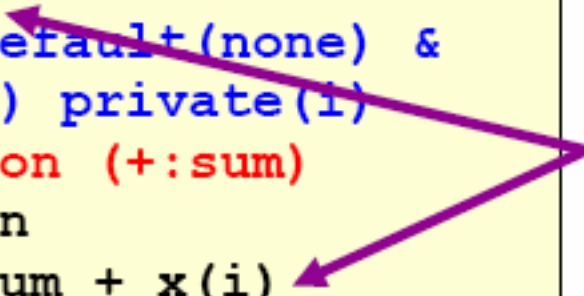
```
!$omp parallel default(none) &  
!$omp shared(n,x,y) private(i)  
!$omp do  
    do i = 1, n  
        x(i) = x(i) + y(i)  
    end do  
!$omp end do  
!$omp end parallel
```

OpenMP programming

- Handling of private and shared data

```
sum = 0.0
!$omp parallel default(none) &
!$omp shared(n,x) private(i)
!$omp do reduction (+:sum)
  do i = 1, n
    sum = sum + x(i)
  end do
!$omp end do
!$omp end parallel
print *,sum
```

Variable SUM is a shared variable



MPI: message passing

- Message Passing Interface: library for explicit communication
- Point-to-point and collective communication
- Looks harder than it is

```
if(myid == 0)
{
    printf("WE have %d processors\n", numprocs);
    for(i=1;i<numprocs;i++)
    {
        sprintf(buff, "Hello %d", i);
        MPI_Send(buff, 128, MPI_CHAR, i, 0, MPI_COMM_WORLD);
    }
    for(i=1;i<numprocs;i++)
    {
        MPI_Recv(buff, 128, MPI_CHAR, i, 0, MPI_COMM_WORLD, &stat);
        printf("%s\n", buff);
    }
}
else
{
    MPI_Recv(buff, 128, MPI_CHAR, 0, 0, MPI_COMM_WORLD, &stat);
    sprintf(idstr, "Processor %d ", myid);
    strcat(buff, idstr);
    strcat(buff, "reporting for duty\n");
    MPI_Send(buff, 128, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
}
```

Some theory

-before we get into the hardware
- Optimally, P processes give $T_P = T_1/P$
- Superlinear speedup not possible in theory, sometimes happens in practice.
- Speedup $S_P = T_1/T_P$, is P at best
- Perfect speedup in “embarrassingly parallel applications”
- Less than optimal: overhead & sequential parts

Some theory

-before we get into the hardware
- Optimally, P processes give $T_P = T_1/P$
- Superlinear speedup not possible in theory, sometimes happens in practice.
- Speedup $S_P = T_1/T_P$, is P at best
- Efficiency $E_P = S_P/P$
- Scalability: efficiency bounded below
- Weak/strong scalability: total problem size constant / increasing with P

Amdahl's Law

- Some parts of a code are not parallelizable
- \Rightarrow they ultimately become a bottleneck
- For instance, if 5% is sequential, you can not get a speedup over 20, no matter P .

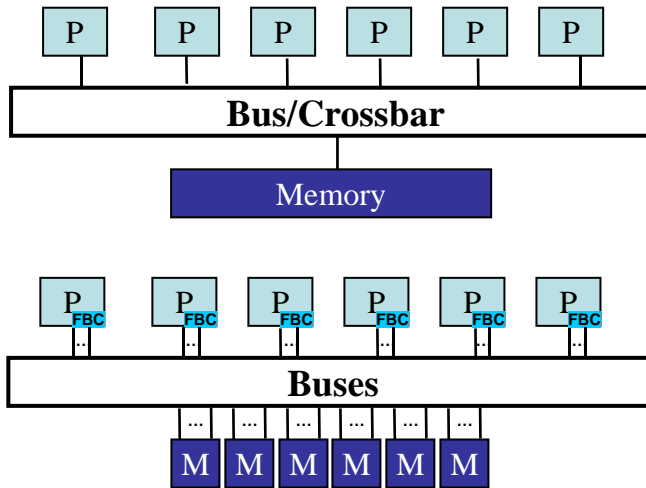
Parallel Computer Architectures

- Top500 List now dominated by MPPs, Constellations and Clusters
- The MIMD model “won”.
- A much more useful way to classification is by memory model
 - *shared* memory
 - *distributed* memory
- Note that the distinction is (mostly) logical, not physical: distributed memory systems could still be single systems (e.g. Cray XT3) or a set of computers (e.g. clusters)

Clusters and Constellations

- (Commodity) Clusters
 - collection of independent nodes connected with a communications network
 - each node a stand-alone computer
 - both nodes and interconnects available on the open market
 - each node may have more than one processor (i.e., be an SMP)
- Constellations
 - clusters where there are more processors within the node than there are nodes interconnected
 - not very many of these any more (SGI Altix)

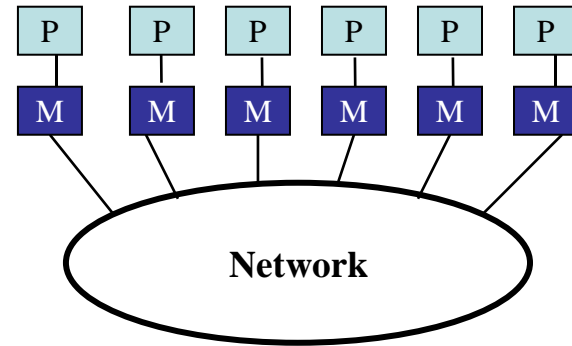
Shared and Distributed Memory



Shared memory: single address space. All processors have access to a pool of shared memory.
(e.g., Single Cluster node (2-way, 4-way, ...), IBM Power5 node, Cray X1E)

Methods of memory access :

- Bus
- Distributed Switch
(Fabric Bus Controller for each Processor)
- Crossbar

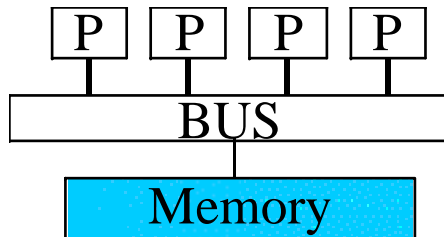


Distributed memory: each processor has it's own local memory. Must do message passing to exchange data between processors.
(examples: Linux Clusters, Cray XT3)

Methods of memory access :

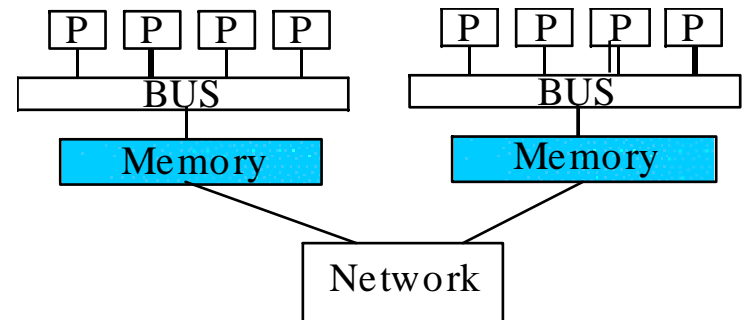
- single switch or switch hierarchy with fat tree, etc. topology

Shared Memory: UMA and NUMA



Uniform Memory Access (UMA):
Each processor has uniform access time to memory - also known as symmetric multiprocessors (SMPs) (example: Sun E25000 at TACC)

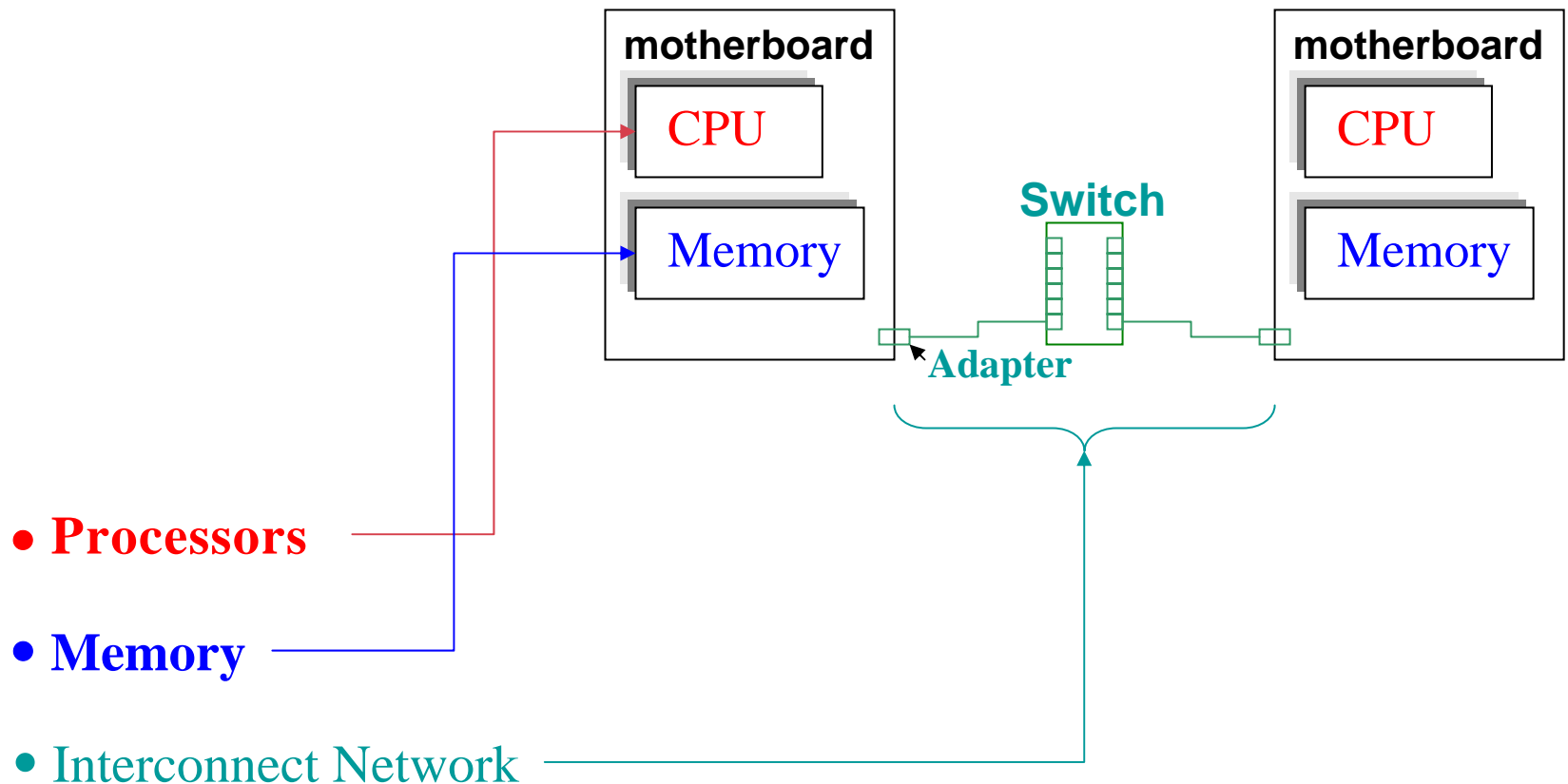
Non-Uniform Memory Access (NUMA):
Time for memory access depends on location of data; also known as Distributed Shared memory machines. Local access is faster than non-local access. Easier to scale than SMPs (e.g.: SGI Origin 2000)



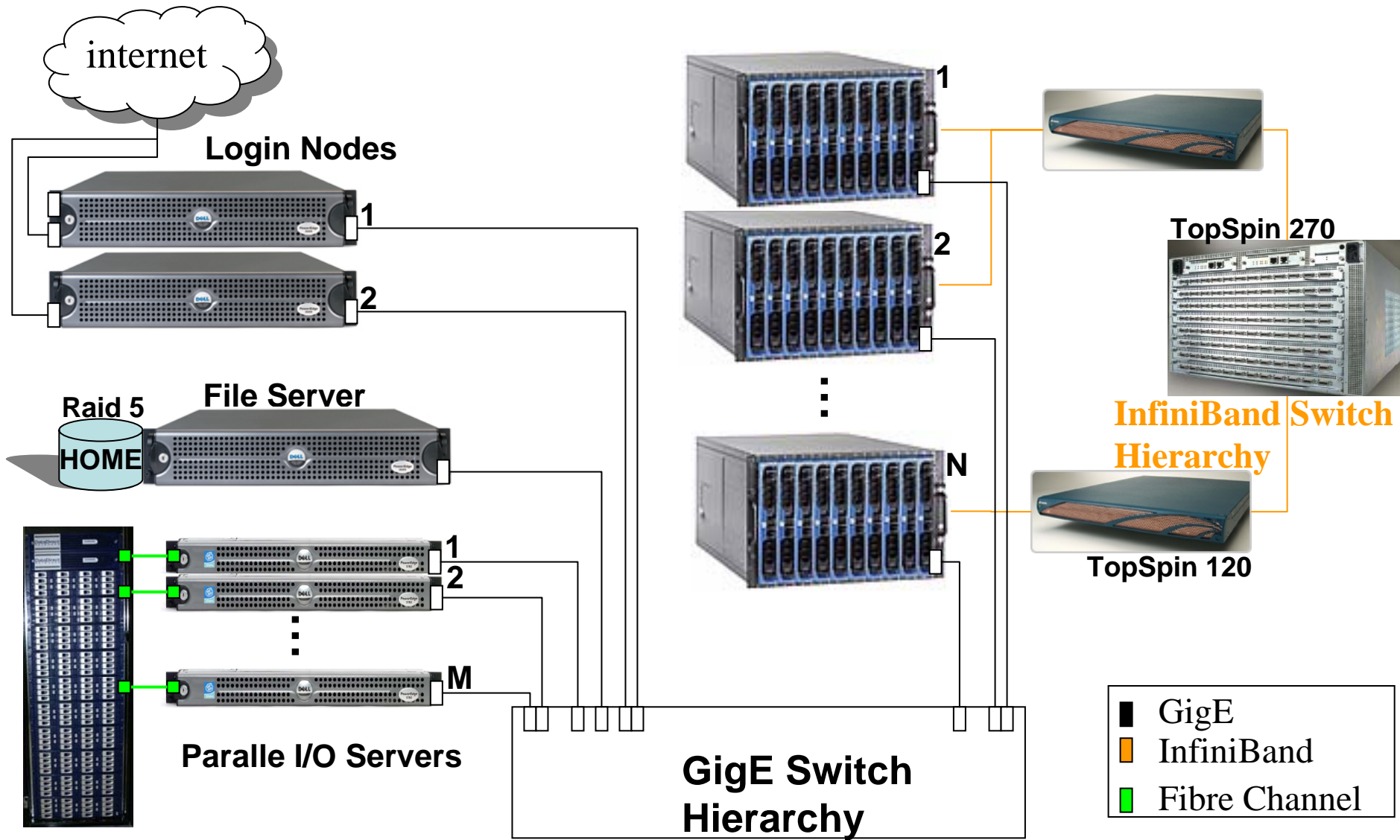
Memory Access Problems

- SMP systems do not scale well
 - bus-based systems can become saturated
 - large, fast (high bandwidth, low latency) crossbars are expensive
 - cache-coherency is hard to maintain at scale (we'll get to what this means in a minute)
- Distributed systems scale well, but:
 - they are harder to program (message passing)
 - interconnects have higher latency
 - makes parallel algorithm development and programming harder

Basic Anatomy of a Server/Desktop/Laptop/Cluster-node



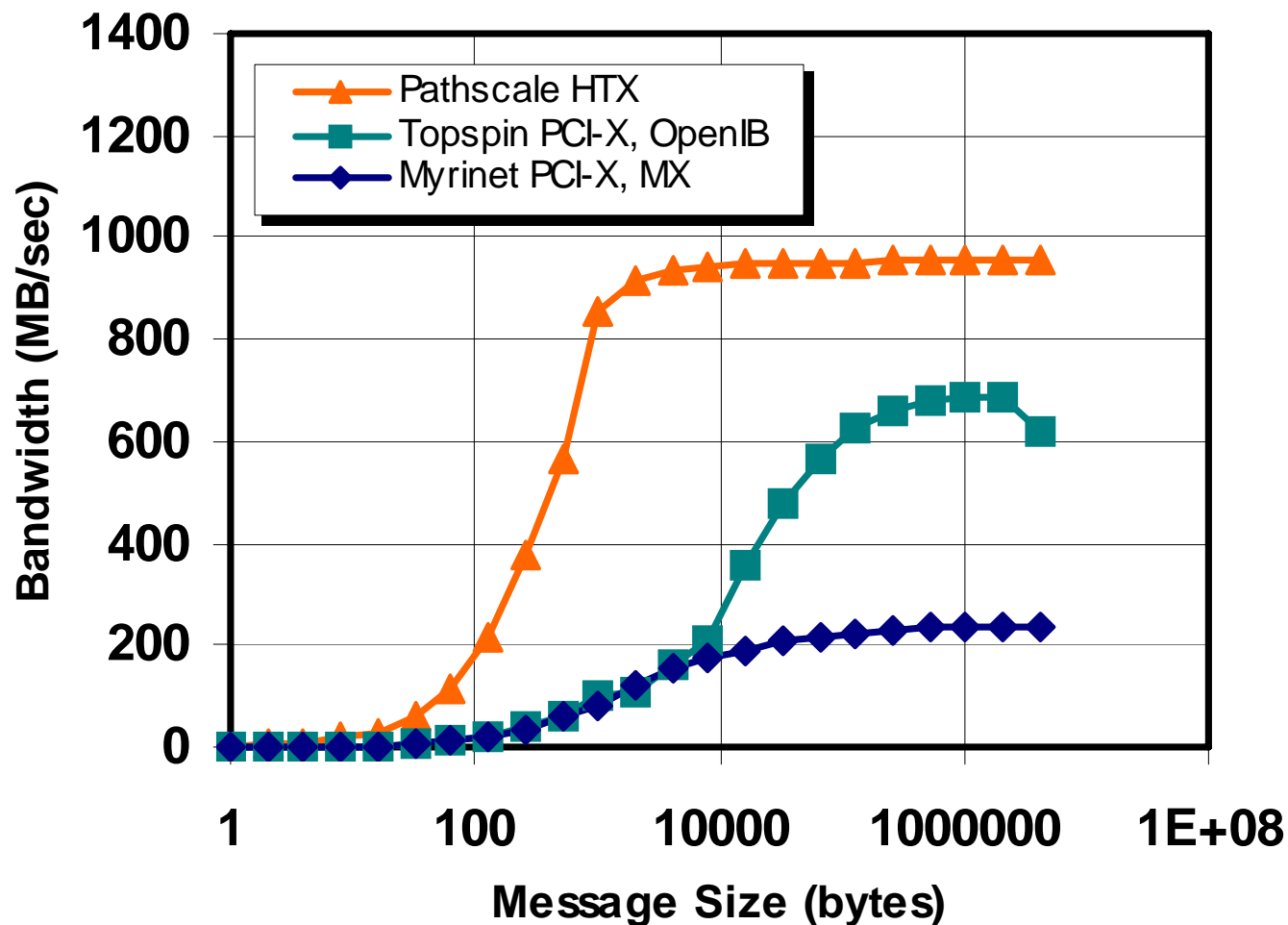
Lonestar @ TACC



Interconnects

- Started with FastEthernet (*Beowulf* @ NASA)
 - 100 Mb/s, 100 μ s latency
 - quickly transitioned to higher bandwidth, lower latency solutions
- Now
 - Ethernet/IP network for administrative work
 - InfiniBand, Myrinet, Quadrics for MPI traffic

Interconnect Performance



Theory of interconnects

- Degree of a node: number of in/outbound connections. The more the better, but also harder to design and more costly
- Diameter: how “far away” can data be
- Topology: What is the actual ‘shape’ of the interconnect? Are the nodes connect by a 2D mesh? A ring? Something more elaborate?
- => some graph theory

Practical issues in interconnects

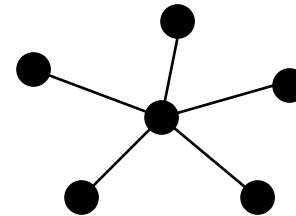
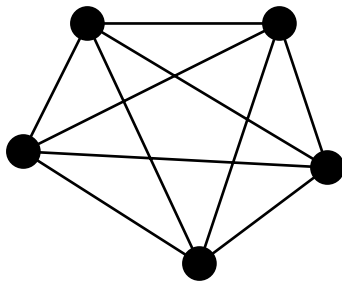
- Latency : How long does it take to start sending a "message"? Units are generally microseconds or milliseconds.
- Bandwidth : What data rate can be sustained once the message is started? Units are Mbytes/sec or Gbytes/sec.

Node Communication in Clusters

- Processors can be connected by a variety of interconnects
- Static/Direct
 - point-to-point, processor-to-processor
 - no switch
 - major types/topologies
 - completely connected
 - star
 - linear array
 - ring
 - n-d mesh
 - n-d torus
 - n-d hyper cube
- Dynamic
 - processors connect to switches
 - major types
 - crossbar
 - fat tree

Completely Connected and Star Networks

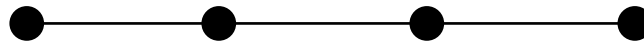
- Completely Connected : Each processor has direct communication link to every other processor



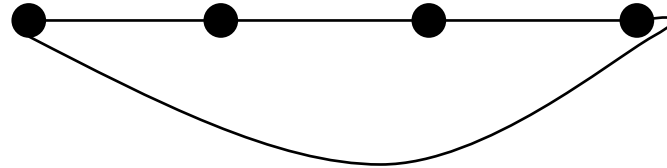
- Star Connected Network : The middle processor is the central processor. Every other processor is connected to it. Counter part of Cross Bar switch in Dynamic interconnect.

Arrays and Rings

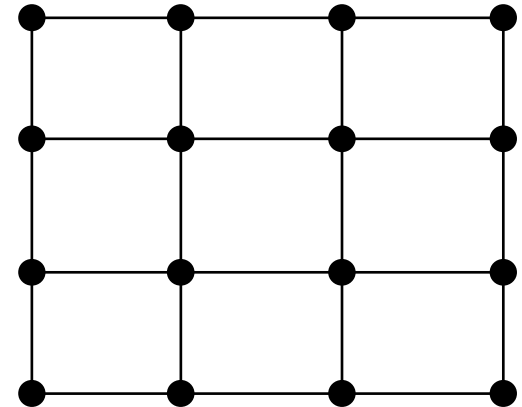
- Linear Array :



- Ring :

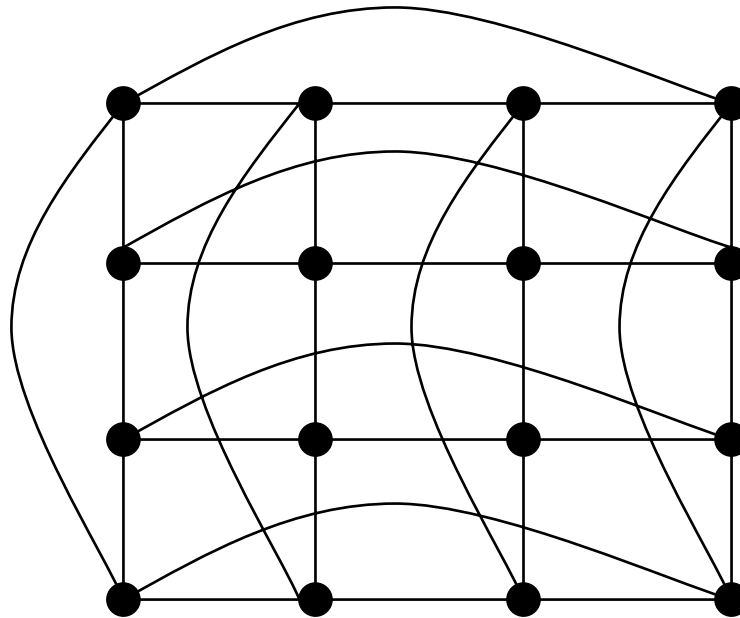


- Mesh Network (e.g. 2D-array)



Torus

2-d Torus (2-d version of the ring)



A word about bandwidth

- Sending data over a single wire incurs latency, then speed determined by bandwidth.
- Multiple wires: multiple latencies, same bandwidth
- Sometimes shortcuts possible: `wormhole routing`

Measures of bandwidth

- Aggregate bandwidth: total data rate if every processor sending: total capacity of the wires. This can be very high and quite unrealistic.
- Imagine processor i sending to $P/2+i$:
`Contention'
- Bisection bandwidth: bandwidth across the minimum number of wires that would split the machine in two.

Hypercubes

- Hypercube Network : A multidimensional mesh of processors with exactly two processors in each dimension. A d dimensional processor consists of

$$p = 2^d \text{ processors}$$

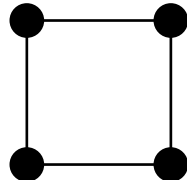
- Shown below are 0, 1, 2, and 3D hypercubes



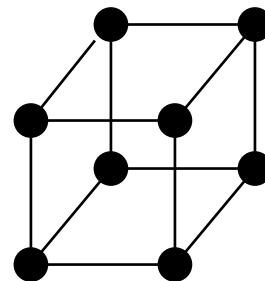
0-D



1-D



2-D



3-D

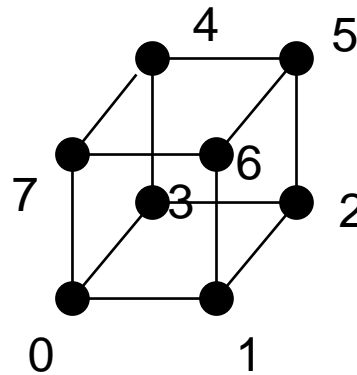
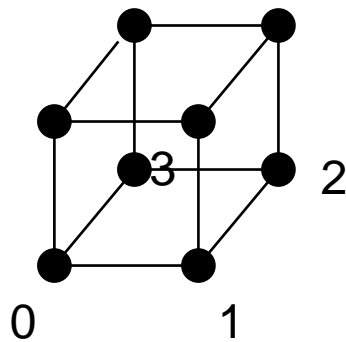
hypercubes

Pros and cons of hypercubes

- Pro: processors are close together: never more than $\log(P)$
- Lots of bandwidth
- Con: the number of wires out of a processor depends on P : complicated design
- Values of P other than 2^p not possible.

Mapping applications to hypercubes

- Is there a natural mapping from 1,2,3D to a hypercube?
- => Gray codes
- Recursive definition: number subcube, then other subcube in mirroring order.



Subsequent processors (in the Linear ordering) all one link apart

Recursive definition:

0 | 1

0 0 | 1 1

0 1 | 1 0

0 0 0 0 | 1 1 1 1

0 0 1 1 | 1 1 0 0

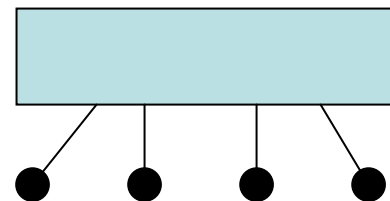
0 1 1 0 | 0 1 1 0

Busses/Hubs and Crossbars

Hub/Bus: Every processor shares the communication links

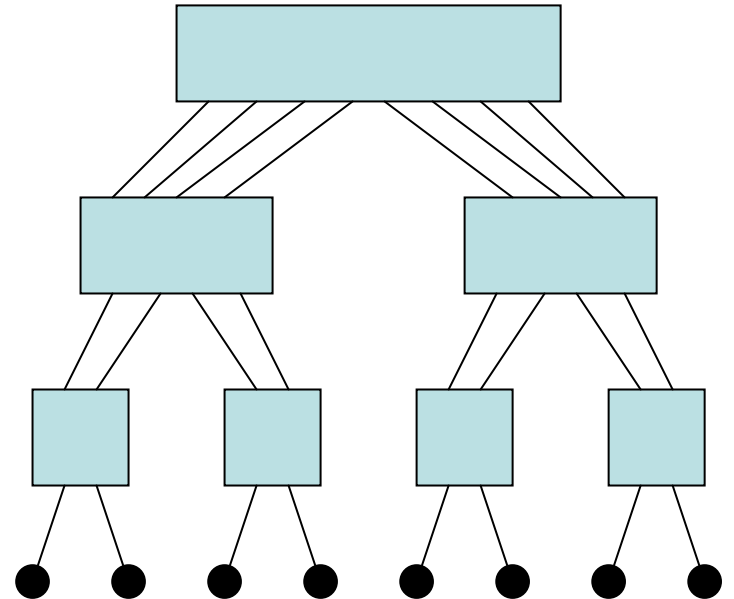


Crossbar Switches: Every processor connects to the switch which routes communications to their destinations



Fat Trees

- Multiple switches
- Each level has the same number of links in as out
- Increasing number of links at each level
- Gives full bandwidth between the links
- Added latency the higher you go



Interconnects

- Diameter
 - maximum distance between any two processors in the network.
 - The distance between two processors is defined as the shortest path, in terms of links, between them.
 - completely connected network is 1, for star network is 2, for ring is $p/2$ (for p even processors)
- Connectivity
 - measure of the multiplicity of paths between any two processors (# arcs that must be removed to break the connection).
 - high connectivity is desired since it lowers contention for communication resources.
 - 1 for linear array, 1 for star, 2 for ring, 2 for mesh, 4 for torus
 - technically 1 for traditional fat trees, but there is redundancy in the switch infrastructure

Interconnects

- **Bisection width**
 - Minimum # of communication links that have to be removed to partition the network into two equal halves. Bisection width is
 - 2 for ring, $\text{sq. root}(p)$ for mesh with p (even) processors, $p/2$ for hypercube, $(p \cdot p)/4$ for completely connected (p even).
- **Channel width**
 - of physical wires in each communication link
- **Channel rate**
 - peak rate at which a single physical wire link can deliver bits
- **Channel BW**
 - peak rate at which data can be communicated between the ends of a communication link
 - = (channel width) * (channel rate)
- **Bisection BW**
 - minimum volume of communication found between any 2 halves of the network with equal # of procs
 - = (bisection width) * (channel BW)

RAID

- Was: Redundant Array of Inexpensive Disks
- Now: Redundant Array of Independent Disks
- Multiple disk drives working together to:
 - increase capacity of a single logical volume
 - increase performance
 - improve reliability/add fault tolerance
- 1 Server with RAIDed disks can provide disk access to multiple nodes with NFS

Parallel Filesystems

- Use multiple servers together to aggregate disks
 - utilizes RAIDed disks
 - improved performance
 - even higher capacities
 - may use high-performance network
- Vendors/Products
 - CFS/Lustre
 - IBM/GPFS
 - IBRIX/IBRIXFusion
 - RedHat/GFS
 - ...