Lecture 11: Distributed Memory Machines and Programming

CSCE 569 Parallel Computing

Department of Computer Science and Engineering
Yonghong Yan
yanyh@cse.sc.edu
http://cse.sc.edu/~yanyh

Topics

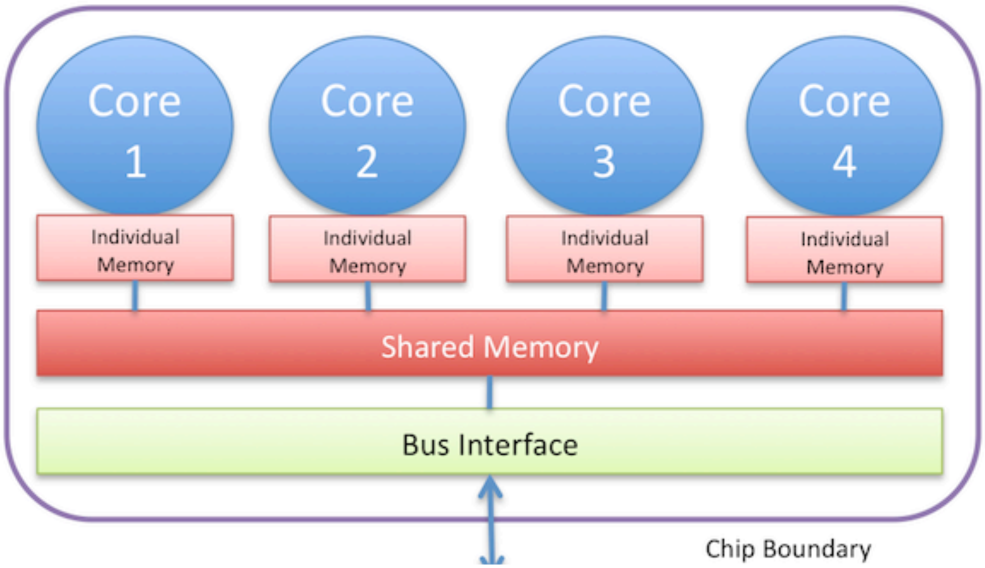
- Introduction
- Programming on shared memory system (Chapter 7)
  - **OpenMP**
- Principles of parallel algorithm design (Chapter 3)
- ☛ **Programming on large scale systems (Chapter 6)**
  - **MPI (point to point and collectives)**
  - Introduction to PGAS languages, UPC and Chapel
- Analysis of parallel program executions (Chapter 5)
  - **Performance Metrics for Parallel Systems**
    - **Execution Time, Overhead, Speedup, Efficiency, Cost**
  - **Scalability of Parallel Systems**
  - **Use of performance tools**

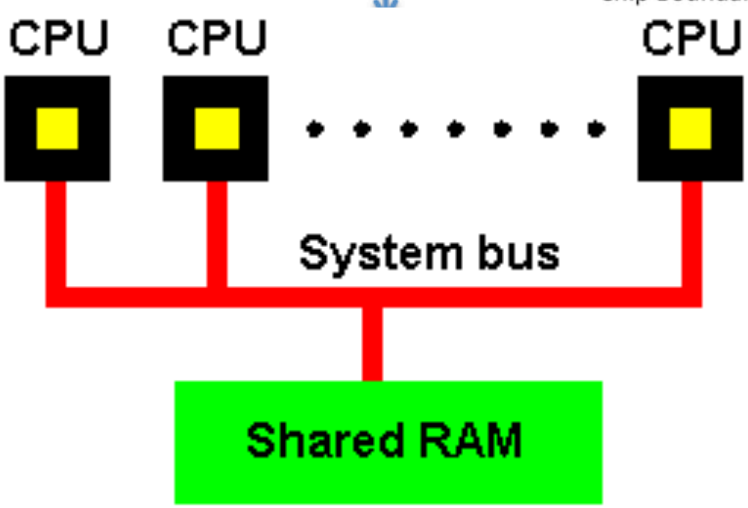
Acknowledgement

- Slides adapted from U.C. Berkeley course CS267/EngC233 Applications of Parallel Computers by Jim Demmel and Katherine Yelick, Spring 2011
  - http://www.cs.berkeley.edu/~demmel/cs267_Spr11/
- And materials from various sources

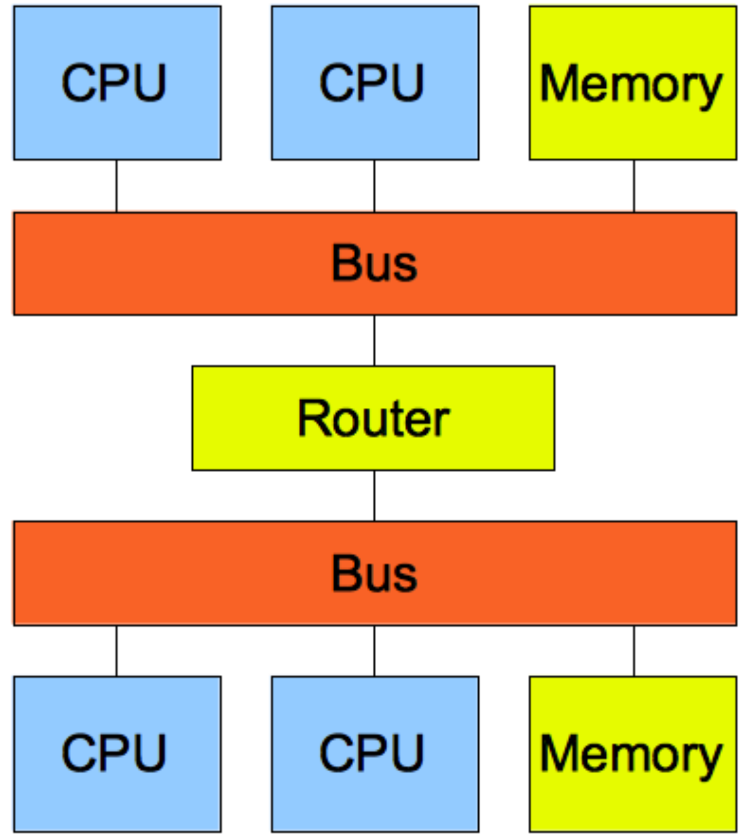
Shared Memory Parallel Systems: Multicore and Multi-CPU

Multi-core Processor





SMP: Multiple processors share RAM and system bus

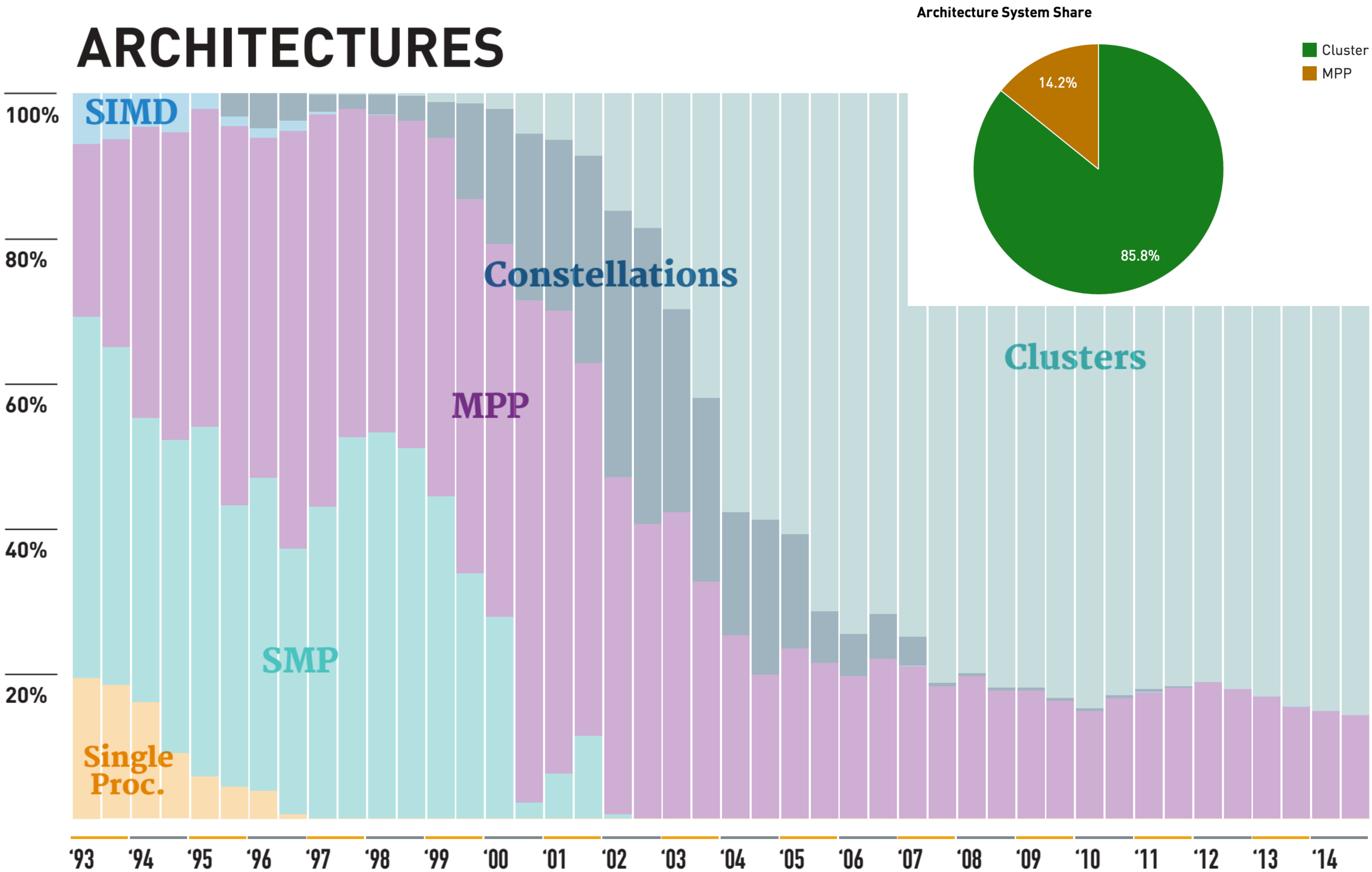


NUMA Architecture

Node-level Architecture and Programming

- Shared memory multiprocessors: multicore, SMP, NUMA
  - Deep memory hierarchy, distant memory much more expensive to access.
  - Machines scale to 10s or 100s of processors
  - Instruction Level Parallelism (ILP), Data Level Parallelism (DLP) and Thread Level Parallelism (TLP)
- Programming
  - OpenMP, PThreads, Cilkplus, etc

HPC Architectures (TOP500, Nov 2014)



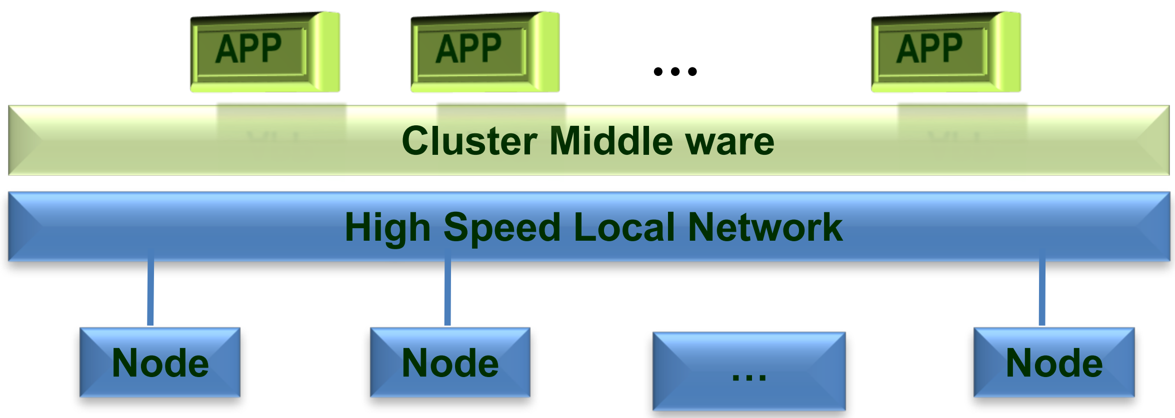
Outline

☛ **Cluster Introduction**

- **Distributed Memory Architectures**
  - **Properties of communication networks**
  - **Topologies**
  - **Performance models**
- Programming Distributed Memory Machines using Message Passing
  - Overview of MPI
  - Basic send/receive use
  - Non-blocking communication
  - Collectives

Clusters

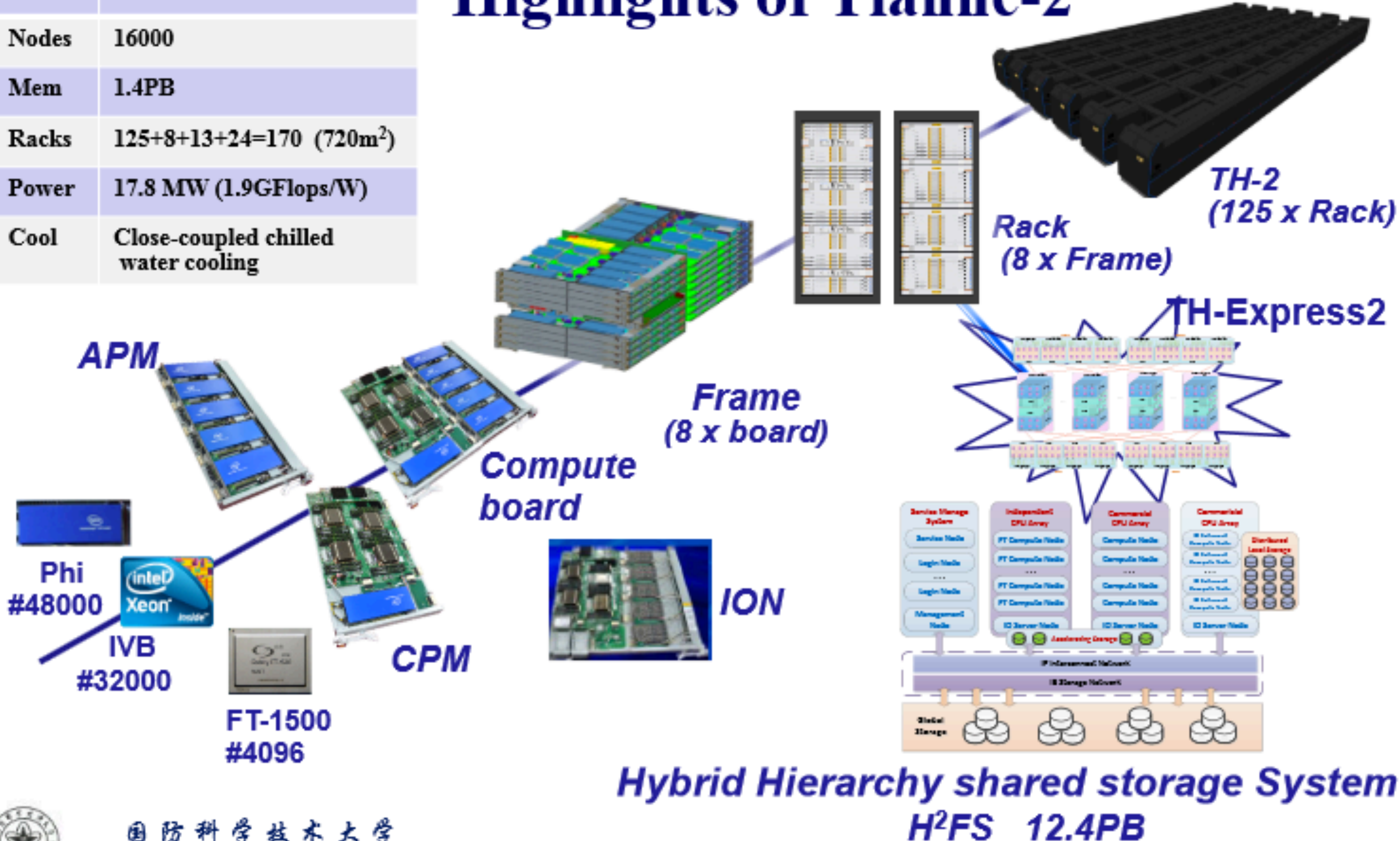
- A group of linked computers, working together closely so that in many respects they form a single computer.
- Consists of
  - Nodes(Front + computing)
  - Network
  - Software: OS and middleware




DELL

Perf	54.9PFlops / 33.86PFlops
Nodes	16000
Mem	1.4PB
Racks	125+8+13+24=170 (720m²)
Power	17.8 MW (1.9GFlops/W)
Cool	Close-coupled chilled water cooling

Highlights of Tianhe-2

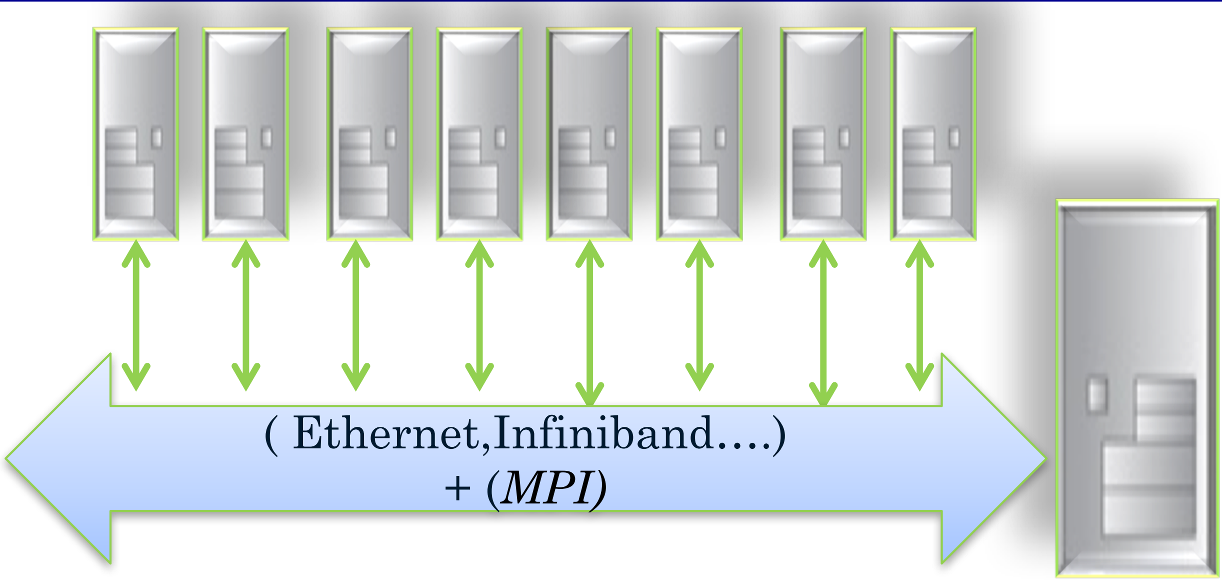


http://www.top500.org/lists/2016/06/

Top 10 of Top500

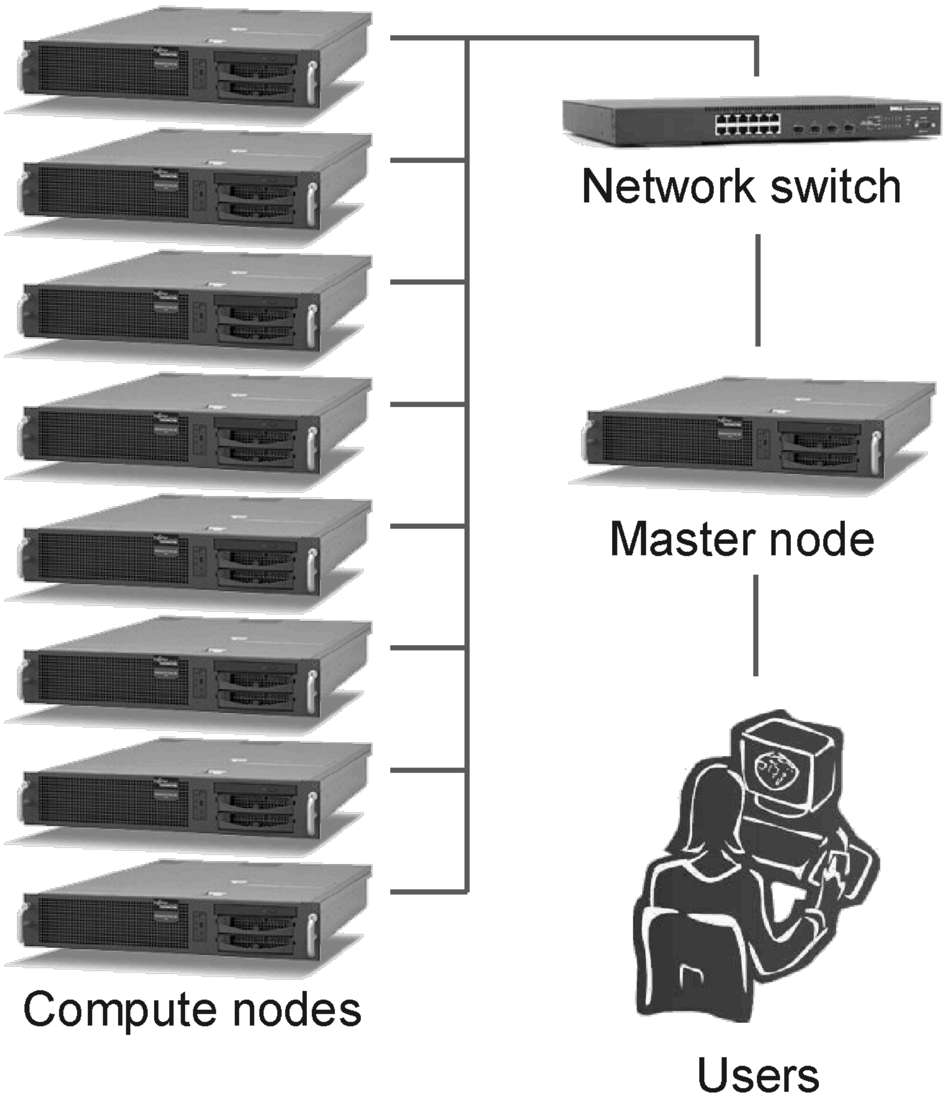
RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
6	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
9	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
10	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834

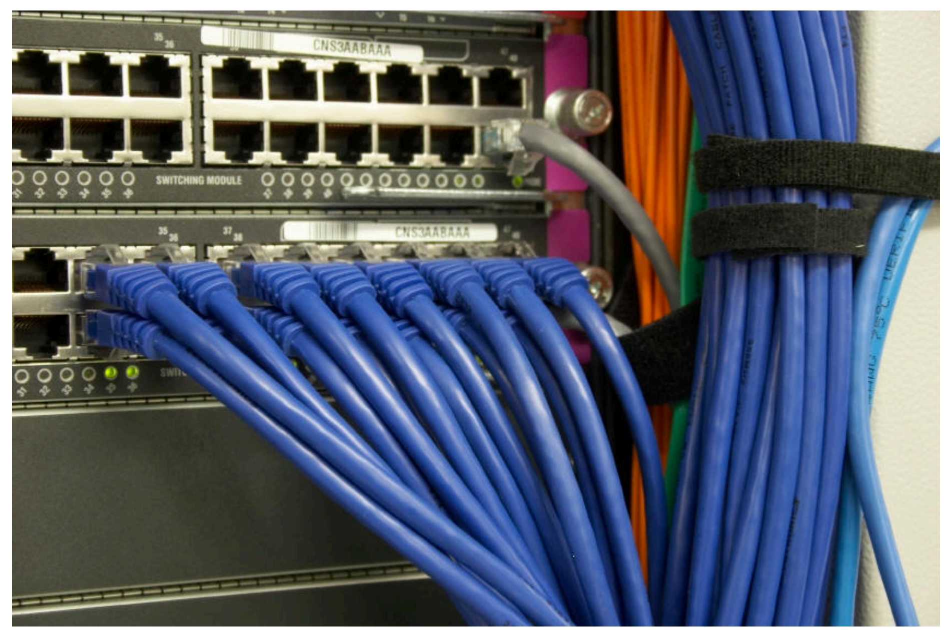
HPC Beowulf Cluster



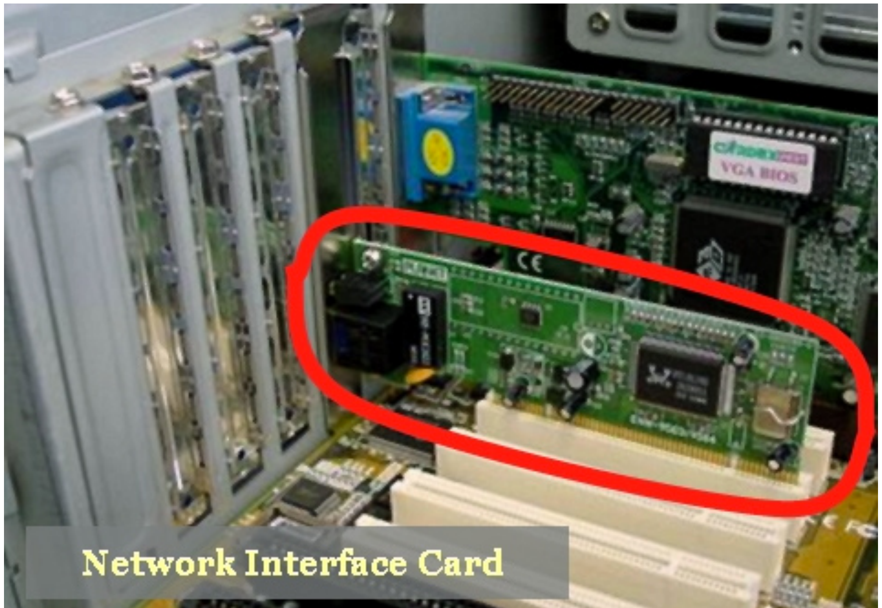
- Master node: or service/front node (used to interact with users locally or remotely)
- Computing Nodes : performance computations
- Interconnect and switch between nodes: e.g. G/10G-bit Ethernet, Infiniband
- Inter-node programming
  - MPI(Message Passing Interface) is the most commonly used one.

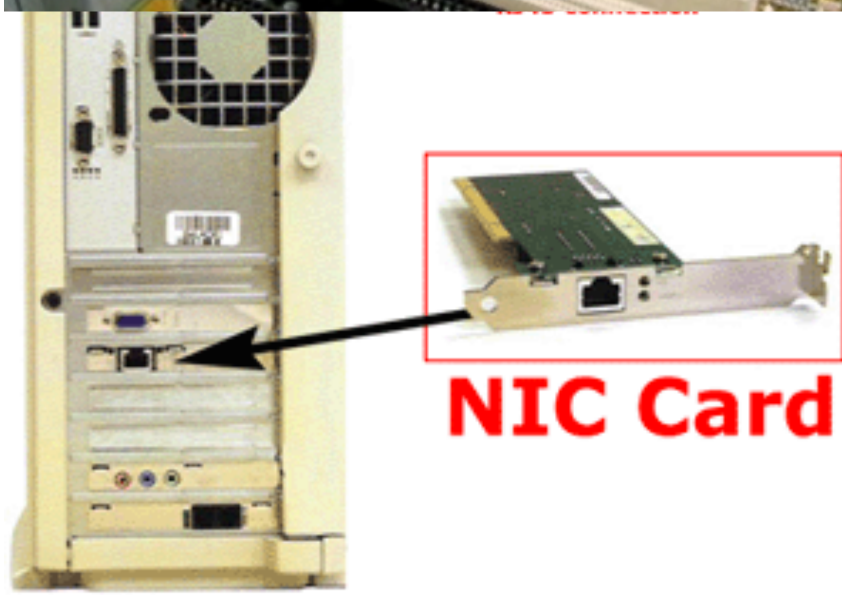
Network Switch

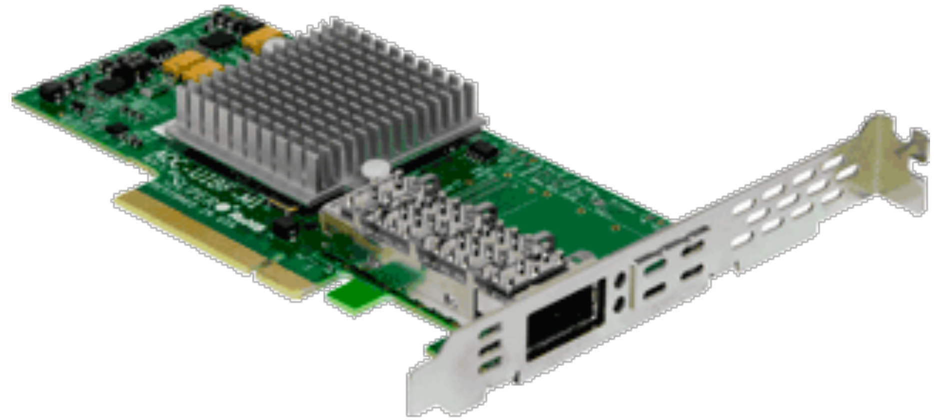




Network Interface Card (NIC)







Outline

- **Cluster Introduction**
- **Distributed Memory Architectures**
  - **Properties of communication networks**
  - **Topologies**
  - **Performance models**
- Programming Distributed Memory Machines using Message Passing
  - Overview of MPI
  - Basic send/receive use
  - Non-blocking communication
  - Collectives

Network Analogy

- To have a large number of different transfers occurring at once, you need a large number of distinct wires
  - Not just a bus, as in shared memory
- Networks are like streets:
  - Link = street.
  - Switch = intersection.
  - Distances (hops) = number of blocks traveled.
  - Routing algorithm = travel plan.
- **Properties:**
  - **Latency: how long to get between nodes in the network.**
  - **Bandwidth: how much data can be moved per unit time.**
    - **Bandwidth is limited by the number of wires and the rate at which each wire can accept data.**

Latency and Bandwidth

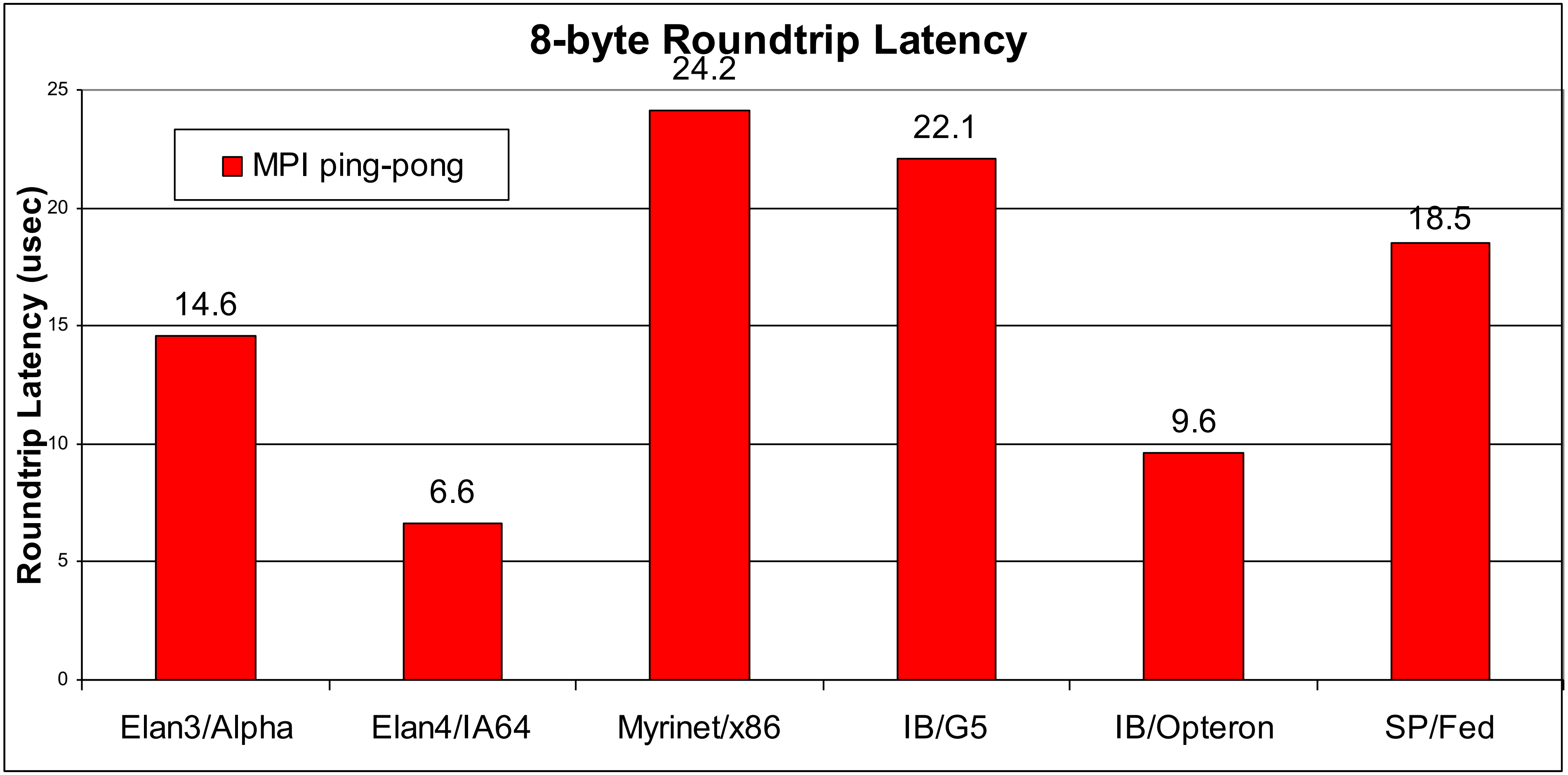
- Latency: Time to travel from one location to another for a vehicle
  - Vehicle type (large or small messages)
  - Road/traffic condition, speed-limit, etc
- Bandwidth: How many cars and how fast they can travel from one location to another
  - Number of lanes

Performance Properties of a Network: Latency

- Diameter: the maximum (over all pairs of nodes) of the shortest path between a given pair of nodes.
- Latency: delay between send and receive times
  - Latency tends to vary widely across architectures
  - Vendors often report hardware latencies (wire time)
  - Application programmers care about software latencies (user program to user program)
- Observations:
  - Latencies differ by 1-2 orders across network designs
  - Software/hardware overhead at source/destination dominate cost (1s-10s usecs)
  - Hardware latency varies with distance (10s-100s nsec per hop) but is small compared to overheads
- **Latency is key for programs with many small messages**

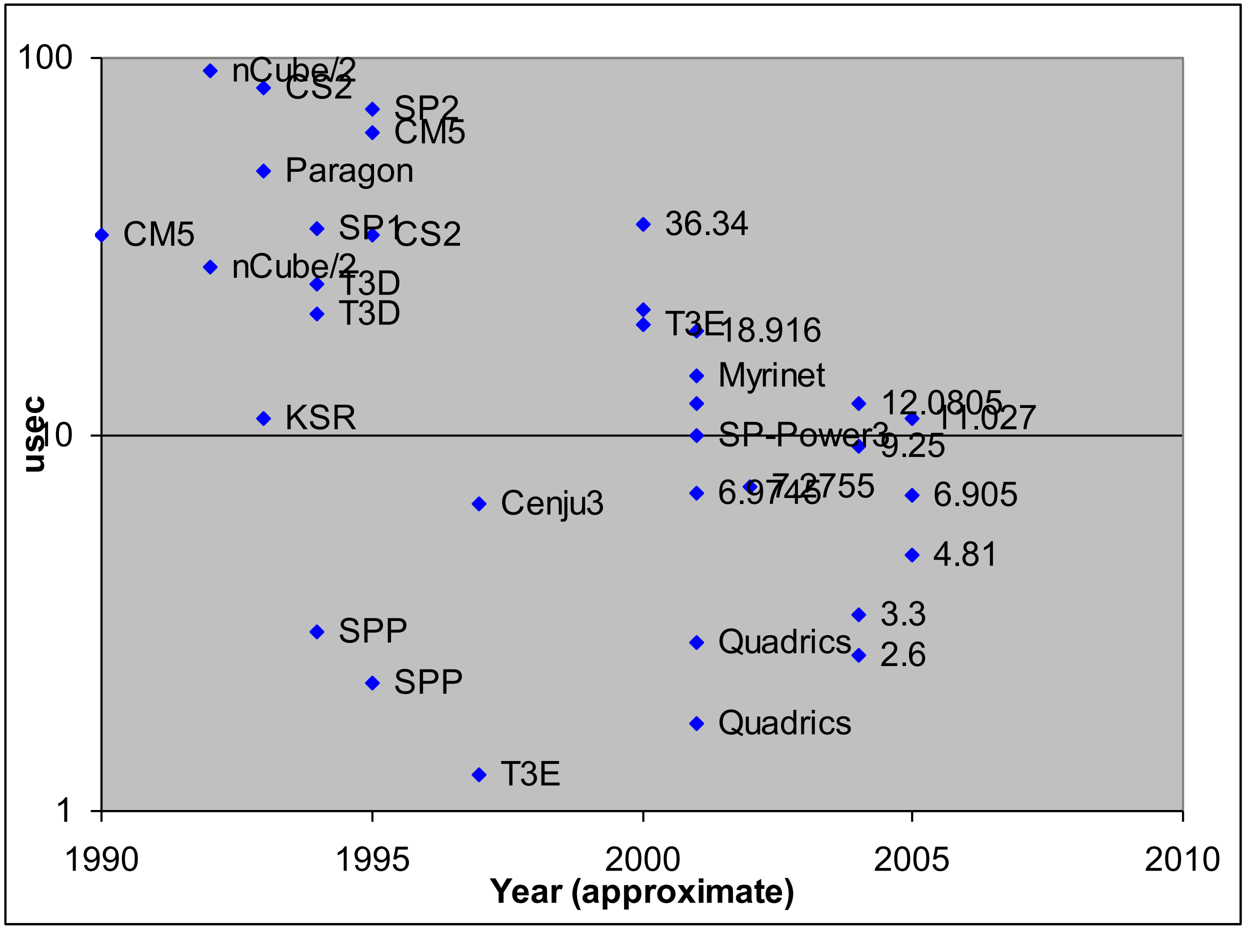
I second = 10^3 millseconds (ms) = 10^6 microseconds (us) = 10^9 nanoseconds (ns)

Latency on Some Machines/Networks



- Latencies shown are from a ping-pong test using MPI
- These are roundtrip numbers: many people use ½ of roundtrip time to approximate 1-way latency (which can't easily be measured)

End to End Latency (1/2 roundtrip) Over Time



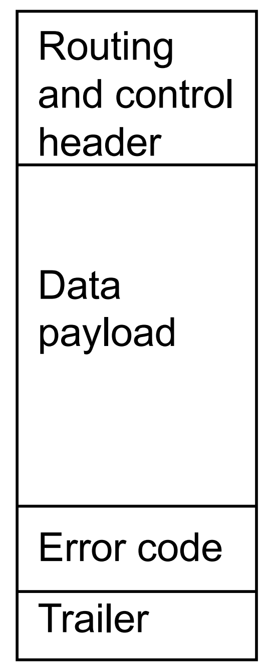
- Latency has not improved significantly, unlike Moore's Law
  - T3E (shmem) was lowest point – in 1997

Data from Kathy Yelick, UCB and NERSC

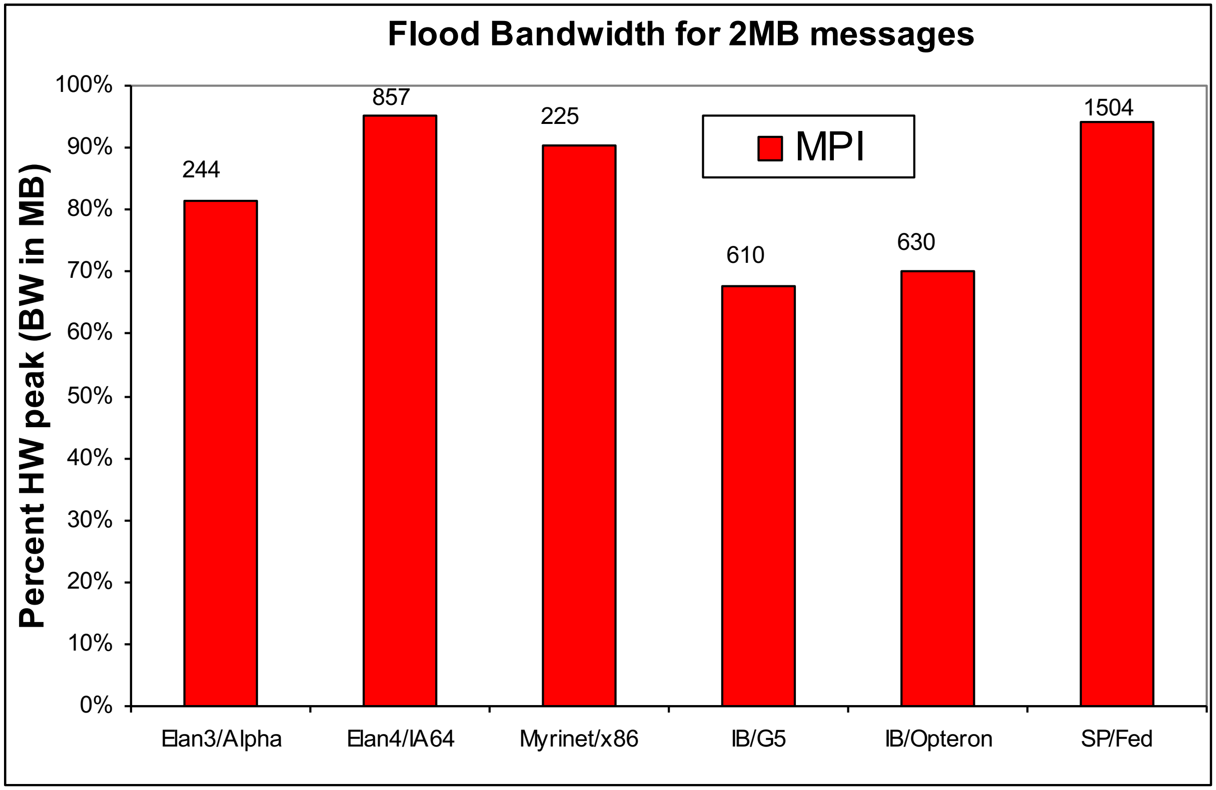
Performance Properties of a Network: Bandwidth

- The bandwidth of a link = # wires / time-per-bit
- Bandwidth typically in Gigabytes/sec (GB/s), i.e., 8* 2^{20} bits per second
- Effective bandwidth is usually lower than physical link bandwidth due to packet overhead.

- **Bandwidth is important for applications with mostly large messages**



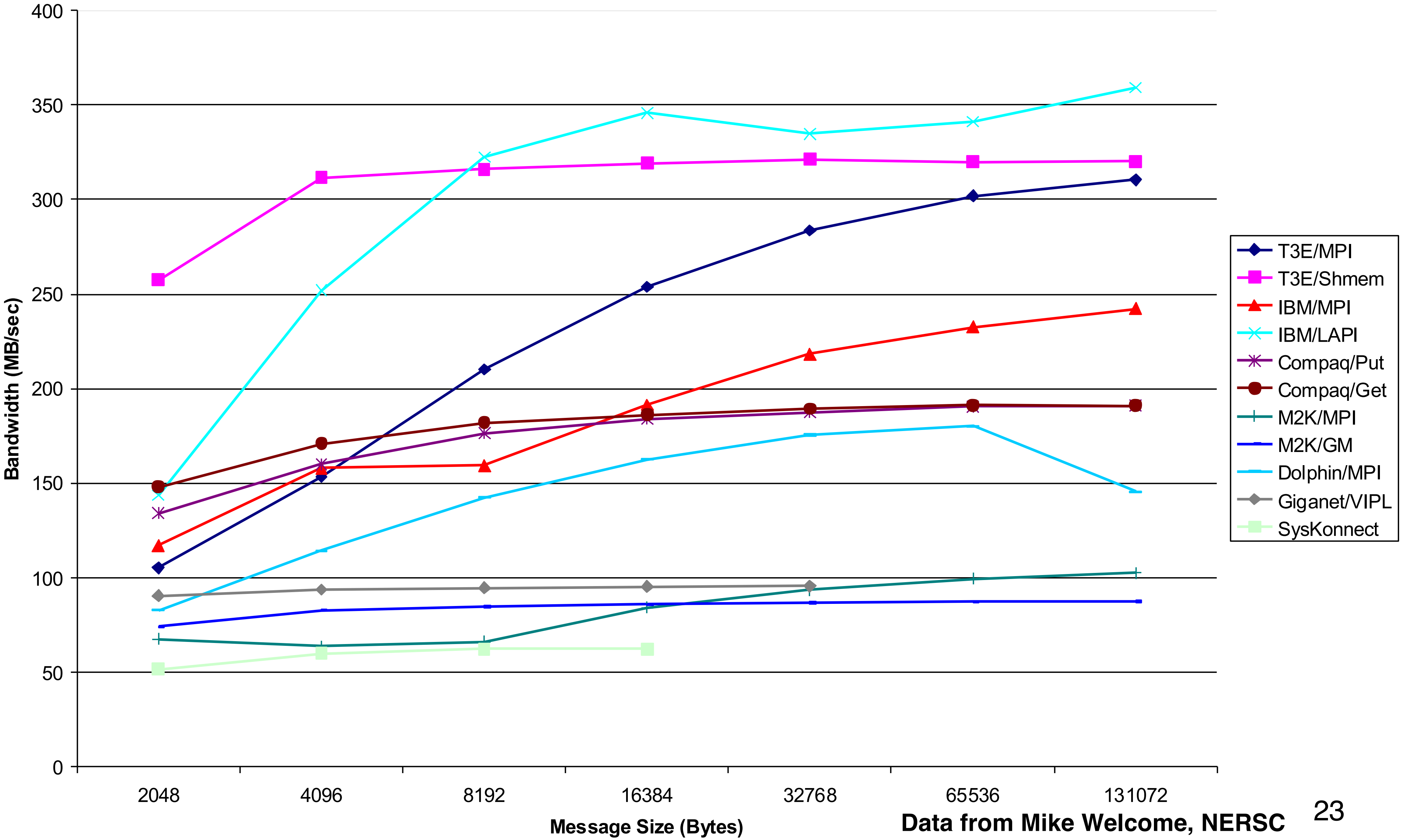
Bandwidth on Some Networks



- Flood bandwidth (throughput of back-to-back 2MB messages)

Bandwidth Chart

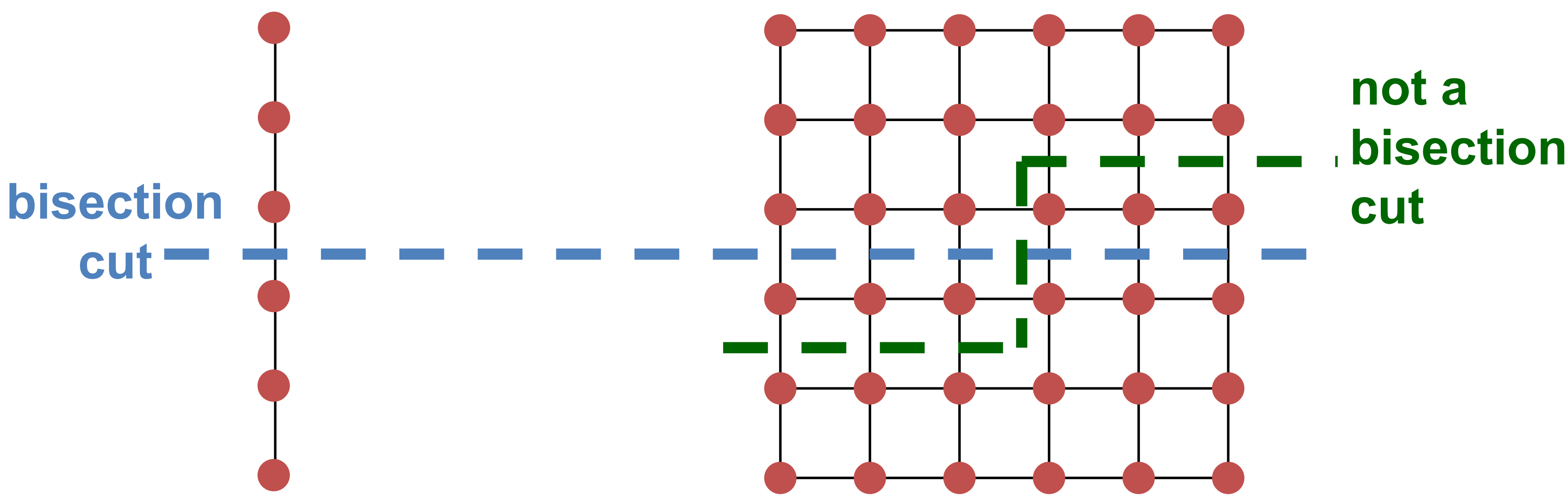
Note: bandwidth depends on SW, not just HW



Data from Mike Welcome, NERSC

Performance Properties of a Network: Bisection Bandwidth

- Bisection bandwidth: bandwidth across smallest cut that divides network into two equal halves
- Bandwidth across "narrowest" part of the network



bisection bw= link bw *bisection bw = sqrt(n) * link bw*

- Bisection bandwidth is important for algorithms in which all processors need to communicate with all others

Other Characteristics of a Network

- Topology (how things are connected)
  - Crossbar, ring, 2-D and 3-D mesh or torus, hypercube, tree, butterfly, perfect shuffle ....
- Routing algorithm:
  - Example in 2D torus: all east-west then all north-south (avoids deadlock).
- Switching strategy:
  - Circuit switching: full path reserved for entire message, like the telephone.
  - Packet switching: message broken into separately-routed packets, like the post office.
- Flow control (what if there is congestion):
  - Stall, store data temporarily in buffers, re-route data to other nodes, tell source node to temporarily halt, discard, etc.

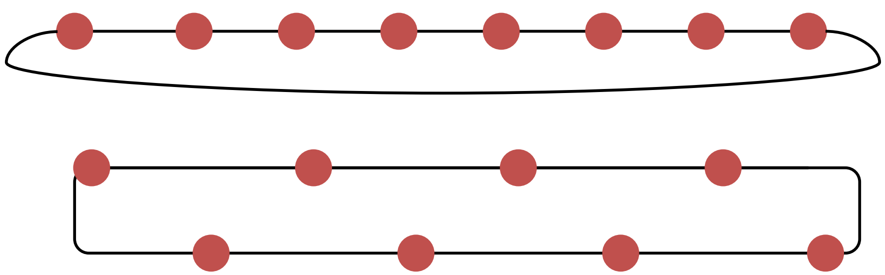
Network Topology

- In the past, there was considerable research in network topology and in mapping algorithms to topology.
  - Key cost to be minimized: number of "hops" between nodes (e.g. "store and forward")
  - Modern networks hide hop cost (i.e., "wormhole routing"), so topology is no longer a major factor in algorithm performance.
- Example: On IBM SP system, hardware latency varies from 0.5 usec to 1.5 usec, but user-level message passing latency is roughly 36 usec.
- Need some background in network topology
  - Algorithms may have a communication topology
  - Topology affects bisection bandwidth.

Linear and Ring Topologies

- Linear array

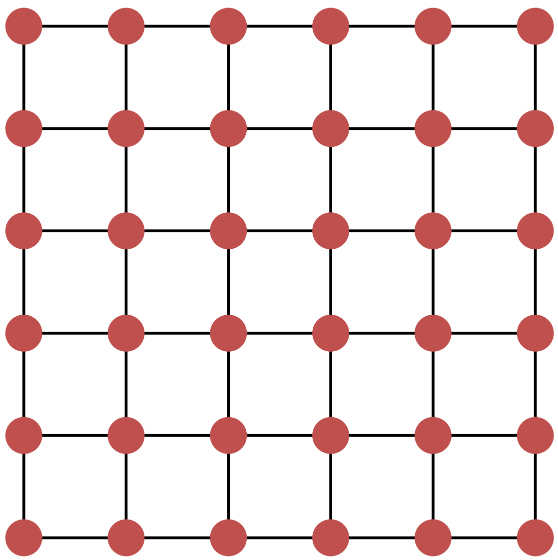
  - Diameter = n-1; average distance ~n/3.
  - Bisection bandwidth = 1 (in units of link bandwidth).
- Torus or Ring



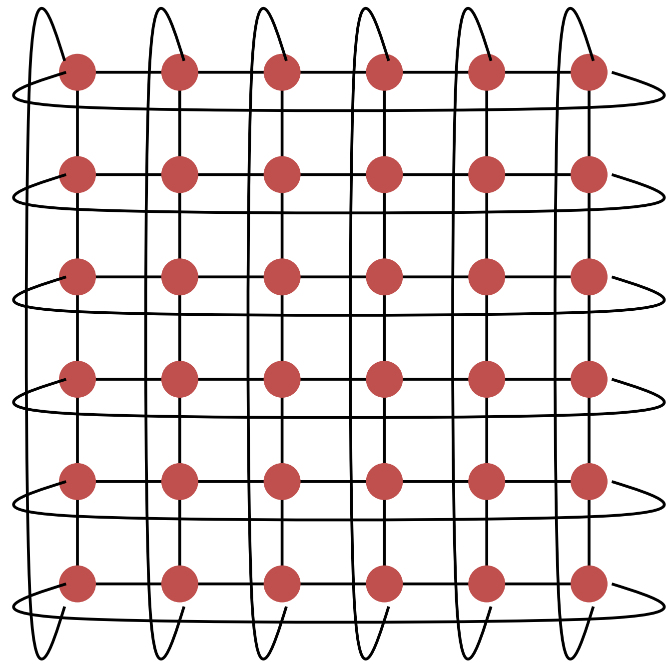
  - Diameter = n/2; average distance ~ n/4.
  - Bisection bandwidth = 2.
  - Natural for algorithms that work with 1D arrays.

Meshes and Tori

- Two dimensional mesh
  - Diameter = 2 * (sqrt(n) – 1)
  - Bisection bandwidth = sqrt(n)



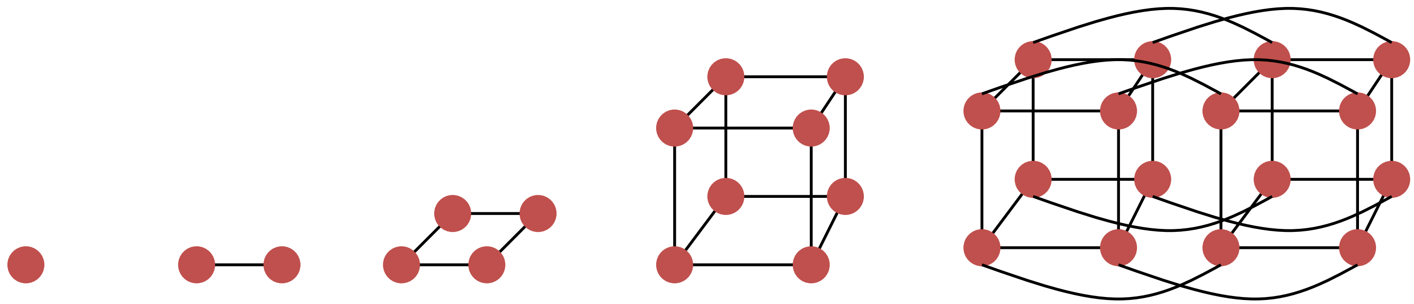
- Two dimensional torus
  - Diameter = sqrt(n)
  - Bisection bandwidth = 2* sqrt(n)



- **Generalizes to higher dimensions**
  - **Cray XT (eg Franklin@NERSC) uses 3D Torus**
- **Natural for algorithms that work with 2D and/or 3D arrays (matmul)**

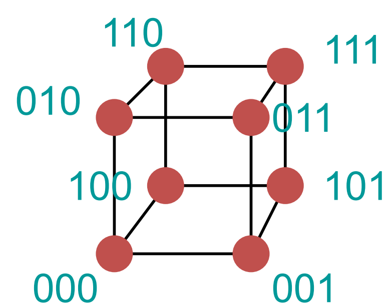
Hypercubes

- Number of nodes n = 2^d for dimension d.
  - Diameter = d.
  - Bisection bandwidth = n/2.

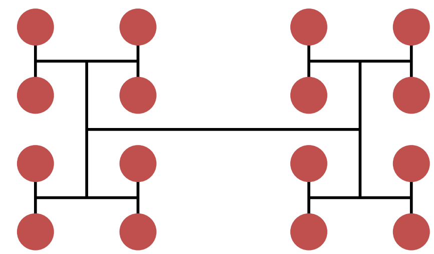


- 0d 1d 2d 3d 4d

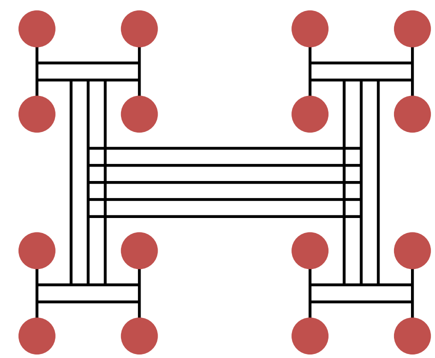
- Popular in early machines (Intel iPSC, NCUBE).
  - Lots of clever algorithms.
- Greycode addressing:
  - Each node connected to others with 1 bit different.

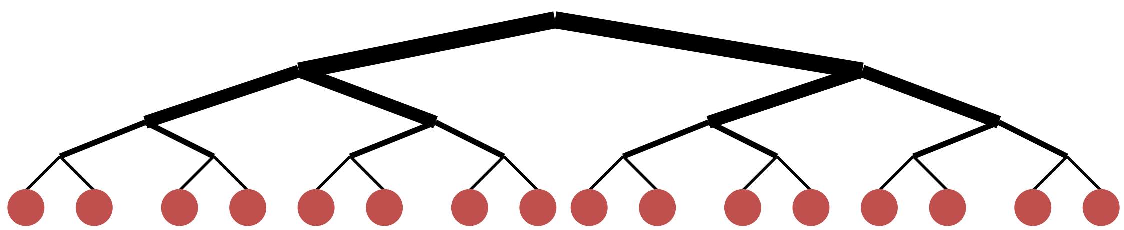


Trees



- Diameter = log n.
- Bisection bandwidth = 1.
- Easy layout as planar graph.
- Many tree algorithms (e.g., summation).
- Fat trees avoid bisection bandwidth problem:
  - More (or wider) links near top.
  - Example: Thinking Machines CM-5.

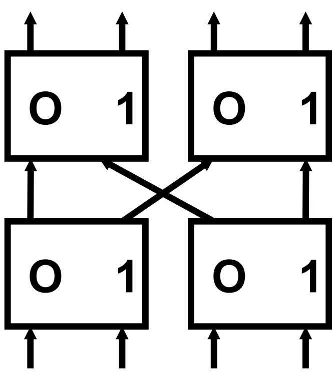




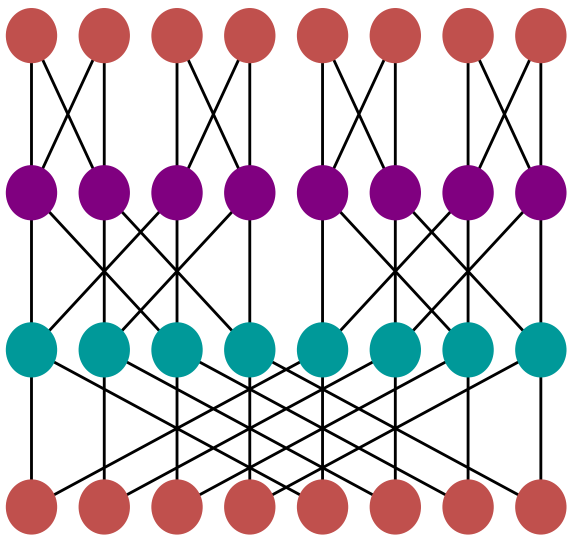
Butterflies

- Diameter = log n.
- Bisection bandwidth = n.
- Cost: lots of wires.
- Used in BBN Butterfly.
- Natural for FFT.

Ex: to get from proc 101 to 110,
Compare bit-by-bit and
Switch if they disagree, else not



butterfly switch



multistage butterfly network

Topologies in Real Machines

newer ↑ older ↓

Cray XT3 and XT4	3D Torus (approx)
Blue Gene/L	3D Torus
SGI Altix	Fat tree
Cray X1	4D Hypercube*
Myricom (Millennium)	Arbitrary
Quadrics (in HP Alpha server clusters)	Fat tree
IBM SP	Fat tree (approx)
SGI Origin	Hypercube
Intel Paragon (old)	2D Mesh
BBN Butterfly (really old)	Butterfly

Many of these are approximations: E.g., the X1 is really a "quad bristled hypercube" and some of the fat trees are not as fat as they should be at the top

Performance Models

Latency and Bandwidth Model

- Time to send message of length n is roughly

$$\textbf{Time} = \textbf{latency} + \textbf{n*cost\_per\_word} = \textbf{latency} + \textbf{n/bandwidth}$$

- Topology is assumed irrelevant.
- Often called "α-β model" and written

$$\textbf{Time} = \alpha + \textbf{n}*\beta$$

- Usually $\alpha \gg \beta \gg$ time per flop.
  - One long message is cheaper than many short ones.

$$\alpha + n*\beta \ll n*(\alpha + 1*\beta)$$

  - Can do hundreds or thousands of flops for cost of one message.
- Lesson: Need large computation-to-communication ratio to be efficient.

Alpha-Beta Parameters on Current Machines

- These numbers were obtained empirically

machine	α	β
T3E/Shm	1.2	0.003
T3E/MPI	6.7	0.003
IBM/LAPI	9.4	0.003
IBM/MPI	7.6	0.004
Quadrics/Get	3.267	0.00498
Quadrics/Shm	1.3	0.005
Quadrics/MPI	7.3	0.005
Myrinet/GM	7.7	0.005
Myrinet/MPI	7.2	0.006
Dolphin/MPI	7.767	0.00529
Giganet/VIPL	3.0	0.010
GigE/VIPL	4.6	0.008
GigE/MPI	5.854	0.00872

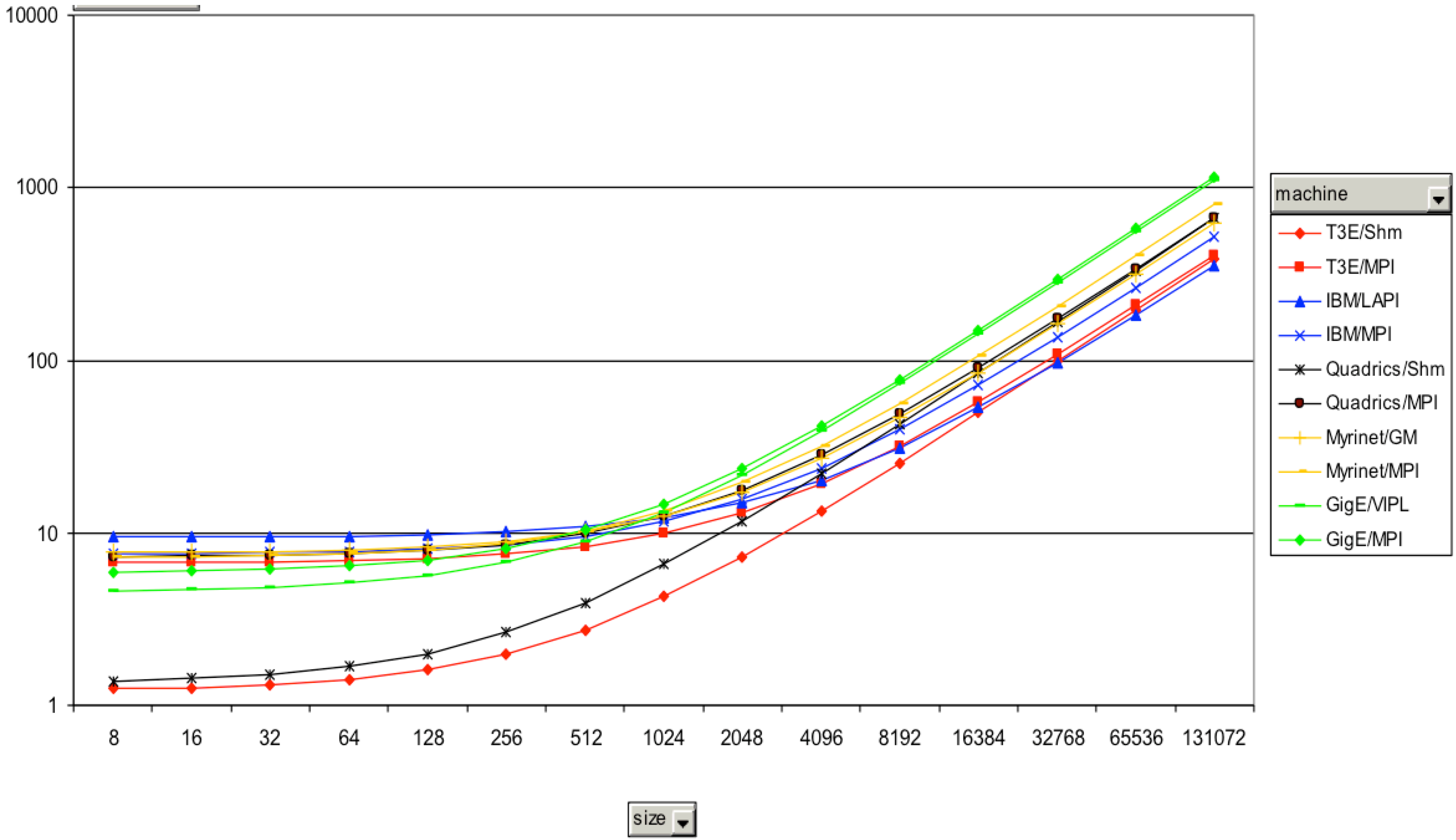
α is latency in usecs
β is BW in usecs per Byte

How well does the model

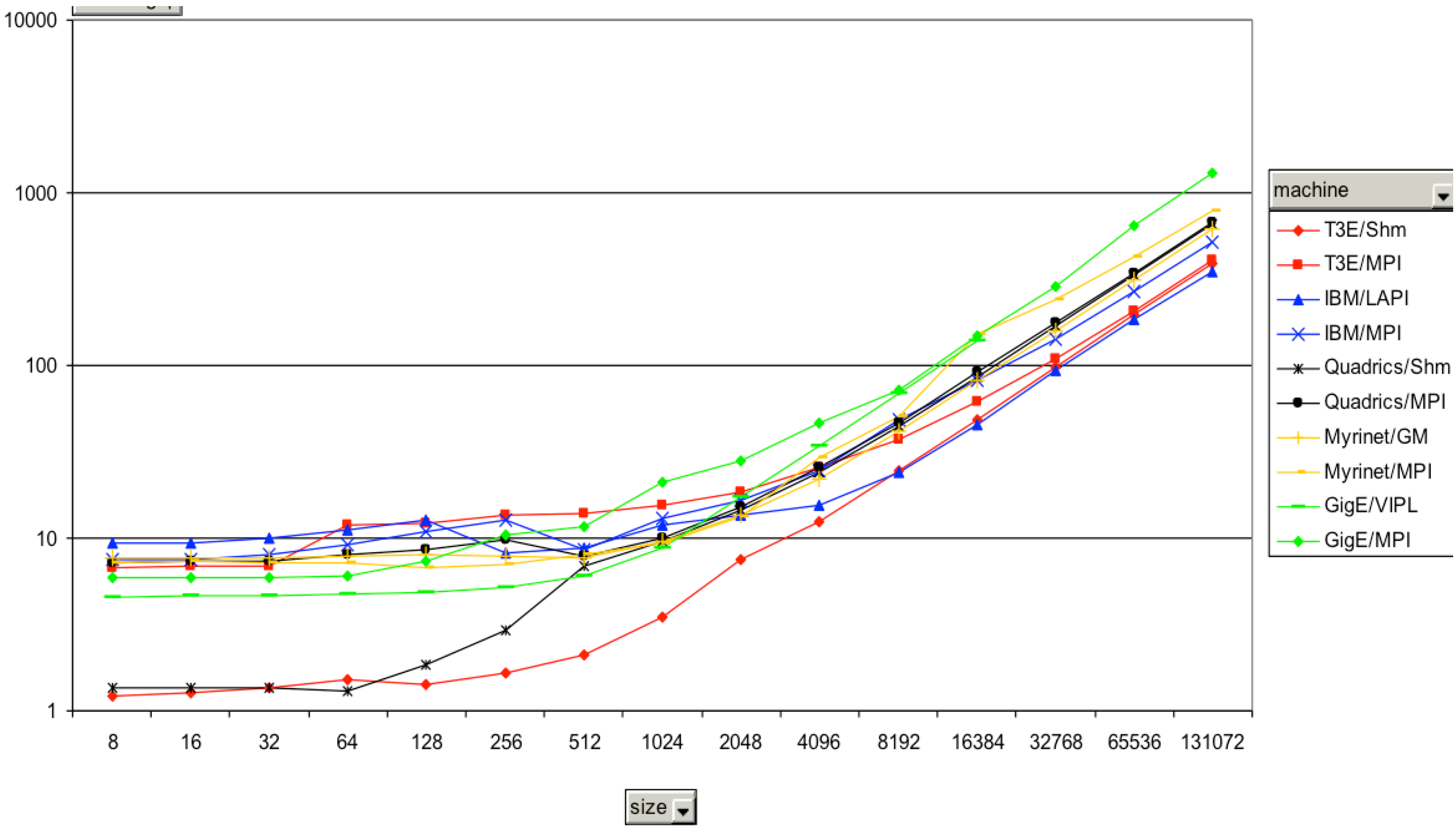
Time = $\alpha + n*\beta$

predict actual performance?

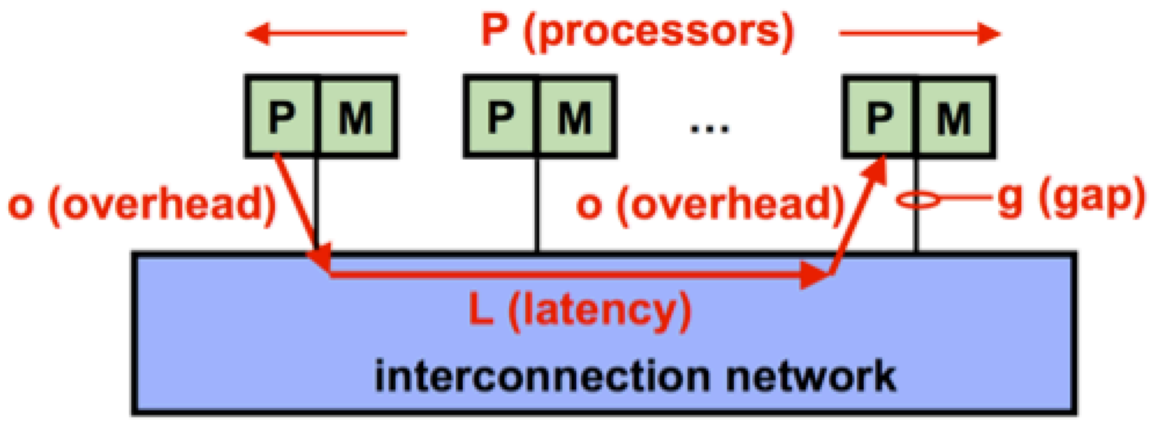
Model Time Varying Message Size & Machines



Measured Message Time



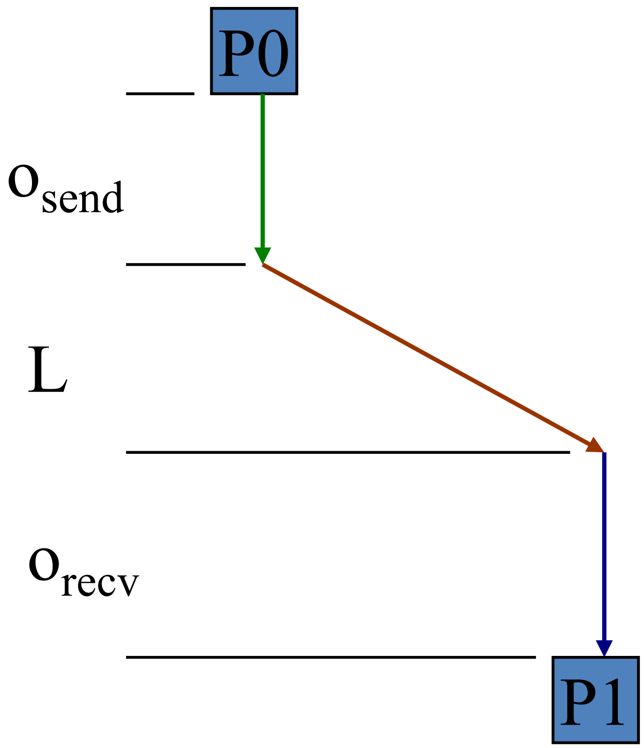
LogP Model



- 4 performance parameters
  - L: latency experienced in each communication event
    - time to communicate word or small # of words
  - o: send/recv overhead experienced by processor
    - time processor fully engaged in transmission or reception
  - g: gap between successive sends or recvs by a processor
    - 1/g = communication bandwidth
  - P: number of processor/memory modules

LogP Parameters: Overhead & Latency

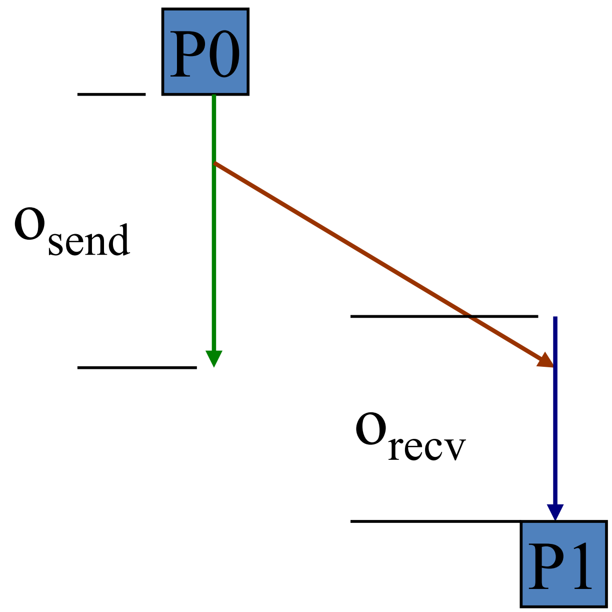
- Non-overlapping overhead



EEL = End-to-End Latency

$$= o_{send} + L + o_{recv}$$

- Send and recv overhead can overlap

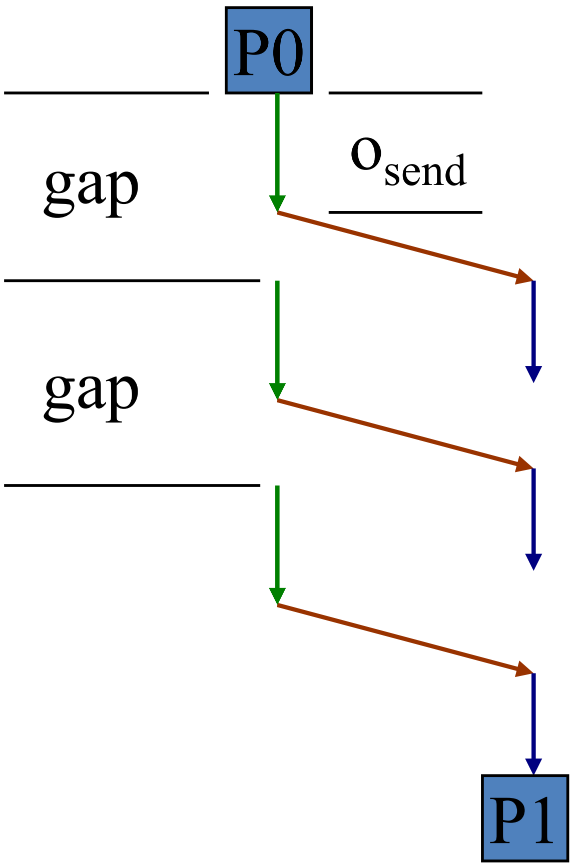


$$\text{EEL} = f(o_{send}, L, o_{recv})$$

$$\geq \max(o_{send}, L, o_{recv})$$

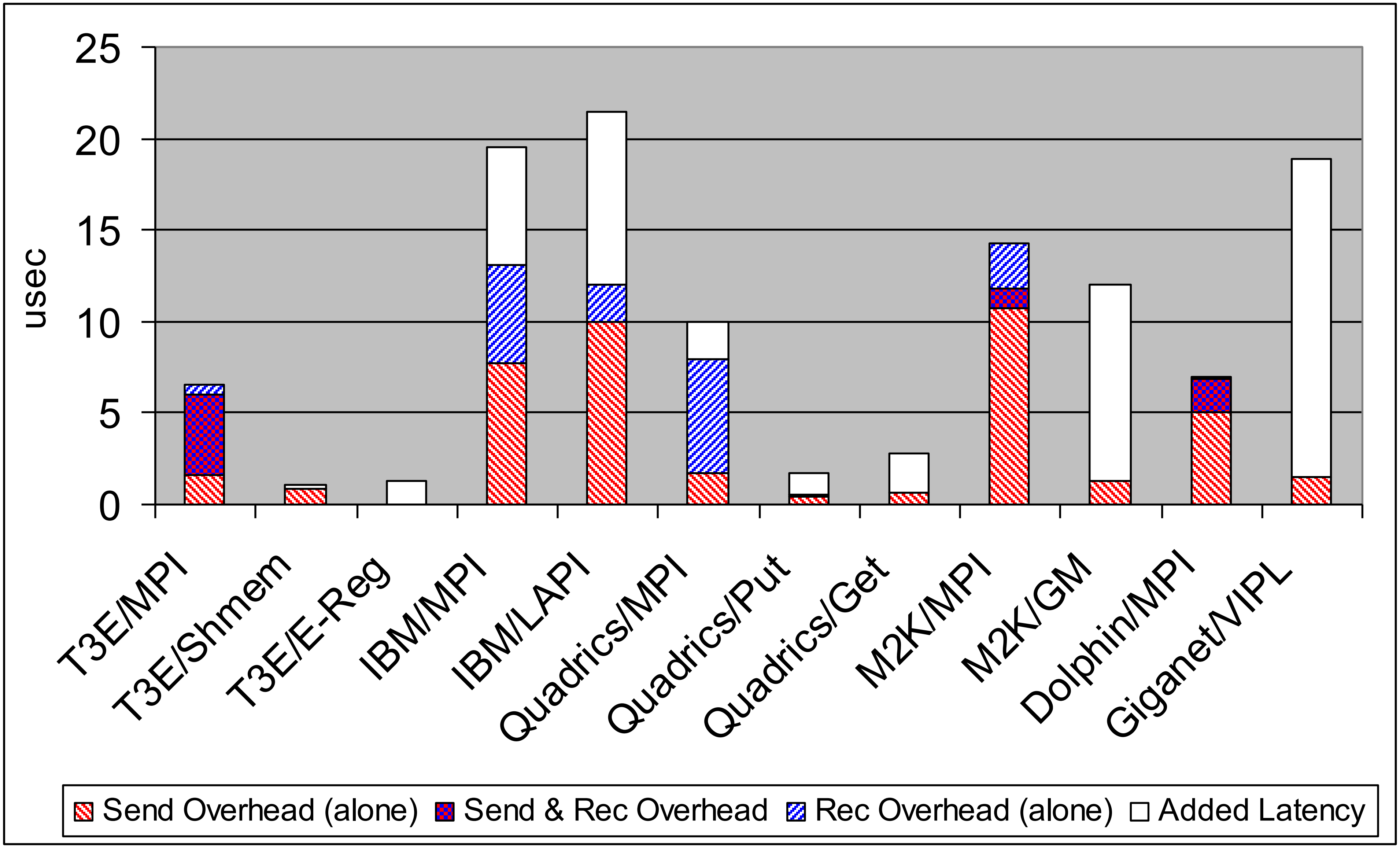
LogP Parameters: gap

- The Gap is the delay between sending messages
- Gap could be greater than send overhead
  - NIC may be busy finishing the processing of last message and cannot accept a new one.
  - Flow control or backpressure on the network may prevent the NIC from accepting the next message to send.
- No overlap $\Rightarrow$ time to send n messages (pipelined) =



$$(o_{send} + L + o_{recv} - gap) + n*gap = \alpha + n*\beta$$

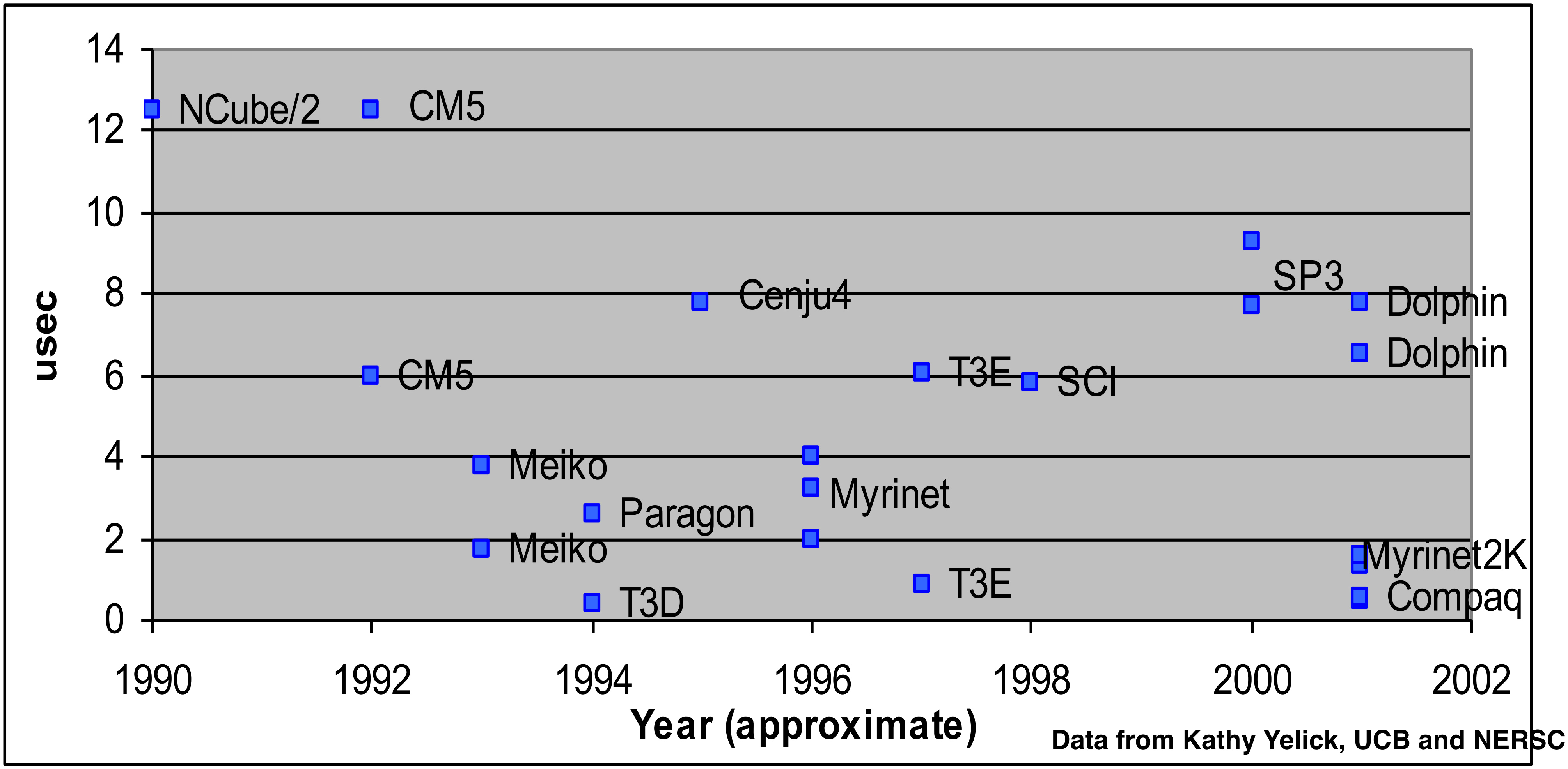
Results: EEL and Overhead



Data from Mike Welcome, NERSC

Send Overhead Over Time

- Overhead has not improved significantly; T3D was best
  - Lack of integration; lack of attention in software



Limitations of the LogP Model

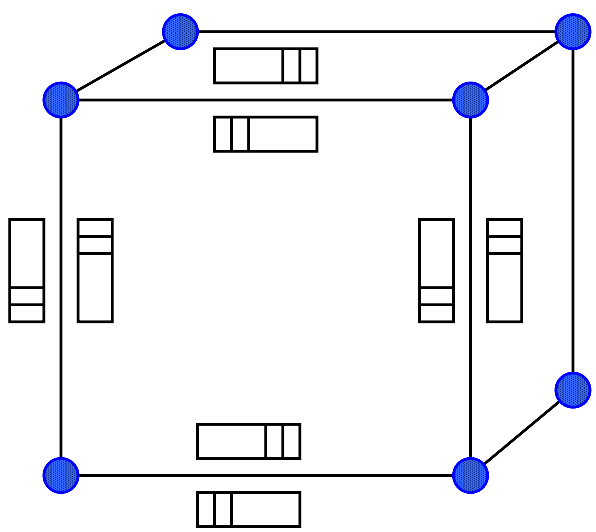
- The LogP model has a fixed cost for each message
  - This is useful in showing how to quickly broadcast a single word
  - Other examples also in the LogP papers
- For larger messages, there is a variation LogGP
  - Two gap parameters, one for small and one for large messages
  - The large message gap is the b in our previous model
- No topology considerations (including no limits for bisection bandwidth)
  - Assumes a fully connected network
  - OK for some algorithms with nearest neighbor communication, but with "all-to-all" communication we need to refine this further
- This is a flat model, i.e., each processor is connected to the network
  - Clusters of multicores are not accurately modeled

Summary

- Latency and bandwidth are two important network metrics
  - Latency matters more for small messages than bandwidth
  - Bandwidth matters more for large messages than bandwidth
  - **Time = $\alpha + n*\beta$**
- Communication has overhead from both sending and receiving end
  - **EEL = End-to-End Latency = $o_{send} + L + o_{recv}$**
- Multiple communication can overlap

Historical Perspective

- Early distributed memory machines were:
  - Collection of microprocessors.
  - Communication was performed using bi-directional queues between nearest neighbors.
- Messages were forwarded by processors on path.
  - "Store and forward" networking
- There was a strong emphasis on topology in algorithms, in order to minimize the number of hops = minimize time



Evolution of Distributed Memory Machines

- Special queue connections are being replaced by direct memory access (DMA):
  - Processor packs or copies messages.
  - Initiates transfer, goes on computing.
- Wormhole routing in hardware:
  - Special message processors do not interrupt main processors along path.
  - Long message sends are pipelined.
  - Processors don't wait for complete message before forwarding
- Message passing libraries provide store-and-forward abstraction:
  - Can send/receive between any pair of nodes, not just along one wire.
  - Time depends on distance since each processor along path must participate.

Outline

- Cluster Introduction
- Distributed Memory Architectures
  - Properties of communication networks
  - Topologies
  - Performance models
- **Programming Distributed Memory Machines using Message Passing**
  - **Overview of MPI**
  - **Basic send/receive use**
  - **Non-blocking communication**
  - **Collectives**