

CS 267: Distributed Memory Computers

Kathy Yelick

<http://www.cs.berkeley.edu/~yelick/cs267>

2/11/2004

CS267 Lecture 7

1

Recap of Last Lectures

- Shared memory multiprocessors
 - Caches in individual processors must be kept coherent -- multiple cached copies of same location must be kept equal.
 - Requires clever hardware (see CS258).
 - Distant memory much more expensive to access.
- Shared memory programming
 - Starting, stopping threads.
 - Synchronization with barriers, locks.
- Distributed memory programming
 - MPI (message passing interface)
 - Lecture Wednesday by Bill Saphir from LBNL

2/11/2004

CS267 Lecture 7

2

Outline

- Distributed Memory Architectures
 - Topologies
 - Cost models
- Trends in High End Machines
 - Clusters today
 - Top500 data

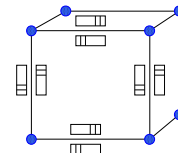
2/11/2004

CS267 Lecture 7

3

Historical Perspective

- Early 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.



2/11/2004

CS267 Lecture 7

4

Network Analogy

- To have a large number of transfers occurring at once, you need a large number of distinct wires.
- 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.

2/11/2004

CS267 Lecture 7

5

Characteristics of a Network

- Topology (how things are connected)
 - Crossbar, ring, 2-D and 2-D torus, hypercube, omega network.
- Routing algorithm:
 - Example: 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.

2/11/2004

CS267 Lecture 7

6

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:
 - Hardware/software latencies often differ by 1-2 orders of magnitude
 - Maximum hardware latency varies with diameter, but the variation in software latency is usually negligible
- Latency is important for programs with many small messages

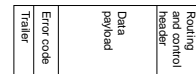
2/11/2004

CS267 Lecture 7

7

Properties of a Network: Bandwidth

- A network is **partitioned** into two or more disjoint sub-graphs if some nodes cannot reach others.
- The **bandwidth** of a link = $w * 1/t$
 - w is the number of wires
 - t is the time per bit
- Bandwidth typically in Gigabytes (GB), i.e., $8 * 2^{20}$ bits
- **Effective bandwidth** is usually lower than physical link bandwidth due to packet overhead.



- Bandwidth is important for applications with mostly large messages

2/11/2004

CS267 Lecture 7

8

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

2/11/2004

CS267 Lecture 7

9

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.

2/11/2004

CS267 Lecture 7

10

Linear and Ring Topologies

- Linear array
 - Diameter = $n-1$; average distance $\sim n/3$.
 - Bisection bandwidth = 1 (units are link bandwidth).
- Torus or Ring
 - Diameter = $n/2$; average distance $\sim n/4$.
 - Bisection bandwidth = 2.
 - Natural for algorithms that work with 1D arrays.

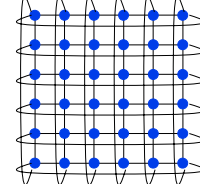
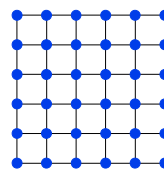
2/11/2004

CS267 Lecture 7

11

Meshes and Tori

- Two dimensional mesh
 - Diameter = $2 * (\text{sqrt}(n) - 1)$
 - Bisection bandwidth = $\text{sqrt}(n)$
- Two dimensional torus
 - Diameter = $\text{sqrt}(n)$
 - Bisection bandwidth = $2 * \text{sqrt}(n)$



- Generalizes to higher dimensions (Cray T3D used 3D Torus).
- Natural for algorithms that work with 2D and/or 3D arrays.

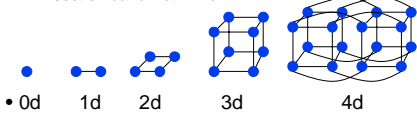
2/11/2004

CS267 Lecture 7

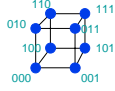
12

Hypercubes

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



- Popular in early machines (Intel iPSC, NCUBE).
 - Lots of clever algorithms.
 - See 1996 online 267 notes.
- Greycodes addressing:
 - Each node connected to d others with 1 bit different.



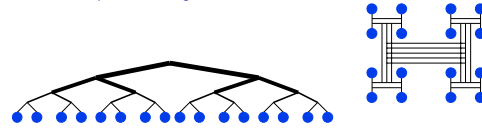
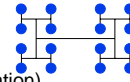
2/11/2004

CS267 Lecture 7

13

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.



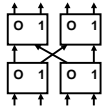
2/11/2004

CS267 Lecture 7

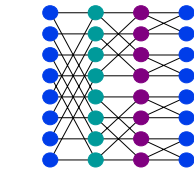
14

Butterflies

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



butterfly switch



multistage butterfly network

2/11/2004

CS267 Lecture 7

15

Topologies in Real Machines

Red Storm (Opteron + Cray network, future)	3D Mesh
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

older
newer

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

2/11/2004

CS267 Lecture 7

16

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.
- Message passing libraries provide store-and-forward abstraction:
 - Can send/receive between any pair of nodes, not just along one wire.
 - Time proportional to distance since each processor along path must participate.
- Wormhole routing in hardware:
 - Special message processors do not interrupt main processors along path.
 - Message sends are pipelined.
 - Processors don't wait for complete message before forwarding.

2/11/2004

CS267 Lecture 7

17

Performance Models

2/11/2004

CS267 Lecture 7

18

PRAM

- Parallel Random Access Memory.
- All memory access operations complete in one clock period -- no concept of memory hierarchy ("too good to be true").
 - OK for understanding whether an algorithm has enough parallelism at all.
 - Parallel algorithm design strategy: first do a PRAM algorithm, then worry about memory/communication time (sometimes works)
- Slightly more realistic: Concurrent Read Exclusive Write (CREW) PRAM.

2/11/2004

CS267 Lecture 7

19

Latency and Bandwidth Model

- Time to send message of length n is roughly.

$$\text{Time} = \text{latency} + n \cdot \text{cost_per_word} \\ = \text{latency} + n/\text{bandwidth}$$

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

$$\text{Time} = \alpha + n \cdot \beta$$
 - Usually $\alpha \gg \beta \gg$ time per flop.
 - One long message is cheaper than many short ones.

$$\alpha + n \cdot \beta \ll n \cdot (\alpha + 1 \cdot \beta)$$
 - Can do hundreds or thousands of flops for cost of one message.
- Lesson: Need large computation-to-communication ratio to be efficient.

2/11/2004

CS267 Lecture 7

20

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
Gigaset/VIPL	3.0	0.010
GigE/VIPL	4.6	0.008
GigE/MPI	5.854	0.00872

α is in usecs
 β is usecs per Byte

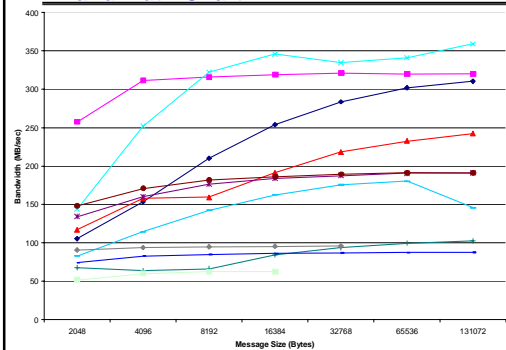
How well does the model
 $\text{Time} = \alpha + n \cdot \beta$
predict actual performance?

2/11/2004

CS267 Lecture 7

21

Bandwidth Chart

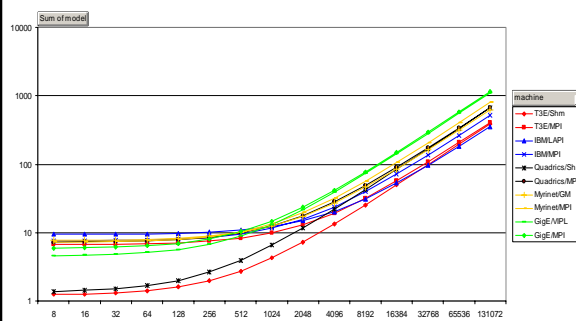


2/11/2004

CS267 Lecture 7

Data from Mike Welc...

Model Time Varying Message Size & Machines

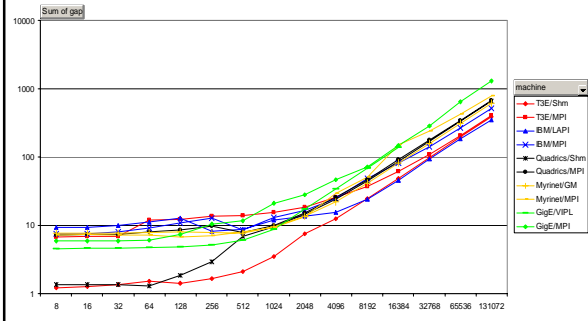


2/11/2004

CS267 Lecture 7

23

Measured Message Time



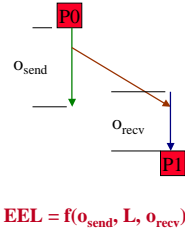
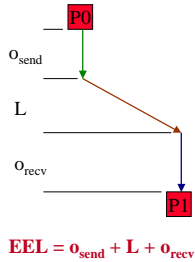
2/11/2004

CS267 Lecture 7

24

LogP Parameters: Overhead & Latency

- Non-overlapping overhead
- Send and recv overhead can overlap



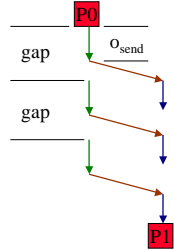
2/11/2004

CS267 Lecture 7

25

LogP Parameters: gap

- The Gap is the delay between sending messages
- Gap could be larger than send ovhd
 - 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.
- The gap represents the inverse bandwidth of the network for small message sends.

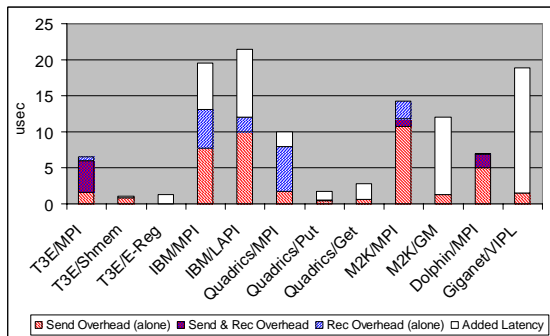


2/11/2004

CS267 Lecture 7

26

Results: EEL and Overhead



2/11/2004

CS267 Lecture 7

Data from Mike Welcome, NERSC

Limitations of the LogP Model

- The LogP model has a fixed cost for each messages
 - This is useful in showing how to quick 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 message
 - The large message gap is the β in our previous model
- No topology considerations (including no limits for bisection bandwidth)
 - Assumes a fully connected network
 - 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 SMPs are not accurately models

2/11/2004

CS267 Lecture 7

28

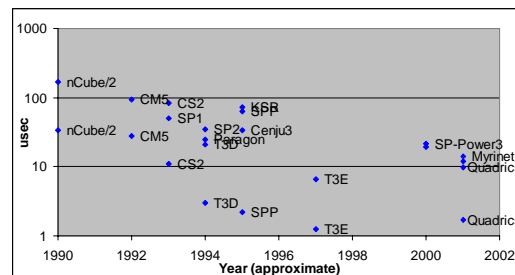
Trends in Real Machines

2/11/2004

CS267 Lecture 7

29

End to End Latency Over Time



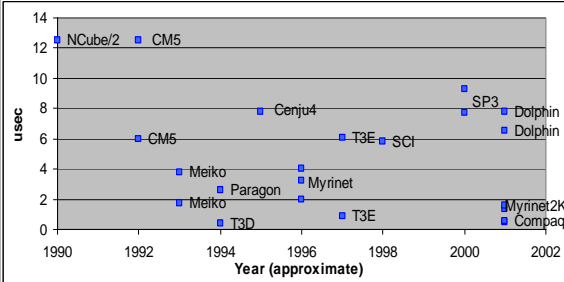
- Latency has not improved significantly
 - T3E (shmem) was lowest point
 - Federation in 2003 will not reach that level – 7 years later!

2/11/2004

CS267 Lecture 7

Data from Kathy Yelick, UCB and NERSC

Send Overhead Over Time



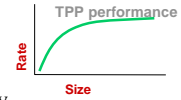
- Overhead has not improved significantly; T3D was best
 - Lack of integration; lack of attention in software

2/11/2004 CS267 Lecture 7 Data from Kathy Yelick, UCB and NERSC

TOP500

- Listing of the 500 most powerful Computers in the World
- Yardstick: R_{max} from Linpack

$$Ax=b, \text{ dense problem}$$



- Updated twice a year:
 - ISC'xy in Germany, June xy
 - SC'xy in USA, November xy

- All data available from www.top500.org

2/11/2004 CS267 Lecture 7 32

TOP500 list - Data shown

- Manufacturer** Manufacturer or vendor
- Computer Type** indicated by manufacturer or vendor
- Installation Site** Customer
- Location** Location and country
- Year** Year of installation/last major update
- Customer Segment** Academic, Research, Industry, Vendor, Class.
- # Processors** Maximal LINPACK performance achieved
- R_{max}** Theoretical peak performance
- R_{peak}** Problemsize for achieving R_{max}
- N_{max}** Problemsize for achieving half of R_{max}
- $N_{1/2}$** Position within the TOP500 ranking
- N_{world}**

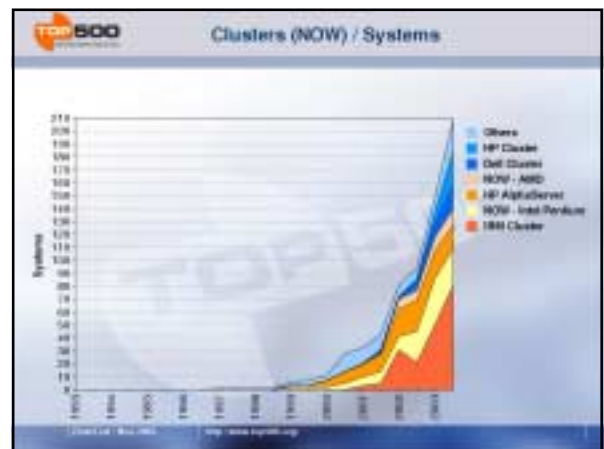
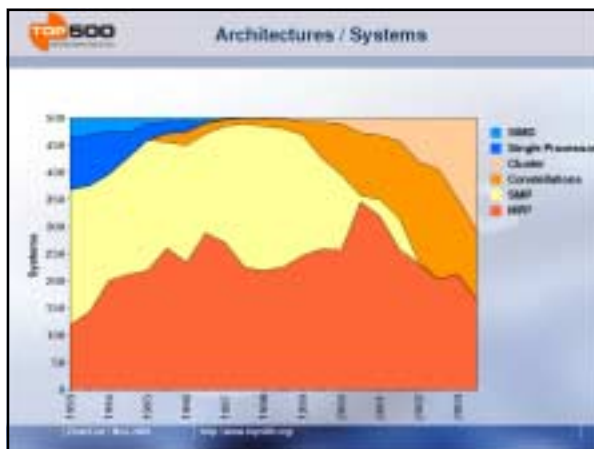
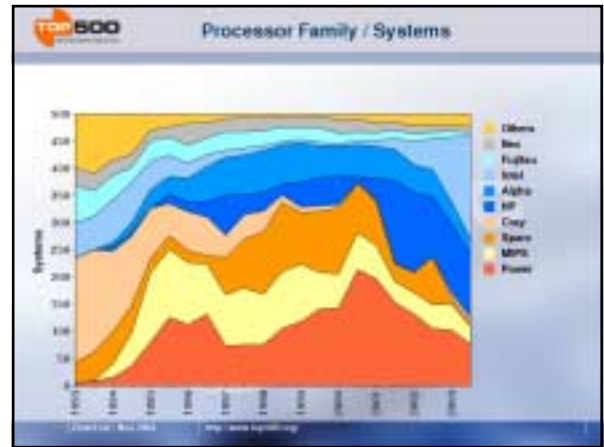
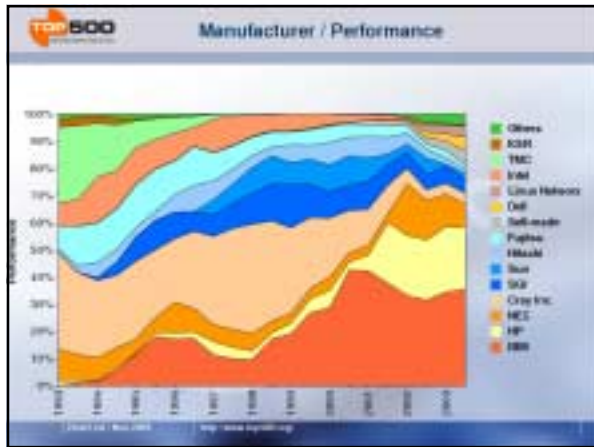
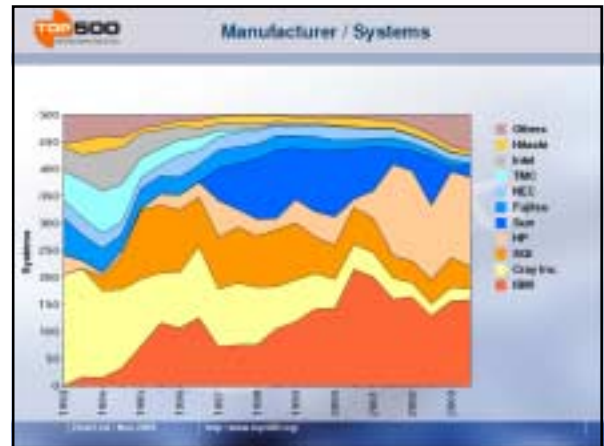
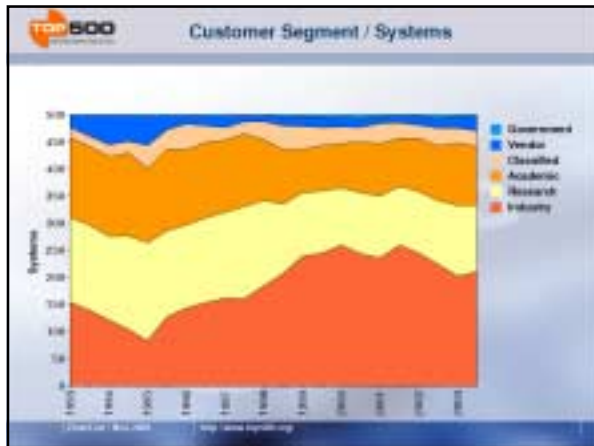
2/11/2004 CS267 Lecture 7 33

22nd List: The TOP10

Rank	Manufacturer	Computer	R_{max} [TFs]	Installation Site	Country	Year	Area of Installation	# Proc
1	NEC	Earth-Simulator	35.86	Earth Simulator Center	Japan	2002	Research	5120
2	HP	ASCI Q AlphaServer SC	13.88	Los Alamos National Laboratory	USA	2002	Research	8192
3	Self-Made	X Apple G5, Mellanox	10.28	Virginia Tech	USA	2003	Academic	2200
4	Dell	Tungsten PowerEdge, Myrinet	9.82	NCSA	USA	2003	Academic	2500
5	HP	Mpx2, Integrity rx2600 Itanium2, Quadrics	8.63	Pacific Northwest National Laboratory	USA	2003	Research	1936
6	Linux Network	Lightning, Opteron, Myrinet	8.05	Los Alamos National Laboratory	USA	2003	Research	2816
7	Linux Network/Quadrics	MCR Cluster	7.63	Lawrence Livermore National Laboratory	USA	2002	Research	2304
8	IBM	ASCI White SP Power3	7.3	Lawrence Livermore National Laboratory	USA	2000	Research	8192
9	IBM	Seaborg SP Power 3	7.3	NERSC Lawrence Berkeley Nat. Lab.	USA	2002	Research	6656
10	IBM/Quadrics	sSeries Cluster Xeon 2.4 GHz	6.59	Lawrence Livermore National Laboratory	USA	2003	Research	1920

2/11/2004 CS267 Lecture 7 34





Analysis of TOP500 Data

- Annual performance growth about a factor of 1.82
- Two factors contribute almost equally to the annual total performance growth
- Processor number grows per year on the average by a factor of 1.30 and the
- Processor performance grows by 1.40 compared to 1.58 of Moore's Law

Strohmaier, Dongarra, Meuer, and Simon, Parallel Computing 25, 1999, pp 1517-1544.

2/11/2004

CS267 Lecture 7

43

Limits to Cluster Based Systems for HPC

- Memory Bandwidth
 - Commodity memory interfaces [SDRAM, RDRAM, DDRAM]
 - Separation of memory and CPU implementations limits performance
- Communications fabric/CPU/Memory Integration
 - Current networks are attached via I/O devices
 - Limits bandwidth and latency and communication semantics
- Node and system packaging density
 - Commodity components and cooling technologies limit densities
 - Blade based servers moving in right direction but are not High Performance
- Ad Hoc Large-scale Systems Architecture
 - Little functionality for RAS
 - Lack of systems software for production environment
- ... but departmental and single applications clusters will be highly successful

2/11/2004

CS267 Lecture 7

After Rick Stevens, Atgonne

Comparison Between Architectures (2001)

	Alvarez	Seaborg	Mcurie
Processor	Pentium III	Power 3	EV-5
Clock speed	867	375	450
# nodes	80	184	644
# processors/node	2	16	
Peak (GF/s)	139	4416	579.6
Memory (GB/node)	1	16-64	0.256
Interconnect	Myrinet 2000	Colony	T3E
Disk (TB)	1.5	20	2.5

Source: Tammy Welcome, NERSC

2/11/2004

CS267 Lecture 7

45

Performance Comparison(2) Class C NPBs

	Alvarez		Seaborg		Mcurie	
	64	128	64	128	64	128
BT	61.0		111.9		55.7	
CG	17.1	13.9	34.0	30.9	9.3	11.8
EP	3.9	3.9	3.9	3.9	2.6	2.6
FT	31.3	20.0	61.2	54.6	30.8	30.1
IS	2.4	2.1	2.1	1.3	1.1	1.0
LU	26.9	38.7	209.0	133.7	60.4	56.0
MG	56.6	46.9	133.2	101.7	93.9	80.0
SP	40.9		100.7		41.8	
per processor	39.0		108.3		48.7	
SSP (Gflops/s)	6.2		318.9		31.3	

Source: Tammy Welcome, NERSC

2/11/2004

CS267 Lecture 7

46

Summary – Wrap-up

- Network structure and concepts
 - Switching, routing, flow control
 - Topology, bandwidth, latency, bisection bandwidth, diameter
- Performance models
 - PRAM, $\alpha - \beta$, and LogP
- Workstation/PC clusters
 - Programming environment, hardware
 - Challenges
- Message passing implementation

2/11/2004

CS267 Lecture 7

47

Extra
Slides

2/11/2004

CS267 Lecture 7

48

Effectiveness of Commodity PC Clusters

- Dollars/performance based on peak
 - SP and Alvarez are comparable \$/TF
- Get lower % of peak on Alvarez than SP
 - Based on SSP, 4.5% versus 7.2% for FP intensive applications
 - Based on sequential NPBs, 5-13.8% versus 6.3-21.6% for FP intensive applications
 - x86 known not to perform well on FP intensive applications
- \$/Performance and cost of ownership need to be examined much more closely
 - Above numbers do not take into account differences in system balance or configuration
 - SP was aggressively priced
 - Alvarez was vendor-integrated, not self-integrated

2/11/2004

CS267 Lecture 7 Source: Tammy Welcome, NERSC 49

Message Passing Libraries

- Many "message passing libraries" available
 - Chameleon, from ANL.
 - CMMD, from Thinking Machines.
 - Express, commercial.
 - MPL, native library on IBM SP-2.
 - NX, native library on Intel Paragon.
 - Zipcode, from LLL.
 - PVM, Parallel Virtual Machine, public, from ORNL/UTK.
 - Others...
 - **MPI, Message Passing Interface, now the industry standard.**
- Need standards to write portable code.
- Rest of this discussion independent of which library.
- Lecture 7 was detailed MPI lecture

2/11/2004

CS267 Lecture 7

50

Workstation/PC Clusters

- Reaction to commercial MPPs:
 - build parallel machines out of commodity components
 - Inexpensive workstations or PCs as computing nodes
 - Fast (gigabit) switched network between nodes
- Benefits:
 - 10x - 100x cheaper for comparable performance
 - Standard OS on each node
 - Follow commodity tech trends
 - Incrementally upgradable and scalable
 - Fault tolerance
- Trends:
 - Berkeley NOW (1994): 100 UltraSPARCs, Myrinet
 - ASCI RED (1997): 4510 dual Pentium II nodes, custom network
 - Millennium (1999): 100+ dual/quad Pentium IIIs, Myrinet
 - Google (2001): 8000+ node Linux cluster, ??? network



2/11/2004

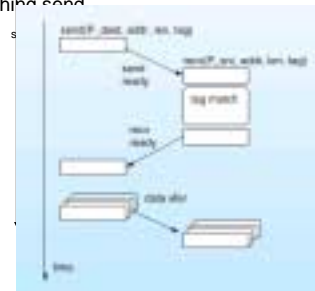
CS267 Lecture 7

51

Implementing Synchronous Message Passing

- Send operations complete after matching receive and source data has been sent.
- Receive operations complete after data transfer is complete from matching send.

- 1) Initiate send
- 2) Address translation on P_{dest}
- 3) Send-Ready Request
- 4) Remote check for posted receive
- 5) Reply transaction
- 6) Bulk data transfer



2/11/2004

Example: Permuting Data

- Exchanging data between Procs 0 and 1, V.1: What goes wrong?

Processor 0	Processor 1
send(1, item0, 1, tag1)	send(0, item1, 1, tag2)
recv(1, item1, 1, tag2)	recv(0, item0, 1, tag1)

- **Deadlock**

- Exchanging data between Proc 0 and 1, V.2:

Processor 0	Processor 1
send(1, item0, 1, tag1)	recv(0, item0, 1, tag1)
recv(1, item1, 1, tag2)	send(0, item1, 1, tag2)

- What about a general permutation, where Proc j wants to send to Proc $s(j)$, where $s(1), s(2), \dots, s(P)$ is a permutation of $1, 2, \dots, P$?

2/11/2004

CS267 Lecture 7

53

Implementing Asynchronous Message Passing

- Optimistic single-phase protocol assumes the destination can buffer data on demand.

- 1) Initiate send
- 2) Address translation on P_{dest}
- 3) Send Data Request
- 4) Remote check for posted receive
- 5) Allocate buffer (if check failed)
- 6) Bulk data transfer



2/11/2004

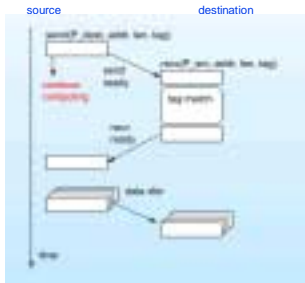
CS267 Lecture 7

54

Safe Asynchronous Message Passing

- Use 3-phase protocol
- Buffer on sending side
- Variations on send completion
 - wait until data copied from user to system buffer
 - don't wait -- let the user beware of modifying data

- 1) Initiate send
- 2) Address translation on P_{dest}
- 3) Send-Ready Request
- 4) Remote check for posted receive
record send-rdy
- 5) Reply transaction
- 6) Bulk data transfer



2/11/2004

Example Revisited: Permuting Data

- Processor j sends item to Processor $s(j)$, where $s(1), \dots, s(P)$ is a permutation of $1, \dots, P$

```
Processor j
send_asynch(s(j), item, 1, tag)
recv_block(ANY, item, 1, tag)
```

- What could go wrong?
- Need to understand semantics of send and receive.
- Many flavors available.

2/11/2004

CS267 Lecture 7

56

Other operations besides send/receive

- "Collective Communication" (more than 2 procs)
 - Broadcast data from one processor to all others.
 - Barrier.
 - Reductions (sum, product, max, min, boolean and, #, ...), where # is any "associative" operation.
 - Scatter/Gather.
 - Parallel prefix -- Proc j owns $x(j)$ and computes $y(j) = x(1) \# x(2) \# \dots \# x(j)$.
 - Can apply to all other processors, or a user-define subset.
 - Cost = $O(\log P)$ using a tree.
- Status operations
 - Enquire about/Wait for asynchronous send/receives to complete.
 - How many processors are there?
 - What is my processor number?

2/11/2004

CS267 Lecture 7

57

Example: Sharks and Fish

- N fish on P procs, N/P fish per processor
 - At each time step, compute forces on fish and move them
- Need to compute gravitational interaction
 - In usual n^2 algorithm, every fish depends on every other fish.
 - Every fish needs to "visit" every processor, even if it "lives" on just one.
- What is the cost?

2/11/2004

CS267 Lecture 7

58

Two Algorithms for Gravity: What are their costs

Algorithm 1

```
Copy local Fish array of length N/P to Tmp array
for j = 1 to N
  for k = 1 to N/P, Compute force of Tmp(k) on Fish(k)
  "Rotate" Tmp by 1
  for k=2 to N/P, Tmp(k) <= Tmp(k-1)
    recv(my_proc - 1, Tmp(1))
    send(my_proc+1, Tmp(N/P))
```

Algorithm 2

```
Copy local Fish array of length N/P to Tmp array
for j = 1 to P
  for k=1 to N/P, for m=1 to N/P, Compute force of Tmp(k) on Fish(m)
  "Rotate" Tmp by N/P
  recv(my_proc - 1, Tmp(1:N/P))
  send(my_proc+1, Tmp(1:N/P))
```

What could go wrong? (be careful of overwriting Tmp)

2/11/2004

CS267 Lecture 7

59

More Algorithms for Gravity

- Algorithm 3 (in sharks and fish code):
 - All processors send their Fish to Proc 0.
 - Proc 0 broadcasts all Fish to all processors.
- Tree-algorithms:
 - Barnes-Hut, Greengard-Rokhlin, Anderson.
 - $O(N \log N)$ instead of $O(N^2)$.
 - Parallelizable with cleverness.
 - "Just" an approximation, but as accurate as you like (often only a few digits are needed, so why pay for more).
 - Same idea works for other problems where effects of distant objects becomes "smooth" or "compressible":
 - electrostatics, vorticity, ...
 - radiosity in graphics.
 - anything satisfying Poisson equation or something like it.

2/11/2004

CS267 Lecture 7

60

Reading Assignment

- Reading for today
- Next week: Current high performance architectures
 - MPI
- The following week
 - UPC