A Case for Networks of Workstations (NOW)

Tom Anderson, David Culler, Dave Patterson et al

Computer Science Division

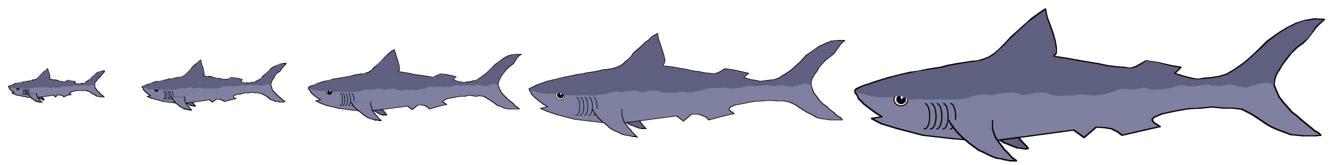
EECS Department

University of California, Berkeley

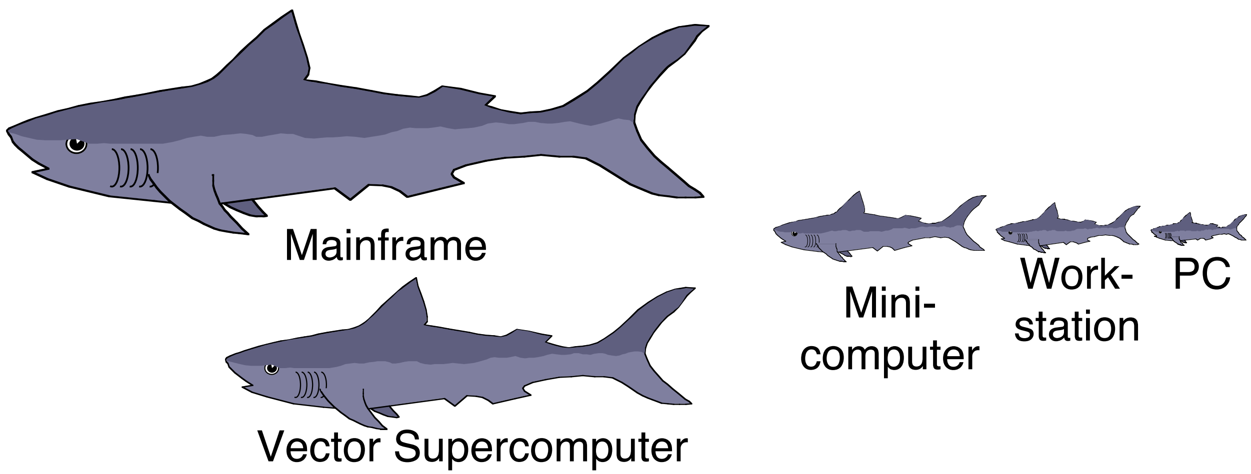
Outline

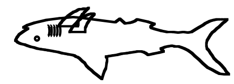
- Background: Evolution of Computer Industry
- Opportunity for Large Scale Computing on NOW
- Why NOW now?
- The NOW Project at Berkeley
- Issues and Potential Solutions
  - Time Lag for NOW using fastest workstations
  - Network Overhead
  - Preserving Response Time for large and small jobs
  - I/O Bottleneck
  - NOW helps only parallel jobs?
- Conclusion

Original Food Chain Picture

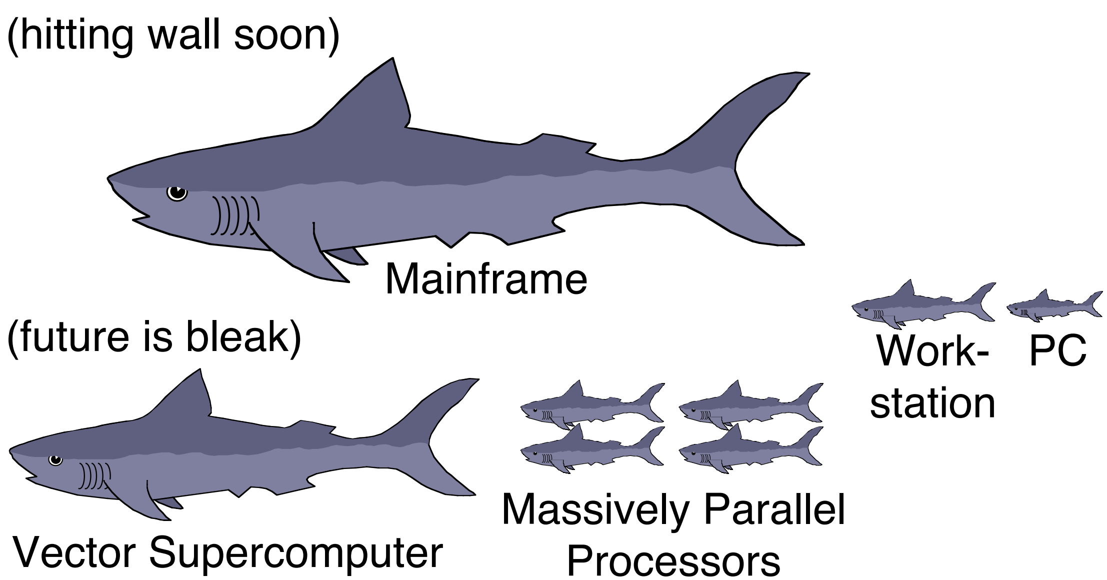


1984 Computer Food Chain





1994 Computer Food Chain



MPP: A Near Miss

- "near commodity" μprocs,DRAMs, boards => delayed shipment:

MPP	Proc	Year	=WS
T3D	150 MHz Alpha	'93/'94	'92/'93
Paragon	50 MHz i860	'92/'93	≈ '91
CM-5	32 MHz SS-2	'91/'92	'89/'90

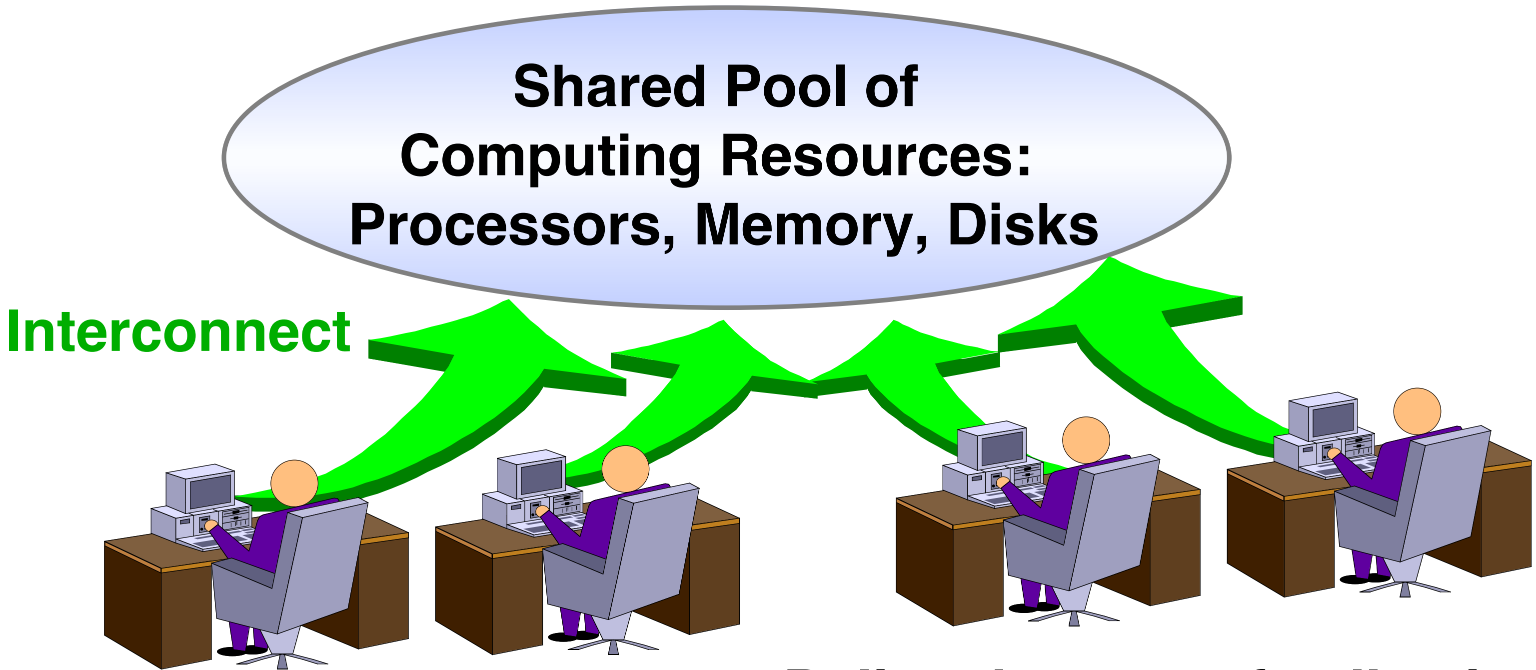
- μproc perf. improves 50% / yr (4%/month)
  - 1 year lag:WS = 1.50 MPP node perf.
  - 2 year lag:WS = 2.25 MPP node perf.
- No economy of scale in 100s => +$
- SW incompatibility (OS & apps) => +$$$$

1990s Building Blocks

- There is no "near commodity" component
- Building block = complete computers (HW & SW) shipped in 100,000s: Killer micro, Killer DRAM, Killer disk, Killer OS, Killer packaging, Killer investment
  - Leverage billion $ per year investment
- Interconnecting Building Blocks => Killer Net
  - High Bandwidth
  - Low latency
  - Reliable
  - Commodity

  (ATM?)

Opportunity of Large-scale Computing on NOW

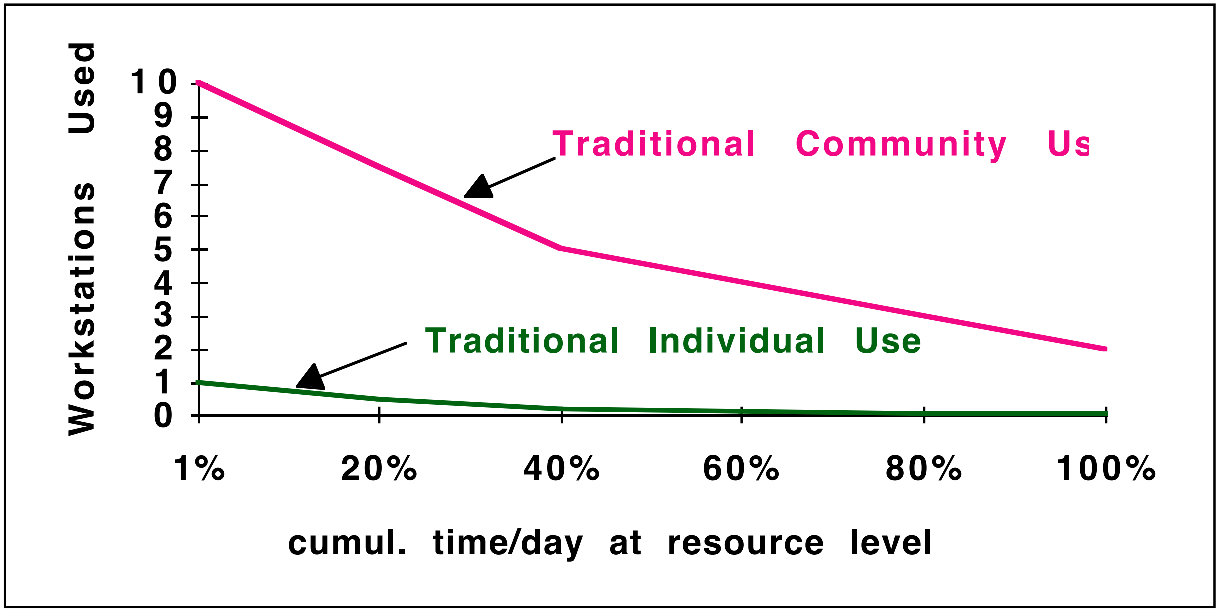


Guarantee at least one workstation to many individuals (when active)

Deliver large % of collective resources to few individuals at any one time

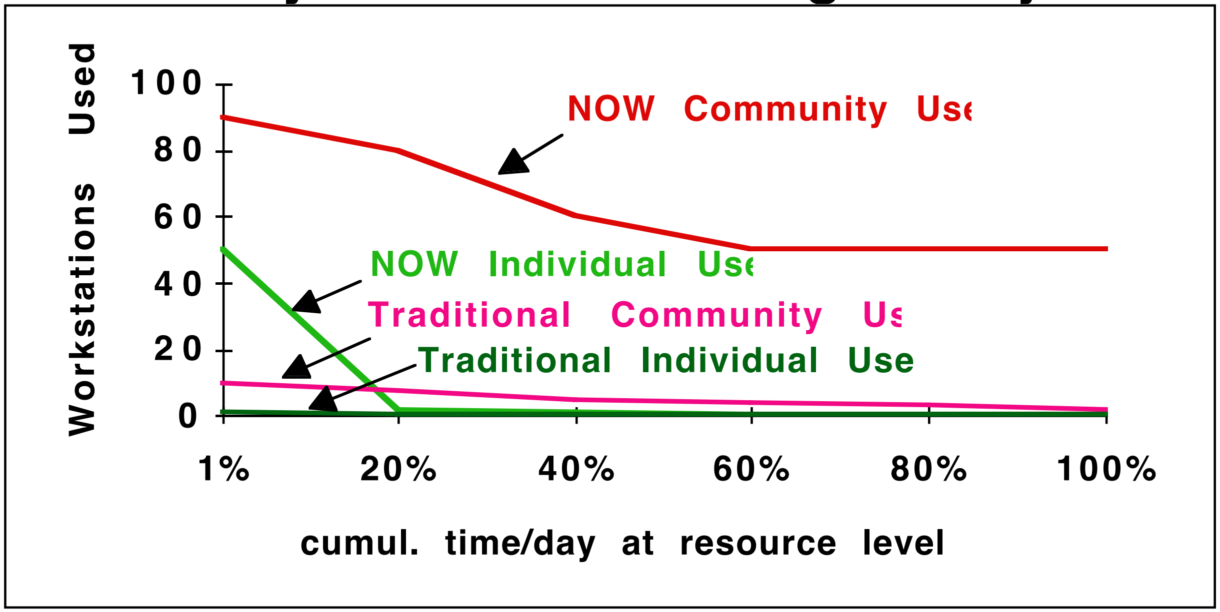
Current Utilization of Resources

- **Out of 100 workstations, how are resources used by individual and by whole community?**



Using Available Resources means Better Performance

- **Higher peak use/person; Higher tail for community due to more background jobs**



Why NOW now?
(Beyond technology and cost)

- **Building block is big enough (v. Intel 8086)**
- **Networks are faster**
  - **Higher link bandwidth (v. 10 Mbit Ethernet)**
  - **Switch based networks coming (ATM)**
  - **Interfaces simple & fast (Active Msgs)**
- **Striped files preferred (RAID)**
- **Demise□ of mainframes, supercomputers, & MPPs**

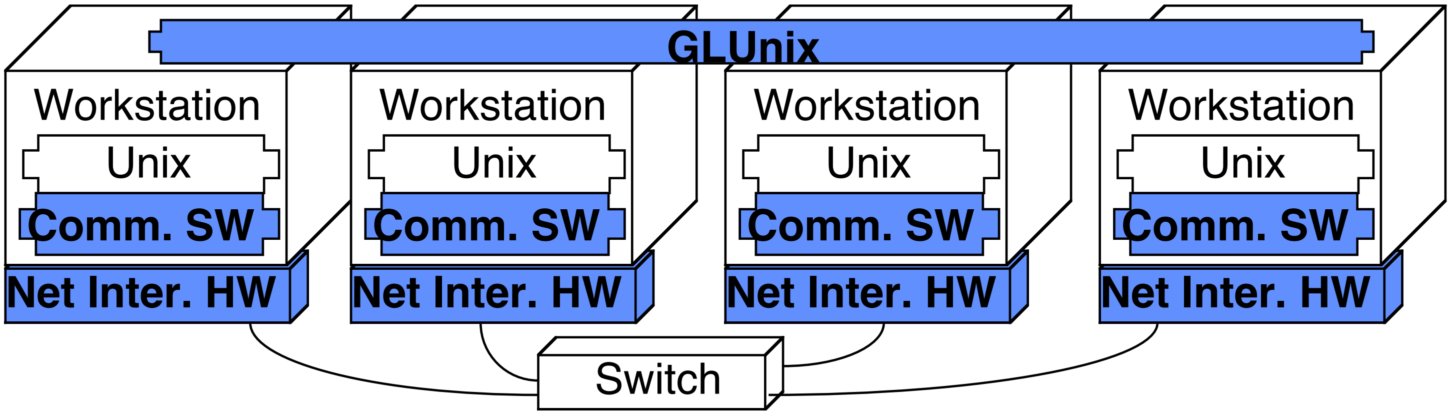
NOW Benefits Parallel Programs: Example MPP Performance

Machine (no. processors)	ODE	Transport	I/O	Total
		(seconds)		
Cray C-90 (16)	7	4	25	38
Intel Paragon (256)	12	24	10	46
RS/6000 (256),Ether	4	23,340	4,030	27,374
+ ATM	4	192	2,015	2,211
+ Parallel FS	4	192	10	206
+ low net. overhead	4	11	10	25

(1 disk/processor, parallel FS for C-90, Paragon)

- Order of importance: ATM bandwidth, Parallel File System, low overhead ATM/SW=> 1000X

NOW @ Berkeley

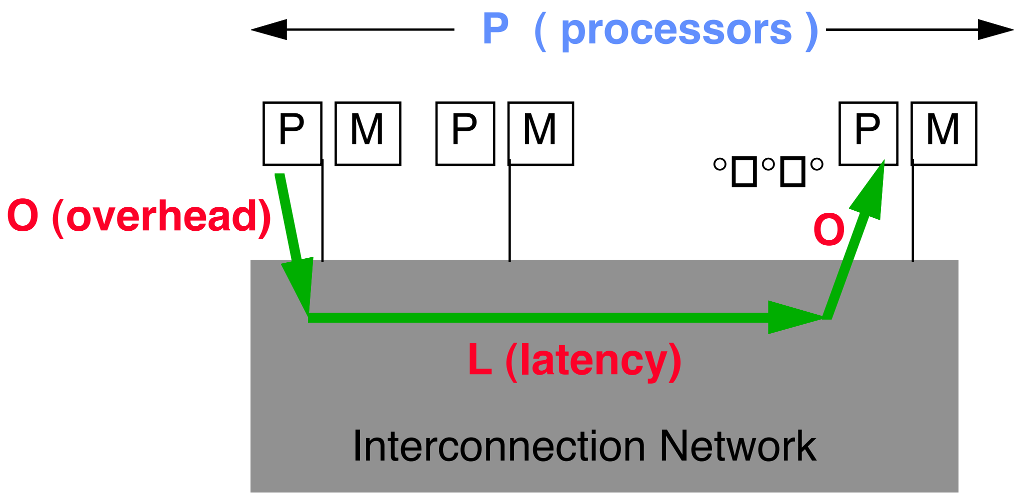


- **Design & Implementation of higher-level system**
  - Global OS (Glunix)
  - Parallel File Systems (xFS)
  - Fast Communication (HW for Active Messages)
  - Application support
- **Overcoming technological shortcomings**
  - Fault tolerance
  - system management
- **NOW Goal: Faster for Parallel AND Sequential**

NOW Issues
and Potential Solutions

- **Network Overhead**
- **Preserving Response Time for large and small jobs**
  - Recruiting□ idle workstations
  - Gang scheduling for parallel tasks
  - Not annoying interactive users
- **I/O Bottleneck**
- **NOW helps only parallel jobs?**
  - NOW File System (xFS): large file cache
  - Network RAM: avoid I/O

Communication Model: Beyond Bandwidth

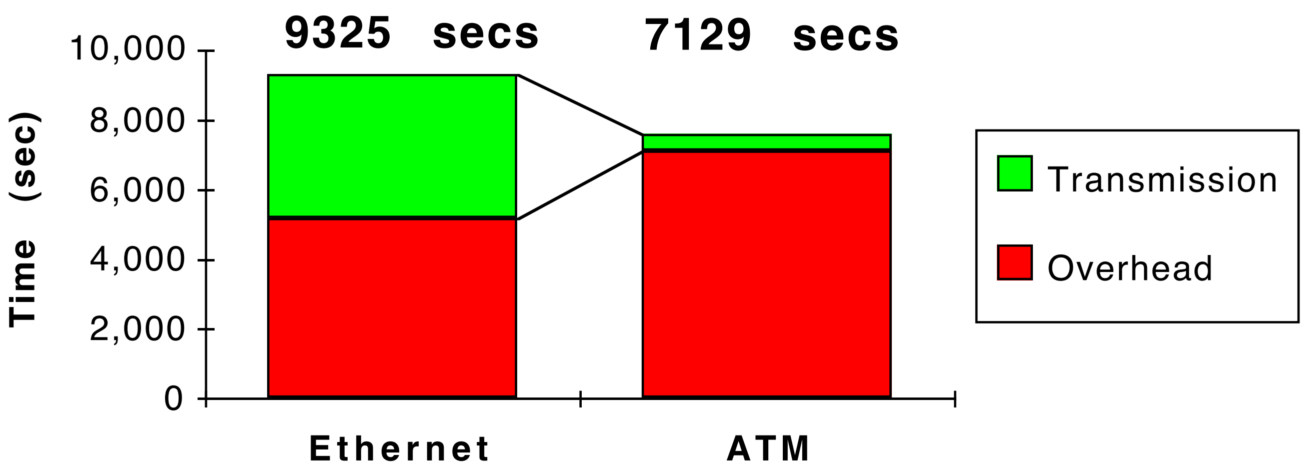


- **L**: **Network latency incurred in sending message between nodes (1-way)**
- **O**: **processor overhead to send or receive a message (1-side)**

Importance of Low Overhead (and Latency)

- **NFS trace over 1 week: 95% msgs < 200 bytes**
- **Ethernet: 9 Mb/s BW, 456 μsecs overhead**
- **ATM Synoptics: 78 Mbit/s BW, 626 μsecs overhead**

Estimated time in overhead & transmission for entire trace:



MPP/LAN Overhead & Latency

- **Bandwidth ≈ MIPS for processors; may be misleading (625 μsec overhead ATM vs. 155 Mb/s BW ATM)**

		Overhead	Latency
MPP	with A.M.	2 μs	5 μs
	w.o. A.M.	25 μs	
LAN	with A.M.	8 μs	5 - 50 μs
	w.o. A.M.	360-625 μs	

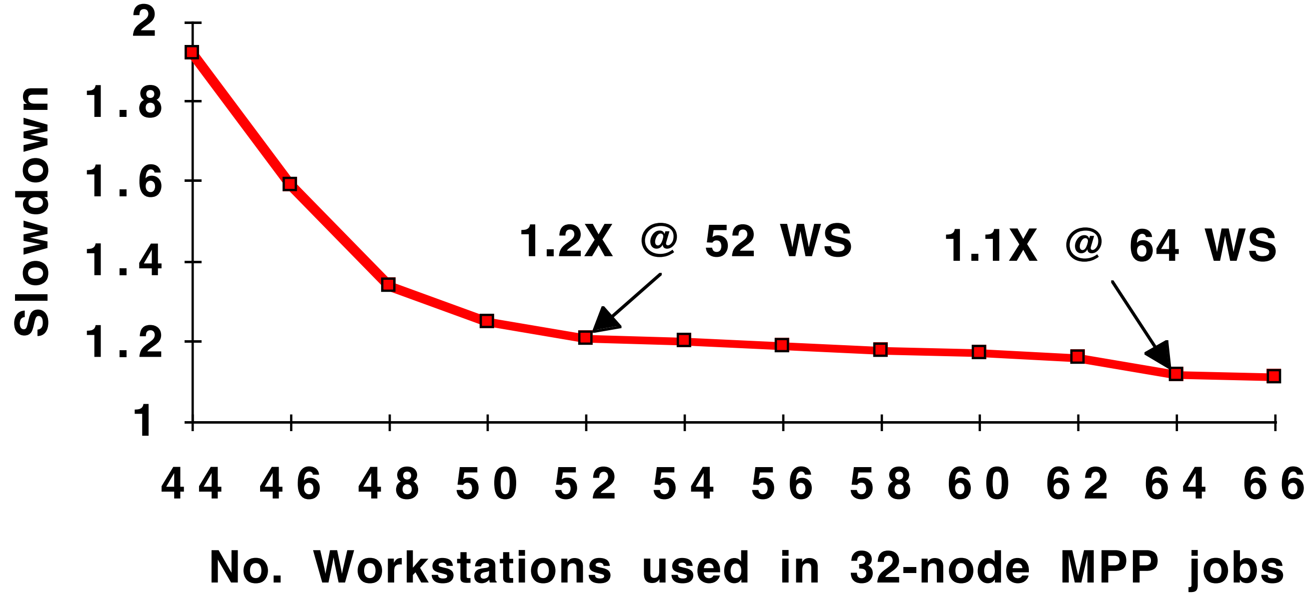
1996 Berkeley Goal: Overhead+Lat ≤ 10 μs for 100 WS

Experiment running MPP workload on NOW running sequential workload

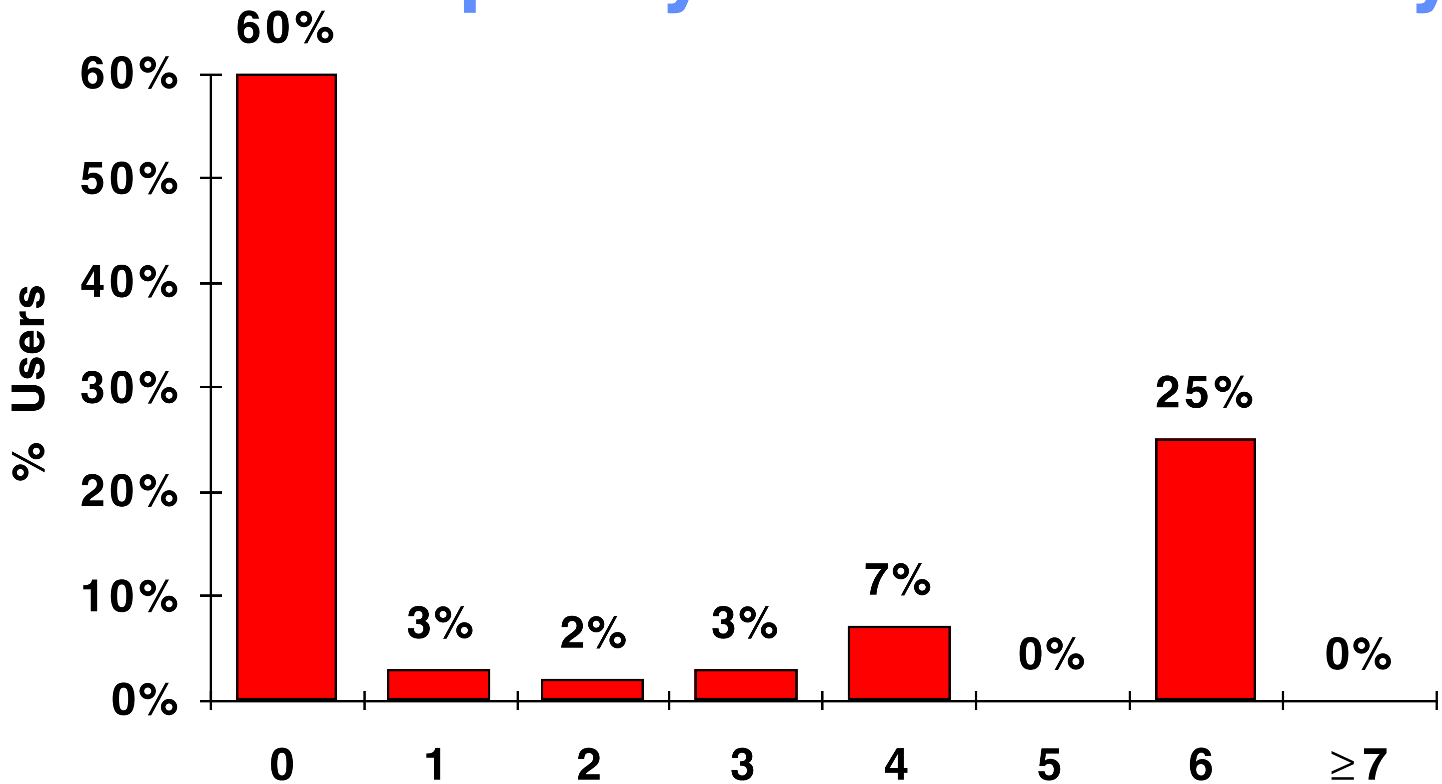
- 51 DECStation 5000s measured for 1 week, local disk and 64 MB memory; for IC design
- Measured CM-5 at Los Alamos National Labs 10/4/93 to 11/10/93 as prototype large program workload
- Simulated 32-node MPP workload on NOW with sequential workload (ignore network)

Sequential & Parallel on 1 System

- **Sequential has priority**
- **Ratio MPP nodes:desktops 3:5=>1.2x slowdown**



Annoyances per Day with MPP workload: policy limit 6 / user / day



User interactivity preserved with simple policy (no policy some users annoyed > 20 times/day)

Glunix Technical Challenge: Interactive Performance

- **Must gang schedule parallel jobs to be as good as dedicated MPP for parallel jobs**
- **Must quickly restore state to be as good as dedicated workstation for uniprocessor jobs**
- **Focus on memory state as well as CPU cycles**
  - **Delay in restoring memory biggest roadblock to harvesting idle cycles**
- **Time to save or restore:**
  - **64MB over Ethernet, single disk 60 seconds**
  - **64MB over ATM, parallel file sys 2 seconds**

Issues and Potential Solutions

- Network Overhead
- Preserving Response Time for large and small jobs
  - Gang scheduling for parallel tasks
  - Recruiting□ idle workstations
  - Not annoying interactive users
- I/O Bottleneck
- NOW helps only parallel jobs?
  - NOW File System (xFS): large file cache
  - Network RAM: avoid I/O

xFS: File System for NOW

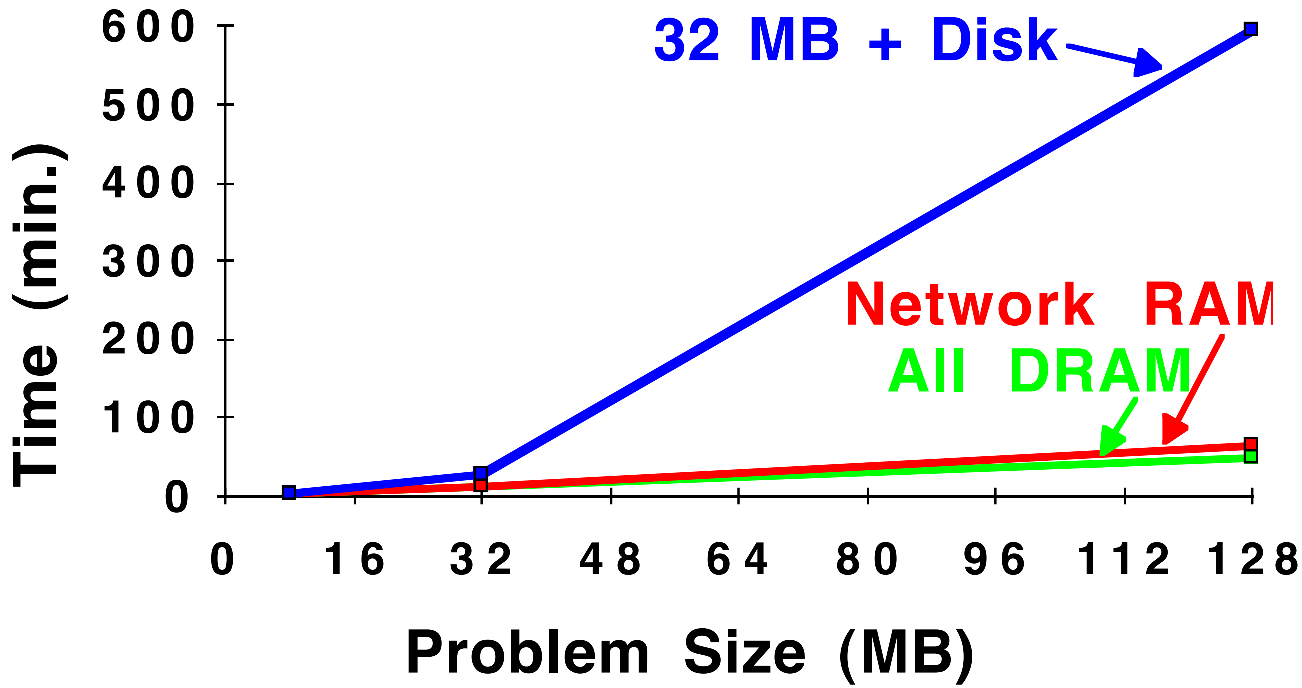
- Serverless File System: All data with clients
  - Use MP cache coherency to reduce traffic
- Files striped for parallel transfer
- Large file cache ("cooperative caching")

	Miss Rate	Response Time
Client/Server	10%	1.8 ms
xFS	4%	1.0 ms

(42 WS, 32 MB/WS, 512 MB/server, 8 KB/access)

- Paper at SIGMETRICS '94
  - Tech. Report: UCB/CSD-94-798
  - anon FTP: cs-tr@cs.berkeley.edu

Network RAM simulation



- 1.1X to 1.3X slower v. all DRAM
- 4X to 9X faster v. DRAM+disk

3 Paths for Applications on NOW?

- **Revolutionary (MPP Style): write new programs from scratch using MPP languages, compilers, libraries, …**
- **Porting: port programs from mainframes, supercomputers, MPPs, …**
- **Evolutionary: take sequential program & use**

increasing programming difficulty

1) Network RAM: first use memory of many computers to reduce disk accesses; if not fast enough, then:

2) Parallel I/O: use many disks in parallel for accesses not in file cache; if not fast enough, then:

3) Parallel program: change program until it uses enough processors that it is fast

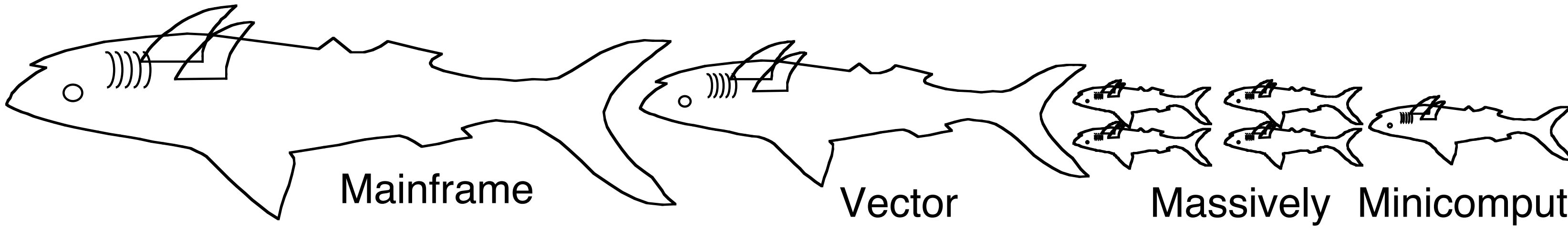
=> Large speedup without fine grain parallel program

Research Focus at New Level

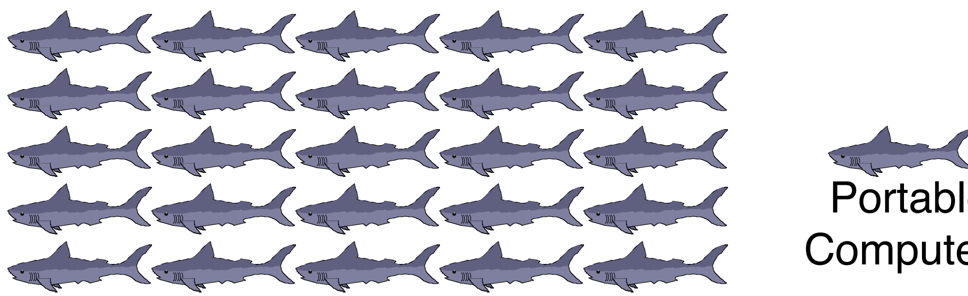
- "Higher Order" Systems Research: building on top of other systems vs. bottom-up
  – Must avoid time lag: neither HW nor OS can delay putting new machines to use
- Advantages:
  + easier to track technological advances
  + less development time
  + easier to transfer technology (reduce lag)
- New challenges:
  – maintaining performance goals
  – system is changing underneath you
  – underlying system has other people's bugs
  – underlying system is poorly documented

Conclusion

- 1990s building block is desktop HW&SW
- Need higher-level system research use building blocks: stand on shoulders, not toes
- NOWs underutilized => add large programs
  - Sequential apps use memories & disks (Network RAM)
  - MPP apps use CPUs, memories, & disks
- Technologies aligned to exploit NOW now
  - 32-bit μprocessors, switch based LANs, active messages, striped files, file caches, process migration
- Challenges for NOW: Leveraging technology yet add low overhead user communication, global OS, parallel file system



2004 Computer Food Chain



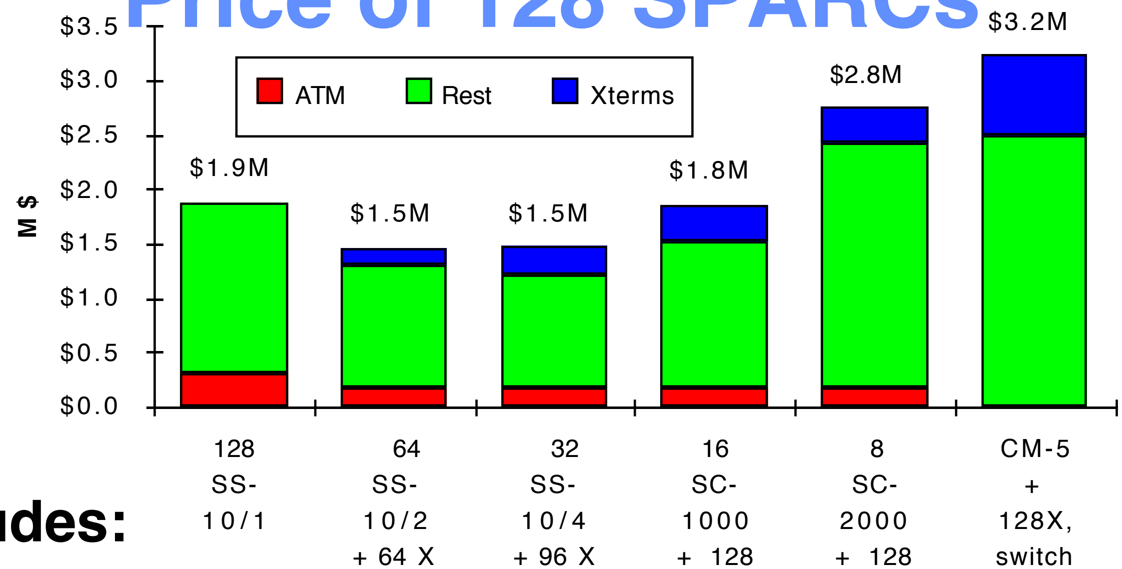
Backup Slides

- (The following slides are only used to answer questions)

Other NOW projects

- Shrimp at Princeton (Li, Clark):
  PCs with Intel Paragon network
- COW at Wisconsin (Hill, Wood):
  SPARCstations with shared address space
- Related projects at MIT, Rice, ...

Price of 128 SPARCs



- **Includes:**
  - **128 50 MHz SuperSPARCs w. 1 MB external cache (3/94)**
  - **4 GB of DRAM (32 MB/processor□)**
  - **134 GB of magnetic disk (128 1.05 GB magnetic disks)**
  - **128 screens (native or Xterms)**
  - **Switch (native or ATM: 1 interface/2 procs+ switch)**
    - » **$700/node for interface + $70,000 per 64-way switch**
- **Cost Xterms for MPP > Cost ATM for NOWs**
- **≈ 2X MPP v. new NOW, ≈10X MPP v. old NOW**

Why Higher Price for Same Components in SBMPs?

- **SparcStation-10 (1 to 4 processor desktop) vs.SparcCenter-2000 (2 to 20 proc. server)**
  - **Same processor and cache as building block**
- **ASIC Costs/Proc**

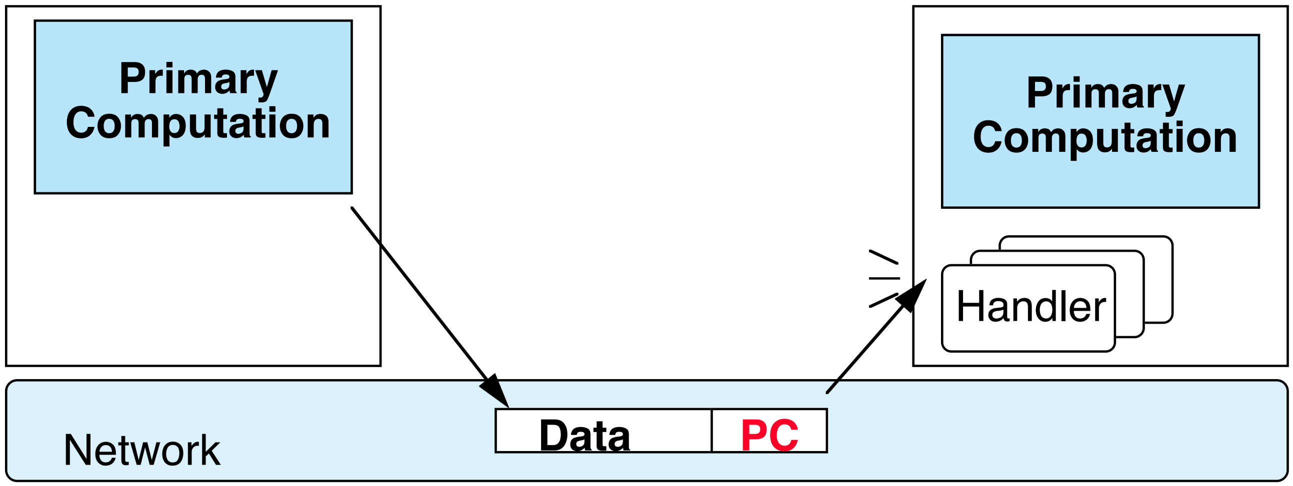
ASIC Costs/Proc	SS-10	SC-2000	Ratio
Number ASICs	5	8	1.6
Total Gates	90k	235k	2.6
Person Months	145	305	2.1
People Costs	$1.5M	$3.0M	2

- **Sales** (9/93-12/93) **≈33,000 ≈1,000 33**
- **Higher development spread over fewer sales => customer pays more for same processor**
- **Worse for MPPs since even smaller volume**

Hidden Costs of Large Systems

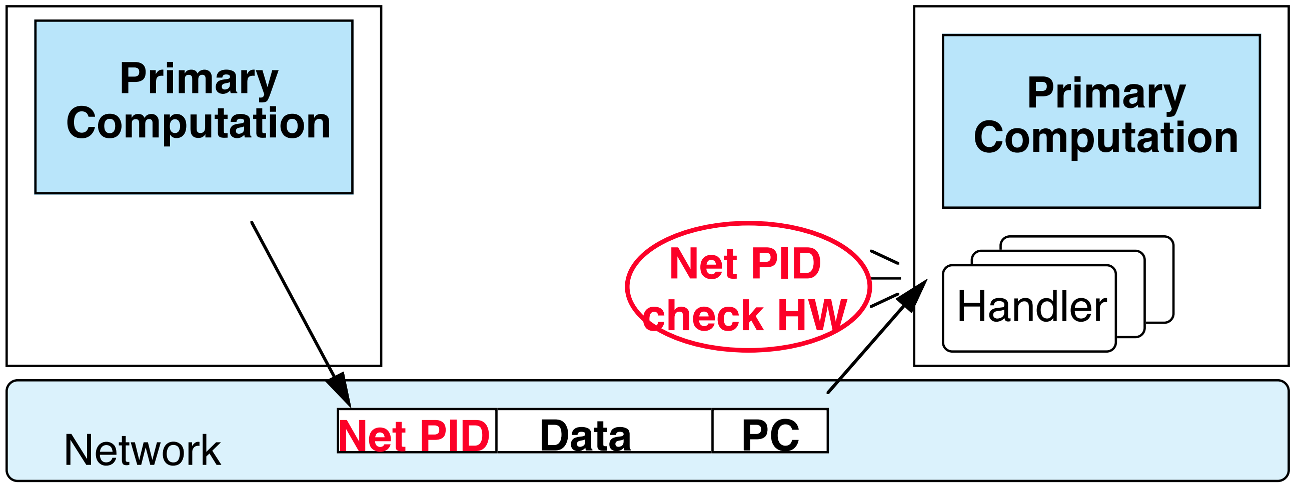
- Spares/Self maintenance for NOW vs. 5% to 10% purchase/ year for SBMP/MPP
- Upgrade components of NOW vs. discard for SBMP/MPP
  - SBMP limited processor upgrade (discard?), can't upgrade bus
  - MPP limited processor upgrade (discard?), can't upgrade network
  - LAN enables individual upgrades of workstations and/or switch
- NOW cheaper at purchase and cheaper to own

MPP Active Messages



- **Key Idea: associate a small user-level handler directly with each message**
  - **Sender injects the message directly into the network**
  - **Handler executes immediately upon arrival**
  - **pulls the message out of the network and integrates it into the ongoing computation, or replies**
  - **No buffering (beyond transport), no parsing, no allocation, primitive scheduling**

NOW Active Messages



- **Key Idea: Network Process ID attached to every message that HW checks upon receipt**
  - **Net PID match, as fast as before**
  - **Net PID mismatch, interrupt and invoke OS**
- **Can mix LAN messages and MPP messages; invoke OS & TCP/IP only when not cooperating (if everyone uses same physical layer format)**

Latency & Overhead for ATM

- **Latency: worse than MPP**
  - **Links latency basically speed of light (1000 ft = 1 µsec)**
  - **Per-hop latencies:**
    - » **SynOptics 50 µsec**
    - » **Fore 10 µsec**
    - » **AN2 ≈ 2 µsec**
  - **Store and Forward vs. Cut through routing**
  - **Bigger switches so fewer hops (1/3): 6 to 150 µsec**
- **Overhead: comparable to MPP**
  - **HP WS UDP (OS) 360 µsec**
  - **HP WS w. A.M. 8 µsec (if can avoid OS)**

Pitfalls for NOWs

- Invoking operating system when communicate
  - 100s μsec overhead added to low latency communication
- Rewrite/Modify WS operating system to include features for NOW
  - Limited to single brand of desktop computer
  - Can't leverage of OS improvements by vendor
  - New HW useless until OS port => lower performance
- Design NOW to only help large programs that are parallel
  - Few applications are parallel => hard to justify fast NOW
  - Many large programs just need memory and disk BW
- Serial file system
  - can't take advantage of 100s of parallel disks

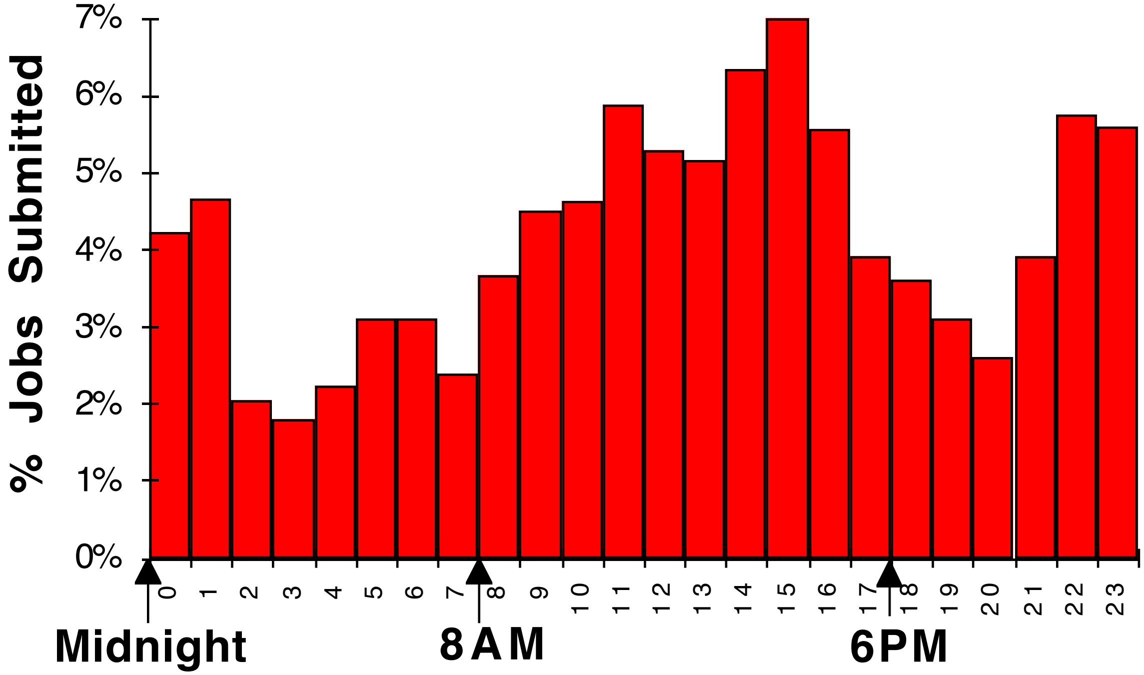
Pitfalls for NOWs (cont'd)

- **Design custom network interface HW &SW for single model of desktop computer**
  - New HW useless until new NI HW, SW port
    => lag time and lower performance
- **Custom proprietary network as new LAN**
  - LAN market demands standardization => multiple suppliers & add new products to network ASAP
  - Too important to rely on a single supplier
- **Scaling WS OS kernel beyond 32 processors**
  - Kernel locks are bottleneck as well as shared bus
- **Parallel tasks don't run at same time**
  - Parallel program communication much slower if nothing to consume messages from other parallel tasks

Heterogeneity of Workstations

	SPARC	D/MIPS	HP PA	RS/6000	Misc.
Berkeley	100	85	23	5	50
Cornell	150	0	11	1	50
Duke	110	0	0	1	29
Washing.	33	65	2	0	21
Wisconsin	48	228	47	0	99

Time of day submit MPP jobs



24% 12am-8am, 52% 8am-6pm => need daytime MPP!

User’s View of GLUnix

- **User’s workstation + aggregate CPUs, DRAMs, & disks of entire network**
  - sequential apps run as if on standard UNIX
  - parallel apps: network process
    - » coordinated scheduling
    - » single system view of OS services
- **System must survive node failures, migrate activity away from interactive use**

GLUnix Tradeoffs

If build kernel from scratch:

- **clean, elegant design possible**
- **hard to keep pace with commercial OS development**

If layer on top of unmodified commercial OS:

- **struggle with existing interfaces**
- **work-arounds may exist for common cases**

Goal: look for minimal set of changes to commercial OS that provide most leverage for demanding apps.

GLUnix Technical Challenges

- Implementing co-scheduling on top of UNIX kernel
- Preserving interactive performance
- Fault tolerance – surviving node failures, software upgrades, hardware expansion
- Free RAM
- Parallel file systems on workstation platforms

  . . .

Technical Challenge: File systems

Technology push to re-think network file systems:

- Aggregate ATM bandwidth > single disk
- workstations cheaper than server machines
- tertiary storage to provide infinite capacity
- wide area access is slow, expensive and unreliable

Application pull:

- high availability is a necessity
- peak demand >> average demand
- parallel program I/O

OS Features for Large Programs

- **Desirable characteristics for Sequential Tasks**
  - reliability
  - use processors for sequential tasks
  - low-overhead user level communication
  - standard services of WS: virtual memory/paging
  - parallel file system for fast I/O
  - system survives node crash
- **Added characteristics for Parallel Tasks**
  - network process
    - » single view of system services (files, sockets, ...)
  - co-ordinatred scheduling of logical program on all nodes
  - effective multiprogramming of sequential interactive programs with parallel programs
  - protected communication

OS Assessment

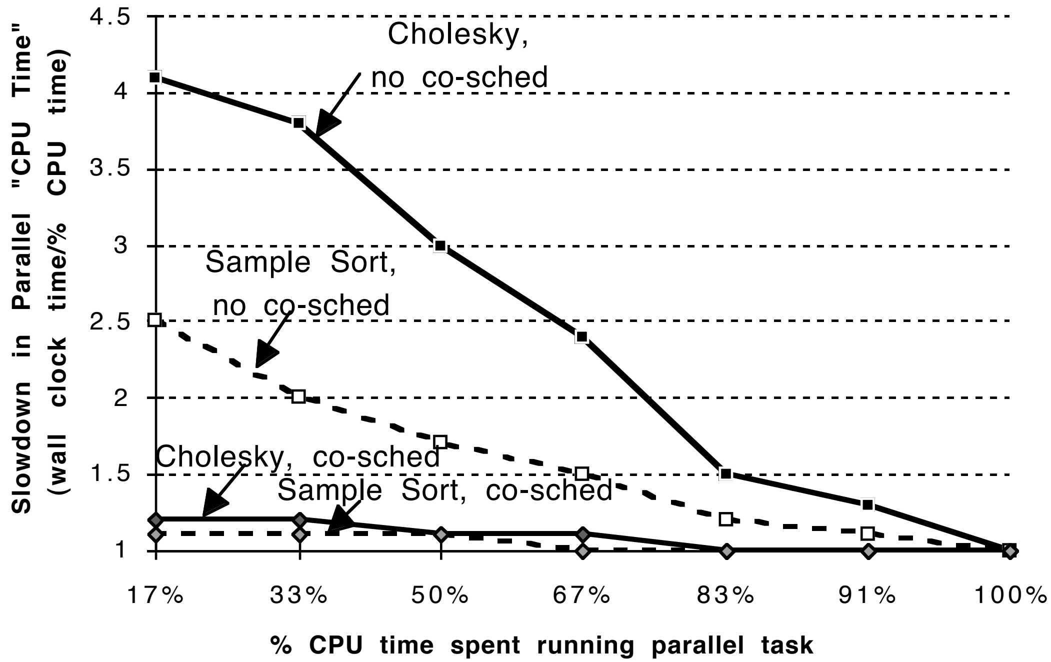
	SBMP	MPP	NOW
reliability	Yes	No	Yes
sequential tasks	Yes	No	Yes
low-overhead comm.	Yes	Yes	No
virtual memory/paging	Yes	No	Yes
parallel file system	No	Yes	No
node crash survival	No	No	Yes
network process	No	Yes	No
co-ordinatred scheduling	No	Yes	No
S/P multiprogramming	No	No	No
protected communication	Yes	No	Yes

- **All OS have weaknesses for large, parallel programs!**

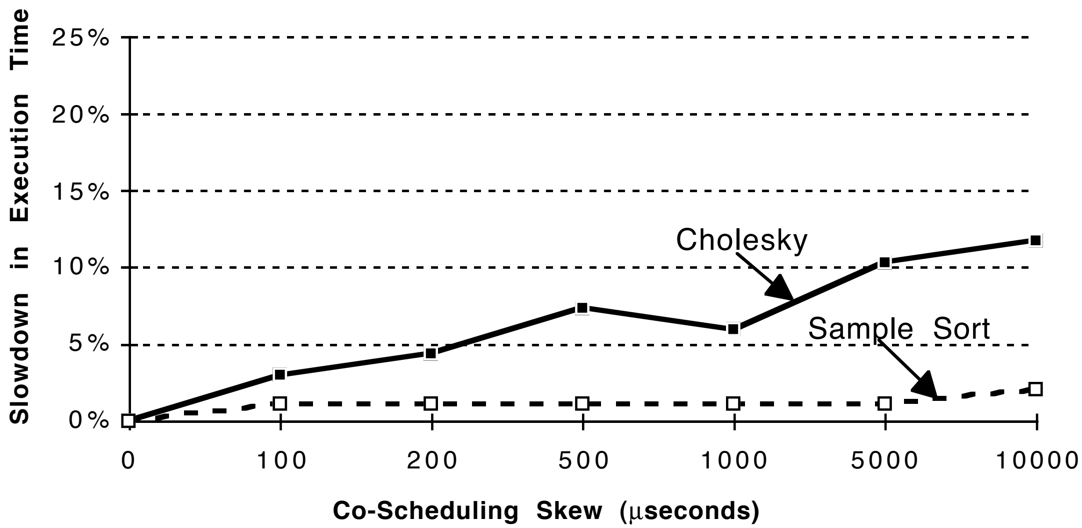
Co-scheduling Experiment

- How important is co-scheduling to performance on MPP programs?
- Measured on CM-5 inserting random process to vary the amount of time processor runs parallel task vs. an independent serial task
- Two programs: Cholesky and Sample sort, with and without co-scheduling: 2.5 to 4X vs. 1.1 to 1.2 with
- Third program, EM3D, goes off the chart without co-scheduling at 17% parallel task (35X slower) vs. 1.2 with co-scheduling
- But skew in time slices not critical

Co-Schedulilng Value



Value of exact start times of process co-scheduling

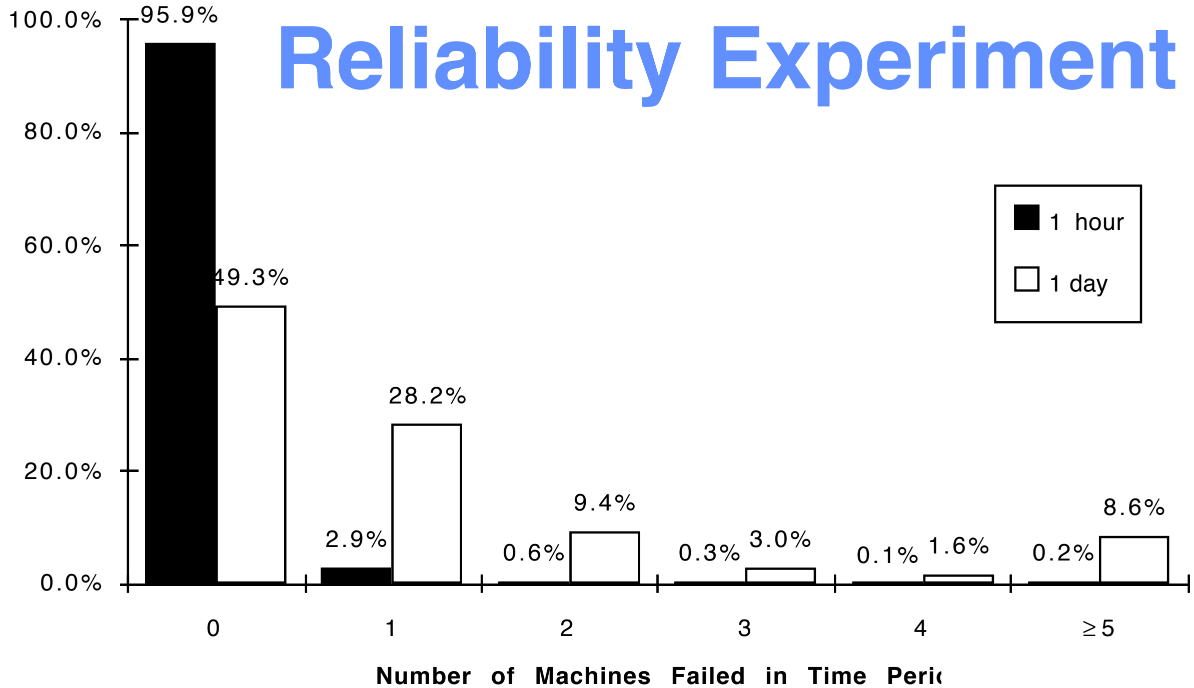


- **Large skews in synchronization of process start times make little difference in run time**
- **Expect real skews < 1000 µsecs (5% impact)**
- **Conclusion: Effective co-scheduling plausible for NOWs**

How About Reliability of WS HW/OS?

- Do workstations fail so frequently that can't handle MPP workload? (all parallel machines stall until dead system reboots)
- 58 DECstation 5000s measured for > 1 year
  – Only 1 time/year all machines unavailable□ (power failure)
  – 632 reboots: 345 Shutdowns + 1 power failure for 58 machines + 229 surprises
  – Virtually every time run in degraded mode

Reliability Experiment



- **Estimated impact on MPP workload if this sample generalizes**
  - **Chance of ≥1 machine of 58 reboot in 1 hour is 4%**
  - **Chance of ≥1 machine of 58 reboot in 1 day is 50%**
  - **Chance of non-user directed reboot in 1 hour is<2%**
  - **Chance of non-user directed reboot in 1 day is <25%**
- **Not a problem if jobs << 1 hour**

MPP Workload & NOW Reliability

- Automatically checkpoint jobs that run longer than 30 minutes every 30 minutes
- Restart if crash
- If checkpoint takes 1 minute & lose 2% jobs taking >30 minutes, total extra time for long jobs:

  $\approx$ 4 x 1 min + 2% x (30/2) = 4.6 minutes
- <5% overhead to make it very likely to finish very long jobs

xFS vs. AFS: Server Load

- **Simulation□ using Berkeley Auspex NFS Traces: 4 Networks, 237 Clients, 6 Days (+1 Day of Cache Warming)**
  - **Networks, CPUs, In-Memory File Caches, Disks**

	Server Messages	Server Data	Server Load
AFS	**1.4 M**	**15.2 GB**	**100%**
xFS	**0.4 M**	**0.0 GB**	**15%**

- **6:1 Reduction in Server Load**
- **Network Bytes Through Server Reduced More Than 99%**