Federated Systems

Lecture 2
Machine Architectures
Richard Martin
Sept. 14th 1999

Today

- Discussion of the major architectures::
  - Shared Memory Machines:
    - cc–NUMA Origin 2000
  - Clusters
    - Berkeley NOW
  - Federated Machine
    - Netstation
  - Database Machine
    - Teradata DBC

Terminology

- Terminology
  - UMA
    - Uniform Memory Access
  - NUMA
    - Non–Uniform Memory Access
  - cc–NUMA
    - Cache–Coherent Non–Uniform Memory Access
  - NORMA
    - No Remote Memory Access

Terminology−Databases

- Shared Memory
  - UMA, and ccNUMA machines
- Shared Disk
  - NORMA with shared I/O system
    - Shared SCSI or Fiber−Channel
- Shared Nothing
  - NORMA with separate I/O

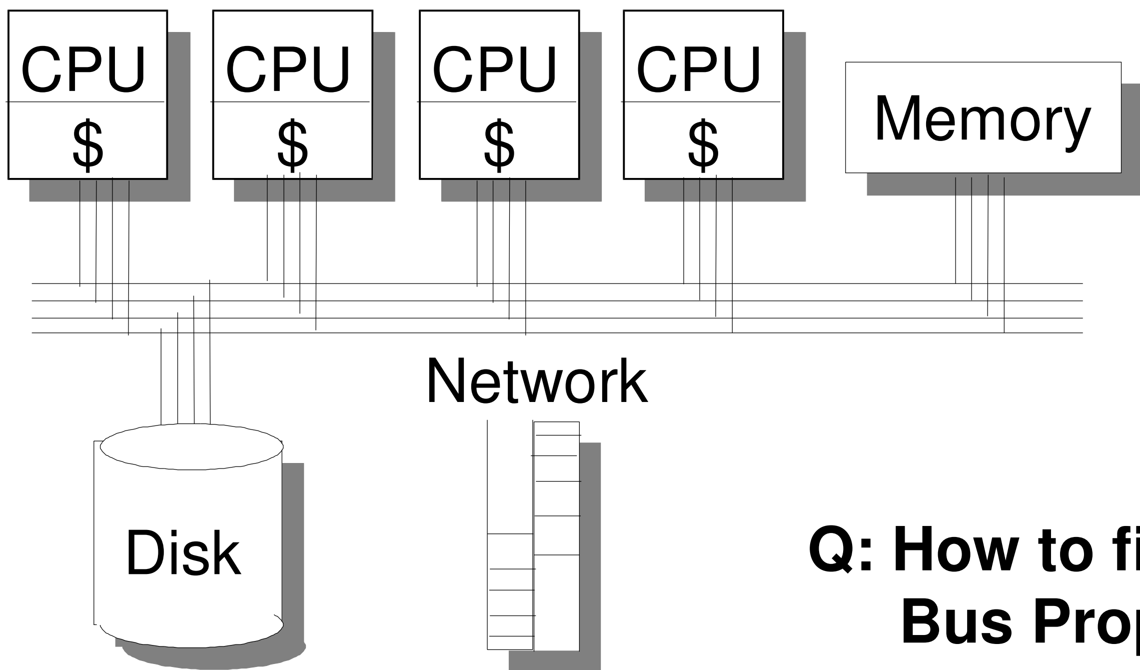
Big Picture

- How to construct the "ideal" server?
  - Managability of an SMP
  - Scalability of Cluster
  - Availabilty of a Tandem

Shared Memory Machines

- History is one of increasing scale
- Started with small scale shared bus
- Hierarchical busses
- DASH: cc–NUMA design
- Origin 2K "ideal" hardware scaling

Small scale bus machines



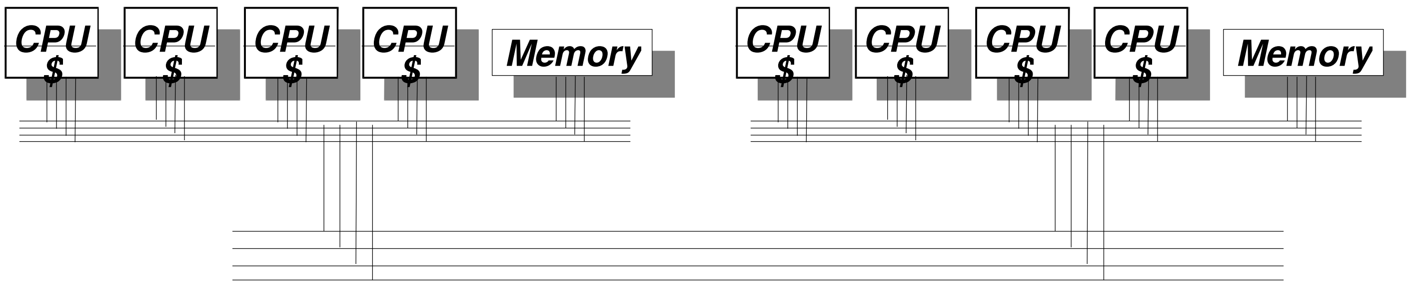
Q: How to find a cache line?

Bus Properties

Broadcast => snooping

Hierarchical Buses

- How to scale up from 4–8 way?



Scaling a snoopy protocol?

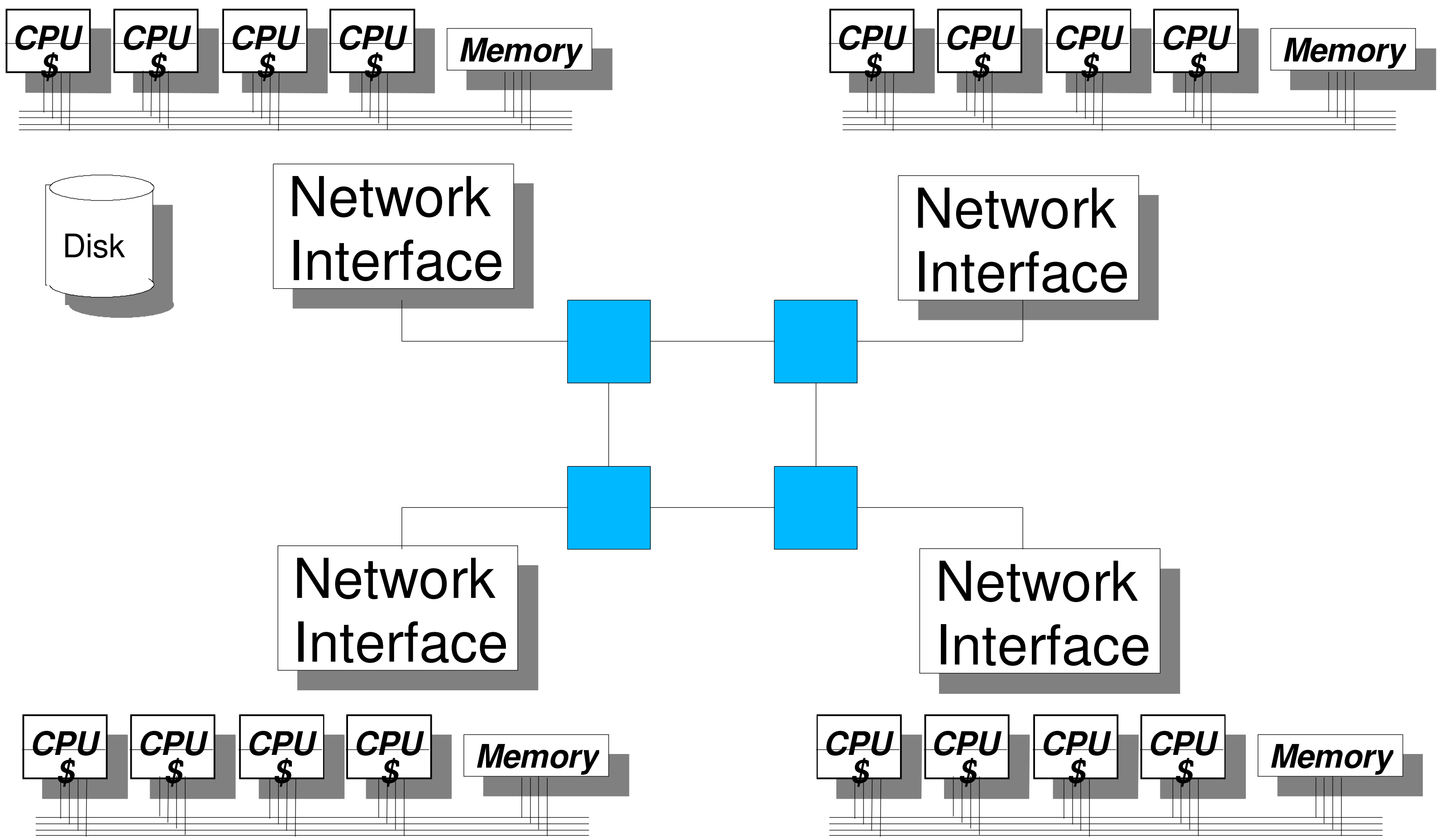
Bus occupancy?

Bandwidth/per node

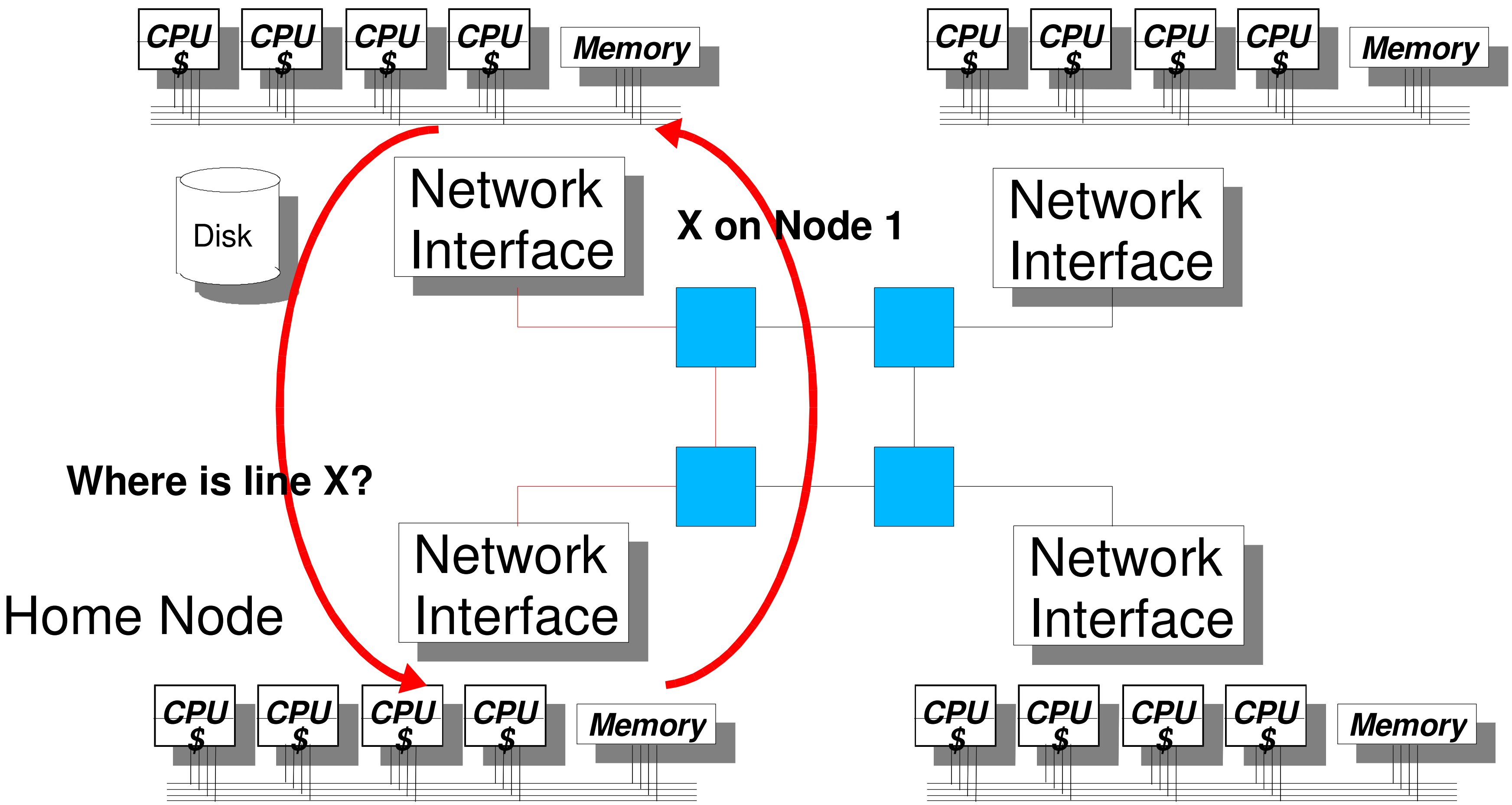
Dash cc–NUMA

- No more snooping!
- How to find a cache–block?
  - Each node holds a directory
    - nodes with copies
    - state of each block
- Coherence protocols very complex!
  - Bugs!
  - Spawned research in coherence protocols

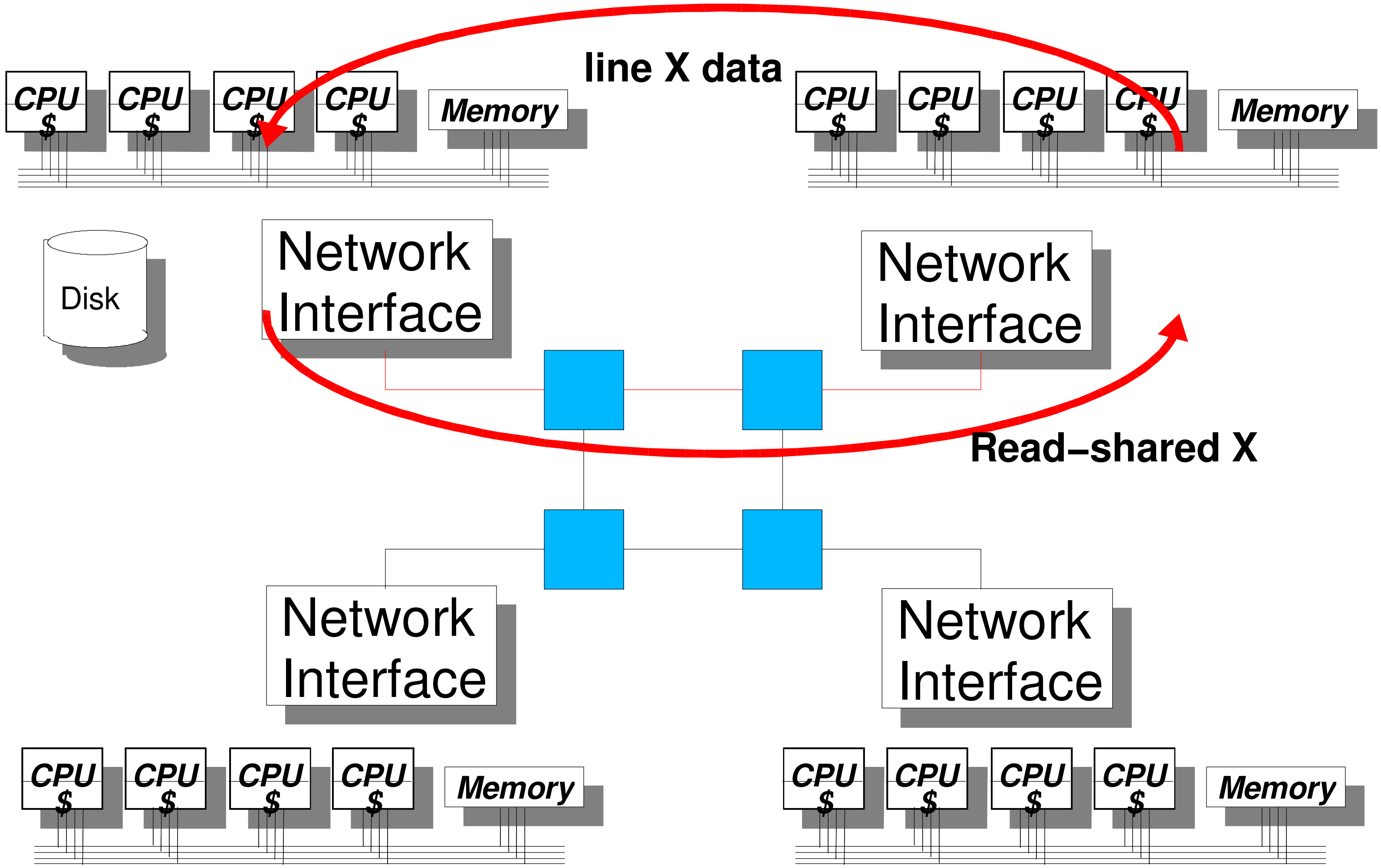
Dash Architecture



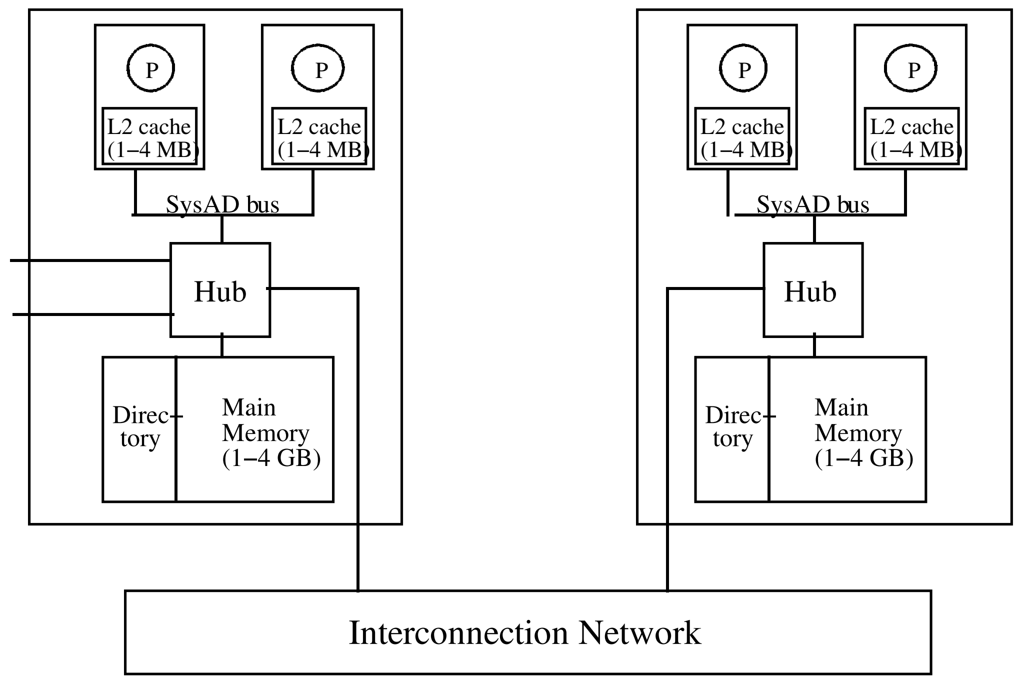
Finding a line



Getting a line

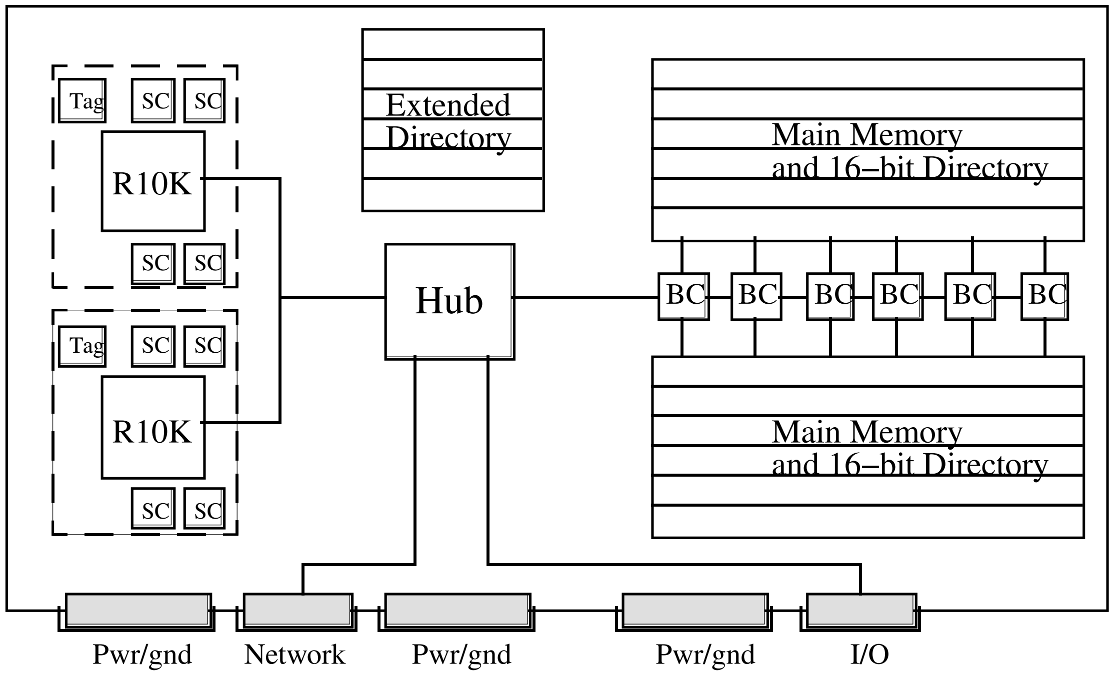


Origin 2000



- Directory held in DRAM
- Scales to 512 nodes (1024 procs)

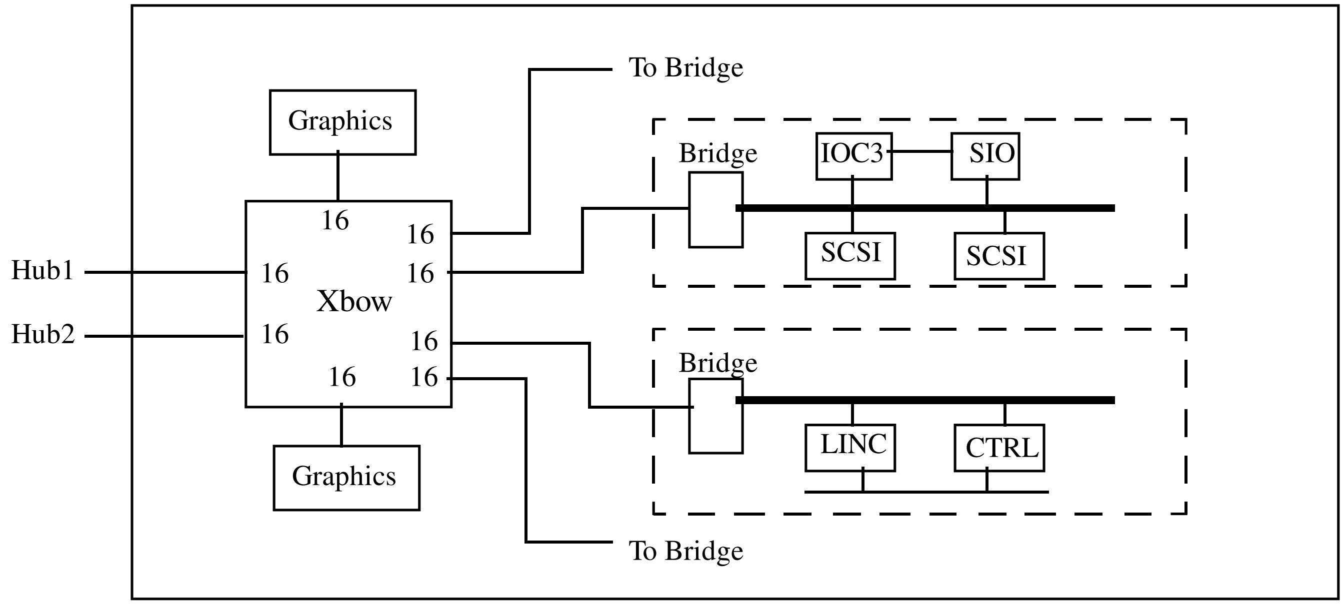
O2K node architecture



Connections to Backplane

- Hub provides support for synch primitives, and for page migration
- Two processors within node not snoopy−coherent (motivation is cost)

O2K node I/O



Global I/O space

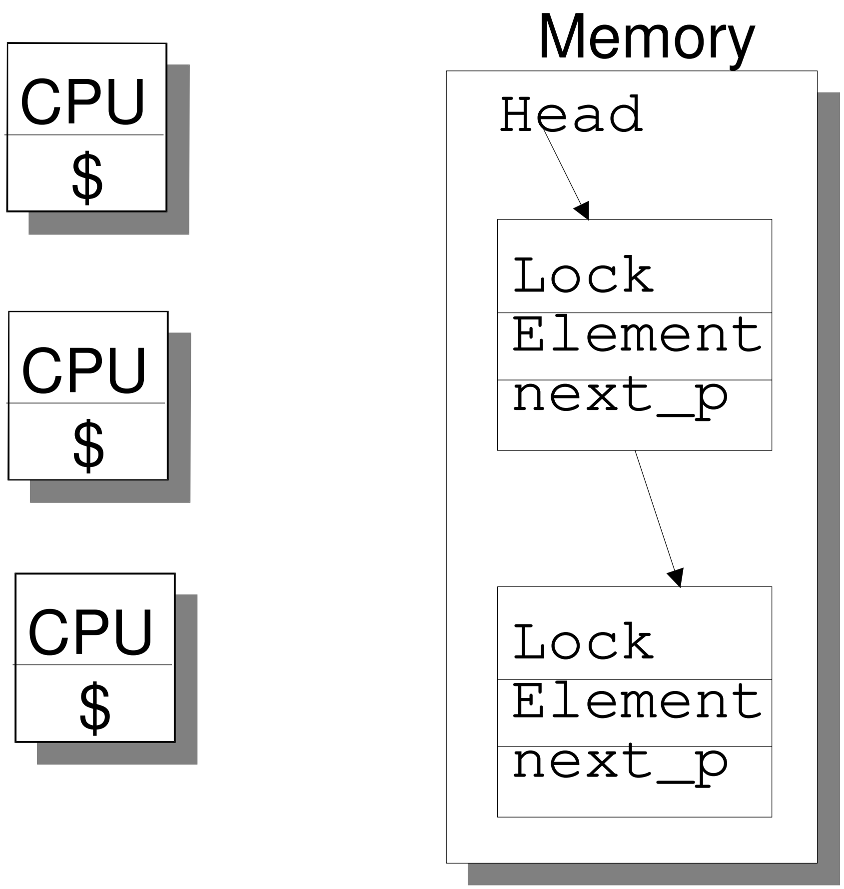
–any CPU can access any device

–any device can access any memory

Shared memory Model

- All that hardware effort for what?
- Maintain the illusion of a shared memory
  - Programming applications
  - Re–Use the Operating system

Shared memory programming

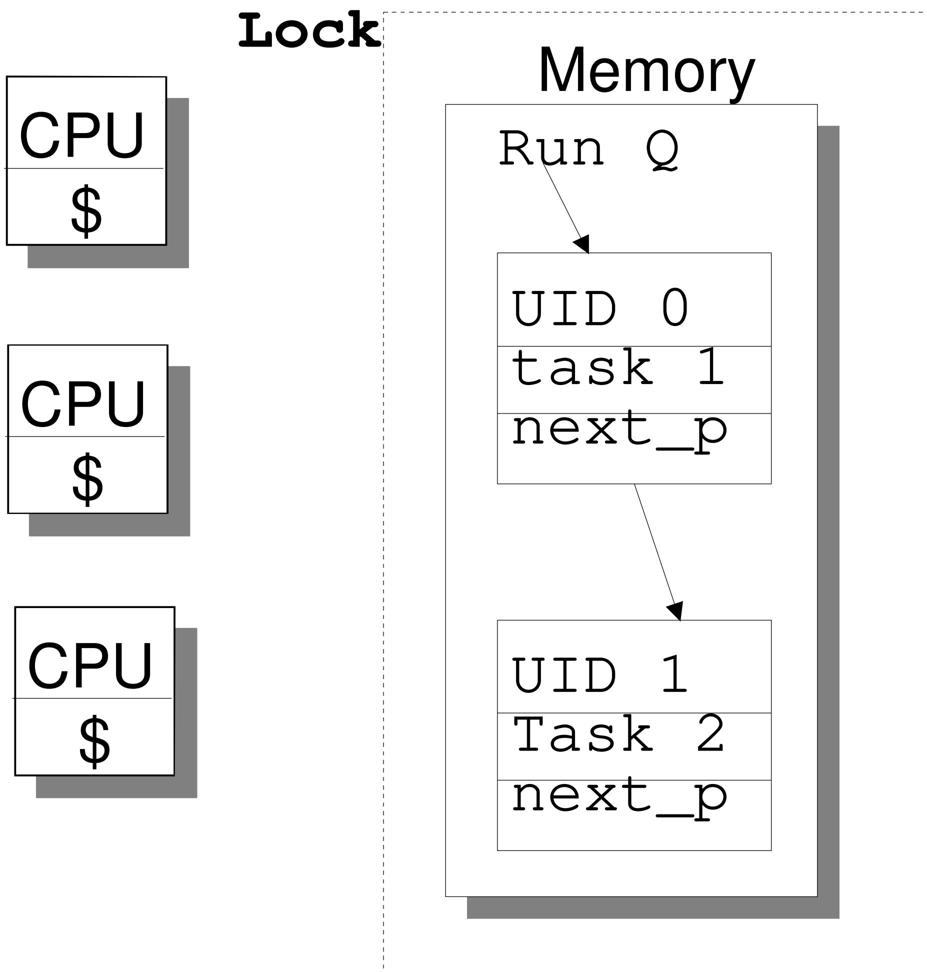


Each processor sees the same structures in memory

Simple to understand model

Uniform access time does not scale past 16–32

Cheap SMP OS Model



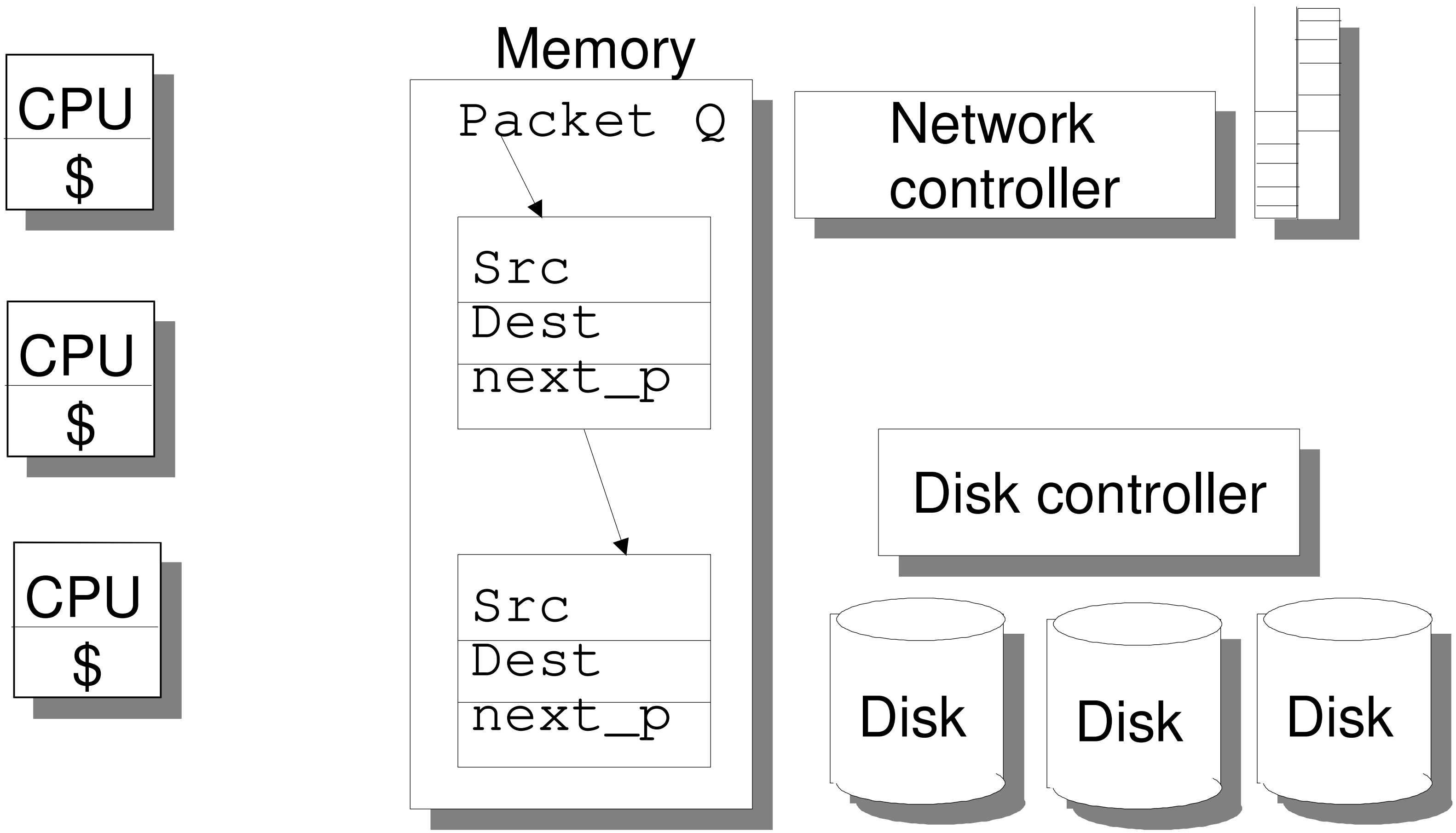
1. Lock whole OS
2. Update structures
3. Return

Scales 2–4 way

SMP OS for very little work!

Can push locks "down"

SMP Device Model



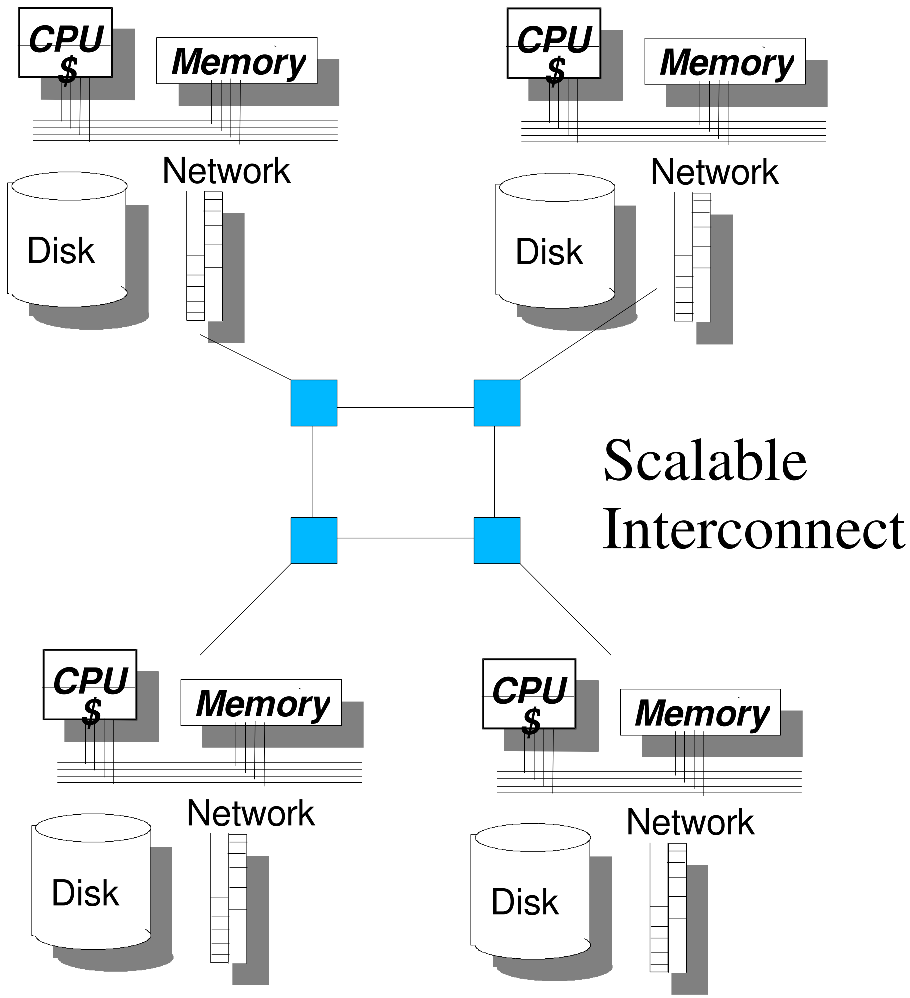
SMP Availability

- Fundamental Sharing vs. Availability trade–off
  - Any CPU or device can corrupt state of all
  - Any performance perturbation can affect all
- What happens to shared state on a failure?
- How to build firewalls inside the machine?
- Software firewalls as well?

SMP Manageability

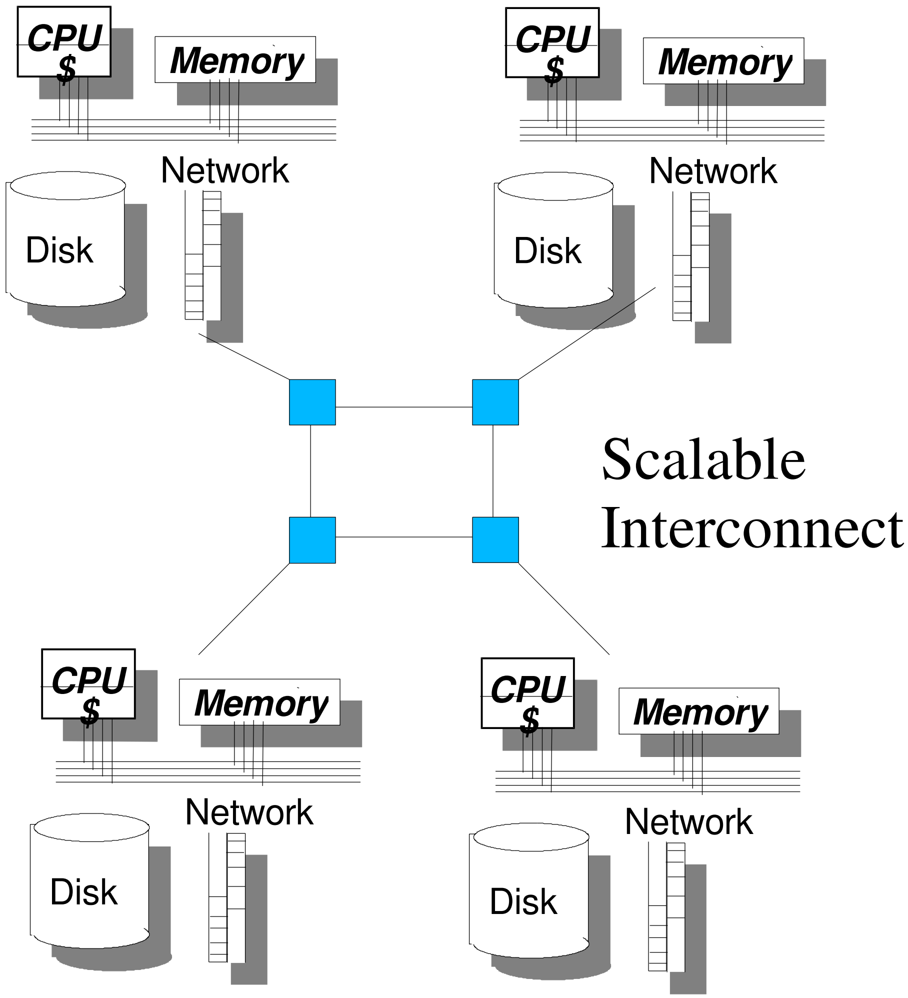
- Looks like 1 machine!
  - Not much more work to manage :)
  - Getting performance a different issue– more like messaging than shared memory

Cluster/MPP architecture



- Try to re–use whole computer as building–block
- System Software?
- Management?

Cluster/MPP scalability



Scaling:
- processing
- disk
- memory
- cache
- network
- cache
- OS

Why NOW then?

- Idle compute power?
  - Finding cycles not a problem
  - Programming cycles is the problem
- Engineering?

  Why build N kinds of systems?

  Re–use the same building blocks!
  - software building blocks?
  - Glue layer an OS on an OS
    - Complexity is fundamental

Why NOW then (cont)

- Scalable servers
  - Memory server
    - VM overhead still high, but still get 4–5x over disk
  - File server
    - XFS
    - cc–NUMA hard, filesystem protocols harder
  - Web Servers (Inktomi)

Why NOW now?

- How to build scalable servers?
  - File servers
  - Database servers
  - Application servers
- Cost and Performance demonstrated
- Availability?
  - Commerce Servers?
  - Web Servers?

Cluster Scheduling

- How do you run jobs on a cluster?

```
foreach i (node1 node2 node3 node4)
    rsh $i <my_program>
```

  - seconds of overhead
    - Not going to break any records that way
- Job cleanup?
- What happens when a node crashes?
- Lots of work on implicit scheduling
  - Assuming you got through start–up.

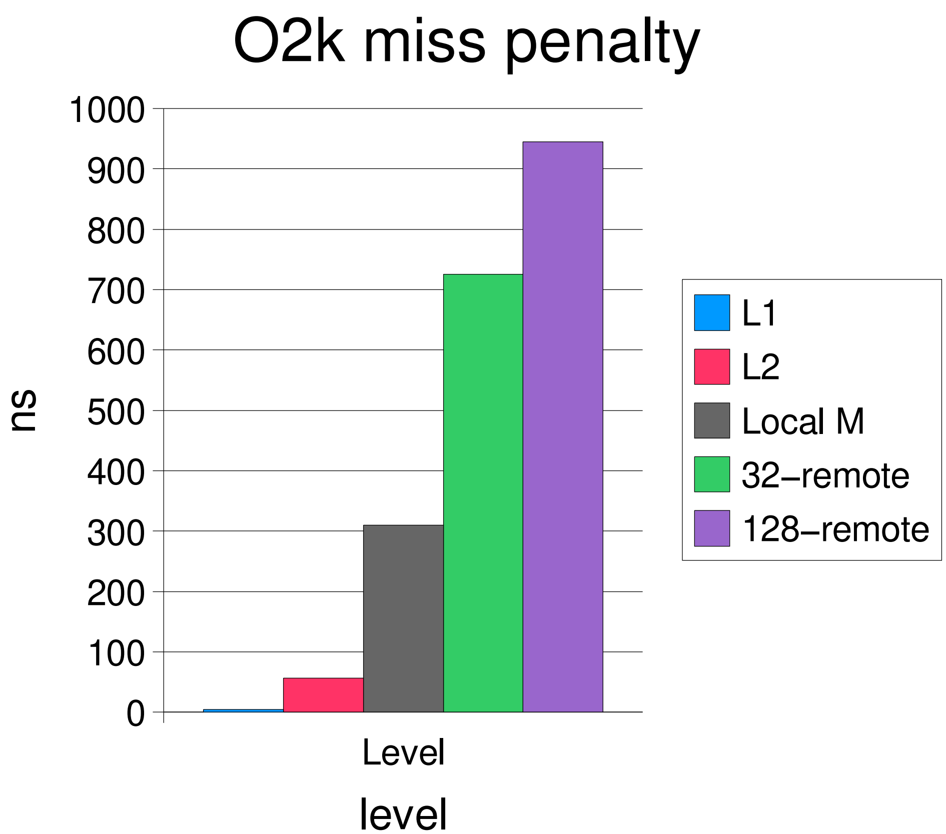
Cluster OS

- Glunix had single master architecture
  - Performance to 100 OK.
  - Maintainability easier
  - Availability terrible
    - Not just flaky software!
    - Single master fundamentally flawed design
- Available cluster OS still open research
- OS re–use a lost cause?

Cluster vs. SMP

- SMP: Re–use operating system!
  - What happens if a node goes bad?
    - Oops! Dangling pointers?
    - Does not scale down as well (used to be worse)
- Cluster
    - hardware scaling a no brainer
      - memory, I/O Bandwidth processor speed
    - Fault tolerance

Shared Memory vs Messaging



3x performance hit local vs remote

Managing locality critical

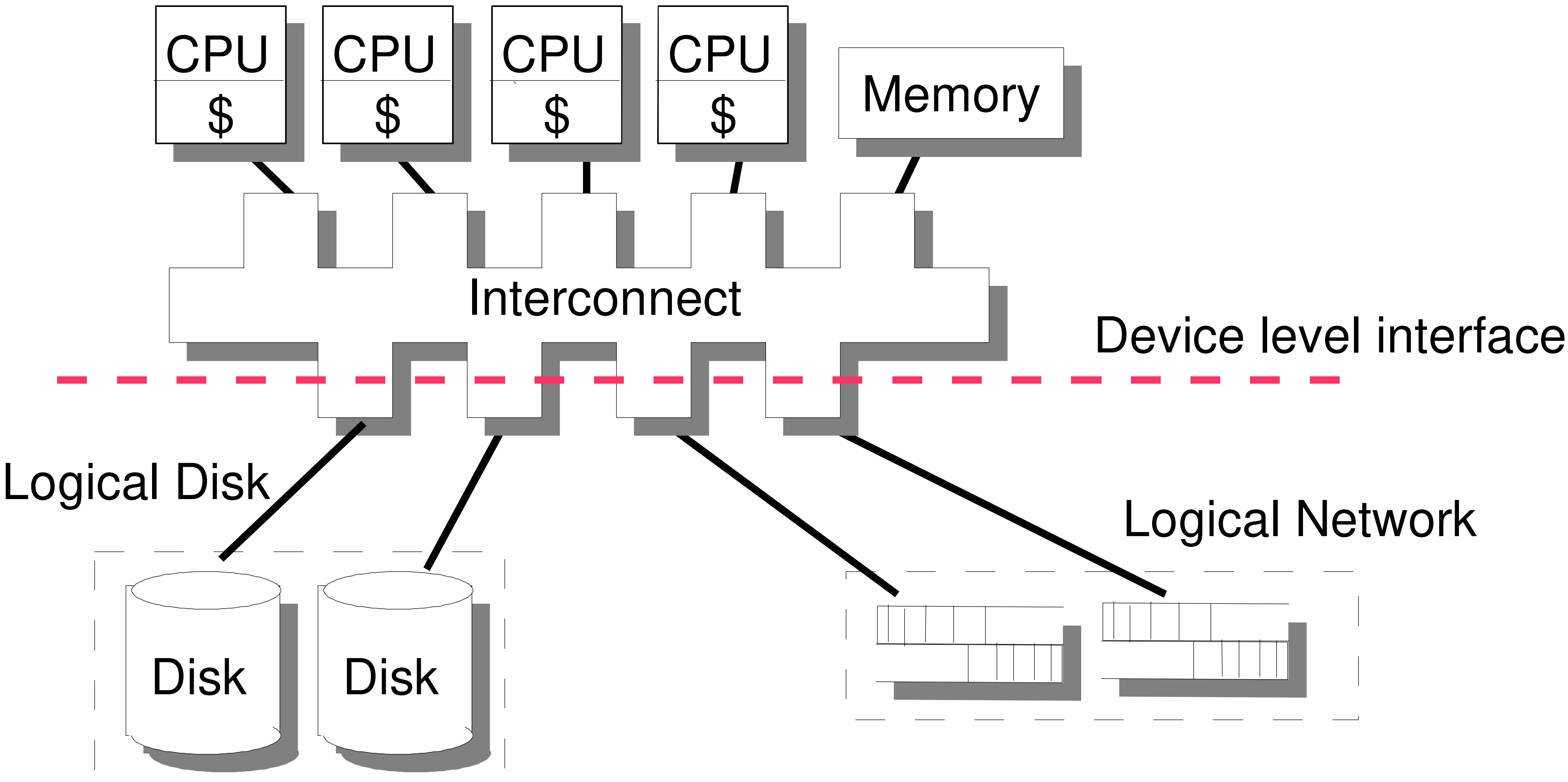
Partitioning and Orchestration almost same amount of work as message passing

Shared memory makes life slightly less painful.

Netstation

- Big idea:
  - Replace Bus with switched interconnect
  - Fiber Channel, Future, NGIO
- Implications?
  - More scalable I/O (How scalable?)
  - More available I/O? (not in paper)
  - Devices have more opportunity for scalability and availability

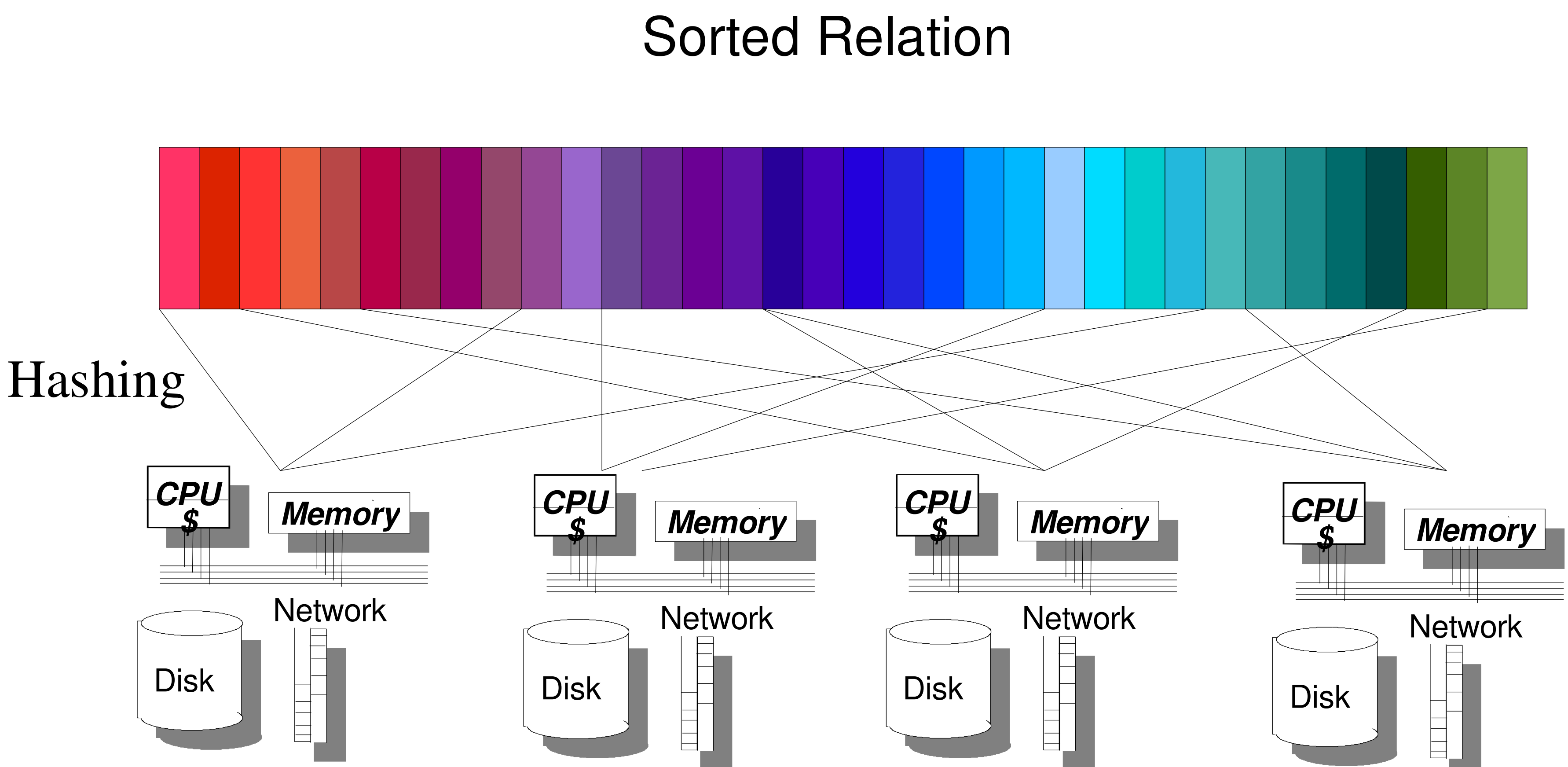
I/O sub–system model



TeraData DBC

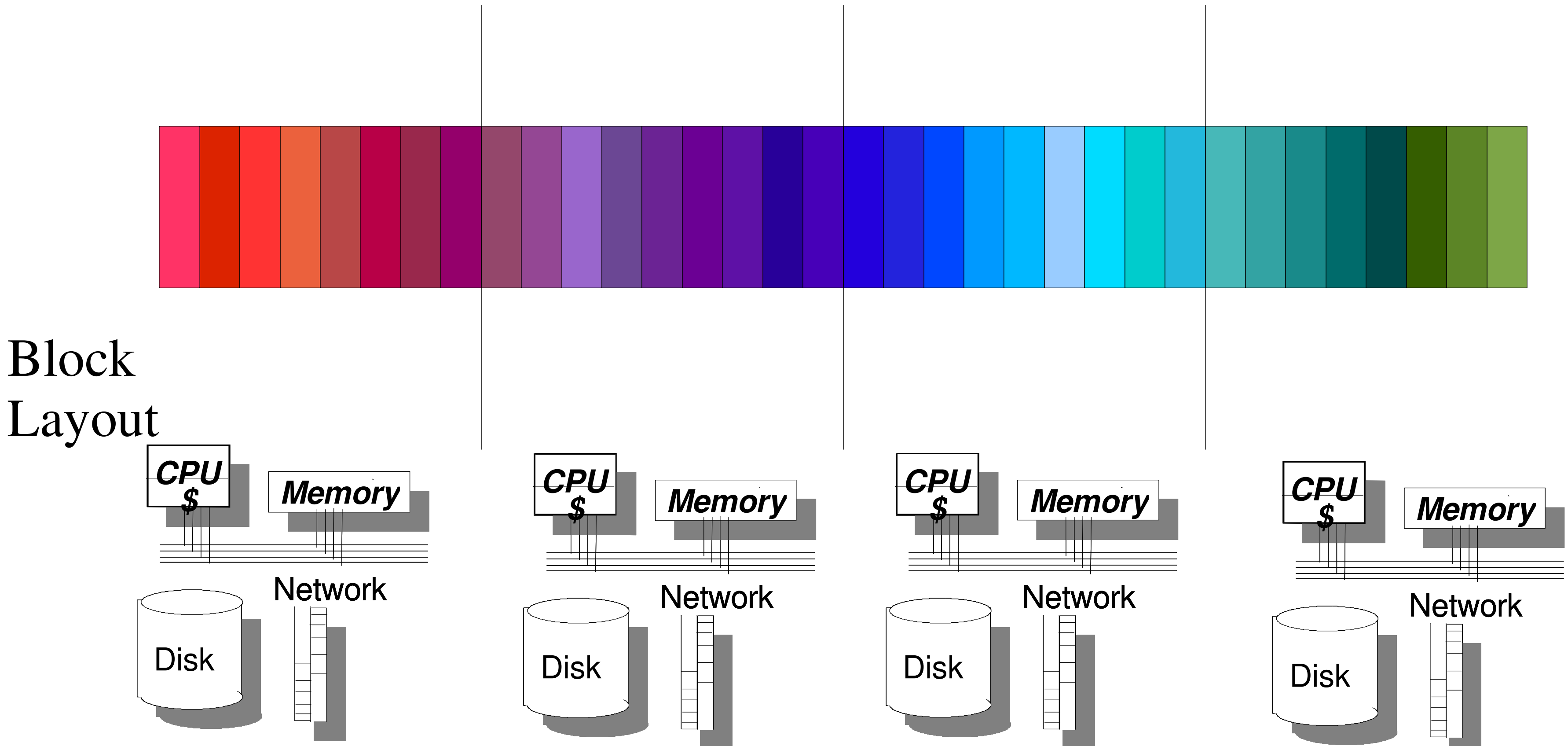
- Basically a NOW/MPP for a single application (relational databases)
  - What about non–relational DB's
- Excellent parallelism!
- Ended up using 4–way SMP nodes and mapping procs to CPUs on the SMP

Relational Locality



Relational Locality

Sorted Relation



Relational Locality

- Hashing
  - Predictable performance
  - no locality
  - All nodes must participate in every Query
- Blocking
  - potential excellent locality
  - potential horrible locality too!
  - Re−partition? When?

Relation Locality

- Full–Scan Query: All nodes look through their data
- Alternative is to build an index
- Trade–off between indexing and full scans
- Join Support in hashing algorithm
  - Have to move data around
  - Motivation for Y networks (sorting)

Putting it all together

- Common theme is locality management
- fundamental architectural principles:
  - Small CPU, memory,disk = faster cycle times
  - big CPU, memory, disk = slower cycle times
- No way around it
  - Problem will get worse with time
  - Can we give models to better help?

Readings for Next Week

- Network Performance analysis
  - LogP: conceptual model of networking
  - Pipelining: How fast is a single operation?
  - NetPerf: How to automatically derive LogP parameters from a probe test
  - Extra:
  - Araki, Bilas, et. al. SC98 LogP Comparison.