Federated Systems

Lecture 5
Storage Interfaces
Rich Martin
October 5th 1999

Today

- Storage Interfaces
  - Cluster (Rosselli Quals)
  - Remote Device (GFS)
  - Sea of blocks (swarm)
  - Virtual Disk (Petal)
- Semantics/Interface

Today

- Why study storage interfaces?
- Storage interface separate from File system

- Devices will become more programmable
  - What’s the interface to the rest of the system.

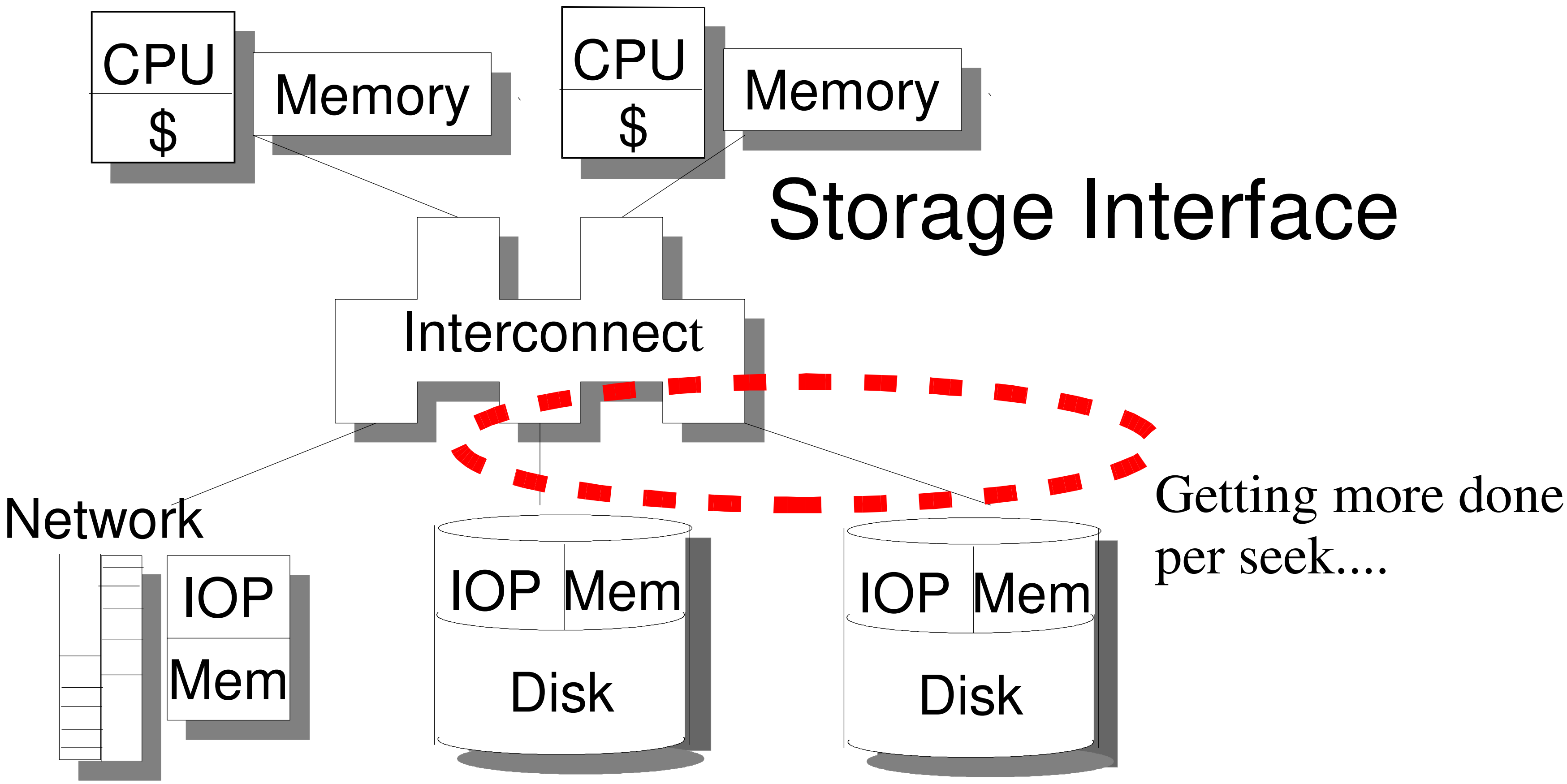
Storage Layers

My program

`Read()/Write()`

Layer	Description
File System	Directories, files, locks
Buffer Cache	Block–level cache
Storage System	Block–level Storage

Model Federated Architecture



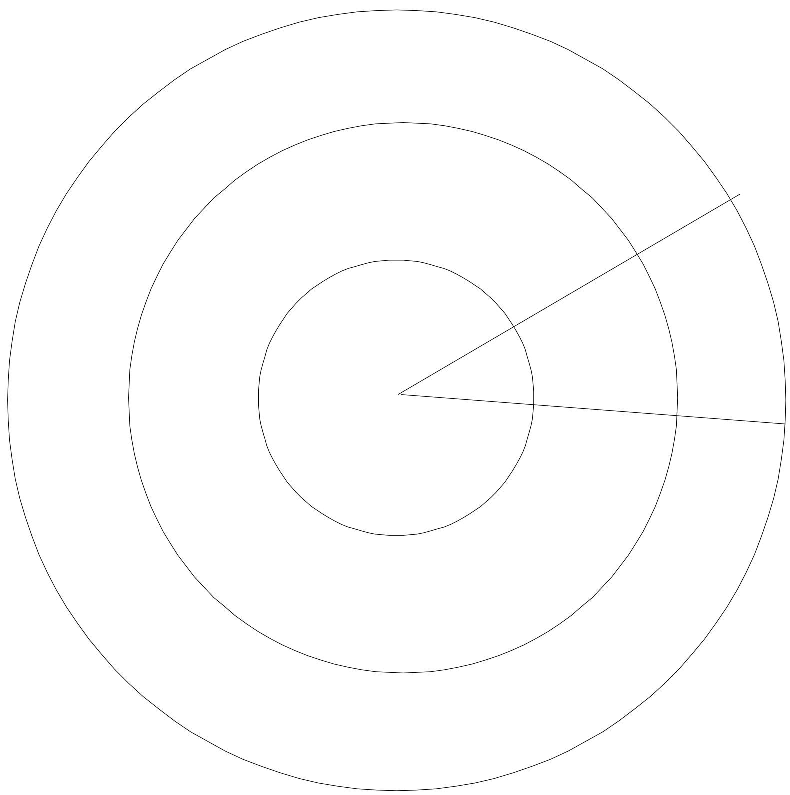
Performance Issues

- Read vs. write performance
- Small files vs. large files
- Locality Management
- Consistency semantics
    - Stability
    - Ordering
    - Atomicity

Locality Management

- Locality in space
  - File system level
    - e.g. FFS groups by directory
  - Storage level (blocks)
    - Buffer cache
    - Disk Cache
    - Strip groups
- Locality in time
  - Byte lifetimes
  - Read hit rate

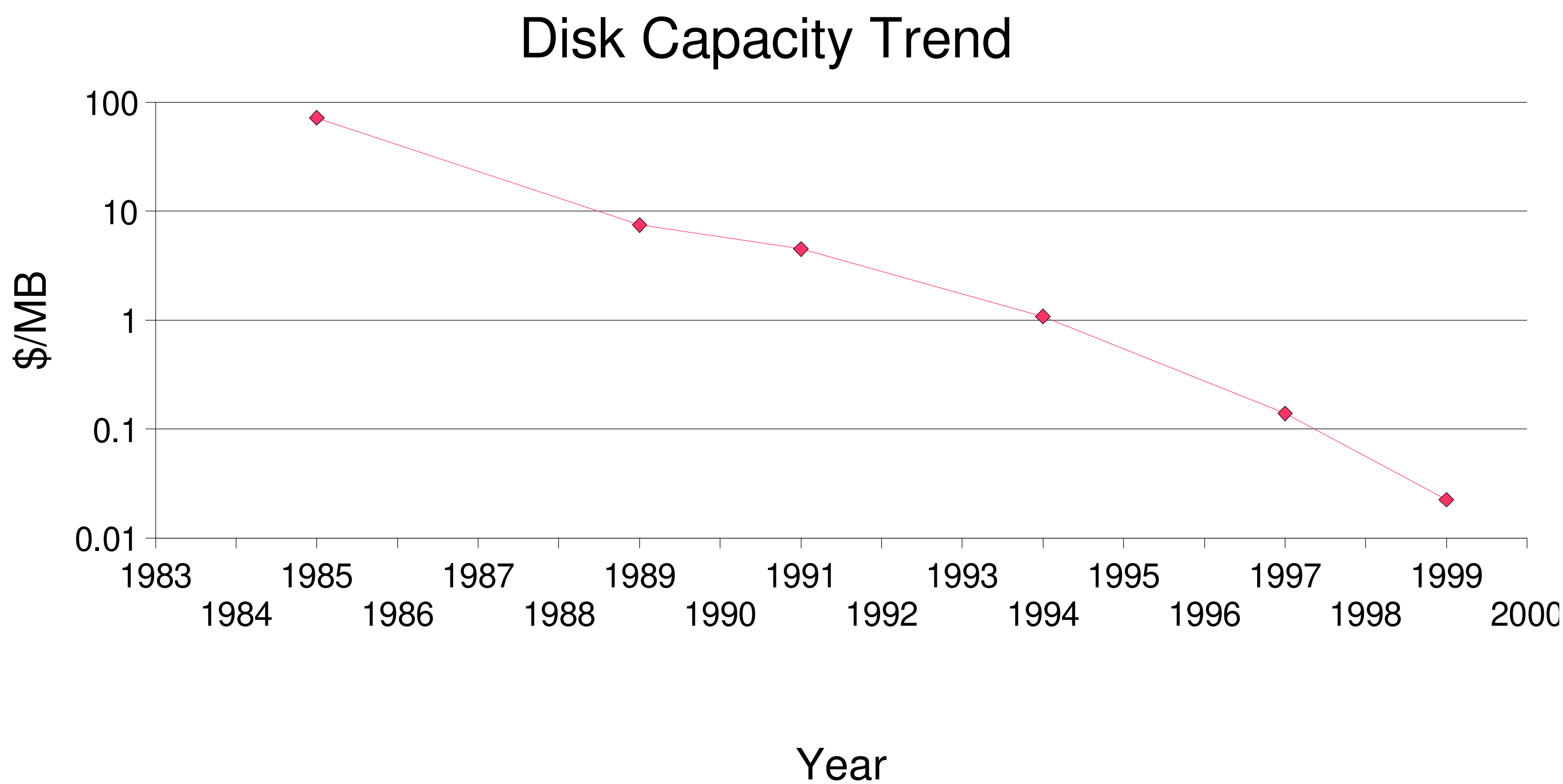
Disk Review



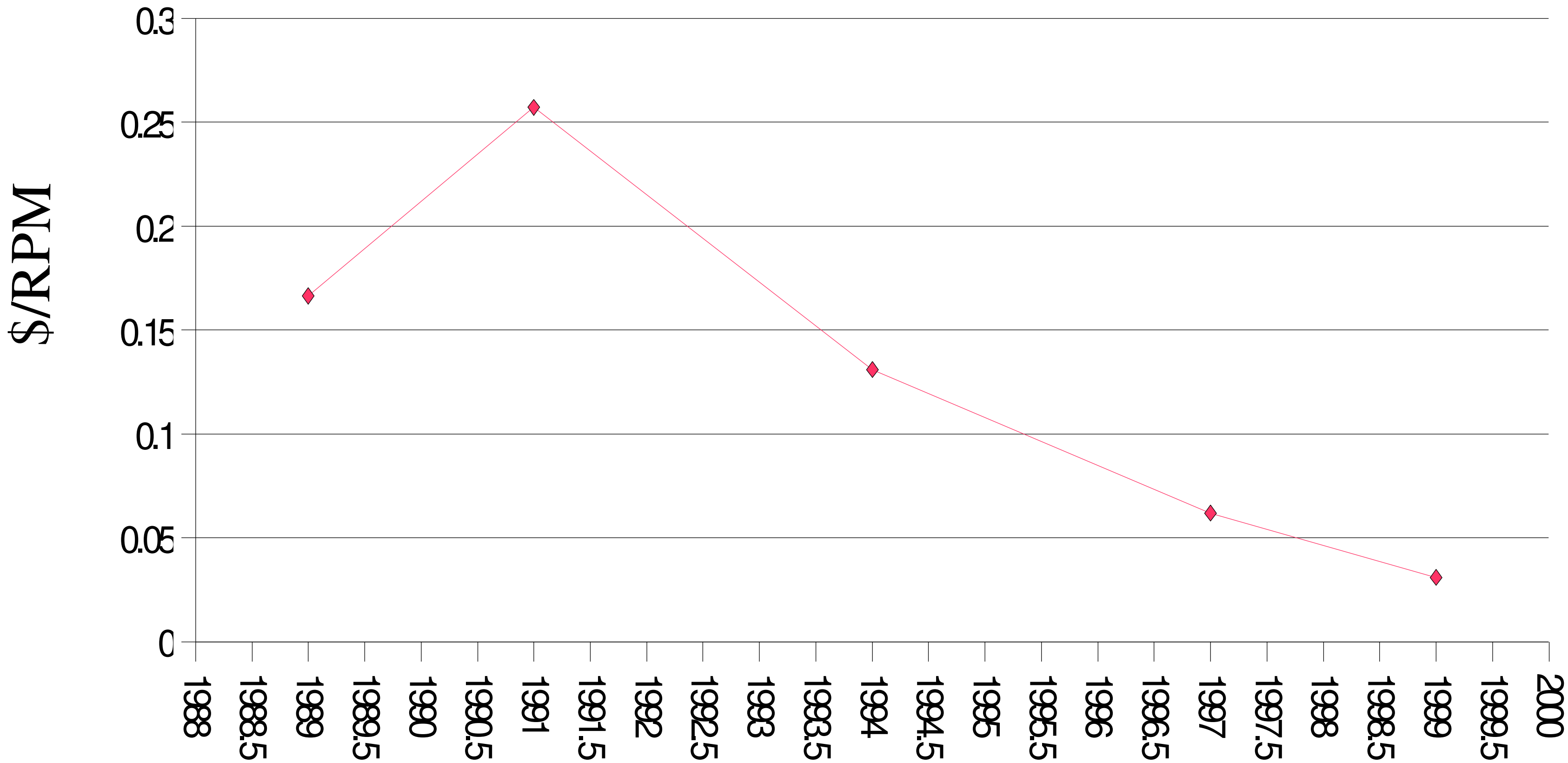
Bandwidth Scales with density and RPM

Access time scales with Seek time and RPM

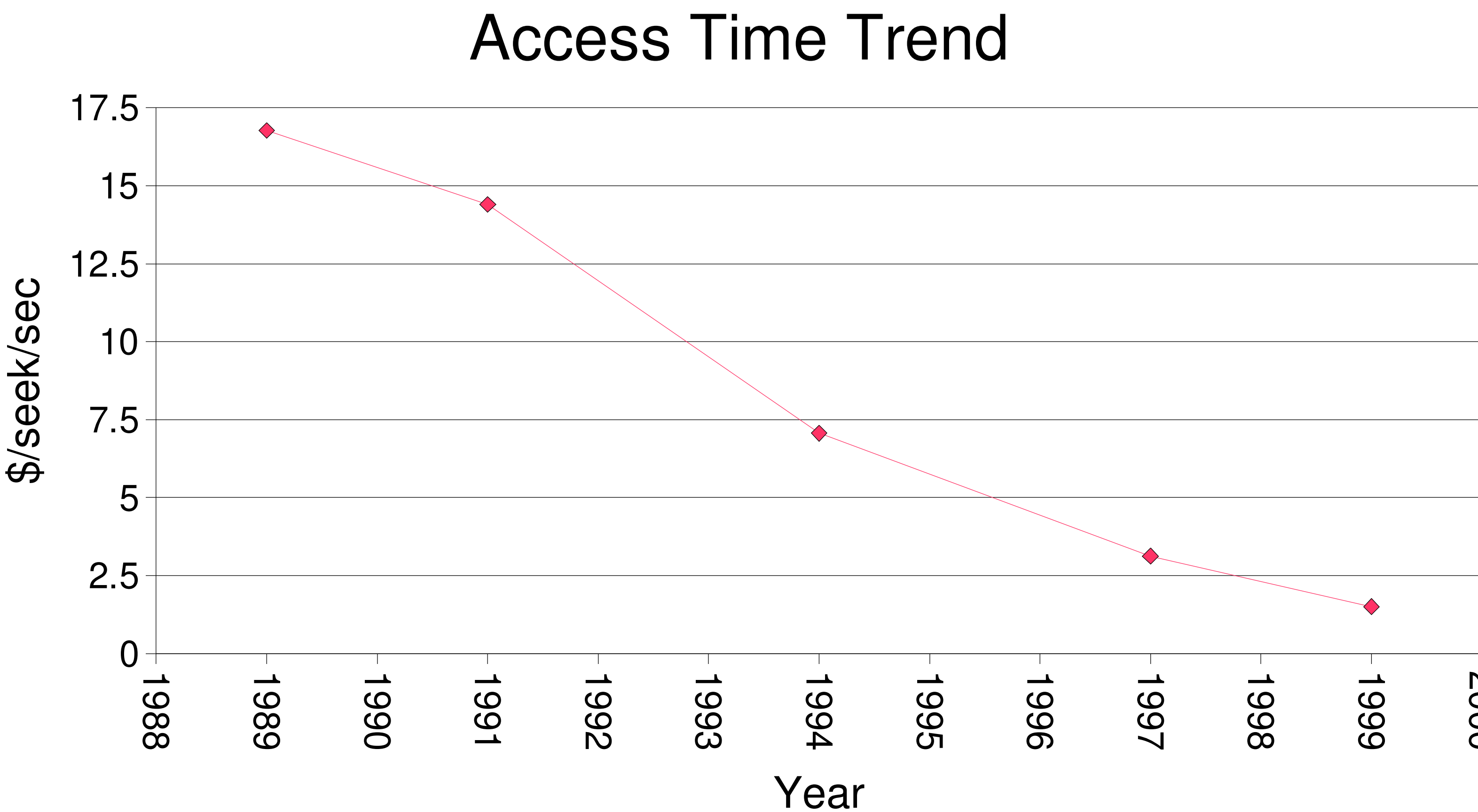
Capacity Trends



Disk RPM Trend



Access Time (seek+RL) trend



Read performance

- Seeks are ***very*** expensive
  - Mechanics scale slowly
  - Relative (seek/transfer) getting VERY high
- Bandwidth scales with capacity
  - Fundamental property of disks
  - bytes/time goes up with arial density
- How to amortize seek over more data?
- Really more work per seek

Storage Interfaces

- Very narrow (read/write blocks)
- Geometries used to be meaningful
    - Heads, Cyl, Tracks ... no longer...
- Not Anymore
    - Internal geometry hidden
    - bad blocks
    - outer tracks

Measurement Issues

- Validity of conclusions from observation
  - Workloads: Instruction, research, web?
- Writes more common than reads?
- Reads absorbed by file cache?
- Time vs. write traffic
  - How long to hold data in memory
  - In NVRAM?

Write–based FS

- Idea: writes are expensive, because must be committed to disk
- Log Structure File System (LFS)
  - cleaning overheads ...
- WAFL
  - Write Anywhere Filesystems

Writes are "easy"

- But writes are easy
- Fundamental: Asynchronous
  - Put write off until more convenient
- Temporal locality
  - bytes written likely overwritten time
  - files liked deleted in time

Reads are hard

- Fundamental architectural property
  - memory systems, file systems, networks....
- Solutions in storage space
  - Prefetching
  - multi–level caching
  - Semantic grouping

Proposals

- Provide storage device type information
  - Life expectancy
  - Read Interval
  - Semantic grouping

Swarm

- Servers maintain a unreliable sea of blocks
- Layer builds a log structure
- clients must synthesize higher structures
  - filesystems
  - shared access

Swarm Performance

- Multiple forms of Pipelining
  - Fragments asynchronous
  - compute parity as fragments written
  - Multiple fragments at once

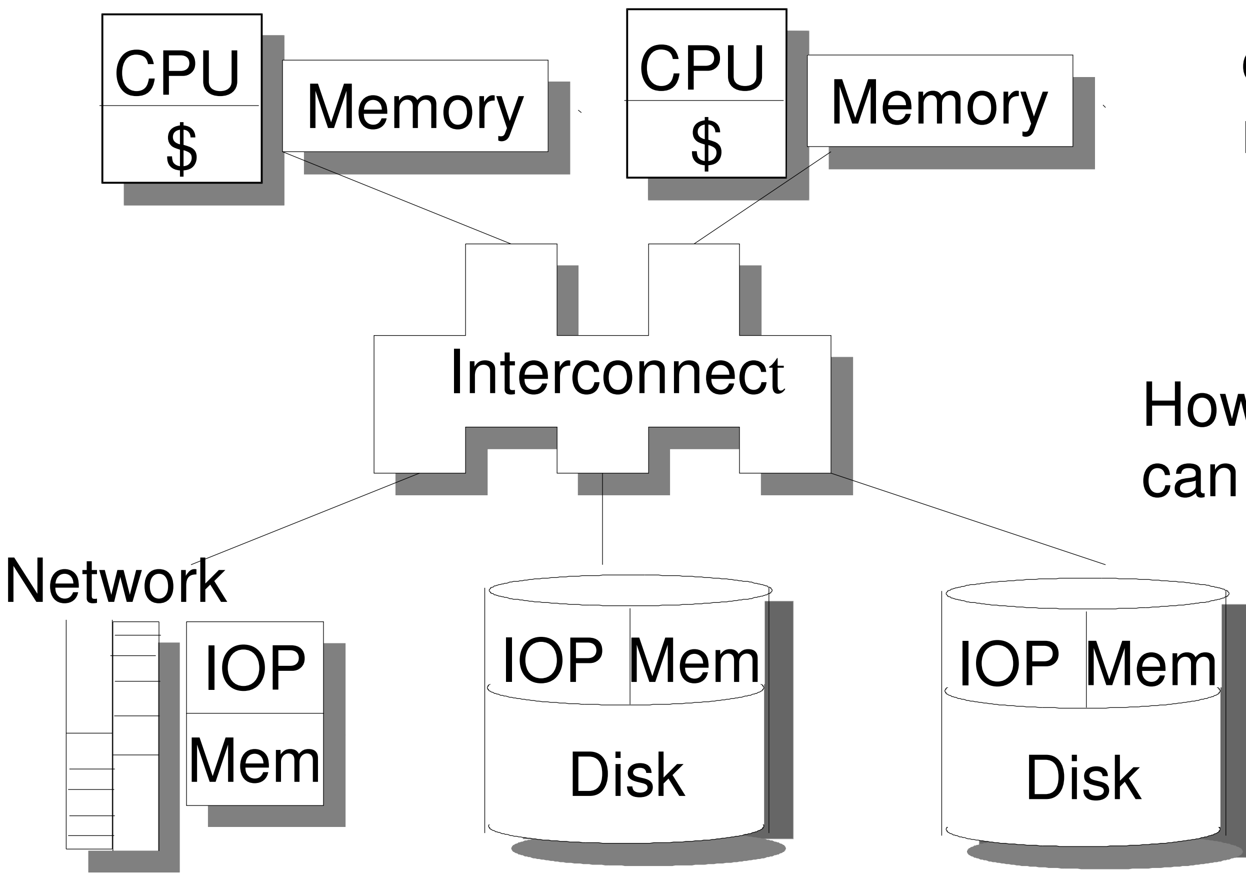
Recovery

- Combination of RAID
  - Strip groups, parity
- .. and LFS
  - Roll back to last checkpoint
  - Must have checkpoints in practice
  - Not really "just an optimization"

GFS

- Idea: Shared disk architecture
- Sharing FS structures is hard
- Push more FS structures into devices!
  - Locks
  - Cache

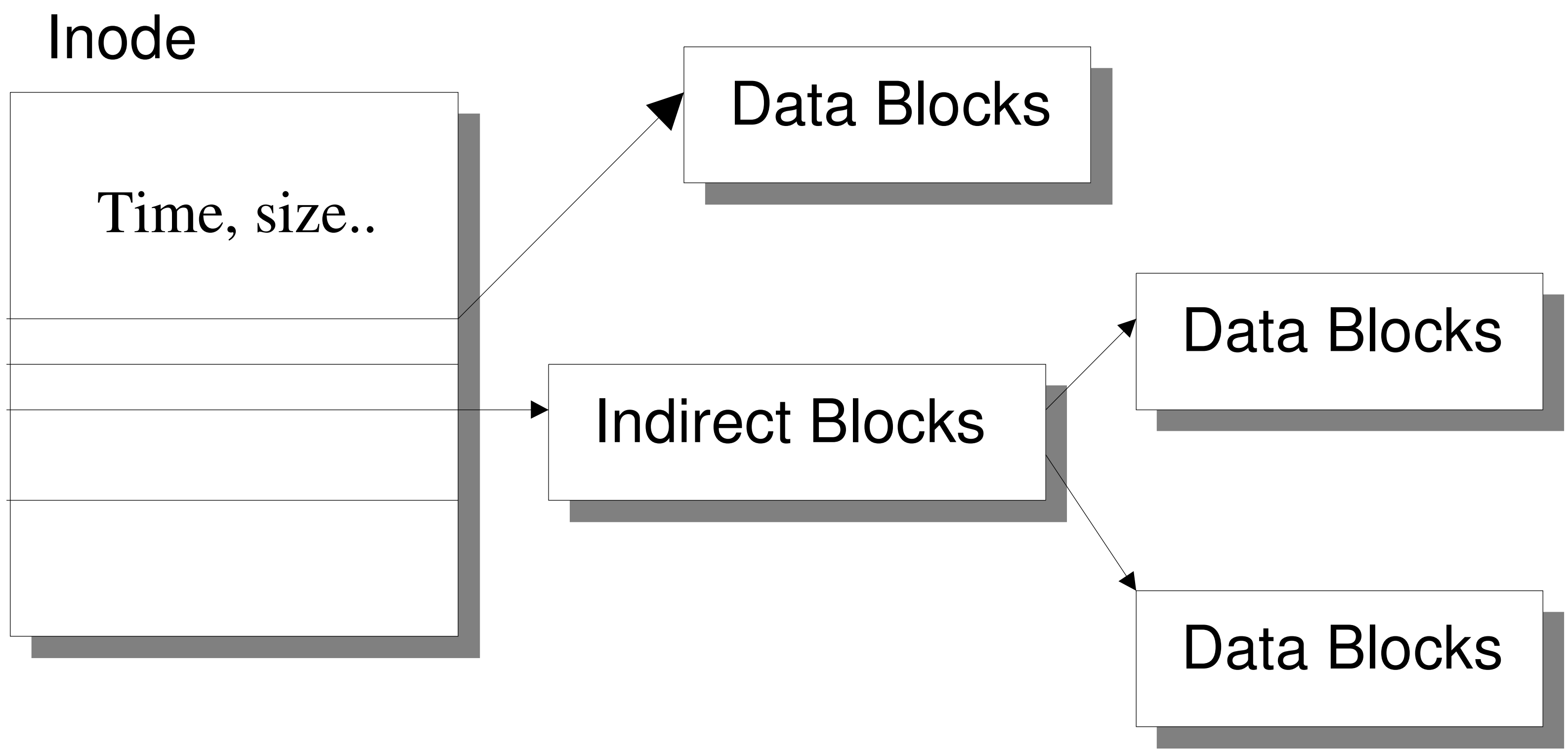
File Cache



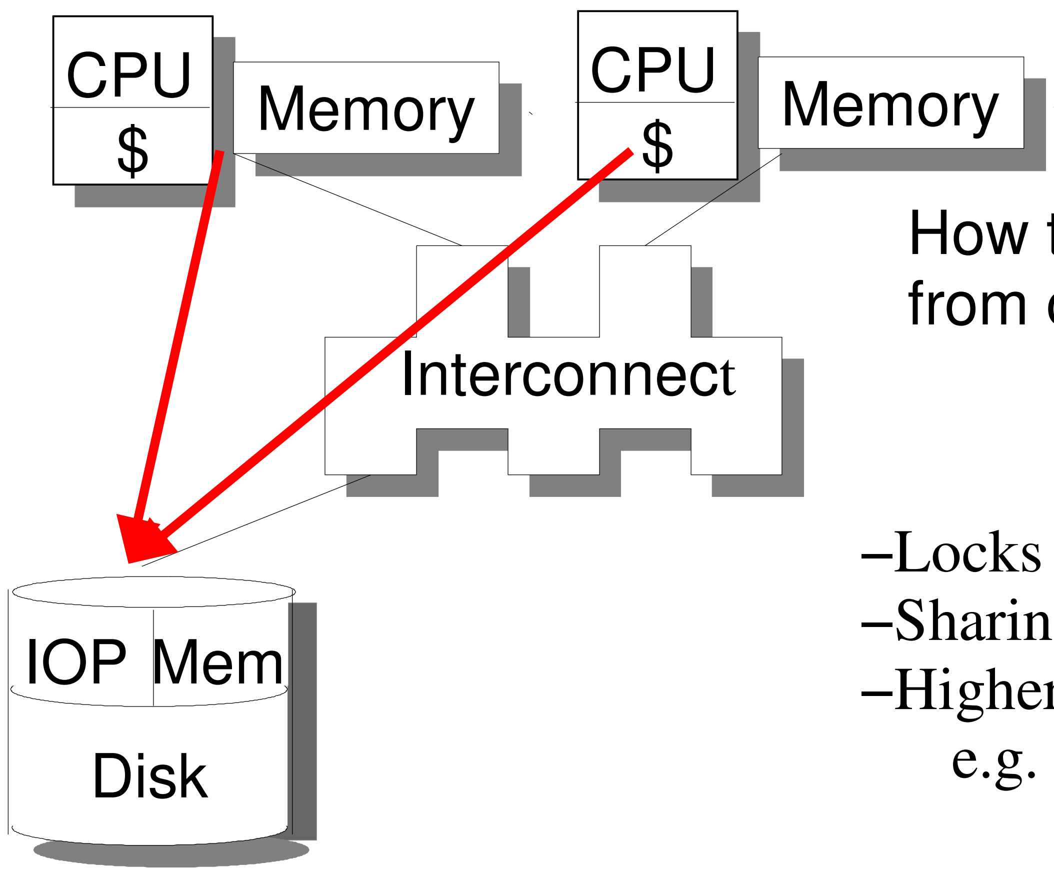
How much cache can you hold in main memory?

How much cache can you hold here?

File System Metadata



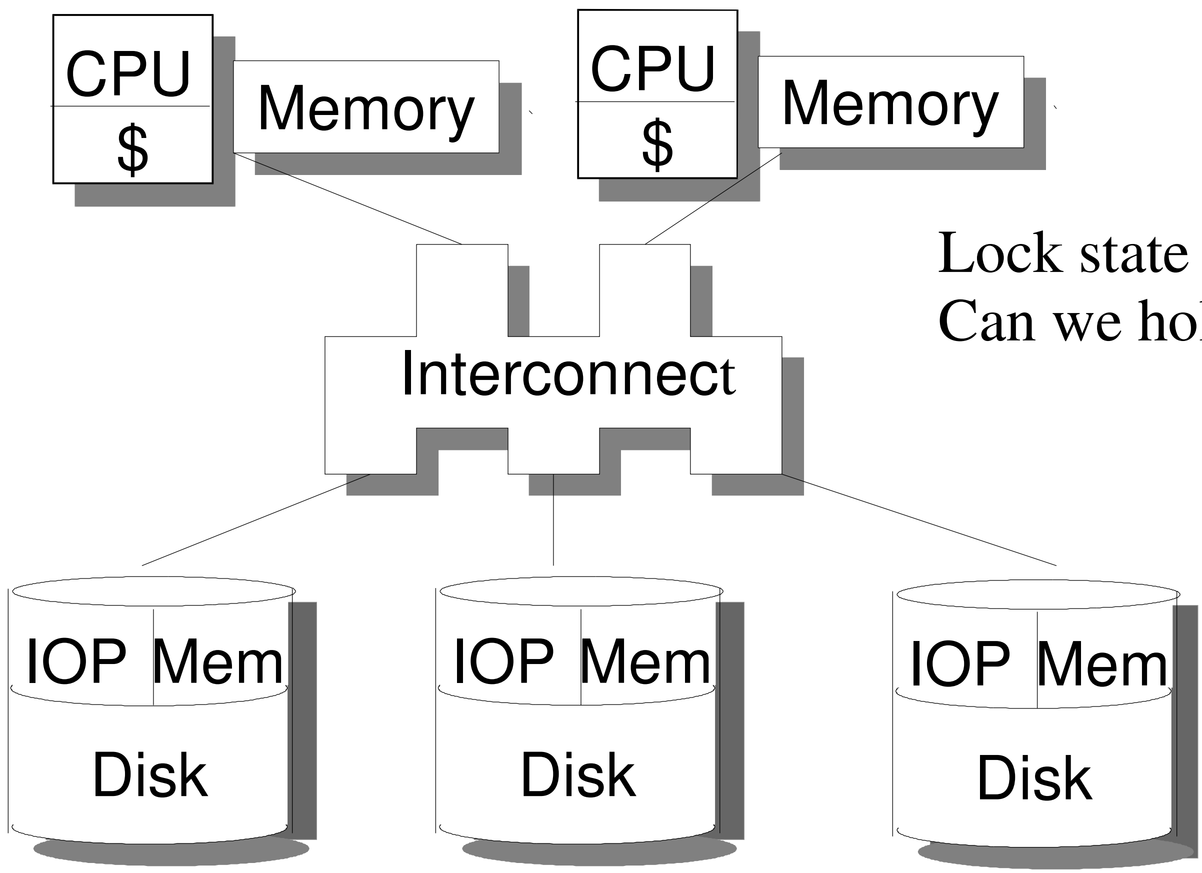
Sharing Problem



How to protect metadata from concurrent access?

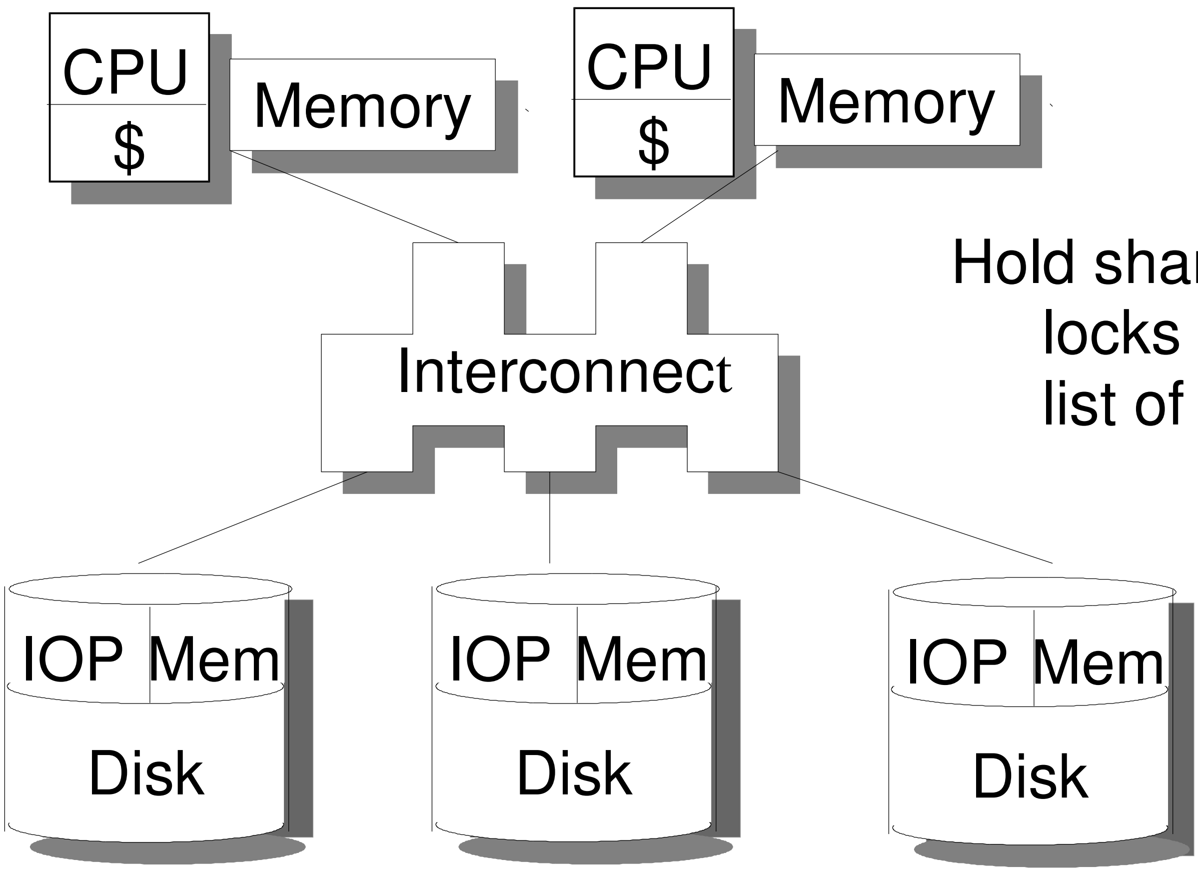
- –Locks
- –Sharing lists
- –Higher level operators
  - e.g. Whole files

GFS Solution: Locking



Lock state is very small
Can we hold it in the devices?

Federated Alternative



Hold shared state in devices
- locks
- list of reads/writers