Federated Systems

Lecture 12

File Systems

Richard Martin
November 30th 1999

Today

File Systems

- xfs (Anderson, Dahlin, Neefe, Patterson, Roselli, Wang)
- Frangipani (Thekkath, Mann, Lee)

Topics

- Semantics
- Scalability
- Recovery
- Engineering

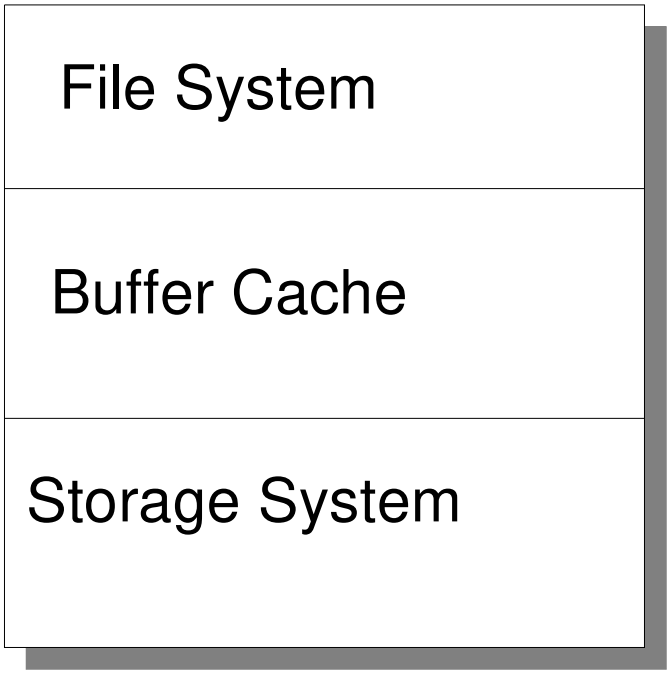
Today

- File system as separate from the storage interface
- What should the file system do?
  - What are the clients of the FS?
  - What needs do they have?
  - How to store/manage data?

Recall Storage Layers

My program

`Read()/Write()`

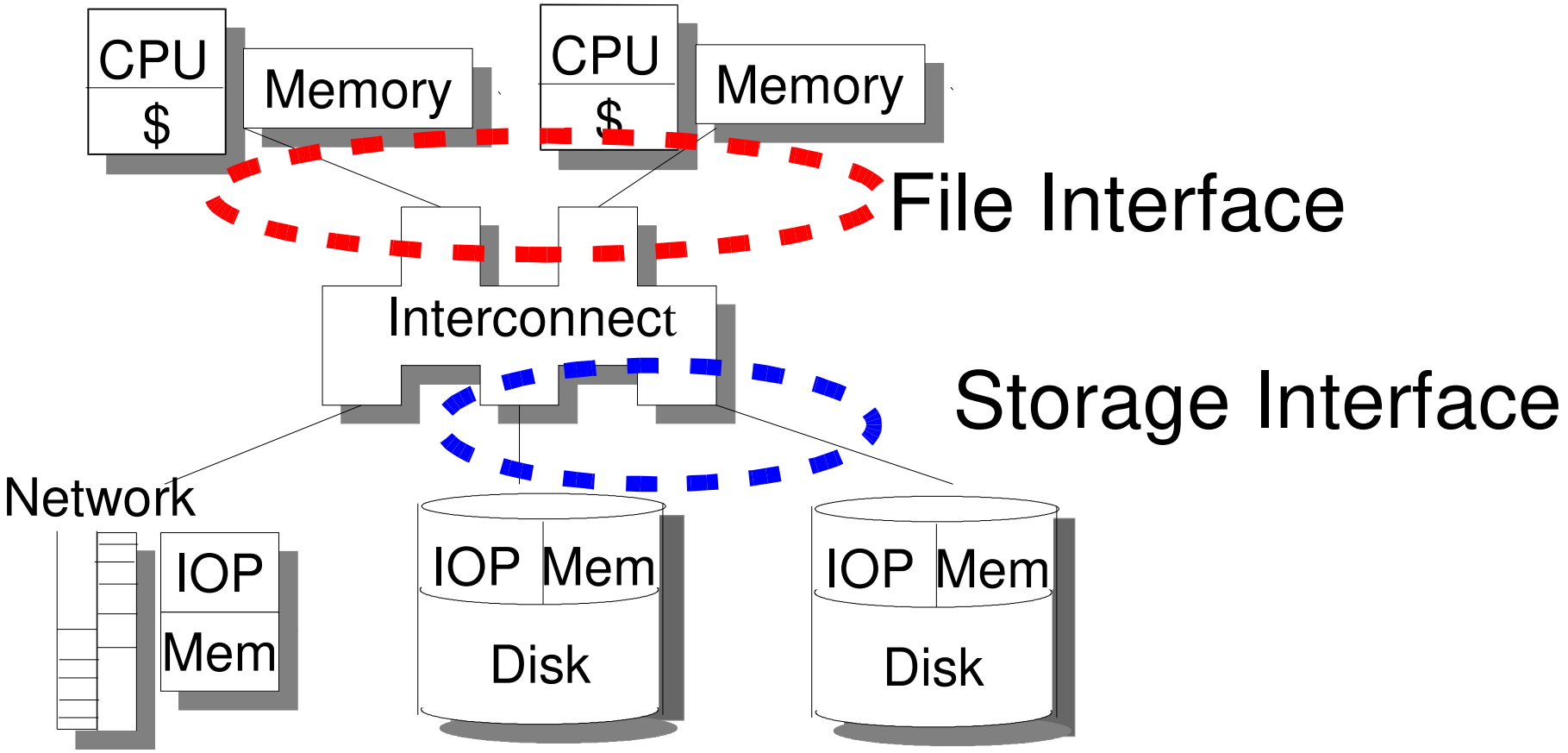


Directories, files, locks

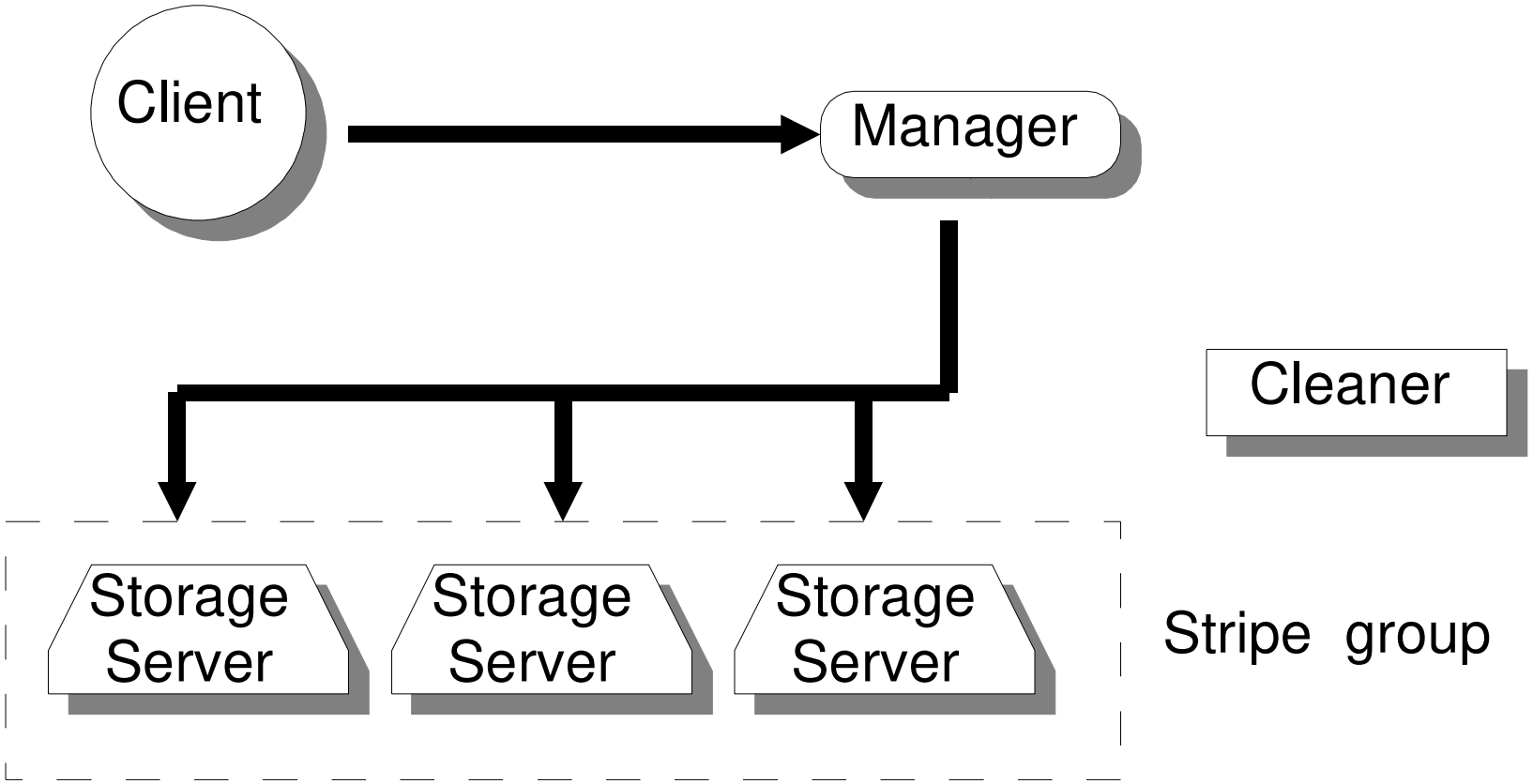
Block–level cache

Block–level Storage

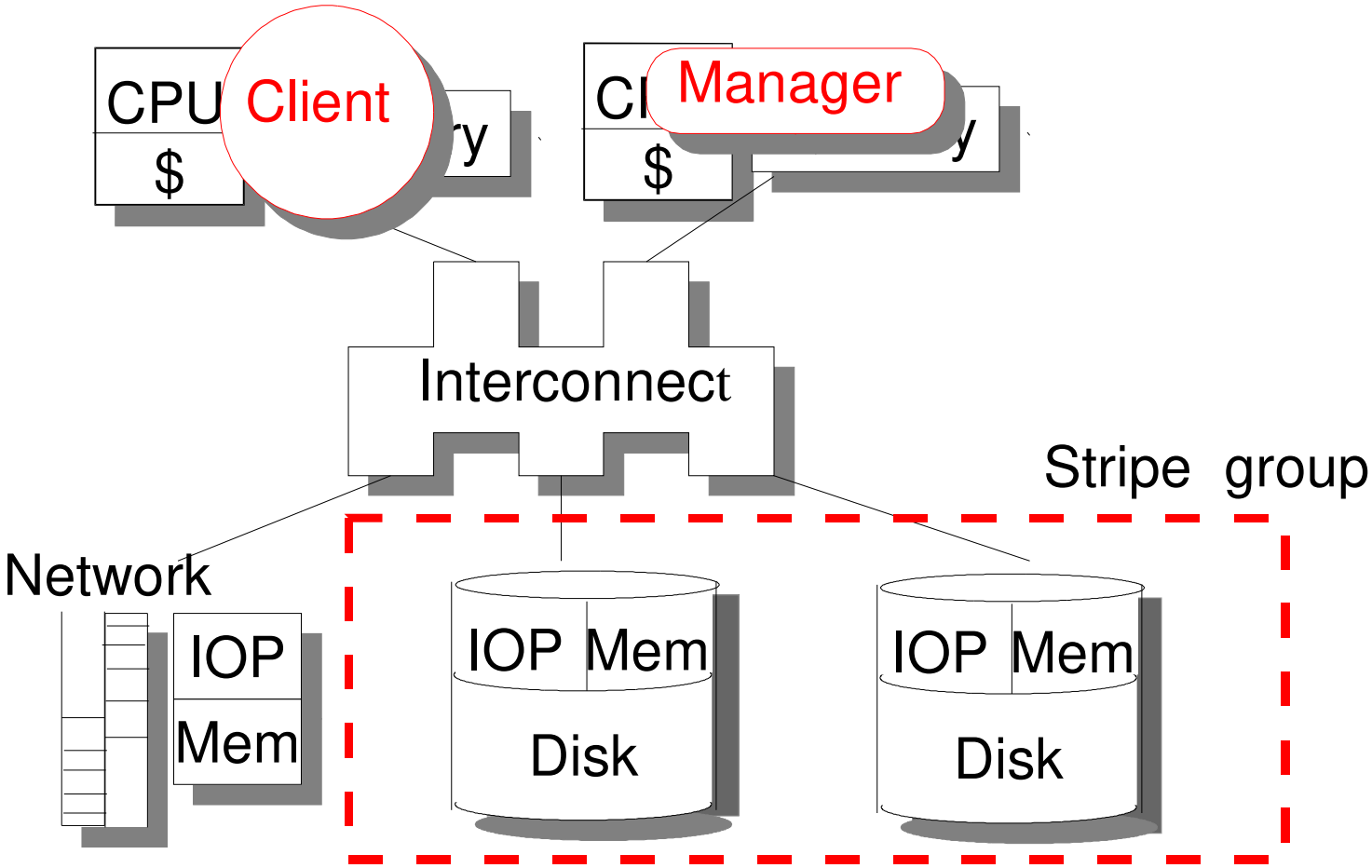
Model Federated Architecture



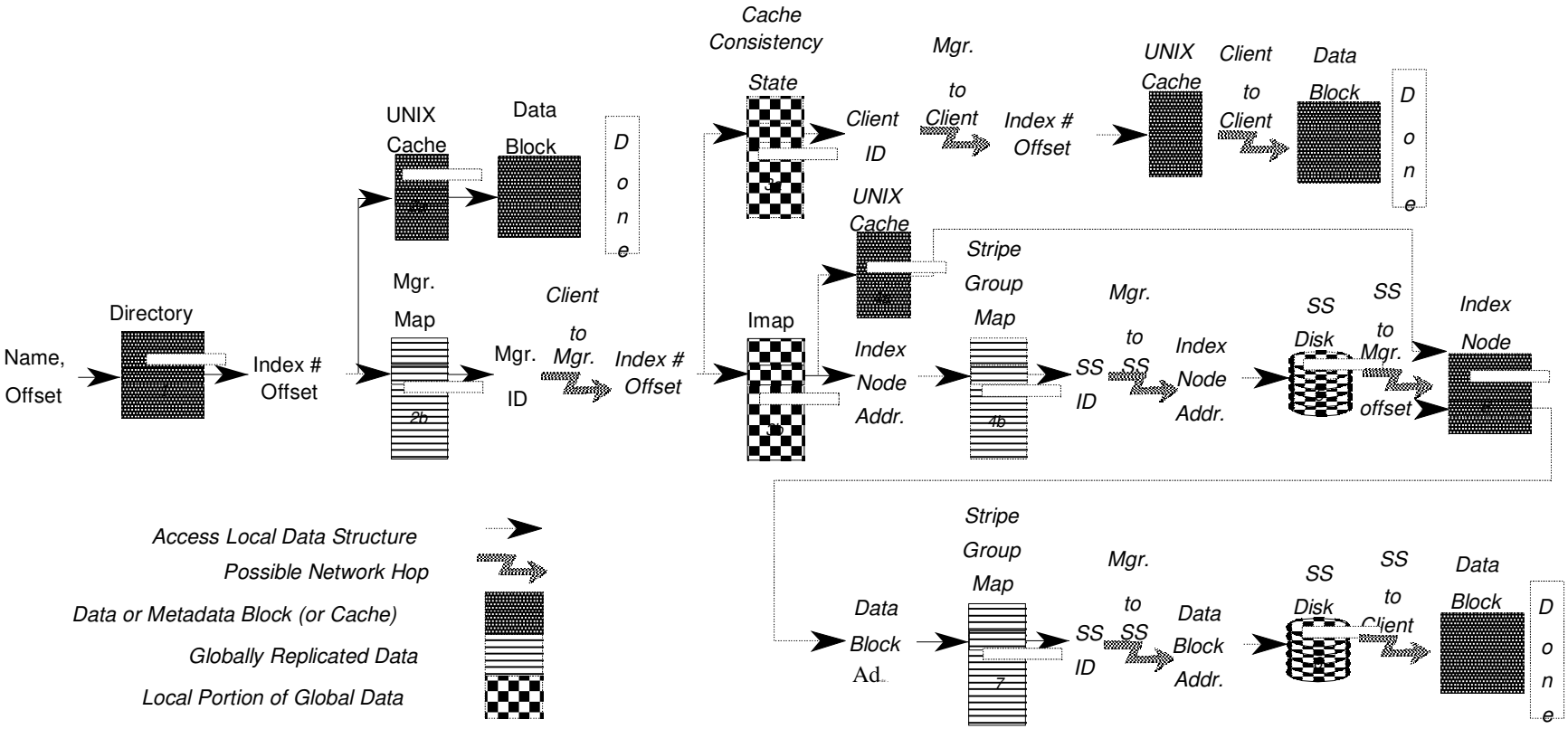
XFS Universe



xFS Fed. Architecture



xFS Dataflow



Frangipanni

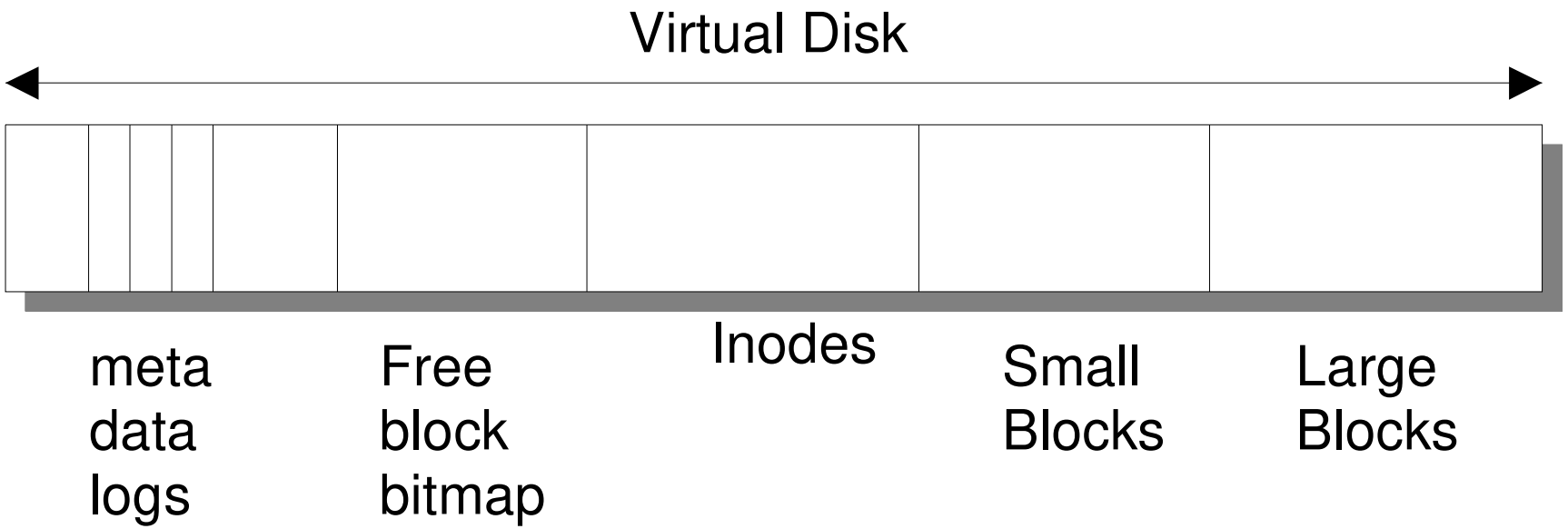
Global FS built using stock FS ideas

- uses per–file locking to reduce contention
- build the FS structures on top of

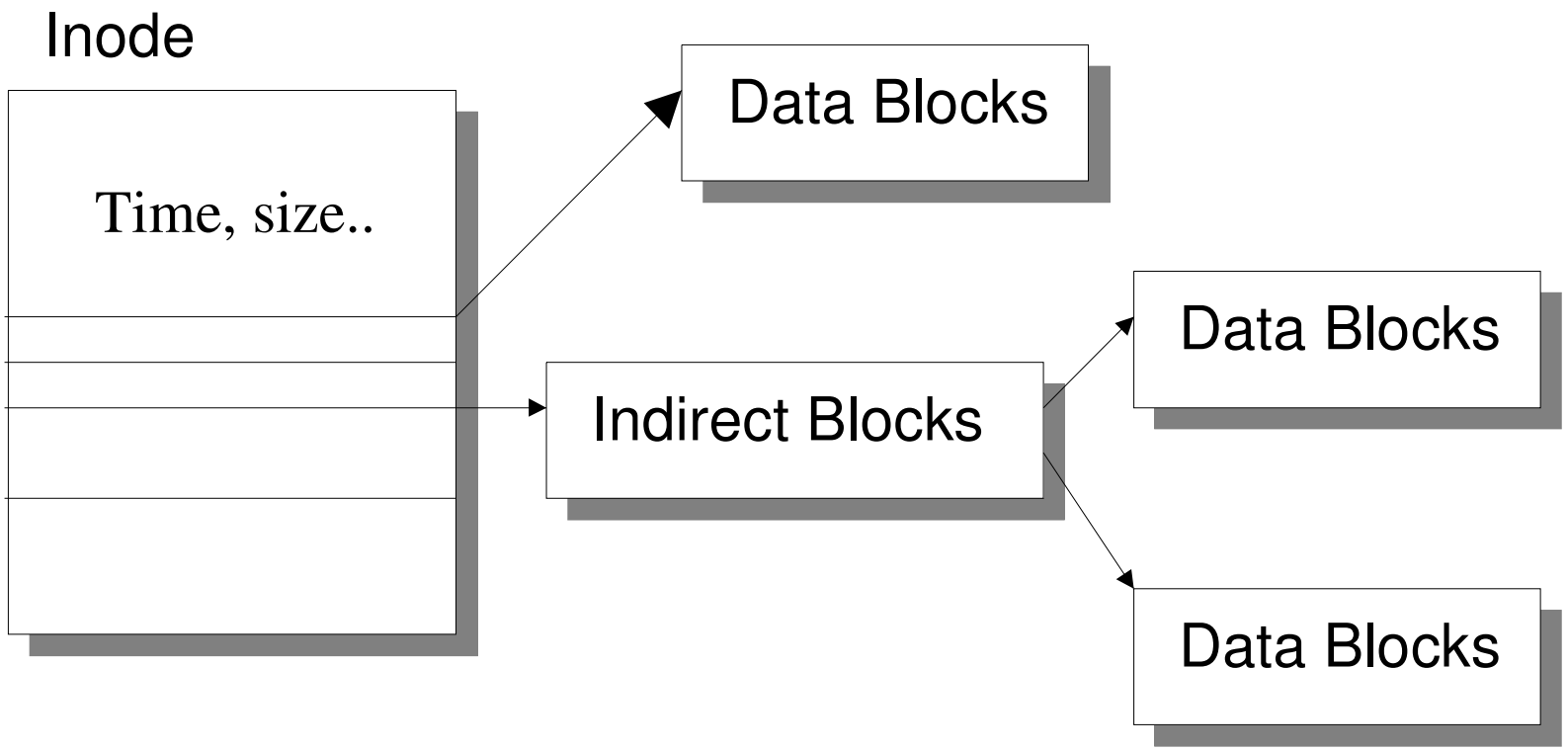
Issues:

- How to partition up disk address space?
- Recovery?
- Co–operation between kernels?

Frangipani Layout



File System Metadata



Scalability

Scalability in the FS context?

- Add resources means FS can grow?
- Human intervention in the FS needed?
- Can the files grow as resources are added?

xFS scalable at the block level

- can use one huge file in a program

Frangipani design limits scalability

- But semantics aren't any worse then regular FS!

Recovery

Frangipani:

- Meta–data logged (just like a regular FS metadata log)
- Detect downed server

xFS:

- layered recovery:
  - Storage servers–> managers –> cleaners
- storage servers first recover locally, then group
- managers use LFS style roll–forward

Engineering

xFS vs. Frangipani

- xFS: Multi–functional partitioning
- Frangipani: pure vertical partitioning

Which one is easier to implement?

- Frangipani's reliance on petal global block abstraction simplifies design
- xFS's functional partitioning
- Which style of design is easier to maintain?

Next Week

No reading

Questions for next time:

- Will I/O processors be viable beyond current uses?
- Which architectures (SMP, Cluster, Federated) will be used in which contexts? Why?