Everyone Loves File: File Storage Service (FSS) in Oracle Cloud Infrastructure

Matteo Frigo, Bradley C. Kuszmaul,
Justin Mazzola Paluska and Alexander (Sasha) Sandler

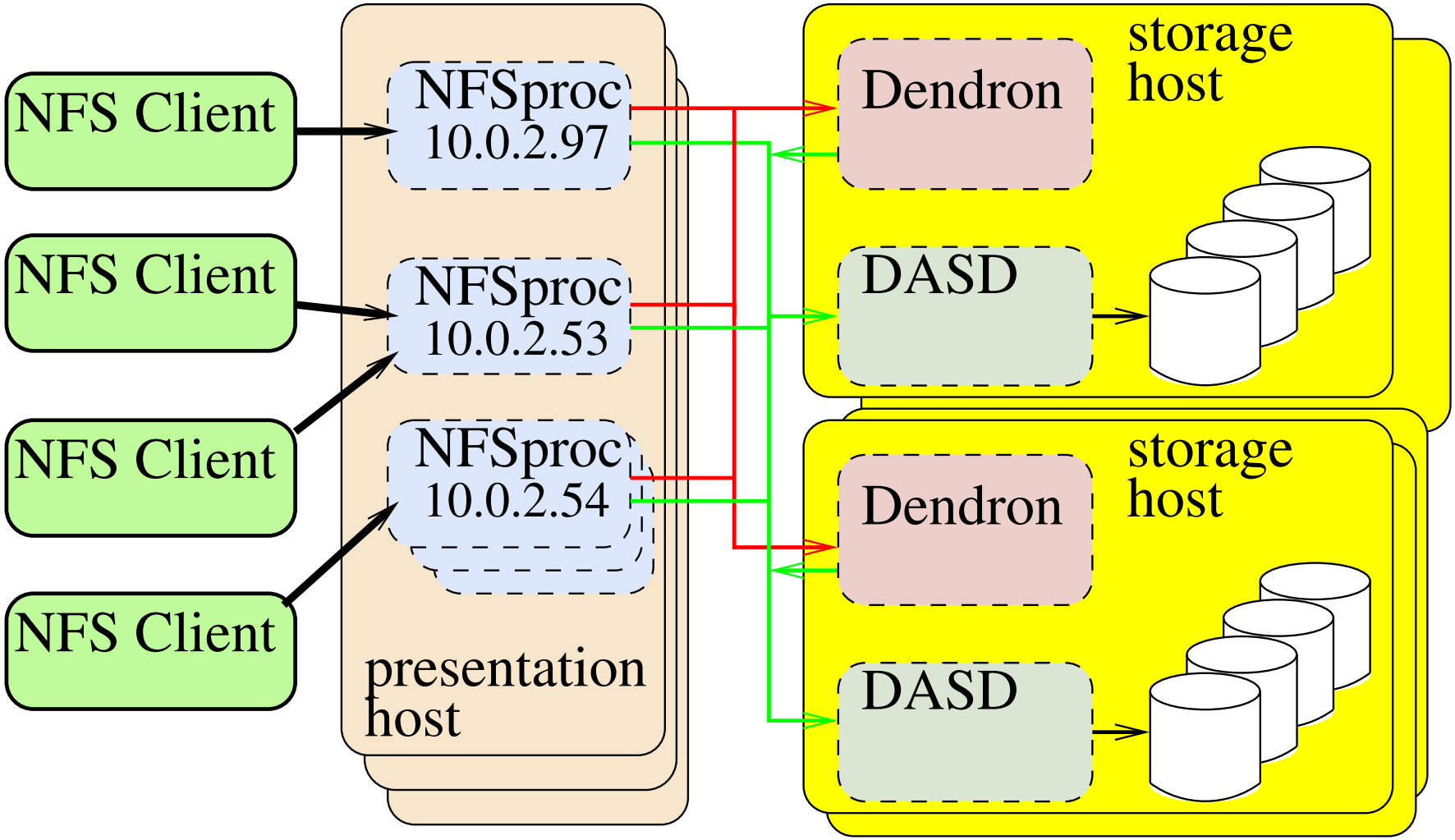
(and many others)

2019-07-10, USENIX ATC

What is File Storage Service

- Oracle operates a cloud.
- We set up a file system for you.
- The file system appears as IP address in your virtual private network. Behind that IP address is a Network File Service (NFS) server.
- You rent NFS clients from us.
- You pay for what you use (some price per gigabyte month).
- You start with 0 bytes, and can grow to as many petabytes as you want.

Architecture of File Storage Service



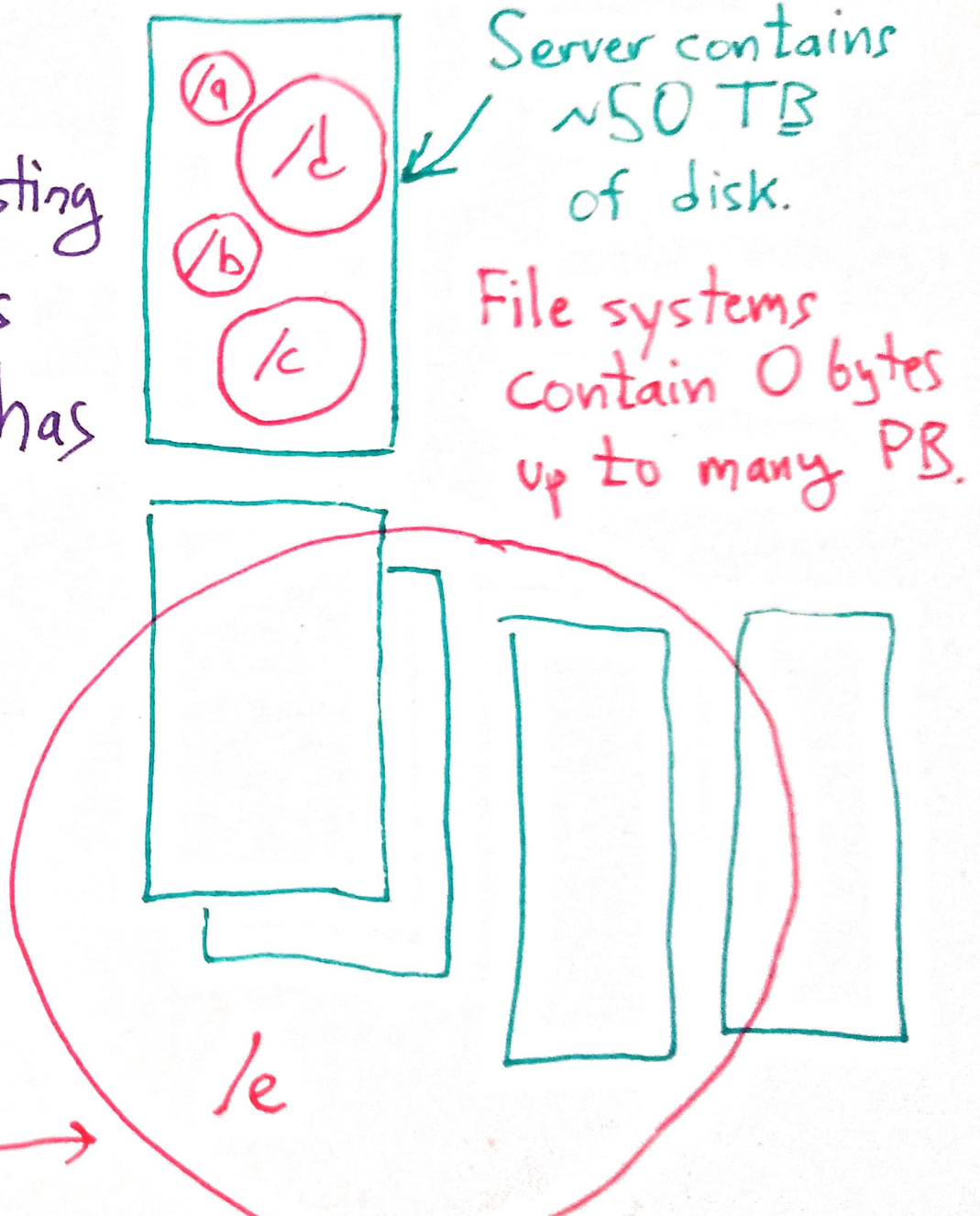
Simpler Idea: Use Standard File Servers

Idea: Deploy servers, each hosting some filesystems each of which has an NFS server.

Migrate filesystems as they grow.

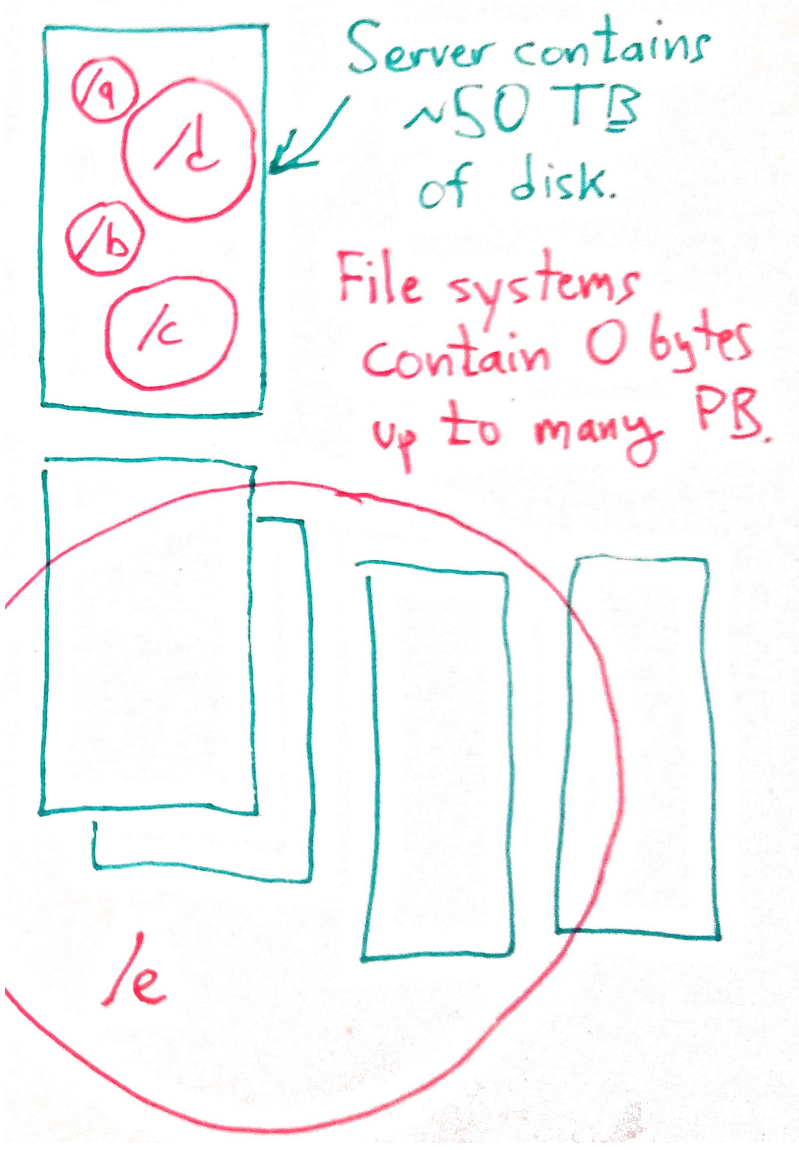
Big filesystems...???

a big file system

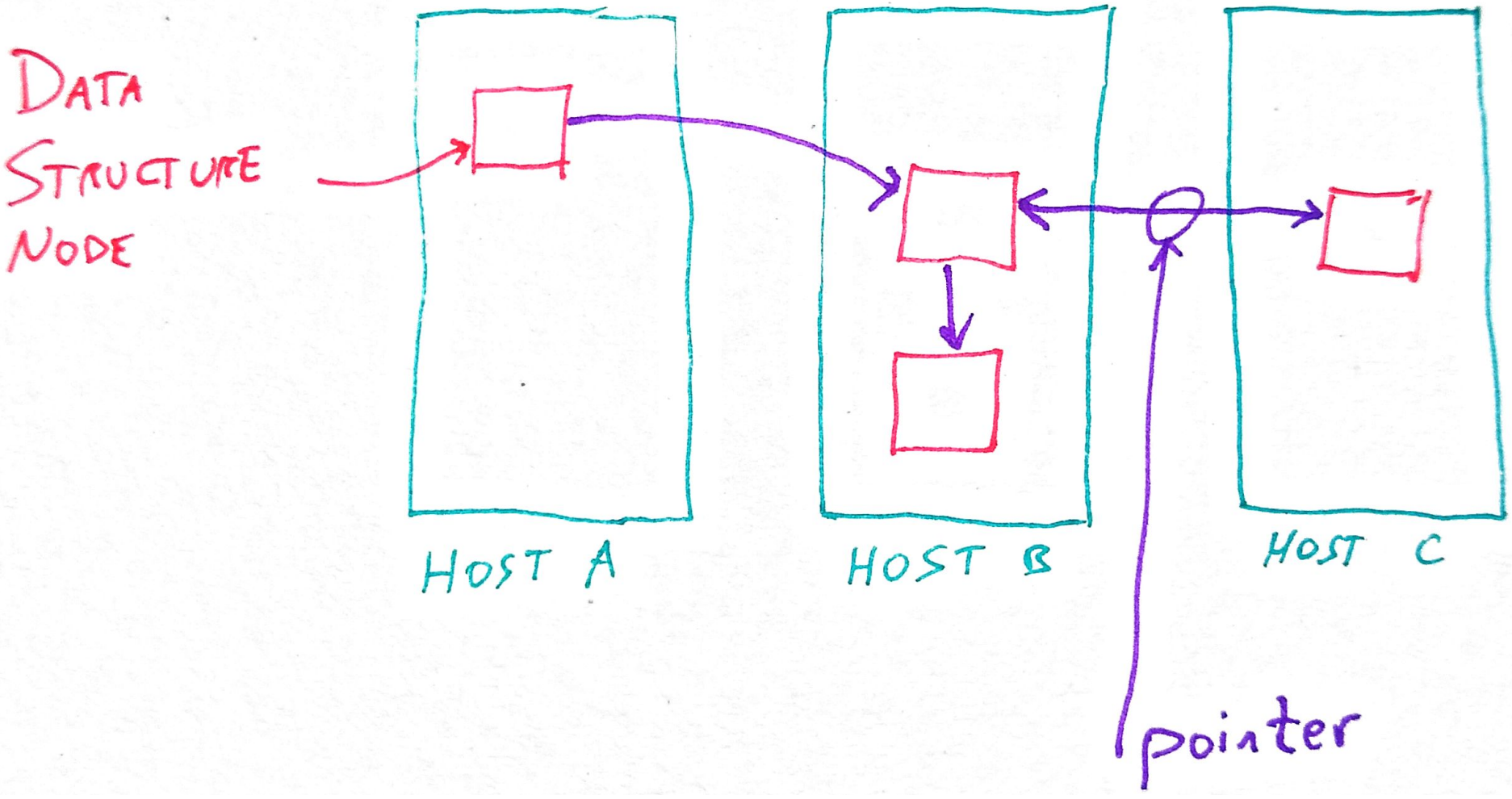


Problems with Standard File Servers

- Failover and Replication.
- Pack several file systems per server, and migrate them as they grow.
- How to handle big file systems? Somehow filesystems must scale to be larger than a single server.



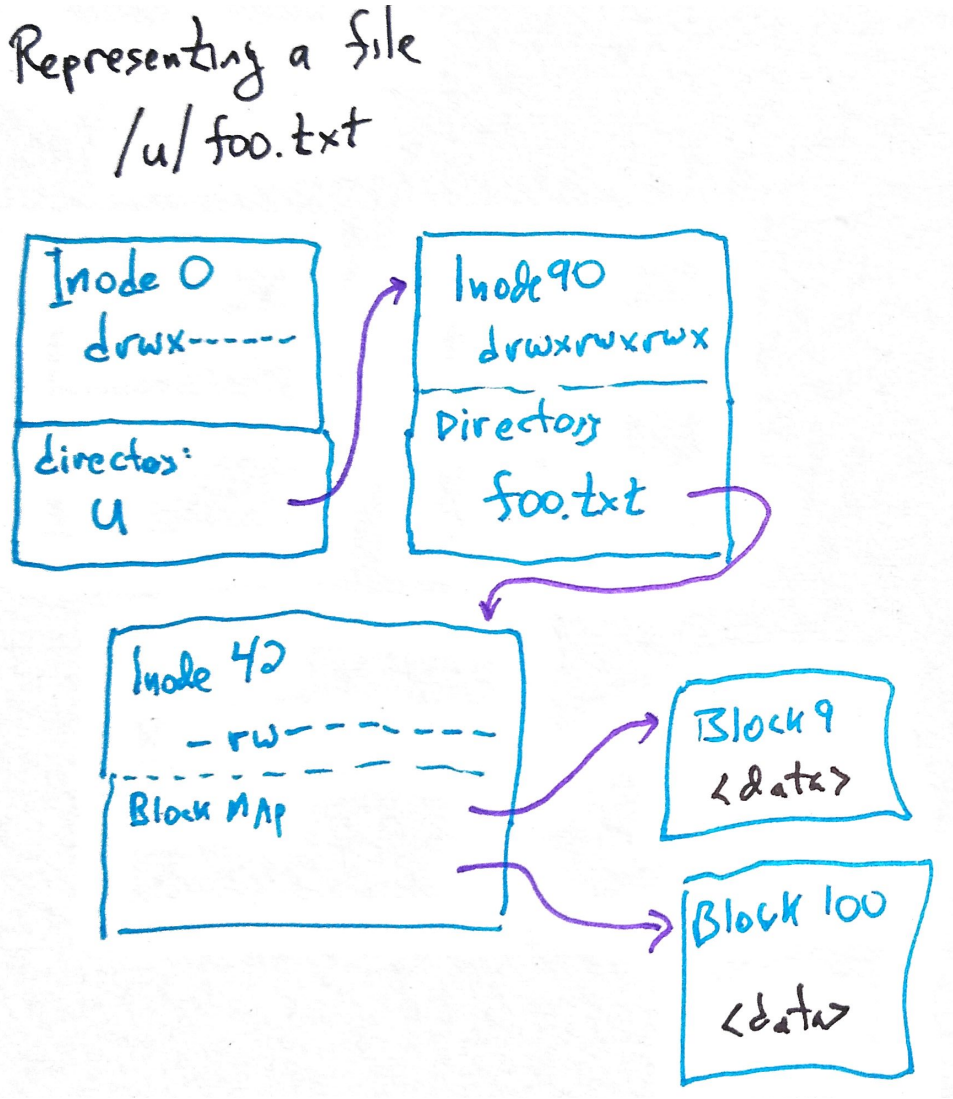
Scale Up: FSS Distributes Data Structures Across Servers



What Data Structure Do We Want?

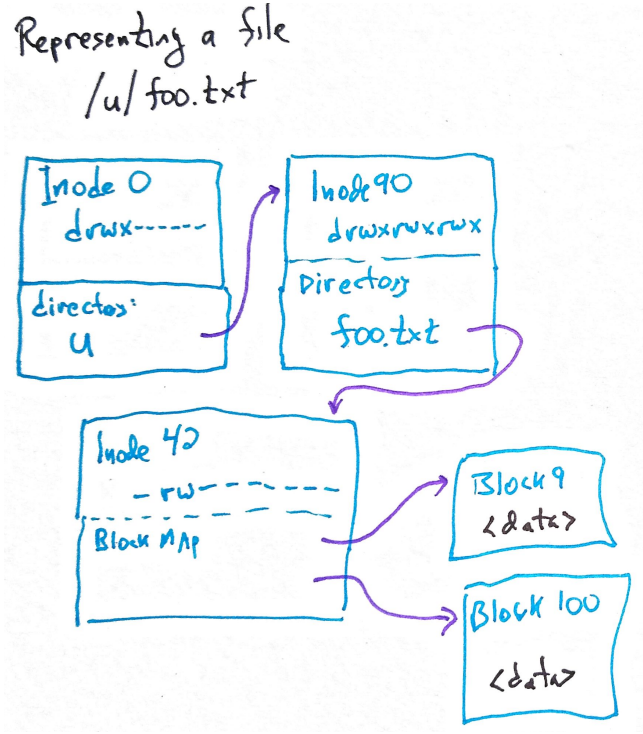
Inodes (Ritchie and Thompson [1972])

- Each file or directory is represented by a numbered inode.
- Directories include a mapping from names to inumbers. Need a data structure for the directory.
- Files include a mapping from offsets to block numbers. Need a data structure for the block map.



Represent Inodes as Tabular Data

instead of implementing the inodes directly.

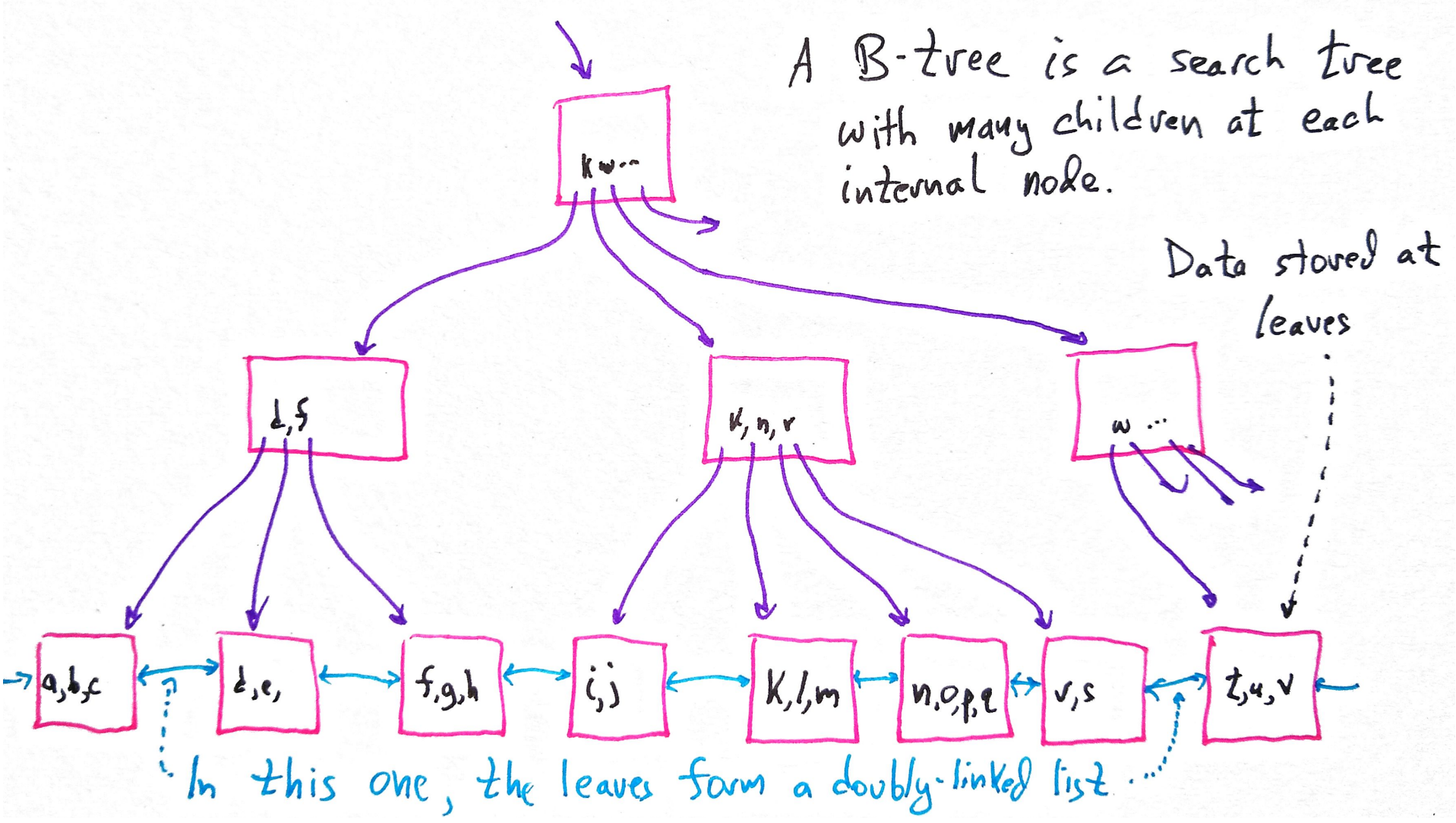


Inode Table		
inumber	permissions	owner
0	`drwx------`	bradley
42	`-rw-r--r--`	bradley
90	`drwxrwxrwx`	bradley

Directory Table	
inumber,name	inumber
0,"u"	90
90,"foo.txt"	42

Block Table	
inumber,offset	inumber
42,0	9
42.4096'	100

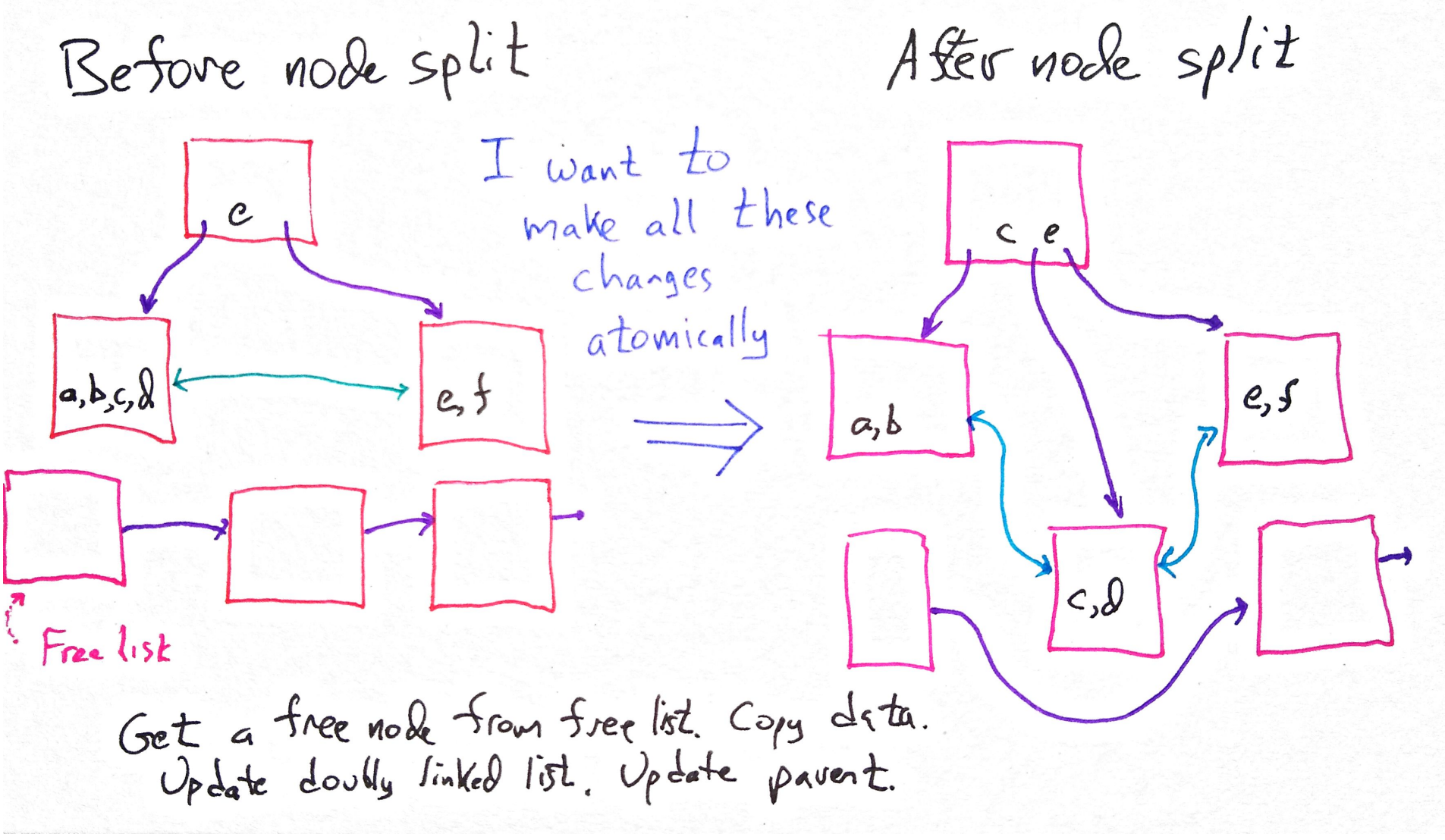
Store the Tabular Data in a Distributed B-tree



Only One B-Tree

- All the filesystems are stored in a single B-tree.
- Smaller filesystems simply use fewer key-value pairs in the B-tree.
- Even small file systems end up distributed across many servers.

Updating the B-tree Requires Atomic Operations Across Servers



Two-Phase Commit (2PC)

The classic strategy [Gray78, Lampson80] for building a distributed data structure. If several servers participate in a transaction, one machine acts as the ***coordinator***.

1. Each participant records its part of the transaction, and it marks its subtransaction as **prepared**. It can either roll it forward or back.
2. When the coordinator hears that all have prepared, it marks the transaction as **committed** (or **aborted**).
3. Participants are told to roll their transactions forward (or back). If a participant misses the notification, it later asks the coordinator what happened.

The Problem with 2PC

Q: What if, just before it marks a transaction **committed**, the coordinator crashes and doesn't ever come back?
Or maybe the crash was just after the transaction **committed**.

A: The clients cannot tell which case it is, and they get stuck forever.

Paxos Doesn't Get Stuck

I'm not going to explain Paxos in detail.

- Paxos [Lamport 98] can be used to come to a consensus on a single value.
- Multi-Paxos can be used to come to a consensus on a log.
- Once you have a log, you can implement an arbitrary state machine.
- Paxos handles messages getting lost or duplicated and servers crashing at inopportune times, without getting stuck.

FSS Uses Paxos to Implement two-phase commit

- Each participant in two-phase-commit is implemented by a replicated state machine.
- The participants are nonstop, since they are replicated.
- A replicated participant is called an ***Extent***.

How Big is an Extent?

- The state of an extent must fit onto a single server. So not too big.
- The overhead of the state machine is large (to implement two-phase commit). So not too small.
- We size the extents so that hundreds of them fit onto a server.
  - Each extent manages several gigabtyes of disk.
  - The extent's state machine is a few megabytes.
  - Extents are small enough to move around for load balancing and to provide parallelism for failure recovery.
  - Extents are 5-way replicated.

Multipage Store Conditional (MPSC)

- On top of 2PC, We program FSS using an optimistic concurrency style.
- ***Multipage store conditional***:
  - Read up to 15 pages into memory, obtaining for each page a ***version tag*** that is guaranteed to change whenever the page is modified.
  - Compute new values for the pages.
  - Present the new pages along with the previously obtained version tags to the MPSC system.
  - The new pages are all written atomically (and only if none of the read pages have changed).
  - If the MPSC operation fails, no changes are made.
- MPSC operations are linearizable.
- An MPSC operation is a limited transaction: Not too big, not too small. Lock-free style.

A Simple Throughput Model

- Each storage server provides some disk and some network bandwidth.
- For writes, every byte is transmitted 5 times.
- So divide all the network bandwidth by 5. That is a peak "not-to-be-exceeded" speed.
- Queueing theory says you cannot run at 100%. We find we can run at about 1/3 of peak.
- The simple model: THe bandwidth is 1/15th of the network bandwidth.
- Surprisingly this simple model seems to work for all workloads we've seen.

Looks Like a Speedup Curve