Recitation 16: ZFS

MIT - 6.033

Spring 2021

Henry Corrigan-Gibbs

Plan

* Opening: File system
* Recitation questions
* The problem with the status quo
* ZFS - "The Last word in file systems"

Logistics

* S^3 is available for you!
* No recitation next week. (Project presentations.)
* MapReduce hands on due 4/22
* Holidays on Mon & Tues!
  ↳ Enjoy your time off.

What is a file system?

- The system that manages mapping of files & dirs on to disk.
- Examples so far: GFS, Unix, your laptop

What makes file-system design difficult?

* A crash can happen at any time
* All important data is stored there
* Limits the performance of your system in many cases

Paranoia, caution, aggression all in one!

Recitation Questions

1. What aspects of the Unix file system was ZFS designed to overcome?

   * Hard to administer — manual process
   * Lack of virtualization
     ↳ Not flexible
   * On disk consistency
   ⋮

2. How is ZFS designed to overcome these issues?

   * malloc for disk
   * checksums
   * copy-on write w/ atomic rename
   * POSIX - layer API

3. Why is it important for ZFS to overcome those aspects? Why is GFS not an adequate solution?

   ⇒ More disks, more space, more complexity!

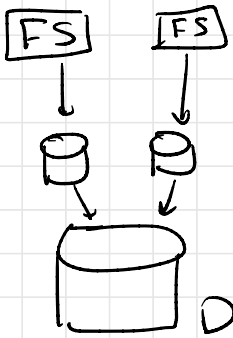
   - GFS is for a distributed system
   - Here, we are storing data on local disk

Demo: Standard FS maintenance

- Open gparted.
  - ↳ Inspect partitions — small disk
  - ↳ Create a new partition
- Mount partition — mount /dev/sdd1 smallfs
- Write stuff to disk
  - ↳ Get file hash
- Corrupt data
- Run fsck and look for data
  - ↳ Problem: Can take a long time
- Fill up disk
- Open gparted — disk full
- Add new disk to machine
- Problem: Now you have two mounts!?
  - ↳ Very annoying to user.

VIRTUALIZATION

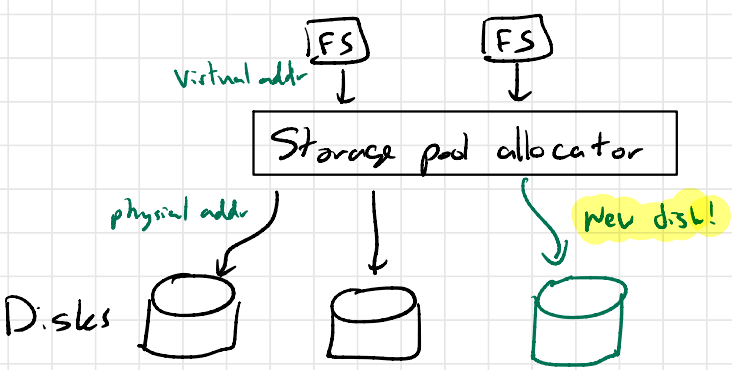
Traditional FS approach



RAID + extra fancy things complicate this a bit...

Problem: Resizing (get new drive, etc.) Manual mgmt

ZFS Approach



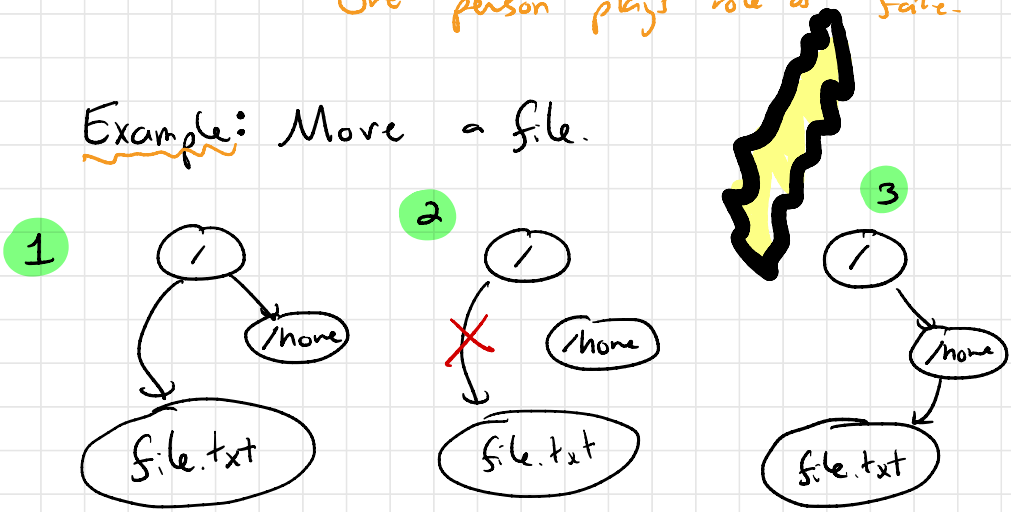
+ Can dynamically configure fancy things: mirroring, striping, ...

- Metadata? Complexity?

CONSISTENCY : Traditional FS

Act this out. One person plays role of OS
One person plays role of fate.

Example: Move a file.



Now FS will never reclaim space for file.txt.

↳ FSck can solve this problem (sort of)
"fscked"

Let's think: Why does this simplify things?

↳ No special-case logic to handle crashes in diff ops... it's all the same.

↳ Can do perf opts when writing changes to disk
↳ sequential writes

⇒ Might lose many seconds of work if machine fails

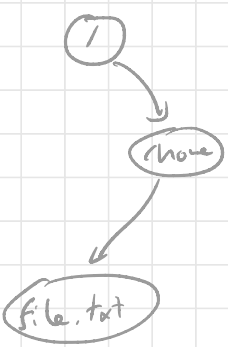
CONSISTENCY: ZFS

All operations (even file updates!) are copy on write. ... like a transaction!

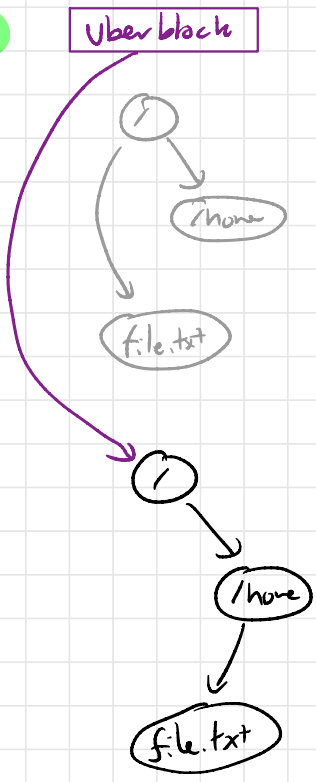
Again, act this out.

1

2



3



If a crash happens, you're either "completely before" or "completely after"! Slick!

Problem: Concurrency?

What happens if two users make writes at same time.

↳ May spend a lot of time building up shadow tree w/o committing it.

"Intent log"

Years later...

- \+ Checksums (esp for large data sizes)
- − Too much(?) flexibility
  - ↳ Hard to configure?
- − Surprisingly heated arguments on which to prefer.