6.824 2012 Lecture 15: Case studies: Frangipani

Frangipani: A Scalable Distributed File System
Thekkath, Mann, Lee
SOSP 1997

Intro
  YFS is a simplified Frangipani
  Frangipani vs YFS:
    recovery from server crashes
    fault-tolerant extent server
    high-performance extent server
  Looking at Frangipani again now that we've studied the ingredients
    replication, paxos, consistency
  look for ideas for final projects!

Overall architectural diagram
  [Clients, Frangipani, Petal, Lock]
  Petal: replicated block storage (no knowledge of FS)
  Frangipani: knows abt inodes, directories, &c

Why lots of Frangipani servers sharing Petal back-end?
  Why not split files over lots of Harp (or AFS) clusters?

Why separate Petal from Frangipani?

What does Petal do?
  [diagram: net, 3 srvrs, 6 disks]
  Looks like a huge disk (block read/write)
  Really striped across a cluster of Petal servers
  Two copies of every block
    Primary/backup replication, Paxos to agree on primaries

What happens if Frangipani server S1 creates file d/f?
  What steps does the server go through?
  lock d, read d's inode/content from petal, append op to *local* log,
  update local meta-data, release lock locally, reply to client.

What if S2 looks up d/f?
  S1 gets the REVOKE for d
  writes log to Petal, writes meta-data to Petal, RELEASEs lock

What if two clients try to create the same file at the same time?

The locks do two main things:
  Atomic multi-write operations.
  Cache consistency: readers see most recent data.

What if a server dies while holding locks?
  Revoke its locks and continue?
  Ignore it until it comes back up and recovers itself?
  What does Frangipani do?

Suppose:
  S1: delete d1/f1   crash
  S2:   create d/f1
  Will recovery re-play the delete?
  Does recovery need to get the lock from S2?
    No, and can't rely on this, since S2 may also have crashed.

Why not ask lock server what locks the dead server held?
  Rather than using version numbers?
  Maybe:

```
    S1: create d/f1         create d/f2 crash
    S2:             del d/f1
    S1 holds "d" lock but should not replay create d/f1!
  Or lock server might have crashed.

What if two servers crash at about the same time?
  And they both recently modified the same inode.
  S1: create d/f1                   crash
  S2:               create d/f2    crash
  Do we have to replay both ops?
    If we don't, do we risk missing an operation?
  Does it matter which order we replay their log records for that file?
  Can both recovery daemons read vers #,
    see that it is old, both write, and the lower # write second?
    Thus losing an update?
  What's the exact version number rule?
    Replay if log# > block#?
    Or if log# >= block#?

What about re-use of freed blocks?
  S1: delete d1/f   delete d1                          crash
  S2:                           create d2/f write d2/f
  Could f re-use d1's content block?
  And will replay of S1's log overwrite f's content?
  Do file blocks have version #s?

What if a server runs out of log space?
  What if it hasn't yet flushed corresponding blocks to Petal?

How does Frangipani find the start/end of the log?
  Could there be ambiguity if log has just the right content?

What's in a Frangipani log entry?
  (paper is not explicit, but says ca 180 bytes, thus not block images)
  operation description, e.g. create "f":inum1 in directory inum2
  may imply other modifications, e.g. remove inum1 from free bitmap
  version #s of each updated block

What if:
  S1 holds a lock
  Network problem, so S2 decides S1 is dead, recovers, releases S1's locks
  But S1 is alive and subsequently writes data covered by the lock
    Or reads cached data

What if the partition heals just before the lease expires?
  Could file server and lock server disagree about who holds the lock?

What if a lock server crashes?

Why does their lock service use Paxos?
  What do they need to agree about?

Why do they have multiple lock servers (Figure 2)?

Is there a replicated state machine hiding inside Frangipani?
  Or does it get fault tolerance from some other approach?

For what workloads is Frangipani likely to have poor performance?

Could its logs be a bottleneck?

Could the lock server be a bottleneck?
```

Table 2: why are creates relatively slow, but deletes fast?

Does NVRAM help?
  Why/when would we expect NVRAM to help?

Why is figure 5 flat?
  Why not more load -> longer run times?

Petal details
  Petal provides Frangipani w/ fault-tolerant storage
    so it's worth discussing
    also it's a good source for Lab 8 project ideas
  block read/write interface
    compatible with existing file systems
  looks like single huge disk, but many servers and many many disks
    big, high performance
    striped, 64-KB blocks
  virtual: 64-bit sparse address space, allocate on write
    why?
    virt addres partitioned over Petal servers
    each Petal srvr maintains translation map, like virtual memory
  primary/backup (one backup server)
    primary sends each write to the backup
  uses Paxos to agree on primary for each virt addr range
  what about recovery after crash?
    suppose pair is S1+S2
    S1 fails, S2 is now sole server
    S1 restarts, but has missed lots of updates
    S2 remembers a list of every block it wrote!
    so S1 only has to read those blocks, not entire disk
  logging
    virt->phys map and missed-write info

Lab 8 project ideas:
  Frangipani-style logs to tolerate yfs_client failure
  Fault-tolerant replicated extent server, like Petal
  Disk-based extent server, fast recovery after down-time