Replication in the Harp File System
Liskov, Ghemawat, Gruber, Johnson, Shrira, Williams
SOSP 1991

Key points
  2b+1 servers, keeps going w/ up to b failures, might recover from > b
  improvements over viewstamped replication:
  supports concurrent/overlapped operations, log to maintain order
  supports huge state, uses log to bring recovered disks up to date
  can recover after simultaneous power failures (of all 2b+1)

Outline basic operation.
  Client, primary, backup, witness.
  Client -> Primary
  Primary -> Backups
  Backups -> Primary, primary waits for all backups
  Primary replies to Client
  Primary tells clients to commit

Why does Harp use a log?
  1. keep track of multiple concurrent ops
  2. log is the in-memory state recovered by VSR
  3. log maintains order so we recover a prefix after failure
  4. log can bring a separated backup's disk up to date

What is in a typical log record?

Why does Harp have so many log pointers?
  FP most recent client request
  CP commit point (real in primary, latest heard in slave)
  AP highest record sent to disk on this node
  LB disk has completed up to here
  GLB all nodes have completed disk up to here?

Why the FP-CP gap?
  So primary doesn't need to wait for ACKs from each backup
    before sending next operation to backups
  Higher throughput: can overlap wait for prev op with exec of next
  Probably most useful when there are multiple clients

Why the CP-AP gap?
  Why not apply to disk at CP?
  exactly what happens at AP? how are ops applied?

Why the AP-LB gap?
  allows delay between issue of op and when it must complete to disk
  why?

What is the LB? How does Harp find out what the current LB is?

Why the LB-GLB gap?
  I.e. why not delete log record when disk write completes?
  Can't throw away log records until we know *everyone* has applied
them
  Because we might need to use our log to bring someone up to date

How does failure recovery work?

Cite as: Robert Morris, course materials for 6.824 Distributed Computer Systems Engineering, Spring 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [DD Month YYYY].

```
Scenarios
  5 servers, 1-5, 1 is usually primary, 2-3 backups, 4-5 witnesses

S2's cooling fan fails, so its cpu melts, and it crashes
  new view
    S4 is promoted (witness -> backup)
    S4 gets copy of log starting at GLB (i.e. all ops not known to be
on disks)
    S4 starts logging all operations, but doesn't apply them
    but GLB advances, so primary discards log entries
  why bother promoting S4?
  S2 gets a new CPU and reboots
  new view
  S4 sends big log to S2, S2 plays it to get all missing operations

S2 suffers a disk failure
  needs to get complete disk image + log from S1 or S3

what if S1 crashes just after replying to a client?
  where will new primary's FP and CP be after view change?
  does new primary have to do anything special about ops between CP and
FP?
  did other backups get the op?
  does this do the right thing for ops that the old primary *didn't*
reply to?

All nodes suffer power failure just after S1 replies to a client.
  Then they all re-start.
  Can they continue?
  Where were the logs stored while the power was out?
  What if they had all crashed -- could they continue?
    Crash == lost memory contents (despite UPS).
  How do they tell the difference?

Why does Harp focus so much on UPS and power failure?
  Since it already has a good story for more serious h/w falures?

S2 and S3 are partitioned (but still alive)
  Can S1+S4+S5 continue to process operations?
  S4 moves to S2/S3 partition
  Can S2+S3+S4 continue?

S2 and S3 are partitioned (but still alive)
  S4 crashes, loses memory contents, reboots in S2/S3 partition
  Can they continue?
  Depends on what S4's on-disk view # says.

Everybody suffers a power failure.
  S4 disk and memory are lost, but it does re-start after repair.
  S1 and S5 never recover.
  S2 and S3 save everything on disk, re-start just fine.
  Can S2+S3+S4 continue?

In general, how do you know you can form a view?
  1. No other view possible.
  2. Know about most recent view.
```

Cite as: Robert Morris, course materials for 6.824 Distributed Computer Systems Engineering, Spring 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [DD Month YYYY].

```
  3. Know all ops from most recent view.
  #1 is true if you have n+1 nodes in new view.
  #2 is true if you have n+1 nodes that did not lose view # since last
view.
    View # stored on disk, so they just have to know disk is OK.
    One of them *must* have been in the previous view.
    So just take the highest view number.
  Now that we know last view number,
    Need a disk image, and a log, that together reflect all operations
      through the end of the previous view.
    Perhaps from different servers, e.g. log from promoted witness,
      disk from backup that failed multiple views ago.

If a node recovers w/ working disk, can you really replay a log into
it?
  What if log contains operations already applied to the disk?

If a node recovers but disk needs fsck
  Is it legal to run fsck?
  Does Harp run fsck?
  Can you avoid fsck and repair by re-doing the log? As in FSD?

If a node recovers w/o disk contents, i.e. w/ empty disk
  Does it work to copy another server's disk?
  What if the other server is actively serving Harp/NFS ops?
  Can we avoid pausing for the entire time of disk copy?

How does primary generate return values for ops?
  It replies at CP, before ops have been applied to the file system!
  For example, how do you know an UNLINK would succeed?
  Or the file handle of a CREATE?

How does Harp handle read-only operations?
  e.g. GETATTR?
  Why doesn't it have to consult the backups?
  Why is it correct to ignore ops between CP and FP when generating the
reply?
    What if client sends WRITE then READ before WRITE reaches CP?

Does Harp have performance benefits?
  Yes, due to UPS, no need for sync disk writes.
  But in general, not 3x performance.

Why graph x=load y=response-time?
  Why does this graph make sense?
  Why not just graph total time to perform X operations?
  One reason is that systems sometimes get more/less efficient w/ high
load.
  And we care a lot how they perform w/ overload.

Why does response time go up with load?
  Why first gradual...
    Queuing and random bursts?
    And some ops more expensive than others, cause temp delays.
  Then almost straight up?
    Probably has hard limits, like disk I/Os per second.
```

Cite as: Robert Morris, course materials for 6.824 Distributed Computer Systems Engineering, Spring 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [DD Month YYYY].