



# HCFS: n-redundant storage for distributed systems with ZFS

- Luke Marsden
- Hybrid Logic Ltd.
- [www.lukemarsden.net](http://www.lukemarsden.net)



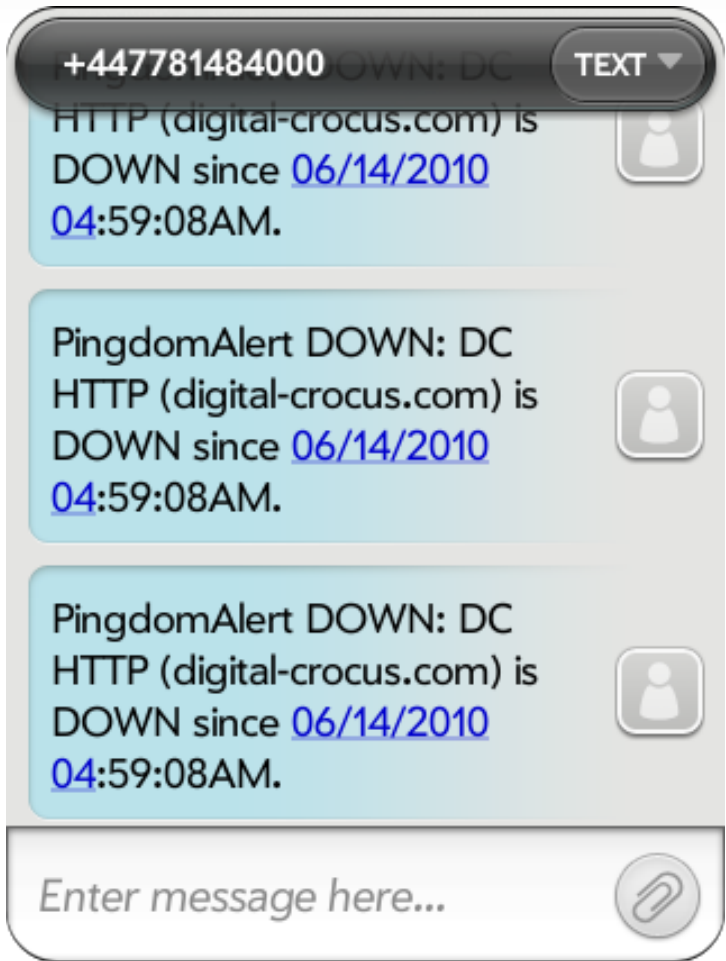
# Introduction



- HCFS builds on ZFS
- Adds distributed system features:
  - near-instant replication
  - failure-tolerance
  - partition-tolerance
- Don't modify ZFS, but just use it
  - Filesystem daemon + group messaging
- Runs on Solaris 10, OpenSolaris & FreeBSD 8



# Motivation for HCFS



- Hybrid Web Cluster is a web hosting platform built on HCFS
- Web hosting is stressful!
- Why spend big bucks to eliminate SPOFs in your hardware infrastructure?
- Better to do it in software
- Need: cross-DC, redundancy, horizontal scalability, load balancing, LAMP compatibility
- No existing solutions!



# What is HCFS?



- Reads and writes are always to a local ZFS filesystem
- One website or database = one filesystem = one master, N slaves, N+1 redundancy
- HCFS implements an asynchronously replicated filesystem by using ZFS snapshots and incremental sends and receives
- Changes replicated quickly due to filesystem change notification: Dtrace or Audit
- This is crucial so that failures can be handled

# Snooping the writes: Dtrace



```
#!/usr/sbin/dtrace -s

#pragma D option quiet
#pragma D option dynvarsize=2000000

syscall::*write:entry
/pid != $pid/
{
    self->fd = arg0;
    self->size = arg2;

    this->filistp = curthread->t_procp->p_user.u_finfo.fi_list;
    this->ufentryp = (uf_entry_t *)((uint64_t)this->filistp +
                                   (uint64_t)self->fd * (uint64_t)sizeof(uf_entry_t));
    this->filep = this->ufentryp->uf_file;
    this->vnodep = this->filep != 0 ? this->filep->f_vnode : 0;
    self->vpath = this->vnodep ? (this->vnodep->v_path != 0 ?
                               cleanpath(this->vnodep->v_path) : "") : "";

    self->output =
        self->vpath[0]== '/' &&
        self->vpath[1]== 'h' &&
        self->vpath[2]== 'c' &&
        self->vpath[3]== 'f' &&
        self->vpath[4]== 's' &&
        self->vpath[5]== '/'
        ? strjoin(strjoin("WRITE: ",self->vpath),"\n") : "";
    ...
}
```



# Snapshotting algorithm



- Local algorithm: when to snapshot?
- Don't want a snapshot for every write()
  - If there was a write, and there hasn't been another write three seconds later, take a snapshot
  - If it's been 15 seconds without a 3-second gap in writes, take a snapshot
  - (Numbers can be tuned for the requirements of the application)
- Pruning snapshots
  - ZFS copy-on-write makes collapsing snapshots work nicely
- Nice side-effect: time machine functionality



# Building the brain: Python Twisted



```
@defer.inlineCallbacks
def correctMountStatuses(self):
    """This function mounts and unmounts filesystems to make the filesystem state match the required state."""

    if not self.handleLock('correctMountStatuses', self.correctMountStatuses):
        return

    try:
        outputs = yield AsyncExecCmds(['/sbin/zfs mount']).deferredOutputLines()
        needs_mounting, needs_unmounting = self.calculateMountDiff(outputs)

        for (filesystems, mount) in (needs_mounting, True), (needs_unmounting, False):
            val = {False: 'umount -f', True: 'mount'}[mount]
            for filesystem in filesystems:
                cmds = []

                database = self.is_database(filesystem)

                if database and mount:
                    cmds.append('/usr/bin/env python /opt/HybridCluster/src/DatabaseSnapshotter.py --database="%s'

                cmds.append("/sbin/zfs %s %s/hcfs/%s" % (val, ZPOOL, filesystem))

                if database and not mount:
                    cmds.append('/usr/bin/env python /opt/HybridCluster/src/DatabaseSnapshotter.py --database="%s'

            # Now cmds contains the entire list of things to do for this filesystem. Run it!

            statuses, responses = yield AsyncExecCmds(cmds).getDeferred()
```



# Taming the unreliable network: Spread Toolkit



- Naive TCP doesn't scale –  $O(n^2)$ , difficult to reason with point-point connections
- Broadcast packet problems
  - message delivery failures
  - messages out-of-order
  - machines, or networks, fail
- Spread gives Extended Virtual Synchrony
  - logical ordering, delivery guaranteed
  - consistent view of group membership



# Redundancy algorithm



- Where and when to do replication?
  - `zfs send -i snap1 rpool/fs@snap2 | ssh remote zfs recv -i rpool/fs`
- Redundancy factor  $n$ 
  - $stateFixable(fs) := 0 < \#copies(fs) \leq n$
  - $stateGood(fs) := \#copies(fs) \geq n+1$
- Redundancy invariant: all filesystems in *stateGood*
- Invariant is broken by failures, fix it by:
  - machine with latest snapshot and lowest IP – the **leader** for that filesystem – copies it to a fresh server for that filesystem



# Filesystems get "rescued"

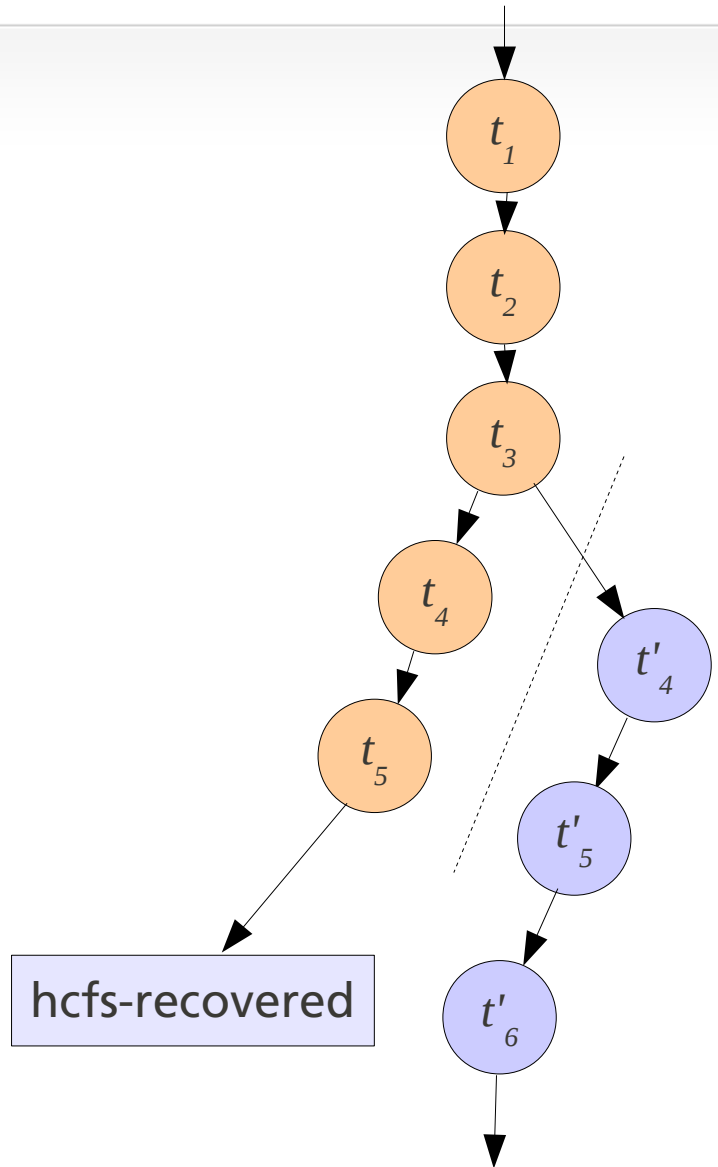


- "Dead" filesystems get "picked up" as quickly as possible when a server fails
- Leader decides who is the new master is for the filesystem
- The chosen master mounts the filesystem from its latest local snapshot
- Requests for the filesystem are directed to the new master via a proxying layer
- This technique useful for load balancing



# Partition tolerance

## Timeline for "site X"



- Partition caused by "netsplit"
  - Orange = Server A, LHS
  - Blue = Server B, RHS
- Clusters behave independently, filesystem becomes live on A and B
- After healing: "most recent snapshot of site X does not match incremental source."
- Latest snapshot wins  $t'_5 > t_5$
- Other timeline "stashed"



# Zones / Jails



- Plan to add “custom application support”
  - An HCFS filesystem could be the root filesystem of a zone or a jail running any app you like
  - “Rescuing” a jail would trigger it to be booted
  - PostgreSQL, Ruby, Java, etc.. could all get the same redundancy benefits
- Define three scripts through web interface: boot, pre-snapshot, post-snapshot for consistent snapshotting



# To summarise



1. Split workload up into separate filesystems
2. Filesystem modification detection
  - DTrace on Solaris, auditd on FreeBSD
3. Filesystem daemon
  - Reads from (1), issues ZFS snapshot commands
  - Each snapshot is replicated to a set of slave servers for the filesystem
  - If a machine comes up with old data, get it caught up



## To summarise



### 4. StateMachine – “Algorithmic brain”

- Connects through Spread to other machines in the cluster
- Ensures the redundancy invariant is always satisfied
- Reacts to server failures by replicating filesystems to other servers through the fsdaemon

### 5. Web hosting platform on top of HCFS

- Handles load-balancing, proxying requests



# Video Demo



**Hybrid**Cluster



# Thank you for listening!



- Questions?
- References:
  - Hybrid Cluster: [www.hybrid-cluster.com](http://www.hybrid-cluster.com)
  - Twisted: [www.twistedmatrix.com](http://www.twistedmatrix.com)
  - Spread: [www.spread.org](http://www.spread.org)
  - Extended Virtual Synchrony:  
[www.cs.jhu.edu/~yairamir/dcs-94.ps](http://www.cs.jhu.edu/~yairamir/dcs-94.ps)