Fossil

an archival file server

Russ Cox

rsc@mit.edu

PDOS Group Meeting

January 7, 2003

`http://pdos/~rsc/talks`

History

Cached WORM file server (Quinlan and Thompson):

- active file system on magnetic disk acts as worm cache
- mark all disk blocks copy-on-write at 5am to take snapshot
- slowly dribble snapshot to worm
- maintain forward linked list of snapshots
- present snapshot tree to users
- became integral part of our computing environment

```
% ls -lp /n/dump/*/*/386/bin/8c | uniq
--rwxrwxr-x presotto sys 243549 Jan 21  1997 8c
...
--rwxrwxr-x presotto sys 298289 Dec 14 18:55 8c
%

% yesterday -D authsrv.c
diff -n /n/dump/2003/0106/sys/src/cmd/auth/authsrv.c authsrv.c
/n/dump/2003/0106/sys/src/cmd/auth/authsrv.c:100 c authsrv.c:100
<            break;
---
>            exits(0);
%
```

Quinlan, ‘‘A Cached WORM File System’’, SP&E December 1991.

`http://plan9.bell-labs.com/~seanq/cw.pdf`

History, ii

WORM was right choice in 1990

- one jukebox is infinite: capacity grows faster than our storage needs
- no head crashes
- plausible random access times
- magnetic disks too small, tape too slow
- bootes (1990): 100MB mem, 1GB disk, 300GB juke box
- emelie (1997): 350MB mem, 54GB disk, 1.2TB juke box

What about 1999?

- disks cheap and big, getting cheaper and bigger
- disks cheaper and bigger than optical disk
- disks much faster than optical disk
- disks have head crashes
- build a better base out of magnetic disk?

Venti

Archival block store (Quinlan and Dorward):

- SHA1-addressed
- blocks never reclaimed
- omit duplicate blocks
- compress

Implementation:

- log of all blocks ever written
- log broken into fixed-size (say, 500MB) chunks called *arenas*
- arenas copied to other media (tape, DVD, etc.) as they fill
- index on the side makes lookups efficient

Initial system:

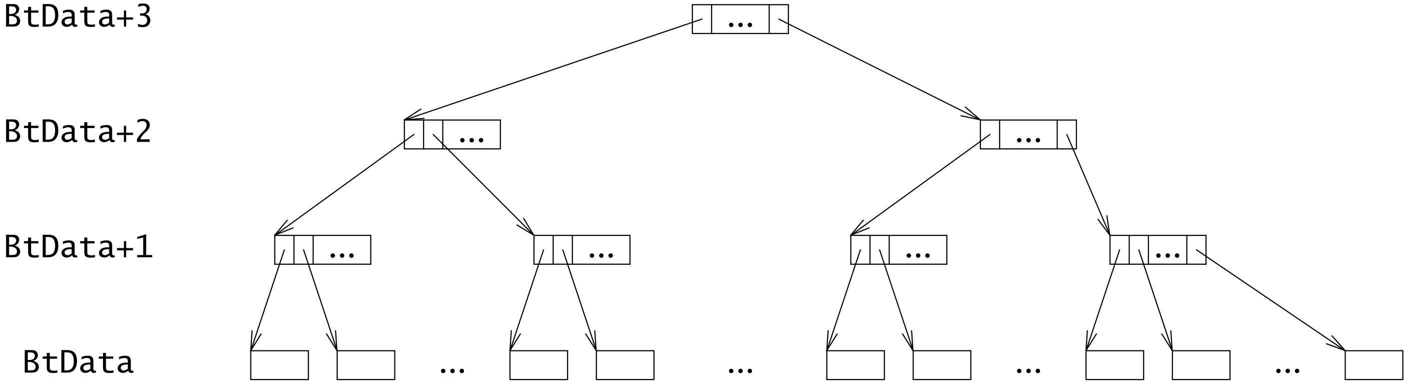
- iolaire (1999): 2GB mem, 36GB index, 480GB hw raid arenas

Quinlan and Dorward, ‘‘Venti: a new approach to archival storage’’, FAST 2002.

`http://plan9.bell-labs.com/sys/doc/venti.pdf`

Venti: storing data streams..

Venti stores blocks. To store large data, use hash tree:



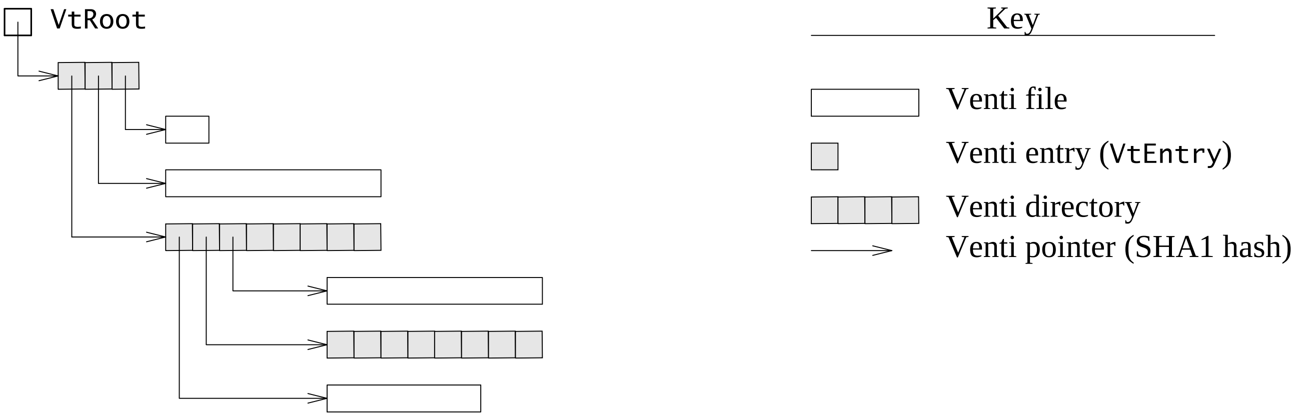
Venti: storing complex data structures

To store a list of streams, use a stream of `VtEntry` blocks.

- same as data but has block types `BtDir`, `BtDir+1`, ...

Can encode tree-like structures

- each stream is all data (a Venti file) or all entry blocks (a Venti directory)



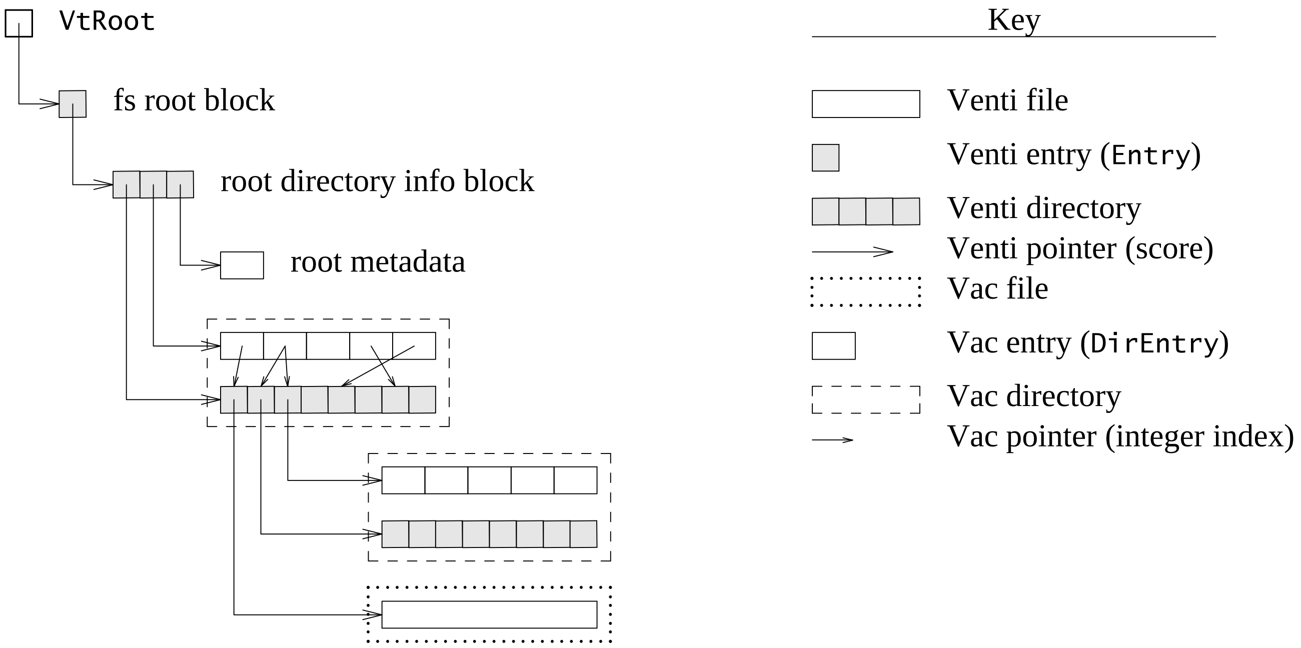
Can traverse hierarchy ignoring higher-level structure

- general purpose copy
- other utilities

Venti: storing a file system

Vac: Venti file system archive format

- vac directory can be thought of as stream of inodes plus stream of directory entries



Venti: storing a file system

Vac compresses everything to 45 bytes:

```
% cd /sys/src/cmd/fossil
% vac -f fossil.vac *
% ls -l fossil.vac
--rw-rw-r-- M 8 rsc sys 45 Jan  6 14:51 fossil.vac
% cat fossil.vac
vac:1bc12e0a81baf8c1ab62aaba382f6c1a0b11633a
% ls -l /n/vac
--rwxrwxr-x rsc sys   61096 Dec 21 15:35 /n/vac/8.9ping
--rwxrwxr-x rsc sys  219307 Jan  5 13:11 /n/vac/8.flchk
--rwxrwxr-x rsc sys  217712 Jan  5 13:11 /n/vac/8.flfmt
...
%
```

Fossil

Archival Venti-based file server (Quinlan, McKie, Cox)

Conceptually, rewrite of cached worm file server

- lots of software engineering advances (not discussed here)
- file system layout identical to vac
- local disk block pointers: 32-bit disk block zero-padded to 160 bits
- replace worm juke box with Venti store
- replace disk-based cache with disk-based write buffer
- write buffer can store file system if not using Venti

Snapshots

Epoch-based snapshot procedure:

- `fs.epoch` is logical snapshot clock (sequence number)
- every block in write buffer records allocation epoch `b.epoch`
- blocks with `b.epoch < fs.epoch` are copy on write.

To take snapshot: increment epoch, rewrite root block

My laptop takes snapshots on the hour:

```
% ls -lp /n/snap/2003/0106/0600/sys/src/cmd/fossil/fs.c
--rw-rw-r-- rsc sys 16943 Jan  5 13:03 fs.c
% ls -lp /n/snap/*/*/*/sys/src/cmd/fossil/fs.c | uniq
--rw-rw-r-- rsc sys 14895 Nov 28 02:05 fs.c
...
--rw-rw-r-- rsc sys 16918 Jan  5 12:48 fs.c
--rw-rw-r-- rsc sys 16943 Jan  5 13:03 fs.c
%
```

No Venti as described so far.

Archival

An *archival* snapshot goes into the archival tree.

My laptop takes archival snapshots daily, at 5AM:

```
% ls -lp /n/dump/2003/0106/sys/src/cmd/fossil/fs.c
--rw-rw-r-- M 1652 rsc sys 16943 Jan  5 13:03 fs.c
% ls -lp /n/dump/*/*/sys/src/cmd/fossil/fs.c | uniq
--rw-rw-r-- rsc sys 14230 Nov  9 02:51 fs.c
...
--rw-rw-r-- rsc sys 16943 Jan  5 13:03 fs.c
%
```

Background process archives tree to Venti

- only knows about Venti hierarchy
- rewrites pointers to point at Venti blocks
- prints Venti hashes to console

```
% grep vac: console.log
...
Sat Jan  4 05:01:46 archive vac:c164dba46cbe319bf5a3a6b93a6aec0aa09198f0
Sun Jan  5 05:01:14 archive vac:96f48562b826b5b95fef854e488fb06e66ad9eca
Mon Jan  6 05:02:12 archive vac:722d61f18fff491d00103be309af66ebb7cba9f2
%
```

Block reclamation

Non-archival snapshots will eventually fill the disk

Want to retire old snapshots to free up disk space

Epoch-based reclamation:

- `fs.epochLow` is epoch of earliest available snapshot
- after copy-on-write, block is no longer in active file system
- `b.epochClose` is epoch when `b` was copied-on-write
- block only needed by snapshots in [`b.epoch`, `b.epochClose`).
- if `b.epochClose ≤ fs.epochLow` then `b` can be reused

Fossil tricks..

Fs won’t boot, need to look at sources (on fs):

```
vacfs <{echo vac:ed62...3504}
cp /n/vac/active/sys/src/cmd/fossil/* /tmp/fossil
```

Reformat with new disk structures for write buffer:

```
fossil/flfmt -v vac:ed62...3504 /dev/sdC0/fossil
```

- loses disk snapshots, not archival snapshots

Backup Venti server to other Venti server:

- walk log, writing new blocks to alternate server
- save pointer in log to make next backup ‘‘incremental’’
- 152-line C program, 25-line shell script wrapper

Robustness

Suppose Venti is reliable (fault-tolerant)

- then archival snapshots are *forever*
- then loss of disk cache not a big deal: maybe lose a day
- bugs cannot destroy old archives:
  if you can read yesterday’s archive today,
  you can read it five years from now

Even without Venti or a WORM,

- having an enforced read-only latch on blocks keeps the present from corrupting the past
- random block tags identify dangling pointers immediately

How to make Venti fault-tolerant?

Mirror one Venti server to another (log trick)

- my laptop mirrors to server in my apartment
- server in my apartment mirrors to DVD and Bell Labs
- works for my data, not so good for Bell Labs

Mirror disks

- a kernel device provides transparent mirroring for apps

RAID?

- good as long as disks are *independent*
- all the disks in our first 480GB RAID array were identically defective
- are RAID controllers debugged yet?
  perhaps: cf. NetApp
  perhaps not: cf. Amsterdam

How to make Venti fault-tolerant?

Peer-to-peer?

- no incentives to run servers
- no machines to kick when system fails
- no humans to kick when system fails
- okay for small network where everyone knows each other?

Future work for Fossil

Keep pounding on it

- limited release just before New Year’s
- open release today

More protocols

- speaks 9P2000 right now
- add translators for NFS, SFS, SMB

What to do for Amsterdam?

Distinguish ‘‘restoration’’ goal from ‘‘archival’’ goal

- archival solutions often provide adequate restoration
- restoration solutions can be very simple

Immediately, can do ‘‘restoration’’ by mirroring:

- disk array mirrored to big IDE drives nightly
- problem with smearing across time (more later)

I’d like an archival solution.

- i don’t use my pdos home directory
- i’ve been spoiled by the dump

Amsterdam on Fossil?

Run Fossil and put our home directories there.

Why not?

- not interested Unix semantics (ctime, symlinks, ...)
- not interested in NFS semantics (locks, wcc, ...)
- (not interested in cruft i don’t use/need)
- we have a working file system that everyone likes
  and that isn’t my problem to debug

Archiving Amsterdam?

Set up a Venti server on some unused disk in Amsterdam (high bw)

Backup nightly disk images but:

- read fs structures so we don’t worry about unused blocks
- store blocks to Venti
- store block-to-SHA1 mapping as big Venti file (30MB for 23GB disk)
- provide access to images with NFS loopback server

Ship Venti blocks to DHash as background process

- store block-to-SHA1+key mapping (60MB for 23GB disk)
- same server can provide access to images

Archiving Amsterdam?, ii

When Venti fills, run copying gc to keep recent snapshots

- if we kept creation and ‘last use’ epochs for blocks, could do gc in one linear scan
- one Venti server will last longer than you think

File system smear..

Need fs disk synced, paused during backup.

- pause disk writes or fs writes (whichever easier)

How long is the pause?

- streaming disk reads on amsterdam: approx. 35MB/s
- MD5 on amsterdam: approx. 600MB/s (not CPU bound)
- SHA1 won’t be different enough to matter
- lower bound: 10 minutes for a full disk
- to be safe: say 20 minutes worst case

20 minutes per disk, in sequence

- 4am-7am worst case
- 4am-5am more likely
- schedule witchel’s disk last
- 8am-9am?
- optimization: scan unpaused fs then pause and rescan

Is ten minutes per fs acceptable?

User-level archives? ....

Time smear is not as serious a problem.

Wouldn’t be exact enough for restoration though.

Need help finding what changed.

A ‘‘change’’ service for disks ....

Keep a 64-bit (128-bit?) epoch as a per-disk clock.

Each write updates the epoch for that disk block and bump the epoch.

Efficient structure gives map from epoch to disk blocks edited since then.

A ‘‘change’’ service for file systems...

Keep a 64-bit (128-bit?) epoch as a per-fs clock.

Each change updates the ‘‘epoch’’ on the file system piece and bumps the epoch:

- changing file contents changes the file’s epoch
- changing file metadata changes the parent directory’s epoch
- removing a file changes the parent directory’s epoch

Efficient structure maps from epoch to files and directories changed since then.

Would be useful for Tra too.

Other ideas? ....