Venti Backups for Unix Hosts

Anthony Sorace
a.9srv.net

Strand 1 Technologies

ABSTRACT

Strand 1 offers backup services for clients', and our own, Unix hosts. This report describes how that is provided and used, using Plan 9 from User Space on the Unix hosts talking to a Plan 9 system elsewhere in the network.

1. Introduction

The main file servers on Plan 9 have long had an integrated view of history. Being able to view the history of your file system under */n/dump*, and having that history be automatically created as you use the system, has proven to be incredibly useful over the years, and has been reproduced, with some variations, by several filesystems since the original. It is a hard feature to give up. While features similar to this can be found on Unix, most notably NetApp's *snapshots*, they are not common and typically attached to expensive storage solutions. Our solution, while not as nicely integrated as either Plan 9's or NetApps, gives many of the familiar benefits, and does so using existing Plan 9 services we already have running. We have been using some variation on this system since 2009, with good results.

Note that "backups" is a very general term, and different parts of an overall backup strategy are useful for different purposes. We find the system described here to be very useful for ensuring key data is secure off-site, recovering accidentally deleted or corrupted data, and comparing revisions across history, among other things; it is not particularly well-suited to full-system recovery. We use mechanisms outside the scope of this report (such as VM snapshots) for those purposes[0].

2. Venti on Plan 9

We run *venti* on a Plan 9 host we'll call $venti. There is nothing remarkable about our venti setup, and given the complexity of setting up venti and sizing things appropriately, we do not suggest that this particular configuration is to be emulated. Still, it works well for us. We use the following partition sizes:

partition	size
557,402 MB	arenas
536 MB	bloom
27,870 MB	isect

We use the following configuration for *venti*, stored on the *arenas* partition:

[0] This is not an issue with venti, but rather with the specific setup described here. We've used venti for full disk images before, using Plan 9 Port's *vbackup*, and it's been fine.

```
index main
isect /dev/sdF0/isect
bloom /dev/sdF0/bloom
arenas /dev/sdF0/arenas
```

Again, nothing surprising there. The only "important" decision made when setting up the *venti* server is that it is located separately from the systems it backs up. Previously, this was a physical machine in our lab; today, it is a virtual machine hosted in a datacenter[1] where we do not host client work[2].

3. Backups from Unix

Aside from the backing store for a Plan 9 *fossil* file system, the primary use of this server is storing backups from Unix hosts. The system uses the *vac* program included in Plan 9 from User space to send data to the server, a *mkfile* to prepare the data to be sent, a *cron* entry to automate it, and a *score submission service* on a Plan 9 host to store the results. Each of these are described below.

3.1. mkfile

Since we're not doing full disk image backups, we need to specify what data to send and do any preparation needed. The two Unix hosts this system is used for most heavily today both run a *PostgreSQL* database, which we want to store the data from, so we use the *pg_dump* and *pg_dumpall* tools to perform a dump of the databases we care about before calling *vac*. The parts of the mkfile relevant to backups are:

```
MKSHELL=rc

none:VQ:
	echo usage: mk backup, dbdump

DBS=kalechips postgres template1
DUMPS=${DBS:%=backup/%.d}

backup/%.d:V:
	pg_dump -F d -f backup/$stem.d -Z0 $stem >[2] /dev/null

dbdump:QV:	$DUMPS
	pg_dumpall -r -f backup/roles.dump >[2] /dev/null
	pg_dumpall -t -f backup/tablespaces.dump >[2] /dev/null
	tar -cf - backup/roles.dump backup/tablespaces.dump backup/*.d | gzip -9 > backup/datab
	rm -r backup/roles.dump backup/tablespaces.dump backup/*.d

backup:VQ:	dbdump
	name = $sysname^.app
	today=`{date +%Y%m%d}
	# We don't really want the .pyc, but whatever.
	vac -h $venti -a backup/^$name^.vac .
	cp backup/$name.vac backup/$name-$today.vac
	{echo $name ; cat backup/$name.vac} | dial tcp!9srv.net!17038
```

[1] We use RamNode for this, which offers "Massive" (their marketing term) virtual machines; these are slower than many alternatives, but offer the most cost-effective solution to getting large storage among VPS providers we've found.

[2] We do cheat a bit and host a Plan 9 cpu server there, for our own use.

The use of *mk* here is mostly historical; the same mkfile used to be used for managing several tasks as part of a client project. For the backups specifically, a simple *rc* (or similar) script could also be used[3], as the backups are effectively always out of date.

This setup arranges for all the files to be backed up, including the database dumps, to be located in the same directory. A more flexible approach would be to name the files, and any exclusions, explicitly, like so: `vac -h $venti -a $name.vac -x $base/$name.xfiles '{cat $name.files}` where *$name* is defined as above, and *$base* represents the directory the vac files will be stored. You'll need to create a *$name.files* file listing the files or directories to back up (or just include them on the command line), and a *$name.xfiles* listing what to exclude. See *vac(1)* for details of the format for the exclude file. An example from a macOS hosts this system has been used with includes */Users* and */Library* in the files to back up and uses this as the exclude file:

```
exclude .../tmp/...
include /Library/Logs
include /Library/WebServer
exclude /Library/*
```

Note the ... in that first exclude line; that matches any string, including slashes. Named files otherwise follow shell expansion rules, as in the last line of the example.

3.2. cron

For backups to be effective, they must be automated, preferably automatic. We have *cron* on the Unix hosts invoke the *backup* target described above. The relevant parts of root's crontab on the Unix host looks like this:

```
PLAN9=/opt/p9p
PATH=.:/home/a/bin/unix:/usr/bin:/usr/sbin:/bin:/sbin:/opt/p9p/bin
sysname=greenhouse
38 7 * * * cd /opt/romanesco ; mk backup
```

3.3. Keeping Score

Having the data safely stored in venti doesn't do much good if we don't have the score[4]. To facilitate that, the mkfile above sends the resulting score to 9srv.net's "score submission service", a simple network listener whereby trusted hosts can submit vac scores to be held for later. Here is the entirity of */rc/bin/service.auth/tcp17038* (with IP addresses anonymized), which provides the service:

[3] A starting point can be found at , https://9p.io/sources/contrib/anothy/bin/rc/vacbak, which reflects a slightly older way of doing what's described here.

[4] Root scores submitted to venti are often recoverable, but doing so is time consuming and requires direct access to the venti's disks. This also suggests you should tightly control such access.

```
#!/bin/rc

# Vac score submission service.

# Authorized hosts: local addresses, Strand 1 hosts, clients, &c.
localhosts=(127.0.0.1 10.0.1.200 10.0.1.2 10.0.1.199)
ourhosts=(1.2.3.4 10.20.30.40)
clienthosts=(11.22.33.44)
guesthosts=(123.132.213.231)
okhosts=($localhosts $ourhosts $clienthosts $guesthosts)

rem=`{cat $3/remote}
port=`{echo $rem|sed 's/^[^!]*!//'}
rem=`{echo $rem|sed 's/!.*//'}

if (! ~ $rem $okhosts) {
	echo $rem is not authorized to submit scores to this host.
	exit hostauth
}
vacname=`{read}
cat > /lib/vac/^$vacname^.vac
```

This runs in *service.auth*, as opposed to *service*, only to avoid having to keep */lib/vac* world-writeable. It requires no other special permissions, and does not talk to the venti itself.

4. Getting at the Data

The best method for getting data back out depends on the use case. Since we're not storing full disk or system images, recovery tends to be interactive. The easiest way to do that is from Plan 9. With the scores submitted to our host as described above, *9fs system.vac* will look for */lib/vac/system.vac* and arrange for it to be mounted at */n/system*. You then get the familiar */n/system/YYYY/MMDD/* directory tree representing the series of snapshots.

If the running Unix system can also be mounted from Plan 9, `/n/host/foo/barcp/n/system/YYYY/MMDD/foo/bar` is often sufficient for restoring your data. If the Unix system is not configured for that, or if the files in question are large enough for the extra network traffic to matter, *unvac* can be used on the host to get data out. For example, to recover the database dump created by the backup described above, `unvac -t greenhouse.app.vac 2024/0301/backup/database.tgz` will give me the version from March 1st, 2024.

5. Conclusion

The system is comfortable and does what is intended, and has performed well for years. It's not perfect. Plan 9 Port's *vnfs* does a better job emulating the traditional *YYYY/MMDD* hierarchy from Unix, but depends on backing up full system images. Combined with *unvac* having limited facilities for interactive use, this makes the most comfortable way to use the system mounting the Unix backups from Plan 9. As Plan 9 is our perferred environment anyway that is a relatively small imposition, but it can sometimes feel awkward hopping between hosts. This is especially true in disaster recovery situations.

We have not described here steps for backing up *venti* itself, nor for securing it (for starters, the one these hosts talks to uses an "allow list" similar to the score submission service). Both of these are important topics which should be covered elsewhere.

Never rely on only one method of backing up important data.