Next Gen Ubik and the VLDB

A Key-Value Store for Ubik

Andrew Deason

June 2021

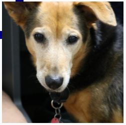
OpenAFS Workshop 2021

Preamble

- Brief general background (Ubik/VLDB)
- Problem background
- Bad solutions
- Good solution
- New commands for administrators

Background: Ubik

- Distributed database
  - Consensus algorithm
  - Data storage (ACID)
- Arbitrary data, single file
- For OpenAFS
  - Volume Location (VLDB)
  - Users/groups (PTDB)
- Old papers
  - Quorum Completion
  - Ubik – A Library for Managing Ubiquitous Data
  - Ubik: Replicated Servers Made Easy



Text/database logos from https://www.libreoffice.org/
Photo courtesy of Bea Deason-Perez, a good girl

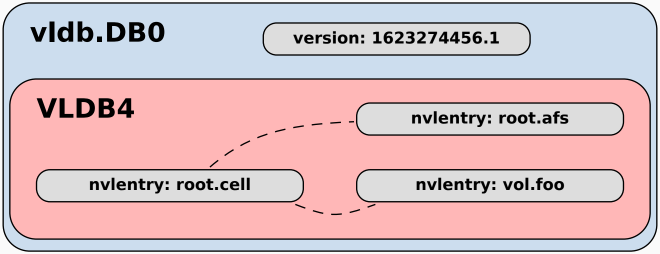
Background: VLDB

- **D**ata**b**ase for **L**ocating **V**olumes
- This stuff:

```
$ vos examine root.cell
root.cell                         536870912 RW          5 K  On-line
    server.example.com /vicepa
    RWrite  536870912 ROnly  536870913 Backup          0
    MaxQuota          0 K
    Creation    Sun May 10 17:24:22 2020
    Copy        Sun May 10 17:24:22 2020
    Backup      Never
    Last Access Mon Apr 26 02:39:14 2021
    Last Update Mon Apr 26 02:39:14 2021
    0 accesses in the past day (i.e., vnode references)

    RWrite: 536870912     ROnly: 536870913
    number of sites -> 2
       server server.example.com partition /vicepa RW Site
       server server.example.com partition /vicepa RO Site
```

Background: VLDB v4

- OpenAFS vlserver's VLDB format, version 4
- Network byte order, struct nvlentry, . . .
- `vldb.DB0`: VLDB4 inside the ubik `.DB0` format



Problem

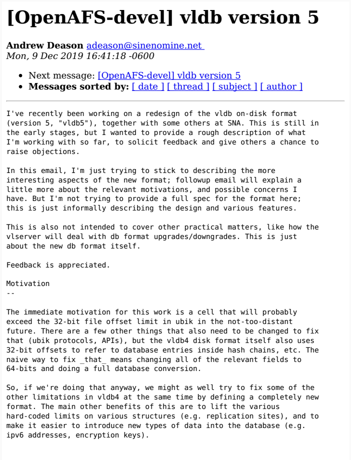
- Cells with millions of volumes
  - Slow lookups
  - Fixed hash table
  - See “VLserver memory cache” from the 2019 workshop
- How many volumes can we have?
- Volume id: 2^{32}, or ∼4 billion
- Ubik 32-bit file size: $\frac{2^{31} - sizeof(headers)}{sizeof(nvlentry)} = 14{,}509{,}076$
- What happens? gerrit 14180
- Limit in ubik and VLDB4 itself

Fixing VLDBv4

- So, change the 32-bit fields to 64-bits, right?
    - Requires a full db conversion
- VLDB4 has many other problems:
    - Fixed hash size
    - Hash chains in values
    - Little room for expansion
    - Flat, fixed-size structs
- Let's fix everything!

VLDB5 (2019)

- Record-based, XDR, B+ trees
- https://lists.openafs.org/pipermail/openafs-devel/2019-December/020616.html



[OpenAFS-devel] vldb version 5

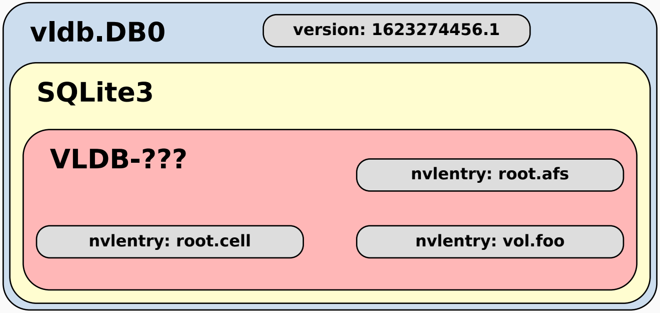
Andrew Deason adeason@sinenomine.net
Mon, 9 Dec 2019 16:41:18 -0600

- Next message: [OpenAFS-devel] vldb version 5
- **Messages sorted by:** [date] [thread] [subject] [author]

```
I've recently been working on a redesign of the vldb on-disk format
(version 5, "vldb5"), together with some others at SNA. This is still in
the early stages, but I wanted to provide a rough description of what
I'm working with so far, to solicit feedback and give others a chance to
raise objections.

In this email, I'm just trying to stick to describing the more
interesting aspects of the new format; followup email will explain a
little more about the relevant motivations, and possible concerns I
have. But I'm not trying to provide a full spec for the format here;
this is just informally describing the design and various features.

This is also not intended to cover other practical matters, like how the
vlserver will deal with db format upgrades/downgrades. This is just
about the new db format itself.

Feedback is appreciated.

Motivation
--

The immediate motivation for this work is a cell that will probably
exceed the 32-bit file offset limit in ubik in the not-too-distant
future. There are a few other things that also need to be changed to fix
that (ubik protocols, APIs), but the vldb4 disk format itself also uses
32-bit offsets to refer to database entries inside hash chains, etc. The
naive way to fix _that_ means changing all of the relevant fields to
64-bits and doing a full database conversion.

So, if we're doing that anyway, we might as well try to fix some of the
other limitations in vldb4 at the same time by defining a completely new
format. The main other benefits of this are to lift the various
hard-coded limits on various structures (e.g. replication sites), and to
make it easier to introduce new types of data into the database (e.g.
ipv6 addresses, encryption keys).
```

SQLite in Ubik

- Looking up values by name/id is “solved”
- Maybe not a custom database format?
- Idea: store a SQLite db (or other db) inside Ubik

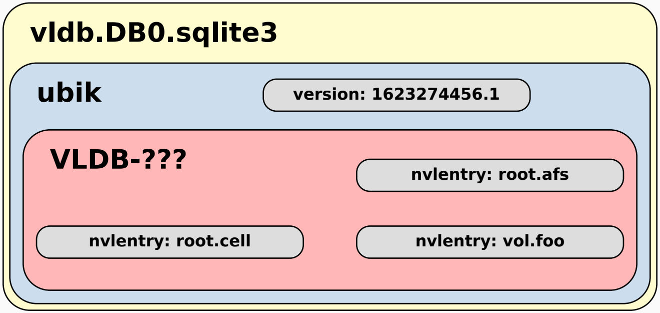


SQLite in Ubik

- Hard to implement
    - Many/most DBs don't have pluggable storage
    - Somewhat possible with SQLite
- Awkward and unusual
    - Good luck running sqlite tools
- Probably slow, no sqlite-level caching, mmap, etc
- Ties us to the sqlite3 format

Rethinking Ubik storage

- SQLite-in-Ubik vs Ubik-in-SQLite
- Sounds like a lot of work, but. . .



ubik-kv

- ubik KV interface (for **K**ey **V**alue storage)
- No longer use read/write/seek
  - `ubik_Seek()`, `ubik_Read()`, `ubik_Write()`
  - `ubik_KVGet()`, `ubik_KVPut()`, `ubik_KVDelete()`
  - New server-to-server RPCs
- Maps key blobs to value blobs
- NoSQL, but SQL dbs can be used
  - Not: `CREATE TABLE volumes (name VARCHAR(x), ...);`
  - More like: `CREATE TABLE kv (key BLOB PRIMARY KEY, value BLOB);`
- Restrictive, but we only ever store databases

ubik-kv

- Skips `udisk` and `uphys`
  - Faster
  - Handles ACID, no `.DBSYS1` / read-during-write
  - Easier VLMH (or no VLMH)
- Known formats understood by other tools
- Reduces code duplication
- Not tied to any db
  - SQLite, LMDB, BerkeleyDB
  - MariaDB, Oracle
  - even custom formats
- Changes invisible to vlserver / other sites

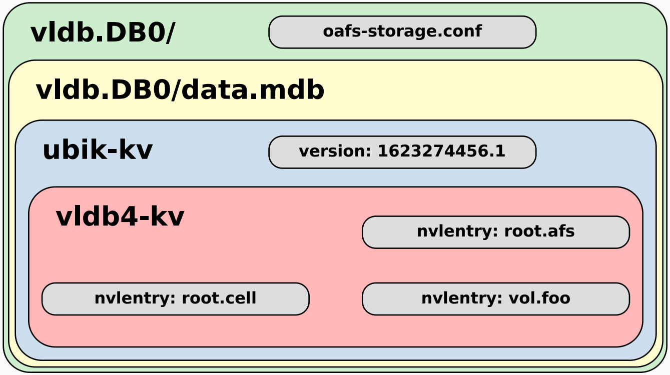
LMDB (Lightning Memory-Mapped Database)

- OpenLDAP's replacement for BerkeleyDB
- Small, `mmap`-based
- Fast for reads (~millions ops/sec)
- A few quirky restrictions
  - Virtual address space
  - Key size
  - Relies on underlying platform
- A good fit!

LMDB logo from https://github.com/LMDB

Implemented Solution: vldb4-kv

- Stuff vldb4's pile of structs into ubik-kv
- Why?
  - Easy first step
  - 32-bit limit, speed, ACID



vldb4-kv

- Prototype complete
  - https://github.com/adeason/openafs/tree/adeason/vldb4-kv
  - vlserver, `vldb_check`, upgrades
- Speed (informal benches)
  - Solaris: 1k $\rightarrow$ 34k (VLMH) $\rightarrow$ 46k reads/sec
  - Linux: 7k $\rightarrow$ 145k reads/sec
  - Linux w/writes: 3k $\rightarrow$ 19k reads/sec
  - Linux w/writes: 29 $\rightarrow$ 138 writes/sec
  - Changes with threads, pos/neg ratio, read/write ratio, etc
- Downsides
  - On-disk size: 283M $\rightarrow$ 1.2G (recovery)
  - `vos listvldb` slightly slower (~80%)
  - Change is scary

Upgrades

- Old procedure
  - Shutdown vlserver
  - Convert vldb
  - Restart vlserver
- New online procedure

```
$ vldb_upgrade -to vldb4-kv -online -backup-suffix .ORIG
Freezing VLDB... done (freezeid 4).
Converting /usr/afs/db/vldb.DB0 (vldb4) -> /usr/afs/db/vldb.DB0.CONV.1623364598 (vldb4-kv)
Converting fileserver entries... done.
Converting volumes... 100% (296139868 / 296139868), done.
Committing changes... done.
Installing /usr/afs/db/vldb.DB0.CONV.1623364598 to ubik... done.
Distributing new database... done.
Unfreezing VLDB... done.

Converted /usr/afs/db/vldb.DB0 from vldb4 to vldb4-kv (1622076124.1 -> 1622076125.1)
Backup saved in /usr/afs/db/vldb.DB0.ORIG
```

New commands

```
$ openafs-ctl vldb-info
vldb database info:
  type: kv
  engine: lmdb (LMDB 0.9.29: (March 16, 2021))
  version: 1622076123.1
  size: 7999994
```

```
$ openafs-ctl vldb-dump /tmp/vldb.DB0
Freezing database... done (freezeid 1, db 15895059050000000.3).
Dumping database... done.
Ending freeze... done.
Database dumped to /tmp/vldb.DB0, version 15895059050000000.3
```

```
$ openafs-ctl vldb-restore /tmp/foo.DB0 -no-backup
Freezing database... done (freezeid 7, db 16220761260000000.1).
Making copy of /tmp/foo.DB0... done.
Installing db /usr/afs/db/vldb.DB0.TMP... done.
Distributing db... done.
Ending freeze... done.

Restored ubik database from /tmp/foo.DB0
```

New commands

```
$ openafs-ctl vldb-freeze-run -rw -cmd ./do_restore.sh
[...]
$ cat do_restore.sh
#!/bin/sh

set -xe

# don't dist db when restoring (yet)
openafs-ctl vldb-restore /path/to/new.vldb.DB0 -backup-suffix .bak -dist skip

vos listvldb vol.important -noresolv -config /path/to/localconf > /path/to/vos.out
diff -u /path/to/vos.out /path/to/expected.out

# vldb looks ok; dist new db to other sites
openafs-ctl vldb-freeze-dist
```

- Sets env vars (`$OPENAFS_VL_FREEZE_VERSION`, et al)
- Reverts installed db on failure

openafs-ctl

- New command suite
- Local “control” only, no network
    - Like FSSYNC (`dafssync-debug`)
    - Local-only for security, reliability
- Stop relying on signals
- Not ubik-specific, more in the future

Postamble

Dev Branch

https://github.com/adeason/openafs/tree/adeason/vldb4-kv

Gerrits

Most recent: https://gerrit.openafs.org/14632

Slides

http://dson.org/talks

Contact

adeason@dson.org

adeason@sinenomine.net