Samba's Cloudy Future

Jeremy Allison
Samba Team

jra@samba.org

Opening Windows to a Wider World

Isn't cloud storage the future ?

Yes, but not usable for many existing apps.

Cloud Storage is a blob store

- Blob stores don't map very well onto the open/read/write/close random access semantics of most applications.
- Apps are changing to cope with no random access semantics of cloud stores, but this will take time.

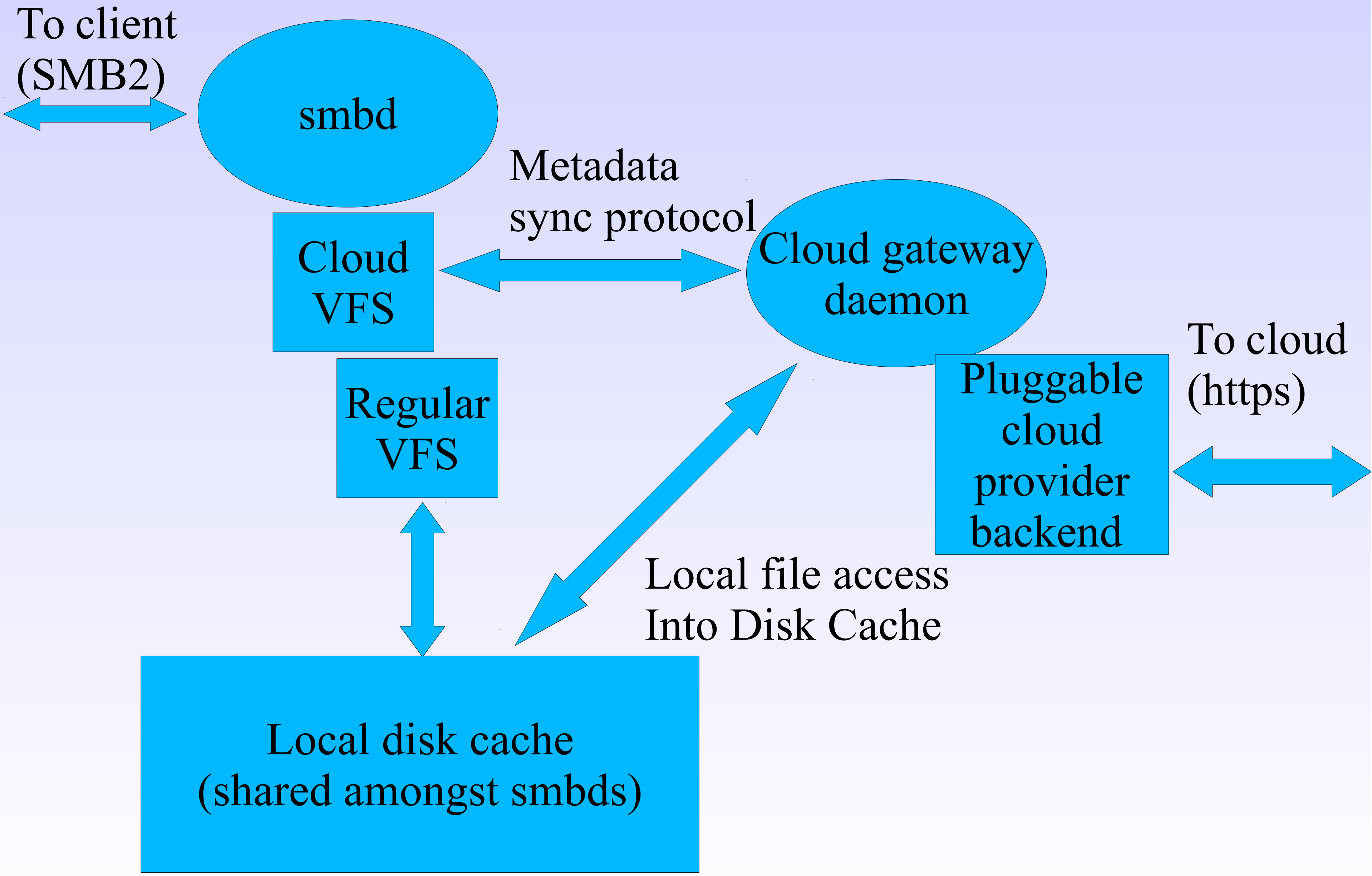
Doesn't FUSE solve this ?

- Mostly, for Linux and MacOS X applications.
- No FUSE on Windows clients.
  - So can't Windows clients access via Samba running on top of a FUSE filesystem ?
  - No, because FUSE system calls can block for an awfully long time..
  - In *theory* Samba can cope with this by making all syscalls async.
  - Might we become a FUSE host ?

Enter flexible Samba

- The VFS can save us !
- Stackable, modular, easily modifiable user-space code.
- Already contains code to cache all reads/writes onto local files.
    - We *just* :-) need a way to ensure we sync to the remote cloud on metadata update
    - File/directory creation/deletion, .
- What should we do with the extra metadata (ACLs etc.) ?

Possible Architecture



What level of coherence can we provide ?

- **NOT** full Windows semantics.
  - Latency going to the the cloud would make this unusable.
  - Only clients going to the same on-premises Samba-Cloud gateway provide Windows semantics.
- Coherence on close/unlink/mkdir/create only.
  - If a client writes then closes, they are guaranteed to have that version uploaded to cloud storage.
  - Open files not synced.
- “Good-enough” semantics for many apps.

Under the covers

- Cloud services daemon reads/writes into private namespace hidden from smbd, but on the same physical device.
- On sync use atomic file operations (rename, unlink) to move to/from private namespace into smbd-exported namespace.
- If we have a underlying btrfs copy-on-write filesystem, use file cloning on close to create instant copies that can be synchronized out to cloud storage.
  - On non COW filesystems, physical copies of files needed.

Operations – Open/Create/Mkdir

- Create operations must sync with cloud daemon.
  - Requires synchronous call to cloud backend.
  - On create success local file/directory created.
  - For files marked TMP, ignore cloud operations ?
- For open existing, open operation must trigger read from cloud.
  - At least fetching file metadata (size etc.) must be synchronous.
  - File data access operations (read/write) can then be forced to go async until the data is synchronized from the cloud.
  - FILE_ATTRIBUTE_OFFLINE can help here.

Operations – Read/Write/Truncate/Unlink/Rename

- Once the on-disk file is known to be in a valid state, reads, writes and truncates can proceed as normal.
  - No cloud communication needed.
- Unlink and rename need synchronous operation to the cloud, but should be a reasonably quick operation (just round-trip latency).

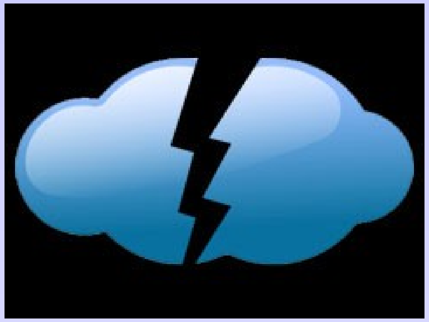
Operations - Close

- Reference count open files – on last close then expensive synchronous operations need to happen.
- If file modified then make a sync call to the cloud sync daemon to make a copy of this file.
  - Use COW if available.
  - Once copy is complete, smbd can reply to the close request.
- Cloud sync daemon then pushes changed file up to the cloud.
  - Should we add a delay time here ? Only guarantee coherence after 5 minutes ?

The cloud sync daemon

- Using existing Samba technologies talloc, asynchronous tevent, with backing threadpool.
  - All smbd's will need to talk to it.
  - Use tdb transactions for crash recovery.
- Create plugable back-ends to allow choice of cloud storage providers.
- Ignore key management/credentials issues.
  - Just expect credentials file to have been magically placed locally by an administrator.
- Many possible tunables – number of simultaneous connections, bandwidth limits etc.

Error recovery



- Shock, horror. Cloud file operations can fail, right in the middle of uploading via https.
  - Do we allow client access to continue whilst there is a cloud outage ?
  - If so, how do we queue operations that occur during cloud outage ?
  - How much queuing do we allow in order to replay operations later before stopping client access ?
- I don't currently have good answers to these problems.

Privacy: "*We have secured ourselves from the NSA, except for the parts that we either don't know about or can't talk about.*" - Bruce Schneier

- Data stored in the cloud unencrypted, or remotely encrypted with keys held by the cloud storage provider can be compromised.
- Local encryption before uploading to the cloud is the only way to reduce this risk.
- Getting this right is hard. Two options I can see:
  - vfs_encfs – Samba VFS module that emulates encfs fuse module.
  - Copy to encfs filesystem before uploading to cloud.

Create backends that target the Big Three Cloud Store Vendors

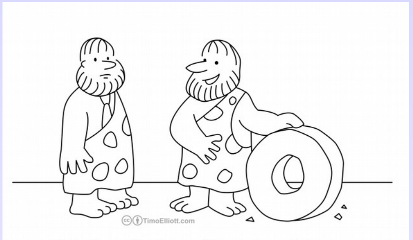
Microsoft Azure

amazon web services™

Google Cloud Platform

Existing vendors

- There are many existing cloud-gateway companies, some of which are already using Samba for this purpose.
- Are we trying to compete with them ?
  - No – In doing this I'm trying to raise the bar on what are the “basic” services provided with Samba.
  - Proprietary cloud-gateways provide more complete service guarantees.



“It does look similar—but this one is powered by Hadoop”

- Are you reinventing the wheel ?
  - Yes. Loughborough University in the UK has this code already. Unfortunately they're not releasing it back to the community (so far, I've asked).

Questions and Comments ?

Email: jra@samba.org

Slides available at:

ftp://samba.org/pub/samba/slides/sambaxp-2015-cloudy-future.odp