bup: the git-based backup system

Avery Pennarun

2010 10 25

The Challenge

- Back up entire filesystems (> 1TB)
- Including huge VM disk images (files >100GB)
- Lots of separate files (500k or more)
- Calculate/store incrementals efficiently
- Create backups in O(n), where n = number of changed bytes
- Incremental backup direct to a remote computer (no local copy)
- ...and don't forget metadata

Why git?

- Easily handles file renames
- Easily deduplicates between identical files and trees
- Debug my code using git commands
- Someone already thought about the repo format (packs, idxes)
- Three-level “work tree” vs “index” vs “commit”

But git isn't perfect...

Problem 1: Large files

- 'git gc' explodes badly on large files; totally unusable
- git bigfiles fork “solves” the problem by just never deltifying large objects: lame
- zlib window size is very small: lousy compression on VM images

bupsplit

- Rolling checksum:
  - for each 64-byte window
  - run rsync's adler32
  - If low 13 checksum bits are 1, end this chunk
  - average chunk size: 8192
- Now we have a list of chunks

- **Inserting/deleting bytes changes at most two chunks!**

bupsplit trees

- Inspired by “Tiger tree” hashing used in some P2P systems
- Arrange chunk list into trees:
  - If low 17 checksum bits are 1, end a superchunk
  - If low 21 bits are 1, end a superduperchunk
  - and so on.
- Superchunk boundary is also a chunk boundary

- **Inserting/deleting bytes changes at most 2*log(n) chunks!**

Advantages of bupsplit

- Avoids the need for deltification
- Never loads the whole file into RAM
- Compresses most VM images more (and faster) than gzip
- Works well on binary *and* text files
- Don't need to teach it about file formats
- Highly efficient for incremental backups
- Diff huge files in about O(log n) time
- Seek to any offset in a file in O(log n) time

Problem 2: Lots of files

- .git/index is too slow:
  - A linear set of filenames+hashes
  - You can't binary search it! (Variable-sized binary blocks)
  - git rewrites the whole file for *each* added file

bupindex

- .bup/bupindex is fast:
  - uses a tree structure instead: O(log n) seeks
  - update entries without rewriting
- Unfortunately, adding files to the bupindex still requires rewriting the whole thing at least once

Problem 3: Millions of objects

- Plain git format:
  - 1TB of data / 8k per chunk: 122 million chunks
  - x 20 bytes per SHA1: 2.4GB
  - Divided into 2GB packs:
    500 .idx files of 5MB each
  - 8-bit prefix lookup table

- Adding a new chunk means searching
  500 * (log(5MB)-8) hops
  = 500 * 14 hops
  = 500 * 7 pages = 3500 pages

Millions of objects (cont'd)

- bup .midx files: merge all .idx into a single .midx
- Larger initial lookup table to immediately narrow search to the last 7 bits
- log(2.4GB) = 31 bits
- 24-bit lookup table * 4 bytes per entry: 64 MB

- **Adding a new chunk *always* only touches two pages**

Idea: Bloom Filters

- Problems with .midx:
  - Have to rewrite the whole file to merge a new .idx
  - Storing 20 bytes per hash is wasteful; even 48 bits would be unique
  - Paging in two pages per chunk is maybe too much
- Solution: Bloom filters
  - Idea borrowed from Plan9 WORM fs
  - Create a big hash table
  - Hash each block several ways
  - Gives false positives, never false negatives
  - Can update incrementally

Problem 4: Direct remote backup

- The git protocol wasn't helpful here
  - Surprisingly inefficient in various cases
- New protocol:
  - Keep .idx files locally, .pack files remotely
  - Only send objects not present in server's .idx files
  - Faster than rsync or git for most data sets

bup command line: tarballs

- tar -cvf - / | bup split -n mybackup
- bup join mybackup | tar -xf -

bup command line overview

bup command line: indexed mode

- Saving:
  - bup index -vux /
  - bup save -vn mybackup /
- Restoring:
  - bup restore (extract multiple files)
  - bup ftp (command-line browsing)
  - bup web
  - bup fuse (mount backups as a filesystem)

Not implemented yet

- Pruning old backups
  - 'git gc' is far too simple minded to handle a 1TB backup set
- Metadata
  - Almost done: 'bup meta' and the .bupmeta files

Crazy Ideas

- Encryption
- 'bup restore' that updates the index like 'git checkout' does
- Distributed filesystem: bittorrent with a smarter data structure

Questions?