Maktabkhooneh Backup System

Requriements and Design of a Reliable Backup System

Mohammad Fazeli

Maktabkhooneh.org

May 20, 2021

Who am I?

Mohammad Fazeli
DevOps Tech Lead at `Maktabkhooneh.org`
Masters Degree in Computer Science
6 years and counting working with Python to deliver education software online :)

Where is Maktabkhooneh?

Online Education Platform :)
Started 8 years ago
More than **300 free courses mostly from Sharif University** and other top universities online.
More than 100 paid courses about practical stuff for out of university.
Use discount code provided in the session for 70,000 toman off on paid ones, and **checkout more than 300 free courses** too.

What is this about?

"I found the Holy Grail of backups."
(Stavros K. about Attic-Backup, 8/2013)

What Can Go Wrong For Web App?(Google 2010, Jeff Dean)

The Joys of Real Hardware

Typical first year for a new cluster:

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures

~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

Reliability/availability must come from software!

Google

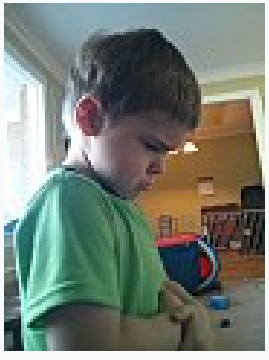
What Else Can Go Wrong? RAM

- RAM bits flip
  - IBM: cosmic rays can flip bits in RAM
  - Server could crash, Data could get corrupted
  - 256MB, one bit flip per month(in 1996)
  - Google: one bit flip per 14 to 40 hours per Gigabyte of RAM, not evenly distributed though
  - Solution: ECC RAM

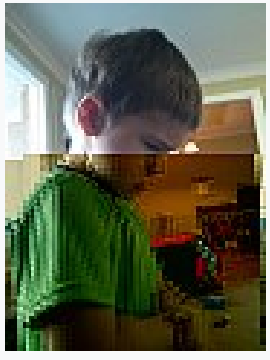
What Else Can Go Wrong? Disks

- Storage bits flip on their own
  - SSD electric charges leaks away
  - HDD magnetic orientation change changes
  - Google back in 2005
  - Today SSD/HDD have ECC inside
  - URE
  - Kingston: URE for consumer SSD 1 in 110TB read, Enterprise 1 in 1110TB read(Don't trust them with numbers)
  - Some drives have way more often
  - Facebook RAID controller wrote data in the middle of the disks on boot(firmware problem)
  - Solution:
    - RAID 5?
    - RAID 1?
    - BTRFS/ZFS/Ceph/... store hash, check it on every read, recover from specific implementation RAID.

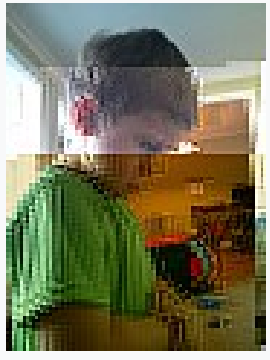
Bit rot effect example



0 bits flipped



1 bit flipped



2 bits flipped

3 bits flipped

What Can Go Wrong? the Human Factor

- Wiped a production db myself :)
- Getting Hacked(like arvan)
- Getting Ransomwares

Solutions

Backup Desiderata

- Take very little space
- Encrypted
  - If Backup storage gets exposed means everything is in the open
- Multiple versions of data
- Pruning versions By Grandfather-Father-Son
- Deduplicated, reduced space

Backup Desiderata: Not using Borg

- Disk failure shouldn't cause data loss
- Server failure shouldn't effect availability
- Disks bit-rot shouldn't affect the data
- Backup files getting exposed should make them accessible
- Malicious ones can't delete backups(even if the backup taker server gets exposed)

My Solution

- Borg + BTRFS RAID1 on at least two locations
- BTRFS RAID1 removes bit-rot problem
- Two locations alleviates availability problems
- Rsync.net and BTRFS snapshots solves immutability

What is BorgBackup

BorgBackup (short: Borg) deduplicating backup program(compressed and encrypted authenticated)

Borg Features

- cli backup tool, separate GUIs
- good arch / platform / fs support
- deduplication
- compression
- auth. encryption
- read from locally mounted fs
- store to local fs or to remote borg
- Fuse-mount backup archives
- HW-accelerated crypto
- Computational intensive stuff written in C, others in Python
- Uses single core(con)

Borg as a Project

- community project
- FOSS (BSD license)
- Python 3.7(current, 3.6-9 supproted)
- for speed: a little Cython & C
- good docs (for users, devs)
- good test coverage(83% today), travis CI
- github: borgbackup/community
- Pays money for security bugs(10,500$ paid till 2020)

In Practice

- size of postgres db
- 48 hourly
- 20 daily
- Weekly 6
- Monthly ...
- 78 instances each 2GB
- all:153.06 GB
- compressed 41.24 GB
- deduplicated 22.50 GB

In Practice

- media folder of a big django project
- 3 instances each about 63 GB
- 182.70 GB compressed all instances(incompressible mostly already compressed images)
- 60.13 GB deduplicated across time

In Practice

- a little changing postgres db
- 12 instances
- 7 daily
- 4 weekly
- 1 monthly
- 2.75 GB all
- compressed 180MB
- deduplicated 30 MB

In Practice

- a wordpress site files
- 32 instances
- 80 GB total
- 70 GB compressed(incompressible mostly images)
- deduplicated 2.68 GB

In Practice

- a wordpress site db
- 32 instances
- 5.5 GB
- 420 MB compressed
- 204 MB deduped

In Practice

- a lot of mail boxes
- 20 instances
- 20 daily
- total 727GB
- 348 GB compressed(compressible since it is text)
- deduped 17.5 GB

In Practice

Total size: 1145 GB
Fitted in 103GB

It scales well

one of developers of borg borg info
ssh://borg@myserver/repos/myrepo
Original size 22.76 TB
Compressed size 18.22 TB
Dedup size 486.20 GB
Unique chunks 6305006
Total chunks 272643223

Where to put the data

Own a server with ssh and free space:
install borg, configure, done!
No remote server? No problem:

- rsync.net (ssh, cli)
- hetzner's storage box
- borgbase.com (ssh, web, easy)
- local repo + rclone to cloud

Deduplicating

dedup does **NOT** depend on:

- file/directory names:
  - move file even between machines, no change required
- Whole file hash and timestamps:
  - only a small part of a file changed,only the change, VMs , or raw disks, only changes are synced.
- Place of chunk in a file:
  - stuff moved around inside a file , doesn't increase it.

Encryption

Enc :

- Tampering/Corruption detection by HMAC or blake2b
- 256 bit AES- CTR
- uses OpenSSL(libcrypto)
- Passphrase and a key, could be separate
- PDKDF2 100K rounds

Compression

works on deduplicated block level

- none
- lz4: fast, low compression(faster than no compression :-|)
- zstd: from high speed low compression to slow and high compression
- zlib: medium speed and compression
- lzma: low speed, high compression
- auto: first lz4 it(superfast), if its size have been reducedin size(compressible) then compress with other schemes.

Safe

borg uses: - checksums - transactions - fs: syncing, atomic ops - checkpoint while backing up

language

```
# initialize a repository:
borg init /tmp/borg

# create a "first" archive inside this repo (verbos
borg create --progress --stats\

 /tmp/borg::first ~/Desktop


# create a "second" archive (less verbose):
borg create /tmp/borg::second ~/Desktop

# even more verbose:
borg create -v --stats /tmp/borg::third ~/Desktop
```

language

```
# list repo / archive contents:

borg list /tmp/borg
borg list /tmp/borg::first

# extract ("restore") from an archive to cwd:

mkdir test ; cd test
borg extract /tmp/borg::third

# simulate extraction (good test):
borg extract -v --dry-run /tmp/borg::third

# check consistency of repo:
borg check /tmp/borg
```

language

```
^^I^^I# list repo / archive contents:
^^I^^I
^^I^^Iborg list /tmp/borg
^^I^^Iborg list /tmp/borg::first
^^I^^I
^^I^^I# extract ("restore") from an archive to cwd:
^^I^^I
^^I^^Imkdir test ; cd test
^^I^^Iborg extract /tmp/borg::third
^^I^^I
^^I^^I# simulate extraction (good test):
^^I^^Iborg extract -v --dry-run /tmp/borg::third
^^I^^I
^^I^^I# check consistency of repo:
^^I^^Iborg check /tmp/borg
^^I
```

New Generation Filesystems

- Bit rot protection(check and fix on each read)
- It has fast space efficient snapshot
- Use RAID1(RAID5/6 not stable)
- ZFS works, Ceph/S3 most other new things have this too.
- Run scrub periodically(reads all data and fix)
- Keep snapshots, and rotate them on your own(even if the client gets exposed, can't delete snapshots)
- openSUSE's default filesystem
- super efficient diff snapshot transfer to other locations

Borgmatic

A tool for automation for backup with a lot of useful stuff

- Config is a YAML file
- Rotation Scheme is written in YAML
- Supports pre/post execution hooks
- Monitoring using healthchecks.io and others

Healthchecks.io

An opensource Scheduled-Job monitoring.
Cron schedule are defined in the site, a token is given.
Check start, end and non execution of job and alert properly

rsync.net

Provide ssh accessible, backup space storage
Has geo-redundant option too
Minimum space is 400GB for 10$/month(17.5$/month for geo-redundant)
ZFS with RAID is the underlaying file system
7 free daily immutable snapshot.
Configurable cron time for more immutable snapshots.
Extra ones is counted against your quota but ZFS snapshots are block levels diffs, thus it is only actually overwritten/added blocks of data not a new copy of it

What we do for backups?

- For new env, setup ssh access on backup client and storage server using ansible
- install proper borg version using ansible
- install borgmatic using pip in the same ansible playbook
- Write a borgmatic YAML config(from templates available for different usecases across the company)
- Add proper cron schedule at healthchecks.io and add the token to YAML file
- put borgmatic in crontab

Faults

Trusting borg
SSH to outside countries sometimes is problematic
Populating a db from SQL dumps could be time consuming