Engineering Against Data Loss
(plus some random asides)

James Cowling
james@dropbox.com
Immutable content-addressable block storage system.
Immutable content-addressable block storage system.

Magic Pocket
Immutable content-addressable block storage system.
Immutable content-addressable block storage system.

hash → block
Immutable content-addressable block storage system.
Immutable content-addressable block storage system.
Multiple exabytes

3+ geographic regions

500+ million users
Build ➔ Validate ➔ Scale ➔ Optimize
Build ➔ Validate ➔ Scale ➔ Optimize
Build ➔ **Validate** ➔ Scale ➔ Optimize
MP Safety

808.9 days since last accident on this site
Build ➡ Validate ➡ **Scale** ➡ Optimize
30-40 racks per day

Supply chain issues, logistics

Migrated >500PB user data in <6 months
Build ➝ Validate ➝ Scale ➝ Optimize
Aside:

hard drive technology is really cool
helium

SMR

HAMR
Build ➡ Validate ➡ Scale ➡ Optimize
Build ➡ Validate ➡ Scale ➡ Optimize
never lose a file ever
durability theater or real-world data safety
Multisite replication!
Markov model!

\[
\frac{n!}{(m-1)!} \times \lambda^{(n-m+1)} \times \frac{(n-m)!}{(n-m)!} \times \mu^{(n-m)}
\]
27 nines!

\[ \frac{n!}{(m-1)!} \times \frac{(n-m+1)!}{(n-m)! \times \mu^{n-m}} \]
if you lose data due to routine disk failure

you’re doing something wrong
Isolation

Protection

Verification

Automation
Isolation
- physical
- logical
- operational

Protection

Verification

Automation
strong isolation
is key to failure independence
Physical Isolation

Logical Isolation

Operational Isolation
Physical Isolation

Store stuff on different stuff
Physical Isolation

Store stuff on different stuff

- Disk
- Machine
- Rack
- Row
- Power feed
- Cluster
- Datacenter
- Region
Physical Isolation

Store stuff on different stuff

- Hardware vendor
- Hardware revision
- Offsite backups
Logical Isolation

Failures cascade
Bugs propagate

Build loosely-coupled systems
Aside:

consensus protocols are poorly isolated
Dropbox West

complicated!

Dropbox East

simple

complicated!
Aside:
secret cold storage project (MPX)
zone

fancy erasure coding!

zone

zone
Operational Isolation

Have to respect isolation in your operations!

- Release process
- Tooling
- Access controls
Release process

- Testing: weeks
- Staging: 1 week
- Zone 1: 1 week
- Zone 2, 3
Isolation
- physical
- logical
- operational

Protection
- safeguards
- recovery
- testing

Verification

Automation
the biggest durability risk is you
gsh --group database lifecycle=reinstall upgrade.sh
gsh --group database lifecycle=reinstall upgrade.sh
gsh --group "database lifecycle=reinstall" upgrade.sh
gsh --group database lifecycle=reinstall upgrade.sh
Post-mortem

“Don’t hate the player, hate the game.”
Post-mortem

“Don’t hate the player, hate the game.”

- Change gsh syntax
gsh --group memcache lifecycle=reinstall -- upgrade.sh
Post-mortem

“Don’t hate the player, hate the game.”

- Change gsh syntax
- Prevent rebooting live memcache/storage/database hosts
- Limit scope of gsh groups
- Build better automation
only execute **sane operations**
team.update(
    where=(team.c.id == self.id),
    values={team.c.quota_used:team.c.quota_used + delta}))
team.update(
    where=(team.c.id == self.id),
    values={team.c.quota_used:team.c.quota_used + delta}))
team.update(
    **where**=(team.c.id == self.id),
    **values**={team.c.quota_used:team.c.quota_used + delta}))

team.update(
    **whereclause**=(team.c.id == self.id),
    **values**={team.c.quota_used:team.c.quota_used + delta}))
SQL Proxy

- Don’t allow updates without a where clause
- Don’t allow drop table statements!

- Connection aggregation
- Transaction rewriting
always have an **undo button**
Identify dangerous transformations

Make sure you can recover from them
Volume
Volume
Volume
Volume lifecycle

- complicated delete process
- delete on storage node
- move volume to “trash”
- unlink
Volume lifecycle

- Complicated delete process
- Delete on storage node
- Move volume to "trash"
- Unlink

Trash inspection
<table>
<thead>
<tr>
<th>Isolation</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical</td>
<td>safeguards</td>
</tr>
<tr>
<td>logical</td>
<td>recovery</td>
</tr>
<tr>
<td>operational</td>
<td>testing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verification</th>
<th>Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>detection</td>
<td></td>
</tr>
<tr>
<td>testing</td>
<td></td>
</tr>
</tbody>
</table>
you will mess up
prioritize failure detection
Is all data in this system 100% correct?

- Prevents excuses later on
<table>
<thead>
<tr>
<th>Scanner</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-zone Verifier</td>
<td>Application-level walker that verifies all data is in storage system and appropriate storage regions</td>
</tr>
<tr>
<td>Storage Watcher</td>
<td>Sampled black-box check that retrieves puts after a minute, hour, day, week, etc.</td>
</tr>
<tr>
<td>Metadata Scanner</td>
<td>Verify all data in the index is on the correct storage nodes</td>
</tr>
<tr>
<td>Disk Scrubber</td>
<td>Verify data on disk is readable and conforms to checksums</td>
</tr>
</tbody>
</table>
Aside: you probably shouldn’t be using a DHT
ugly but it works
untested protections
don’t protect anything
This slide contained a list of extents in the storage system and the various ways we intentionally corrupted them in our staging clusters to ensure they were detected by our verification stack.

Elided from the online version of these slides since they contained datacenter locations etc.
Isolation
physical
logical
operational

Protection
safeguards
recovery
testing

Verification
detection
testing

Automation
responsiveness
safety
you don’t have time
to tend to your storage system
Time to response → Window of vulnerability

When operational load increases, people start cutting corners
Time to response → Window of vulnerability

When operational load increases, people start cutting corners

**Naoru**
- prescription → remediation
you are not as reliable as a shell script
Disk remediation

- bad sector
  - failure
  - repair extent
  - X failures

- bad fs
  - Y failures
  - re-replicate data, reformat disk

- bad disk
  - re-replicate data, file DC Ops ticket
Isolation
physical
logical
operational

Protection
safeguards
recovery
testing

Verification
detection
testing

Automation
responsiveness
safety
james@dropbox.com