Data Checking at Dropbox

David Mah
Dropbox
Problems we are tackling
Examples of Checkers
Generic Model for a Checker
Our garbage collector had a rarely hit off by one bug
Our garbage collector had a rarely hit off by one bug that resulted in removing user data that should not have been deleted
The erasure encoding library we use actually is not thread-safe
The erasure encoding library we use actually is not thread-safe, and in 0.0001% of re-encodes, we would corrupt our user data blocks.
As data passed through a particular machine
As data passed through a particular machine, it would flip some bits of user data.
Some classes of problems

Conditions of Scale
Race Conditions
Hardware Unreliability
Problems we are tackling
Examples of Checkers
Generic Model for a Checker
Block Scrubber

[Checksum 1][Block 1]
[Checksum 2][Block 2]
..
Block Scrubber

[Checksum 1][Block 1]
[Checksum 2][Block 2]

.. Loop over every block, recompute the checksum, compare
Hash Database Scanner

key → [server, server, server ...]
key → [server, server, server ...]
...

Hash Database Scanner

key $\rightarrow$ [server, server, server ...]
key $\rightarrow$ [server, server, server ...]
...

Loop over every key, RPC to those servers, “Do you have this block?”
Filesystem Verifier

File Tree ID → [mutation 1, mutation 2, mutation 3.. ]
File Tree ID → [mutation 1, mutation 2, mutation 3.. ]
(a log of mutations)
Filesystem Verifier

File Tree ID $\rightarrow [\text{mutation 1, mutation 2, mutation 3.. }]$
File Tree ID $\rightarrow [\text{mutation 1, mutation 2, mutation 3.. }]$
(a log of mutations)

Read in rows for a file tree, running 15-20 checks against each
What is the Pattern?
Loop over every ‘unit’
Run a sanity check for each

Not particularly complex
Quantity of checks is high...
Problems we are tackling
Examples of Checkers
Generic Model for a Checker
Data Model

Unit
Data Model

Unit → []Check
Data Model

Unit $\rightarrow$ [ ] Check $\rightarrow$ [ ] Violation
Data Model

Unit $\rightarrow$ []Check $\rightarrow$ []Violation

Partition $\rightarrow$ []Unit
Data Model

Unit $\rightarrow$ [ ] Check $\rightarrow$ [ ] Violation

Partition $\rightarrow$ [ ] Unit

Run $\rightarrow$ [ ] Partition
Check Scheduling

Split the dataset into partitions
Check Scheduling

Split the dataset into partitions
For each partition, maintain a cursor
Check Scheduling

Split the dataset into partitions
For each partition, maintain a cursor
Hand out cursors to check runners
(Use a distributed worker system)
Check Scheduling

RunId: 0
  Partition: “1”, Cursor: “a”
  Partition: “2”, Cursor: “b”
Check Scheduling

RunId: 0
Partition: “1”, Cursor: “a”
Partition: “2”, Cursor: “b”

CheckChunk(Partition, CursorStart) 
Returns [[]Violation, CursorEnd}
Reporting

Shove all Violations into a database.
Dashboard graphs:
  Previous run: num violations per check
  Current run: num violations per check
  Current run: cursor progress
Reporting
Shove all Violations into a database.
Dashboard graphs:
- Previous run: num violations per check
- Current run: num violations per check
- Current run: cursor progress
Alert the team if nonzero
Remediation

Correction scripts are extremely dangerous!

Back-up your data
After correction, re-run checks
Checking the Checkers

Periodically, pick a unit and corrupt it

Make sure the checker detects it
Thanks for stopping by!

David Mah
mah@dropbox.com