# Solidifying The Cloud

## How to back up the Internet

Raymond Blum
Staff Site Reliability Engineer,
International Man of Mystery, Google

# Lessons Learned Backing Up Google

Ensuring durability and integrity of user data is job one

A lapse in availability can be ridden out, but data loss can be hard to recover from, if even possible

Redundancy does not bring recoverability

Corruption and deletes can replicate quite nicely

Distributed processing imposes data consolidation

You need to collect shards into one cohesive world view at some point

The backup process has to scale with data volume

If you haven't restored, you haven't backed up

The Payoff: A case study

# User Data is King

Google

Availability of 99.99% is less than 53 minutes of downtime in a year
*Users will recover from this and life goes on*

Integrity of Four 9s for a 2 Gigabyte artifact is 200+ Kilobytes gone
*It can take a lot of time and effort to get over these effects*

- A document has only a part of its contents
- An executable is useless
- A database is corrupt
- A video is garbled

# Redundancy is Not a Backup

Google

The primary purpose of redundancy of data location is to support scalable processing

- Locality of I/O can be achieved
- This frees you to distribute processing without being tied to remote data sources
- Overall downtime due to local data loss is minimized
  - location X does not drag down location Y

Redundancy is not a guarantee of integrity or recoverability

- Massively parallel systems are massively parallel potential sources of data loss
- The same storage devices run the same software stack
- The same code on the same data means the same latent bugs

Local copies don't protect against site outages

- *That fire is going to melt your* entire *RAID array and the tape drive next to it*
- *A fibre cut to a building won't leave **any** machines in a site connected*

Diversity of storage technologies further guards against bugs taking out data

- A bug in an online storage system is unlikely to reach into an offline (tape) software stack

# The Only Good Backup is One That You've Restored

Google

We want reassurance that the backups are working

- Run continuous restores
- Run automatic comparisons
- Alert us if there are unexpected types or rates of failure

# The Only Good Tape is One That Doesn't Break Until After You're Done With it

Google

Murphy says

*"The tape with the customer's CEO's Inbox is the one that will break when it is mounted for an urgent restore request."*

Things Break

- We expect source data to remain until we notify the customer that their data is on tape
- We handle a tape failure by trying to tape the data again if it is still present and unchanged
- We provide additional protection from media failure through, in effect, "RAID 4 on tape"

Durability is not always evident:

- Tapes are actually more durable than disk (ie. lower failure rates)
- You just don’t see them failing until it’s too late.
  *When do light bulbs burn out?*

# Replicated Data Has to be Consolidated, Eventually...

Google

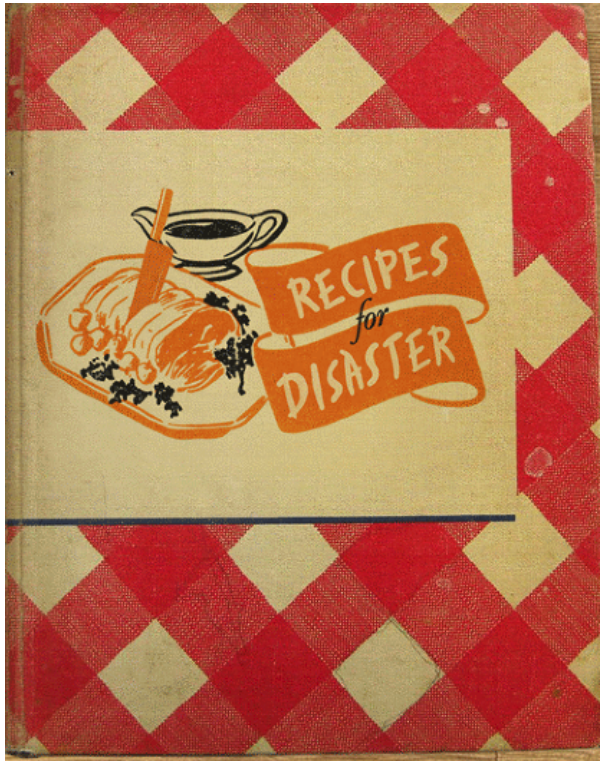
Distributed processing may yield many local data stores
Two approaches offer themselves up

- You can back them each up
  - This presents the greatest flexibility...
  - ...but adds complexity to the recovery* when you have to resolve deltas
- You can back up some resolved or consolidated view
  - This adds some corresponding complexity to the backup...

...but the recovery is more straightforward

We prefer the latter for the gained simplicity of recovery but have found that there are situations that make the former a requirement.

* as seen in...




Oldfield, Avery. Retrieved from “http://www.averyoldfield.com/Recipes-for-Disaster”

*As a rule, when there's a situation calling for data recovery, you don't want to have to do a lot of thinking*

# Backup and Restore Strategies Have To Scale

Google

We all know how to handle backups involving

- One tape drive
- One tape library
- One cluster
- One datacenter/site

Now try dozens of datacenters across the globe
Do you...

- Ship backup data to "backup sites"?
- Make every site a "backup site"?
- Remotely write data to "data sites"?

What are the relative costs and reasonable expectations of each of these options?

# What Needs to be Scalable? Everything!

Google

"At the current rate of growth, within 5 years 30% of the US population will be working as telephone operators" *

* *apocryphal quote of Bell Telephone Company in the early days of telephone adoption*

What saved them?

Automation!

**Automate!  Automate!  Automate!**

If the demand for operations staff and sys admins was linear with backup system usage, we'd ***all*** be doing nothing but backing up the data.

So we automate whatever we can

- Backup scheduling
- Restore testing
- Integrity checking
- Broken tape handling
- Hardware operations tasks
- Logistics
- Library software maintenance

# How Do You Coordinate the Processing of Thousands of Tapes?

Google

We need to do some things that are highly parallelizable

- Collect source data from many locations
- Shard out reconstruction of files
- Verification

**MapReduce** is really good at

- Sharding
- Fault tolerance
- Blocking on dependencies

*It's the Swiss Army knife of distributed processing*

So we use what we've got

# Case Study - the GMail Restore of 2011

Google

1. A series of bugs and mishaps caused a data loss
2. A restore from tape was issued
3. Data was recovered from many tapes
4. User data was restored and verified
5. Whew!

# Distribute the System

Google

Our backup system is globally homed

The site where a request is serviced is not written in stone

- Backup requests are sent to available sites based on a number of criteria
- Restore requests are tied to where the tape resides
- Tapes can be moved

In effect we provide one big planetary backup repository, the physical and operational details of which are transparent to the end-user.

# GMail Restore

How much data?
**How** many tapes?

# The Scale of a GMail Restore

Google

Lets start with some reasonable and unsubstantiated third-party speculations:
http://tech.fortune.cnn.com/2011/02/28/google-goes-to-the-tape-to-get-lost-emails-back/
says: 200,000 tapes

Some facts about LTO4 drives

- An LTO4 drive reads ~ 120MB/s
  - A tape takes ~2h to read

That translates into 400,000 drive hours to read back those tapes

- That's 16,667 drive days

How long can this take?

- The data has to be restored ASAP while still guaranteeing data integrity
- We restored the data in less than 2 days
  - That would require 8,000 drives dedicated to this one task

Typical tape libraries hold several dozen drives each

- That would require ~100 libraries doing nothing else for over a day

Where could you put those if you had them?

- Everywhere, no one room is big enough or one power feed hot enough

# GMail Restore

## How to service an effort of this scale

# Challenges in Balancing Backups and Restores

Google

Backups are known, steady state operations

- Resources and capacity can be planned and monitored
- You have the luxury of scheduling and proactively allocating resources

A restore is usually a surprise, and is a non-maskable interrupt

- You have to either:
  - Preempt backup resources
  - Have sufficient idle available resources for a typical restore
- A restore request is not as portable as a backup request
  - This is a constraint on where the request can be serviced
    Backups only need drives and media, whereas restores need specific volumes

# Summary of Our Strategy

Google

- The more data we have, the greater the need for its durability and integrity
- Efficiency has to keep pace with growth, since resources probably cannot
- Investments in infrastructure pay off
  - Such as applying MapReduce to backup and restore tasks

- We went from a few to dozens of libraries in a short time
  - Without as many times the engineers, ops or elapsed time to deploy

- As scaling is the hardest part...
  - Have a plan to scale up everything in parallel
    - Software
    - Infrastructure
    - Hardware
    - Processes
  - Design your system to be scalable, throughout, from the beginning
  - Effort should not be a linear function of demand (or capacity)

- Don’t take anything for granted!
  - If you’ve not tested it, it’s unknown
    - Eliminating the unknown is what we do
  - DiRT!