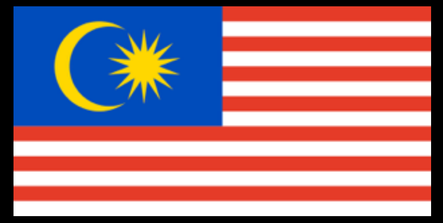
Lessons from database failures

Colin Charles, Team MariaDB, MariaDB Corporation
colin@mariadb.com / byte@bytebot.net
http://www.bytebot.net/blog/ | @bytebot on Twitter
MySQL NYC Meetup, New York, USA
27 June 2016

whoami



- Work on MariaDB at MariaDB Corporation (~~SkySQL Ab~~)
  - Merged with Monty Program Ab, makers of MariaDB
- Formerly MySQL AB (exit: Sun Microsystems)
- Past lives include Fedora Project (FESCO), OpenOffice.org
- MySQL Community Contributor of the Year Award winner 2014

“I run a small hosting provider with more or less 1535 customers and I use Ansible to automate some operations to be run on all servers,” wrote Marco Marsala. “Last night I accidentally ran, on all servers, a Bash script with a rm -rf {foo}/{bar} with those variables undefined due to a bug in the code above this line.”

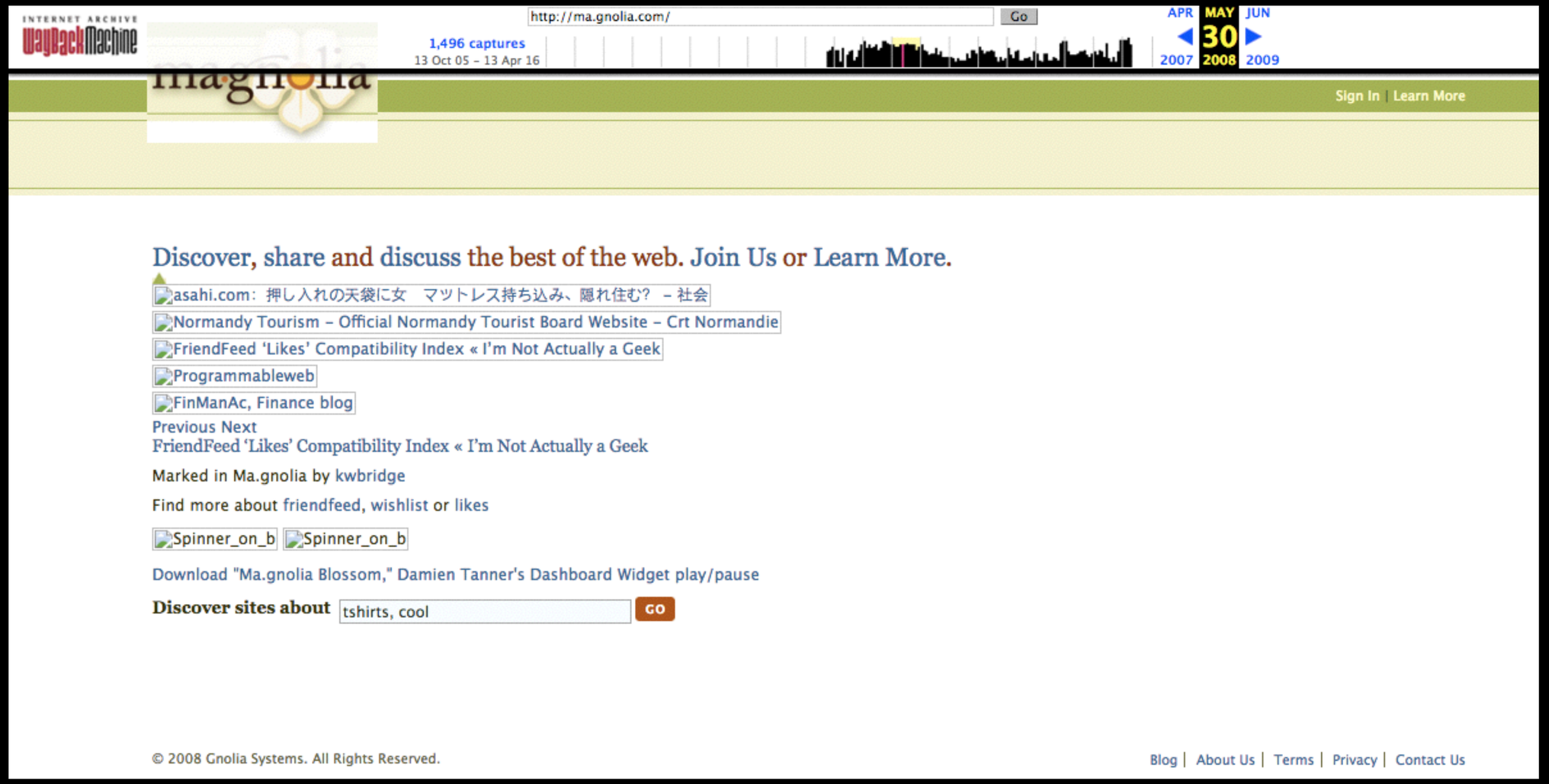
Mr Marsala confirmed that the code had even deleted all of the backups that he had taken in case of catastrophe. Because the drives that were backing up the computers were mounted to it, the computer managed to wipe all of those, too.

“All servers got deleted and the offsite backups too because the remote storage was mounted just before by the same script (that is a backup maintenance script).”

Agenda

- Backups (and verification)
- Replication (and failover)
- Security (and encryption)

ma.gnolia.com



ma.gnolia.com's failure

- January 30 2009: complete outage
- February 17 2009: data corruption in the UDB, essentially dead
- What happened?
  - Ruby on Rails on four self-hosted Mac Mini's, a couple of XServe's, 500GB+ MySQL 5 DB
  - Filesystem corruption, corrupted database backup
  - No versioning, didn't check if the backups worked, made use of `rsync` to backup the database over Firewire network

ma.gnolia.com today?

- EC2 for the app with EBS snapshots, RDS with snapshots, Multi-AZ deployment
- Self-hosted?
  - `xtrabackup`
  - `START TRANSACTION WITH CONSISTENT SNAPSHOT + mysqldump –single-transaction –master-data`
  - Backup a replica
  - Replication event checksums

Couchsurfing, 2006

Two days ago CouchSurfing experienced what could be described as the perfect storm. The database administrators we hired made two critical mistakes. First, we had a major, avoidable hard drive crash. Secondly, the incremental back-ups weren't executed in the correct manner, and twelve of our most important data files didn't survive.

I have been working non-stop trying to repair the data, but as difficult as it is for me to say, it has become clear that certain essential pieces are not recoverable. This crash happened at a particularly vulnerable time, in a transition between two back-up methods. If the crash had happened a week ago, or next week, we would have had a different outcome.

It is with a heavy heart that I face the truth of this situation. CouchSurfing as we knew it doesn't exist anymore. We've had an amazing two and a half years.

Time-delayed replication

- MySQL 5.6+ has time-delayed replication. Stop replication when you know a mistake has happened before it propogates to all the slaves.
- Feature suggestion since 2001! Bug reported August 2006 (mysql#21639). Pushed June 2010 (WL#344). GA February 2013.

Why replicate?

- Scale out
- [automatic] (master) failover
- Geographical redundancy across multiple data centres
- Online schema changes

Replication

- Asynchronous (default)
- Semi-synchronous (plugin)
- Synchronous (Galera, group replication, NDBCLUSTER)
- DRBD

Frameworks

- MySQL-MMM
- SeveralNines ClusterControl
- Orchestrator
- MySQL MHA
- Tungsten Replicator

- 5.6+ utilities: `mysqlfailover`, `mysqlrpladmin`
- Percona Replication Manager
- MariaDB Replication Manager

GitHub

During a maintenance window in mid-August our operations team replaced our aging pair of DRBD-backed MySQL servers with a 3-node cluster. The servers collectively present two virtual IPs to our application: one that's read/write and one that's read-only. These virtual IPs are managed by Pacemaker and Heartbeat, a high availability cluster management stack that we use heavily in our infrastructure. Coordination of MySQL replication to move 'active' (a MySQL master that accepts reads and writes) and 'standby' (a read-only MySQL slave) roles around the cluster is handled by Percona Replication Manager, a resource agent for Pacemaker. The application primarily uses the 'active' role for both reads and writes.

This new setup provides, among other things, more efficient failovers than our old DRBD setup. In our previous architecture, failing over from one database to another required a cold start of MySQL. In the new infrastructure, MySQL is running on all nodes at all times; a failover simply moves the appropriate virtual IP between nodes after flushing transactions and appropriately changing the `read_only` MySQL variable.

GitHub

Monday's migration caused higher load on the database than our operations team has previously seen during these sorts of migrations. So high, in fact, that they caused Percona Replication Manager's health checks to fail on the master. In response to the failed master health check, Percona Replication manager moved the 'active' role and the master database to another server in the cluster and stopped MySQL on the node it perceived as failed.

At the time of this failover, the new database selected for the 'active' role had a cold InnoDB buffer pool and performed rather poorly. The system load generated by the site's query load on a cold cache soon caused Percona Replication Manager's health checks to fail again, and the 'active' role failed back to the server it was on originally.

At this point, I decided to disable all health checks by enabling Pacemaker's `maintenance-mode`; an operating mode in which no health checks or automatic failover actions are performed. Performance on the site slowly recovered as the buffer pool slowly reached normal levels.

GitHub

Upon attempting to disable `maintenance-mode`, a Pacemaker segfault occurred that resulted in a cluster state partition. After this update, two nodes (I'll call them 'a' and 'b') rejected most messages from the third node ('c'), while the third node rejected most messages from the other two. Despite having configured the cluster to require a majority of machines to agree on the state of the cluster before taking action, two simultaneous master election decisions were attempted without proper coordination. In the first cluster, master election was interrupted by messages from the second cluster and MySQL was stopped.

As a result of this data drift, inconsistencies between MySQL and other data stores in our infrastructure were possible. We use Redis to query dashboard event stream entries and repository routes from automatically generated MySQL ids. In situations where the id MySQL generated for a record is used to query data in Redis, the cross-data-store foreign key relationships became out of sync for records created during this window.

Consequentially, some events created during this window appeared on the wrong users' dashboards. Also, some repositories created during this window were incorrectly routed. We've

GitHub

The automated failover of our main production database could be described as the root cause of both of these downtime events. In each situation in which that occurred, if any member of our operations team had been asked if the failover should have been performed, the answer would have been a resounding **no**. There are many situations in which automated failover is an excellent strategy for ensuring the availability of a service. After careful consideration, we've determined that ensuring the availability of our primary production database is not one of these situations. To this end, we've made changes to our Pacemaker configuration to ensure failover of the 'active' database role will only occur when initiated by a member of our operations team.

https://github.com/blog/1261-github-availability-this-week

Fully automated failover a good idea?

- False alarms
- Repeated failover
  - Overloaded master? MHA doesn't allow a failover within 8h, unless — `last_failover_min=n` is set
- Data loss
  - id=103 latest, relay logs at id=101 => loss
  - group commit in the binary log
- Split brain

Proxies

- MariaDB MaxScale
  - MaxScale as binlog server @ Booking - to replace intermediate masters (downloads binlog from master, saves to disk, serves to slave as if served from master)
  - Popular use: load balancing Galera clusters
- MySQL Router + MySQL Fabric
- ProxySQL

Sharding

- SPIDER
- Tungsten Replicator
- Tumblr JetPants

Vitess

- Servers & tools to scale MySQL for web written in Go
- Has MariaDB support too (*)
- Python client interface
- DML annotation, connection pooling, shard management, workflow management, zero downtime restarts
- Become super easy to use: http://vitess.io/ (with the help of Kubernetes)

Failwhales

- Twitter started on MySQL, and is still MySQL - you just need to “evolve”
  - Gizzard (sharding), Mesos + Apache Cotton
- Digg started on MySQL, migrated to Cassandra, and came back to MySQL

Security

- Philippines voter data leave 55m at risk: 338GB MySQL dump
- Ashley Madison: 6.9GB compressed dump, 36m email addresses leaked, 9.6m credit card transactions
- Patreon: 13.7GB MySQL dump, 99 tables

Mossack Fonseca: Panama Papers

An in-depth look at Mossack Fonseca's Drupal installation was performed by Unicorn Riot, which, at the time of their report, noted that a "portfolio" module exists that appears to serve the purpose of allowing account holders to view their documents, and the "Oracle" module exists allowing Drupal to use Oracle as the database backend (rather than the default MySQL).

The Drupal installation itself is vulnerable to the highest-profile vulnerability in the history of the project, a vulnerability which allows any anonymous user to achieve privilege escalation or execute arbitrary PHP code. Unicorn Riot inferred that this vulnerability would have allowed attackers to gain access to the "portfolio" data store, which can reasonably be presumed to be the contents of the Panama Papers provided to the press.

More about IT Security

- Screenshots: Protect your system with one of these five free anti-malware tools
- The dark side of wearables: How they're secretly jeopardizing your security and privacy
- Tech Pro Research Internet and Email usage policy
- Subscribe to our Information Security newsletter

While some attempts have been made to secure the Drupal installation since the disclosure of the Panama Papers, some of the aforementioned security issues appear to persist. Directory view has been disabled, though error pages are still served by Oracle instead of Joomla—the error pages still report the Oracle server as 2.2.15, which is still woefully insecure.

Prevent SQL injections

- MariaDB MaxScale database firewall filter
  - Configurable filter actions on rule match (Allow the query, block the query or ignore the match), Logging of matching and/or non-matching queries
- MySQL Enterprise firewall

Encryption at rest

- MariaDB Server 10.1: table or tablespace encryption
  - design goal: Encrypt all user data that may touch the disk — InnoDB data, InnoDB logs, binary logs, temporary tables, temporary files
  - key management on the filesystem? [no key rotation] Amazon KMS?
  - caveats: `mysqlbinlog` needs work with encrypted binlogs; Galera Cluster gcache isn't encrypted
- MySQL 5.7: only encrypts InnoDB tablespaces (innodb_file_per_table; logs unencrypted)

In conclusion…

- Use semi-sync replication with a failover solution that ensures you don't failover too often
- Make good backups. Test them. Save them.
- You'll most definitely need to shard your data, use proven frameworks and get a proxy involved. Complete backups with multi-source replication when needed.
- Use `mysqldump` and `xtrabackup` together (and `mydumper` for parallel backup/restore; `mysqlpump`)
- Security is key: prevent SQL injections, encrypt your data at rest

It’s 2016, you don’t want this…

On Friday morning, *Malaysiakini* encountered technical difficulties that resulted in the editorial team being unable to upload new stories to the website.

The problem was due to a catastrophic hardware failure of our content database server.

Working with our cloud hosting providers, the technology team managed to gain access to the content database, but were unable to fully recover the database. Unfortunately, the backups were also not usable.

In order to keep the stories flowing, new Malaysiakini stories were published on *Malaysiakini’s* English, BM and Chinese Facebook pages.

Our technology team is working around the clock to put together a system that will support the website’s core publishing functions.

Publishing on the website will resumed soon, although some functions may not be fully operational. We estimate it will take 10 to 14 days to make a full recovery.

Thank you. Q&A?

colin@mariadb.com / byte@bytebot.net
@bytebot on Twitter | http://www.bytebot.net/blog/
slides: slideshare.net/bytebot