5 Proactive Measures to Minimize MySQL Downtime

Fernando Ipar
Percona Webinars 2013
About me...

- Using MySQL since 2000
- With Percona since 2009
  - Consulting, Emergencies, Data Recovery, etc.
About this talk...

- 100% based on real world experience
- Involves failures, problems, disaster.
- How to avoid them or minimize their impact.
what about disaster?
It’s bound to happen!
(total of 556 analyzed cases)

- **Avoidable or Mitigated**: 43%
- **Unavoidable**: 57%
The measures
1. Keep tested backups
Things that are not a backup:
- Replica
- RAID
- DRBD/SAN
- any **faithful** and **live** copy of your data.
Things that are a backup:
- mysqldump
- xtrabackup
- FS snapshot
- any faithful and consistent copy of your data at one point in time.
Things to consider
- Production impact
- Purpose
- Data set size
- Backup/Restore time
Testing backups
- If it’s not tested, it’s not a backup
- Restore to dev/stage/etc.
- Restore to a replica, then pt-table-checksum
- Big Data considerations
2. Build redundant components
Things that are **not** a backup but that provide you redundancy:

- Replica
- RAID
- DRBD/SAN
- any **faithful** and **live** copy of your data.
If possible, support a degraded mode.
- fallback to static content
- disable writes
- modularity and configurability are important for this
3. Test changes
- Test upgrades
  - pt-upgrade / pt-table-checksum
Testing upgrades

hostA
Production MySQL
(5.1.X)
Testing upgrades

hostA
Production MySQL (5.1.X)

replication

hostB
Target MySQL (5.5.X)
Testing upgrades

- hostA
  Production MySQL (5.1.X)

- replication

- hostB
  Target MySQL (5.5.X)

pt-table-checksum --replicate percona.checksums h=hostA

pt-upgrade queries.txt h=hostA h=hostB
- Test upgrades
  - pt-upgrade / pt-table-checksum
- Test app changes
  - Jmeter, sysbench, mysqlslap, etc.
- Test **realistic** workloads and data set **sizes**.
CREATE TABLE `bad_scaling_test` (  `id` int NOT NULL AUTO_INCREMENT,  `payload` text,  `rank` int unsigned DEFAULT '0',  PRIMARY KEY (`id`) ) ENGINE=InnoDB

select * from test.bad_scaling_test order by rank desc limit 10
Testing app changes

- without key
- with key

Response time (s)

Concurrency (# of threads)

Response time with key:

- 1 thread: 0.25 s
- 5 threads: 0.25 s
- 20 threads: 3 s
- 50 threads: 7.5 s
- 100 threads: 15 s

Response time without key:

- 1 thread: 0.25 s
- 5 threads: 0.25 s
- 20 threads: 3 s
- 50 threads: 7.5 s
- 100 threads: 15 s

Note: The chart shows the response time for app changes with and without a key for different concurrency levels.
- Test upgrades
  - pt-upgrade / pt-table-checksum
- Test app changes
  - Jmeter, sysbench, mysqlslap, etc.
  - Test realistic workloads and data set sizes.
- Have a rollback plan
- Link queries to code paths

SELECT ... /* /app/cart/get */
4. Practice safe replication
- Prefer RBR when possible
- Make sure you have PK/UK
- Prefer RBR when possible
  - Make sure you have PK/UK
- Monitor lag (pt-heartbeat)
- Prefer RBR when possible
  - Make sure you have PK/UK
- Monitor lag (pt-heartbeat)
- Avoid huge trx on master
- Prefer RBR when possible
  - Make sure you have PK/UK
- Monitor lag (pt-heartbeat)
- Avoid huge trx on master
- Set passive nodes @@read_only
- Reserve SUPER for admin accounts
- Prefer RBR when possible
  - Make sure you have PK/UK
- Monitor lag (pt-heartbeat)
- Avoid huge trx on master
- Set passive nodes @@read_only
- Reserve SUPER for admin accounts
- Periodic pt-table-checksum
5. Monitor, Trend
  - Nagios
  - Cacti
  - Zabbix, Munin, Zenoss, etc.
- Avoids or minimizes impact of:
  - running out of disk space
  - degraded HW
  - reaching max_connections
  - workload changes
  - etc.

- Provides ‘Normal’ baseline when troubleshooting
The take home message