Tips from the Trenches
Preventing downtime for the over extended DBA

Andrew Moore
Senior Remote DBA
Percona Managed Services
Your Presenter

- Andrew Moore
- @mysqlboy on twitter
- 1+ year in Manager Services Team
- UK based Remote DBA
- Open Source Technology (docker, ELK)
Percona Managed Services

- Percona Remote DBA for MySQL® database administration
  - 24x7 around the clock support
  - USA, South America, Europe, APAC
- Percona Backup Service for backup management
  - Standalone backup service
  - Fully managed and monitored
- Mydumper and MHA contributors
Agenda

• What or who is the 'over extended' DBA?
• The reason for some of this avoidable downtime
• How to defend your systems against it
The Over Extended DBA
The Over Extended DBA++

- Usually the
  - CTO/CEO
  - Software Developer
  - System Administrator
  - DBA of another discipline (SQLServer, Oracle)
  - MySQL DBA with an already full schedule
Symptoms of the OED

- Missing backups
- Frequent outages
- Performance problems
- Lack of HA
- Legacy MySQL versions
- Pager firefighter
Downtime

“periods when a system fails to provide it's primary function”
What Downtime?

- Disk is full
- Rogue Statements
- Slow Statements
- Backups
- Blocking DDL
- No HA plan

- Hardware failure
- Software bugs
- Inadequate configuration
- Data inconsistencies
Disk is Full

• What happens?
  • MySQL unable to write to data files
  • MySQL unable to write binary logs
  • MySQL unable to write in tmpdir

• How to avoid?
  • Monitor
  • Reactive Proactive
  • Compression
Disk is Full

• How to avoid?
  • Monitor
    – percona-monitoring-plugins
  • Reactive Proactive
    – Archiving strategy
    – Capacity planning, add space in good time
  • Compression
    – Data compression
    – Log compression

I'm a squash. Get it?
**Disk is Full**

- **How to avoid?**
  - **Compression**
    - Data compression
      - Compress data in the application
      - MySQL compress() function
      - InnoDB Row Compression
      - TokuDB Engine
      - Archive Engine (hmmm)
    - Log compression
      - Cron job to run gzip and manage PURGE
      - Don't compress logs being used!

I'm squashed?
Inadequate Configuration

- Why it happens?
  - Over allocation
  - Under allocation
  - my-large.cnf

- How to avoid?
  - [https://tools.percona.com/wizard](https://tools.percona.com/wizard)
  - Observe performance monitoring
  - Avoid scripts such as mysqltuner.pl
  - Revisit
Inadequate Configuration

• How to avoid?

  • `innodb_buffer_pool_size` – if you have a dedicated MySQL server, set it to 75% of total available RAM
  
  • `innodb_buffer_pool_instances` –
    – on MySQL 5.5, set it to 4,
    – on MySQL 5.6 – 8 or even 16.
  
  • `innodb_flush_log_at_trx_commit`
  
  • `innodb_flush_method = O_DIRECT`
  
  • `innodb_log_file_size = ~ 1 hour of transactions`
Inadequate Configuration

• How to avoid?
  • Check into revision control (git, svn)
  • Know what you are modifying and why, then document
  • Change one thing at a time and observe the effect
Rogue SQL Statements

• Why it happens?
  • DML accidents
  • Large update/delete operations
  • Oops, I thought I was on DEV env!

• How to avoid?
  • Access limitation
  • Know your DML
  • Have reliable logical backups!
Rogue SQL Statements

• How to avoid?
  • Access limitation
    – Only GRANT users/apps access and privileges to what they really need
  • Know your DML
    – Statements like REPLACE and TRUNCATE may appear obvious but might perform actions that you're not expecting in some circumstances
  • sql_safe_updates or WHERE
  • Ensure backups are tested!

What's a WHERE clause?
Rogue SQL Statements

- How to avoid?
  - Large UPDATE/DELETE
    - Oak-chunk-update
    - Pt-archiver (delete/move)
    - common_schema (query_script)
  - [https://code.google.com/p/common-schema/](https://code.google.com/p/common-schema/)
Rogue SQL Statements

• How to avoid?

mysql> CREATE TABLE buildings (  
    -> building_no int(11) NOT NULL AUTO_INCREMENT,  
    -> building_name varchar(255) NOT NULL,  
    -> address varchar(355) NOT NULL,  
    -> PRIMARY KEY (building_no)  
    -> ) ENGINE=InnoDB;  
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE rooms (  
    -> room_no int(11) NOT NULL AUTO_INCREMENT,  
    -> room_name varchar(255) NOT NULL,  
    -> building_no int(11) NOT NULL,  
    -> PRIMARY KEY (room_no),  
    -> KEY building_no (building_no),  
    -> CONSTRAINT rooms_ibfk_1  
    -> FOREIGN KEY (building_no)  
    -> REFERENCES buildings (building_no)  
    -> ON DELETE CASCADE  
    -> ) ENGINE=InnoDB;  
Query OK, 0 rows affected (0.01 sec)
Rogue SQL Statements

• How to avoid?

```sql
mysql> INSERT INTO buildings(building_name,address)
    -> VALUES ('ACME Headquaters', 
    -> '3950 North 1st Street CA 95134'),
    -> ('ACME Sales','5000 North 1st Street CA 95134');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO rooms(room_name,building_no)
    -> VALUES('Amazon',1),
    -> ('War Room',1),
    -> ('Office of CEO',1),
    -> ('Marketing',2),
    -> ('Showroom',2)
    -> ;
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```
Rogue SQL Statements

- How to avoid?

```sql
mysql> select * from rooms;
+---------+---------------+-------------+
| room_no | room_name     | building_no |
+---------+---------------+-------------+
|       1 | Amazon        |           1 |
|       2 | War Room      |           1 |
|       3 | Office of CEO |           1 |
|       4 | Marketing     |           2 |
|       5 | Showroom      |           2 |
+---------+---------------+-------------+
5 rows in set (0.00 sec)

mysql> select * from buildings;
+-------------+------------------+--------------------------------+
| building_no | building_name    | address                        |
+-------------+------------------+--------------------------------+
|           1 | ACME Headquaters | 3950 North 1st Street CA 95134 |
|           2 | ACME Sales       | 5000 North 1st Street CA 95134 |
+-------------+------------------+--------------------------------+
2 rows in set (0.00 sec)
```
Rogue SQL Statements

• How to avoid?

```sql
mysql> REPLACE INTO buildings
    
    (building_no,building_name,address) VALUES (2,'ACME Revenue','4999 North 1st Street CA 95134');
Query OK, 2 rows affected (0.00 sec)

mysql> select * from buildings;
+-------------+------------------+--------------------------------+
| building_no | building_name    | address                        |
+-------------+------------------+--------------------------------+
|           1 | ACME Headquaters | 3950 North 1st Street CA 95134 |
|           2 | ACME Revenue     | 4999 North 1st Street CA 95134 |
+-------------+------------------+--------------------------------+
2 rows in set (0.00 sec)

mysql> select * from rooms;
+---------+---------------+-------------+
| room_no | room_name     | building_no |
+---------+---------------+-------------+
|       1 | Amazon        |           1 |
|       2 | War Room      |           1 |
|       3 | Office of CEO |           1 |
+---------+---------------+-------------+
3 rows in set (0.00 sec)
```
Slow SQL Statements

- Why it happens?
  - Lack of feedback to development
  - Questionable design
  - Long history list
  - Poor disk speeds
  - Bad configuration

- How to avoid?
  - Review SQL regularly
  - Indexes
  - Cache
Review SQL Statements

- How to avoid?
  - Mysql slow log
  - Pt-query-digest (--processlist)
  - Explain
  - Deploy
Review SQL Statements

- Percona Cloud Tools – https://cloud.percona.com

Query Profile

<table>
<thead>
<tr>
<th>Rank</th>
<th>Query (distilled)</th>
<th>Query ID</th>
<th>Queries</th>
<th>QPS</th>
<th>Load</th>
<th>Load %</th>
<th>Total Time</th>
<th>Avg Time</th>
<th>99% Max Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMMIT</td>
<td>813081EB808CDB3EE3D29</td>
<td>329.7K</td>
<td>216.97</td>
<td>1.963</td>
<td>44.36%</td>
<td>50m14s</td>
<td>9.17ms</td>
<td>212.43ms</td>
</tr>
<tr>
<td>2</td>
<td>INSERT order_line</td>
<td>108EBF5T21A765F6</td>
<td>125.14K</td>
<td>82.33</td>
<td>0.593</td>
<td>15.17%</td>
<td>16m10s</td>
<td>7.23ms</td>
<td>16.79ms</td>
</tr>
<tr>
<td>3</td>
<td>SELECT stack</td>
<td>9E61F6668A9BB466</td>
<td>109.09K</td>
<td>71.74</td>
<td>0.436</td>
<td>9.67%</td>
<td>6m10s</td>
<td>6.10ms</td>
<td>14.48ms</td>
</tr>
<tr>
<td>4</td>
<td>UPDATE stock</td>
<td>6F70D41CD63ACD21</td>
<td>93.06K</td>
<td>61.22</td>
<td>0.392</td>
<td>8.7%</td>
<td>9m45s</td>
<td>6.43ms</td>
<td>14.82ms</td>
</tr>
<tr>
<td>5</td>
<td>SELECT stack</td>
<td>898D455A5690D0C3</td>
<td>64.87K</td>
<td>42.68</td>
<td>0.273</td>
<td>6.07%</td>
<td>6m20s</td>
<td>6.42ms</td>
<td>14.46ms</td>
</tr>
<tr>
<td>6</td>
<td>SELECT fam</td>
<td>BB4A4C7016F28EAE</td>
<td>52.84K</td>
<td>34.63</td>
<td>0.217</td>
<td>4.83%</td>
<td>5m30s</td>
<td>6.29ms</td>
<td>14.01ms</td>
</tr>
<tr>
<td>7</td>
<td>UPDATE order_line</td>
<td>A0532A7A5F9CD5DF</td>
<td>47.87K</td>
<td>31.43</td>
<td>0.153</td>
<td>4.28%</td>
<td>4m53s</td>
<td>6.14ms</td>
<td>16.79ms</td>
</tr>
<tr>
<td>8</td>
<td>UPDATE customer</td>
<td>B40A246086D2774</td>
<td>42.07K</td>
<td>7.94</td>
<td>0.051</td>
<td>1.14%</td>
<td>1m17s</td>
<td>6.60ms</td>
<td>15.06ms</td>
</tr>
<tr>
<td>9</td>
<td>SELECT order_line</td>
<td>DCD0952BD16DA4A4</td>
<td>10.60K</td>
<td>6.98</td>
<td>0.046</td>
<td>1.03%</td>
<td>1m10s</td>
<td>6.65ms</td>
<td>15.65ms</td>
</tr>
<tr>
<td>10</td>
<td>SELECT order_line</td>
<td>AC360B122042A66</td>
<td>3.29K</td>
<td>2.16</td>
<td>0.043</td>
<td>0.95%</td>
<td>1m05s</td>
<td>47.71ms</td>
<td>230.83ms</td>
</tr>
</tbody>
</table>

10 queries, 93.99%
Blocking DDL Statements

- Why it happens?
  - Rushed DDL
  - Un-vetted Upgrade script
  - Misunderstanding of DDL in MySQL

- How to avoid?
  - Use Online DDL of 5.6
  - Use online schema change tool
  - Perform ALTER table in maintenance window
Blocking DDL Statements

• How to avoid?
  • Use Online DDL of 5.6
    – ALGORITHM=INPLACE
    – LOCK=NONE
  • Use Percona's pt-online-schema-change
    – Alter table without blocking DML
    – Take care with FK
  • Perform ALTER table in maintenance window
    – If your change is not compatible with pt-osc run direct alter in a time where blocking is permitted
Blocking DDL Statements

- **How to avoid?**

  - Monitoring on processlist
    
      - *pmp-check-mysql-processlist*

```
mysql> select user, db, command, time, state, left(info,50) from information_schema.processlist;
+---------------+------+---------+------+---------------------------------+----------------------------------------------------+
| user          | db   | command | time | state                           | left(info,50)                                      |
+---------------+------+---------+------+---------------------------------+----------------------------------------------------+
| percona-agent | NULL | Sleep   |    0 |                                 | NULL                                               |
| root          | test | Query   |  347 | Waiting for table metadata lock | alter table rooms add key name (room_name)         |
| root          | test | Query   |  231 | Waiting for table metadata lock | insert into rooms (...)                          |
| root          | test | Query   |  132 | Waiting for table metadata lock | insert into rooms (...)                          |
| root          | NULL | Query   |    0 | executing                       | select user, db, command, time, state, left(info,5 |
+---------------+------+---------+------+---------------------------------+----------------------------------------------------+
6 rows in set (0.00 sec)
```
DDL

5.6 Online Alter
MySQL 5.6 offers online DDL in certain scenarios.
ALGORITHM:=INPLACE and LOCK=NONE

Direct Alter
Small tables will alter quickly. Large tables will be locked for the duration or copying the data. Deferring the change until period of allowed downtime or light load unless an emergency change.

Slave Swap
Perform the DDL on the slave(s) and once complete promote the slave, repeat steps on previous master to avoid locking the table from DML whilst the change completes.

pt-osc Extended
Attention to workload and foreign keys review the pt-osc docs to ensure all potential pitfalls are addressed
--noswap-tables
--notouch-old-table

pt-osc
Use pt-online-schema-change to provide low impact table maintenance. It is always best to review the documentation and remember --dry-run before --execute
http://bit.ly/1umr65f

Key
least locking
manual intervention
most locking
High Availability

• Why it happens?
  • HA not in place (yet)
  • The wrong HA solution in place

• How to avoid?
  • Deploy HA solution (MHA, PRM, Heartbeat, Pacemaker, MMM, PXC) and learn it
  • Seek guidance if HA remains an issue, your business might depend on it
High Availability

• How to avoid?
  • MHA
    – Provides simple role changes using VIP
    – Takes lifting out of slave reconfiguration
    – Can provide automatic or managed failover
  
• https://code.google.com/p/mysql-master-ha/
Hardware Failure

• Why it happens?
  • Failed disks are common place
  • Single points of failure (SAN)

• How to avoid?
  • RAID HDD
  • Commercial grade components
  • Use HA for failover means
  • Consider deployment to cloud provider to reduce $TCO
Software Bugs

- Why it happens?
  - Software is not perfect
  - Legacy versions in use
  - Legacy configuration in use

- How to avoid?
  - Ensure upgrades are deployed and tested
  - Observe release notes
Software Bugs

• How to avoid?
  • Ensure upgrades are deployed and tested
    - Subscribe to your vendor's mailing list
    - Deploy in QA/Dev environments before Prod
  • Observe release notes
    - Most vendors produce lists of known bugs or maintain a bugs database
      • http://bugs.mysql.com
      • https://bugs.launchpad.net/percona-server
Questions?
Using Percona to Prevent Downtime

- Extra help and guidance
  - Percona Consulting and Support Services for MySQL®
  - Inject expertise or compliment your existing team
    - Percona Remote DBA for MySQL® database administration
    - Percona Backup Service for backup management

www.percona.com/about-us/careers
www.percona.com/contact/sales