



Database Defense in Depth

Geoffrey Anderson
Percona Live 2015





Geoffrey Anderson

- DBA @ Box, Inc.
- MySQL, HBase, and more!
- #DBHangOps





box



OUR VISION:

Share, manage and access your
content from any device, anywhere





275K+
BUSINESSES



32MM+
USERS



99%
FORTUNE 500



Global Enterprises Trust Box



HEALTHCARE



INDUSTRIAL



HIGH TECH



MEDIA



RETAIL



SERVICES





Defenses? For a Database?

Huh?



Defenses

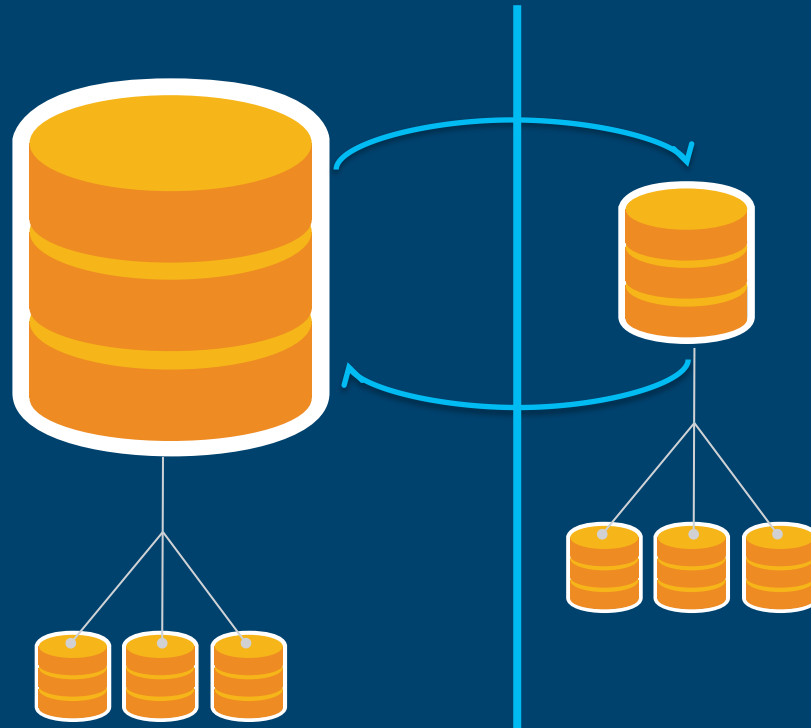
- Query Tuning
- Anemometer
- Raingauge
- Table Partitioning
- User Management / Least Privilege
- Graphs / Monitoring
 - OpenTSDB / Graphite / Kibana
 - statsd / tcollector
- user_statistics plugin
- Percona Server
- Proxying
- Sharding
- Vertically Scaling
- Replication
 - Multiple read replicas
- Backups
 - ...and recovery!
- Binary Logs
- Percona Toolkit
 - pt-query-digest
 - pt-slave-delay
 - pt-stalk
 - pt-table-checksum
 - pt-table-sync
- Killing connections
- Killing bad queries
- Killing bad transactions
- Nagios alerts / PagerDuty
- Innotop
- Health checking

<http://ceilingcat.ninja>



Let's take a journey

You have your database





Queries



Yay business!



Queries

```
mysql [localhost] {root} (sakila) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
10	root	localhost	sakila	Query	0	init	show processlist
426586	unauthenticated user	localhost	NULL	Connect	NULL	Reading from net	NULL
426596	app	localhost	sakila	Query	0	Writing to net	SELECT * FROM rental WHERE customer_id=908 AND return_date IS NULL
426597	app	localhost	sakila	Query	0	Opening tables	SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL
426598	app	localhost	sakila	Query	0	Opening tables	SELECT * FROM rental WHERE customer_id=825 AND return_date IS NULL
426599	app	localhost	sakila	Query	0	Opening tables	SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL
426600	app	localhost	sakila	Query	0	Opening tables	SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL
426601	app	localhost	sakila	Query	0	Opening tables	SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL

139 rows in set (0.00 sec)



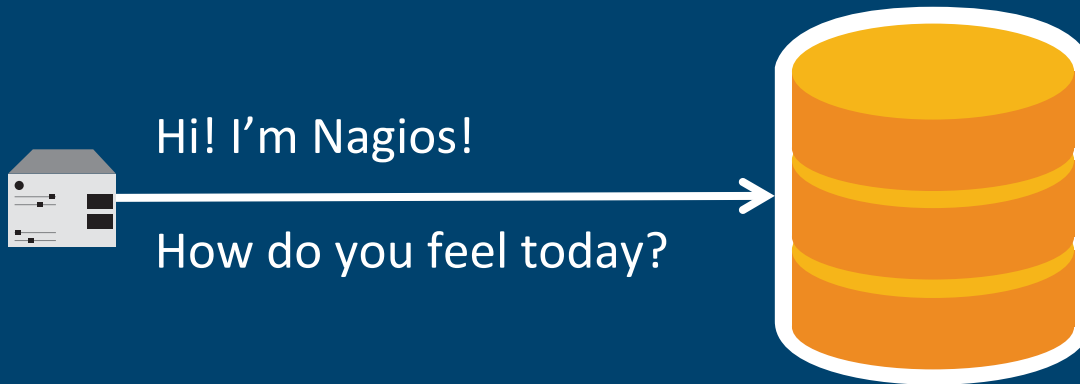


Defense #1 – Query comments!

Queries

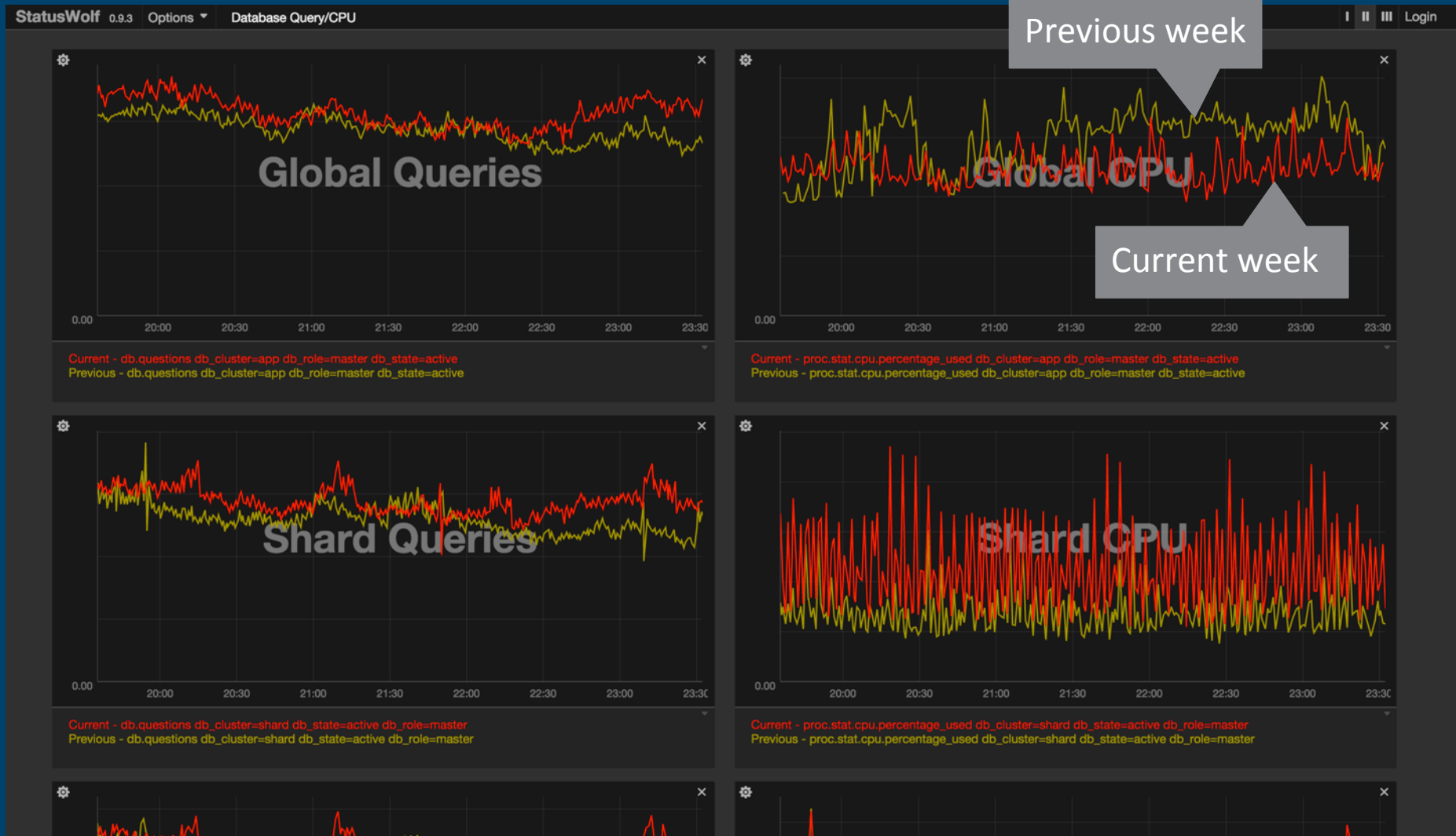
```
mysql [localhost] {root} (sakila) > show full processlist\G
***** 1. row *****
  Id: 10
  User: root
  Host: localhost
  db: sakila
  Command: Query
  Time: 0
  State: init
  Info: show full processlist
***** 2. row *****
  Id: 573358
  User: app
  Host: localhost
  db: NULL
  Command: Sleep
  Time: 0
  State:
  Info: NULL
***** 3. row *****
  Id: 573374
  User: app
  Host: localhost
  db: sakila
  Command: Query
  Time: 0
  State: Sending data
  Info: SELECT * FROM rental WHERE customer_id=491 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(349)::Customer_Login(1493) request=5021e5e46a457da5e89782f80b7d9abc2e64d0e0&user=491 */
***** 4. row *****
  Id: 573375
  User: app
  Host: localhost
  db: sakila
  Command: Query
  Time: 0
  State: statistics
  Info: SELECT * FROM rental WHERE customer_id=491 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(349)::Customer_Login(1493) request=5021e5e46a457da5e89782f80b7d9abc2e64d0e0&user=491 */
```

Defense #2 – Alerting on the Database

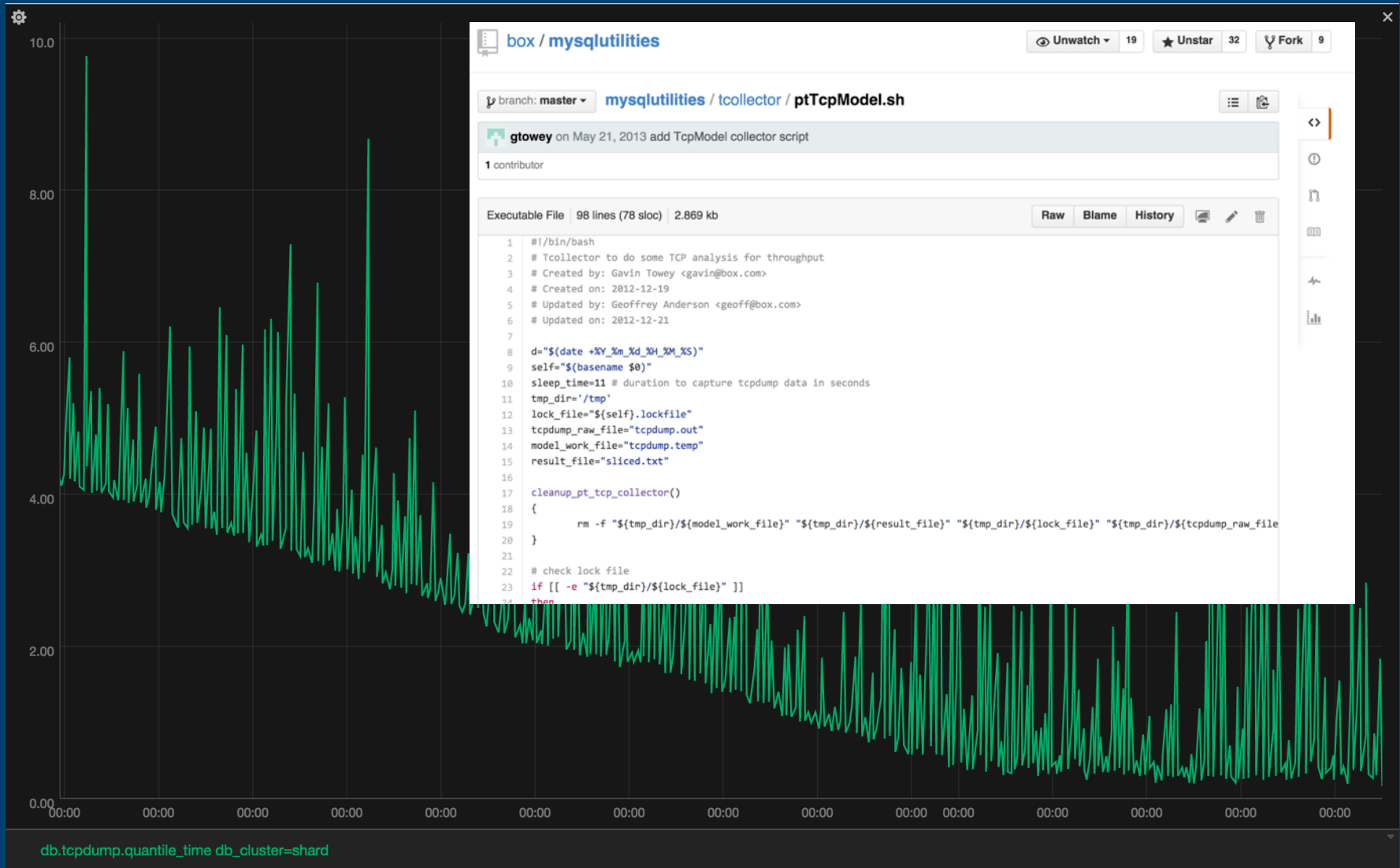


MySQL		OK	21:17:06	31d 9h 16m 24s	1/4	IO: Yes Slave SQL: Yes Seconds Behind Master: 0
MySQL - Long Queries		OK	21:16:56	31d 9h 18m 44s	1/4	OK - no long running queries
MySQL Config Changes		OK	21:14:56	26d 10h 19m 3s	1/4	OK creating temp file in /tmp/mysql_vars
MySQL Connections		OK	21:17:07	24d 7h 35m 45s	1/4	OK utilization=3% threads_connected=1043 max_connections=30000
MySQL History List		OK	21:14:56	31d 9h 16m 33s	1/4	Innodb_history_list_length = 1561
MySQL Read Only		OK	21:17:06	18d 10h 45m 54s	1/4	global.read_only = 0: db_role = master
MySQL Replication		OK	21:17:07	0d 1h 23m 6s	1/4	Replication OK.
MySQL Slow Query Log		OK	21:17:16	0d 0h 0m 6s	1/4	global.slow_query_log = 0

Defense #3 – Dashboards



Defense #3 – Dashboards



Defense #4 – User Statistics



```
geoff@db.example.net [(none)]> show user_statistics\G
***** 1. row *****
User: geoff
Total_connections: 24
Concurrent_connections: 0
Connected_time: 0
Busy_time: 0
Cpu_time: 0
Bytes_received: 2112
Bytes_sent: 0
Binlog_bytes_written: 0
Rows_fetched: 48
Rows_updated: 0
Table_rows_read: 0
Select_commands: 24
Update_commands: 0
Other_commands: 24
Commit_transactions: 0
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
***** 2. row *****
User: #mysql_system#
Total_connections: 1
Concurrent_connections: 0
Connected_time: 19
Busy_time: 0
Cpu_time: 0
Bytes_received: 0
Bytes_sent: 0
Binlog_bytes_written: 373033
Rows_fetched: 0
Rows_updated: 3775
Table_rows_read: 2624
Select_commands: 0
Update_commands: 1178
Other_commands: 34
Commit_transactions: 34
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
2 rows in set (0.00 sec)
```

Per-user connection counts

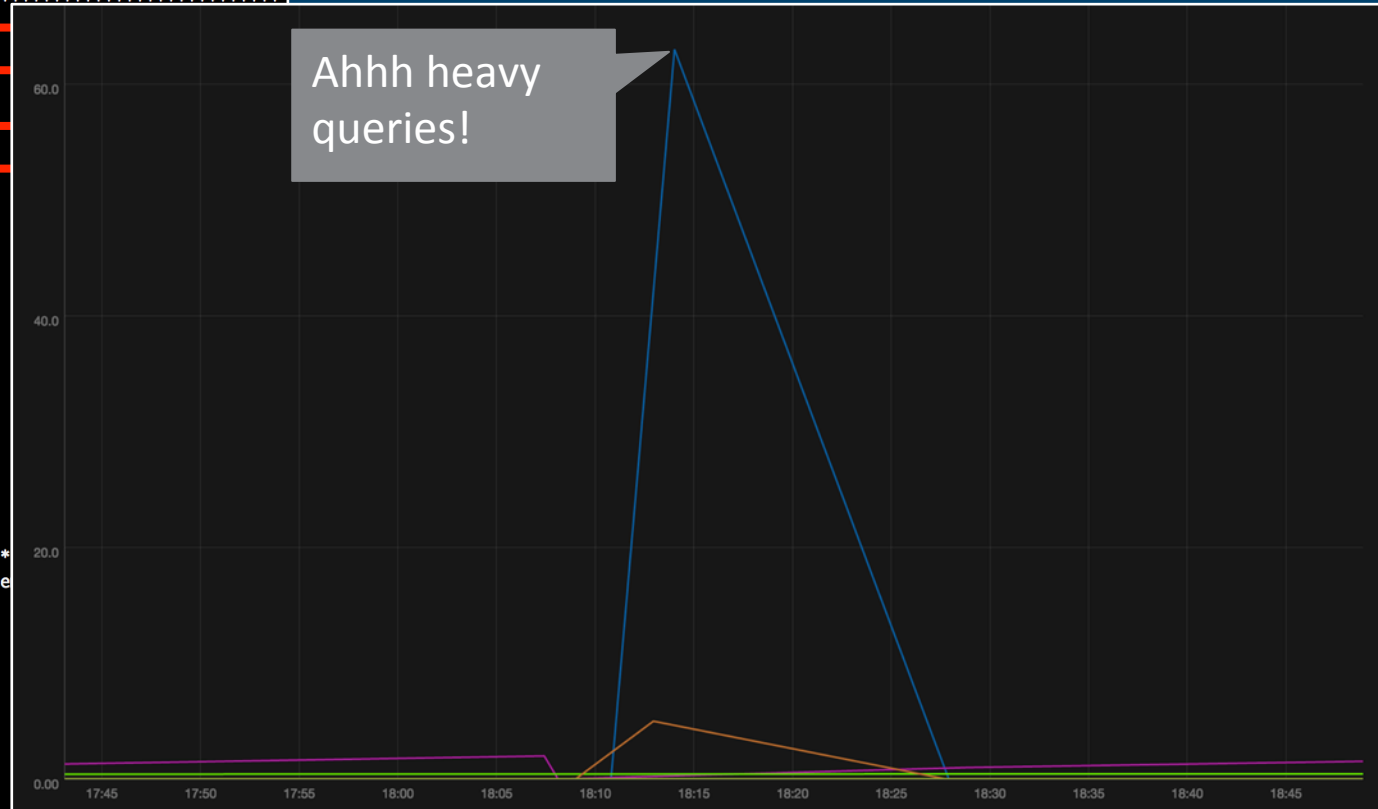
Per-user cpu time

```
mysql> select user, host, user_time, sys_time from sys.user_summary_by_file_io;
+-----+-----+-----+-----+
| root      |      | 159 | 5.11 h |
| app       |      | 43777705 | 57.06 m |
| background |      | 2174 | 754.67 ms |
+-----+-----+-----+-----+
3 rows in set (0.04 sec)
```

Defense #4 – User Statistics



```
geoff@db.example.net [(none)]> show user_statistics\G
***** 1. row *****
User: geoff
Total_connections: 24
Concurrent_connections: 0
Connected_time: 0
Busy_time: 0
Cpu_time: 0
Bytes_received: 2112
Bytes_sent: 0
Binlog_bytes_written: 0
Rows_fetched: 48
Rows_updated: 0
Table_rows_read: 0
Select_commands: 24
Update_commands: 0
Other_commands: 24
Commit_transactions: 0
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
***** 2. row *
User: #mysql_system
Total_connections: 1
Concurrent_connections: 0
Connected_time: 19
Busy_time: 0
Cpu_time: 0
Bytes_received: 0
Bytes_sent: 0
Binlog_bytes_written: 373033
Rows_fetched: 0
Rows_updated: 3775
Table_rows_read: 2624
Select_commands: 0
Update_commands: 1178
Other_commands: 34
Commit_transactions: 34
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
2 rows in set (0.00 sec)
```



Defense #4 – User and Table statistics



```
geoff@db.example.net [(none)]> show user_statistics\G
```

```
***** 1. row *****
User: geoff
Total_connections: 24
```

```
ganderson@ops-db.ve.box.net [(none)]> show table_statistics;
```

Table_schema	Table_name	Rows_read	Rows_changed	Rows_changed_x_indexes
sakila	rental	117535	3927	9382
sakila	film	42504	44	1100

```
2 rows in set (0.00 sec)
```

Per-table and per-schema counts!

```
Select_commands: 24
Update_commands: 0
Other_commands: 0
Commit_transactions: 0
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
***** 2. row *****
User: #mysql_system
Total_connections: 1
Concurrent_connections: 0
Connected_time: 19
Busy_time: 0
Cpu_time: 0
Bytes_received: 0
Bytes_sent: 0
Binlog_bytes_written: 373033
Rows_fetched: 0
Rows_updated: 3775
Table_rows_read: 2624
Select_commands: 0
Update_commands: 1178
Other_commands: 34
Commit_transactions: 34
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 0
Total_ssl_connections: 0
2 rows in set (0.00 sec)
```



Defense #4 – User and Table statistics



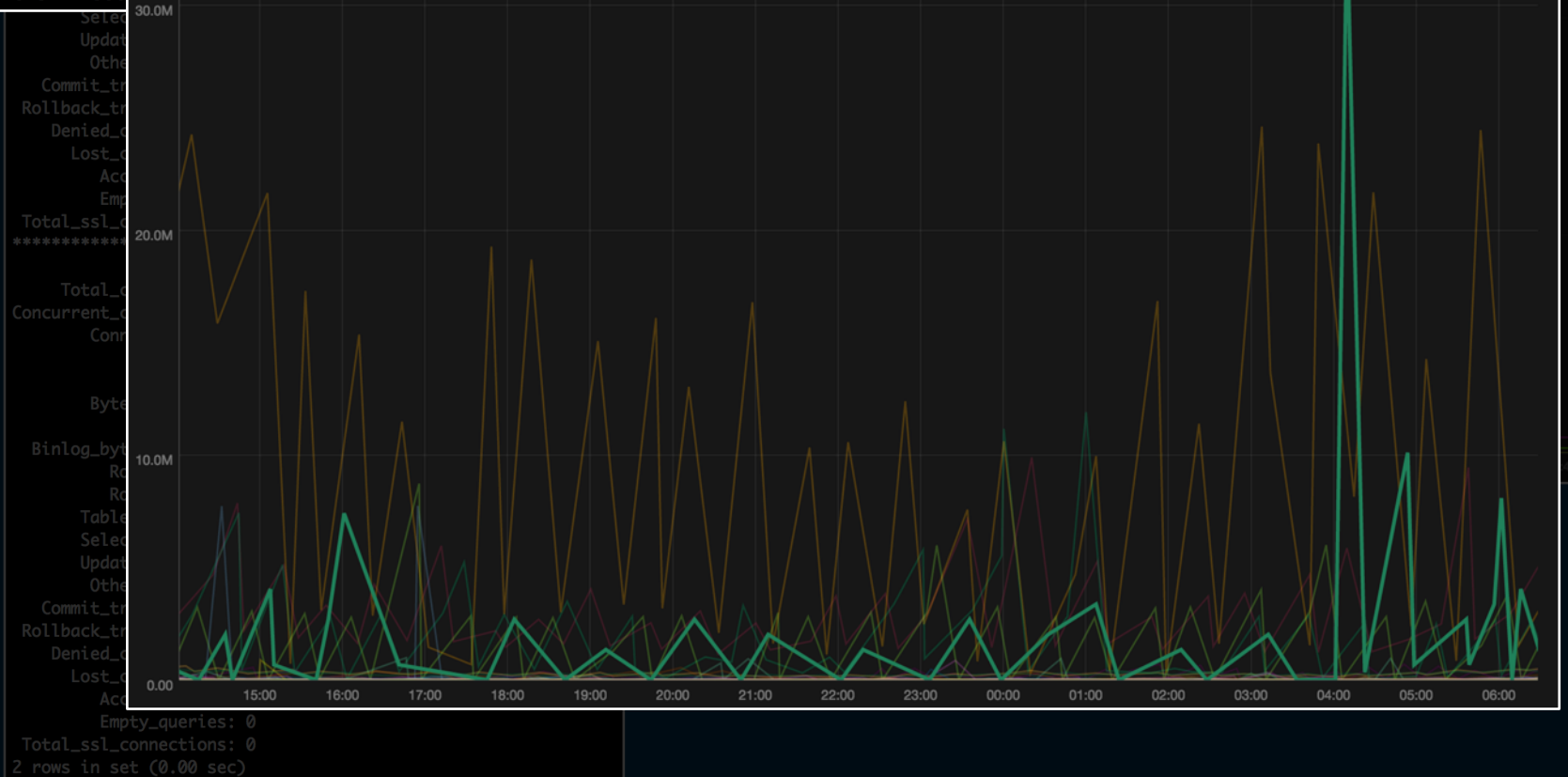
```
geoff@db.example.net [(none)]> show user_statistics\G
```

```
***** 1. row *****
User: geoff
Total_connections: 24
```

```
ganderson@ops-db.ve.box.net [(none)]> show table_statistics;
```

```
+-----+-----+-----+-----+-----+
| Table_schema | Table_name | Rows_read | Rows_changed | Rows_changed x #indexes |
```

```
+-----+-----+-----+-----+-----+
| sakila       | sakila     |           |              |                          |
+-----+-----+-----+-----+-----+
2 rows in set
```





Queries





Defense #5 – Box Rain gauge



Box Rain Gauge

Servers ▾

localhost.localdomain:3306

Sat, 11 Apr 2015 04:51:52 +0000

Previous time

Next time

2015_04_11_04_51_52

df	disk-space	diskstats	hostname	innodbstatus1
innodbstatus2	interrupts	log_error	lsnf	meminfo
mutex-status1	mutex-status2	mysqladmin	netstat	netstat_s
output	pmap	processlist	procstat	procvstat
ps	slabinfo	sysctl	top	trigger
variables	vmstat	vmstat-overall		

Search in files: lines of context regex pattern

Sift | Netstat Summary

```
===== localhost.localdomain at [34m2015_04_11_04_51_52 [31mDEFAULT[0m (1 of 1) =====
--diskstats--
#ts device   rd_s rd_avkb rd_mb_s rd_mrg rd_cnc  rd_rt   wr_s wr_avkb wr_mb_s wr_mrg wr_cnc  wr_rt busy in_prg   io_s qtime stime
{29} dm-0    122.4  44.8    5.4    0%    0.2    1.8    342.0  3.9    1.3    0%    3.0    8.7  18%    0    464.4  6.5  0.4
dm-0  0% . . . . . 5% 10% 40% 35% 5% . 10% 0% 10% 35% . 30% 35% 30% 0%
--vmstat--
 r b swpd free buff  cache si so  bi  bo  in  cs us sy id wa st
11 1  304 6216 8924 266120 0 0 100 385 188 193 3 3 94 0 0
 4 0 2680 7768 5412 205872 2 80 5314 1386 1561 2669 52 45 3 1 0
wa 0% . . . . . 5% 0% . . . . .
--innodb--
txns: 1xnot (0s)
0 queries inside InnoDB, 0 queries in queue
Main thread: waiting for server activity, pending reads 0, writes 0, flush 0
Log: lsn = 0, chkp = 0, chkp age = 0
Threads are waiting at:
```

Defense #5 – Box Raingauge



Box Rain Gauge

Servers

RainGauge, the new killer tool ?

By Cédric PEINTRE in Admin

09/09/2012 2 Comments

I'm sure you've heard of Box Anemometer from the Box (MySQL) team, an excellent UI tool based on *pt-query-digest*. Now the guys from the box team offer us another killer UI tool based on *pt-stalk*, and I'm sure you will really appreciate to use it !

What is it ?

"RainGauge" consists of three parts :

- A set of scripts that collect data about the health of your system and your databases (based on *pt-stalk*)
- A process that push these data on a centralised place
- A very simple interface to navigate in the collected data

But the collection begins only when specific conditions are triggered, and you choose what these conditions are. Find below the general flowsheet :



```
Main thread: waiting for server activity, pending reads 0, writes 0, flush 0
Log: lsn = 0, chkp = 0, chkp age = 0
Threads are waiting at:
```

Sat, 11 Apr 2015 04:51:52 +0000

Previous time

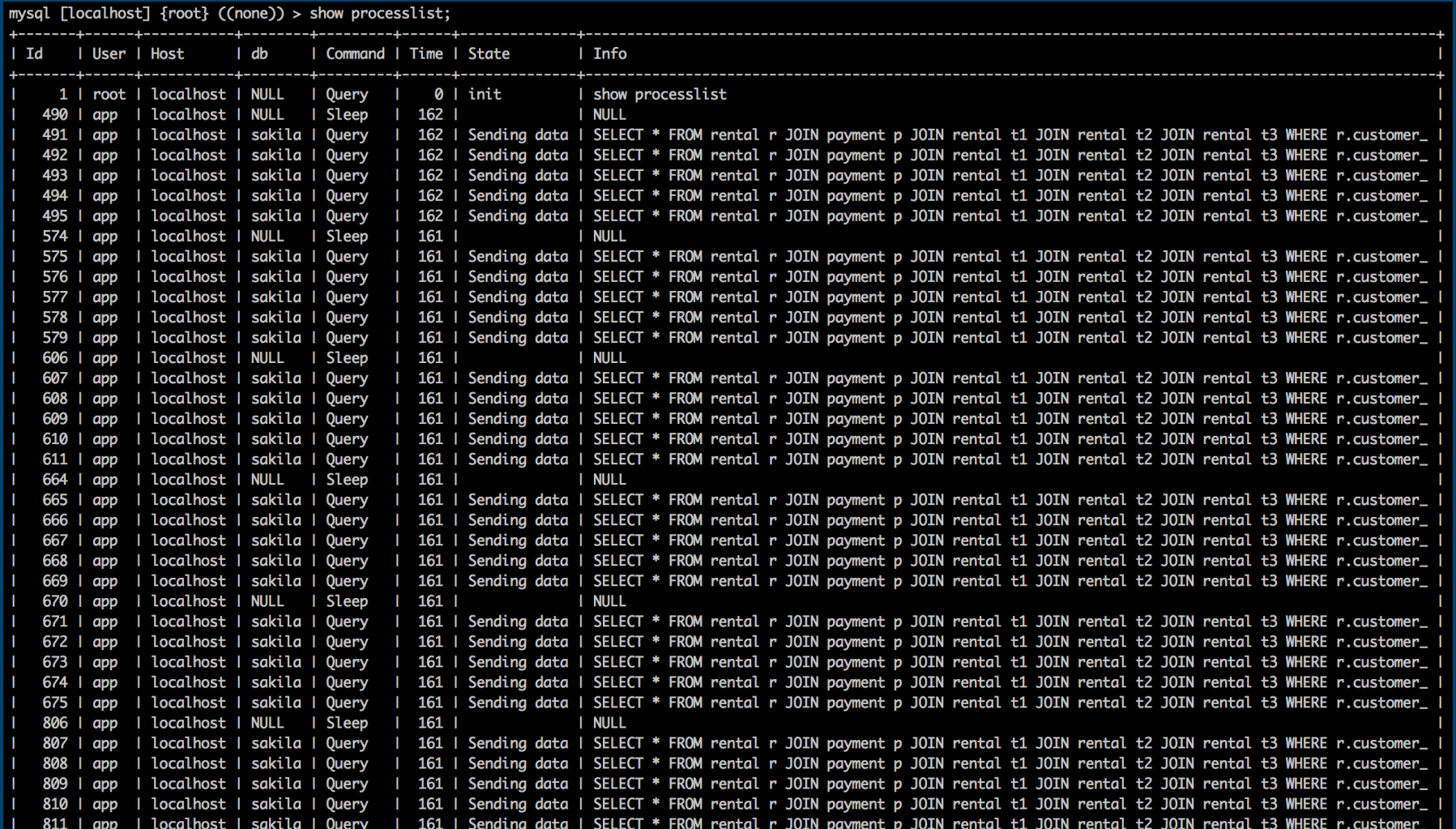
Next time



Anemometer & Raingauge

Gavin Towey
Senior Database Administrator @ Box
gavin@box.com







mysql-sys

```
mysql [localhost] {root} (> select * from statement_analysis LIMIT 1\G  
***** 1. row *****  
  
query: SELECT `intcol1`, `charcol1` FROM `t1`  
db: mysqlslap  
full_scan: *  
  
FROM rent      exec_count: 361628  
FROM rent      err_count: 52  
FROM rent      warn_count: 0  
FROM rent      total_latency: 51.21 m  
FROM rent      max_latency: 239.39 ms  
               avg_latency: 8.50 ms  
FROM rent      lock_latency: 1.52 m  
FROM rent      rows_sent: 134632249  
FROM rent      rows_sent_avg: 372  
FROM rent      rows_examined: 134641306  
FROM rent      rows_examined_avg: 372  
               rows_affected: 0  
FROM rent      rows_affected_avg: 0  
FROM rent      tmp_tables: 0  
FROM rent      tmp_disk_tables: 0  
FROM rent      rows_sorted: 0  
FROM rent      sort_merge_passes: 0  
  
digest: 239eeeb1e52b81b52e3020c23988c8cc  
first_seen: 2015-04-05 16:10:47  
last_seen: 2015-04-05 20:41:17  
1 row in set (0.01 sec)  
  
FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
  
FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
  
* FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
* FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
* FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
* FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_  
* FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_
```

Defense #6 – Monitor the queries



pt-query-digest

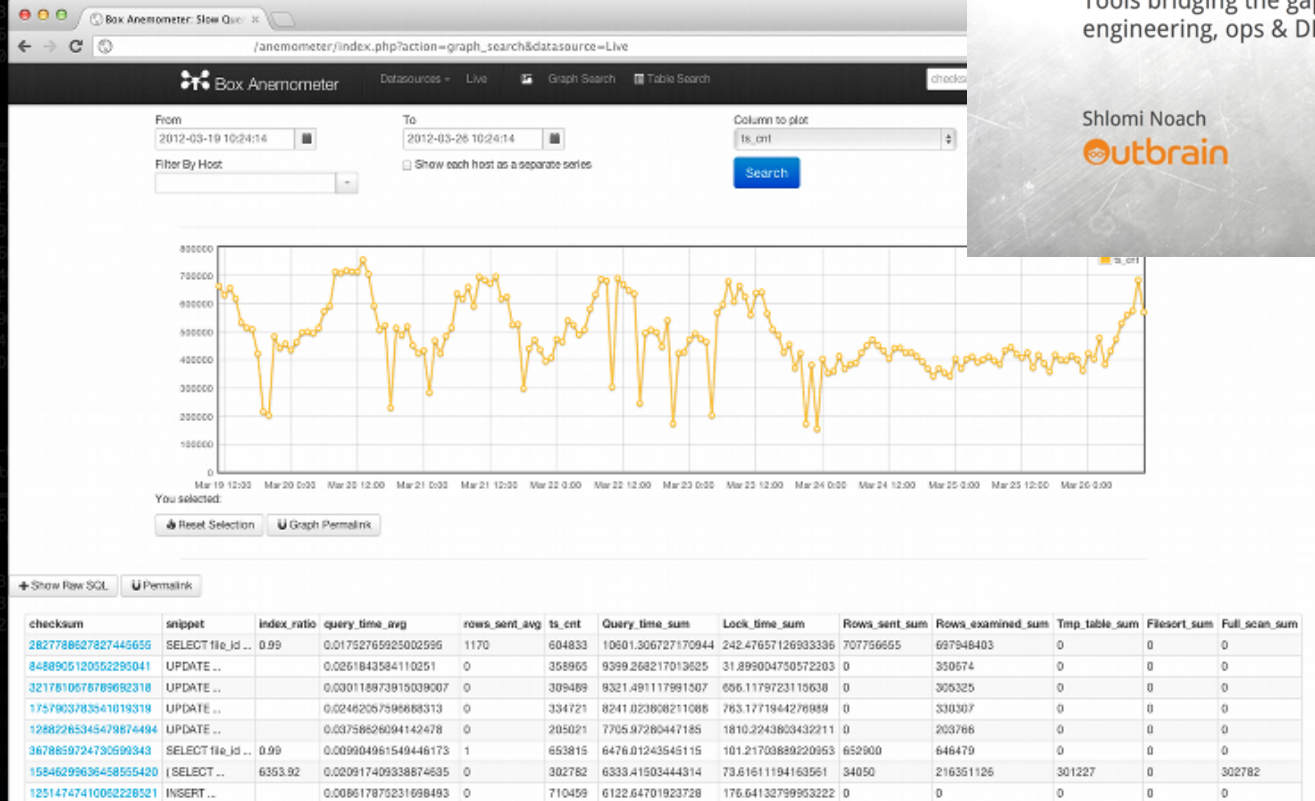
mysql-sys

Anemometer

MySQL DevOps @ Outbrain

Tools bridging the gap between MySQL engineering, ops & DBAs

Shlomi Noach





Queries





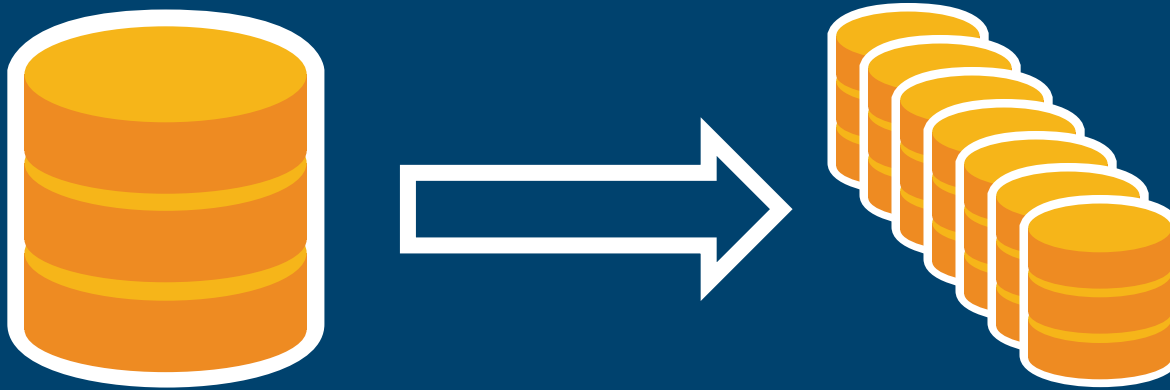
Queries



Time to
break up
the data



Defense #7 – Sharding



Defense #7 – Sharding



The Etsy Shard Architecture

Starts With **S** and

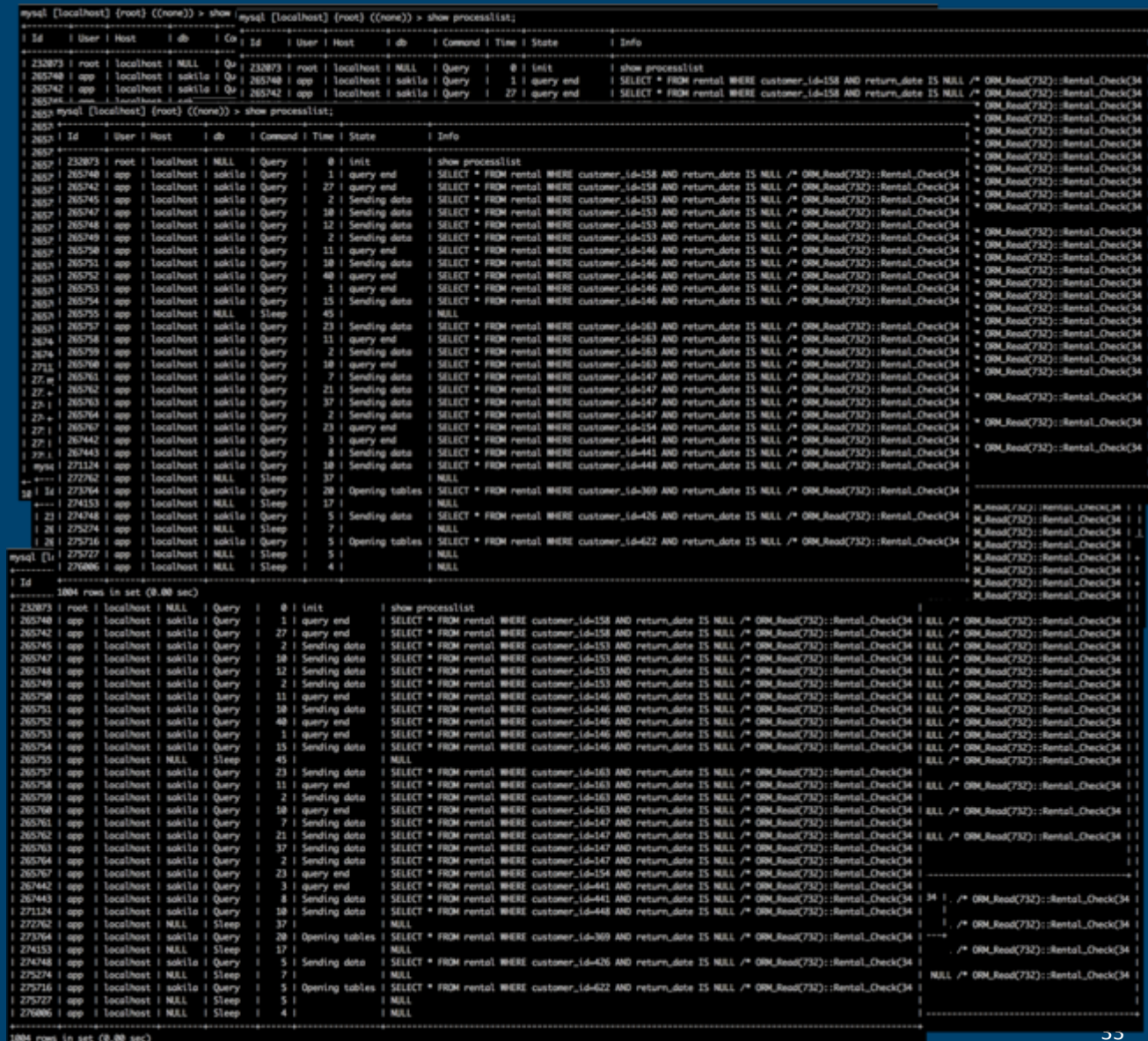
jgoulah@etsy.com

The Story of Sharding at Box

Tamar Bercovici

@TamarBercovici





[illegible]

Defense #8 – innotop



[R0] Dashboard (? for help) Servers: all_active

Conn	Uptime	QPS	QPS	MaxSQL	Run	Run	Curs	TbIs	Repl	ReplLog	SQL
db9000	515d	_____^_____			_____^_____	1	104	743	Yes	1	
db9001	518d	_____^_____			_____^_____		134	3071	Yes		
db9002	143d	_____^_____			_____^_____	2	1103	4096	Yes	7	
db9003	449d	_____^_____			_____^_____		126	2902	Yes		
db9004	31d	_____^_____			_____^_____	1	114	1982	Yes		
db9005	10d	_____^_____			_____^_____		570	3995	Yes	5	
db9006	10d	_____^_____			_____^_____		828	645	Yes		
db9007	28d	_____^_____		46s	_____^_____	1	554	4096	Yes	3	UPDATE
db9008	422d	_____^_____			_____^_____		628	4096	Yes	3	
db9009	254d	_____^_____			_____^_____	1	1330	1062	Yes	1	
db9010	15d	_____^_____			_____^_____		404	2486	Yes	5	
db9011	15d	_____^_____			_____^_____		568	452	Yes		
db9012	44d	_____^_____			_____^_____		534	4096	Yes	5	
db9013	44d	_____^_____			_____^_____	1	572	697	Yes		
db9014	149d	_____^_____		1s	_____^_____	1	542	4096	Yes	4	SELECT
db9015	149d	_____^_____			_____^_____		871	1672	Yes	1	
db9016	15d	_____^_____			_____^_____		512	3012	Yes	3	
db9017	15d	_____^_____			_____^_____		455	544	Yes		
db9018	10d	_____^_____			_____^_____		434	623	Yes		
db9019	36d	_____^_____			_____^_____		1101	2671	Yes	4	
db9020	32d	_____^_____			_____^_____		341	463	Yes		
db9021	9d	_____^_____		3s	_____^_____	5	519	1139	Yes	2	SELECT
db9022	253d	_____^_____		23s	_____^_____	12	684	4096	Yes	2	SELECT
db9023	220d	_____^_____			_____^_____	1	770	1240	Yes		
db9024	32d	_____^_____			_____^_____		568	3513	Yes	2	
db9025	520d	_____^_____			_____^_____		133	909	Yes		
db9026	149d	_____^_____			_____^_____		432	871	Yes		
db9027	21d	_____^_____			_____^_____	1	285	754	Yes		
db9028	10d	_____^_____			_____^_____	1	365	599	Yes	1	
db9029	149d	_____^_____			_____^_____		428	851	Yes		
db9030	14d	_____^_____			_____^_____		432	839	Yes		
db9031	149d	_____^_____			_____^_____	1	784	1540	Yes		
db9032	31d	_____^_____			_____^_____		635	1176	Yes		
db9033	149d	_____^_____			_____^_____		361	688	Yes		
db9034	149d	_____^_____			_____^_____		574	1086	Yes		
db9035	150d	_____^_____			_____^_____	1	496	1020	Yes		
db9036	414d	_____^_____			_____^_____		14	222	Yes	1	
db9037	413d	_____^_____			_____^_____		20	1325	Yes	2	
db9038	241d	_____^_____			_____^_____		15	1471	Yes	1	
db9039	127d	_____^_____			_____^_____		29	2428	Yes	2	
db9040	129d	_____^_____			_____^_____		26	4096	Yes	2	
db9041	127d	_____^_____		3m	_____^_____	5	19	4096	Yes	2	SELECT
db9042	504d	_____^_____			_____^_____		38	214	Yes	1	
db9043	145d	_____^_____			_____^_____		1442	2911	Yes	6	
db9044	149d	_____^_____			_____^_____		23	380	Yes		
db9045	143d	_____^_____			_____^_____		22	195	Off	1	
db9046	527d	_____^_____		2s	_____^_____	5	937	1492	Yes	1	SELECT
db9047	28d	_____^_____			_____^_____		766	2071	Yes		

Defense #8 – innotop



```
[general]
mode=A
[/general]
```

```
[connections]
mysql56Test=user= dsn=DBI:mysql;;host=127.0.0.1;port=3306;mysql_read_default_group=client
anotherDb=user= dsn=DBI:mysql;;host=127.0.0.1;port=3306;mysql_read_default_group=client
[/connections]
```

```
[server_groups]
all=mysql56Test anotherDb
[/server_groups]
```

Assign connections
to server groups

Define connections
(with custom names)

```
[active_server_groups]
A=all
[/active_server_groups]
```

Set the default
active group

Define views

```
[active_columns]
health_dashboard=cxn uptime spark_qps qps max_query_time spark_run run connections open slave_running
time_behind_master longest_sql
[/active_columns]
```

```
[visible_tables]
A=health_dashboard
[/visible_tables]
```

Set the default view

Setup color
highlighting

```
[colors]
health_dashboard=col='slave_running' op='eq' arg='No' color='black on_red'
health_dashboard=col='time_behind_master' op='>' arg='30' color='cyan'
health_dashboard=col='max_query_time' op='>' arg='60' color='red'
health_dashboard=col='max_query_time' op='>' arg='30' color='yellow'
[/colors]
```

Defense #8 – innotop



[RO] Dashboard (? for help)Servers: all_active

CNN	Uptime	QPS	QPS	MaxSQL	Run	Run	Cxns	TbIs	Repl	ReplLag	SQL
db9000	515d					1	104	743	Yes	1	
db9001	518d						134	3071	Yes		
db9002	143d					2	1103	4096	Yes	7	
db9003	449d						126	2902	Yes		
db9004	31d					1	114	1982	Yes		
db9005	10d						570	3995	Yes	5	
db9006	10d						828	645	Yes		
db9007	28d			46s		1	554	4096	Yes	3	UPDATE
db9008	422d						628	4096	Yes	3	
db9009	254d					1	1330	1062	Yes	1	
db9010	15d						404	2486	Yes	5	
db9011	15d						568	452	Yes		

PERCONA MYSQL PERFORMANCE BLOG

PERCONA.COM / COMMUNITY / PERFORMANCE BLOG / INSIGHT FOR DBAS / INNOTOP: A REAL-TIME, ADVANCED INVESTIGATION TOOL FOR MYSQL

Innotop: A real-time, advanced investigation tool for MySQL

October 14, 2013 | by *Przemysław Malkowski* | 4 Comments

f Like1

g+18

Twitter Tweet1

in Share2

db9035	150d					1	496	1020	Yes		
db9036	414d						14	222	Yes	1	
db9037	413d						20	1325	Yes	2	
db9038	241d						15	1471	Yes	1	
db9039	127d						29	2428	Yes	2	
db9040	129d						26	4096	Yes	2	
db9041	127d			3m		5	19	4096	Yes	2	SELECT
db9042	504d						38	214	Yes	1	
db9043	145d						1442	2911	Yes	6	
db9044	149d						23	380	Yes		
db9045	143d						22	195	Off	1	
db9046	527d			2s		5	937	1492	Yes	1	SELECT
db9047	28d						766	2071	Yes		



**Now for the
fun stuff**

[illegible]



I woke up at 3am?
FOR THIS!?



Who added that crappy query...?

This should take care of itself... >:[:



Defense #9 – Query killer

```
$ pt-kill --interval          10 \
        --match-user         '^app_rw.*' \
        --busy-time          120s \
        --ignore-command     '(Prepare|Execute)' \
        --match-info          '^\\s*(?\\s*(SELECT|select))' \
        --ignore-info         '(. * SQL_NO_CACHE . * |.*&plz_no_kill=1.*)' \
        --idle-time           28800 \
        --victims              all \
        --print                \
        --kill                  \
        --log                  /var/log/qkill.log \
        --execute-command     '(mail -s "Killed queries on ${HOSTNAME}" \
        "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkill.log:
$(tail -10 /var/log/qkill.log)
EOF
)' \
        --daemonize            \
        --pid                  /var/run/qkill.pid
```

Defense #9 – Query killer



```
$ pt-kill --interval 10
--match-user '^app rw.*'
--busy-time 120s
--ignore-command '(Prepare|Execute)'
--match-info '^\\s*\\(\\?\\s*(SELECT|select)\\)'
--ignore-command 'SQL_NO_CACHE .*|.*&plz_no_kill=1.*)'
--idle-time 0
--verbose
--print
--kill
--log /var/log/qkill.log
--execute-command 'echo "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkill.log:
$(tail -10 /var/log/qkill.log)
EOF
)'
--daemonize
--pid /var/run/qkill.pid
```

Kill queries over 2min!

E-mail the last killed query!

Only kill SELECT statements

Idle for 8 hours?

Don't kill queries matching this...

Defense #9 – Query killer



```
$ pt-kill --interval 10
--match-user '^app_rw.*'
--busy-time 120s
--ignore-command '(Pre
--match-info '^\\s*
--ignore-info '(* SQL NO _CHE .*|.*&plz_no_kill=1.*)' \
```

Some sharp edges...



Search

Login / Register

[Developer Zone](#) [Bugs Home](#) [Report a bug](#) [Statistics](#) [Advanced search](#) [Saved searches](#) [Tags](#)

Bug #68051

Killing a query inside InnoDB causes it to eventually crash with an assertion

Submitted: 8 Jan 2013 19:22

Modified: 11 Mar 2013 19:21

Reporter: [Davi Arnaut](#) (OCA)

Email Updates:

Status: Closed

Impact on me:

Category: MySQL Server: InnoDB storage engine

Severity: S1 (Critical)

Version: 5.5.29

OS: Any

Assigned to:

Target Version:

Tags: [assertion failure](#), [btr0pcur.c](#), [crash](#), [KILL QUERY](#), [regression](#)

```
I
$
EOF
)'
```

```
--daemonize
--pid /var/run/qkill.pid
```

Long transactions happen...





Defense #10 – Transaction Monitor (and killer)

```
my $query = "SELECT *, unix_timestamp()-unix_timestamp(trx_started) as len FROM INFORMATION_SCHEMA.INNODB_TRX
ORDER BY trx_started LIMIT 1";
print $query . "\n";
my $sth = $dbh->prepare($query);
$sth->execute() or die $dbh->errstr;
my $data1 = get_query_id($sth);
my $query_id = $data1->{trx_mysql_thread_id};
```

```
$query = "SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST WHERE id=$query_id";
$sth = $dbh->prepare($query);
$sth->execute() or die $dbh->errstr;

my $data2 = get_source($sth);
my $source = $data2->{HOST};
my ($host,$port) = split(/:/, $source);
```

```
print "Found $host $port; attaching tcpdump\n";
my $pid = handle_tcpdump($host, $port);
wait_for_queries($dbh, $query_id);
kill_tcpdump($pid);
print "\n";
if ($opt{'kill-trx'}) {
    kill_query($dbh,$query_id);
}
exit;
```

Find the connection for
the longest transaction

Get the source host
for that connection

Try to catch a query
from the connection
before killing it

Defense #10 – Transaction Monitor (and killer)



```
my $query = "SELECT *, unix_timestamp()-unix_timestamp(trx_started) as len FROM INFORMATION_SCHEMA.INNODB_TRX
ORDER BY len DESC";
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
849	root	localhost	NULL	Query	0	init	show processlist
3179	app	localhost:56828	NULL	Sleep	7		NULL
3195	app	localhost:56844	NULL	Sleep	7		NULL
3206	app	localhost:56855	NULL	Sleep	7		NULL
3212	app	localhost:56861	NULL	Sleep	7		NULL

```
my $sth = $dbh->prepare($query);
$sth->execute() or die $dbh->errstr;

my $data2 = get_source($sth);
my $source = $data2->{HOST};
my ($host,$port) = split(/:/, $source);

print "Found $host $port: attaching tcpdump\n";
my $pid = handle_tcpdump($host, $port);
wait_for_queries($dbh, $query_id);
kill_tcpdump($pid);
print "\n";
if ($opt{'kill-trx'}) {
    kill_query($dbh,$query_id);
}
exit;
```

```
sub handle_tcpdump(@, @) {
    my ($host,$port) = @_;

    my $pid = fork();
    if (!$pid) {
        my $cmd = "sudo tcpdump -i vlan64 -nn -s0 -q -w - 'src host $host
and dst port 3306 and src port $port' | strings";
        print $cmd."\n";
        exec($cmd);
    } else {
        return $pid;
    }
}
```

Too many connections!



```
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
232073	root	localhost	NULL	Query	0	init	show processlist
265740	app	localhost	sakila	Query	1	query end	SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265742	app	localhost	sakila	Query	27	query end	SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265745	app	localhost	sakila	Query	2	Sending data	SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265747	app	localhost	sakila	Query	10	Sending data	SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265748	app	localhost	sakila	Query	12	Sending data	SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265749	app	localhost	sakila	Query	2	Sending data	SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265750	app	localhost	sakila	Query	11	query end	SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265751	app	localhost	sakila	Query	10	Sending data	SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265752	app	localhost	sakila	Query	40	query end	SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265753	app	localhost	sakila	Query	1	query end	SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265754	app	localhost	sakila	Query	15	Sending data	SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265755	app	localhost	NULL	Sleep	45		NULL
265757	app	localhost	sakila	Query	23	Sending data	SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265758	app	localhost	sakila	Query	11	query end	SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265759	app	localhost	sakila	Query	2	Sending data	SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265760	app	localhost	sakila	Query	10	query end	SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265761	app	localhost	sakila	Query	7	Sending data	SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265762	app	localhost	sakila	Query	21	Sending data	SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265763	app	localhost	sakila	Query	37	Sending data	SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265764	app	localhost	sakila	Query	2	Sending data	SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265767	app	localhost	sakila	Query	23	query end	SELECT * FROM rental WHERE customer_id=154 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
267442	app	localhost	sakila	Query	3	query end	SELECT * FROM rental WHERE customer_id=441 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
267443	app	localhost	sakila	Query	8	Sending data	SELECT * FROM rental WHERE customer_id=441 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
271124	app	localhost	sakila	Query	10	Sending data	SELECT * FROM rental WHERE customer_id=448 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
272762	app	localhost	NULL	Sleep	37		NULL
273764	app	localhost	sakila	Query	20	Opening tables	SELECT * FROM rental WHERE customer_id=369 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
274153	app	localhost	NULL	Sleep	17		NULL
274748	app	localhost	sakila	Query	5	Sending data	SELECT * FROM rental WHERE customer_id=426 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
275274	app	localhost	NULL	Sleep	7		NULL
275716	app	localhost	sakila	Query	5	Opening tables	SELECT * FROM rental WHERE customer_id=622 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
275727	app	localhost	NULL	Sleep	5		NULL
276006	app	localhost	NULL	Sleep	4		NULL

1004 rows in set (0.00 sec)



Too many connections!



```
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
232873	root	localhost	NULL	Query	0	init	show processlist
265740	app	localhost	sakila	Query	1	query end	SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265742	app	localhost	sakila	Query	27	query end	SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
265745	app	localhost					
265747	app	localhost					
265748	app	localhost					
265749	app	localhost					
265750	app	localhost					
265751	app	localhost					
265752	app	localhost					
265753	app	localhost					
265754	app	localhost					
265755	app	localhost					
265757	app	localhost					
265758	app	localhost					
265759	app	localhost					
265760	app	localhost					
265761	app	localhost					
265762	app	localhost					
265763	app	localhost					
265764	app	localhost					
265767	app	localhost					
267442	app	localhost					
267443	app	localhost					
271124	app	localhost					
272762	app	localhost					
273764	app	localhost					
274153	app	localhost					
274748	app	localhost					
275274	app	localhost					
275716	app	localhost	sakila	Query	5	Opening tables	SELECT * FROM rental WHERE customer_id=622 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34
275727	app	localhost	NULL	Sleep	5		NULL
276806	app	localhost	NULL	Sleep	4		NULL

```
mysql [localhost] {root} ((none)) > show global status like 'threads_connected';
```

Variable_name	Value
Threads_connected	5000

```
mysql [localhost] {root} ((none)) > select @@max_connections, @@max_user_connections;
```

@@max_connections	@@max_user_connections
5000	5000

```
1004 rows in set (0.00 sec)
```





Defense #11 – Dynamic Query Killer Thresholds

```
$ pt-kill --interval          10 \
        --match-user          '^app_rw.*' \
        --busy-time           120s \
        --ignore-command      '(Prepare|Execute)' \
        --match-info           '^\\s*\\(?:\\s*(SELECT|select))' \
        --ignore-info          '(.* SQL_NO_CACHE .*|.*&plz_no_kill=1.*)' \
        --idle-time            28800 \
        --victims              all \
        --print                \
        --kill                 \
        --log                   /var/log/qkill.log \
        --execute-command      '(mail -s "Killed queries on ${HOSTNAME}" \
        "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkill.log:
$(tail -10 /var/log/qkill.log)
EOF
)' \
        --daemonize            \
        --pid                   /var/run/qkill.pid \
        --dynamic-time         60 \
        --min-busy-time         10 \
        --min-idle-time         30 \
```

Defense #11 – Dynamic Query Killer Thresholds



JAN

31

Enhancing pt-kill to Better Protect your Servers

I believe in automation as much as possible, and I'm always working to make the day to day tasks of operations as smooth as possible. Also I try not to be afraid to take good tools and make them better.

Here in Database Ops at Box, we use pt-kill running as a service to constantly monitor our servers and help protect against long running queries. But our thresholds are pretty generous, and in some cases it's possible for unforeseen circumstances to cause enough queries to storm the database such that we can have problems before any of them hit the threshold for "busy time." Ditto for idle connections.

The response is that someone has to be available to manually run another copy of pt-kill with much lower thresholds to clear out these thundering herds. But what if we could let pt-kill handle both the "normal" mode and still protect us from herds?

That's what we've done by adding a couple new variables and small code changes. Here's an example:

```
pt-kill --busy-time=60 --dynamic-time 50
```

The extra option `--dynamic-time` causes pt-kill to monitor the ratio of `(max_connections-threads_connected)/max_connections`. This gives us a percentage of connections available, and if that percentage drops **below** the argument to `--dynamic-time`, then pt-kill will modify the value of busy-time so it can kill queries more aggressively.

For example, if `max_connections` was 1000 and `threads_connected` was 500, then that would trigger the new behavior.

```
connections_free = 0.5
new_busy_time = busy_time * connections_free
```

The new busy time is the percent of connections free times the original value. In this case it would reduce the value by half and our new busy time is 30.

Similarly, if there were 900/1000 threads connected then our busy-time would go all the way down to 6 seconds.

You may want to set a floor for your busy-time so it doesn't drop below some minimum value. Maybe we want to never go below 10 seconds. There's an option for that:

```
pt-kill --busy-time=60 --dynamic-time 50 --min-busy-time=10
```

And there are similar options for idle-time. Thats it! I've provided a patch for percona toolkit 2.1.1. When I get a chance I'll try to update it for the current version (or someone feel free to do that and submit one back to me!)

Here's a link to the patch: https://github.com/box/mysqlutilities/blob/master/patches/pt-kill_dynamic-time_2.1.1.patch

Defense #11 – Dynamic Query Killer Thresholds



 **box / mysqlutilities**

Unwatch 19 Unstar 32 Fork 8

branch: master mysqlutilities / patches / +

Added link to the blog article on pt-kill

 **asheepapart** authored on Feb 4, 2014 latest commit 972c8df08b

..

 README.md	Added link to the blog article on pt-kill	a year ago
 pt-kill_dynamic-time_2.1.1.patch	added pt-kill dynamic time 2.1.1 patch	a year ago

 **README.md**

Patches

A collection of patches to improve existing tools.

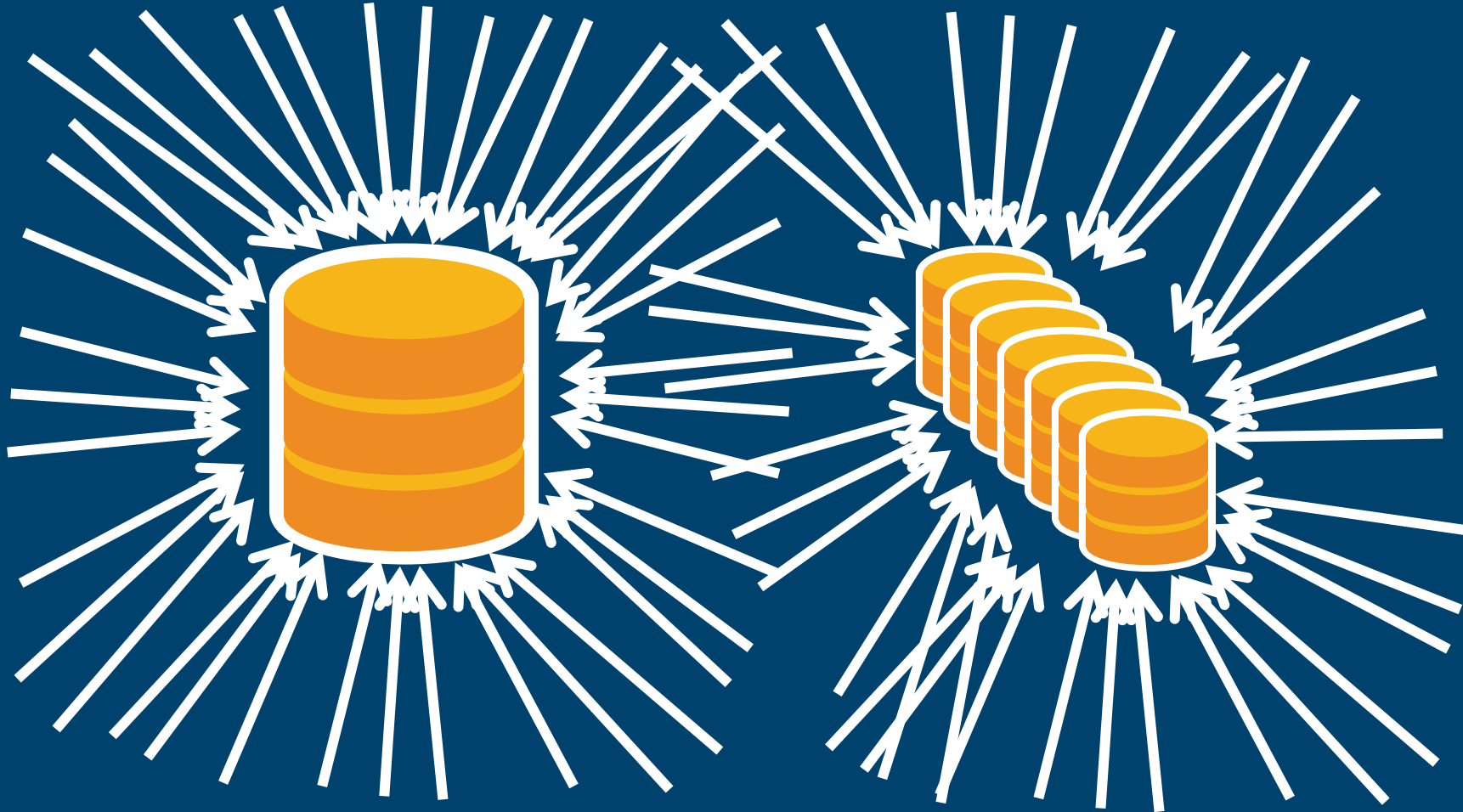
Percona

A quick improvement to Percona's `pt-kill` command. It adds The extra option `--dynamic-time` causes `pt-kill` to monitor the ratio of `(max_connections-threads_connected)/ max_connections` . This gives us a percentage of connections available, and if that percentage drops below the argument to `--dynamic-time` , then `pt-kill` will modify the value of busy-time so it can kill queries more aggressively.

The thundering herd..



Connections



The thundering herd..



Connections



The thundering herd..



Connections



Defense #12 – Healthchecking



Connections

Are the DBs up?

```
{
  "names": {
    "global": {
      "node_age": 106,
      "state": "ON"
    },
    "shard1": {
      "node_age": 276,
      "state": "ON"
    },
  },
  "max_allowed_age": 6000
}
```



SELECT 1



'1', duh...

Defense #12 – Healthchecking



Connections

```
{
  "names": {
    "global": {
      "node_age": 106,
      "state": "ON"
    },
    "shard1": {
      "node_age": 276,
      "state": "ON"
    },
  },
  "max_allowed_age": 6000
}
```



Defense #12 – Healthchecking



Connections

Gimme Data!

SELECT 1

'1', duh...

```
{
  "names": {
    "global": {
      "node_age": 106,
      "state": "ON"
    },
    "shard1": {
      "node_age": 276,
      "state": "ON"
    },
  },
  "max_allowed_age": 6000
}
```



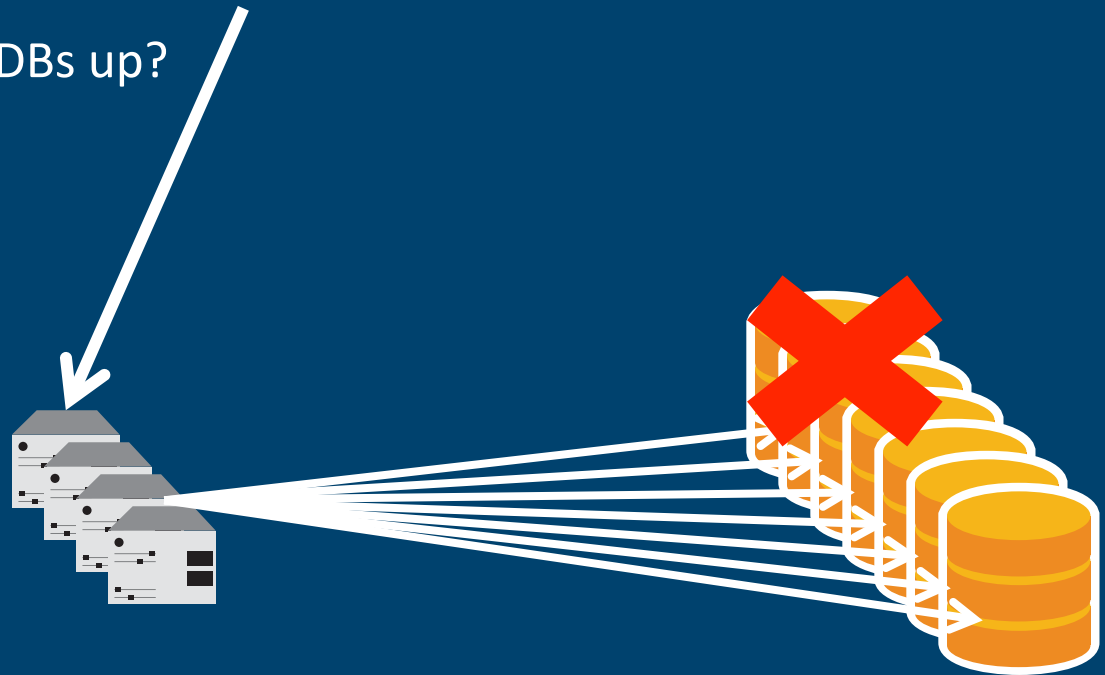
Defense #12 – Healthchecking



Connections

Are the DBs up?

```
{
  "names": {
    "global": {
      "node_age": 106,
      "state": "OFF"
    },
    "shard1": {
      "node_age": 276,
      "state": "ON"
    },
  },
  "max_allowed_age": 6000
}
```



Defense #12 – Healthchecking



Connections

```
{
  "names": {
    "global": {
      "node_age": 106,
      "state": "OFF"
    },
    "shard1": {
      "node_age": 276,
      "state": "ON"
    },
  },
  "max_allowed_age": 6000
}
```



[MySQL Fabric Resource Kit »](#)

- The `mysqldfabric` process which processes any management requests. When using the HA feature, this process can also be made responsible for monitoring the master server and initiating failover to promote a slave to be the new master should it fail.
- MySQL Fabric-aware connectors store a cache of the routing information that it has fetched from MySQL Fabric and then uses that information to send transactions or queries to the correct MySQL Server.





Connections

Gimme Data!

SELECT 1

- Cache hit?
- Rewrite Query?
- Send to RO replica?

'1', duh...

Defense #13 – Data Layer as a Service



Connections

- Cache hit?
- Rewrite Query?
- Send to RO replica?



Get Data **LIMIT 1**



Defense #13 – Data Layer as a Service



Connections

Here ya go..

Data

- Cache hit?
- Rewrite Query?
- Send to RO replica?



Defense #13 – Data Layer as a Service



Vitess

A set of servers and tools meant to facilitate scaling of MySQL databases for the web.

[Start Using Vitess](#) or [View on GitHub](#)



Shard-Query

MPP Query engine for MySQL

MariaDB MaxScale

Scalability, High Availability and Security with MariaDB MaxScale

MariaDB MaxScale is an open-source, database-centric proxy that works with MariaDB Enterprise, MariaDB Enterprise Cluster, MariaDB 5.5, MariaDB 10 and Oracle MySQL®. It's pluggable architecture is designed to increase flexibility and aid customization. Built upon a lightweight, high-speed networking core designed to facilitate throughput. MariaDB MaxScale runs between the client application and the database cluster offering connection and statement-based load balancing. MariaDB MaxScale allows scaling of an organization's database infrastructure while keeping the needs of DBAs, Developers and Data Architects in mind.

Scalability

MariaDB MaxScale helps scale your database infrastructure with no application-level code changes. MaxScale provides connection-based and statement-based load balancing for MariaDB Galera Cluster, MariaDB Master-Slave, and Oracle MySQL.

renecannao / proxysql

High Performance Proxy for MySQL

164 commits 1 branch 0 releases 2 contributors

Locks happen...



Defense #14 – Lock Monitor



```
SELECT *,
    UNIX_TIMESTAMP() - UNIX_TIMESTAMP(t.TRX_STARTED) as age
FROM INFORMATION_SCHEMA.INNODB_LOCKS l
JOIN INFORMATION_SCHEMA.INNODB_TRX t
ON t.trx_id=l.lock_trx_id
ORDER BY age DESC
```

trx_mysql_thread_id	trx_state	lock_table	...	age
419	RUNNING	`sakila`.`rental`	...	544
581	LOCK WAIT	`sakila`.`rental`	...	49
631	LOCK WAIT	`sakila`.`rental`	...	49
573	LOCK WAIT	`sakila`.`rental`	...	49
...				

500 rows in set (0.00 sec)

Defense #14 – Lock Monitor (and killer)



```
KILL 419 /* WITH FIRE */;
```

```
+-----+-----+-----+-----+
| trx_mysql_thread_id | trx_state | lock_table          | ... | age |
+-----+-----+-----+-----+
| 419 | RUNNING | `sakila`.`rental`   | ... | 544 |
+-----+-----+-----+-----+
| 581 | LOCK WAIT | `sakila`.`rental`   | ... | 49 |
| 631 | LOCK WAIT | `sakila`.`rental`   | ... | 49 |
| 573 | LOCK WAIT | `sakila`.`rental`   | ... | 49 |
+-----+-----+-----+-----+
...
500 rows in set (0.00 sec)
```

Idle connections...



```
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
1	root	localhost	NULL	Query	0	init	show processlist
7231	app	localhost:63146	NULL	Sleep	4		NULL
7242	app	localhost:63157	NULL	Sleep	28800		NULL
7450	app	localhost:63367	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=9 AN
7451	app	localhost:63366	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=9 AN
7466	app	localhost:63382	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=11 A
7468	app	localhost:63384	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=11 A
7469	app	localhost:63385	sakila	Query	0	Writing to net	SELECT * FROM rental WHERE customer_id=11 A
7470	app	localhost:63387	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=5 AN
7471	app	localhost:63386	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=5 AN
7472	app	localhost:63388	sakila	Query	0	Sending data	SELECT * FROM rental WHERE customer_id=5 AN
7473	app	localhost:63389	sakila	Sleep	0		NULL

Idle connections...



```
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command	Time	State	Info
1	root	localhost	NULL	Query	0	init	show processlist
7231	app	localhost:63146	NULL	Sleep	4		NULL
7242	app	localhost:63157	NULL	Sleep	28800		NULL
7450	app	localhost:63367	sakila	Sleep	28800		NULL
7451	app	localhost:63366	sakila	Sleep	28800		NULL
7466	app	localhost:63382	sakila	Sleep	28800		NULL
7468	app	localhost:63384	sakila	Sleep	28800		NULL
7469	app	localhost:63385	sakila	Sleep	28800		NULL
7470	app	localhost:63387	sakila	Sleep	28800		NULL
7471	app	localhost:63386	sakila	Sleep	28800		NULL
7472	app	localhost:63388	sakila	Sleep	28800		NULL
7473	app	localhost:63389	sakila	Sleep	28800		NULL

Idle connections...



```
mysql [localhost] {root} ((none)) > show processlist;
```

Id	User	Host	db	Command
----	------	------	----	---------



7472	app	localhost:63388	sakila	Sleep
7473	app	localhost:63389	sakila	Sleep

How to find an errant MySQL client

A common story: You've got some connection, either it's busy running something it shouldn't be, it's in Sleep but holding some important lock, or you just don't know why it's connected to your database server in the first place. You see it in your SHOW PROCESSLIST like so:

```
mysql> show processlist G
***** 1. row *****
Id: 5979887
User: root
Host: localhost:55997
db: NULL
Command: Sleep
Time: 475
State:
Info: NULL
```

How do you find that client, especially if it's on another host? MySQL is providing you all the information you need above: localhost:55997. Of course localhost is the host or IP address, and 55997 is the source port of the socket; the port number (usually randomly assigned) on the far end of the socket, from the MySQL server's perspective. You can turn that number into something useful—the PID and user—, by running the following command on the host that made the connection:

```
# lsof -nPi :55997
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
mysqld 5026 mysql 1654u IPv4 303329996 TCP 127.0.0.1:3306->127.0.0.1:559
97 (ESTABLISHED)
mysql 9146 jcole 3u IPv4 303329995 TCP 127.0.0.1:55997->127.0.0.1:33
06 (ESTABLISHED)
```

(Note that here you can see both sides of the socket, since I'm running these commands on localhost. Disregard the first entry, as it's the half of the connection owned by the MySQL server.)

You can then find out the full command-line of the process with ps:

```
# ps -fp 9146
UID PID PPID C STIME TTY TIME CMD
jcole 9146 8740 0 12:53 pts/3 00:00:00 mysql -h 127.0.0.1 -u root -p
```

And see what it's doing with strace (waiting on a read of stdin in this dumb test):

```
# strace -p 9146
Process 9146 attached - interrupt to quit
read(0, <unfinished ...>
Process 9146 detached
```

I hope that's helpful! It's a pretty common debugging trick for me.



Defense #15 – strace errant connections

```
$ db-strace-longest-trx mydb.example.com
```

```
Longest transaction on mydb.example.com:3306
```

```
MySQL thread id is 15, client is 10.0.0.5:51540, running for 08:00:07
```

```
Warning: Permanently added '10.0.0.5' (RSA) to the list of known hosts.
```

```
Process 19820 attached - interrupt to quit
```

```
write(3, "...SELECT * FROM demo_test /* ORM_READ(732):...", 28) = 28
```

```
read(3, "\1\0\0\1\0050\0\0\2\3def\4test\tdemo_test\t...", 16384) = 309
```

```
pipe([4, 5]) = 0
```

```
[pid 19820] write(5, "+-----+-----+-----+-----+-----+-----+\n| c1
```

```
| c2          | c3    | c4    | c5    |\n+-----+-----+-----+-----+-----+-----+--
```

```
----+-----+\n| AA | HELLO, HELLO | 8 | 0 | 0 |\n+-----+-----
```

```
-----+-----+-----+-----+\n", 215 <unfinished ...>
```

```
pid [19820] close(5) = 0
```



Even idler connections!

```
$ db-strace-longest-trx mydb.example.com
```

```
Longest transaction on mydb.example.com:3306
```

```
MySQL thread id is 19, client is 10.0.0.5:51740, running for 08:00:12
```

```
Warning: Permanently added '10.0.0.5' (RSA) to the list of known hosts.
```

```
Process 1833 attached - interrupt to quit
```

Where's the thing!?

Defense #16 – pt-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

Usage

```
pt-pmp [OPTIONS] [FILES]
```

pt-pmp is a poor man's profiler, inspired by <http://poormansprofiler.org>. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

```
gdb -ex "set pagination 0" \
    -ex "bt" \
    -ex "source /path/to/php-src/.gdbinit" \
    -ex "dump_bt executor_globals.current_execute_data" \
    -batch \
    -p "$pid" \
    >> "$output_file"
```

Defense #16 – ~~pt-pmp~~ php-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

Usage

```
pt-pmp [OPTIONS] [FILES]
```

pt-pmp is a poor man's profiler, inspired by <http://poormansprofiler.org>. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

```
gdb -ex "set pagination 0" \
    -ex "bt" \
    -ex "source /path/to/php-src/.gdbinit" \
    -ex "dump_bt executor_globals.current_execute_data" \
    -batch \
    -p "$pid" \
    >> "$output_file"
```

Defense #16 – ~~pt-pmp~~ php-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

Usage

pt-pmp [OPTIONS] PROCESSES

pt-pmp is a poor man's profiler, inspired by <http://poormansprofiler.org>. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

But this
is still
too manual....

```
gdb -ex "set pagination 0" \
    -ex "bt" \
    -ex "source /path/to/php-src/.gdbinit" \
    -ex "dump_bt executor_globals.current_execute_data" \
    -batch \
    -p "$pid" \
    >> "$output_file"
```



Defense #16.1 – Autorun php-pmp

```
$ pt-kill --interval 10 \
--match-user '^app_rw.*' \
--busy-time 120s \
--ignore-command '(Prepare|Execute)' \
--match-info '^\\s*(?\\s*(SELECT|select))' \
--ignore-info '(. * SQL_NO_CACHE .*|.*&plz_no_kill=1.*)' \
--idle \
--victim \
--print \
--kill \
--log /var/log/qkill.log \
--execute-command '(mail -s "Killed queries on ${HOSTNAME}" \
"database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkill.log:
$(tail -10 /var/log/qkill.log)
EOF
) ' \
--daemonize \
--pid /var/run/qkill.pid
```

Change this to...

Defense #16.1 – Autorun php-pmp



```
#!/usr/bin/env bash
SUDOUSER="geoff"
QK_FILE="/var/log/qkill.log"
date="$(date +%Y_%m_%d_%H_%M_%S)"
TMP_FILE="/tmp/${date}-qkill-trace"
trace_file="/tmp/${HOSTNAME}-${date}.tar.gz"
trace_url="http://raingauge.example.com"
PORT='3306'

cleanup() {
    rm -f "$TMP_FILE" "$trace_file"
}

query="$(tac "${QK_FILE}" | sed -n '0,/^\#/ p' | tac)"
mysql_client="$(tac "${QK_FILE}" | sed -n '0,/^\#/ p' | tail -n1 | awk '{ print $8 }' | cut -d'"' -f4)"
echo "mysql client host: ${mysql_client}" | tee "$TMP_FILE"
if [[ -z "${mysql_client}" ]]
then
    echo "Can't extract client host and port from pt-kill log ${mysql_client}"
    cleanup
    exit 1;
fi
IFS=: read -ra mysql_host <<< "${mysql_client}"

pid="$(sudo -u "${SUDOUSER}" ssh -o StrictHostKeyChecking=no "${mysql_host[0]}" 'sudo /usr/sbin/lsof | grep "${mysql_host[1]}"' | awk '{ print $2 }')'"
echo "remote pid: ${pid}" | tee -a "$TMP_FILE"
echo "${query}" | tee -a "$TMP_FILE"
if [[ -z "${pid}" ]]
then
    echo "Can't find remote pid: ${mysql_host[0]}, ${mysql_host[1]}"
    cleanup
    exit 1
fi

sudo -u "${SUDOUSER}" ssh -n -o StrictHostKeyChecking=no ${mysql_host[0]} 'sudo /bin/php-pmp "${pid}"' | tee -a "$TMP_FILE"

# Package and ship this recent stack dump to the qkviewer interface
tar -zcf "$trace_file" -C "${TMP_FILE%/*}" "${TMP_FILE##*/}"
curl -F "file=@${trace_file}" "${trace_url}/index.php?action=upload&hostname=${HOSTNAME}&port=${PORT}"
cleanup
```

Defense #16.1 – Autorun php-pmp



```
#!/usr/bin/env bash
SUDOUSER="geoff"
QK_FILE="/var/log/qkill.log"
date="$(date +%Y_%m_%d_%H_%M_%S)"
TMP_FILE="/tmp/${date}-qkill-trace"
trace_file="/tmp/${HOSTNAME}-${date}.tar.gz"
trace_url="http://raingauge.example.com"
PORT='3306'
```

```
cleanup() {
    rm -f "$TMP_FILE" "$trace_file"
}
```

```
query="$(tac "${QK_FILE}" | sed -n '0,/^\#/ p' | tac)"
mysql_client="$(tac "${QK_FILE}" | sed -n '0,/^\#/ p' | tail -n1 | awk '{ print $8 }' | cut -d'"' -f4)"
echo "mysql client host: ${mysql_client}" | tee "$TMP_FILE"
```

Extract source client info

```
if [[ -z "${mysql_client}" ]]
then
    echo "Can't extract client host and port from pt-kill log ${mysql_client}"
    cleanup
    exit 1;
fi
IFS=: read -ra mysql_host <<< "${mysql_client}"
```

```
pid="$(sudo -u "${SUDOUSER}" ssh -o StrictHostKeyChecking=no "${mysql_host[0]}" "${mysql_host[1]}" | awk '{ print $2 }' | sed -n 1p)"
echo "remote pid: ${pid}" | tee -a "$TMP_FILE"
```

Run php-pmp on client pid

```
echo "${query}" | tee -a "$TMP_FILE"
if [[ -z "${pid}" ]]
then
```

Send to raingauge!

```
    echo "Can't find remote pid: ${mysql_host[0]}"
    cleanup
    exit 1
fi
```

Discover remote pid

```
sudo -u "${SUDOUSER}" ssh -n -o StrictHostKeyChecking=no "${mysql_host[0]}" 'sudo /bin/php-pmp "${pid}" | tee -a "$TMP_FILE"
```

```
# Package and ship this recent stack dump to the overview interface
tar -zcf "$trace_file" -C "${TMP_FILE%/*}" "${TMP_FILE##*/}"
curl -F "file=@${trace_file}" "${trace_url}/index.php?action=upload&hostname=${HOSTNAME}&port=${PORT}"
cleanup
```

Defense #16.1 – Autorun php-pmp



```
#!/usr/bin/env bash
SUDOUSER="geoff"
QK_FILE="/var/log/qk
date="$(date +%Y_%m
TMP_FILE="/tmp/${da
trace_file="/tmp/${
trace_url="http://r
PORT='3306'
```

```
cleanup() {
    rm -f "$TMP
}
```

```
query="$(tac "${QK_
mysql_client="$(tac
echo "mysql client
if [[ -z "${mysql_c
then
```

```
    echo "Can't
    cleanup
    exit 1;
```

```
fi
IFS=';' read -ra my
```

```
pid="$(sudo -u "${S
echo "remote pid: $
echo "${query}" | t
if [[ -z "${pid}" ]
then
```

```
    echo "Can't
    cleanup
    exit 1
```

```
fi
```

```
sudo -u "${SUDOUSER
```

```
# Package and ship
```

```
tar -zcf "$trace_fi
curl -F "file=@${tr
cleanup
```

Box Rain Gauge Servers

db.example.com:3306

Sat, 11 Apr 2015 22:14:42 -0700

Previous timeNext time

2015_04_11_22_14_42
qkill-trace

Search in files: 0 lines of context regex pattern Search

Sift | Netstat Summary

Dumping backtraces for pid 9224 to php-pmp.zhjqgq for 1 iterations.
Sleeping for 1s.
Finished dumping to php-pmp.zhjqgq.
[Thread debugging using libthread_db enabled]
0x00000000005ddeb5 in zval_mark_grey ()
#0 0x00000000005ddeb5 in zval_mark_grey ()
#1 0x00000000005def0d in gc_collect_cycles ()
#2 0x00000000005df134 in gc_zval_possible_root ()
#3 0x00000000005cc43b in zend_hash_destroy ()
#4 0x00000000005be5db in _zval_dtor_func ()
#5 0x0000000000646688 in ZEND_ASSIGN_SPEC_CV_VAR_HANDLER ()
#6 0x0000000000627e98 in execute ()
#7 0x000007fdeea1543d4 in augmented_types_execute(_zend_op_array*) () from /usr/lib64/php/modules/augmented_types.so
#8 0x000007fdee9f25a99 in xdebug_execute (op_array=0x7fdef467a808) at /root/rpmbuild/BUILD/xdebug-2.2.1/xdebug.c:1391
#9 0x00000000005be900 in zend_execute_scripts ()
#10 0x0000000000561dc7 in php_execute_script ()
#11 0x000000000066a05c in do_cli ()
#12 0x000000000066a7d8 in main ()
[0x7fdef4640060] ??? /home/ganderson/test.php:46
15 405834029.1428815501 2015-04-11 22:11:41

```
{ print $2 }'"")"
```



```
pt-stalk \  
  --daemonize  
  --variable=Threads_running  
  --threshold=150  
  --pid=/usr/var/run/ptstalk.pid  
  --cycles=2  
  --sleep=60  
  --exec-after-sleep=/usr/bin/raingauge_package_and_send.sh \  
  --dest=/tmp/raingauge
```

What about Threads_Created?
What about CPU?

Defense #17 – Multivariable pt-stalk



pt-stalk \

```
# Function called by pt-stalk-raingauge
```

```
trg_plugin() {
```

```
# Evaluate all trigger commands, calling get_delta if necessary
```

```
# e.g. trigger_name="$(hash_command_to_execute)"
```

```
seconds_behind_master="$(mysql $EXT_ARGV -e "SHOW SLAVE STATUS\G" -ss | grep -i seconds_behind | awk '{print $2}')" 
```

```
threads_running="$(mysql $EXT_ARGV -e "SHOW GLOBAL STATUS LIKE 'Threads_running'" -ss | awk '{print $2}')" 
```

```
cpu_percentage="$(top -d 0.1 -bn2 | grep "Cpu(s)" | tail -n 1 | awk '{print $2+$4+$6}')" 
```

```
threads_created="$(get_delta "threads_created" "$(mysql $EXT_ARGV -e "SHOW GLOBAL STATUS LIKE 'Threads_created'" -ss | awk '{print $2}'))"
```

```
# Check triggers against their threshold
```

```
# Nonzero value will cause pt-stalk-raingauge to collect and is used to reference which trigger was set off
```

```
# See which trigger set off collector by peeking in /var/log/pt-stalk.log
```

```
# Checks are formatted this way to handle comparisons involving floating point numbers
```

```
# To add new trigger, copy the format below and change variables, and increment echo number
```

```
if (( "$(echo "$seconds_behind_master" | awk '{print ($1 > 10)}')" )); then
```

```
    echo '1'          # seconds behind master
```

```
elif (( "$(echo "$threads_running" | awk '{print ($1 > 150)}')" )); then
```

```
    echo '2'          # threads_running
```

```
elif (( "$(echo "$cpu_percentage" | awk '{print ($1 > 50)}')" )); then
```

```
    echo '3'          # cpu_percentage
```

```
elif (( "$(echo "$threads_created" | awk '{print ($1 > 100)}')" )); then
```

```
    echo '4'          # threads_created
```

```
# elif (( "$(echo "$_triggername" | awk '{print ($1 _comparison _threshold)}')" )); then
```

```
    # echo '5'          # _triggername
```

```
else
```

```
    echo '0'          # Nothing triggered
```

```
fi
```

```
}
```

Defense #17 – Multivariable pt-stalk



 **box / RainGauge**

 Unwatch ▾ 63

Adds support for custom pt-stalk triggers #21

 **Merged** **gtowey** merged 3 commits into `box:master` from `mattagra:custom_triggers2` on Aug 7, 2014

 Conversation 3  Commits 3  Files changed 4



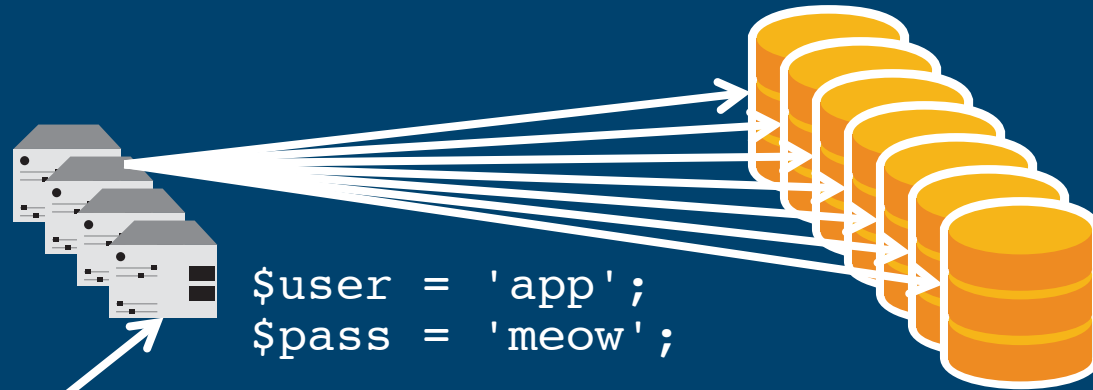
mattagra commented on Aug 4, 2014

This allows us to provide multiple test cases in which pt-stalk should collect server data
To use original pt-stalk functionality, leave PT_STALK_FUNCTION in scripts/raingauge_rc empty
Add/remove custom triggers in raingauge_triggers.sh
Refer to /var/log/pt-stalk.log to see which trigger caused collection
Adds function get_delta in raingauge_triggers.sh to calculate change in counter values

  Adds support for custom pt-stalk triggers ...

a677dea

Time to rotate passwords!



Defense #18 – Automated User Management

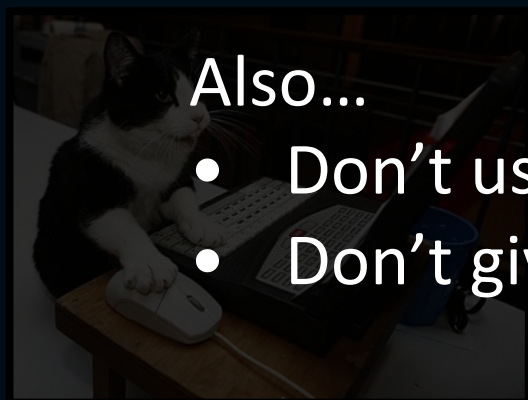


In all seriousness...why?

- Security exposures
- Accidental deletions by shared accounts
- Setup quotas!

Also...

- Don't use your root account
- Don't give nagios the SUPER privilege...



Defense #18 – Automated User Management



MySQL Workbench - MyFirstConnection

Administration - Users and Privileges

MyFirstConnection

Users and Privileges

User Accounts

User	From Host	Login	Account Lock
general	localhost		
mysqlbackup	localhost		
root	localhost		

Details for account

Role

- DBA
- Maintenance
- ProcessAdmin
- UserAdmin
- SecurityAdmin
- MonitorAdmin
- DBManage
- DBDesign
- Replication
- BackupAdmin
- Custom

ORACLE MySQL Enterprise Monitor

Dashboards Events Query Analyzer Reports & Graphs Configuration

Manage Users

Create User

Show 10 entries

User Login	Full Name	Role	Authentication	Password set date	Last Login	Failed logins	Last failed login
agent	agent	Built-in		Aug 19, 2013 6:49:53 PM		0	
manager	manager	Built-in		Aug 19, 2013 6:49:53 PM	Aug 22, 2013 11:17:50 AM	0	

google / mysql-tools

Watch 9 Star 18 Fork 5

branch: master mysql-tools / permissions.py

flamingcow@google.com on Feb 11, 2013 Lint cleanup, switch to gflags.

0 contributors

Executable File 189 lines (155 sloc) 6.156 kb

Raw Blame History

https://forge.puppetlabs.com/puppetlabs/mysql#mysql_user

mysql_user

Creates and manages user grants within MySQL.

```
1 mysql_user { 'root@127.0.0.1':
2   ensure           => 'present',
3   max_connections_per_hour => '0',
4   max_queries_per_hour   => '0',
5   max_updates_per_hour   => '0',
6   max_user_connections   => '0',
7 }
```

Reserved.

version 2.0 (the "License");

compliance with the License.

at

CENSE-2.0

agreed to in writing, software

tributed on an "AS IS" BASIS,

ANY KIND, either express or implied.

uage governing permissions and

ons.

Defense #18 – Automated User Management



MySQL Workbench
MyFirstConnection x
File Edit View Query Database Server Tools Scripting Help

MySQL 5.6 Reference Manual :: 6 Security :: 6.3 MySQL User Account Management :: 6.3.8.5 The PAM Authentication Plugin

6.3.8.5 The PAM Authentication Plugin

Note

The PAM authentication plugin is a commercial extension. To learn more about commercial products (MySQL Enterprise Edition), see <http://www.mysql.com/products/enterprise/>

As of MySQL 5.6.10, commercial distributions of MySQL include an authentication plugin that

enables MySQL Server to use PAM to authenticate users. PAM enables MySQL Server to use the same authentication method as the operating system.

The PAM plugin uses the same authentication method as the operating system. The plugin checks the user's name, password, and other information against the operating system's user database.

The plugin checks the user's name, password, and other information against the operating system's user database. If the authentication succeeds, the user is granted access to the MySQL Server.

The PAM authentication plugin is a commercial extension. To learn more about commercial products (MySQL Enterprise Edition), see <http://www.mysql.com/products/enterprise/>

• External authentication

MariaDB Products Services Customers Resources
Home » Resources » Knowledge Base » MariaDB » MariaDB Documentation » Managing Users

Roles Overview

Home

Open Questions

MariaDB

All Topics

History

Edit

Source

Flag as

Spam/inappropriate

Translate

Created

1 year, 5 months ago

Modified

MariaDB starting with 10.0.5

Roles were introduced in MariaDB 10.0.5.

A role bundles a number of privileges together. It assists larger organizations where, typically, a number of users would have the same privileges, and, previously, the only way to change the privileges for a group of users was by changing each user's privileges individually.

Alternatively, multiple external users could have been assigned the same user, and there would have been no way to see which actual user was responsible for which action.

With roles, managing this is easy. For example, there could be a number of users assigned to a journalist role, with identical privileges. Changing the privileges for all the journalists is a matter of simply changing the role's privileges, while the individual user is still linked with any changes that take place.

Roles are created with the `CREATE ROLE` statement, and dropped with the `DROP ROLE` statement. Roles are then assigned to a user with an extension to the `GRANT` statement, while privileges are assigned to a role in the regular way with `GRANT`. Similarly, the `REVOKE` statement can be used to both revoke a role from a user, or revoke a privilege from a role.

Once a user has connected, he can obtain all privileges associated with a role by **setting** a role with the `SET ROLE` statement. The `CURRENT_ROLE` function returns the currently set role for the session, if any.

Only roles granted directly to a user can be set, roles granted to other roles cannot. Instead the privileges granted to a role, which is, in turn, granted to another role (grantee), will be immediately

Contents

1. Examples
2. [Roles and views \(and stored routines\)](#)
3. Other resources

Roles Overview

CREATE ROLE

DROP ROLE

CURRENT_ROLE

SET ROLE

SET DEFAULT ROLE

GRANT

REVOKE

Information

Schema

APPLICABLE_ROLES

Table

Information

Schema

ENABLED_ROLES

Table

SecuRich



mySQL
Create
1
2
3
4
5
6
7

M

T

W

Th

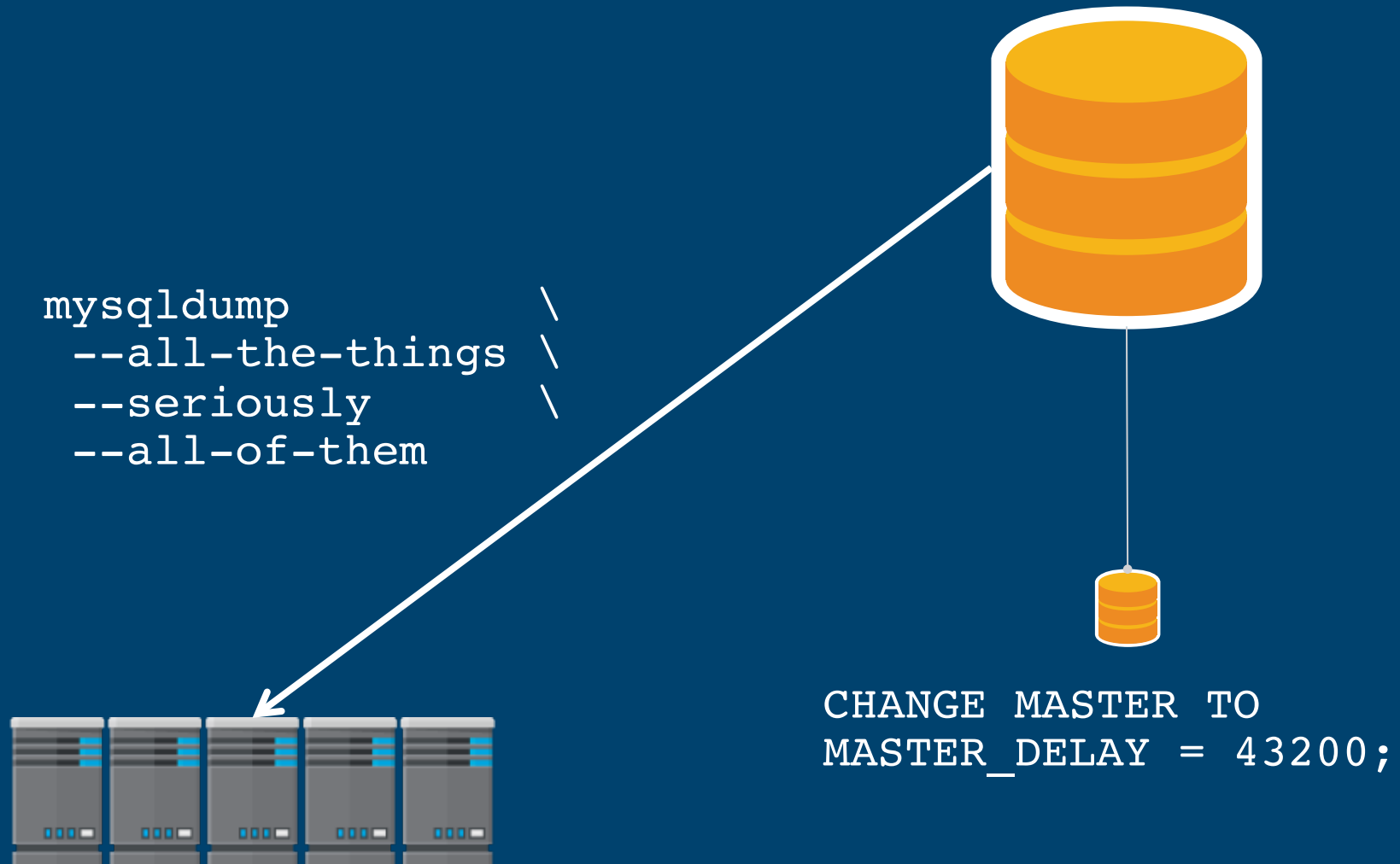
F



Backups

How many of you will take backups?
How many of you actually take backups?

Defense #19 – Backups



Defense #19 – Backups



Using LVM for MySQL Backup and Replication Setup



August 21, 2006 | by [Peter Zaitsev](#) | 56 Comments

[Like](#) 0

[+1](#) 4

[Tweet](#) 0

[in Share](#)

« PREVIOUS POST

If someone asks me about MySQL Backup advice my first question would be if they have some systems with similar features set for other operation systems. Veritas File S...

PERCONA XTRABACKUP



PERCONA
XTRABACKUP

over 800,000 times since its release. achieve the following benefits:

- Backups that complete quickly and
- Uninterrupted transaction processing
- Savings on disk space and network
- Automatic backup verification
- Higher uptime due to faster restore



zmanda
A Carbonite Company

[store](#) | [wiki](#) | [forum](#) | [blog](#)

Home

Amanda Network Backup

MySQL Backup

Zmanda Cloud Backup

Company

MySQL Backup and Recovery

Zmanda Recovery Manager for MySQL

Zmanda Recovery Manager (ZRM) for MySQL simplifies the life of a Database Administrator who needs an easy-to-use yet flexible and robust backup and recovery solution for MySQL server. With ZRM for MySQL you can:

MySQL Enterprise Backup User's Guide (Version 3.12.0)

MySQL Enterprise Backup User's Guide (Version 3.12.0)

Abstract

Defense #19 – Backups



A backup today saves you tomorrow

Because bad things do happen

Andrew Moore, MySQL DBA

Ben Mildren, MySQL Team Technical Lead

over 800,000 times since its release. With Percona achieve the following benefits:

- Backups that complete quickly and reliably
- Uninterrupted transaction processing during backup
- Savings on disk space and network bandwidth
- Automatic backup verification
- Higher uptime due to faster restore time



Search Facebook



Under the Hood: Automated backups

By Eric Barrett on Monday, January 14, 2013 at 11:30am

Facebook has one of the largest MySQL installations in the world, with thousands of database servers in multiple regions, so it's no surprise that we have unique challenges when we take backups. Our job is to keep every piece of information you add safe, while ensuring that anything that's been deleted is purged in a timely manner.

To accomplish this, we've built a highly automated, extremely effective backup system that moves many petabytes a week. Rather than extensive front-loaded testing, we emphasize rapid detection of failures and quick, automated correction. Deploying hundreds of new database servers requires very little human effort and lets us grow at the pace and flexibility required to support more than a billion active users. Here's a bit about how it works.



breathe

Defense #20 – Prevent burnout

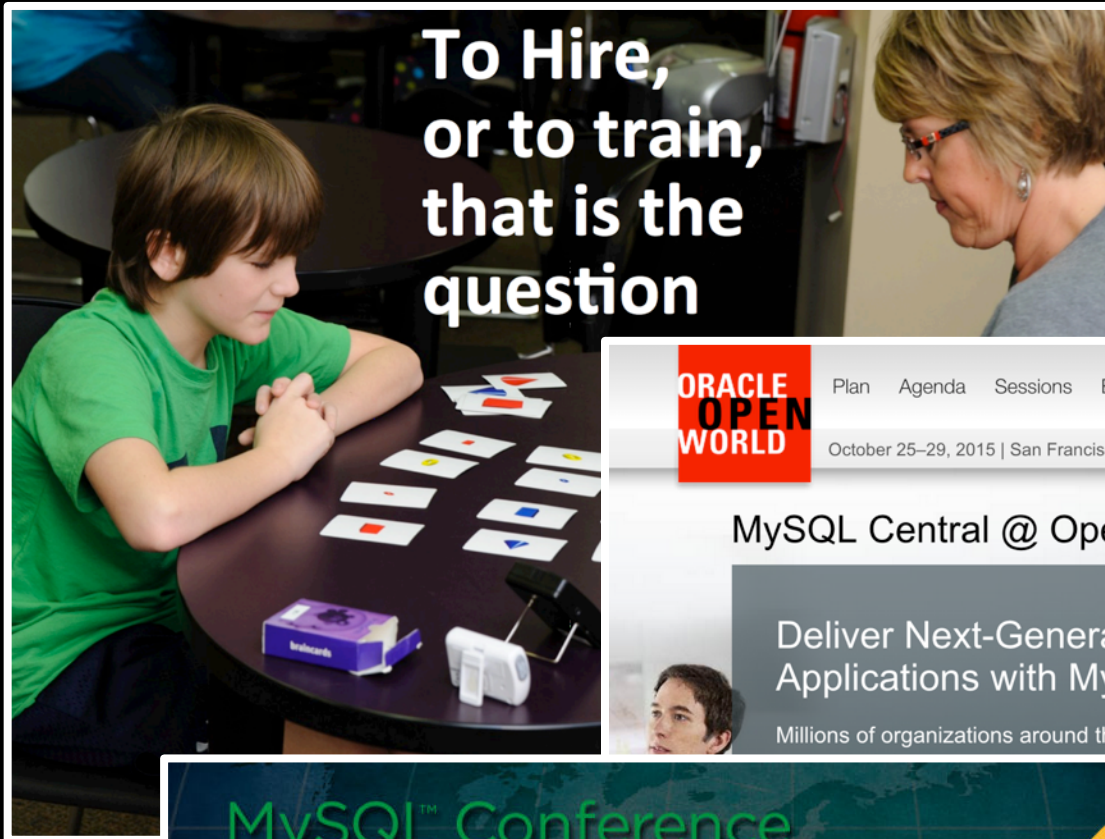
Relax

Take vacations

Hire more DBAs

breathe

Defense #20 – Prevent burnout



ORACLE OPEN WORLD

Plan Agenda Sessions Exhibitor / Sponsor Network

October 25-29, 2015 | San Francisco Conferences

MySQL Central @ OpenWorld

Deliver Next-Generation Applications with MySQL

Millions of organizations around the world trust MySQL to

MySQL

MySQL™ Conference & Expo 2015 | April 13-16 Santa Clara, CA

Learn from leading MySQL experts.

openstack LIVE Includes admission to OpenStack Live! > Check it out

MYSQL WORLDWIDE CONFERENCE & EXPO April Santa Clara

Defense #20 – Prevent burnout



A BLOG ABOUT ANALYTICS, MARKETING AND TESTING



Infographics



Marketing Guides



Webinars

FREE EMAIL UPDATES

Get the latest content first.

Join Us

FREE WEBINAR:

APR. 16TH,
1PM EST
10AM PST

CASSIE
OUMEDIAN
FROM
HANAPIN
MARKETING



Analyzing Your Key
Metrics: 7 Areas You Must
Examine for PPC Success

REGISTER NOW

How to Prevent Employee Burnout

We all know what burnout looks like.

It's the stressed-out manager who proclaims they've "**had it**" and books the next flight to Cancun.

The start-up co-founder **working 130 hour weeks** who claims he "**can't take all this bullsh*t anymore**" and goes on a week-long bender in Miami.

The boss who **yells with such intensity** you fear he might have a heart attack.

It is so common that everyone in any career – engineers, sales managers, tech support personnel, etc. – is susceptible to burnout. Despite being so common, many managers aren't aware of why burnout happens or how to keep it from happening. But they need to know.

Being able to understand burnout, its causes, and how to prevent it is essential in order to maintain a positive environment and keep the best talent on the team.

Burnout Defined

Burnout is an individual's response to chronic emotional and interpersonal stressors within the workplace (Maslach et al., 2001).

66

Like

555

Tweet

40

+1

173

Share



Defenses

- Defense #1 – Query Comments
- Defense #2 – Alerting on the Database
- Defense #3 – Dashboards
- Defense #4 – User and Table Statistics
- Defense #5 – Box Raingauge
- Defense #6 – Monitor the Queries
- Defense #7 – Sharding
- Defense #8 – Innotop
- Defense #9 – Query Killer
- Defense #10 – Transaction Monitor (and killer)
- Defense #11 – Dynamic Query Killer Thresholds
- Defense #12 – Healthchecking
- Defense #13 – Data Layer as a Service
- Defense #14 – Lock Monitor (and killer)
- Defense #15 – strace Errant Connections
- Defense #16 – ~~pt-ppp~~ php-pmp
- Defense #17 – Multivariable pt-stalk
- Defense #18 – Automated User Management
- Defense #19 - Backups
- Defense #20 – Prevent Burnout

COMMUNITY NETWORKING RECEPTION (IN EXPO HALL), CO-SPONSORED BY: YELP, BOX AND GITHUB - MYSQL COMMUNITY AWARDS & LIGHTNING TALKS

▼ [Schedule info](#)

Status: Accepted

Time Slot:

15 April 5:30PM - 7:00PM

Room:

None

#DBHANGOPS IN REAL LIFE

14 April 6:00PM - 7:00PM @ Room 203

Experience level: -

Duration: Birds of a Feather

A live, in-person meeting with some of the participants of bi-weekly virtual meetup to simply talk about anything database-related and solutions to common problems. Also be sure to check out <http://dbhangops.github.io> for previous recordings and discussions!



Come to the **#DBHangOps BoF @ 6pm in room 203!**

Email geoff@box.com

Twitter [@geodbz](https://twitter.com/geodbz)

Tech Blog www.box.com/blog/engineering

Github github.com/box





Links of Supreme Interest!

- <https://www.percona.com/live/mysql-conference-2014/sessions/box-weather-station-open-source-tools-performance-and-forensic-monitoring>
- <https://www.percona.com/live/mysql-conference-2014/sessions/mysql-devops-outbrain>
- <http://www.percona.com/live/mysql-conference-2012/sessions/etsy-shard-architecture-starts-s-and-ends-hard>
- <http://www.infoq.com/presentations/box-mysql-sharding>



Image Credit / Source

- http://ak-hdl.buzzfed.com/static/2014-06/19/12/enhanced/webdr06/anigif_enhanced-26665-1403194377-3.gif
- http://31.media.tumblr.com/5f6ffc7bf7f456ba0442596d07faa925/tumblr_n4zmymPU191s0teago2_r1_400.gif
- https://www.flickr.com/photos/patrick_verstappen/8421238929
- <https://www.flickr.com/photos/apocalust/5092520561>
- http://38.media.tumblr.com/e1a7a9881573597bbe97ade824f578a3/tumblr_ngcsl73nLt1u4i1dio4_1280.gif
- <http://stream1.gifsoup.com/view3/4891599/lion-king-stampede-o.gif>
- http://24.media.tumblr.com/tumblr_lv5jvwWTyT1r4hwmoo1_500.gif
- <http://cl.jroo.me/z3/2/R/v/e/a.baa-the-hacker-cat.jpg>
- https://www.flickr.com/photos/daryl_mitchell/1199598508
- <https://www.flickr.com/photos/patrice-photographiste/14909768574>