Java MySQL Connector & Connection Pool

Features & Optimization

Kenny Gryp
<kenny.gryp@percona.com>
April 14, 2015
@gryp

PERCONA
LIVE

Please excuse me for not being a Java developer

DISCLAIMER

What I Don’t Like

- Brussels Sprouts
- Taxes
- Calories
- Java(’s chattiness)

MYSQL CONNECTORS

CONNECTION POOLS

Connectors

Configuring Connector

Creating A Database Connection

Prepared Statements

Example Transaction

MYSQL CONNECTORS

CONNECTION POOLS

MySQL Connector/J & MariaDB Java Client

- MySQL Connector/J
  - Oracle
  - 5.1.35 Latest
  - Compatible with
    - MySQL
    - Percona Server
    - MariaDB

MySQL Connector/J & MariaDB Java Client

- MariaDB Java Client
  - MariaDB
  - Fork from Drizzle Connector
  - Latest 1.1.8
  - Compatible with
    - MySQL
    - Percona Server
    - MariaDB

MySQL Connector/J Features

- Enterprise Plugin: Query Analyzer
- MySQL Fabric Integration
- Load Balancing
- Failover
- Replication

http://ceilingcat.ninja

Connectors

Configuring Connector

Creating A Database Connection

Prepared Statements

Example Transaction

MYSQL CONNECTORS

CONNECTION POOLS

Creating Connection


```
Connection con =
    DriverManager.getConnection
    (“jdbc:mysql://node2/employees?
        user=connj&password=test");
Statement stmt = con.createStatement();
String query =
    "select * from employees
        where emp_no = 20000;";
ResultSet rs = stmt.executeQuery(query);
...
```

MariaDB:

```
jdbc:mariadb://node2/employees
    ?user=connj&password=test"
```

Creating Connection - Tomcat w. JDBC-Pool


```
context.xml (local):
<Resource name="jdbc/test"
 auth="Container"
 type="javax.sql.DataSource"
 username="jdbc-pool" password="test"
 driverClassName="com.mysql.jdbc.Driver"
 url="jdbc:mysql://node2:3306/employees"
/>
```

MariaDB:

```
 driverClassName="org.mariadb.jdbc.Driver"
```

Creating Connection - JDBC URL


```
jdbc:mysql://node2:3306/employees?
useServerPrepStmts=true&...
```

http://ceilingcat.ninja

Connectors

Configuring Connector

Creating A Database Connection

Prepared Statements

Example Transaction

MYSQL CONNECTORS

CONNECTION POOLS

Connector/J - Creating Connection


```
SHOW VARIABLES WHERE
   Variable_name ='language' OR…
SELECT
   @@session.auto_increment_increment;
SET NAMES latin1;
SET character_set_results = NULL;
SET autocommit=1;
SET sql_mode=
   'NO_ENGINE_SUBSTITUTION,STRICT_TRAN
S_TABLES';
```

MariaDB - Creating Connection


```
set autocommit=1;
USE employees;
show variables like 'sql_mode';
```

Creating Connection - Defaults

- Connector/J:

- MariaDB Java Client:

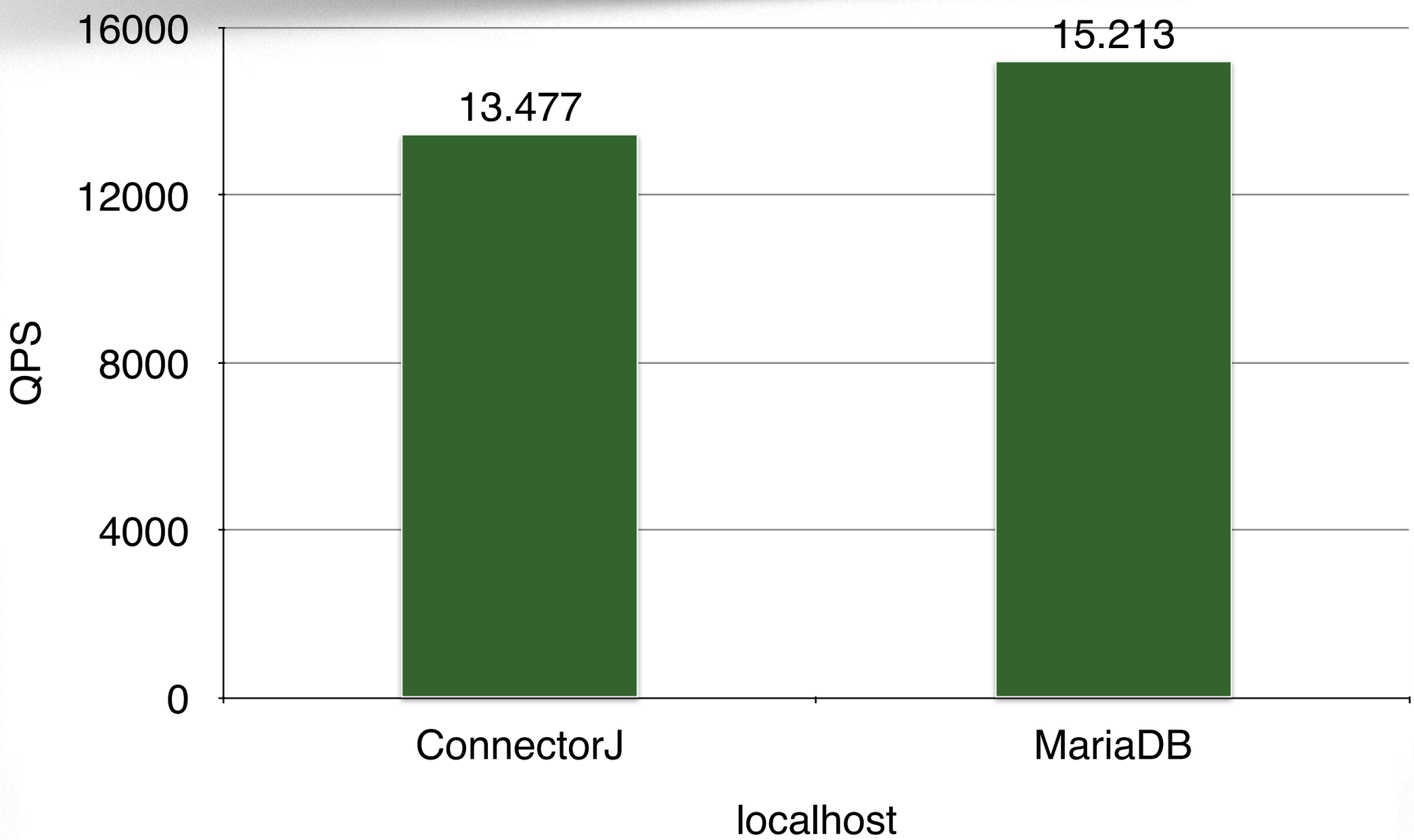
Connector/J & MariaDB Java Client - Verbosity

- Connector/J is more verbose when starting a connection
- Usually not a problem:
  - connection pools are commonly used (more coming soon…)
  - connections are reused
- Actually I like  but not too much.

Optimization

- MariaDB Java Client vs MySQL Connector/J
- Prepared Statements

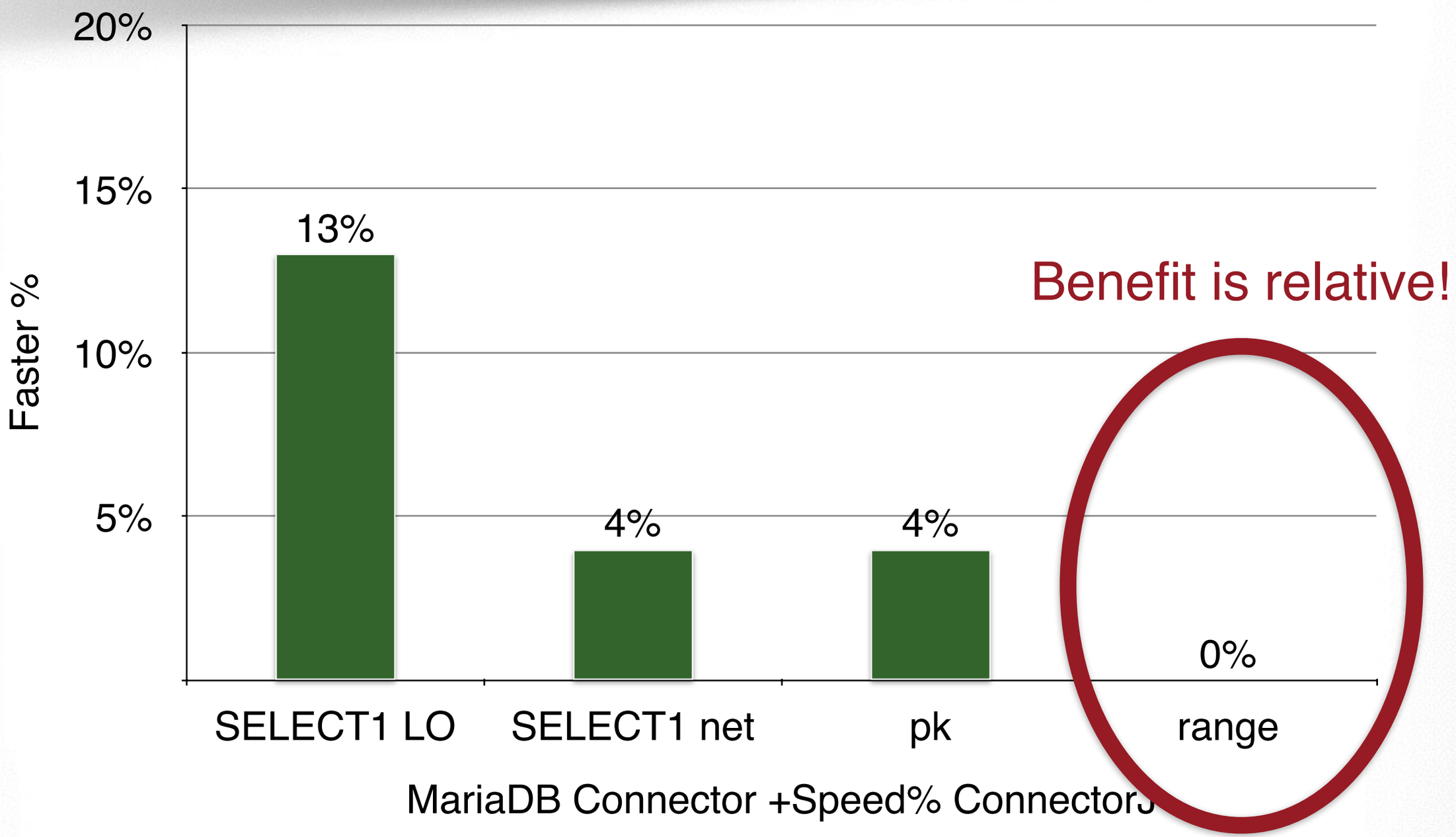
Connector Performance - SELECT 1 localhost, single threaded



Connector Performance - MariaDB %faster

MariaDB Connector +Speed% ConnectorJ

Connector Performance - MariaDB %faster



Connectors

Configuring Connector

Creating A Database Connection

Prepared Statements

Example Transaction

MYSQL CONNECTORS

CONNECTION POOLS

Client or Server Side Prepared Statements

- Server side Prepared statements:
  - reduce network traffic
  - query is already optimized on server.
- Support:
  - MariaDB Java client only supports client side
  - Connector/J default in client side

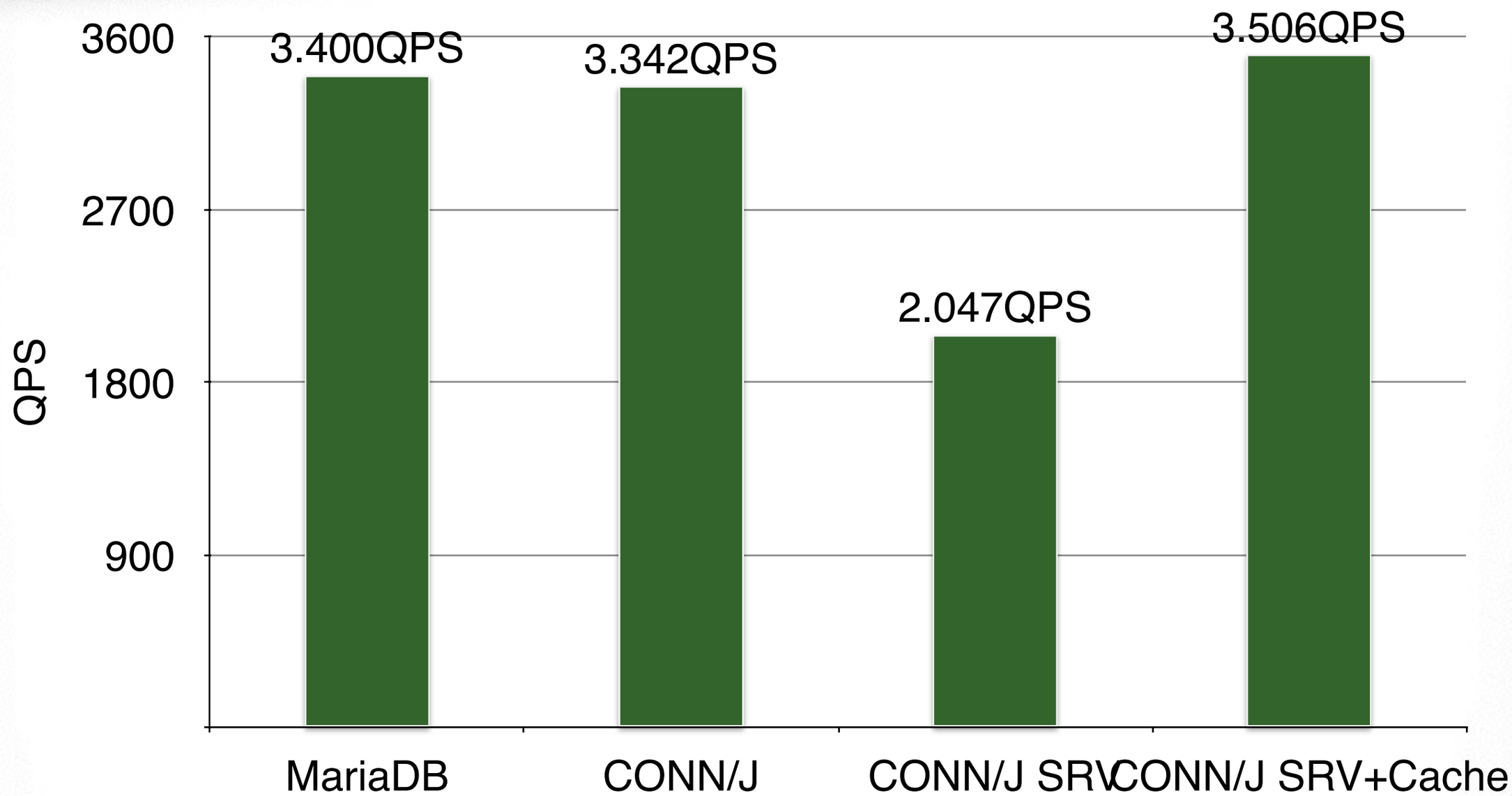
Server Side Prepared Statements


```
PREPARE stmt1 FROM
   select * from employees
      where emp_no = ?;
EXECUTE # API CALL
   select * from employees
      where emp_no = 20000;
DEALLOCATE PREPARE stmt1;
```

Connector/J: Server Side Prepared Statements

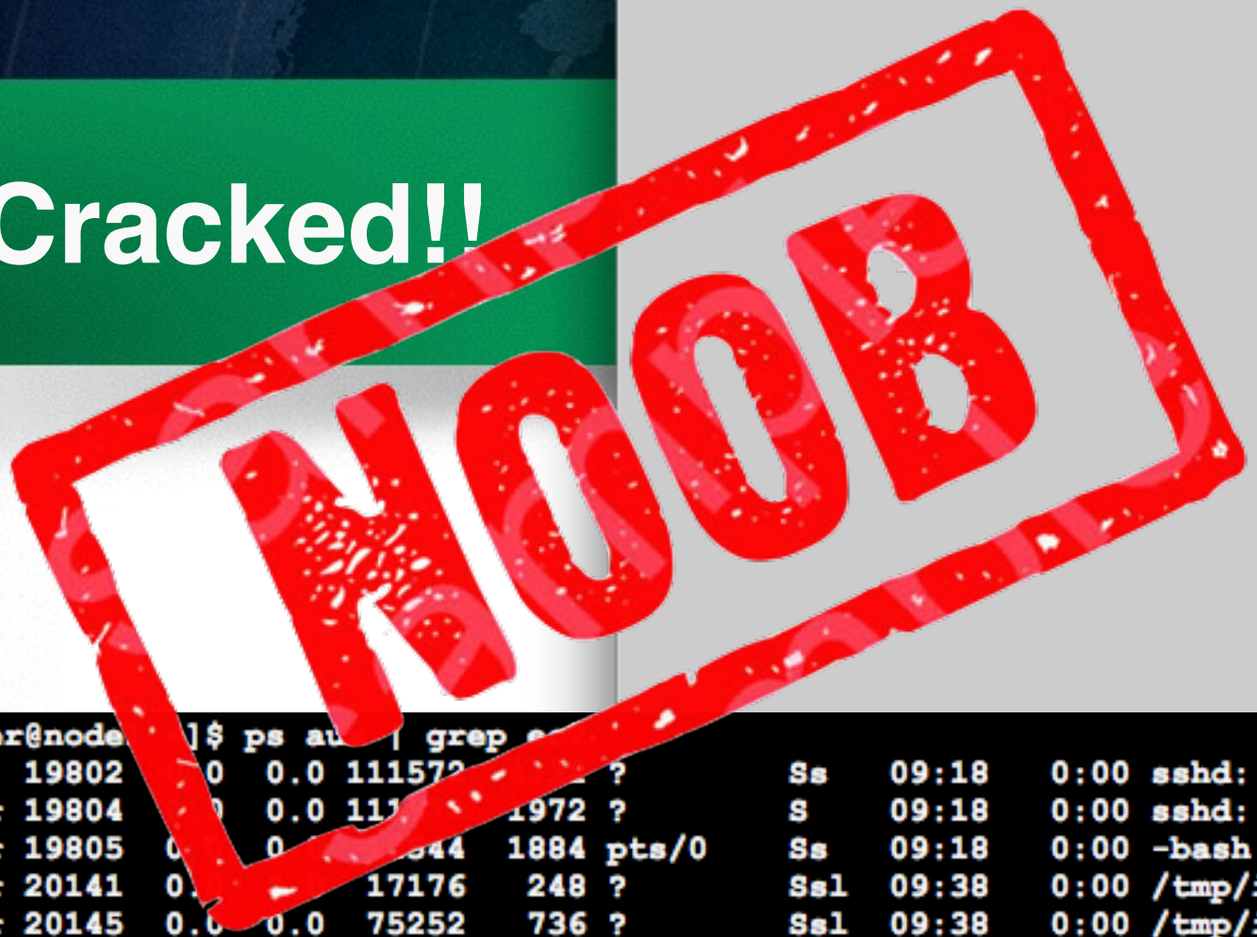
- `useServerPrepStmts = false`
  - disabled by default
- Mind looking at:
  - `cachePrepStmts = false`
    - do `PREPARE, EXECUTE, DEALLOCATE` every time…, 3 round trips?
  - `prepStmtCacheSize = 25`
  - `prepStmtCacheSqlLimit = 256`
    - low defaults https://bugs.mysql.com/bug.php?id=74932

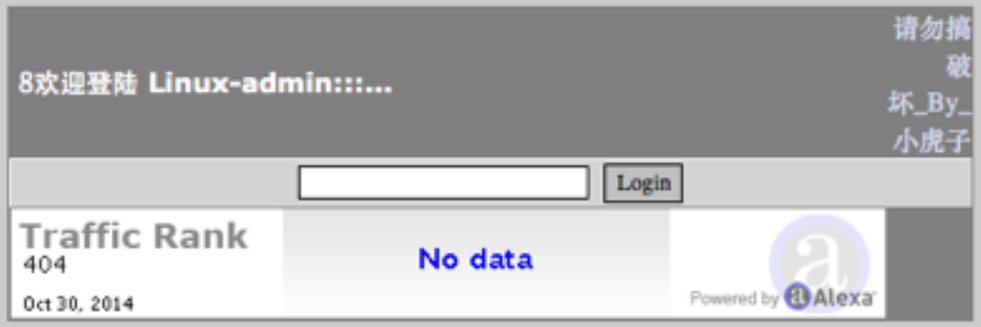
Benchmark: Prepared Statements




```
select * from employees_alotofindexes
        where first_name='moss' and birth_date > "1954-06-14"
        and gender="M" and hire_date > "1998-01-01"\G
```

Cracked!!






```
[ec2-user@node1 ~]$ ps au | grep ec2
root     19802  0.0  0.0 111572  [illegible] ?        Ss   09:18   0:00 sshd: ec2-user [priv]
ec2-user 19804  0.0  0.0 111[illegible]1972 ?        S    09:18   0:00 sshd: ec2-user@pts/0
ec2-user 19805  0.[illegible]  0.[illegible]544  1884 pts/0    Ss   09:18   0:00 -bash
ec2-user 20141  0.[illegible]  [illegible] 17176   248 ?        Ssl  09:38   0:00 /tmp/initial.lock
ec2-user 20145  0.0  0.0  75252   736 ?        Ssl  09:38   0:00 /tmp/inittd.lock
ec2-user 20164  0.0  0.0  13440  1104 pts/0    R+   09:38   0:00 ps aux
ec2-user 20165  0.0  0.0   6676   620 pts/0    S+   09:38   0:00 grep ec2
ec2-user 29459  0.0  0.0  17176   248 ?        Ssl  Oct24   1:05 /tmp/initial.lock > /dev/null 2>&1 &
[ec2-user@node1 ~]$
```

```
[root@node1 ec2-user]# lsof -p 20145
COMMAND     PID     USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
inittd.lo 20145 ec2-user  cwd    DIR  202,1     4096 402816 /home/ec2-user/hacked
inittd.lo 20145 ec2-user  rtd    DIR  202,1     4096      2 /
inittd.lo 20145 ec2-user  txt    REG  202,1  1223123     78 /tmp/inittd.lock
inittd.lo 20145 ec2-user    0u   CHR    1,3      0t0   1028 /dev/null
inittd.lo 20145 ec2-user    1u   CHR    1,3      0t0   1028 /dev/null
inittd.lo 20145 ec2-user    2u   CHR    1,3      0t0   1028 /dev/null
inittd.lo 20145 ec2-user    3uW  REG  202,1        5     79 /tmp/gates.lod
inittd.lo 20145 ec2-user    4u  IPv4 281082      0t0    TCP node1:54301->183.60.202.2:icl-twobase2 (ESTABLISHED)
[root@node1 ec2-user]# lsof -p 29459
COMMAND     PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
initial.1 29459 ec2-user  cwd    DIR  202,1     4096    2 /
initial.1 29459 ec2-user  rtd    DIR  202,1     4096    2 /
initial.1 29459 ec2-user  txt    REG  202,1   591801   77 /tmp/initial.lock
initial.1 29459 ec2-user    0u   CHR    1,3      0t0 1028 /dev/null
initial.1 29459 ec2-user    1u   CHR    1,3      0t0 1028 /dev/null
initial.1 29459 ec2-user    2u   CHR    1,3      0t0 1028 /dev/null
initial.1 29459 ec2-user    3u  IPv4 143246      0t0  TCP node1:49665->199.168.100.78:49870 (ESTABLISHED)
```

Connectors

Configuring Connector

Creating A Database Connection

Prepared Statements

Example Transaction

MYSQL CONNECTORS

CONNECTION POOLS

Connector/J Optimization + Default JDBC-Pool


```
Connection con = ds.getConnection();
con.setTransactionIsolation
  (Connection.TRANSACTION_READ_COMMITTED);
con.setAutoCommit(false);
PreparedStatement stmt =
con.prepareStatement("select * from
employees where emp_no = ?");
stmt.setInt(1, 20000);
ResultSet rs = stmt.executeQuery();
stmt.close();
rs.close();
con.commit();
con.close();
```

Connector/J Optimization + Default JDBC-Pool


```
SET SESSION TRANSACTION
      ISOLATION LEVEL READ COMMITTED;

SET autocommit=0;

# administrator command: Prepare;

select * from employees
          where emp_no = 20000;

# administrator command: Close stmt;

commit;
```

Connector/J Optimization

- useConfigs=maxPerformance
  - cachePrepStmts=true
  - cacheCallableStmts=true
  - cacheServerConfiguration=true
  - useLocalSessionState=true
  - elideSetAutoCommits=true
  - alwaysSendSetIsolation=false
  - enableQueryTimeouts=false

Connector/J Optimization

- `useLocalTransactionState=true`
  `commit() / rollback()`

Connector/J Optimization - Tuned

JDBC URL: **useConfigs=maxPerformance&useServerPrepStmts=true:**

```
select * from employees
   where emp_no = 20000;
commit;
```

MariaDB Java Client Optimization


```
SET SESSION TRANSACTION
      ISOLATION LEVEL READ COMMITTED;

select * from employees
          where emp_no = 20000;

COMMIT;
```

MariaDB Java Client Optimization - Code


```
if
(
 con.getTransactionIsolation() !=
        Connection.TRANSACTION_READ_COMMITTED
)
{
 con.setTransactionIsolation
        (Connection.TRANSACTION_READ_COMMITTED);
}
```

MariaDB Java Client Optimization - Interceptors


```
SELECT @@tx_isolation;
select * from employees
    where emp_no = 20000;
COMMIT;
```

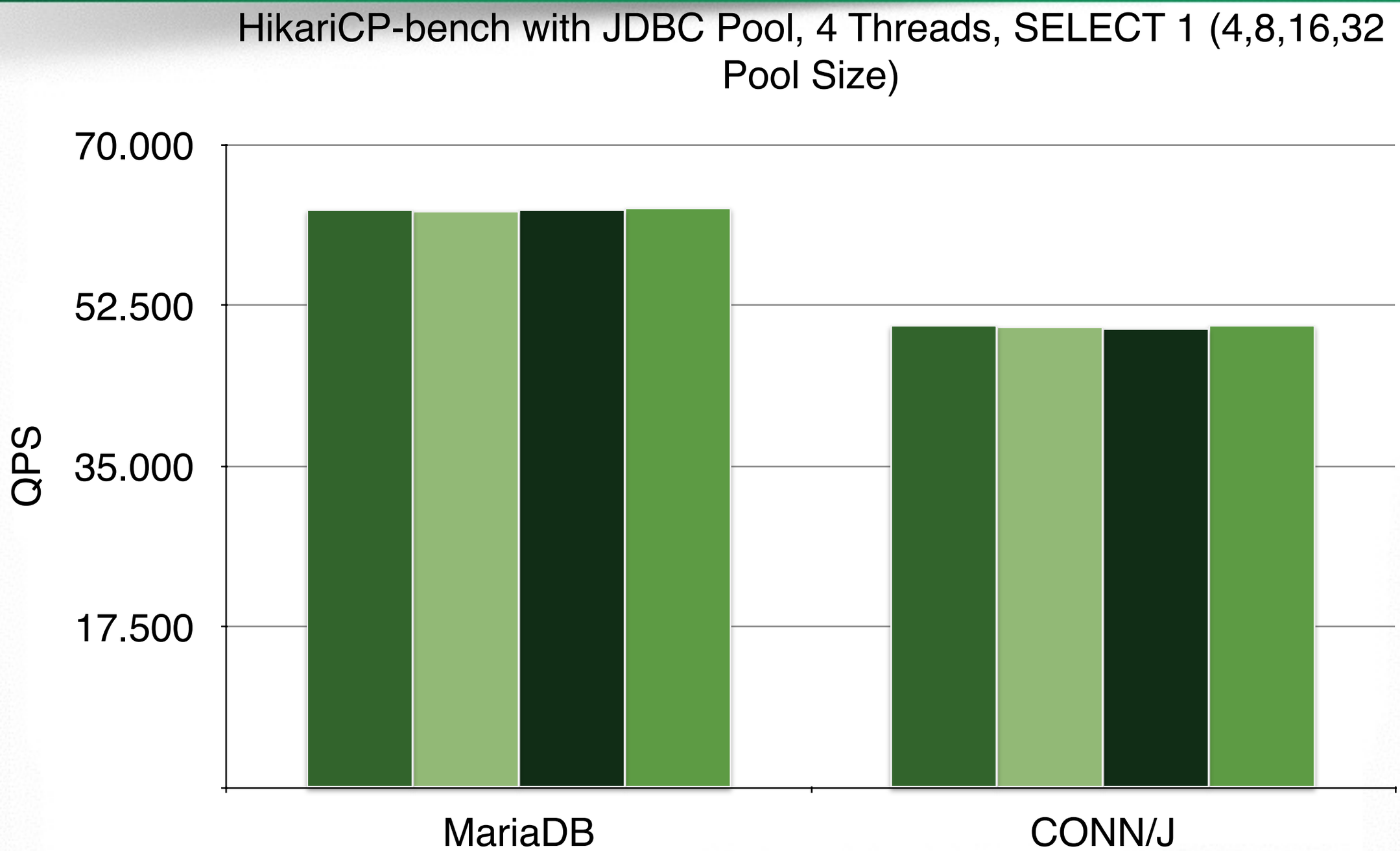
Still @@tx_isolation. Now add JDBC Interceptor:

```
<Resource name="jdbc/test"
auth="Container"
factory=
"org.apache.tomcat.jdbc.pool.DataSourceFactory"
jdbcInterceptors="ConnectionState"
driverClassName="org.mariadb.jdbc.Driver"
url="jdbc:mysql://node2:3306/employees"/>
```

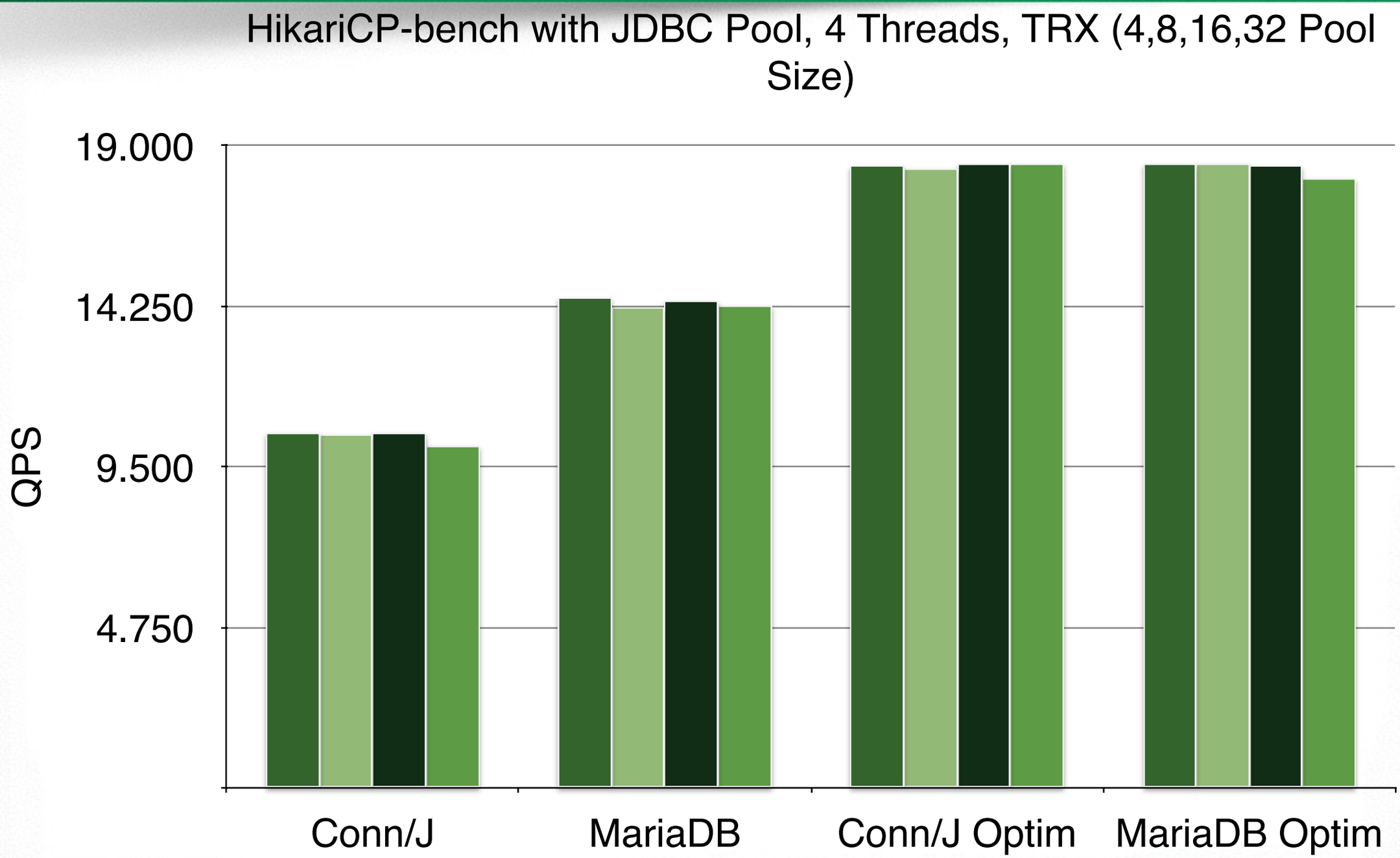
MariaDB Java Client Optimization - Optimized!


```
select * from employees
   where emp_no = 20000;
COMMIT;
```

Benchmark - Connector Concurrency - SELECT 1



Benchmark - Connector Concurrency - TRX



MYSQL CONNECTORS
CONNECTION POOLS

Connection Pools

Issues

Resetting Environment

Testing Connectivity

Pool Sizing

Lingering Transactions

Analysis

Examples

Graceful Failover

Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Java Connection Pools

- The Usual:
  - C3P0
  - Commons-DBCP (v1&v2)
  - JDBC Pool (fork commons-DBCP)
- Out In The Wild:
  - Vibur-DBCP
  - HikariCP
  - BoneCP

Java Connection Pools

- The Usual:
  - **C3P0**
  - **Commons-DBCP** (v1&**v2**)
  - **JDBC Pool** (fork commons-DBCP)
- Out In The Wild:
  - Vibur-DBCP
  - **HikariCP**
  - BoneCP

Connection Pool Key Points

- Connection Management
- Pool Sizing
- Connection Testing
- Avoid Lingering Transactions

http://ceilingcat.ninja

Issues

Resetting Environment
Analysis
Testing Connectivity
Examples
Pool Sizing
Graceful Failover
Lingering Transactions
Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool Issues

- Coming from DBA side, I do not like

5 Client Behavior

5.1 Connection Pool settings

Diving into the query workload, we noticed some typical issues when making use of the commons-dbcp and Spring connection pools, as illustrated in the following excerpt from the earlier pt-query-digest output.

```
# Profile
# Rank Query ID           Response time  Calls   R/Call Apdx V/M   Item
# ==== ================== ============== ======= ====== ==== ===== =======
# 1    0x813031B8BBC3B329 387.7963 45.7% 814893  0.0005 1.00 0.01  COMMIT
...
# 5    0x3AEAAD0E15D725B5 29.6623  3.5%  1630696 0.0000 1.00 0.00  SET
...
# 8    0x16219655761820A2 22.7324  2.7%  815082  0.0000 1.00 0.00  SELECT
...
# 11   0x943798A09019B333 14.4427  1.7%  1020842 0.0000 1.00 0.00  SHOW WARNINGS
...
# 17   0x19C8068B5C1997CD 6.3505   0.7%  522227  0.0000 1.00 0.00  ROLLBACK
```

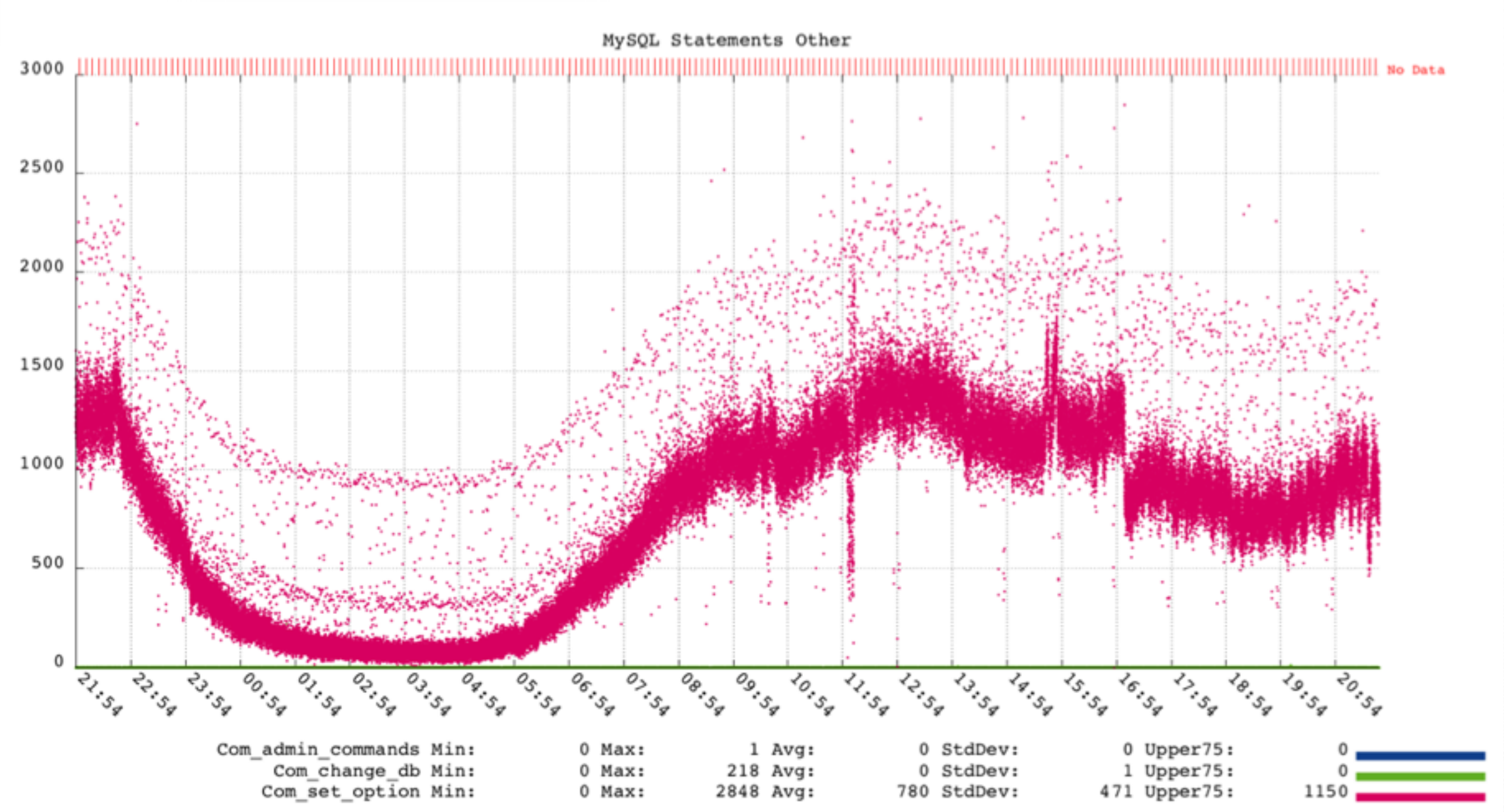
Together, these queries account for 54.2% of all response times, and 82% of all executed queries. This led us to investigate the settings used with the connections pools a bit closer, and we suggest the following changes the greatly reduce the response times involved in queries from these connections pools as well as the overall amount of unnecessary queries sent to the server.

Connection Pool Issues

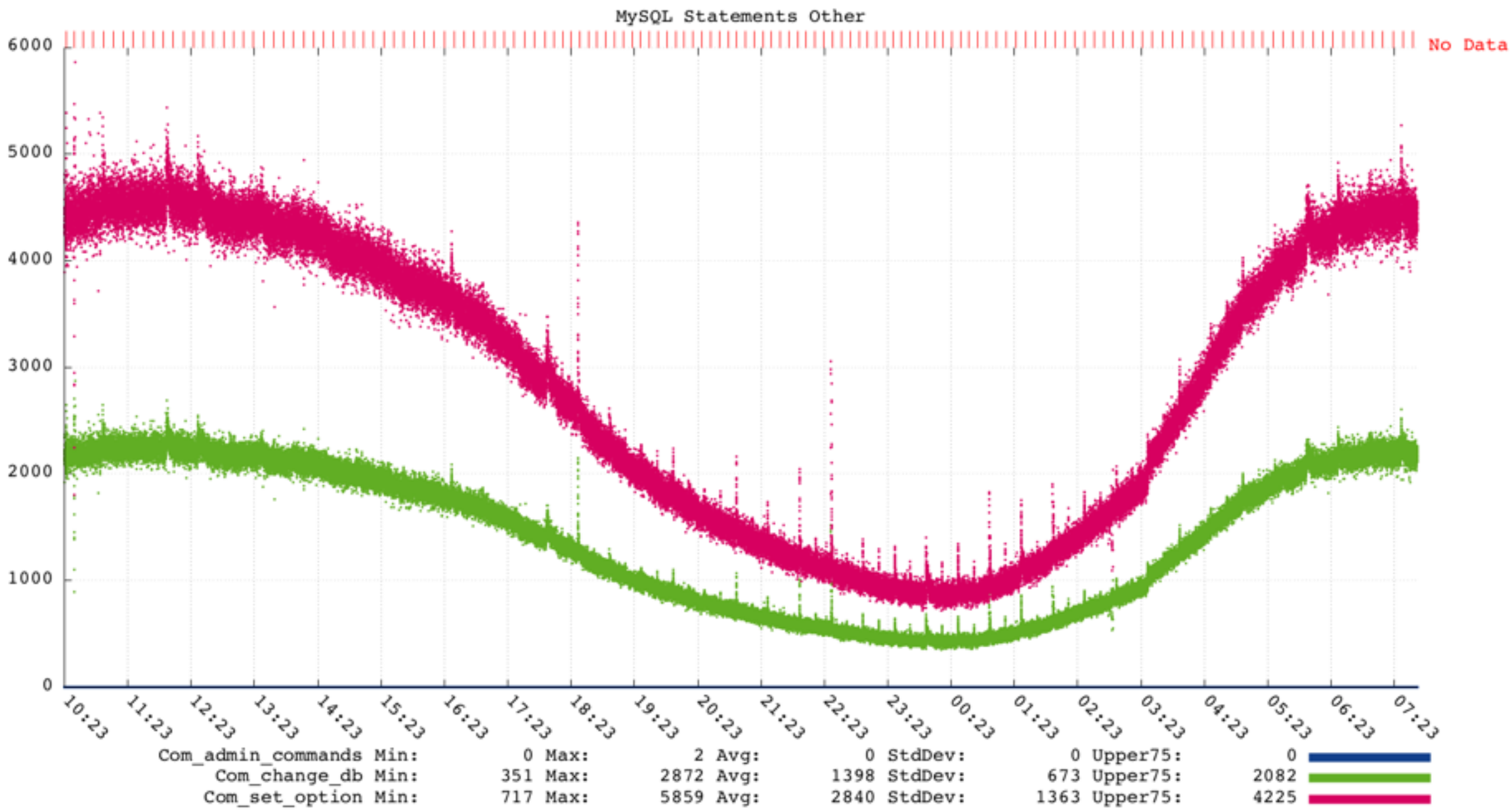

```
# Profile
# Rank Query ID           Response time   Calls   R/Call  Apdx V/M   Item
# ==== ================== =============== ======= ======= ==== ===== =====
#    1 0x171D5928F53168DB 2827.0382 39.7%  196669  0.0144 1.00  0.01 SELECT
                                                                     concre
                                                                     jsptem
#    2 0x5D51E5F01B88B79E 2241.7852 31.5%      93 24.1052 0.70 95.76 ADMIN CONNECT
#    3 0x813031B8BBC3B329  826.7319 11.6% 1421053  0.0006 1.00  1.37 COMMIT
#    4 0x16219655761820A2  749.7217 10.5% 1436267  0.0005 1.00  7.92 SELECT
#    5 0x3AEAAD0E15D725B5  311.9388  4.4% 1944762  0.0002 1.00  0.78 SET
#    6 0xEE4BE8C8FA45075D  114.2480  1.6%  205527  0.0006 1.00  0.01 SELECT pro
                                                                     concretete
                                                                     jsptemplat
#    7 0x348032401B50B8DE   28.2576  0.4%    3262  0.0087 1.00  0.01 ADMIN STMT_PREPARE
```

```
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000248  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
commit;
# Time: 140519 22:23:57.657887
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000047  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
SET autocommit=1;
# Time: 140519 22:23:57.658193
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000056  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
SELECT 1;
# Time: 140519 22:23:57.66424
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000056  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
SET autocommit=0;
# Time: 140519 22:23:57.667454
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000336  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
commit;
# Time: 140519 22:23:57.668208
# Client:                :55497
# Thread_id: 4294967298
# Query_time: 0.000085  Lock_time: 0.000000  Rows_sent: 0  Rows_examined: 0
SET autocommit=1;
```

Connection Pool Issues



Connection Pool Issues



Because of
'application bugs'

WHY DOES IT HAVE TO DO THAT?

Connection Pools - Why Chattiness Examples

- *maybe* forgot to `COMMIT / ROLLBACK`
- wanting `AUTOCOMMIT=1`
  but a previous TRX set it to `0`
- Changing TRX Isolation Level
- Is connection still working?

Connection Pools

Issues

Resetting Environment

Testing Connectivity

Pool Sizing

Lingering Transactions

Analysis

Examples

Graceful Failover

Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool - Resetting Status

	JDBC-Pool	C3P0	DBCP2	HikariCP
Rollback	rollbackOnReturn=false	autoCommitOnClose=false	rollbackOnReturn =true	
Commit	commitOnReturn=false	autoCommitOnClose=false	n/a	n/a
Avoid	see above	forceIgnoreUnresolvedTransactions=false	see above	
Auto Commit	Driver	Driver	enableAutoCommitOnReturn=true	Driver

http://ceilingcat.ninja

Connection Pools
Issues
Resetting Environment
Testing Connectivity
Pool Sizing
Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool - Testing

- Making sure the connection is still active
- If not, maybe reopen a connection
- Not recommended as DB
- However, applications:
  - do not like errors
  - do not retry gracefully

Connection Pool - Testing

- If connections REALLY need to be tested…
- do not specify test query like:
  - `SELECT 1`
  - `SELECT * FROM DUAL`
- Leave default, all of the connection pools use:
  JDBC4 `isValid();`

Connection Pool - Testing

	JDBC-Pool	C3P0	DBCP2	HikariCP
Test Before	testOnBorrow=false	testConnectionOnCheckOut =false	testOnBorrow=false	n/a
Test After	testOnReturn=false	testConnectionOnCheckIn =false	testOnReturn=false	n/a
Test While Idle	testWhileIdle=false	idleConnectionTestPeriod =0	testWhileIdle=false	n/a
JDBC4 isValid()	default	default	default	jdbc4ConnectionTest =true (default)
Query	validationQuery (isValid)	preferredTestQuery=null	validationQuery (isValid)	connectionTestQuery =none
Interval?	validationInterval=30000	n/a	n/a	n/a

Connection Pool - Testing

- JDBC: `validationInterval=30s`
  WHY? It defeats the whole purpose!

http://ceilingcat.ninja

Connection Pools

Issues

Resetting Environment

Analysis

Testing Connectivity

Examples

Pool Sizing

Graceful Failover

Lingering Transactions

Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool - Pool Sizing

- Funnelling on Application Level, is good
- Smaller Number is Better
  - +- * CPU's on DB
    - maybe a bit more (waiting on IO…)
  - all application servers combined
- Response Time vs Throughput

Connection Pool - Pool Sizing

	JDBC-Pool	C3P0	DBCP2	HikariCP
Amount of Connections	maxActive=100	maxPoolSize=15	maxTotal=8	maximumPoolSize=10
Maximum Idle Connections	maxIdle=100	maxIdleTime=0**	maxIdle=8	n/a
Minimum Idle Connections	minIdle=10	minPoolSize=3	minIdle=0	minimumIdle=max
Startup Size	initialSize=10	initialPoolSize=3	initialSize=0	minimumIdle

http://ceilingcat.ninja

Connection Pools
Issues
Resetting Environment
Testing Connectivity
Pool Sizing

Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool - Avoid Lingering Transactions

- Application forgets to return the connection
- Statements that take longer than …

- Avoid this!
- Fix Application

Connection Pool - Avoid Lingering Transactions

	KILL	Warning
JDBC-Pool	removeAbandoned=false removeAbandonedTimeout=60 abandonWhenPercentageFull=0	suspectTimeout=0
C3P0	unreturnedConnectionTimeout=0	n/a
DBCP	removeAbandoned=false removeAbandonedTimeout=300 Only When: getNumIdle() < 2 and getNumActive() > getMaxTotal() - 3)	n/a
HikariCP	n/a	leakDetectionThreshold=0

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pools - How To Look At Workload?

- Slow Query Log
- `tcpdump`
- `pt-query-digest`
- Percona Cloud Tools

http://ceilingcat.ninja

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pool - Example Transaction


```
Connection con = ds.getConnection();
con.setTransactionIsolation
  (Connection.TRANSACTION_READ_COMMITTED);
con.setAutoCommit(false);
PreparedStatement stmt =
con.prepareStatement("select * from
employees where emp_no = ?");
stmt.setInt(1, 20000);
ResultSet rs = stmt.executeQuery();
stmt.close();
rs.close();
con.commit();
con.close();
```

For Connectors - RECAP

- MySQL Connector/J
  - `useConfigs=maxPerformance`
  - `useServerPrepStmts=true`
- MariaDB Java Client
  - HikariCP: Built in
  - JDBC-Pool: `jdbcInterceptors="ConnectionState"`
  - Other Pools: UNKNOWN

Connection Pool - TRX JDBC


```
select * from employees
         where emp_no = 20000;

commit;
```

Connection Pool - TRX C3P0


```
SET SESSION TRANSACTION
   ISOLATION LEVEL READ COMMITTED;
SET autocommit=0;
select * from employees
            where emp_no = 20000;
commit;
SET autocommit=1;
SET SESSION TRANSACTION
   ISOLATION LEVEL REPEATABLE READ;
```

Connection Pool - TRX C3P0


```
mysql> set global
   tx_isolation="READ-COMMITTED";

forceIgnoreUnresolvedTransactions=true
```

Connection Pool - TRX DBCP


```
SET autocommit=1;
# administrator command: Ping;
SET autocommit=0;
select * from employees
          where emp_no = 20000;
commit;
rollback;
SET autocommit=1;
```

Connection Pool - TRX DBCP


```
testOnBorrow=false
rollbackOnReturn=false
enableAutoCommitOnReturn=false

jdbcUrl: useLocalTransactionState=true
```

Connection Pool - TRX HikariCP

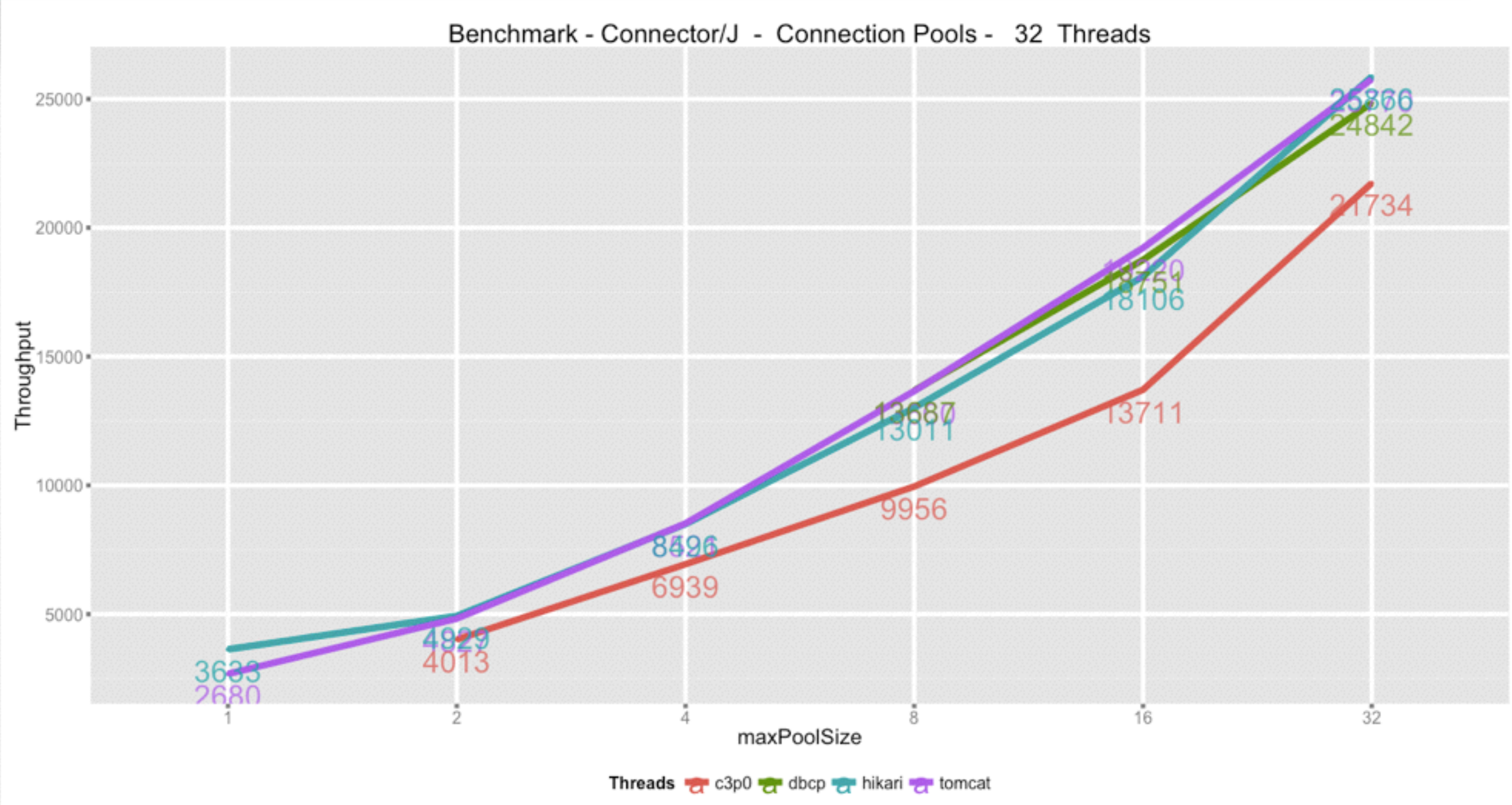

```
SET SESSION TRANSACTION
   ISOLATION LEVEL READ COMMITTED;
SET autocommit=0;
select * from employees
            where emp_no = 20000;
commit;
SET autocommit=1;
SET SESSION TRANSACTION
   ISOLATION LEVEL REPEATABLE READ;
```

Connection Pool - TRX HikariCP

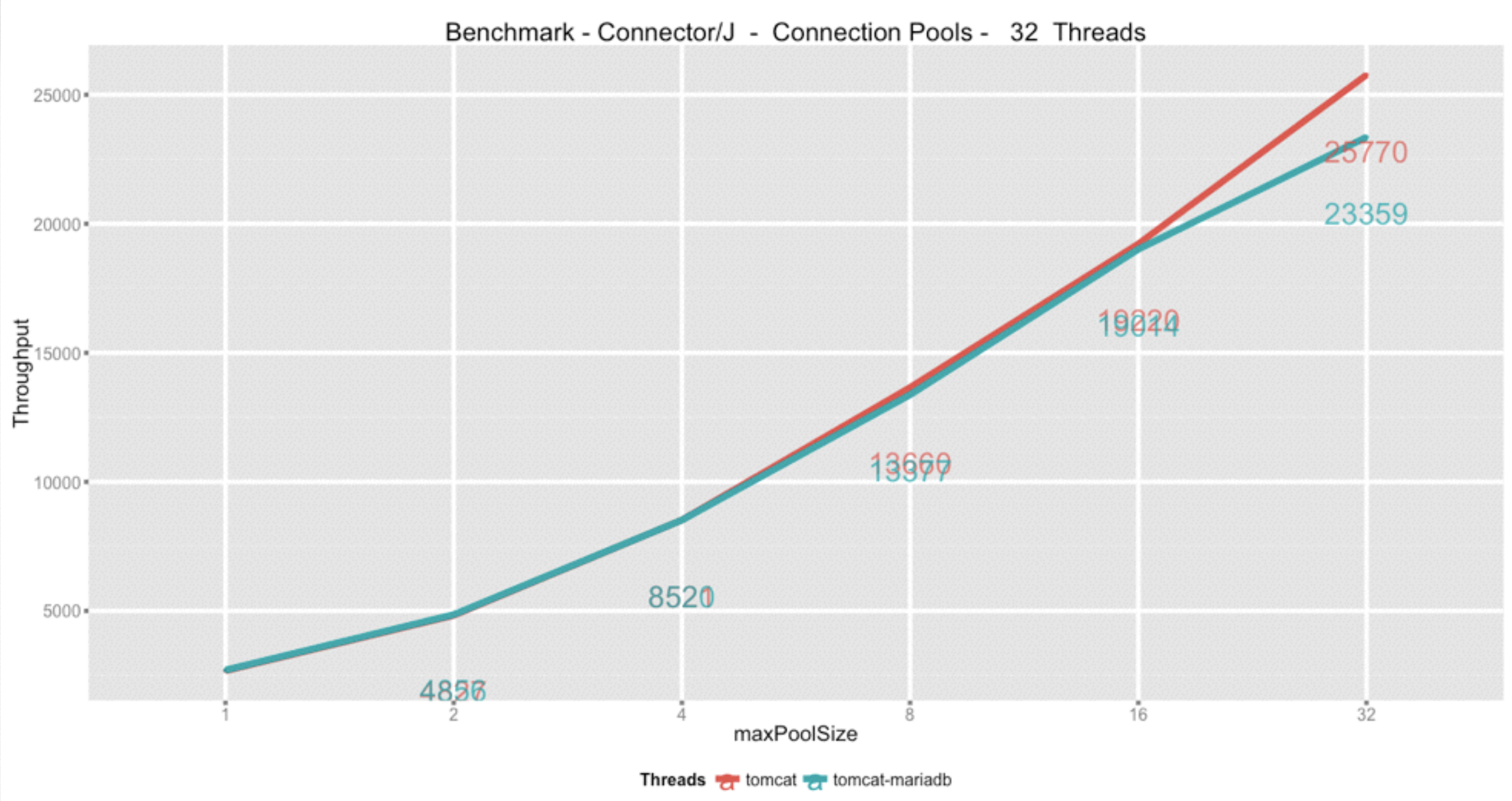

```
mysql> set global
   tx_isolation="READ-COMMITTED";

autoCommit=false
```

Connection Pools



MariaDB vs. Connector/J



http://ceilingcat.ninja

Connection Pools

Issues

Resetting Environment

Analysis

Testing Connectivity

Examples

Pool Sizing

Graceful Failover

Lingering Transactions

Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connection Pools - Graceful Failover

Statistics Report for HAProxy

n1:8088

HAProxy

Statistics Report for pid 10724

> General process information

pid = 10724 (process #1, nbproc = 1)
uptime = 0d 0h03m37s
system limits: memmax = unlimited; ulimit-n = 8207
maxsock = 8207; **maxconn** = 4096; **maxpipes** = 0
current conns = 16; current pipes = 0/0
Running tasks: 1/18

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: UP with load-balancing disabled is reported as "NOLB".

Display option:
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.4)
- Online manual

database

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	10	-	15	20	2 000	35		6 770	4 270	0	0	0					OPEN								
node1	0	0	-	0	10		15	20	-	35	35	6 770	4 270		0		0	0	0	0	1m33s UP	L4OK in 0ms	1	Y	-	0	1	1m42s	-
node2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	3m37s UP	L4OK in 0ms	1	-	Y	0	0	0s	-
Backend	0	0		0	10		15	20	2 000	35	35	6 770	4 270	0	0		0	0	0	0	3m37s UP		1	1	1		0	0s	

stats

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				2	3	-	1	1	2 000	56		18 810	578 247	0	0	0					OPEN								
Backend	0	0		0	0		0	0	2 000	0	0	18 810	578 247	0	0		0	0	0	0	3m37s UP		0	0	0		0		

Connection Pools - Graceful Failover

- HAProxy ‘stats socket’

```
/etc/haproxy/haproxy.cfg
global
  . . .
stats socket /tmp/haproxy.sock level admin
```

- Disable Node

```
# echo "disable server database/node1"
        | socat stdio /tmp/haproxy.sock
```

Connection Pools - Graceful Failover

Statistics Report for HAProxy

n1:8088

n1:808... | n1:808... | n1:808... | n1:808... | 201110... | r - cha... | theme.... | vagrant... | Statisti...

HAProxy

Statistics Report for pid 10724

> General process information

pid = 10724 (process #1, nbproc = 1)
uptime = 0d 0h04m20s
system limits: memmax = unlimited; ulimit-n = 8207
maxsock = 8207; **maxconn** = 4096; **maxpipes** = 0
current conns = 16; current pipes = 0/0
Running tasks: 1/18

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: UP with load-balancing disabled is reported as "NOLB".

Display option:
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.4)
- Online manual

database

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	10	-	15	20	2 000	35		6 770	4 270	0	0	0					OPEN								
node1	0	0	-	0	10		15	20	-	35	35	6 770	4 270		0		0	0	0	0	7s MAINT	L4OK in 0ms	1	Y	-	0	2	1m49s	-
node2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	4m20s UP	L4OK in 0ms	1	-	Y	0	0	0s	-
Backend	0	0		0	10		15	20	2 000	35	35	6 770	4 270	0	0		0	0	0	0	4m20s UP		1	0	1		0	0s	

stats

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	3	-	1	1	2 000	57		19 152	588 805	0	0	0					OPEN								
Backend	0	0		0	0		0	0	2 000	0	0	19 152	588 805	0	0		0	0	0	0	4m20s UP		0	0	0		0		

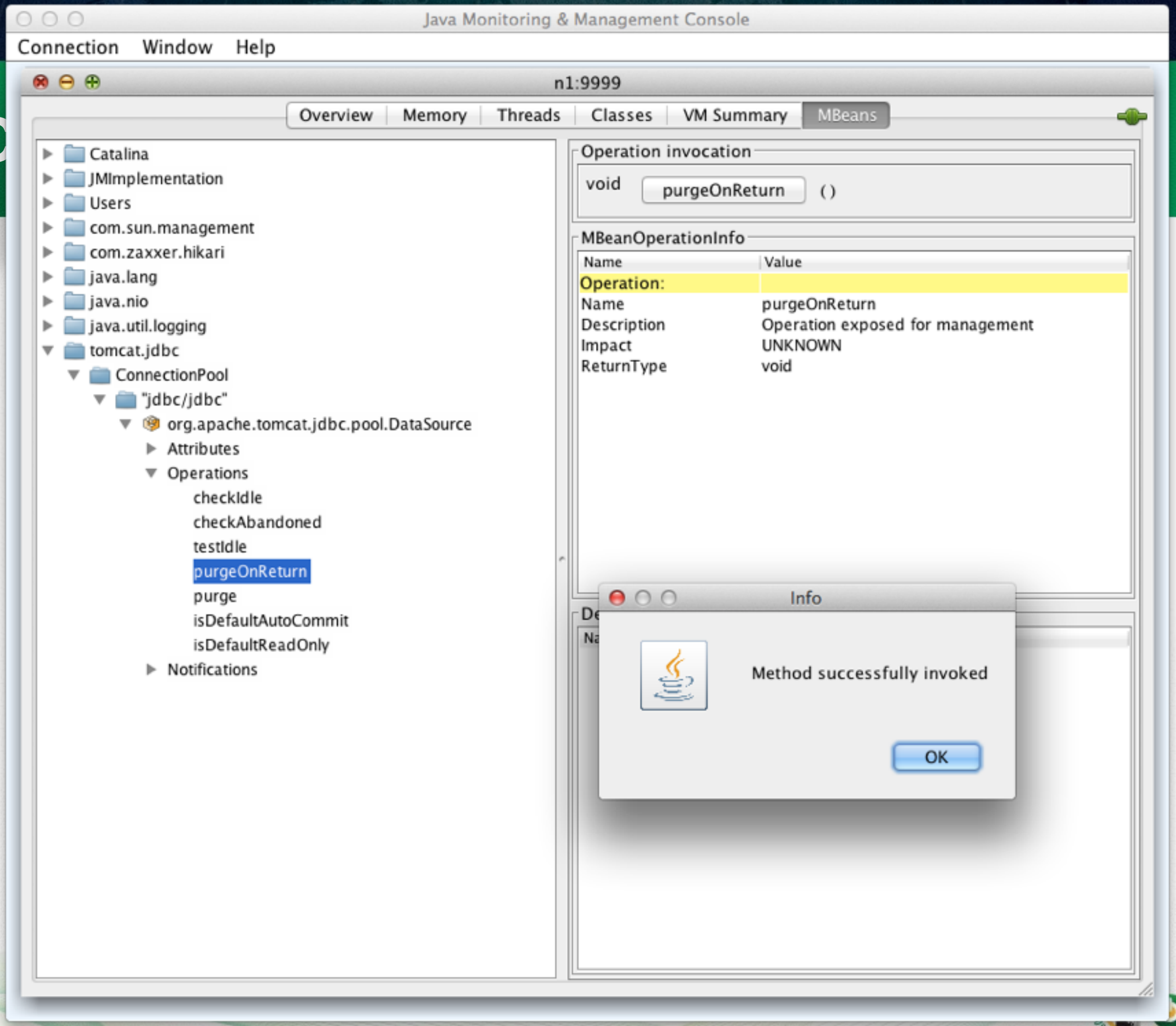
Connection Pools - Graceful Failover

- During ‘maintenance’, what do we do?
  - `KILL` old connections?
  - Wait until connections are closed? (Define lifetimes?)
  - Ignore it?

Connection Pools - Graceful Failover

- Some connection pools can close connections gracefully, when idle.
  - For ‘synchronous’ replication systems
  - using JMX
  - No Application Errors!

	Method
JDBC-Pool	purgeOnReturn()
C3P0	softResetAllUsers()
DBCP	n/a
HikariCP	softEvictConnections(), suspendPool(), resumePool() <—- ASYNC



Java Monitoring & Management Console

Connection Window Help

n1:9999

Overview | Memory | Threads | Classes | VM Summary | MBeans

- Catalina
- JMImplementation
- Users
- com.mchange.v2.c3p0
  - C3P0Registry
  - PooledDataSource
    - f2f5sy95xvo5q729r712|7750e391
      - f2f5sy95xvo5q729r712|7750e391
        - Attributes
        - Operations
          - close
          - getEffectivePropertyCycle
          - getNumBusyConnections
          - getNumConnections
          - getNumIdleConnections
          - getNumThreadsAwaitingCheckout
          - getNumUnclosedOrphanedConnections
          - getStatementCacheNumCheckedOut
          - getStatementCacheNumConnectionsWithCache
          - getStatementCacheNumStatements
          - getStatementDestroyerNumConnectionsInUse
          - getStatementDestroyerNumConnectionsWithDe
          - getStatementDestroyerNumDeferredDestroySt
          - hardReset
          - isWrapperFor
          - sampleLastAcquisitionFailureStackTrace
          - sampleLastAcquisitionFailureStackTraceDefaul
          - sampleLastCheckinFailureStackTrace
          - sampleLastCheckinFailureStackTraceDefaultUs
          - sampleLastCheckoutFailureStackTrace
          - sampleLastCheckoutFailureStackTraceDefaultU
          - sampleLastConnectionTestFailureStackTrace
          - sampleLastConnectionTestFailureStackTraceD

Operation invocation

void softResetAllUsers ()

MBeanOperationInfo

Name	Value
Operation:	
Name	softResetAllUsers
Description	
Impact	ACTION
ReturnType	void

Info

Method successfully invoked

OK

Java Monitoring & Management Console

Connection Window Help

n1:9999

Overview | Memory | Threads | Classes | VM Summary | MBeans

- Catalina
- JMImplementation
- Users
- com.sun.management
- com.zaxxer.hikari
  - Pool (HikariPool-0)
    - Attributes
    - Operations
      - closeIdleConnections
      - dumpPoolState
  - PoolConfig (HikariPool-0)
- java.lang
- java.nio
- java.util.logging
- tomcat.jdbc

Operation invocation

void closeIdleConnections ()

MBeanOperationInfo

Name	Value
Operation:	
Name	closeIdleConnections
Description	Operation exposed for management
Impact	UNKNOWN
ReturnType	void

Descriptor

Name	Value

Info

Method successfully invoked

OK

Connection Pools - Graceful Failover

- 0 Application Errors
- Completely seamless

Connection Pools

Issues

Resetting Environment

Testing Connectivity

Pool Sizing

Lingering Transactions

Analysis

Examples

Graceful Failover

Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS

Connector/J - Extra Features

- Load Balancing
  - `jdbcUrl: "jdbc:mysql:loadbalance://node1,node2/db?loadBalanceConnectionGroup=lb&loadBalanceEnableJMX=true"`
  - `loadBalanceStrategy (random/bestResponseTime)`
  - Failover
  - ReplicationDriver (`setReadOnly`)
  - Combining with Connection Pools is less useful
- Fabric

Java MySQL Connector & Connection Pool Optimization

- http://dev.mysql.com/doc/connector-j/en
- https://mariadb.com/kb/en/mariadb/client-libraries/mariadb-java-client/
- http://tomcat.apache.org/tomcat-7.0-doc/jdbc-pool.html
- http://www.mchange.com/projects/c3p0
- http://commons.apache.org/proper/commons-dbcp/
- https://github.com/brettwooldridge/HikariCP

MYSQL CONNECTORS CONNECTION POOLS

Kenny Gryp
<kenny.gryp@percona.com>
November 4, 2014
@gryp