Java MySQL Connector & Connection Pool

Features & Optimization

Kenny Gryp <kenny.gryp@percona.com>
November 4, 2014
@gryp
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
  – 5.1.33 Latest
  – Compatible with
    • MySQL
    • Percona Server
    • MariaDB
MySQL Connector/J & MariaDB Java Client

- MariaDB Java Client
  - MariaDB
  - Fork from Drizzle Connector
  - Latest 1.1.7
  - Compatible with
    - MySQL
    - Percona Server
    - MariaDB
MySQL Connector/J Features

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

callback.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&...
Connectors
Configuring Connector
Creating A Database Connection
Prepared Statements
Example Transaction

MYSQL CONNECTORS
CONNECTION POOLS
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';
set autocommit=1;
USE employees;
show variables like 'sql_mode';
Creating Connection - Defaults

- Connector/J:

- MariaDB Java Client:
• 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

QPS

16000
13.477
12000
11.477
10000
9.477
8000
7.477
6000
5.477
4000
15.213
3000
2000
1000
0

localhost

ConnectorJ

MariaDB
Connector Performance - MariaDB %faster

MariaDB Connector  +Speed% ConnectorJ

SELECT1 LO

13%
Connector Performance - MariaDB %faster

Faster %

<table>
<thead>
<tr>
<th></th>
<th>Faster %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT1 LO</td>
<td>13%</td>
</tr>
<tr>
<td>SELECT1 net</td>
<td>4%</td>
</tr>
<tr>
<td>pk</td>
<td>4%</td>
</tr>
<tr>
<td>range</td>
<td>0%</td>
</tr>
</tbody>
</table>

MariaDB Connector +Speed% ConnectorJ
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
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"
```
Connectors
Configuring Connector
Creating A Database
Connection
Prepared Statements

Example Transaction

MYSQL CONNECTORS
CONNECTION POOLS
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();
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;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;

select * from employees
   where emp_no = 20000;

COMMIT;
if
(
  con.getTransactionIsolation() !=
  Connection.TRANSACTION_READ_COMMITTED
)
{
  con.setTransactionIsolation
    (Connection.TRANSACTION_READ_COMMITTED);
}
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"/>
select * from employees
    where emp_no = 20000;
COMMIT;
Benchmark - Connector Concurrency - SELECT 1

HikariCP-bench with JDBC Pool, 4 Threads, SELECT 1 (4,8,16,32 Pool Size)
Benchmark - Connector Concurrency - TRX

HikariCP-bench with JDBC Pool, 4 Threads, TRX (4,8,16,32 Pool Size)

QPS

Conn/J  | MariaDB | Conn/J Optim | MariaDB Optim
--- | --- | --- | ---
9.500  | 14.250 | 19.000 | 19.000
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
Connection Pools

Issues
Resetting Environment
Testing Connectivity
Pool Sizing

Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS
## 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.

<table>
<thead>
<tr>
<th>#</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>Query ID</td>
</tr>
<tr>
<td>1</td>
<td>0x813031B8BBC3B329</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>0x3AEAAD0E15D725B5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>0x16219655761820A2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>0x943798A09019B333</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>0x19C8068B5C1997CD</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Query ID</th>
<th>Response time</th>
<th>Calls</th>
<th>R/Call</th>
<th>Apdx</th>
<th>V/M</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x171D5928F53168DB</td>
<td>2827.0382</td>
<td>196669</td>
<td>0.0144</td>
<td>1.00</td>
<td>0.01</td>
<td>SELECT concretexsptem</td>
</tr>
<tr>
<td>2</td>
<td>0x5D51E5F01B8B8B79E</td>
<td>2241.7852</td>
<td>93</td>
<td>24.1052</td>
<td>0.70</td>
<td>95.76</td>
<td>ADMIN CONNECT</td>
</tr>
<tr>
<td>3</td>
<td>0x813031B8BBC3B329</td>
<td>826.7319</td>
<td>1421053</td>
<td>0.0006</td>
<td>1.00</td>
<td>1.37</td>
<td>COMMIT</td>
</tr>
<tr>
<td>4</td>
<td>0x16219655761820A2</td>
<td>749.7217</td>
<td>1436267</td>
<td>0.0005</td>
<td>1.00</td>
<td>7.92</td>
<td>SELECT</td>
</tr>
<tr>
<td>5</td>
<td>0x3AEAAD0E15D725B5</td>
<td>311.9388</td>
<td>1944762</td>
<td>0.0002</td>
<td>1.00</td>
<td>0.78</td>
<td>SET</td>
</tr>
<tr>
<td>6</td>
<td>0xEE4BE8C8FA45075D</td>
<td>114.2480</td>
<td>205527</td>
<td>0.0006</td>
<td>1.00</td>
<td>0.01</td>
<td>SELECT proconcretesptemplat</td>
</tr>
<tr>
<td>7</td>
<td>0x348032401B50B8DE</td>
<td>28.2576</td>
<td>3262</td>
<td>0.0087</td>
<td>1.00</td>
<td>0.01</td>
<td>ADMIN STMT_PREPARE</td>
</tr>
</tbody>
</table>
# 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

MySQL Statements Other

Com_admin_commands Min: 0 Max: 1 Avg: 0 StdDev: 0 Upper75: 0
Com_change_db Min: 0 Max: 218 Avg: 0 StdDev: 1 Upper75: 0
Com_set_option Min: 0 Max: 2848 Avg: 780 StdDev: 471 Upper75: 1150
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

<table>
<thead>
<tr>
<th>JDBC-Pool</th>
<th>C3P0</th>
<th>DBCP2</th>
<th>HikariCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rollback</td>
<td>rollbackOnReturn=false</td>
<td>autoCommitOnClose=false</td>
<td>rollbackOnReturn=true</td>
</tr>
<tr>
<td>Commit</td>
<td>commitOnReturn=false</td>
<td>autoCommitOnClose=false</td>
<td>n/a</td>
</tr>
<tr>
<td>Avoid</td>
<td>see above</td>
<td>forceIgnoreUnresolvedTransactions=false</td>
<td>see above</td>
</tr>
<tr>
<td>Auto Commit</td>
<td>Driver</td>
<td>Driver</td>
<td>enableAutoCommitOnReturn=true</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>JDBC-Pool</th>
<th>C3P0</th>
<th>DBCP2</th>
<th>HikariCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Before</strong></td>
<td>testOnBorrow=false</td>
<td>testConnectionOnCheckOut=false</td>
<td>testOnBorrow=false</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>Test After</strong></td>
<td>testOnReturn=false</td>
<td>testConnectionOnCheckIn=false</td>
<td>testOnReturn=false</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>Test While Idle</strong></td>
<td>testWhileIdle=false</td>
<td>idleConnectionTestPeriod=0</td>
<td>testWhileIdle=false</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>JDBC4 isValid()</strong></td>
<td>default</td>
<td>default</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>validationQuery</td>
<td>preferredTestQuery=null</td>
<td>validationQuery</td>
<td>connectionTestQuery=none</td>
</tr>
<tr>
<td><strong>Interval?</strong></td>
<td>validationInterval=30000</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Connection Pool - Testing

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

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

<table>
<thead>
<tr>
<th></th>
<th>KILL</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC-Pool</td>
<td>removeAbandoned=false removeAbandonedTimeout=60 abandonWhenPercentageFull=0</td>
<td>suspectTimeout=0</td>
</tr>
<tr>
<td>C3P0</td>
<td>unreturnedConnectionTimeout=0</td>
<td>n/a</td>
</tr>
<tr>
<td>DBCP</td>
<td>removeAbandoned=false removeAbandonedTimeout=300</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>Only When: getNumIdle() &lt; 2 and getNumActive() &gt; getMaxTotal() - 3</td>
<td></td>
</tr>
<tr>
<td>HikariCP</td>
<td>n/a</td>
<td>leakDetectionThreshold=0</td>
</tr>
</tbody>
</table>
Connection Pools
Issues
Resetting Environment
Testing Connectivity
Pool Sizing

Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MYSQL CONNECTORS
CONNECTION POOLS
Connection Pools - How To Look At Workload?

- Slow Query Log
- tcpdump
- pt-query-digest
- Percona Cloud Tools
Connection Pools
Issues
Resetting Environment
Testing Connectivity
Pool Sizing

Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MYSQL CONNECTORS

CONNECTION POOLS
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
  - JDBC-Pool:
    jdbcInterceptors="ConnectionState"
- Other Pools: UNKNOWN
select * from employees
    where emp_no = 20000;

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

forceIgnoreUnresolvedTransactions=true
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

jdbcTemplate: useLocalTransactionState=true
SET SESSION TRANSACTION
  ISOLATION LEVEL READ COMMITTED;
SET autocommit=0;
select * from employees
  where emp_no = 20000;
commit;
rollback;
SET autocommit=1;
SET SESSION TRANSACTION
  ISOLATION LEVEL REPEATABLE READ;
mysql> set global
    tx_isolation="READ-COMMITTED";

autoCommit=false

--> will always do rollback!!!!!
Connection Pools

Benchmark - Connector/J - Connection Pools - 32 Threads

Throughput vs. maxPoolSize graph showing performance for different connection pools with 32 threads. The graph compares the throughput for different maxPoolSize values for c3p0, dbcp, hikari, and tomcat connection pools.
MariaDB vs. Connector/J
Connection Pools
Issues
Resetting Environment
Testing Connectivity
Pool Sizing

Lingering Transactions
Analysis
Examples
Graceful Failover
Conn/J Extra Features

MySQL Connectors
Connection Pools
**HAPerf**,  **Statistics Report for pid 10724**

> **General process information**

```
pid = 10724 (process #1, nbproc = 1)
uptime = 0d 0h 02m 37s
system limits: memmax = unlimited; ulimit-n = 8207
maxsock = 8207; maxconn = 4096; maxpipes = 0
current conns = 16; current pipes = 0
Running tasks: 1/18
```

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

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

---

### Database

<table>
<thead>
<tr>
<th>Queue</th>
<th>Session rate</th>
<th>Sessions</th>
<th>Bytes</th>
<th>Denied</th>
<th>Errors</th>
<th>Warnings</th>
<th>Status</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cur</td>
<td>Max</td>
<td>Limit</td>
<td>Cur</td>
<td>Max</td>
<td>Limit</td>
<td>Total</td>
<td>LbTot</td>
</tr>
<tr>
<td>Frontend</td>
<td>0</td>
<td>10</td>
<td>-15</td>
<td>20</td>
<td>2000</td>
<td>35</td>
<td>6770</td>
<td>4270</td>
</tr>
<tr>
<td>node1</td>
<td>0</td>
<td>0</td>
<td>-10</td>
<td>15</td>
<td>20</td>
<td>-35</td>
<td>35</td>
<td>6770</td>
</tr>
<tr>
<td>node2</td>
<td>0</td>
<td>0</td>
<td>-10</td>
<td>15</td>
<td>20</td>
<td>-35</td>
<td>35</td>
<td>6770</td>
</tr>
<tr>
<td>Backend</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

### Stats

<table>
<thead>
<tr>
<th>Queue</th>
<th>Session rate</th>
<th>Sessions</th>
<th>Bytes</th>
<th>Denied</th>
<th>Errors</th>
<th>Warnings</th>
<th>Status</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cur</td>
<td>Max</td>
<td>Limit</td>
<td>Cur</td>
<td>Max</td>
<td>Limit</td>
<td>Total</td>
<td>LbTot</td>
</tr>
<tr>
<td>Frontend</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
<td>56</td>
<td>18810</td>
</tr>
<tr>
<td>Backend</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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
## HAProxy

### Statistics Report for pid 10724

#### General process information

pid = 10724 (process #1, nbproc = 1)
up_time = 0d 0h 0m 4s
system limits: memmax = unlimited; ulimit-n = 8207
maxsock = 8207; maxconn = 4096; maxpipes = 0
current conn = 16; current pipes = 0
Running tasks: 1/18

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

<table>
<thead>
<tr>
<th>Database</th>
<th>Queue</th>
<th>Session rate</th>
<th>Sessions</th>
<th>Bytes</th>
<th>Denied</th>
<th>Errors</th>
<th>Warnings</th>
<th>Status</th>
<th>LastChk</th>
<th>Wght</th>
<th>Act</th>
<th>Bck</th>
<th>Chk</th>
<th>Dwn</th>
<th>Dwntme</th>
<th>Thrtle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frontend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>node1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>2000</td>
<td>35</td>
<td>6770</td>
<td>4270</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>node2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6770</td>
<td>4270</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Backend</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6770</td>
<td>4270</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Stats

<table>
<thead>
<tr>
<th>Queue</th>
<th>Session rate</th>
<th>Sessions</th>
<th>Bytes</th>
<th>Denied</th>
<th>Errors</th>
<th>Warnings</th>
<th>Status</th>
<th>LastChk</th>
<th>Wght</th>
<th>Act</th>
<th>Bck</th>
<th>Chk</th>
<th>Dwn</th>
<th>Dwntme</th>
<th>Thrtle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

External resources:
- Primary site
- Updates (v1.4)
- Online manual
Connection Pools - Graceful Failover

• During ‘maintenance’, what do we do?
  • **KILL** old connections?
  • Wait until connections are closed? (Define lifetimes?)
  • Ignore it?
Some connection pools can close connections gracefully, when idle.

- For ‘synchronous’ replication systems
- using JMX
- No Application Errors!

<table>
<thead>
<tr>
<th>Library</th>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC-Pool</td>
<td>purgeOnReturn()</td>
<td></td>
</tr>
<tr>
<td>C3P0</td>
<td>softResetAllUsers()</td>
<td></td>
</tr>
<tr>
<td>DBCP</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>HikariCP</td>
<td>closeIdleConnections()</td>
<td>Only closes current idle connections</td>
</tr>
</tbody>
</table>
### MBeanOperationInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation:</td>
<td>purgeOnReturn</td>
</tr>
<tr>
<td>Description</td>
<td>Operation exposed for management</td>
</tr>
<tr>
<td>Impact</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>ReturnType</td>
<td>void</td>
</tr>
</tbody>
</table>

**Method successfully invoked**

```java
void purgeOnReturn() {
}
```
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
Victory!

@leboros 12h
Java is nice technology, @gryn will talk about it at pluk. Absolutely recommended.
Java MySQL Connector & Connection Pool Optimization

- 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