DRIVOLUTION: RETHINKING THE DATABASE DRIVER LIFECYCLE

Emmanuel Cecchet (UMass Amherst)
Joint work with George Candea (EPFL)
THIS RESEARCH IS BASED ON TRUE EVENTS

APPROPRIATE FOR BOTH ACADEMIC AND INDUSTRIAL AUDIENCES
DATABASE DRIVERS

MySQL drivers: 63 platforms, 14+ languages

Java Application
- Connector/J
- JDBC Driver
- JVM

PHP Application
- libmysl
- native lib
- Connector/ODBC

.Net Application
- Connector/Net

Python
- MySQLdb DB-API

Perl script
- DBD::mysql

mysql network protocol
DATABASE DRIVER LIFECYCLE

1) Get an appropriate driver package from vendor
2) Install the driver on the client application machine
3) Configure the client application to use the driver
4) Start the application and load the database driver
5) Connect to database and check protocol compatibility
6) Authenticate
7) Execute requests

Driver update:
8) Stop the application
9) Uninstall old driver
10) Repeat all steps 1 through 7
STATE OF THE ART IN DATABASE DRIVERS

- Drivers are not sexy...
- Driver distribution separate from database engine
- Driver installation manual process on client machine
- Driver upgrades are disruptive (client application reconfiguration + restart)
- No protection against malicious drivers
A REAL LIFE EXAMPLE

- 200 application servers accessing a cluster of 4 DBs
- Driver upgrade more complex than database upgrade
- Online upgrades?
DRIVOLUTION

Think different

• Concepts
• Implementation
• Use cases
DRIVOLUTION CONCEPTS

- 2 Components: Bootloader and Server
- Drivers are stored in the database information schema
- Generic bootstrap receives driver a la DHCP from Drivolution server
- Lease associated to driver with various renewal policies

![Diagram of Drivolution Concepts]

<table>
<thead>
<tr>
<th>API</th>
<th>API version</th>
<th>platform</th>
<th>binary format</th>
<th>binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC</td>
<td>4.0</td>
<td>JRE 1.6</td>
<td>jar</td>
<td>Driver 1</td>
</tr>
<tr>
<td>ODBC</td>
<td>3.0</td>
<td>linux_i386</td>
<td>zip</td>
<td>Driver 2</td>
</tr>
<tr>
<td>ODBC</td>
<td>3.0</td>
<td>ADO.Net</td>
<td>zip</td>
<td>Driver 3</td>
</tr>
</tbody>
</table>
DRIVOLUTION CONCEPTS

Application

Drivolution Bootloader

send(db, user, API, platform)

find_available_driver(API, platform)
send(lease, driver_bin_format)

DRIVOLUTION_REQUEST

TFTP or secure transfer

send(driver_code)

update lease table
set (user, client@, driver id, lease)

DRIVOLUTION_OFFER

Upgrade driver
update lease table
send(driver_code)

 DRIVOLUTION_REQUEST

DRIVOLUTION_OFFER

set(lease_expire_time, expire_policy)
download(driver_code)
load(driver)
connect()

Lease expires

Install new driver
Switch according to policy
DRIVOLUTION

Think different

- Concepts
- Implementation
- Use cases
DRIVOLUTION BOOTLOADER

- Database driver agnostic but API[/Platform] specific
- Implements the Drivolution protocol
- Can load multiple drivers and switch from 1 version to the other
- Does not need to be upgraded
DRIVOLUTION SERVER

- Server logic uses regular SQL
  - re-uses all existing database mechanisms
  - logic can be implemented as a stored procedure
- In-core implementation
  - Drivolution server embedded in the database
  - can enforce connection closing on server side upon lease expiration
- External server
  - Server runs outside the DB
  - uses legacy driver to access DB
- Standalone server
  - database independent service
  - generic driver distribution server
DRIVOLUTION IN A JAVA WORLD

- Bootloader can be integrated in JVM runtime
- Very small footprint (8 KB)
- Drivers loaded in separate classloaders
- No runtime overhead after 1\textsuperscript{st} connect (original driver code)
DRIVOLUTION

Think different

- Concepts
- Implementation
- Use cases
**DRIVOLUTION USE CASES: HETEROGENEOUS DATABASE ADMIN**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Current State-of-the-Art</th>
<th>Drivolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing a new database</td>
<td>1. Download drivers for DBA&lt;sub&gt;1&lt;/sub&gt; platform&lt;br&gt;2. Configure DBA&lt;sub&gt;1&lt;/sub&gt; console to find driver&lt;br&gt;3. DBA&lt;sub&gt;1&lt;/sub&gt; connects to db&lt;br&gt;4. Download drivers for DBA&lt;sub&gt;2&lt;/sub&gt; platform&lt;br&gt;5. Configure DBA&lt;sub&gt;2&lt;/sub&gt; console to find driver&lt;br&gt;6. DBA&lt;sub&gt;2&lt;/sub&gt; connects to db</td>
<td>1. DBA&lt;sub&gt;1&lt;/sub&gt; connects to db&lt;br&gt;2. DBA&lt;sub&gt;2&lt;/sub&gt; connects to db</td>
</tr>
<tr>
<td>Database driver upgrade</td>
<td>1. Copy right driver for DBA&lt;sub&gt;1&lt;/sub&gt; platform&lt;br&gt;2. Remove DBA&lt;sub&gt;1&lt;/sub&gt; old driver&lt;br&gt;3. Restart DBA&lt;sub&gt;1&lt;/sub&gt; console&lt;br&gt;4. Copy right driver for DBA&lt;sub&gt;2&lt;/sub&gt; platform&lt;br&gt;5. Remove DBA&lt;sub&gt;2&lt;/sub&gt; old driver&lt;br&gt;6. Restart DBA&lt;sub&gt;2&lt;/sub&gt; console</td>
<td>1. Insert drivers in database&lt;br&gt;2. Revoke old driver</td>
</tr>
</tbody>
</table>
DRIVOLUTION USE CASES: DYNAMIC RECONFIGURATION OF CLIENTS

- Master/slave failover for planned maintenance
- 0-downtime operation
- Failback = downgrade

Diagram:

1. Application
2. Drivolution Bootloader
3. DB\textsubscript{master} driver
4. DB\textsubscript{slave} driver
5. DB\textsubscript{master}
6. DB\textsubscript{slave}
7. DB\textsubscript{m} driver
8. DB\textsubscript{s} driver
DRIVOLUTION USE CASES: DATABASE CLUSTERS

- Drivolution server as a standalone driver distribution service
- Useful for both legacy and homogeneous systems
- Can be replicated for HA
DRIVOLUTION: OTHER USE CASES

- Assembling drivers on-demand
  - extensions on-demand (NLS, GIS, Kerberos...)
  - driver code signing to identify malicious drivers
  - dynamic generation of preconfigured drivers

- License server
  - license keys in separate files
  - static or dynamic distribution

- Integration of Bootloaders in runtime libraries
WHAT IS THE MESSAGE TO TAKE BACK HOME?
CONCLUSION

- Drivers are not sexy… but they deserve our attention

https://sourceforge.net/projects/drivolution/

- More open issues in Sigmod’08 paper

  MIDDLEWARE-BASED DATABASE REPLICATION: THE GAPS BETWEEN THEORY AND PRACTICE
QUESTIONS?

Contact: cecchet@cs.umass.edu