C-JDBC: a High Performance Database Clustering Middleware

Nicolas Modrzyk
Nicolas.Modrzyk@inrialpes.fr
Motivations

Use-Cases

C-JDBC concepts

Performance

Monitoring

Community

Conclusion
Motivations

J2EE performance scalability bounded by database performance

Database tier must be

- scalable
- fault tolerant (high availability + failover)
- without modifying the client application
- using open source databases
- on commodity hardware
What is C-JDBC Controller

Scalability - Fault tolerance - Failover - Monitoring - Caching - Logging - ...

Database JDBC driver

MySQL, PostgreSQL, Oracle, DB2, InstantDB, ...

Servlet container

Tomcat, Jetty, ...

EJB Container

JOnAS, WebLogic, JBoss, WebSphere, ...

Java client program

C-JDBC driver

JVM

No scalability
No fault tolerance
No failover

Database

MySQL, PostgreSQL, Oracle, DB2, InstantDB, ...

Java client program

C-JDBC driver

JVM

Servlet container

Tomcat, Jetty, ...

EJB Container

JOnAS, WebLogic, JBoss, WebSphere, ...

Java client program

C-JDBC driver

JVM
Redundant Array of Inexpensive Databases

**RAIDb controller**
- gives the view of a single database to the client
- balance the load on the database backends

**RAIDb levels**
- RAIDb-0: full partitioning
- RAIDb-1: full mirroring
- RAIDb-2: partial replication
- composition possible
Middleware implementing RAIDb

Two components

- generic JDBC 2.0 driver (C-JDBC driver)
- C-JDBC Controller

C-JDBC Controller provides

- performance scalability
- high availability
- failover
- caching, logging, monitoring, …

Supports heterogeneous databases
Outline - Use-Cases

Motivations

Use-Cases

C-JDBC concepts

Performance

Monitoring

Community

Conclusion
What C-JDBC offers

Usually, we do this:

Applicatio
n
JDBC
Driver

Database
What C-JDBC offers

Now we have this:

Applicatio

Virtual Database

Controller

Driver

Cache

Pooling

Backend

Driver

JDBC

Controller

http://c-jdbc.objectweb.org/ - c-jjdbc@objectweb.org
What C-JDBC offers

And, finally we have all this:

- Backend recovery
- Backend
- Cache Pooling
- Load balancing
- JDBC Driver
- Controller
- Virtual Database
- Driver
- Application
application already written for a specific [commercial] database user defined rules for on-the-fly query rewriting to execute on heterogeneous backends

Heterogeneity support

C-JDBC Controller
RAIDb-2

C-JDBC driver

JVM

Java client program

Servlet container
Tomcat, Jetty, ...

C-JDBC driver

JVM

EJB Container
JOnAS, WebLogic, JBoss, WebSphere, ...

Oracle JDBC driver

MySQL JDBC driver

Oracle

MySQL

MySQL

MySQL
Outline - C-JDBC concepts

Motivations
Use-Cases
C-JDBC concepts
Performance
Monitoring
Community
Conclusion
Controller

C-JDBC Controller

Virtual database

Authentication Manager

Request Manager

Scheduler

Recovery Log

Request Cache

Load balancer

Database Backend

Connection Manager

MySQL JDBC driver

Oracle JDBC driver

MySQL

MySQL

MySQL

Oracle

Oracle

Configuration & administration

XML engine

XML configuration file

JMX

Java client program
(Servlet, EJB, ...)

C-JDBC driver

Sockets

Sockets
Virtual Database

gives the view of a single database
establishes the mapping between the database name used by the application and the backend specific settings
backends can be added and removed dynamically configured using an XML configuration file
Building the initial state

- Octopus is an ETL tool
- Use Octopus to store a dump of the initial database state

Octopus → C-JDBC Controller

C-JDBC Controller → PostgreSQL JDBC driver

PostgreSQL JDBC driver → PostgreSQL

disable

dump for initial checkpoint

EJB Container
JOnAS, WebLogic, JBoss, WebSphere, ...

C-JDBC driver

JVM

disabled

Recovery Log

Nicolas Modrzyk - http://c-jdbc.objectweb.org/ - c-jdbc@objectweb.org
Journaling

- Backend is enabled
- All database updates are logged (SQL statement, user, transaction, ..)

![Diagram showing C-JDBC Controller, PostgreSQL JDBC driver, EJB Container, JVM, C-JDBC driver, JDBC Recovery Log, Octopus, and Recovery Log.]

- PostgreSQL enabled
Adding backend on the fly

» Add new backends while system online
» Restore dump corresponding to initial checkpoint with Octopus

- Add new backends while system online
- Restore dump corresponding to initial checkpoint with Octopus

Octopus

C-JDBC Controller

PostgreSQL JDBC driver

JDBC Recovery Log

EJB Container
JOnAS, WebLogic, JBoss, WebSphere, ...

C-JDBC driver

JVM

dump for initial checkpoint

dumped

enabled

disabled

PostgreSQL

Recovery Log

PostgreSQL

disabled

enabled

disabled

PostgreSQL

Synchronizing backends

Replay updates from the log

Flowchart:
- "C-JDBC Controller"
- "PostgreSQL JDBC driver"
- "JVM"
- "EJB Container"
  - "JOnAS, WebLogic, JBoss, WebSphere, ...
- "C-JDBC driver"
- "Recovery Log"
- "JDBC Recovery Log"
- "Octopus"
- "dump for initial checkpoint"
- "PostgreSQL disabled"
- "PostgreSQL enabled"
Expanded Cluster

Enable backends when done

C-JDBC Controller

JDBC Recovery Log

Octopus

dump for initial checkpoint

C-JDBC driver

JVM

EJB Container
JOnAS, WebLogic, JBoss, WebSphere, ...

PostgreSQL JDBC driver

PostgreSQL enabled

PostgreSQL enabled

PostgreSQL enabled

Recovery Log

R e c o v e r y

L o g

C-JDBC Recovery Log

PostgreSQL enabled
Handling a backend failure

- A node fails!
- Automatically disabled but should be fixed or changed by administrator

![Diagram of backend failure handling]

- PostgreSQL JDBC driver
- C-JDBC Controller
- EJB Container
  - JOnAS, WebLogic, JBoss, WebSphere, ...
- JVM

- Recovery Log
- Octopus
- C-JDBC Driver

- Recovery Log dump for last checkpoint
- Checkpoint dump for initial checkpoint
- Checkpoint dump for last checkpoint
Restoring a backend

- Restore latest dump with Octopus

- Dump for initial checkpoint
- Dump for last checkpoint

- Octopus
- C-JDBC Controller
- Recovery Log
- EJB Container
  - JOnAS, WebLogic, JBoss, WebSphere, ...

- C-JDBC driver
- PostgreSQL JDBC driver

- JVM
- Recovery Log
- PostgreSQL JDBC driver

- PostgreSQL disabled
- PostgreSQL enabled
Re-synchronization

➨ Replay missing updates from log

- EJB Container
  JOnAS, WebLogic, JBoss, WebSphere, ...
- C-JDBC driver
- JVM

- C-JDBC Controller
  Recovery Log
  PostgreSQL JDBC driver

- PostgreSQL disabled
- PostgreSQL enabled
- PostgreSQL enabled

- Octopus
- JDBC Recovery Log

- dump for initial checkpoint
- dump for last checkpoint
Healed Cluster

Re-enable backend when done

Octopus

dump for initial checkpoint

dump for last checkpoint

C-JDBC Controller

Recovery Log

EJB Container

JOINAS, WebLogic, JBoss, WebSphere, ...

C-JDBC driver

JVM

dumped

enabled

PostgreSQL JDBC driver

PostgreSQL enabled

PostgreSQL enabled

PostgreSQL enabled
Outline - Performance

Motivations
Use-Cases
C-JDBC concepts
Performance
Monitoring
Community
Conclusion
TPC-W Performance
(Amazon.com)

Throughput in requests per minute

Number of nodes

Single DB
RAIDb-0
RAIDb-1
RAIDb-2
RUBiS- Tomcat without C-JDBC caching

Throughput: 3900 pages/min

1 Database
100% cpu

Tomcat
~50% cpu
RUBiS- Tomcat with C-JDBC caching

Throughput: 4200 pages/min

Tomcat
~55% cpu

1 Database
~20% cpu

C-JDBC
<10% cpu
Outline - Monitoring

Motivations
Use-Cases
C-JDBC concepts
Performance
Monitoring
Community
Conclusion
Monitoring/Trace

Trace, save, get statistic content of different modules

Controller, database, users, backend, cache, load, memory usage ...
SQL Console: Squirrel

- Execute a set of atomic SQL requests
- Verify content of clustered database
- Verify cluster schemas
**View graphic remote logs**

- **Watch execution:**
  - per backend
  - per controller
  - per virtual database

### Logs Table

<table>
<thead>
<tr>
<th>Date</th>
<th>Level</th>
<th>Category</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>ControllerWorkerThread terminating.</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
<tr>
<td>Mon Feb 02 16</td>
<td>DEBUG</td>
<td>org.objectweb.jjdbc.controller</td>
<td>Released idle connection (idle timeout reached).</td>
</tr>
</tbody>
</table>

Displaying 11 records out of a total of 11 records.
Outline - Community

Motivations
Use-Cases
C-JDBC concepts
Performance
Monitoring
Community
Conclusion
Stats as of Feb, 2004

Downloads
- total: 11260 downloads since may 2003
- 2004: > 1300 downloads
- Top 5 of the most downloaded ObjectWeb project

Mailing lists
- c-jdbc@objectweb.org: 124 subscribers

Team
- 11 committers
- 1 full-time INRIA engineer
The developer community

Mathieu Peltier (ObjectWeb)
- build scripts, automatic installer, JUnit test

Julie Marguerite (ObjectWeb)
- JDBCRecoveryLog, automatic schema detection

Christiana Amza (Rice University), Anupam Chanda (Rice University), Sara Bouchenak (EPF Lausanne)
- SQL query caching

Guillaume Bort (INRIA Lorraine)
- JBoss support

Marek Prochazka (INRIA Rhone-Alpes)
- DataSource implementation

Greg Ward (dplanet.ch)
- Sybase support, design, debug

Marc Wick (monte-bre.ch)
- HSQL support, design debug and ideas

Duncan Smith (mightybot.com)
- IP binding, security concerns, console, JMX, distributed management

Vadim Kassin (Kazakhstan Stock Exchange)
- Autogenerated keys support
Outline - Conclusion

Motivations
Use-Cases
C-JDBC concepts
Performance
Monitoring
Community
Conclusion
Current status

C-JDBC 1.0 rc1 release

- Generic JDBC 2.0 driver
- Schedulers and load balancers for RAIDb 0, 1 and 2
- Fine grain query caching and sql monitoring
- JDBC recovery log
- Logger/request player
- Java installer
- User documentation
- Octopus integration
On-going work and efforts

Listen to the needs of users, quick answers on the mailing list

Horizontal scalability

Fully featured administration console

Graphical configuration and deployment of centralized/distributed backeds and controllers (offline/online)

Dynamic reconfiguration

Automated Load testing, report page updated by users

RPM packaging (Jpackage version 1.0b15 done)

C-ODBC (asked by a lot of people)
Take this message at home

Database Clustering Middleware
(100% java)
Based on JDBC Standard
No code modification
(application or database)
Open source (LGPL)
Questions & Answers

__________
Thanks to all users and contributors ...

http://c-jdbc.objectweb.org
Prototype

C-JDBC Management Framework

Shared design
Request cache

caches results from SQL requests
improved SQL statement analysis to limit cache invalidations
  – table based invalidations
  – column based invalidations
  – single-row SELECT optimization
request parsing possible in the C-JDBC driver
  – offload the controller
  – parsing caching in the driver
Load balancer

RAIDb-0
- query directed to the backend having the needed tables

RAIDb-1
- read executed by current thread
- write executed in parallel by a dedicated thread per backend
- result returned if one, majority or all commit
- if one node fails but others succeed, failing node is disabled

RAIDb-2
- same as RAIDb-1 except that writes are sent only to nodes owning the written table
Connection Manager

Connection pooling for a backend

- Simple: no pooling
- RandomWait: blocking pool
- FailFast: non-blocking pool
- VariablePool: dynamic pool

Connection pools defined on a per login basis

- resource management per login
- dedicated connections for admin
Scheduler

Manages concurrency control

Specific implementations for Single DB, RAIDb 0, 1 and 2

Query-level

Optimistic and pessimistic transaction level

- uses the database schema that is automatically fetched from backends
Recovery Log

Checkpoints are associated with database dumps
Record all updates and transaction markers since a checkpoint
Used to resynchronize a database from a checkpoint

JDBCRecoveryLog
- store information in a database
- can be re-injected in a C-JDBC cluster for fault tolerance
Making new checkpoints

- Disable one backend to have a coherent snapshot
- Mark the new checkpoint entry in the log
- Use Octopus to store the dump
Making new checkpoints

Replay missing updates from log

- C-JDBC Controller
- EJB Container
  JOnAS, WebLogic, JBoss, WebSphere, ...
- PostgreSQL JDBC driver
- JVM
- enabled

- Recovery Log
- PostgreSQL JDBC driver

- dump for initial checkpoint
- dump for last checkpoint

- Octopus
- PostgreSQL disabled
- PostgreSQL enabled
- PostgreSQL enabled
Making new checkpoints

Re-enable backend when done

C-JDBC Controller

JVM

EJB Container

JOnAS, WebLogic, JBoss, WebSphere, ...

C-JDBC driver

Recovery Log

PostgreSQL JDBC driver

Octopus

dump for initial checkpoint

dump for last checkpoint

dump for last checkpoint

PostgreSQL enabled

PostgreSQL enabled

PostgreSQL enabled