Outline
- J2EE Clustering
- C-JDBC
- Démo
- Conclusion

J2EE Clustering
- Load balancing on Web Servers
  - hardware: L4-switch
  - software: One-IP techniques, RR-DNS, ...
- Web server to Servlet server
  - Load balancing with JK module (mod_jk)
  - Static weighted round-robin
  - Session affinity

J2EE Clustering
- Servlet/JSP server clustering
  - Tomcat in-memory session replication
  - failover ensured by mod_jk
- Servlet/JSP to EJB server
  - clustered JNDI
  - load-balancing and failover by cluster-aware stubs

J2EE Clustering
- EJB Server clustering
  - cluster stubs for load-balancing
  - transparent failover for idempotent methods
  - session-beans
    - stateless: no state to replicate
    - stateful: in-memory replication or database-based persistency
  - entity beans
    - usually database based persistency
**J2EE Clustering**
- Database clustering
  - Commercial offers
    - Oracle RAC (60,000$ / cpu)
    - based on expensive SAN (Storage Area Network)
  - Open-source solutions
    - No real clustering
      - master/slave replication in MySQL
      - Postgres-R (still in alpha)
      - Clustered JDBC

**Outline**
- J2EE Clustering
- C-JDBC
- Démo
- Conclusion

---

**motivations**
- J2EE performance scalability bounded by database performance
- Large SMP are not commodity
- Database tier must be
  - scalable
  - fault tolerant (high availability + failover)
  - without modifying the client application
  - using open source databases
  - on commodity hardware

---

**RAIDb**
- Redundant Array of Inexpensive Databases (RAIDb)
  - better performance and fault tolerance than a single database,
  - at a low cost,
  - by combining multiple DB instances into an array of DB.
  - 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 (best performance)
    - RAIDb-1: full mirroring (best fault tolerance)
    - RAIDb-2: partial replication (best tradeoff)

---

**C-JDBC – Key ideas**
- 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

---

**RAIDb with**
- RAIDb Controller
  - Scalability, Fault tolerance, Failover
  - Monitoring, Caching, Logging
  - C-JDBC driver
  - MySQL, PostgreSQL, Oracle, DB2, InstantDB, ...
  - EJB Container
    - JOnAS, WebLogic, JBoss, ...
  - Servlet container
    - Tomcat, Jetty, ...

---

**C-JDBC motivations**
- J2EE performance scalability bounded by database performance
- Large SMP are not commodity
- Database tier must be
  - scalable
  - fault tolerant (high availability + failover)
  - without modifying the client application
  - using open source databases
  - on commodity hardware
C-JDBC RAIDb-1 example
- no client code modification
- original PostgreSQL driver and RDBMS engine
- C-JDBC provides scalable performance and high availability

C-JDBC RAIDb-2 example
- unload a single Oracle DB with several MySQL
- add caching, fault tolerance, and monitoring for free

Inside the Controller
- Horizontal scalability
  - prevents the controller to be a Single Point Of Failure
  - distributes the load among several controllers
  - coherency ensured by group communications
- C-JDBC Driver
  - multiple controllers automatic failover
    - jdbc:c-jdbc://node1:1099:c1,node2:1234:c2/myDB
  - connection caching
  - URL parsing/controller lookup caching

C-JDBC – Horizontal scalability
- Vertical scalability
  - allows nested RAIDb levels
  - allows tree architecture for scalable write broadcast
  - necessary with large number of backends
  - C-JDBC driver re-injected in C-JDBC controller
**C-JDBC vertical scalability**
- RAIDb-1-1 with C-JDBC
- no limit to composition deepness

**Performance - TPC-W**

![Graph showing performance metrics](image)

**Fine-grain caching**
- Cache hit rate with TPC-W
- browsing mix

<table>
<thead>
<tr>
<th>Method</th>
<th>Throughput</th>
<th>Response time</th>
<th>Hit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>No cache</td>
<td>9.1 req/s</td>
<td>3.30s</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>12.9 req/s</td>
<td>1.96s</td>
<td>12.6%</td>
</tr>
<tr>
<td>Column</td>
<td>16 req/s</td>
<td>1.36s</td>
<td>46.6%</td>
</tr>
<tr>
<td>Column+ single-row</td>
<td>16 req/s</td>
<td>1.35s</td>
<td>49.2%</td>
</tr>
</tbody>
</table>

**Outline**
- J2EE Clustering
- C-JDBC
- Démo
- Conclusion
Outline
- J2EE Clustering
- C-JDBC
- Démo
- Conclusion

Conclusion
- J2EE Clustering
  - One-IP or RR-DNS
  - Apache + mod_jk
  - Tomcat + in memory session replication
- EJB Server
  - distributed JNDI
  - cluster-aware stubs
- C-JDBC + any RDBMS

Questions?
- Apache/Tomcat: http://www.apache.org
- JOnAS: http://www.objectweb.org/jonas
- C-JDBC: http://c-jdbc.objectweb.org

Bonus slides

RAIDb - Definition
- Redundant Array of Inexpensive Databases
- better performance and fault tolerance than a single database, at a low cost, by combining multiple database instances into an array of databases
- RAIDb controller
  - gives the view of a single database to the client
  - balance the load on the database backends

RAIDb levels
- RAIDb-0
  - partitioning
  - no duplication and no fault tolerance
  - at least 2 nodes
RAIDb levels
- RAIDb-1
  - mirroring
  - performance bounded by write broadcast
  - at least 2 nodes

- RAIDb-1ec
  - mirroring + error checking
  - error checking
    - read request sent to multiple databases
    - replies compared
    - result returned only if a quorum is reached
  - at least 3 nodes

RAIDb levels
- RAIDb-2
  - partial mirroring + partial partitioning
  - at least 2 copies of each table
  - at least 3 nodes

RAIDb levels composition
- RAIDb-0-1
  - RAIDb-0 at the top level
  - RAIDb-1 underneath

Horizontal scalability