ObjectWeb
Open Source Middleware

INRIA
RHÔNE-ALPES

# C-JDBC status


## Emmanuel Cecchet

*c-jdbc@objectweb.org*

# Outline

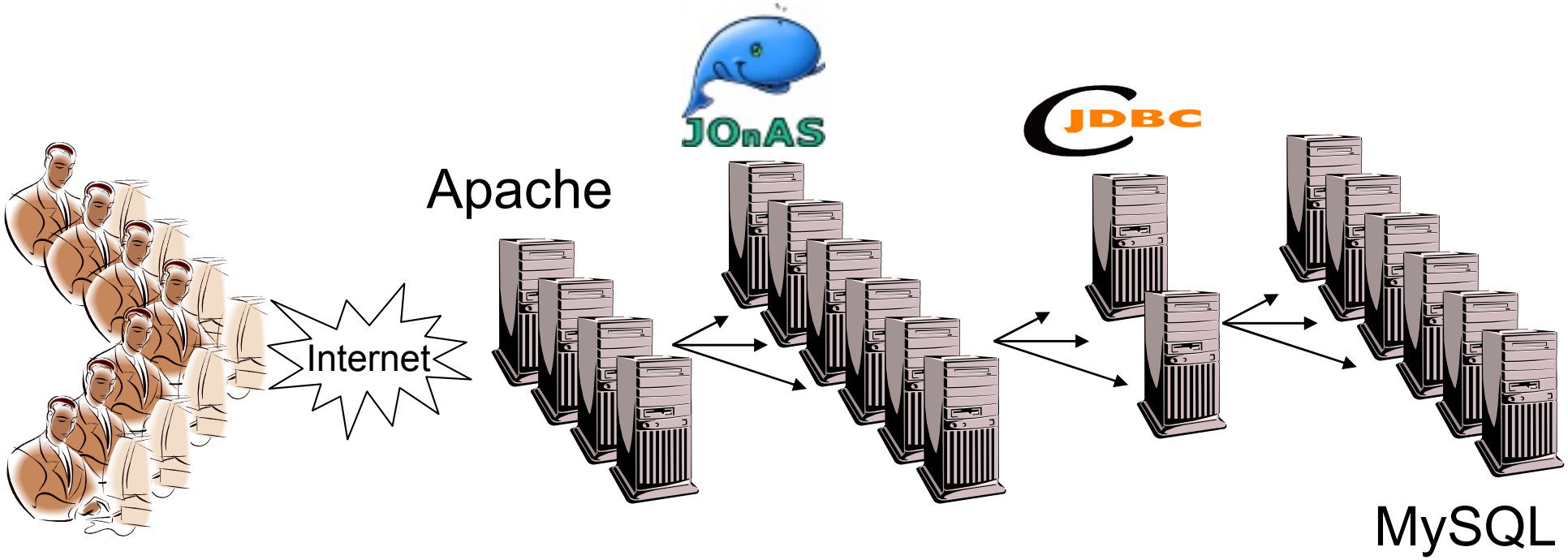
- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

# 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

# How do we want to use C-JDBC ?

- **end-to-end open source solution for large scale J2EE clustered application servers**
- **cost-effective solution for both small and large configurations**



# Redundant Array of Inexepensive Databases

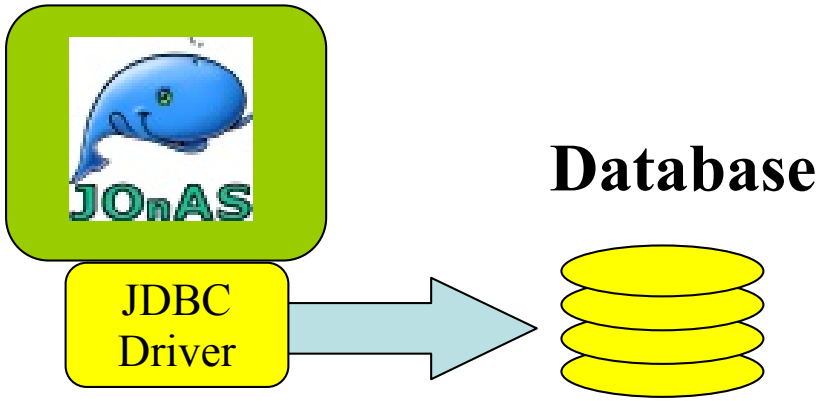
- **RAIDb controller**
  - gives the view of a single database to the client
  - balance the load on the database backends
- **RAIDb levels**
  - RAIDb-0: full partitioning
  - RAIDb-1: full mirroring
  - RAIDb-2: partial replication
  - composition possible

# C-JDBC

- **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**

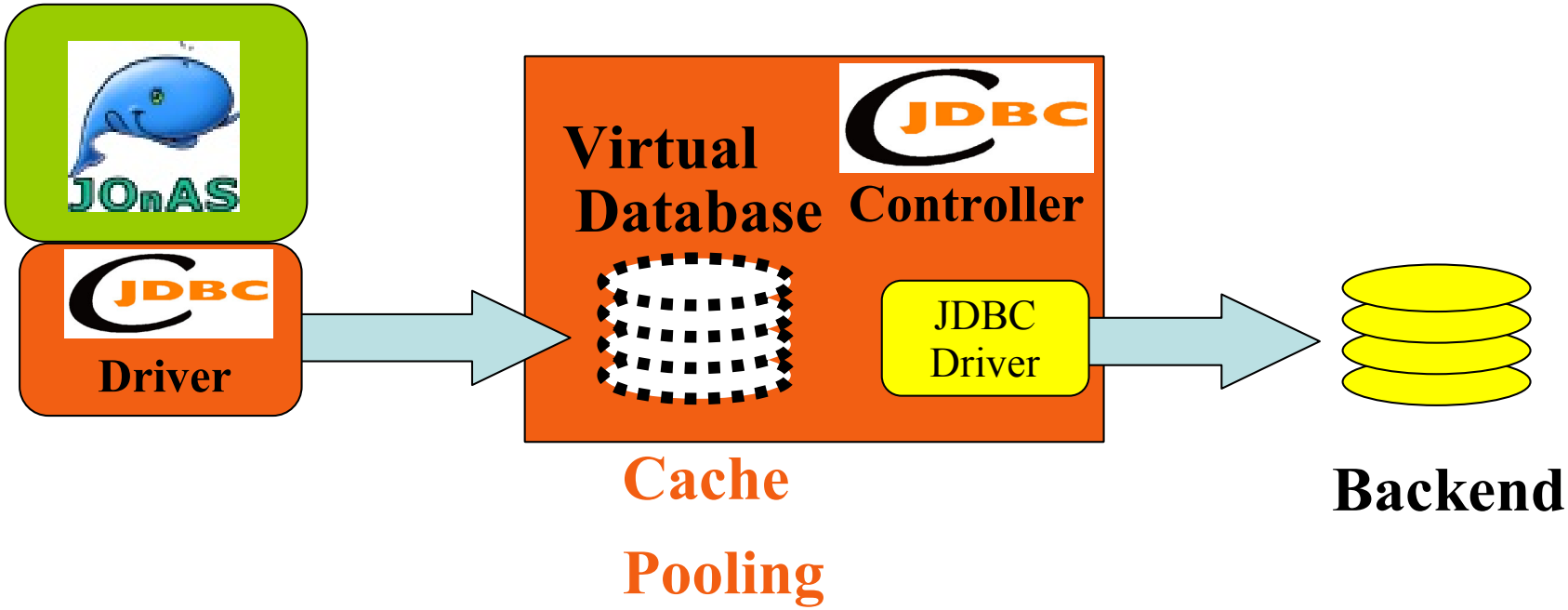
# Using JOnAS with C-JDBC

- **Without C-JDBC**



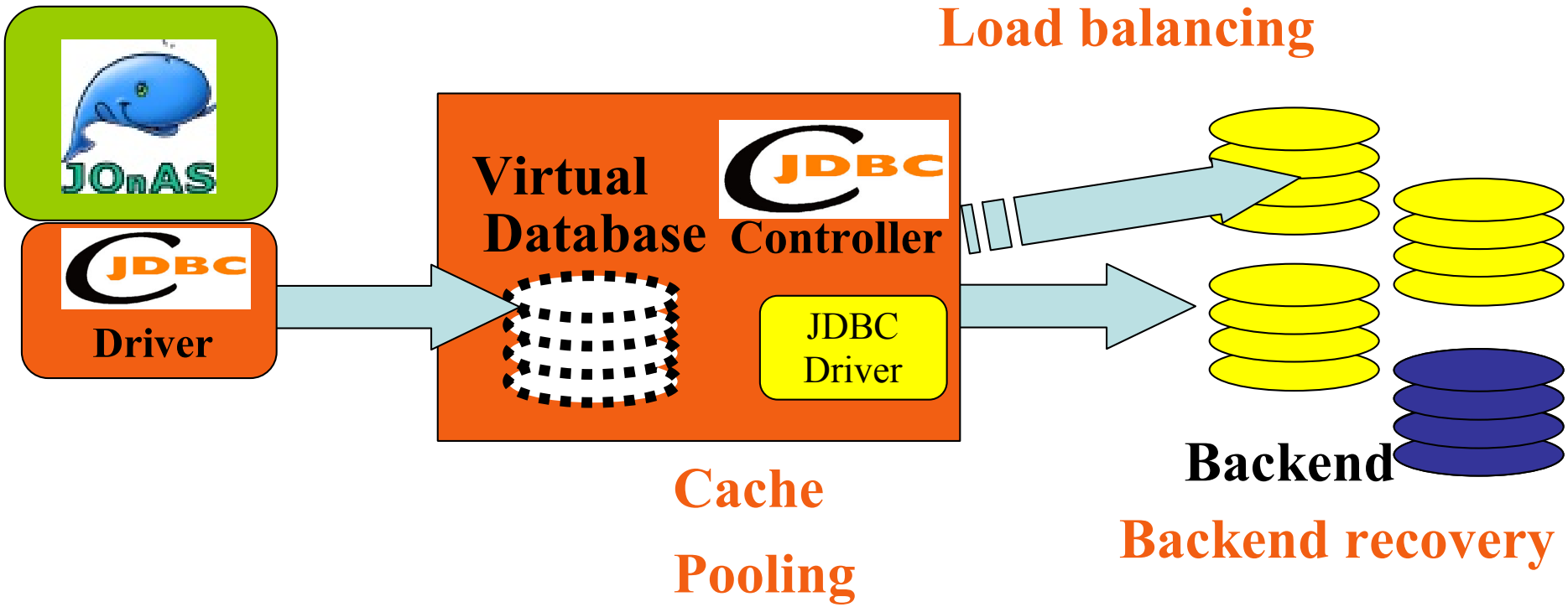
# Using JOnAS with C-JDBC

- **Update the datasource to use the C-JDBC driver**



# What C-JDBC offers

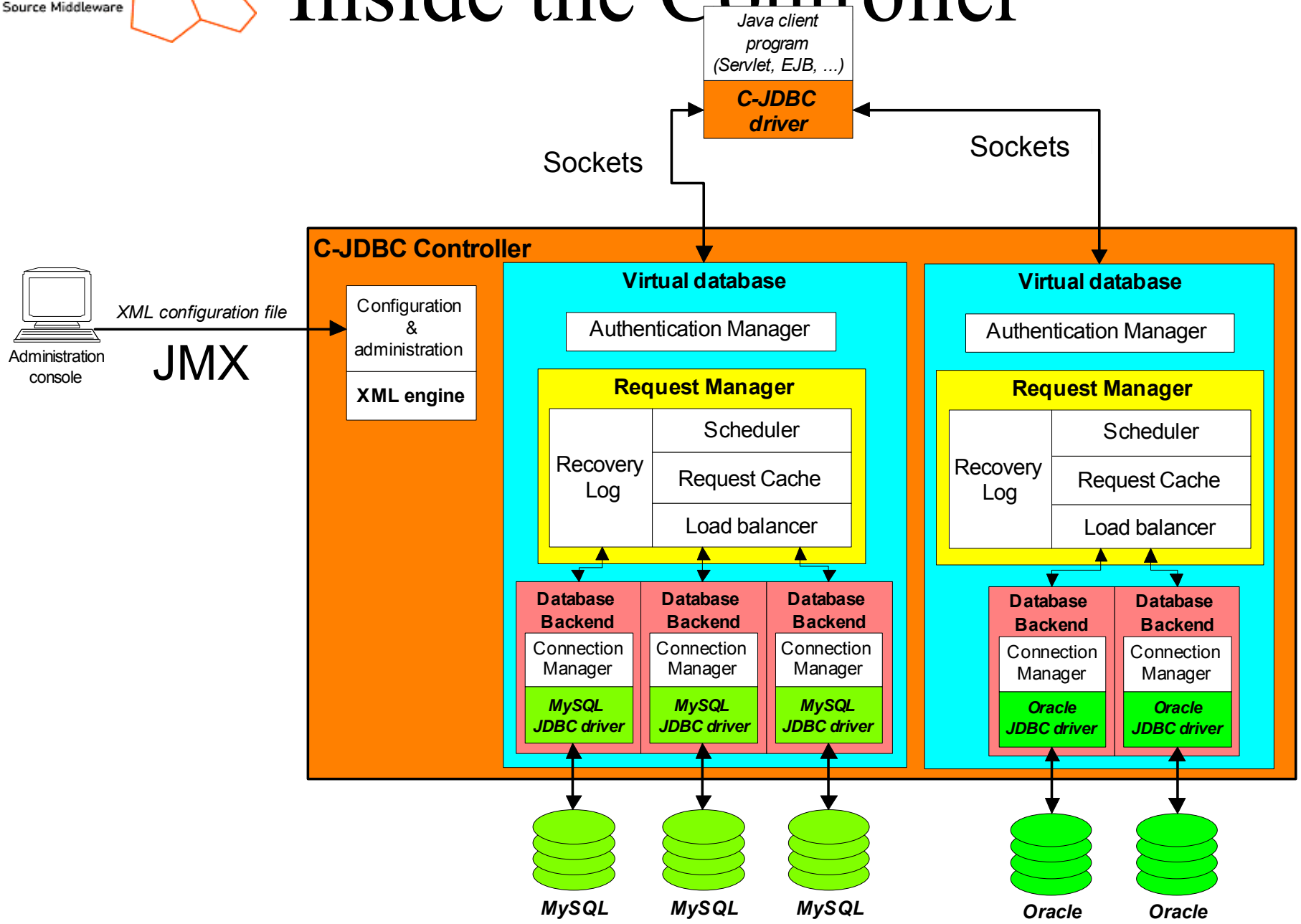
- No application modification
- No JOnAS modification
- No database modification



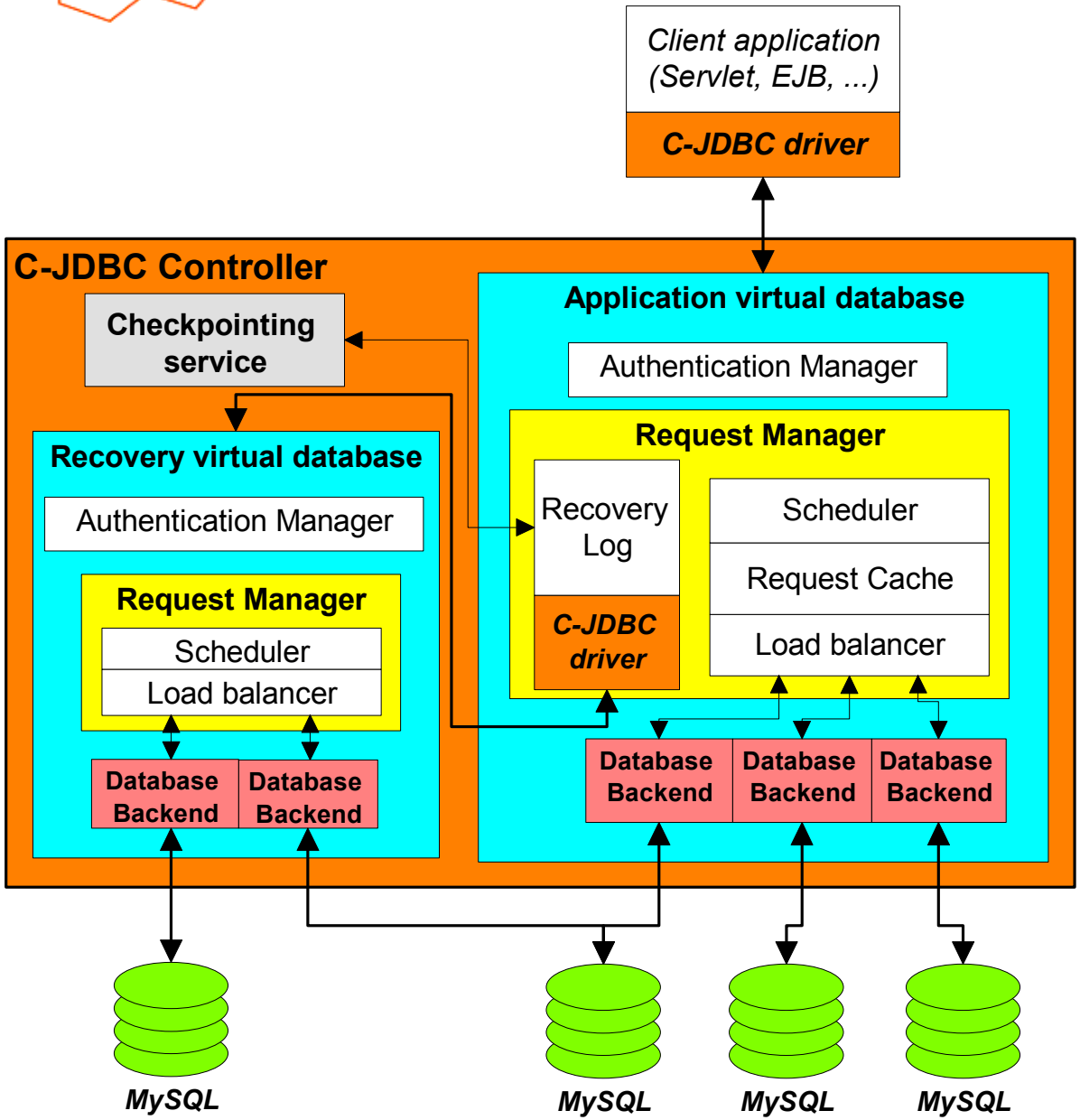
# Outline

- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

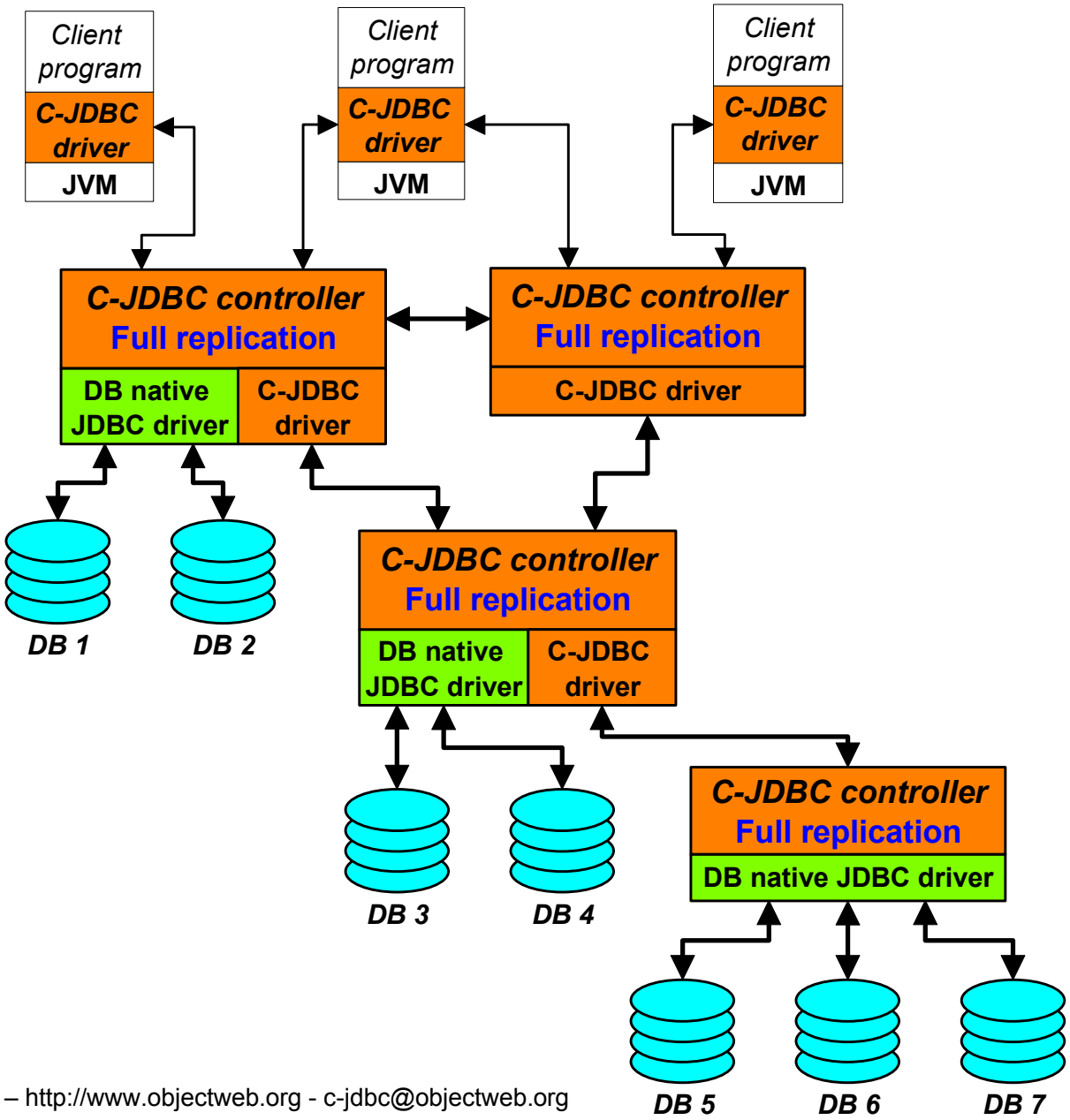
# Inside the Controller



# Fault tolerant recovery log

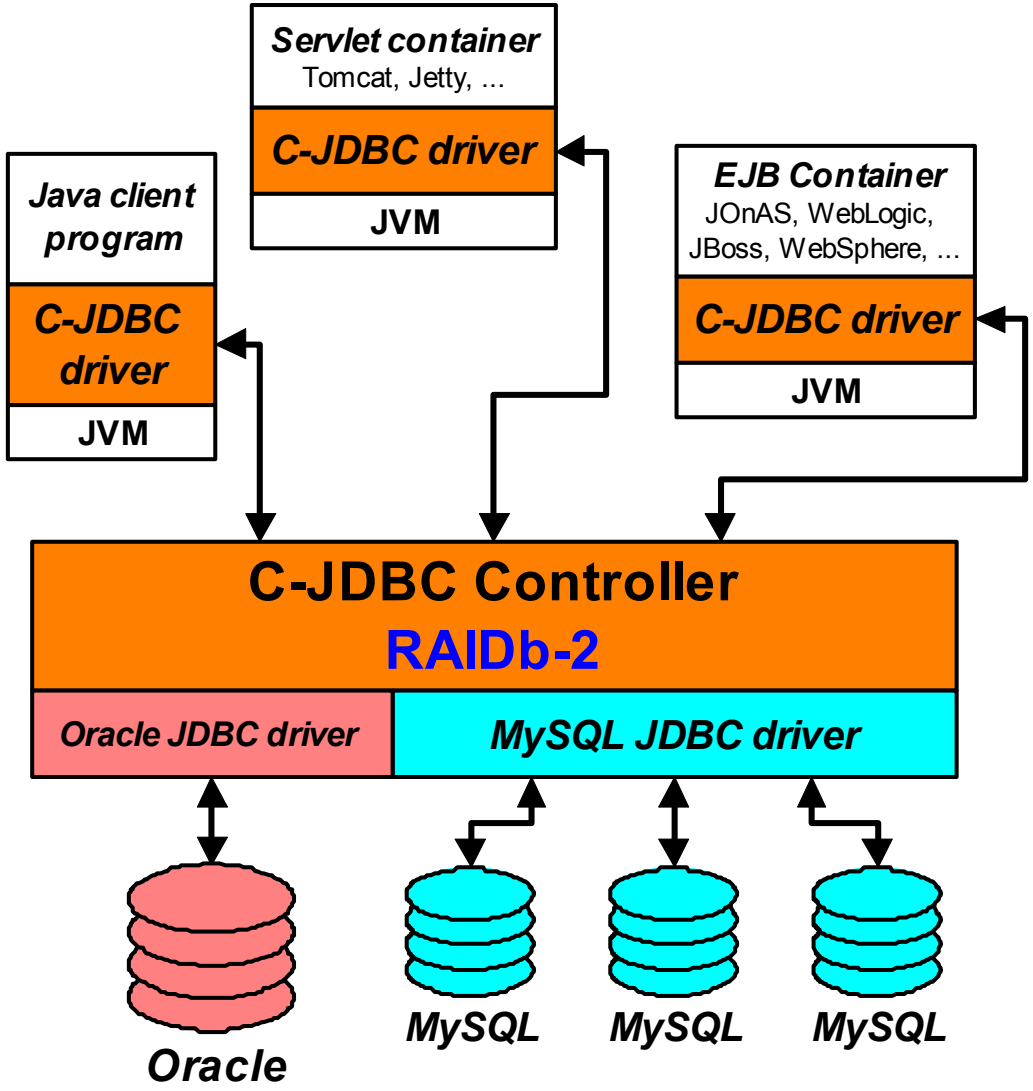


# Cascading controller



# Heterogeneity support

- application already written for a specific [commercial] database
- user defined rules for on-the-fly query rewriting to execute on heterogeneous backends



# Outline

- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

# Query caching status

- **Cache contains a list of SQL->ResultSet**
- **Policy defined by queryPattern->Policy**
- **3 policies**
  - EagerCaching: variable granularities for invalidations
  - RelaxedCaching: invalidations based on timeout
  - NoCaching: never cached

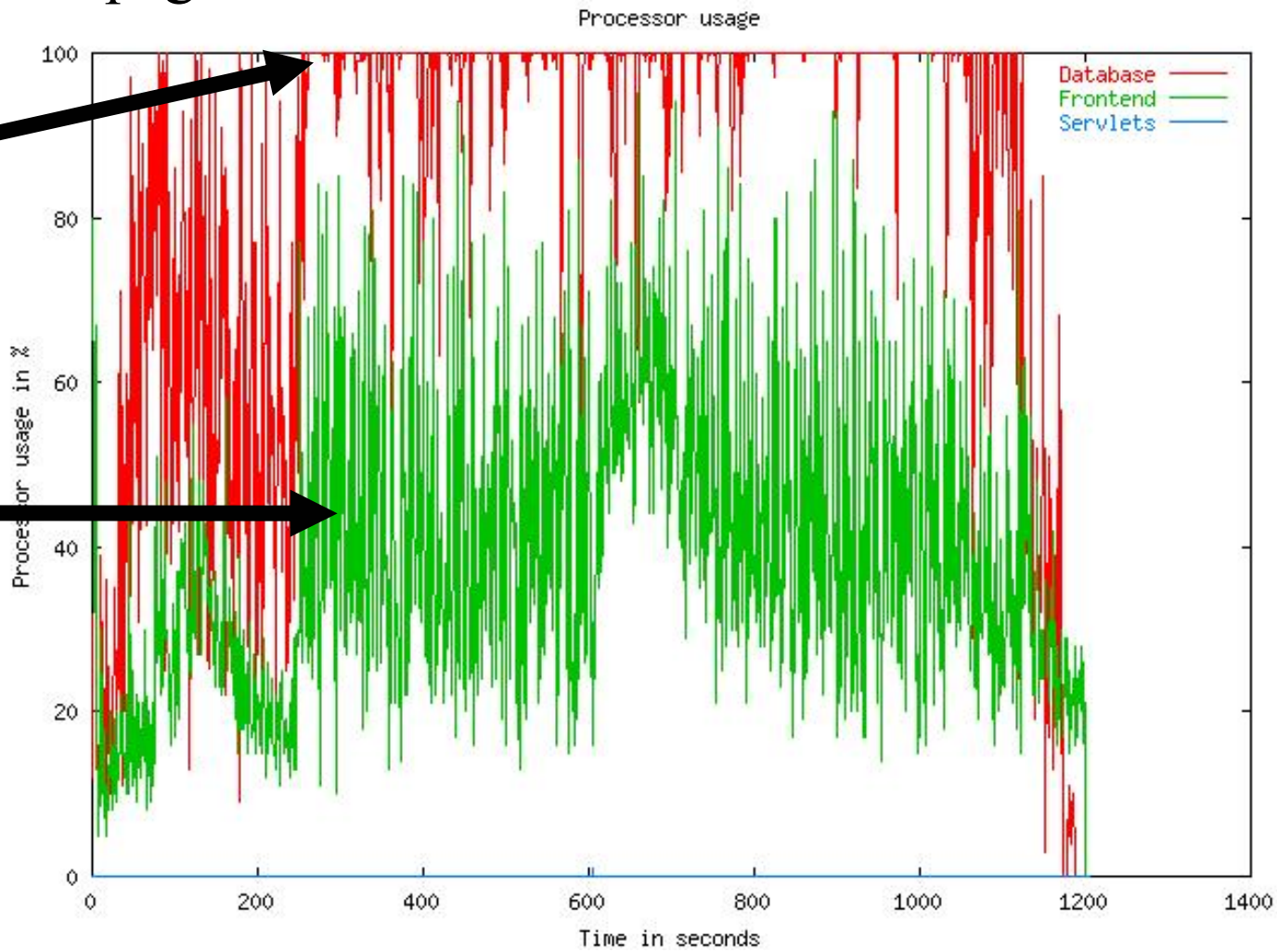
| *RUBiS bidding mix with 450 clients* | No cache | Coherent cache | Relaxed cache |
|---|---|---|---|
| Throughput (rq/min) | 3892 | 4184 | 4215 |
| Avg response time | 801 ms | 284 ms | 134 ms |
| Database CPU load | 100% | 85% | 20% |
| C-JDBC CPU load | - | 15% | 7% |

# RUBiS - Tomcat without C-JDBC caching

**Throughput:** 3900 pages/min

1 Database
100% cpu

Tomcat
~50% cpu



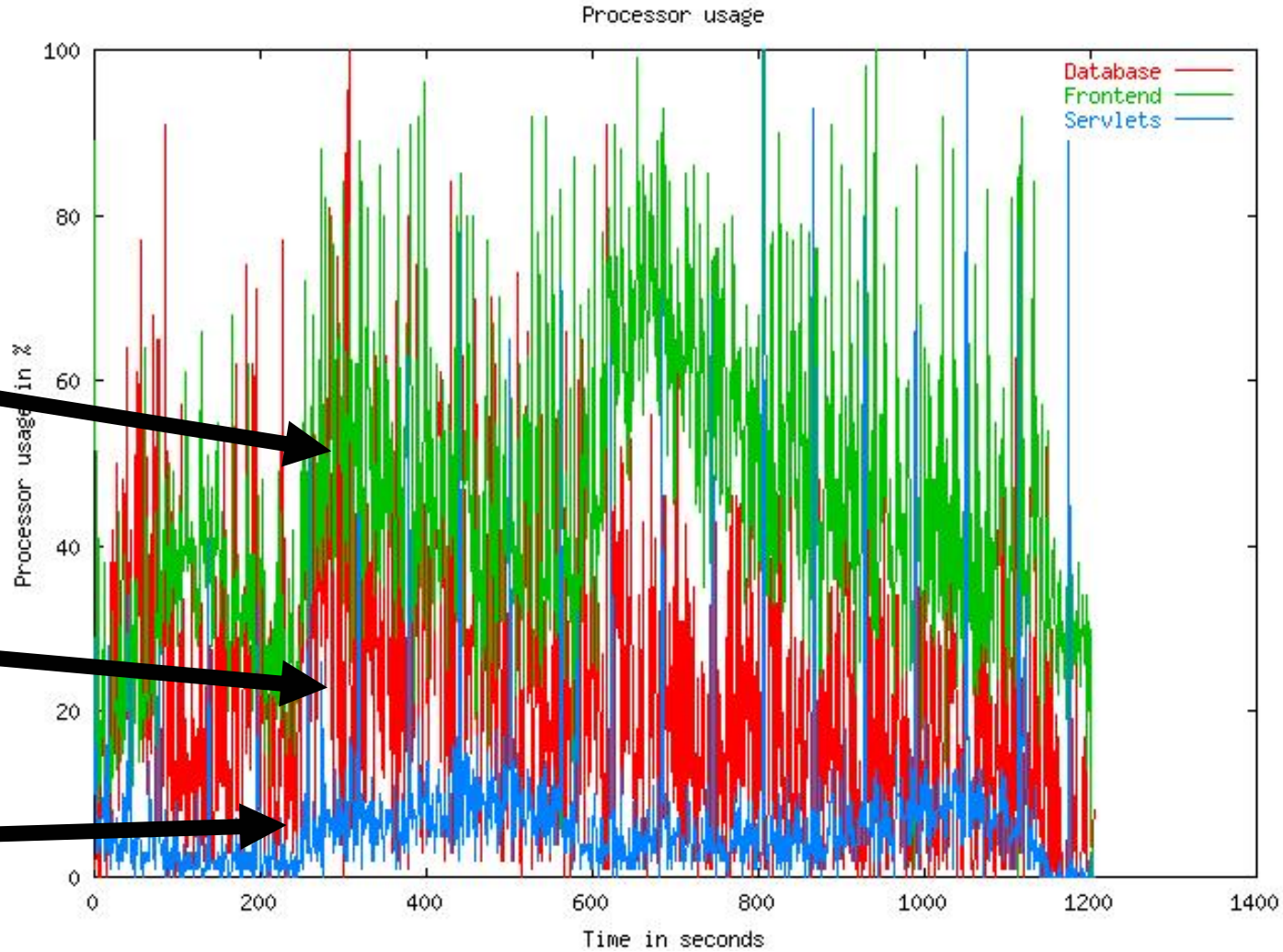
# RUBiS - Tomcat with C-JDBC relaxed caching

**Throughput:** 4200 pages/min

Tomcat
~55% cpu

1 Database
~20% cpu

C-JDBC
<10% cpu



# JOnAS and C-JDBC query caching

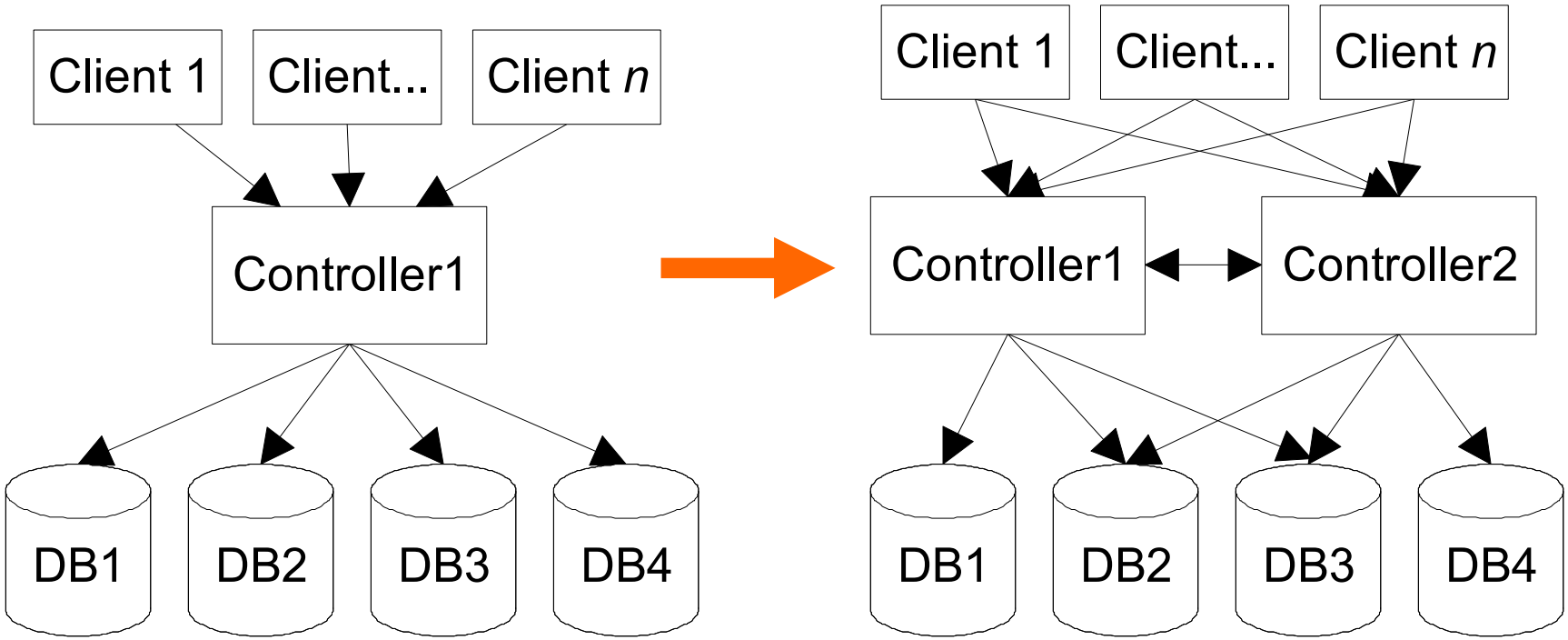
| RUBiS | Throughput (req/min) | | Response time (ms) | |
|---|---|---|---|---|
| | No cache | C-JDBC | No cache | C-JDBC |
| Servlet | 3890 | 4140 | 744 | 310 |
| JOnAS SF CMP 1.1 | 2947 | 3418 | 2929 | 2045 |
| JOnAS CMP 2.0 | 2178 | 2450 | 6372 | 4630 |

# Outline

- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

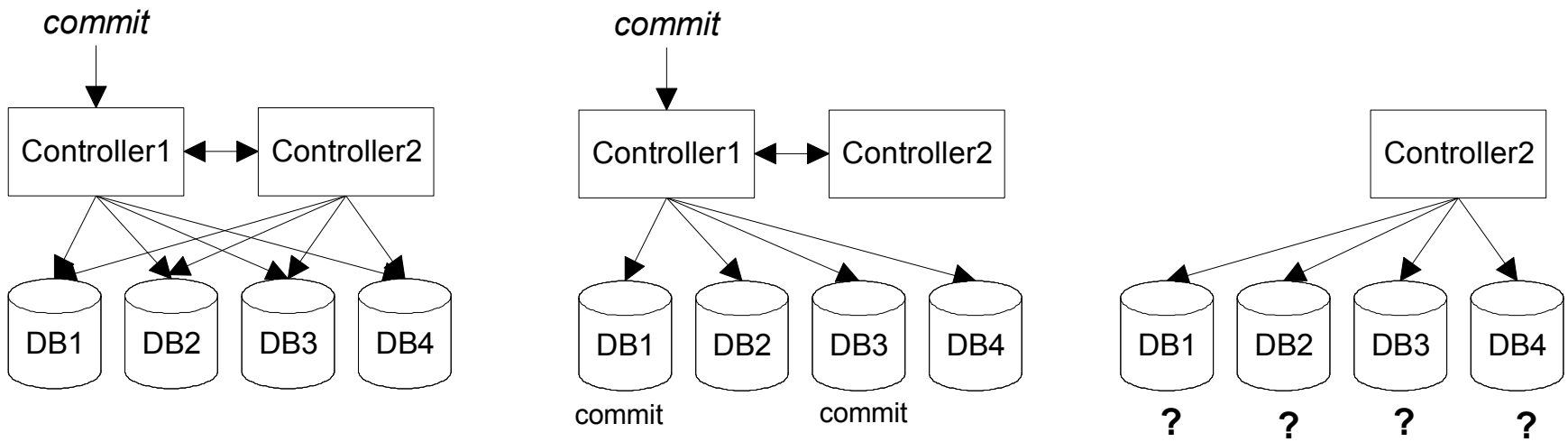
# Horizontal scalability

- **JGroups for controller synchronization**
- **Groups messages for writes only**

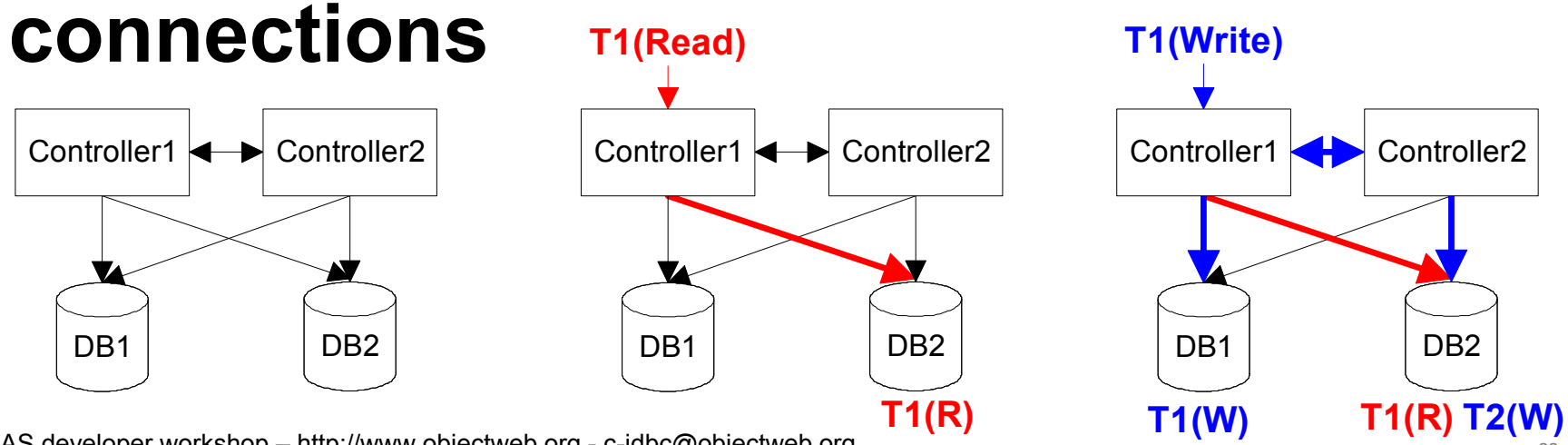


# Horizontal scalability

## Centralized write approach issues



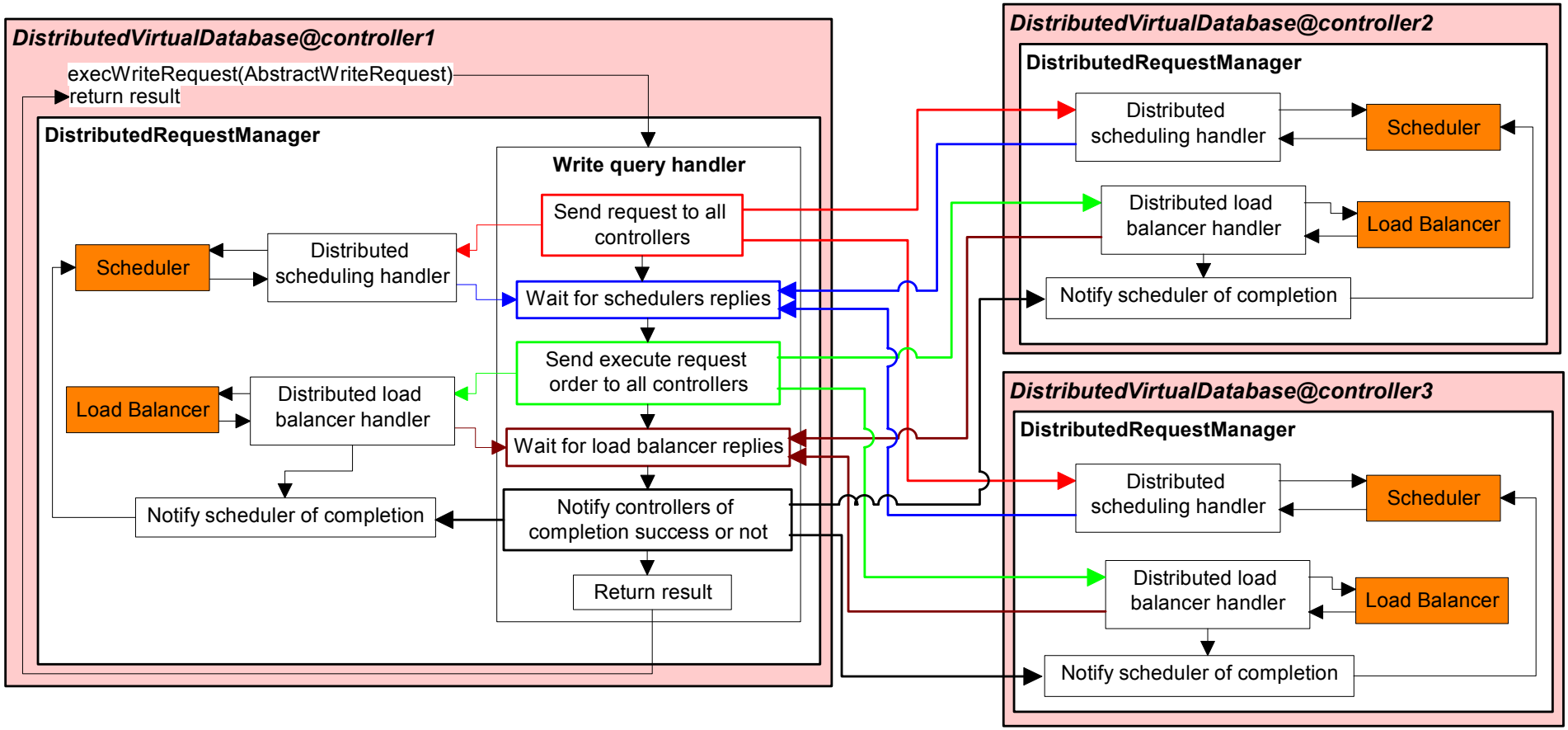

## Issues with transactions assigned to connections



# Horizontal scalability

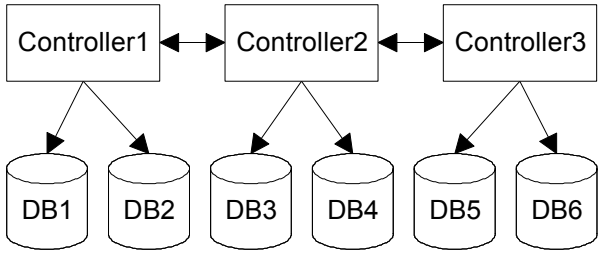
## General case for a write query

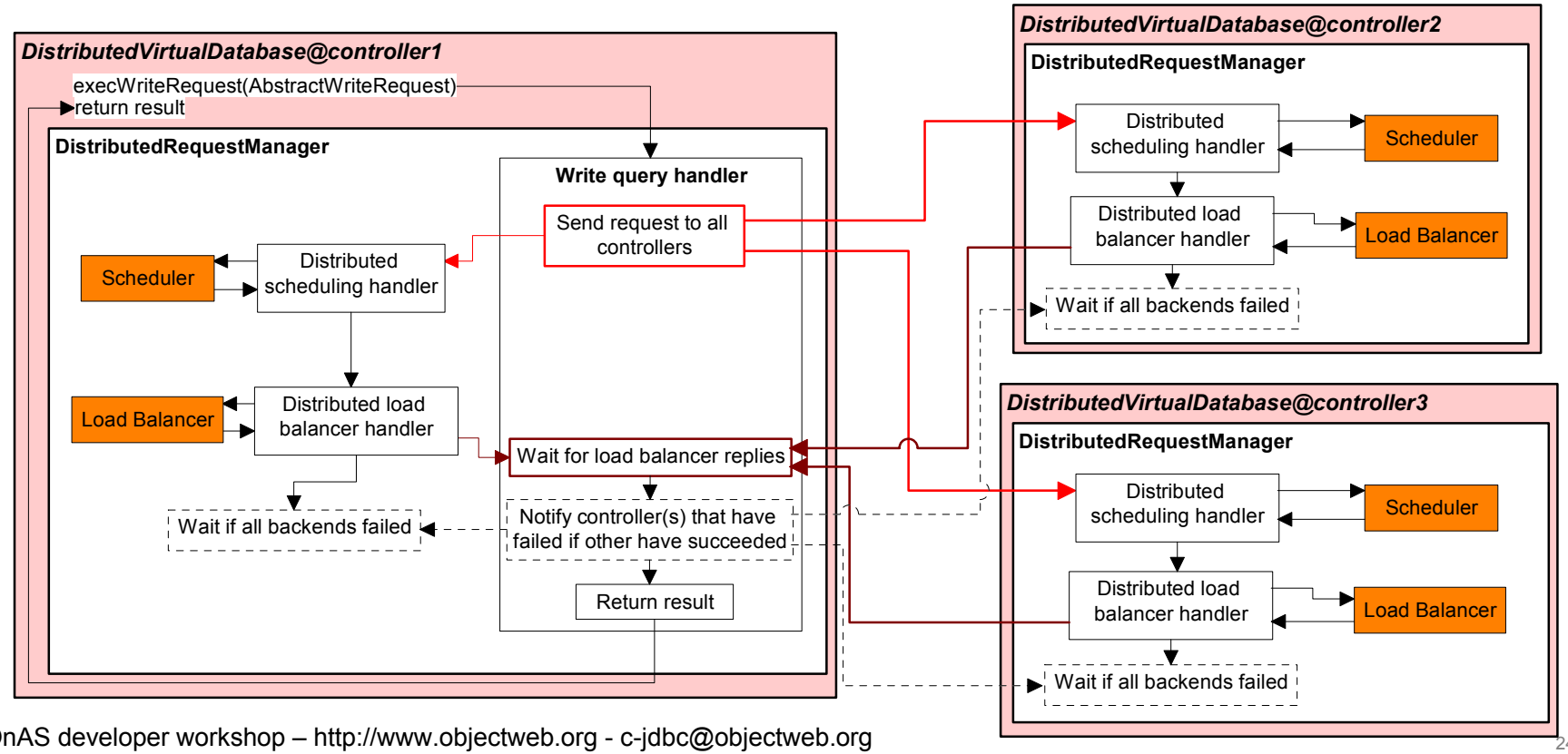
- 3 multicast + 2n unicast



# Horizontal scalability

- **Solution: No backend sharing**
  - 1 multicast + *n* unicast [+ 1 multicast]



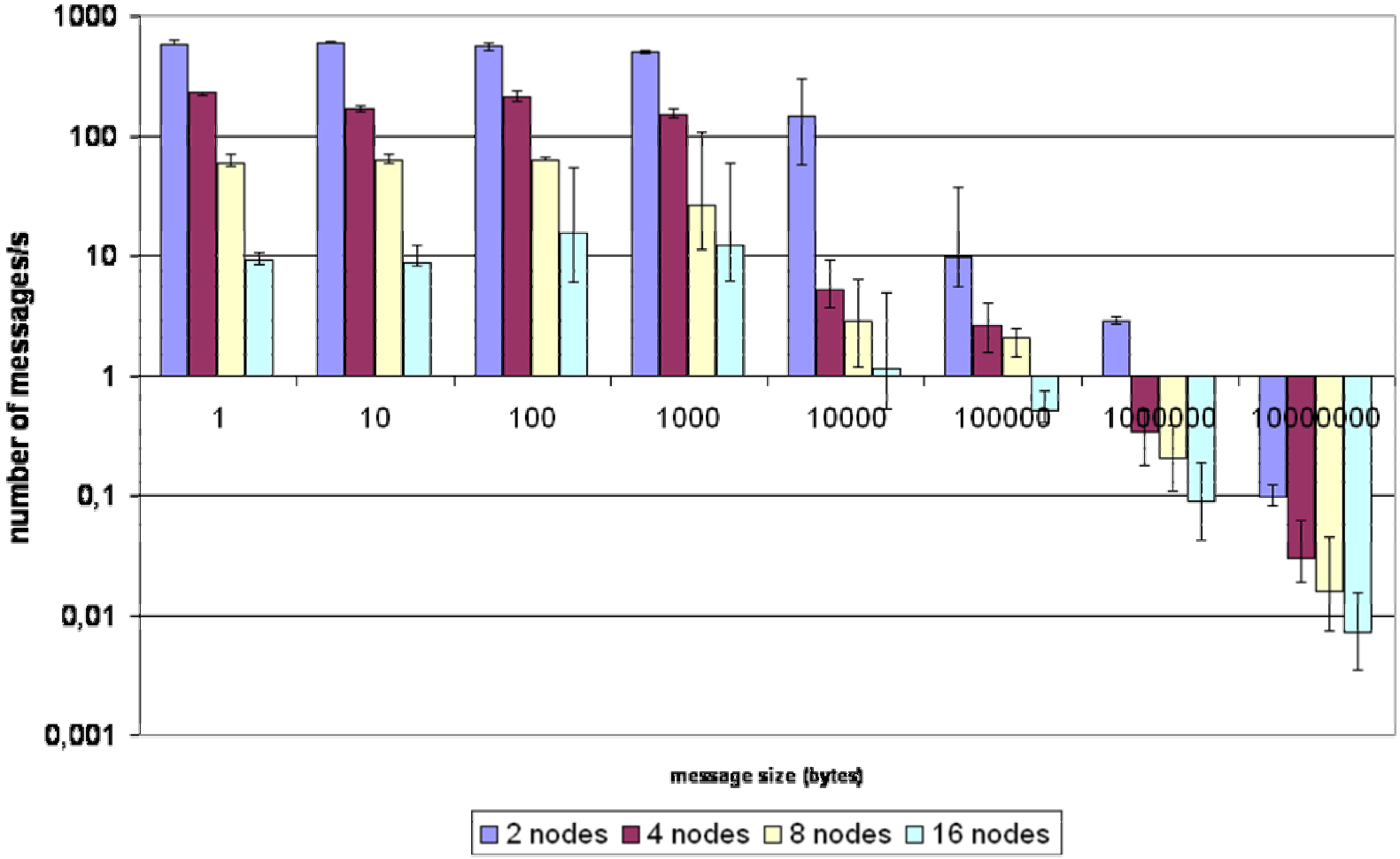



# Horizontal scalability

- **Issues with JGroups**
  - resources needed by a channel
  - instability of throughput with UDP
  - performance scalability
- **TCP better than UDP but**
  - unable to disable reliability on top of TCP
  - unable to disable garbage collection
  - ordering implementation is sub-optimal
- **Need for a new group communication layer optimized for cluster**

# Horizontal scalability

- **JGroups performance on UDP/FastEthernet**



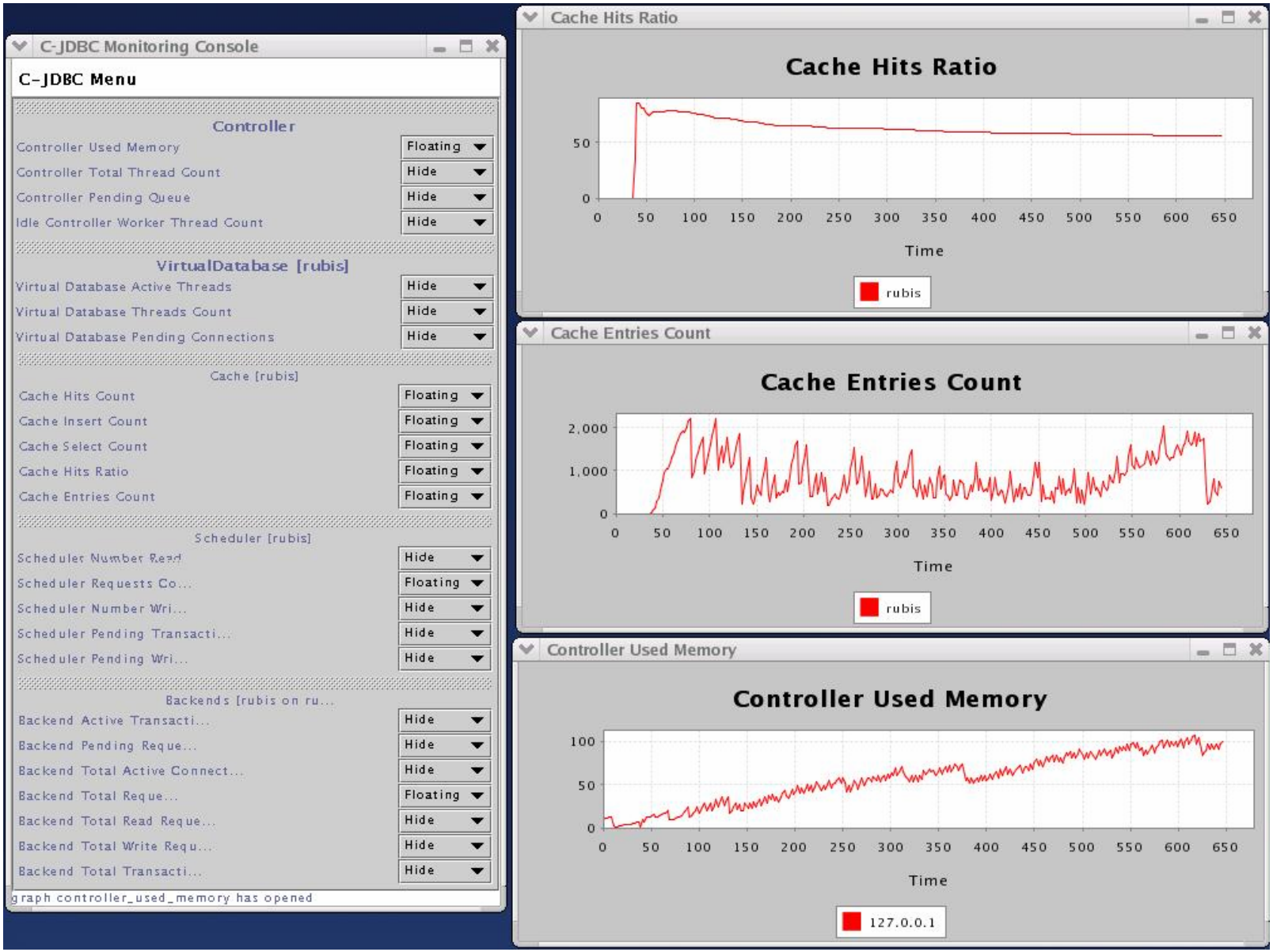
# Outline

- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

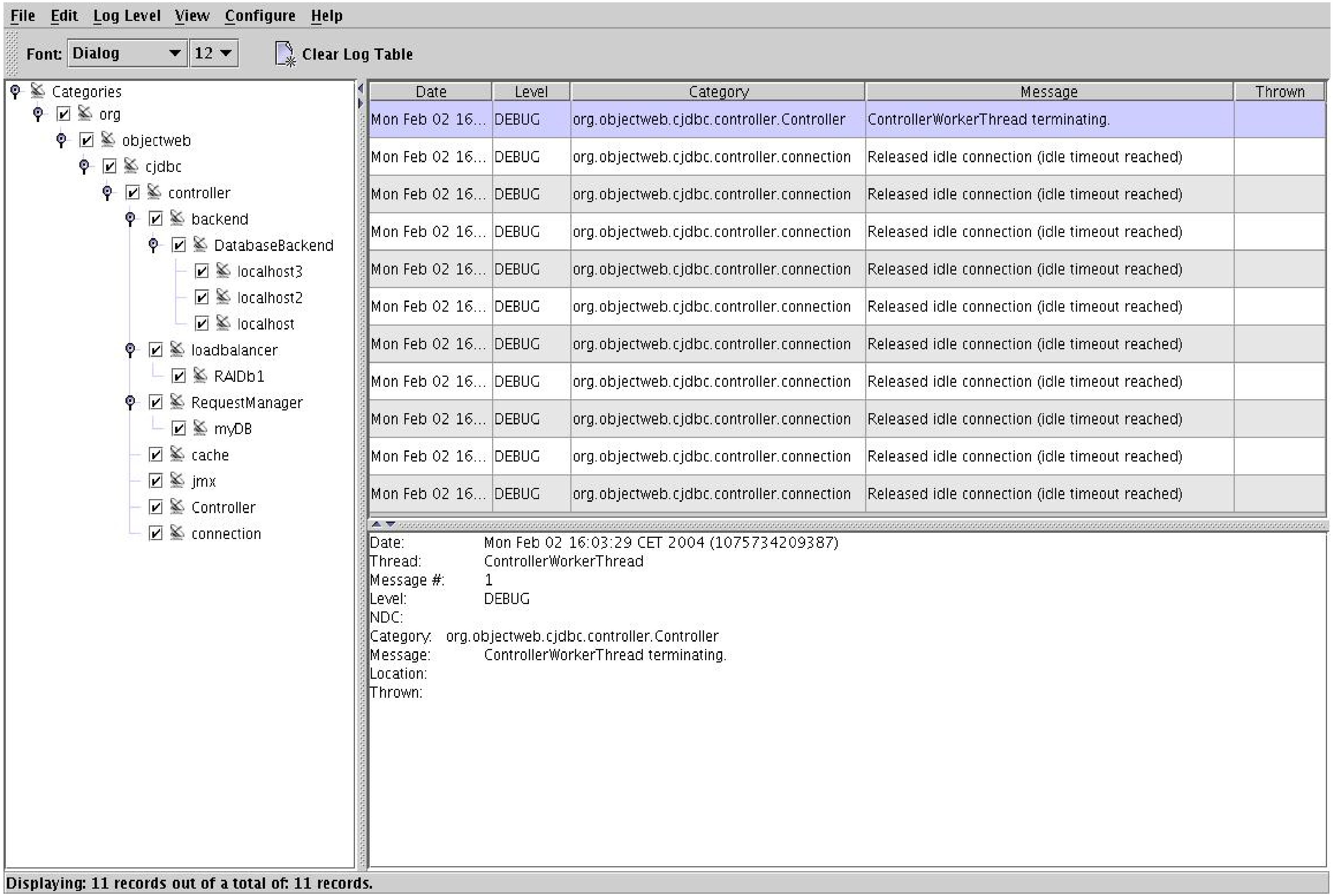
# Design choices

- **JMX for both administration and monitoring**
  - JMX server is MX4J
- **HTTP console for basic features but not suited for monitoring**
- **Swing console for monitoring**
  - uses JMX/RMI connector
- **Currently working on a scalable monitoring infrastructure**

# Online Monitoring



# Viewing logs remotely



| Date | Level | Category | Message | Thrown |
|---|---|---|---|---|
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.Controller | ControllerWorkerThread terminating. | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |
| Mon Feb 02 16... | DEBUG | org.objectweb.cjdbc.controller.connection | Released idle connection (idle timeout reached) | |

```
Date:          Mon Feb 02 16:03:29 CET 2004 (1075734209387)
Thread:        ControllerWorkerThread
Message #:     1
Level:         DEBUG
NDC:
Category:  org.objectweb.cjdbc.controller.Controller
Message:       ControllerWorkerThread terminating.
Location:
Thrown:
```

Displaying: 11 records out of a total of: 11 records.

# Outline

- Overview
- Advanced concepts
- Query caching
- Horizontal scalability
- Monitoring
- Perspectives

# Current status

- C-JDBC 1.0 rc2 release
- Web site: >100.000 hits per month
- Downloads: >1200 per month
- Mailing lists: 134 subscribers

- Ready for production environments

# Ongoing work

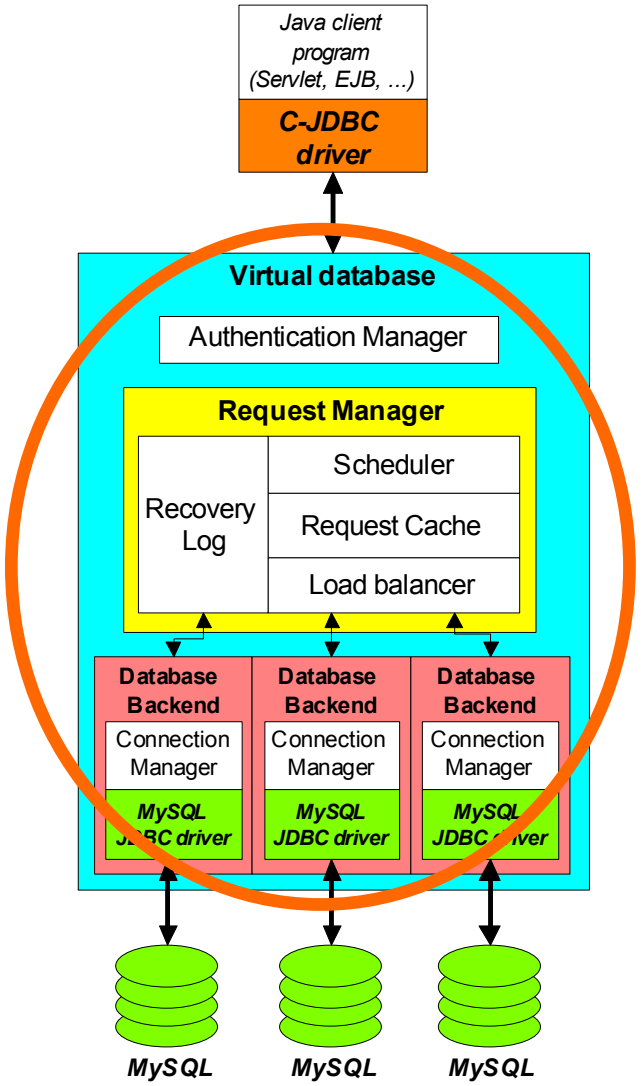
- **Scalable group communication middleware for clusters**
  - Needed for horizontal scalability
  - Decouple application streams from network channels
- **Monitoring**
  - Continuous monitoring stored in repositories
  - Generic monitoring console for both online monitoring and monitoring history browsing
- **Administration**
  - Fully featured administration console
  - Graphical configuration and deployment of centralized/distributed backeds and controllers (offline/online)
  - Dynamic reconfiguration
- **Misc.**
  - Automated Load testing
  - RPM packaging (JPackage)

# Questions & Answers

## Thanks to all users and contributors …

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

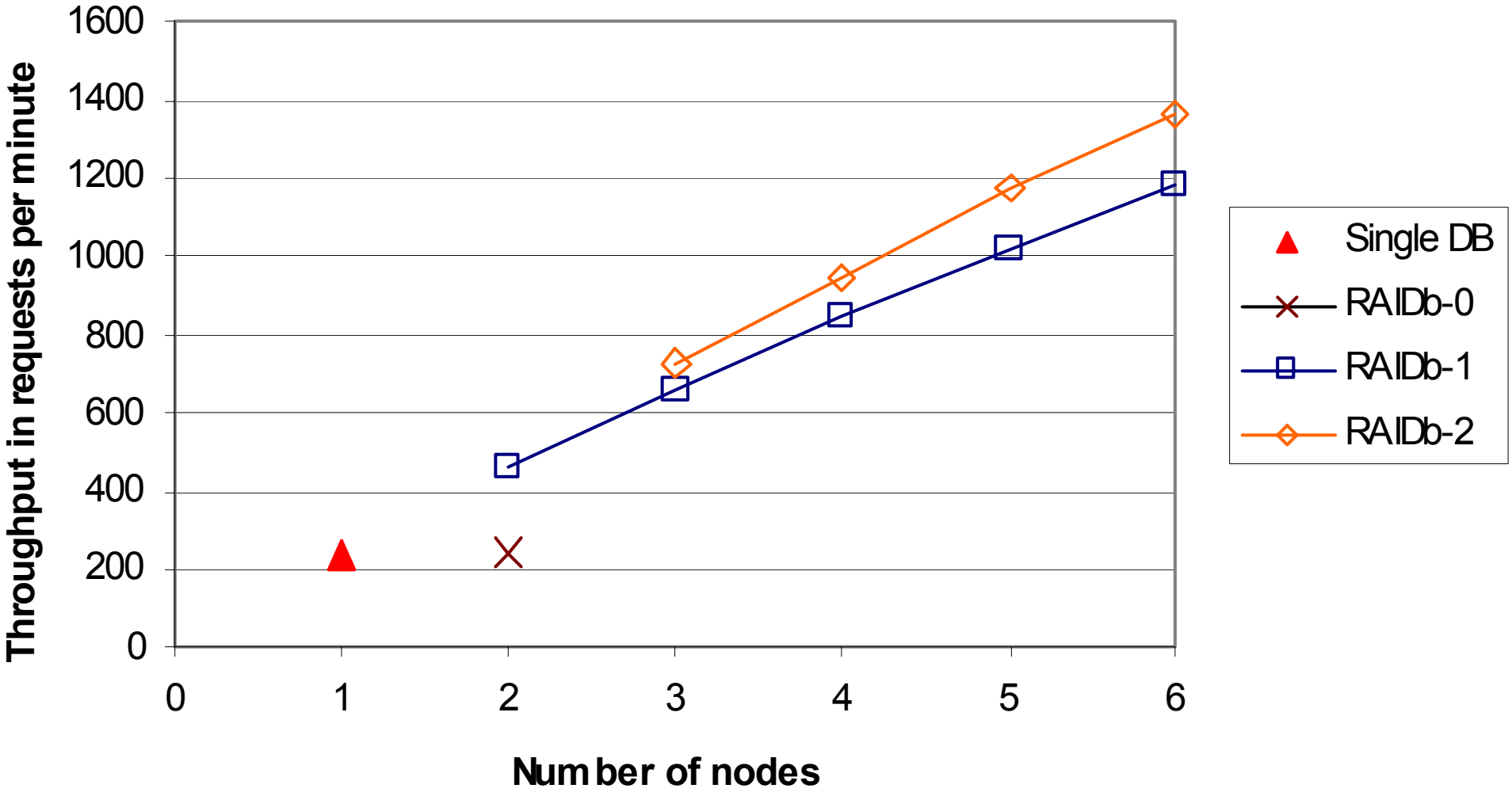
# Virtual Database




- 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

# TPC-W Performance

(Amazon.com)

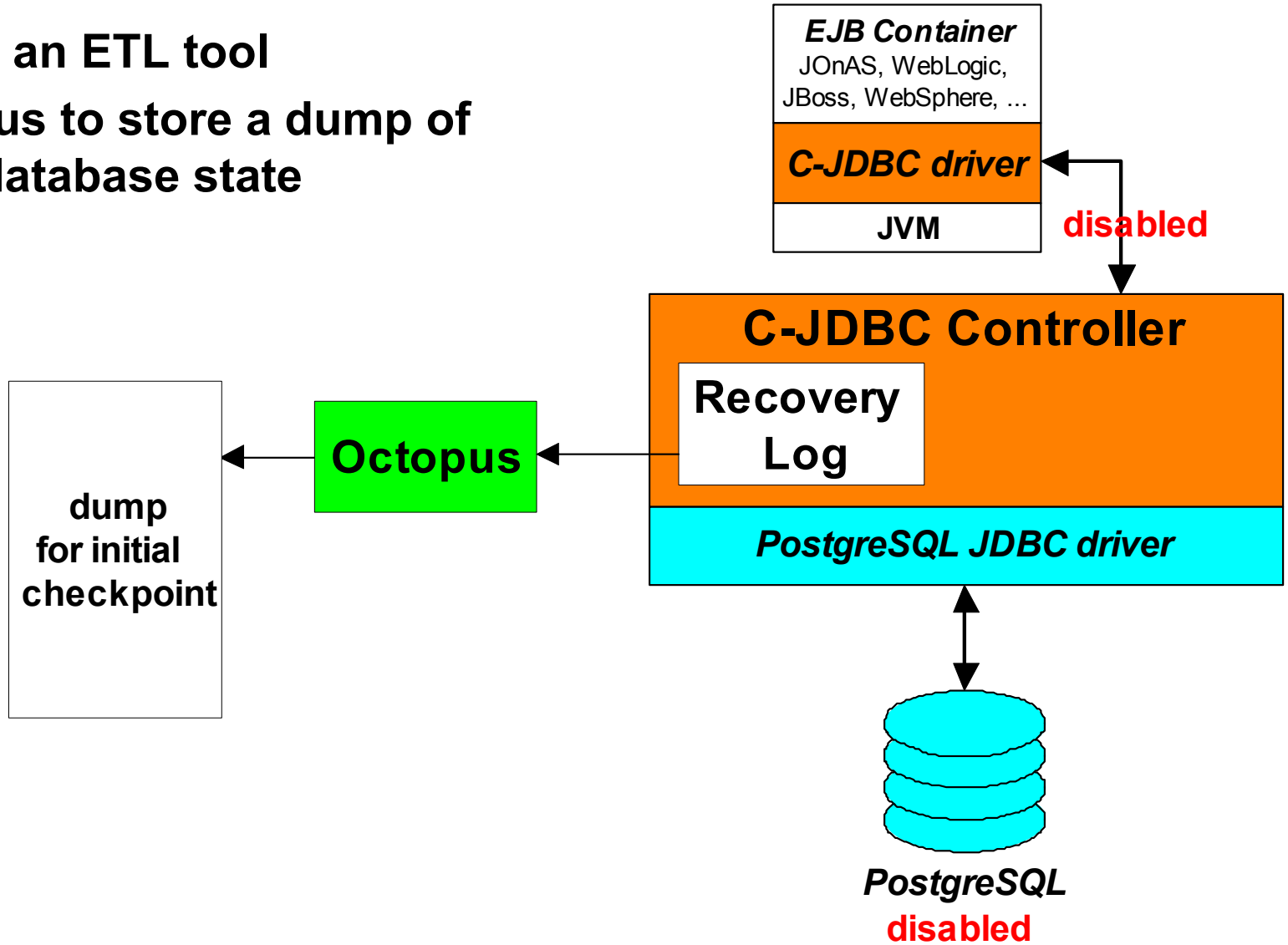


# Outline

- Overview
- Fault tolerance
- Performance
- New features
- Roadmap

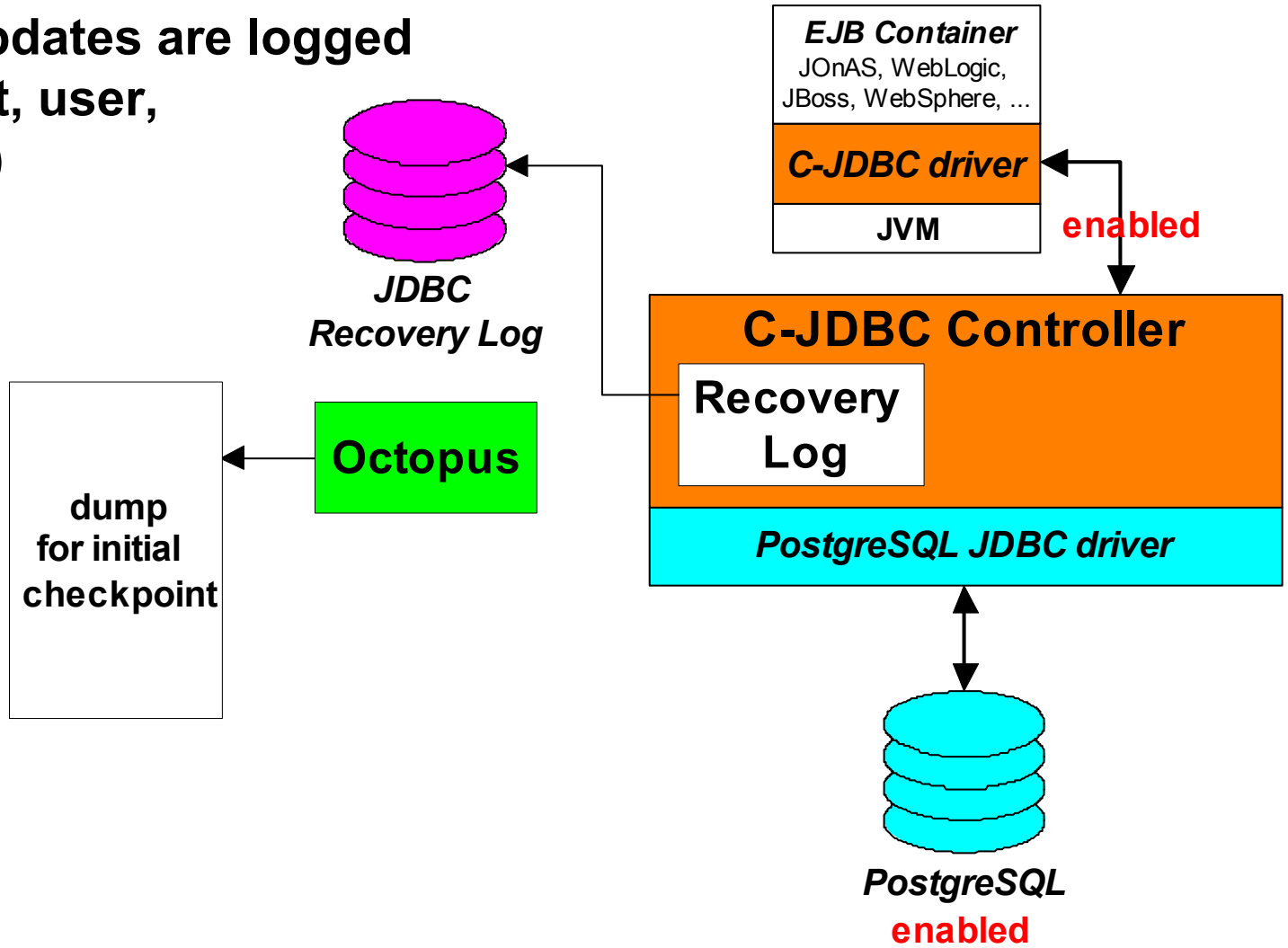
# Building the initial state

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



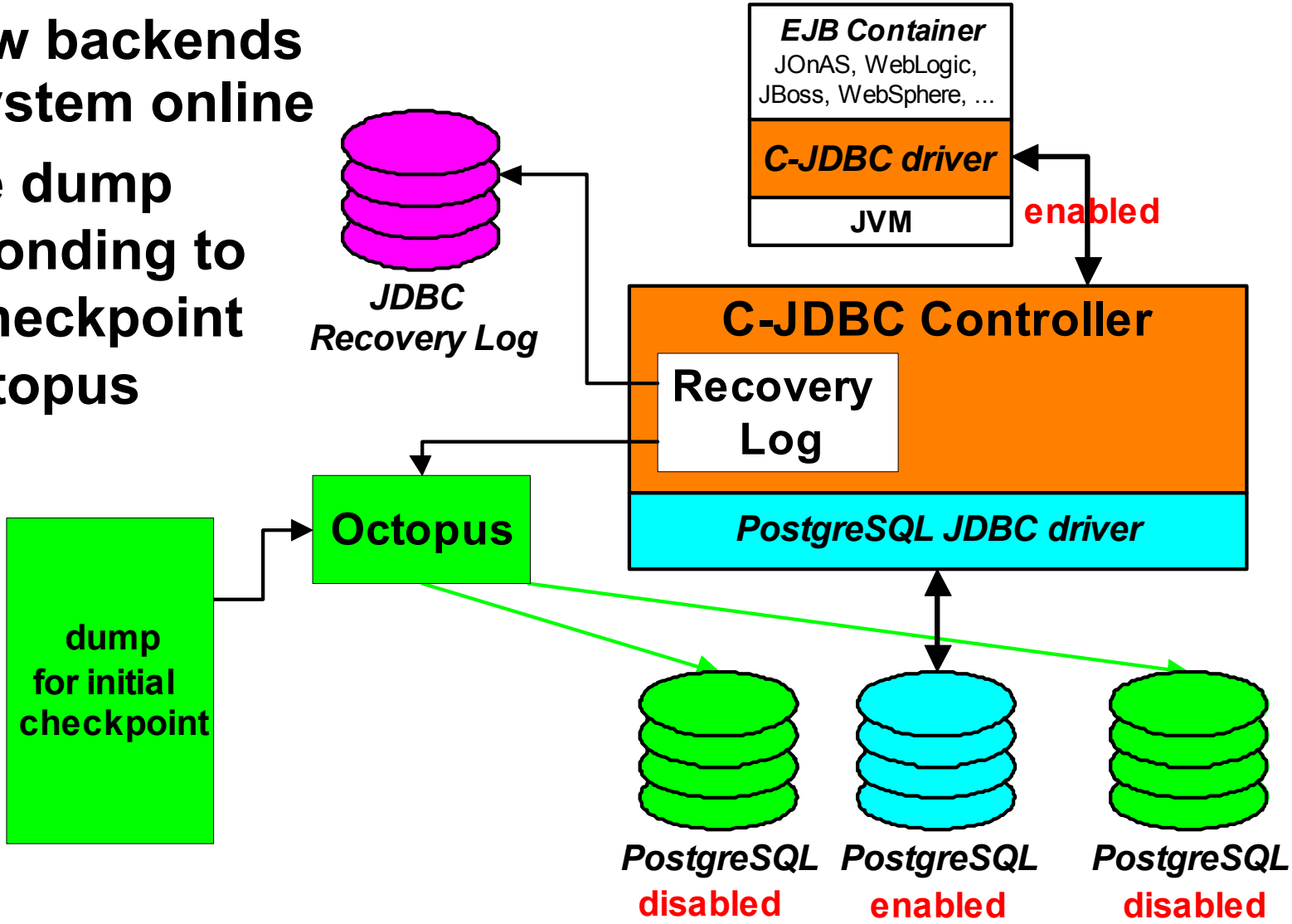
# Journaling

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



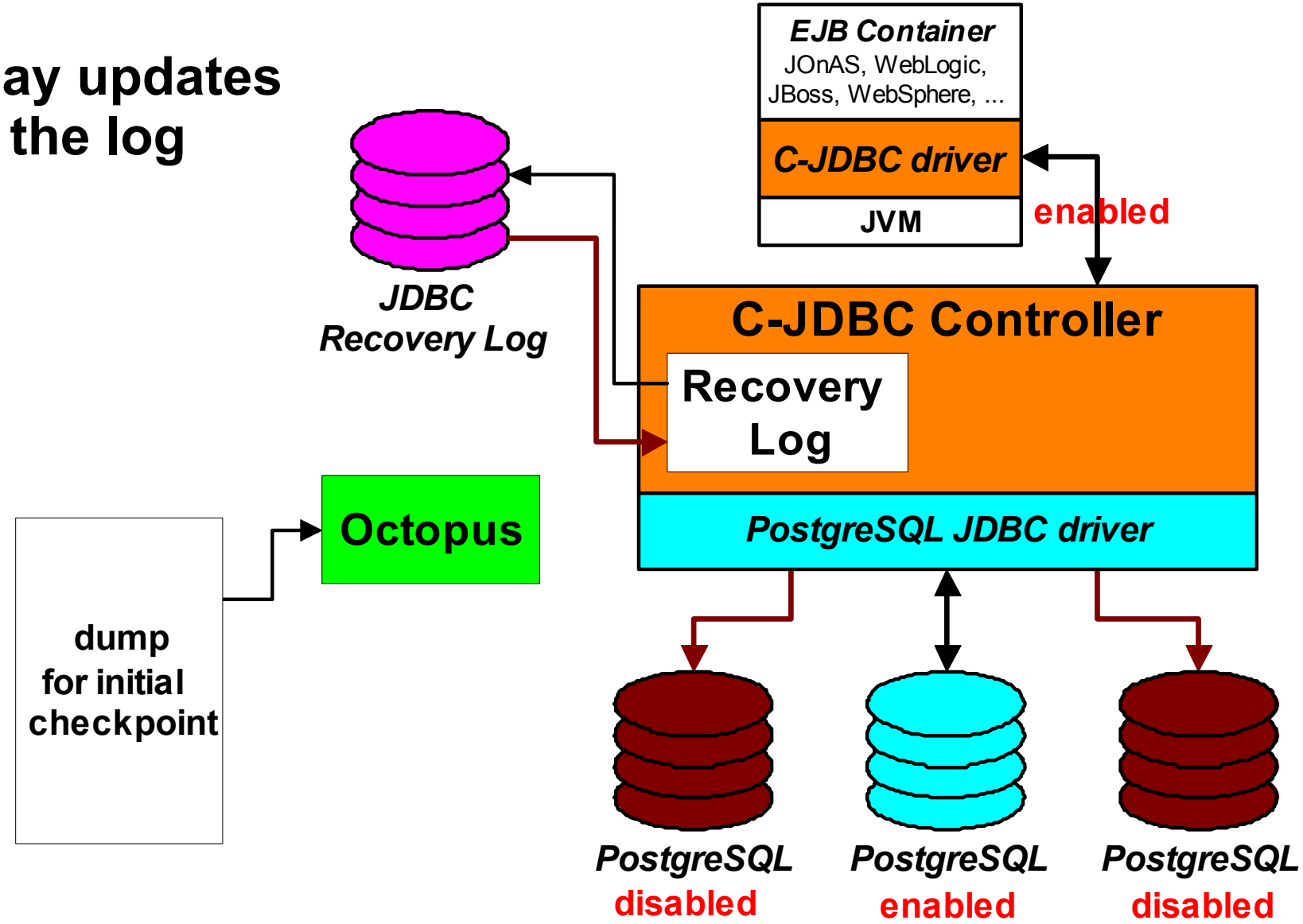
# Adding backend on the fly

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



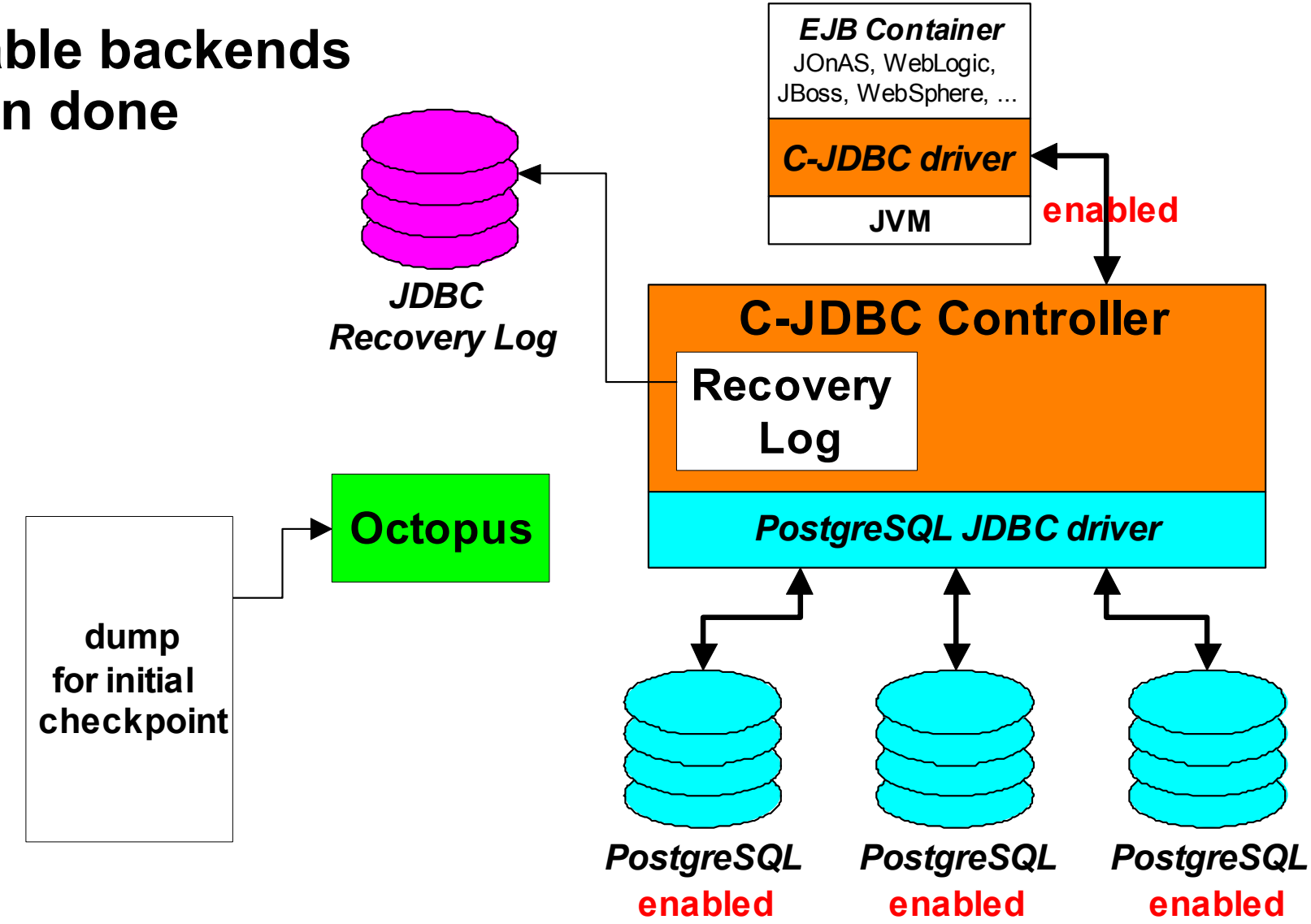
# Synchronizing backends

- Replay updates from the log



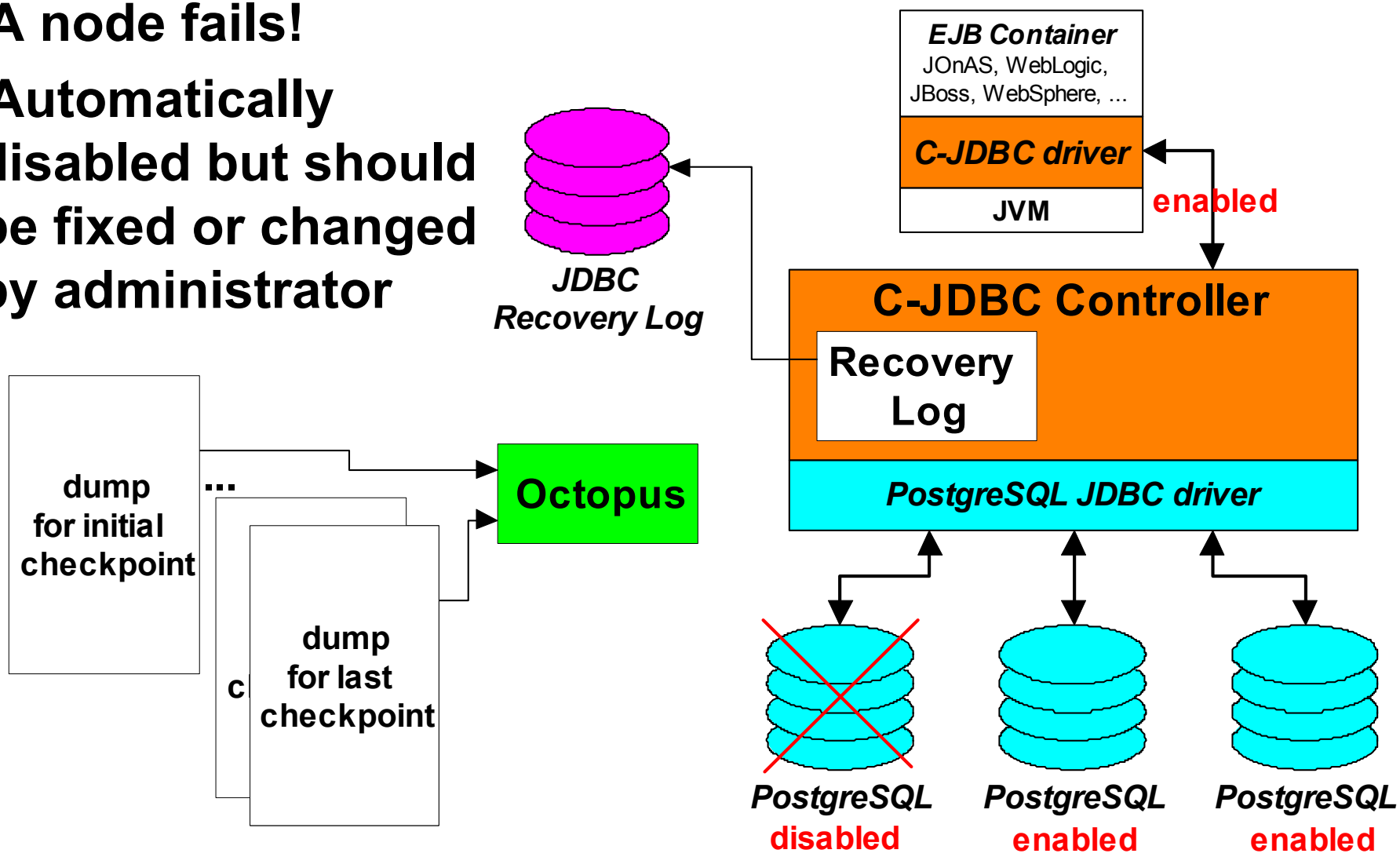
# Expanded Cluster

- Enable backends when done



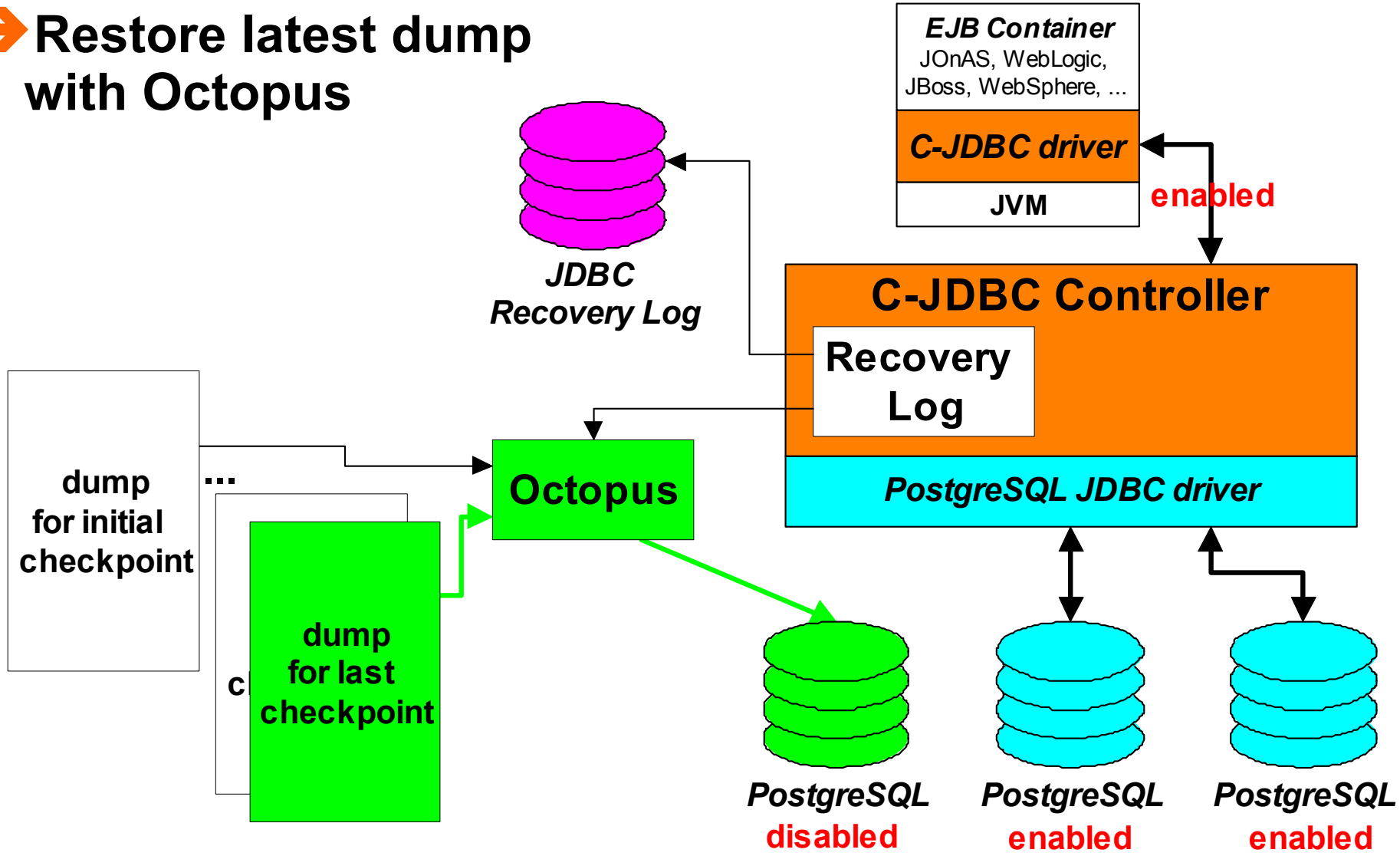
# Handling a backend failure

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



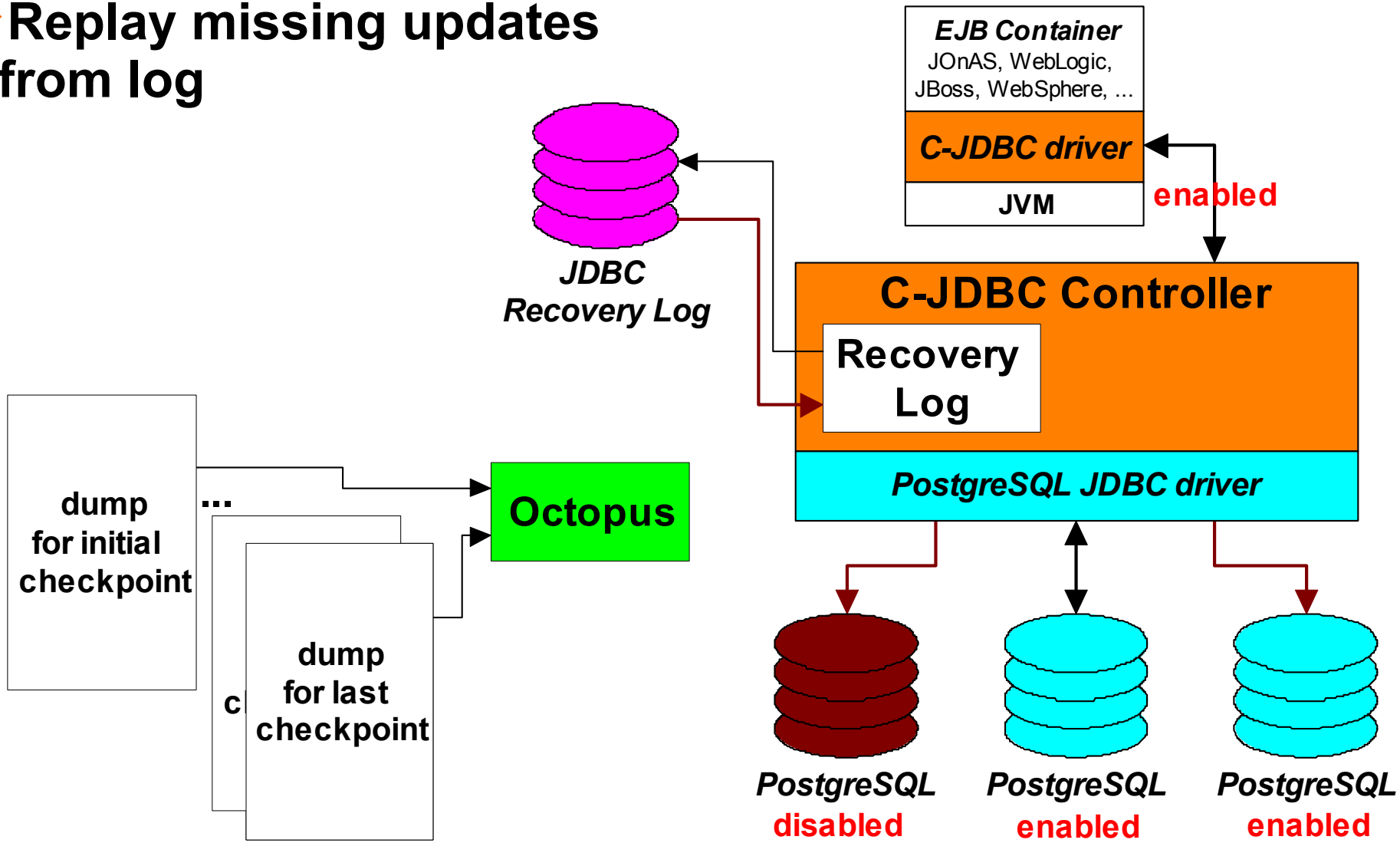
# Restoring a backend

- **Restore latest dump with Octopus**



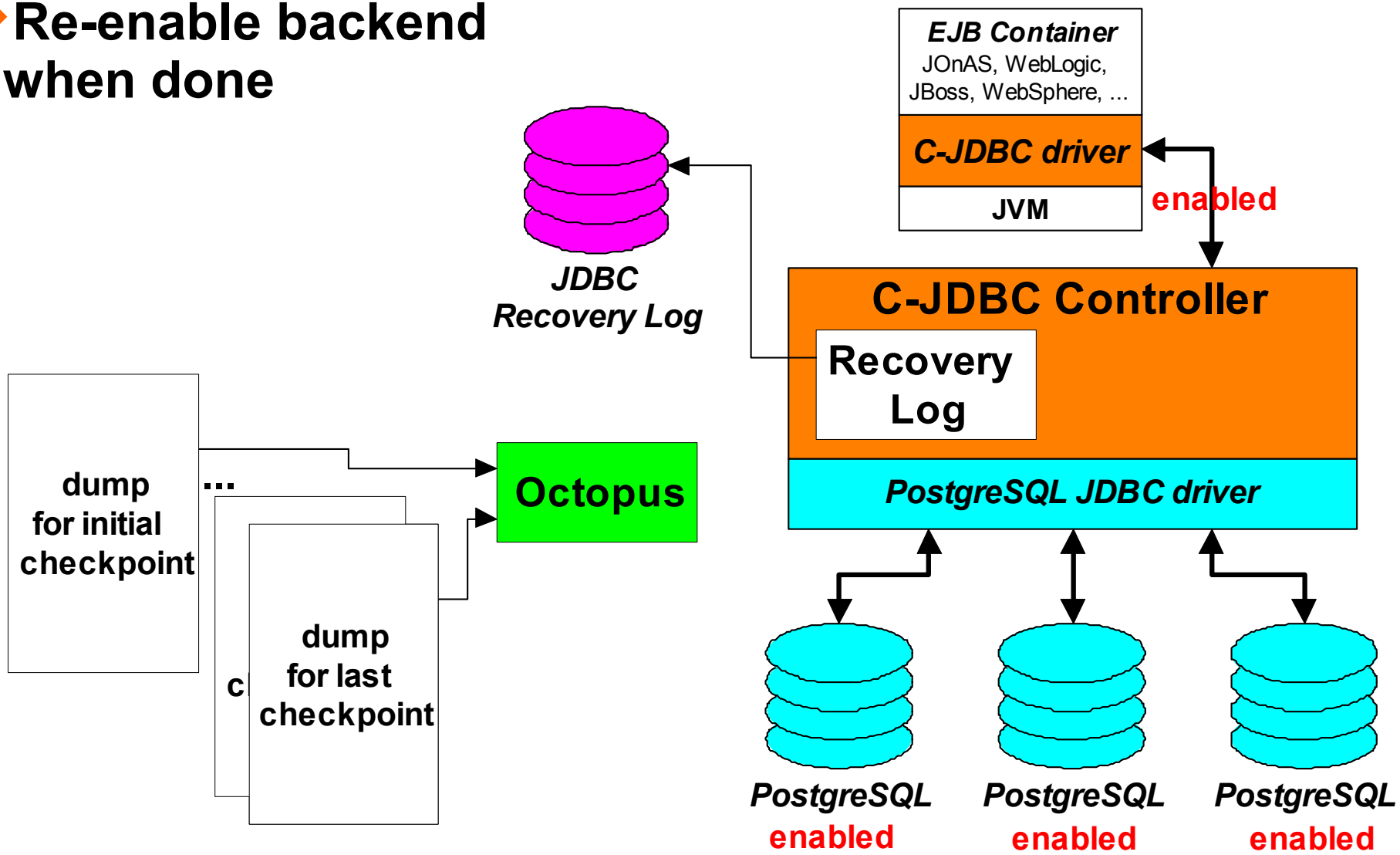
# Re-synchronization

- Replay missing updates from log



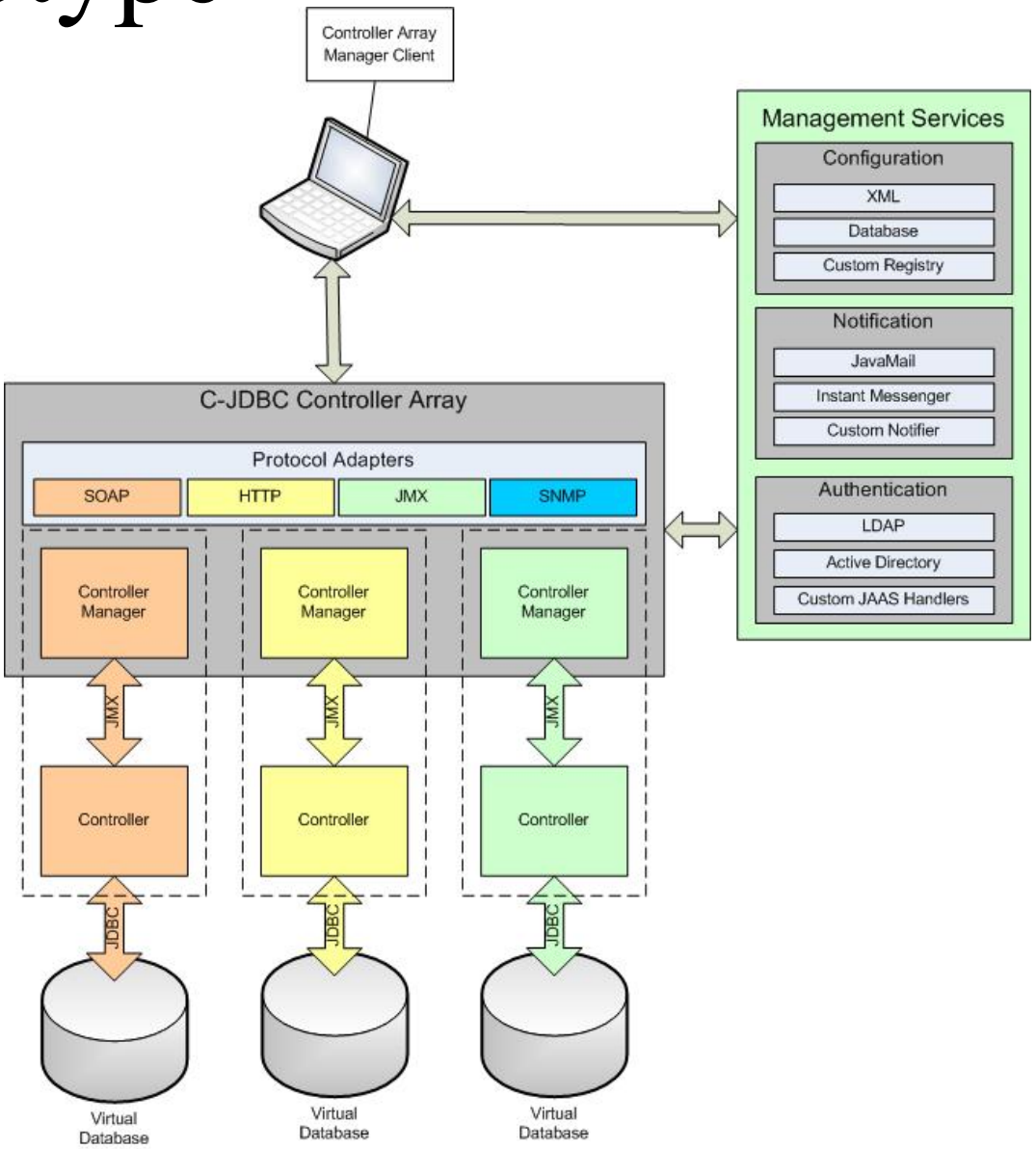
# Healed Cluster

- **Re-enable backend when done**



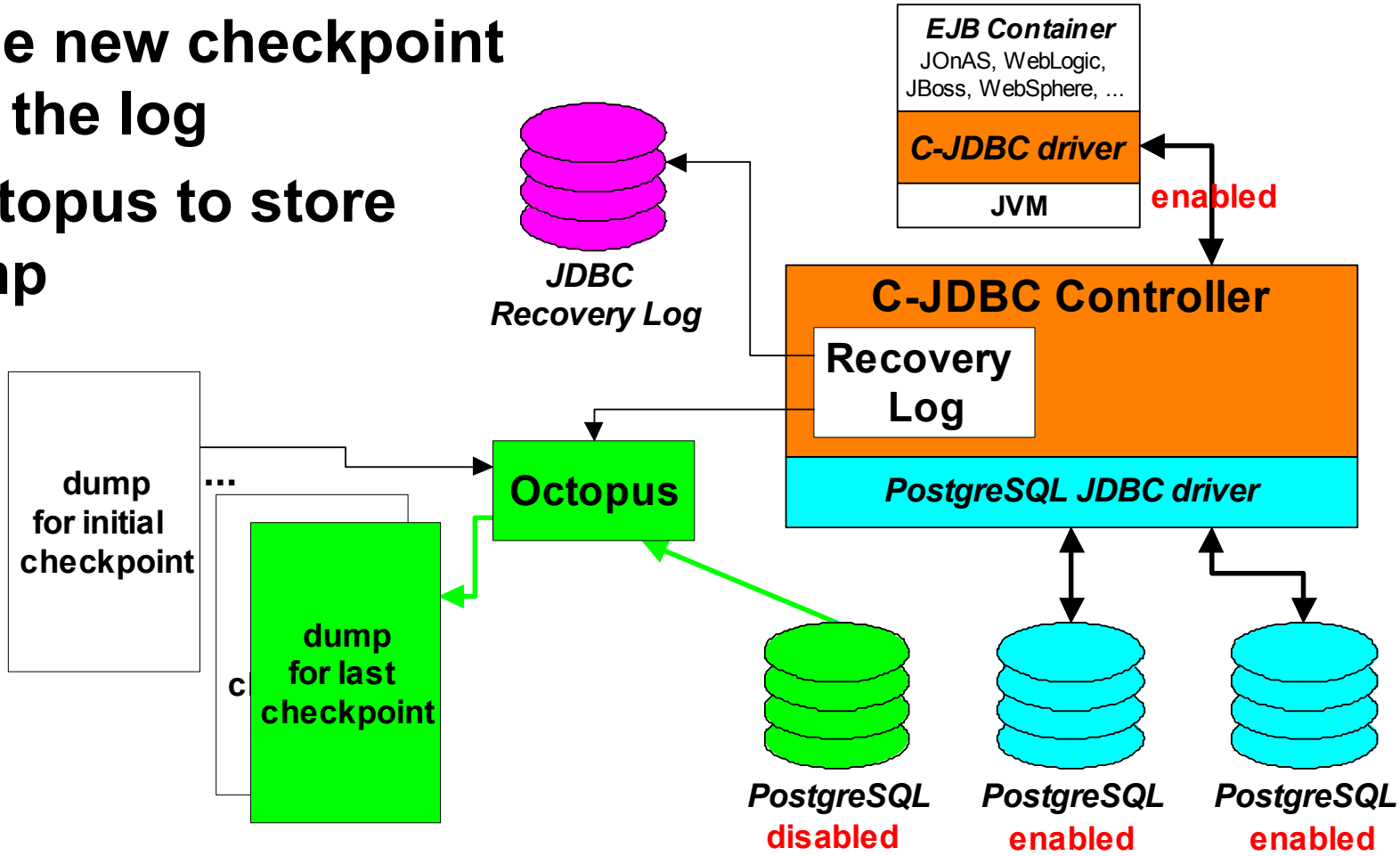
# Prototype

- C-JDBC Management Framework
- Shared design



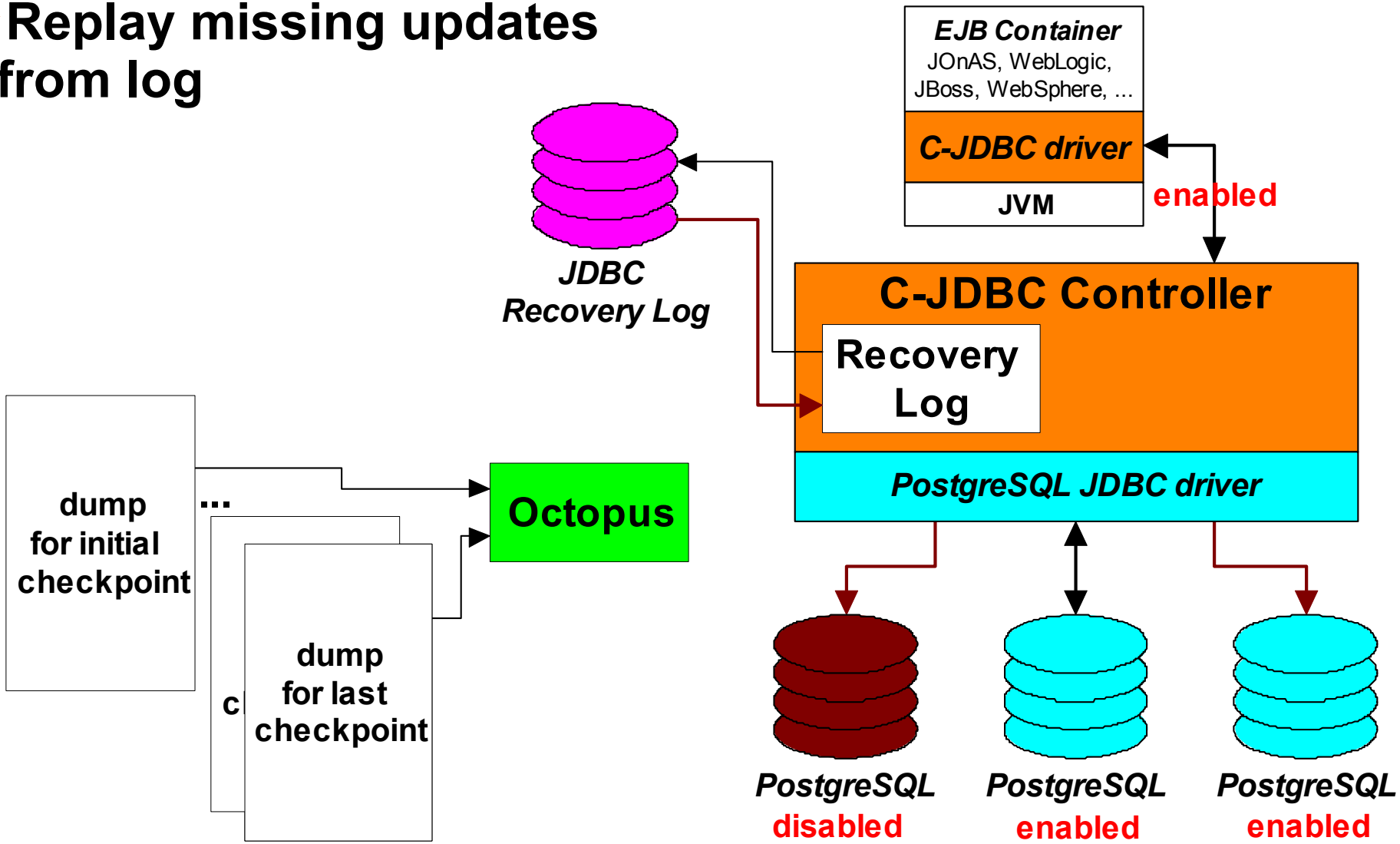
# Making new checkpoints

- 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



# Making new checkpoints

- **Re-enable backend when done**