Our experience of open source

emmanuel.cecchet@inria.fr

ObjectWeb Architecture Meeting, 25 sept 03

INRIA RHÔNE-ALPES

ObjectWeb Open Source Middleware

Outline

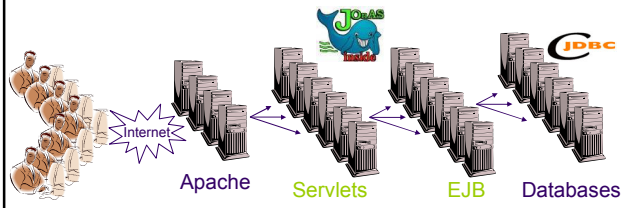
- Why and how?
- C-JDBC community
- Architectural concerns
- Conclusion

Why did we design JDBC ?

- **Scalability evaluation of J2EE servers**
  - performance bounded by database even with a single server
  - how to compare middleware performance ?
  - how to evaluate clustering features in J2EE servers ?
- **Solutions**
  - Large SMP machine: too expensive
  - Open source solution: do it yourself!
- **Features we wanted ordered by priority**
  - scalability
  - on commodity hardware
  - using open source databases
  - fault tolerance (high availability + failover)
  - without modifying the client application

How do we want to use JDBC ?

- ***From* small dynamic content web sites using a centralized open source database**
- ***To* an end-to-end open source solution for large scale J2EE clustered application servers**



Why users are using JDBC?

- **JDBC standard**
- **Open source solution**
- **Features they want ordered by priority**
  - fault tolerance (high availability + failover) [was 4/5]
  - using ~~open source~~ existing databases [was 3/5]
  - on commodity hardware [was 2/5]
  - administration tools [was not]
  - security [was not]
  - scalability [was 1/5]
  - without modifying the client application [was 5/5]

How users are using JDBC?

- **Hard to really know**
- **Just default settings!**
- **Most common usage**
  - existing applications (Tomcat/JBoss/JOnAS) with one MySQL/Postgres backend
  - add a second backend for fault tolerance and scalability
- **For things it was not designed for**
  - write mostly workloads
  - distributed databases
  - hosting centers (administration tools missing)

Lessons learned

- Users do not use it for what it was first designed for
  - advanced features are never used
  - concerned about ease of use and TCO
- Default settings are important
- Good technology is necessary but not sufficient
  - [administration] tools are needed
  - bugs are ok

Outline

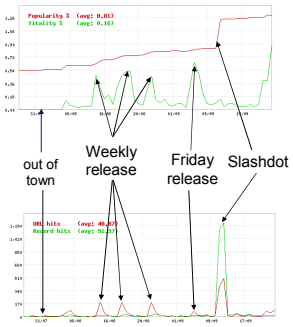
- Why and how?
- C-JDBC community
- Architectural concerns
- Conclusion

Stats as of Sep 25, 03

- Downloads
  - total : > 5200 downloads since may 2003
  - last 30 days : > 1700 downloads
  - 2nd most downloaded ObjectWeb project
- Mailing lists
  - c-jdbc@objectweb.org: 93 subscribers
  - c-jdbc-commits@objectweb.org: 16 subscribers
- Web site
  - > 100.000 hits for september 03
  - US 35%, EU 25%, Canada 5%, Australia 5%, China 3%, Brazil 2%, India 2%, Japan 2%, …

Freshmeat.net

- Links to projects
- Users can subscribe to be notified of new releases
- Need to register project in all possible categories to have good visibility
- One release per week is a good timing
- 3 new subscribers with every release



How do we build a community?

- Necessary features (but not sufficient)
  - open source
  - standard API
  - responsiveness on the mailing list
- Visibility
  - Web: slashdot, TheServerSide, freshmeat, …
  - Conferences: JAX, Middleware, LinuxWorld, ICAR, …
- Our weak points
  - no detailed design documentation
  - beta phase

How do we interact with the user community?

- only one mailing list
- being very responsive on the mailing list
  - reply even if we don't have a response yet
  - no direct communication with team but share everything on the mailing list
- benefit from engineers who work 1 week full-time to evaluate C-JDBC for their corporation
- plan every feature request in the task list

How do we interact with the developer community?

- single user/developer mailing list
- post all design questions/choices on the mailing list
  - most users use default settings
  - hard to get feedback about usage
- very permissive to accept new commiters
  - 8 committers (3 outside ObjectWeb – 2 full time)
  - 2 contributors who didn't want to become commiters
  - no problem so far
- involve people in testing

Lessons learned

- Visibility
  - perpetual involvement
  - time consuming but necessary
- Responsiveness to user queries
  - always on the mailing list
  - makes first impression for many users
- Involve users in all decisions
- Be open
  - source, CVS, contributions, patches, …

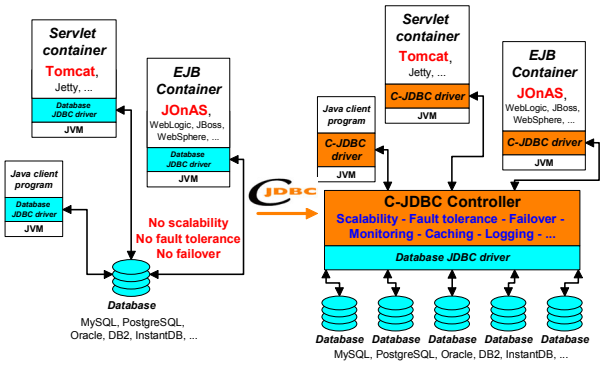
Outline

- Why and how?
- C-JDBC community
- Architectural concerns
  - C-JDBC overview
  - Octopus integration
  - Clustering
  - Open problems
- Conclusion

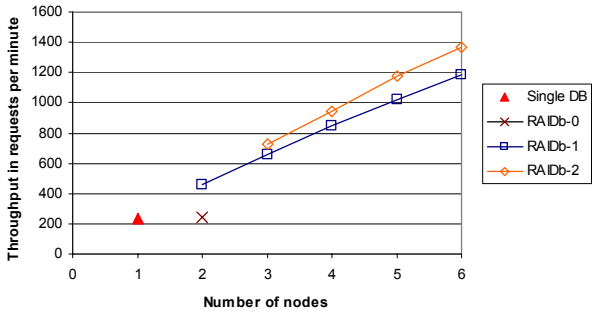
JDBC quick facts

- Redundant Array of Inexpensive Databases (RAIDb)
  - RAIDb-0: full partitioning
  - RAIDb-1: full replication
  - RAIDb-2: partial replication
- Two components
  - generic JDBC 2.0 driver (C-JDBC driver)
  - C-JDBC Controller
- C-JDBC Controller provides
  - scheduler for concurrency control
  - load balancer for performance
  - recovery log for fault tolerance
- Supports heterogeneous databases

Overview of JDBC



TPC-W Performance

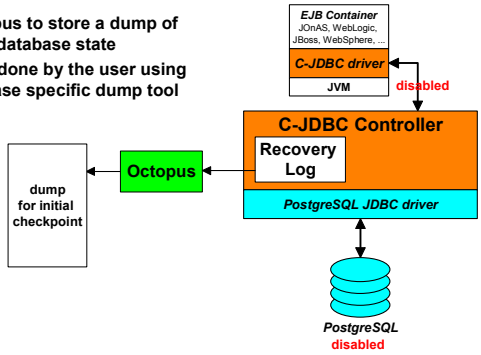


Outline

- Why and how?
- C-JDBC community
- Architectural concerns
  - C-JDBC overview
  - Octopus integration
  - Clustering
  - Open problems
- Conclusion

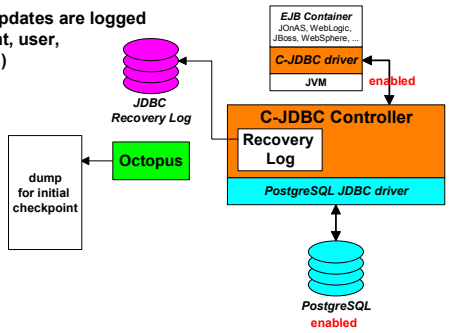
Octopus integration

- Use Octopus to store a dump of the initial database state
- Currently done by the user using the database specific dump tool



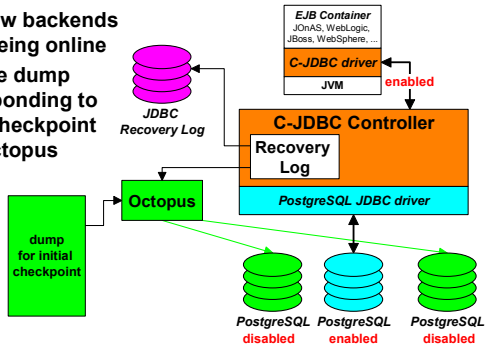
Octopus integration

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



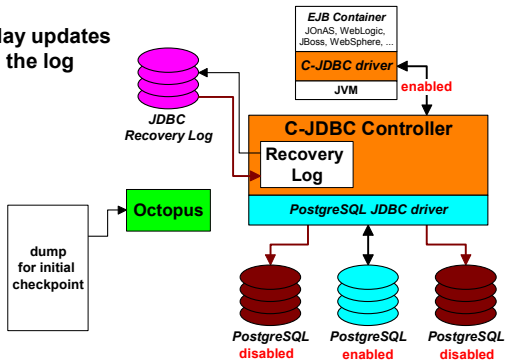
Octopus integration

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



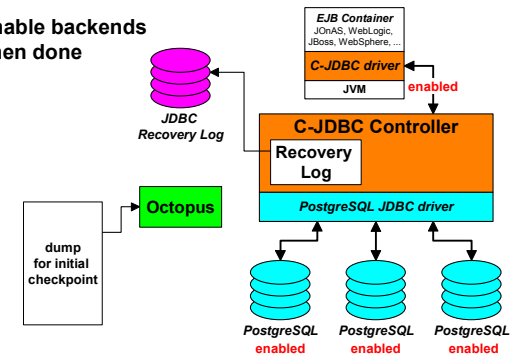
Octopus integration

- Replay updates from the log



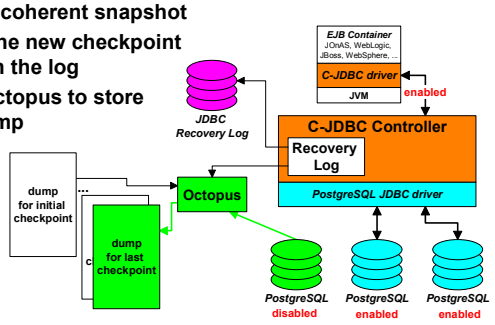
Octopus integration

- Enable backends when done



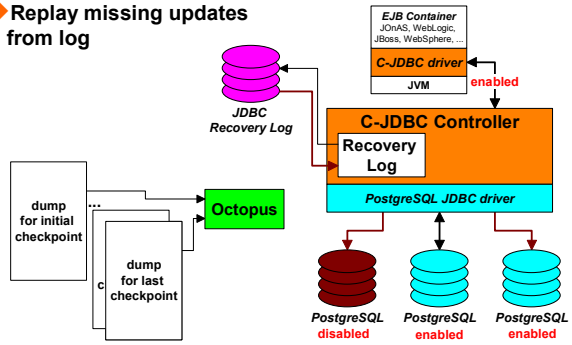
Making new checkpoints

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



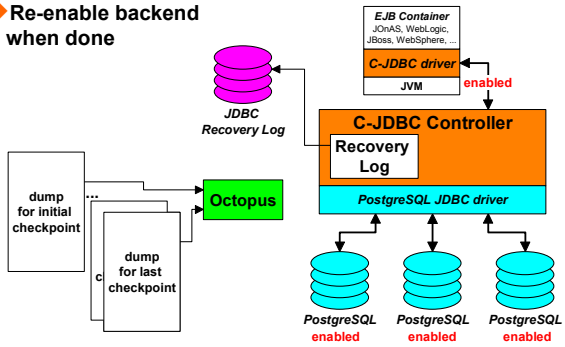
Making new checkpoints

- Replay missing updates from log



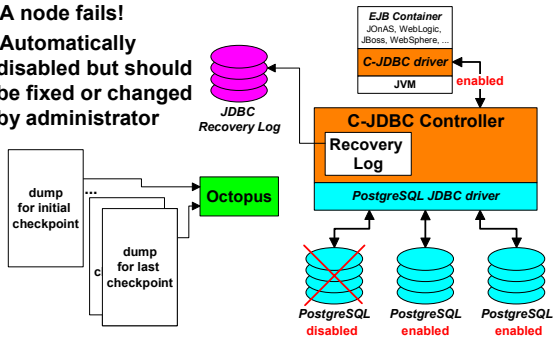
Making new checkpoints

- Re-enable backend when done



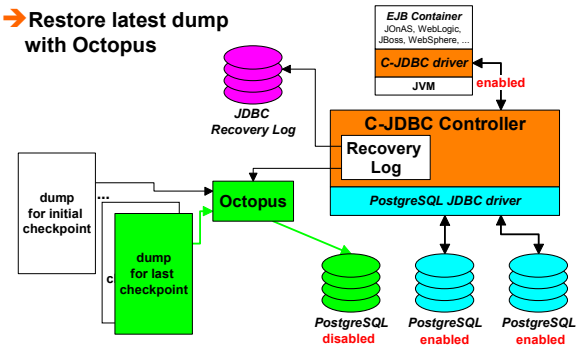
Fault tolerance

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



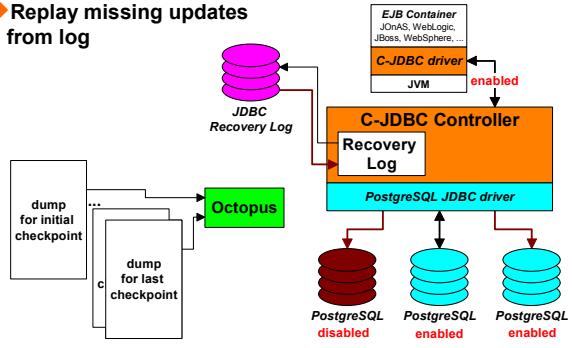
Fault tolerance

- Restore latest dump with Octopus



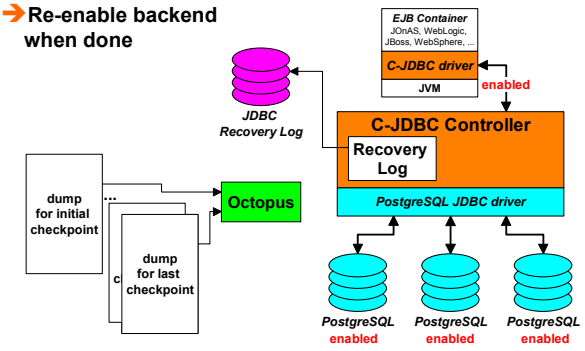
Fault tolerance

- Replay missing updates from log



Fault tolerance

- Re-enable backend when done

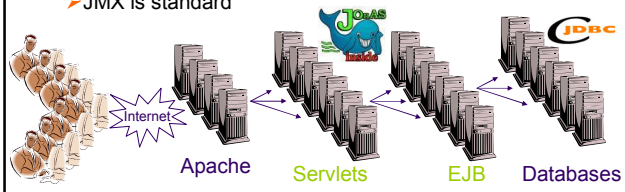


Outline

- Why and how?
- C-JDBC community
- Architectural concerns
  - C-JDBC overview
  - Octopus integration
  - Clustering
  - Open problems
- Conclusion

Clustering

- JavaGroups everywhere
  - performance and scalability study needed
- Administration
  - integration of various consoles (Tomcat, JOnAS, …)
  - common vocabulary needed
  - JMX is standard



Clustering continued …

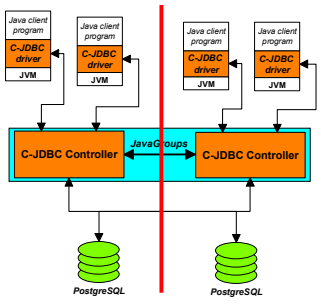
- Monitoring framework needed
  - health of the system
  - building efficient load balancing algorithms
- JMX performance impact?
  - resource usage
  - design patterns
  - results to be published in JMOB
- User console
  - text, Swing and/or HTTP ?
  - SQuirreL SQL or iSQL for JDBC console

Outline

- Why and how?
- C-JDBC community
- Architectural concerns
  - C-JDBC overview
  - Octopus integration
  - Clustering
  - Open problems
- Conclusion

Open problems

- Partition of clusters
- Users want control on failure policy
- Reconciliation must also be user controlled



Open problems

- Opening the architecture to the users
  - user defined strategies when a fault or exception occurs
  - which interfaces/callbacks to provide ?
- Monitoring
  - needed for more accurate load balancing algorithms
- Benchmarking
  - need automatic evaluation of clustered servers
  - platform available: new INRIA 208 itanium-2 cluster
- Sun Test Suite
  - should help strenghening C-JDBC code
  - interoperability with J2EE servers

Fractal ?

- Fractal version of C-JDBC
- Benefits for C-JDBC
  - short term: good question !
  - long term: resource usage tracing
- Benefits for Fractal
  - feedback
  - performance comparison with POJO version

Outline

- Why and how?
- C-JDBC community
- Architectural concerns
- Conclusion

Conclusion

- Integration of “coarse grain” ObjectWeb components
- Standard APIs (not OW defined) allows easy interaction with other components (OW or not)
- Common tools needed for administration and monitoring

Heterogeneity support

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