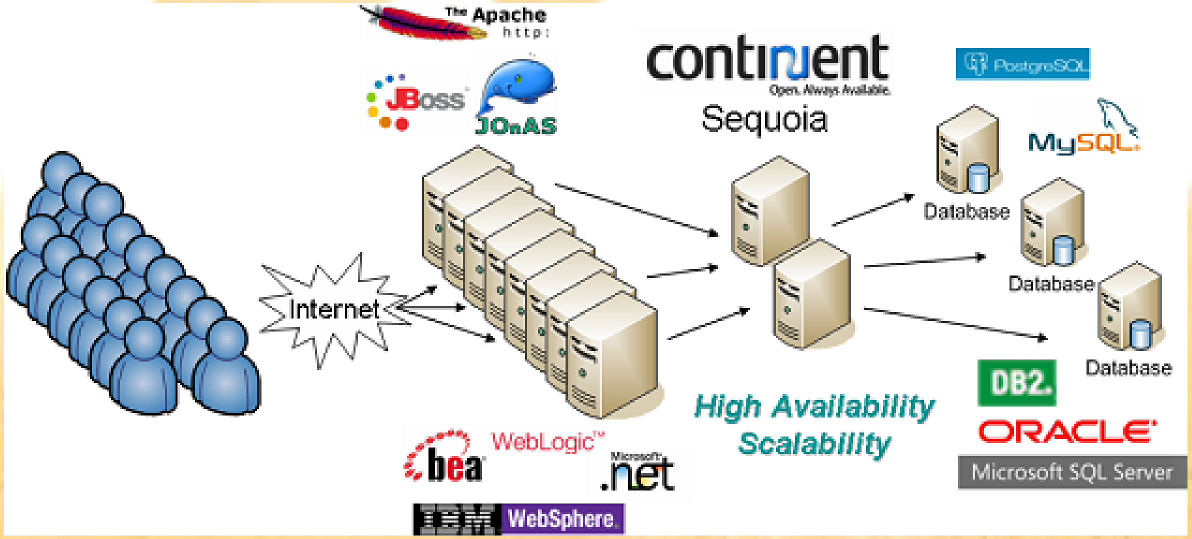
Distributed Event Metadata System (SQL) using

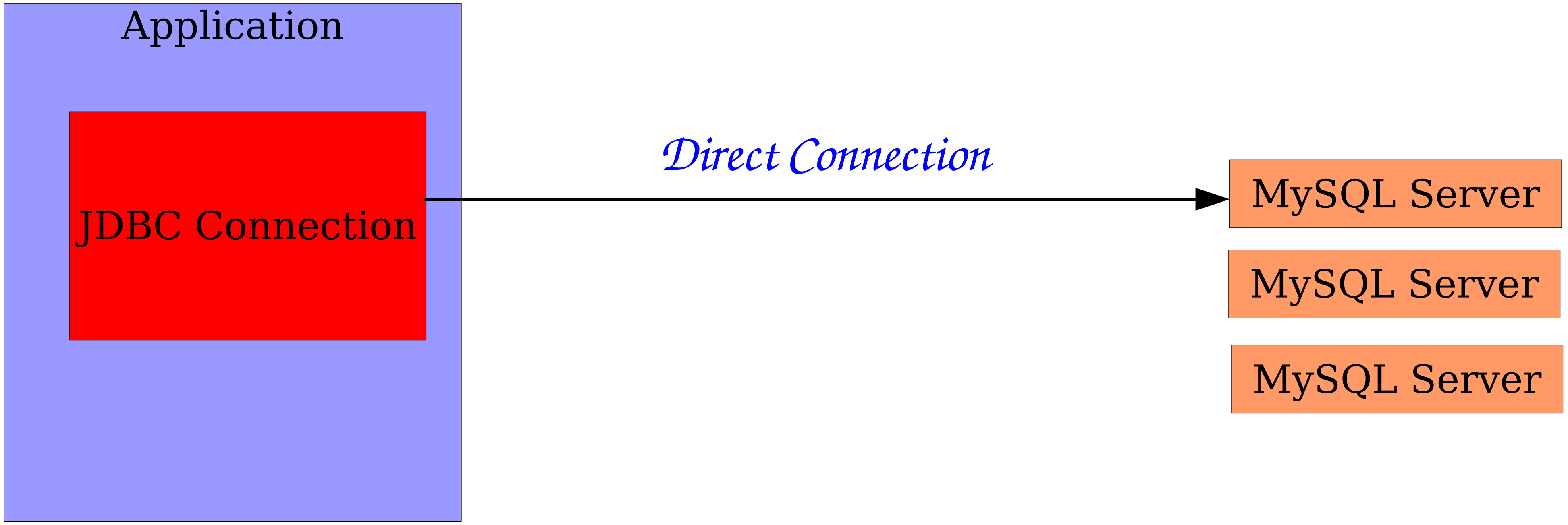
Sequoia



- Sequoia:
  - Architecture
  - Standard Features
  - Plans for Plugins
  - Tools
- Carob
- Distributed Architecture

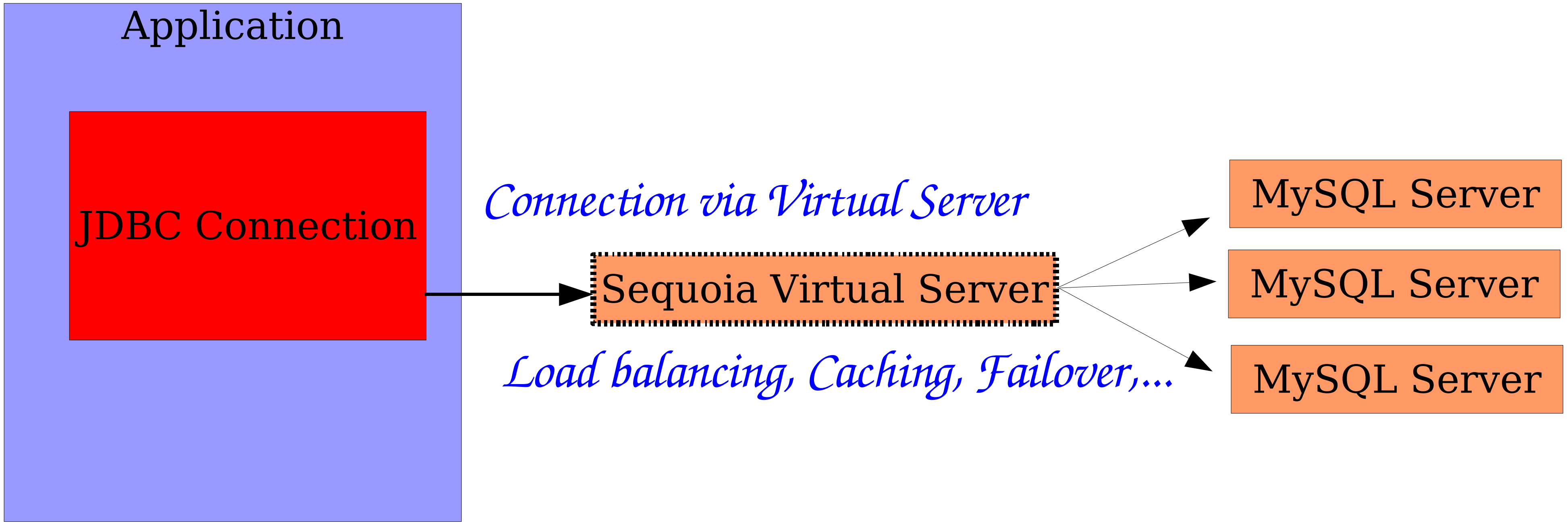
Sequoia Architecture

```
// Direct connection to MySQL server
Connection connection = DriverManager.getConnection("jdbc:mysql://mysqlserver.cern.ch/Tuples",
                                                    "user", "passwd");
```



Sequoia Architecture

```
// Direct connection to MySQL server
Connection connection = DriverManager.getConnection("jdbc:mysql://mysqlserver.cern.ch/Tuples",
                                                     "user", "passwd");
```

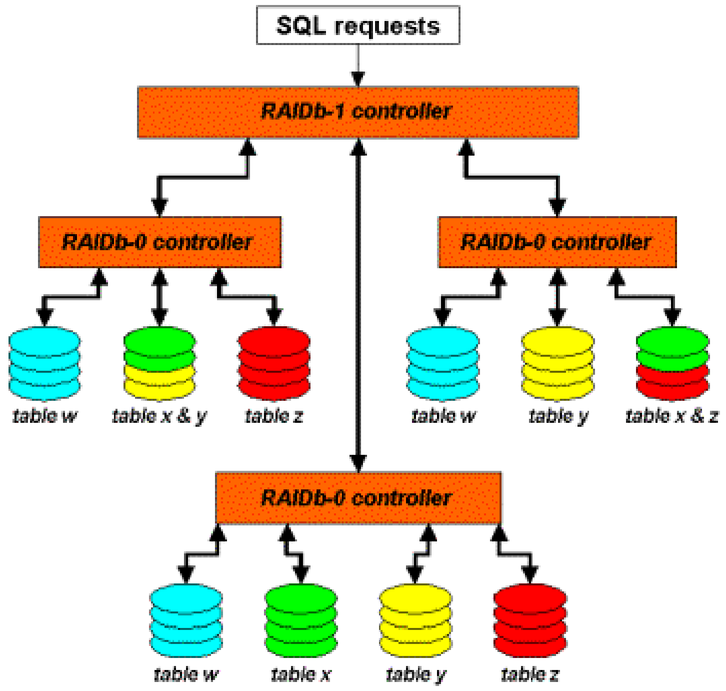


```
// Connection via Sequoia virtual server
Connection connection = DriverManager.getConnection("jdbc:cjdbc://seuoiaserver.cern.ch/Tuples",
                                                     "user", "passwd");
```

The only change in the Application (can be configured via Job Options).

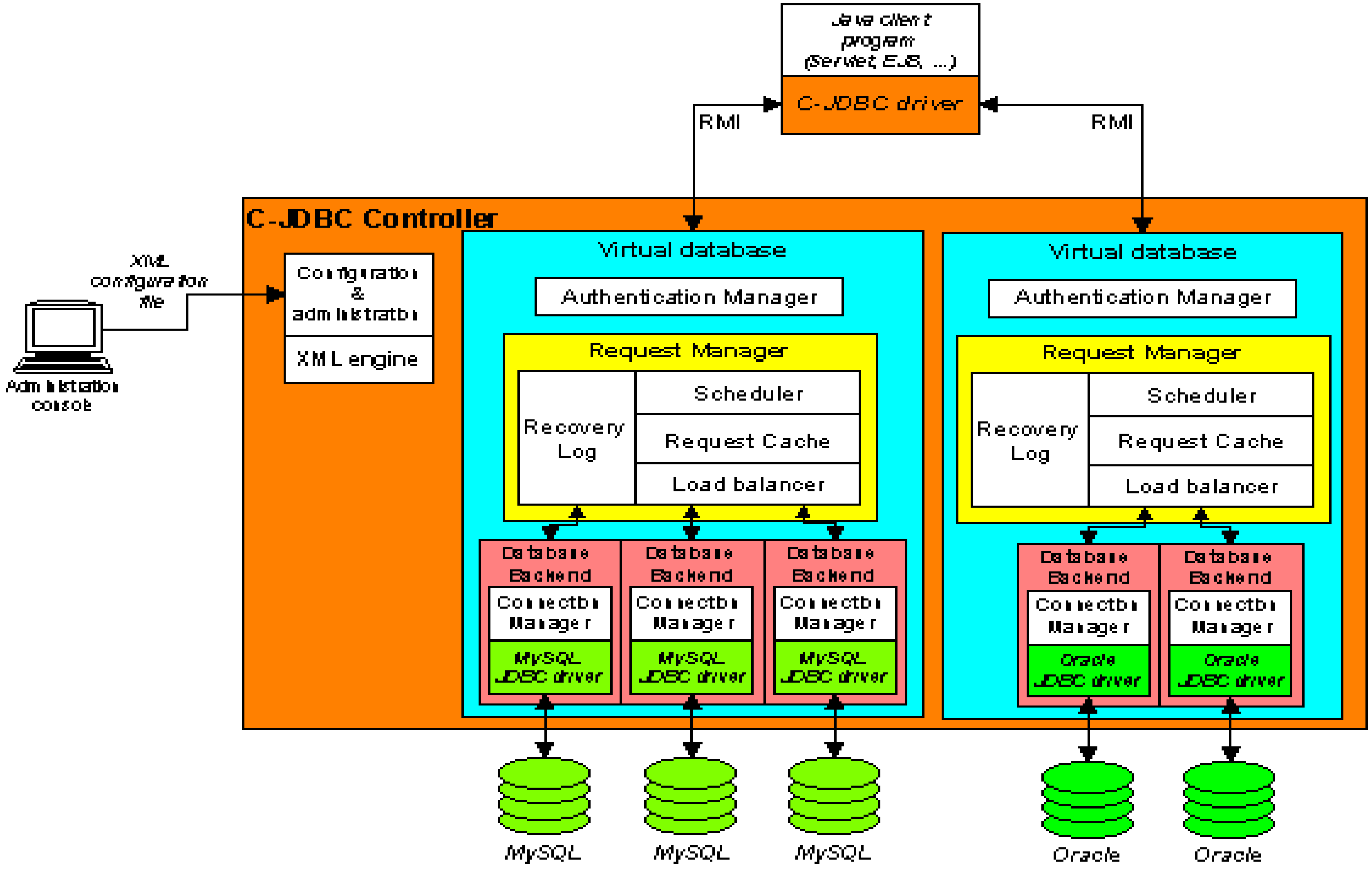
Sequoia Architecture

- *SQL tables can be spread on several database Servers, some tables may be replicated. User wants a single front-end.*
- *Sequoia acts as a (Proxy) Virtual SQL Server forwarding all requests to appropriate databases (real or another virtual). Replicated and/or complementary tables are supported (even on heterogeneous Servers), similar do RAID disks.*
- *Sequoia is used via its JDBC driver, so any application using JDBC API can directly use Sequoia. No application modification is required to use Sequoia.*



Sequoia is continuation of C-JDBC.

Sequoia Architecture



Standard Sequoia Features

- *<u>Load balancing</u>: Several strategies are available (round-robin, round-robin with weights, adaptable round-robin), others can be introduced.*
- *<u>Caching</u>: Results of SQL queries are cached, depending on chosen strategy.*
- *<u>Connection Pooling</u>: Connections are reused at the level of Sequoia Server.*
- *<u>Failover</u>: Two kinds of Server replication are available:*
  - *<u>Horizontal Scaling</u>: User connects to a group of Sequoia Servers, where at least one should be available.*
  - *<u>Vertical Scaling</u>: Servers with tables replicas are hidden behind one SequoiaServer.*
- *<u>Backup/Restore</u>: Tables or whole database can be backuped or replicated (using Enhydra Octopus).*
- *<u>Journaling/CheckPointing</u>: Database transactions are recorded and saved on request for later recovery.*
- *<u>Monitoring</u>: All transactions are monitored to allow performance tuning.*
- *<u>Replication</u>: Writing updates all replicas.*
- *<u>Authentication</u>: Sequoia Server maps user credentials to all backend Servers.*

Plans for Plugins

- *<u>Parallel Processing</u>: The data are spread over several tables and servers and accessed transparently as one table. Partitioned tables.*
- *<u>Query Prediction</u>: Cached query results are used to predict future query result, or at least an estimation of needed time.*
- *<u>Adaptive Indexing and Replication</u>: Monitoring information is used to tune databases for performance.*
- *<u>Query filtering</u>: User Queries are analyzed and optimized (or refused if wrong).*
- *…*

Development is going on in ObjectWeb.

Sequoia Tools

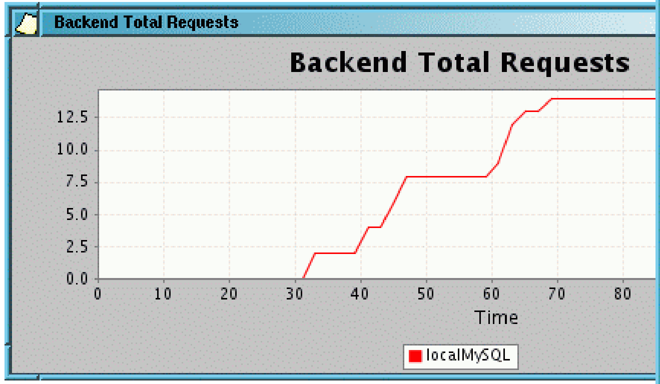
- *Sequoia Configuration is specified via an XML file.*
- *Operations can be managed by:*
  - *GUI*
  - *Command line (and scripts)*
  - *Code*

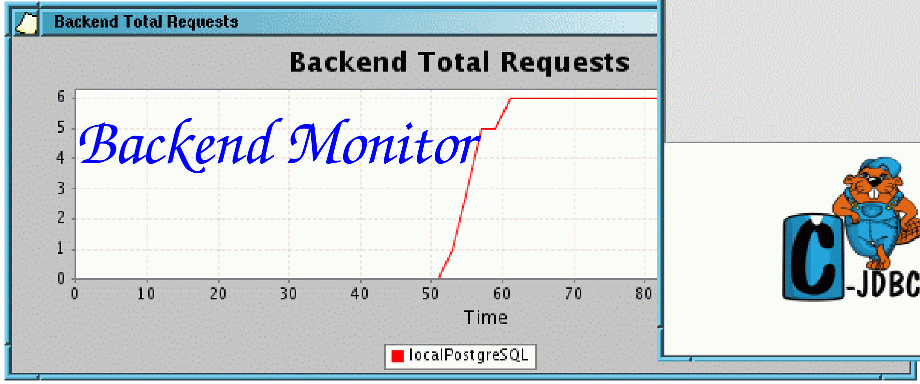
```
<DatabaseBackend name="cernOracle"
                          driver="oracle.jdbc.driver.OracleDriver"
                          driverPath="/opt/Oracle/ojdbc14_g.jar"
                          url="jdbc:oracle:thin:@oradev9.cern.ch:1521:D9"
                          connectionTestStatement="select * from dual">
     <ConnectionManager vLogin="test" rLogin="user" rPassword="password">
             <VariablePoolConnectionManager initPoolSize="40"/>
             </ConnectionManager>
     </DatabaseBackend>

<RequestManager beginTimeout="0" commitTimeout="0" rollbackTimeout="0">
     <RequestScheduler>
             <RAIDb-2Scheduler level="pessimisticTransaction"/>
             </RequestScheduler>
     <RequestCache>
             <ResultCache granularity="database">
                     <DefaultResultCacheRule>
                              <EagerCaching/>
                              </DefaultResultCacheRule>
                     </ResultCache>
             </RequestCache>
     <LoadBalancer>
             <RAIDb-2>
                     <CreateTable policy="roundRobin" numberOfNodes="1">
                              <BackendName name="local"/>
                              <BackendName name="cern"/>
                              </CreateTable>
                      <RAIDb-2-RoundRobin/>
                      </RAIDb-2>
               </LoadBalancer>
   </RequestManager>
```

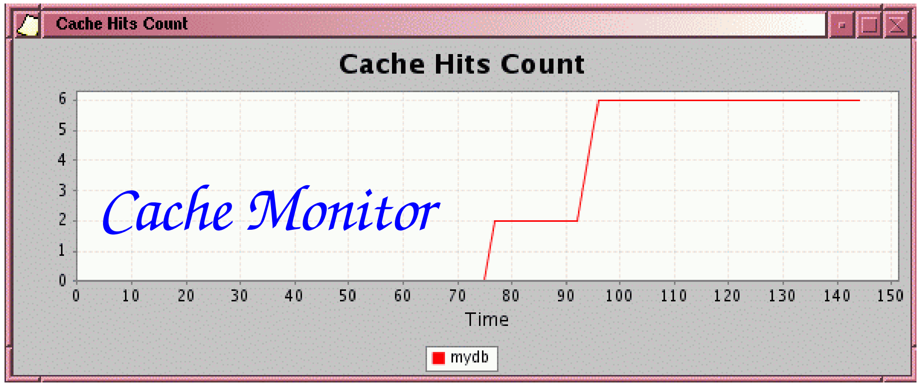
Sequoia Tools

- *Many other components exist*
- *Everything (and more) possible from command-line*

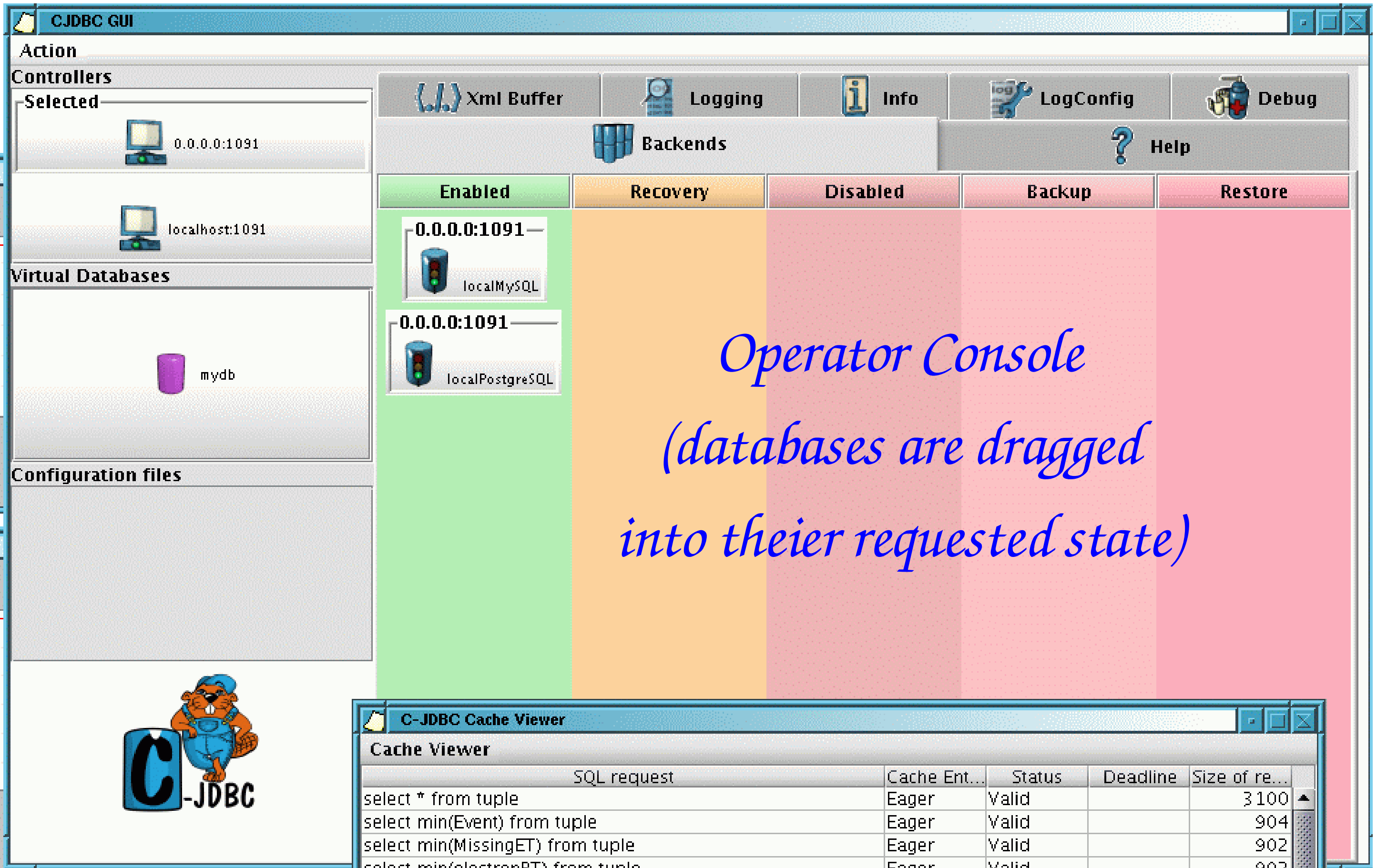




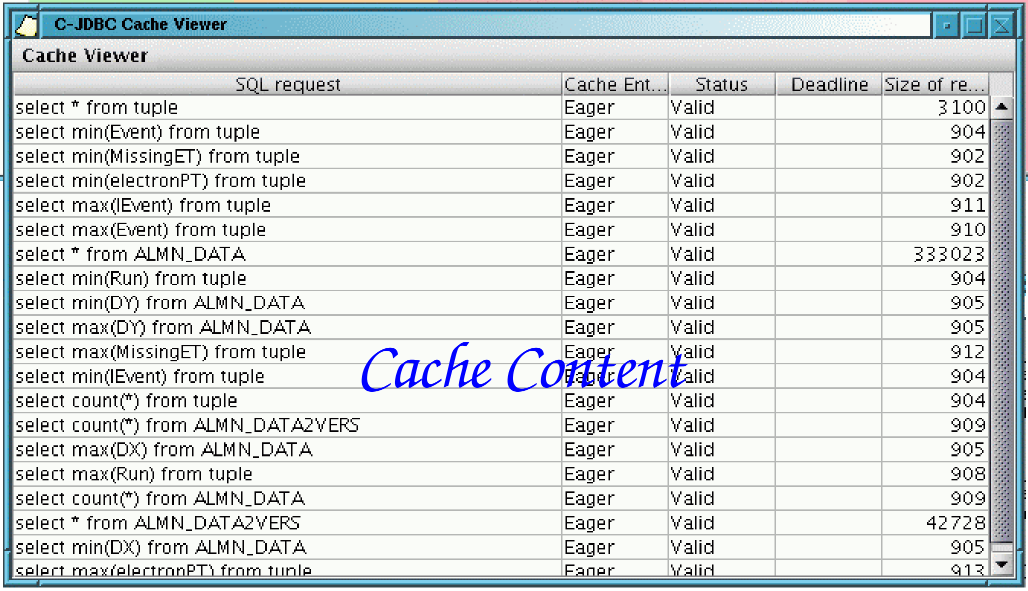
Backend Monitor



Cache Monitor



Operator Console (databases are dragged into theier requested state)



Cache Viewer

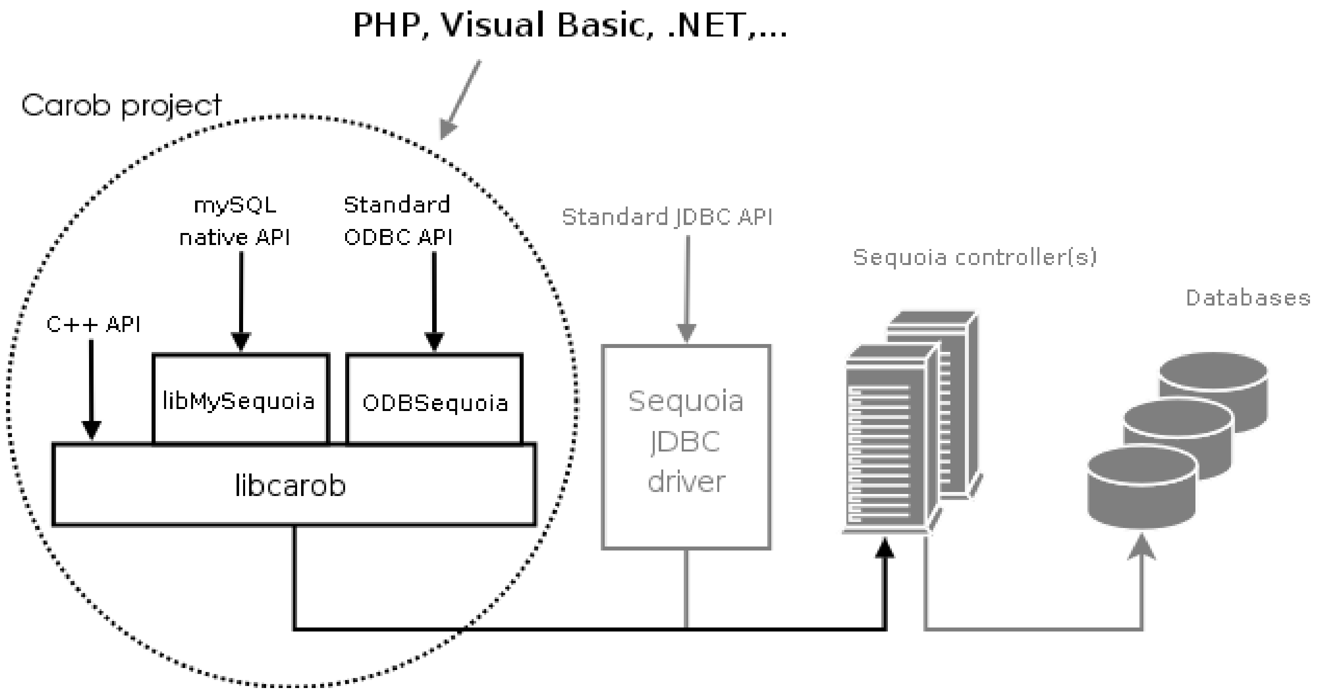
SQL request	Cache Ent...	Status	Deadline	Size of re...
select * from tuple	Eager	Valid		3100
select min(Event) from tuple	Eager	Valid		904
select min(MissingET) from tuple	Eager	Valid		902
select min(electronPT) from tuple	Eager	Valid		902
select max(lEvent) from tuple	Eager	Valid		911
select max(Event) from tuple	Eager	Valid		910
select * from ALMN_DATA	Eager	Valid		333023
select min(Run) from tuple	Eager	Valid		904
select min(DY) from ALMN_DATA	Eager	Valid		905
select max(DY) from ALMN_DATA	Eager	Valid		905
select max(MissingET) from tuple	Eager	Valid		912
select min(lEvent) from tuple	Eager	Valid		904
select count(*) from tuple	Eager	Valid		904
select count(*) from ALMN_DATA2VERS	Eager	Valid		909
select max(DX) from ALMN_DATA	Eager	Valid		905
select max(Run) from tuple	Eager	Valid		908
select count(*) from ALMN_DATA	Eager	Valid		909
select * from ALMN_DATA2VERS	Eager	Valid		42728
select min(DX) from ALMN_DATA	Eager	Valid		905
select max(electronPT) from tuple	Eager	Valid		913

Cache Content

Carob

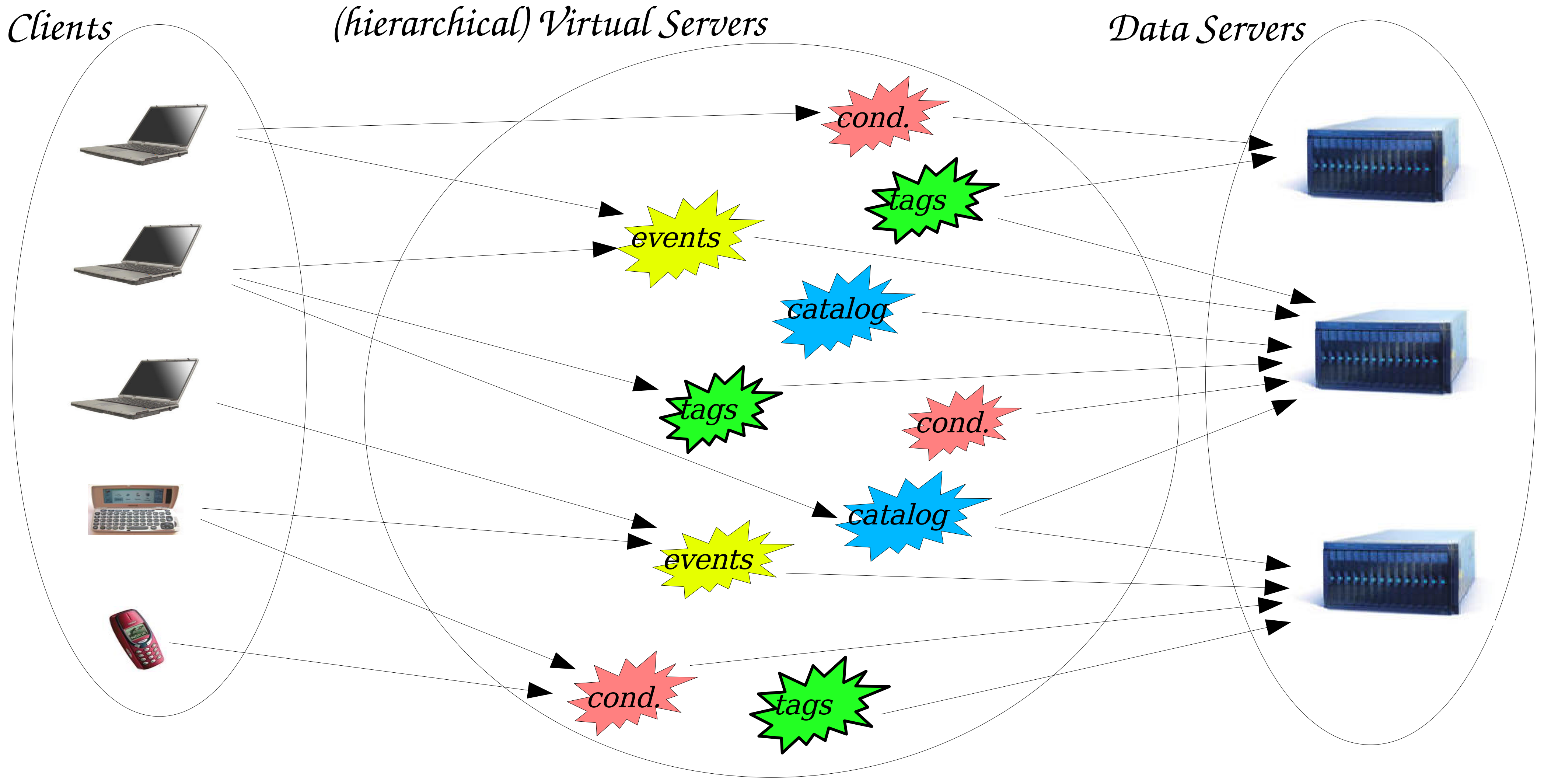
Sequoia for Legacy C/C++

- *libCarob for C++: JDBC API directly accessible from C++*
- *libMySequoia for C: implementing MySQL C API, it can be used directly by any applications interfaced to MySQL C API*
- *ODBSequoia: it can be used directly by any application interfaced to ODBC*



Distributed Interactive Environment

Architecture Project



Architecture Advantages

with respect to LCG/AA solutions

- Light local client: all distribution logic (pooling, load balancing, failover, caching, ...) is managed by Virtual Servers, Clients just have to know Virtual Servers URLs
  - unlike LCG/Pool “Connection Library”
- Schema independence, Standard communication protocols: Virtual Servers don't depend on Clients, they operate on SQL; any SQL can be processed
  - unlike FronTier
- Modular architecture: easily extensible via Plugins
- Support for all SQL databases
- Multilanguage: Java natively, C/C++ via Carob

Documentation

C-JDBC

- Sequoia is provided by ObjectWeb Open Source Middleware Consortium, released under GPL, they has active user base and responsive developers. They are probably the only (so the best) such OpenSource Tool.
- Sequoia work swell with other ObjectWeb Tools, like Octopus, JOnAS Application Server, Speedo JDO, etc.
- Documentation:
  - Sequoia Homes: http://c-jdbc.objectweb.org, http://sequoia.continuent.org