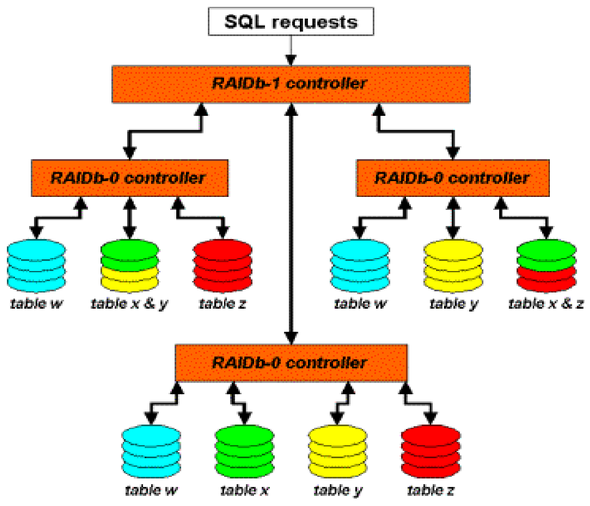
Julius Hrivnac

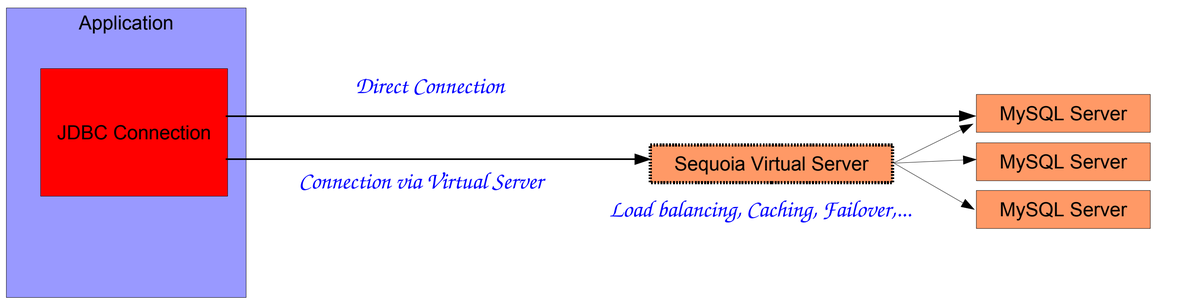
International Conference on Computing in High Energy and Nuclear Physics
2-7 Sept 2007 Victoria BC Canada

Distributed Interactive Access to Large Amount of Relational Data using Sequoia

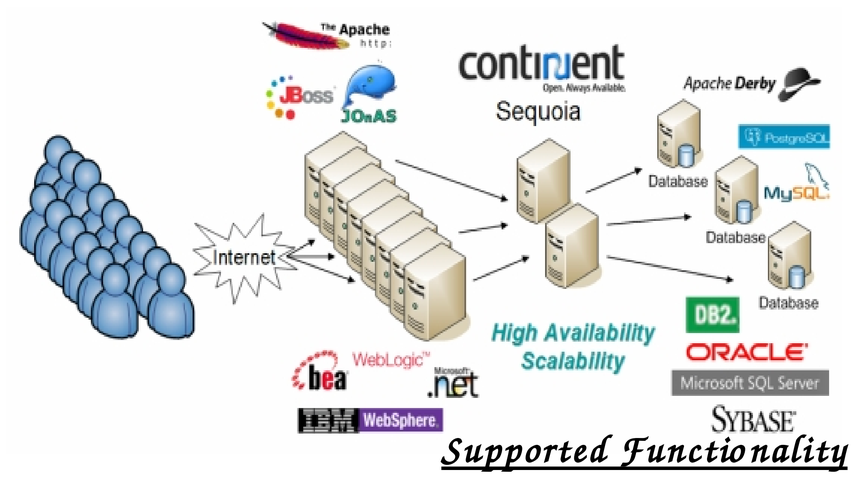


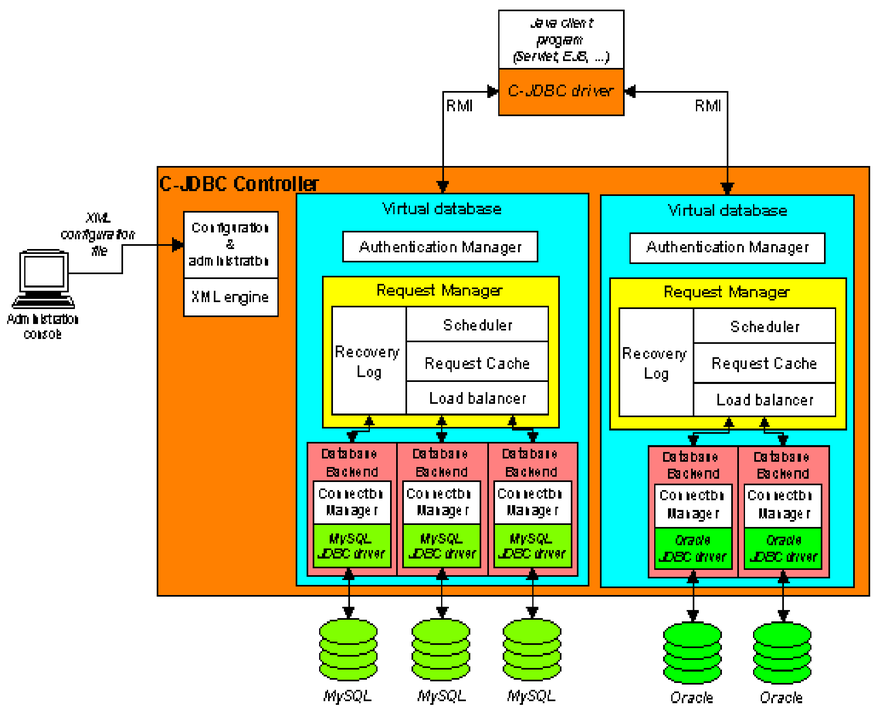
- SQL tables can be spread on several database Servers, some tables may be replicated. User sees a single front-end.
- Sequoia acts as a (Proxy) Virtual SQL Server forwarding all requests to appropriate databases (real or another virtual). Replicated and/or complementary tables are supported (even on 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.
- 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 other "connection libraries"
- Schema independence, Standard communication protocols: Virtual Servers don't depend on Clients, they operate on SQL; any SQL can be processed
  - unlike other "proxy caches"
- Modular architecture: easily extensible via Plugins
- Support for all SQL databases
- Multilanguage: Java natively, C/C++ via Carob

```
// 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:sequoia://sequoiaserver.cern.ch/Tuples", "user", "passwd");
```





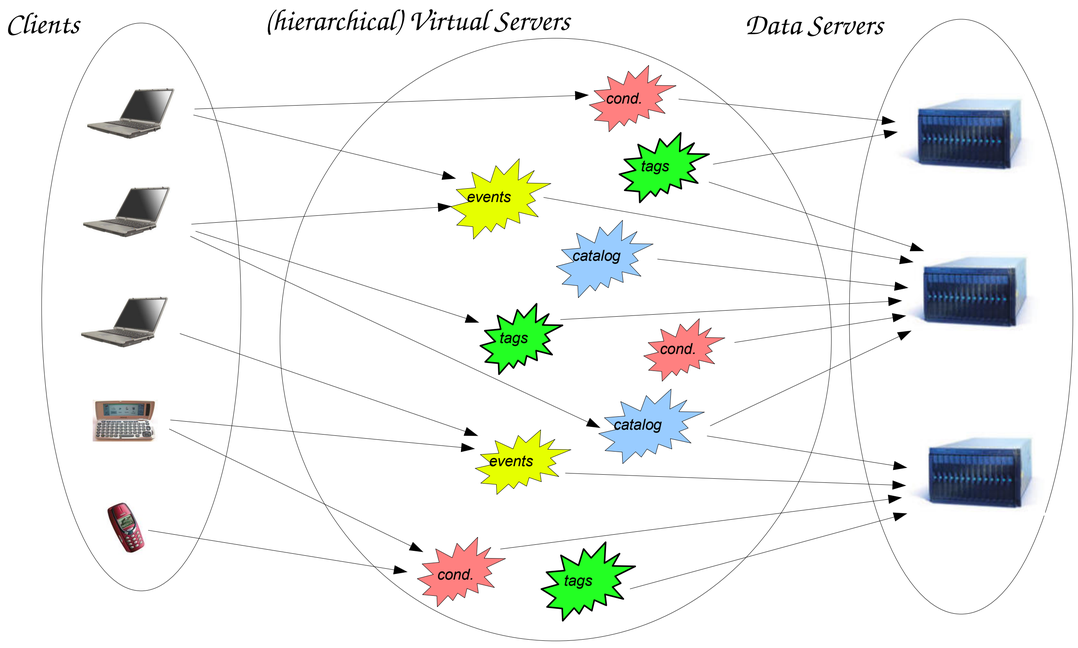
Supported Functionality

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

Plugins under Development

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

Distributed Database Architecture



Access from Legacy C/C++ code

- 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