

# CORAL Server

*A middle tier for accessing relational database servers  
from CORAL applications*

**Andrea Valassi, Alexander Kalkhof** (DM Group – CERN IT)

**Martin Wache** (University of Mainz – ATLAS)

**Andy Salnikov, Rainer Bartoldus** (SLAC – ATLAS)

*Distributed Database Operations Workshop, 26th November 2009*

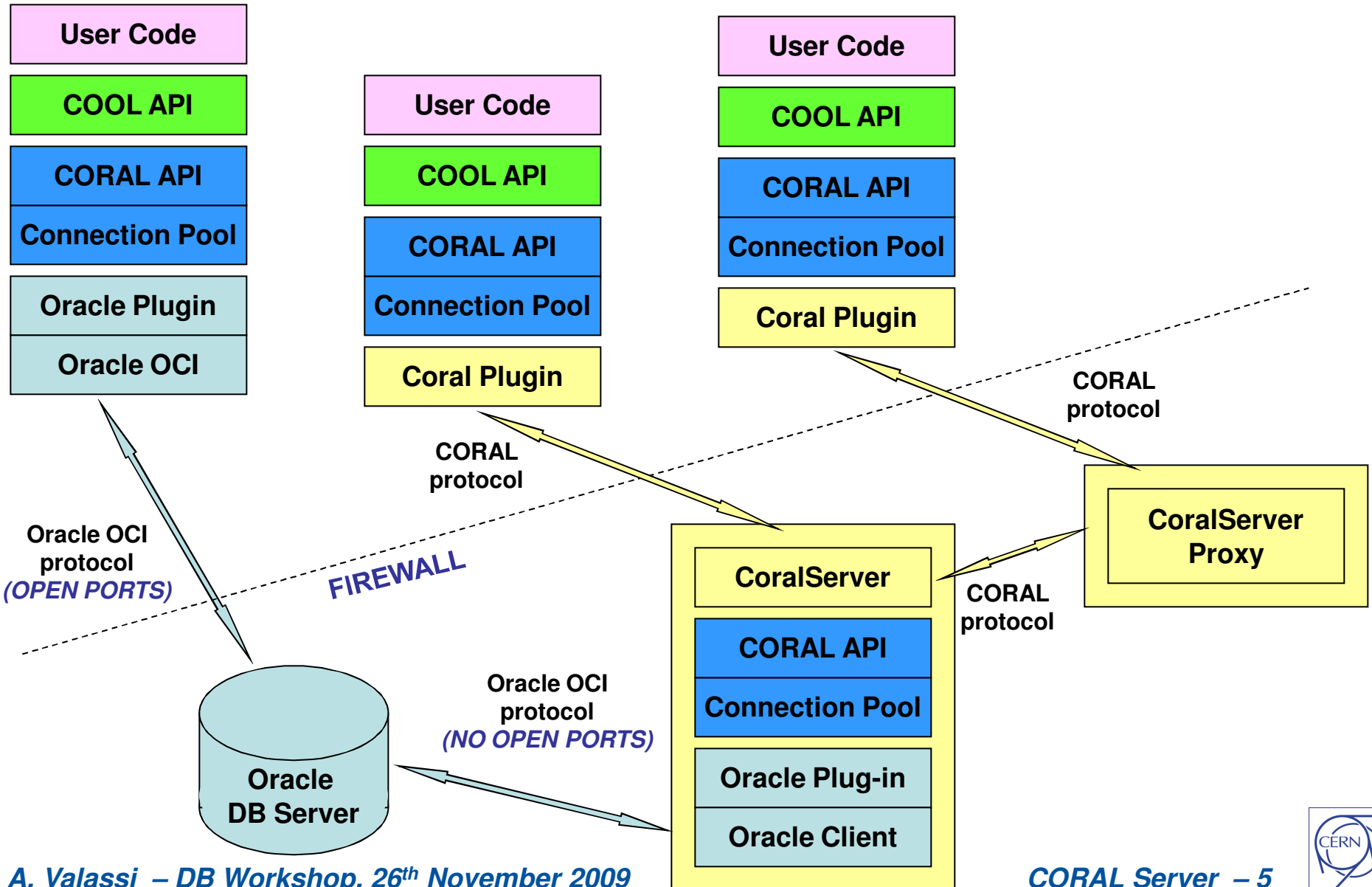
*Thanks for useful suggestions and contributions  
to Dirk Duellmann, Zsolt Molnar, the Physics Database team  
and all other members of the CORAL, POOL and COOL teams!*

- **Motivation**
- **Development and deployment status**
  - ATLAS deployment at Point1
  - Software architecture design
- **Outlook**
  - Planned functional and performance enhancements
  - Secure access deployment scenario
- **Conclusions**

- **CORAL used to access main physics databases**
  - Both directly and via COOL/POOL
    - Important examples: conditions data of Atlas, LHCb, CMS
  - Oracle is the main deployment technology at T0 and T1
    - Main focus of 3D distributed database operation
- **No middle tier in current client connection model**
  - Simple client/server architecture
    - No Oracle application server
  - One important exception: Frontier (for read-only access)
    - Used by CMS for years, adopted by ATLAS a few months ago
- **Limitations of present deployment model**
  - Security, performance, software distribution (see next slide)
  - *Several issues may be addressed by adding a **CORAL middle tier***
    - *Proposed in PF reports at the [2006 AA review](#) and the [2007 LHCC](#)*

- **Secure access (R/O and R/W) to database servers**
  - Authentication via Grid certificates
    - No support of database vendor for X.509 proxy certificates
    - Hide database ports within the firewall (reduce vulnerabilities)
  - Authorization via VOMS groups in Grid certificates
- **Efficient and scalable use of database server resources**
  - Multiplex clients using fewer physical connections to DB
  - Option to use additional caching tier (CORAL server proxy)
    - Also useful for further multiplexing of logical client sessions
- **Client software deployment**
  - CoralAccess client plugin using custom network protocol
    - No need for Oracle/MySQL/SQLite client installation
- **Interest from several stakeholders**
  - Service managers in IT: physics DB team, security issues...
  - LHC users: Atlas HLT (replace MySQL-based DbProxy)...

# Deployment scenario (e.g. COOL)



- **First phase of development in 2007-2008**
  - User requirements (Atlas HLT) analyzed in November 2007
  - Main focus of this development phase was on server components
    - Design based on template meta-programming techniques
  - Passed simple COOL R/O tests, failed HLT tests against COOL DB
- **New developments in 2009** ← *Only focus of this talk*
  - Several changes in the team at the end of 2008
    - Software review in December 2008 to identify areas for improvement
    - Development restarted (from ~scratch) in January 2009 using a new comprehensive architecture design covering both server and client
  - Kept focus on R/O access with caching proxy for Atlas HLT
    - This is only the first customer – but is an excellent benchmark
  - Aggressive schedule led to successful deployment in only 9 months
    - March: pass standalone multi-client HLT tests on COOL DB
    - June: pass standalone tests on trigger and geometry DBs – first release
    - September: software deployed at Point1 and fully validated/operational
    - Details in Andy's talk at a recent [LCG AA meeting](#)

Andy Salnikov – LCG Applications Area Meeting 2009-11-04

## Configuration Problem

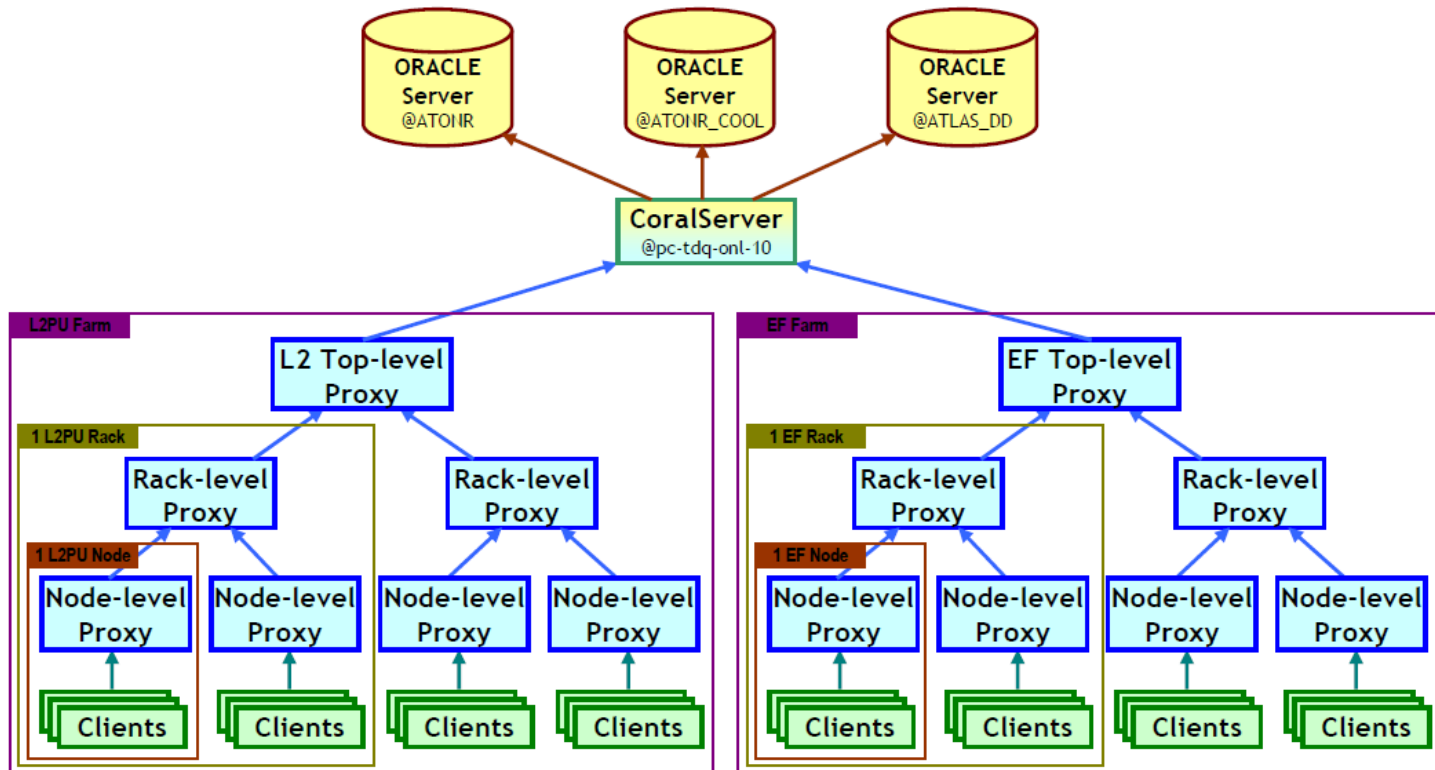
- HLT configuration moves a lot of data
  - ◆ around 2000 nodes, up to 8 processes per node, tens to hundred MB of configuration data
$$2000 \times 8 \times 10\text{MB} = 160\text{GB}$$

(~30 minutes over 1Gbps connection)
  - ◆ all clients request configuration data from database at the same instant
  - ◆ single server cannot handle such load
- Positive points:
  - ◆ all clients get identical data from database (one set for LVL2 and wider set for EF)
  - ◆ database server needs to ship only single copy of the data O(10MB) (fraction of a second over 1Gbps)

Andy Salnikov – LCG Applications Area Meeting 2009-11-04

## Proxy Tree with CoralServer

- Top-level proxies connect to the CoralServer
- CoralServer gets data from three Oracle Servers





Andy Salnikov – LCG Applications Area Meeting 2009-11-04

## CORAL Server Deployment

- Tests with CoralServer were performed on SLAC TDAQ farm during whole development progress
  - ◆ stability, crashes, memory leaks, performance
  - ◆ tests were done in real HLT environment
- Learned also how to run ATLAS HLT client application standalone
- Pre-deployment tests were done at pre-series farm
- Thanks to all these tests the deployment was very smooth
  - ◆ One issue with transaction mode quickly resolved
  - ◆ Maybe the least problematic deployment at Point1 in ATLAS history

Andy Salnikov – LCG Applications Area Meeting 2009-11-04

## Current Status

- Basic set of features is enough to run ATLAS HLT
  - ◆ Performance is adequate at current scale, on par with M2OProxy
  - ◆ Stability is OK, no problems so far
  - ◆ Simplified authentication handling
  - ◆ We are still learning
- We need tools to monitor and understand how our system behaves (both CoralServer and ATLAS HLT)
  - ◆ Need monitoring built-in into CoralServer and Proxy (already under development)
  - ◆ Need tools to watch/analyze monitoring data, integrated into TDAQ
- There is interest from other fronts
  - ◆ Using proxy for writing into database

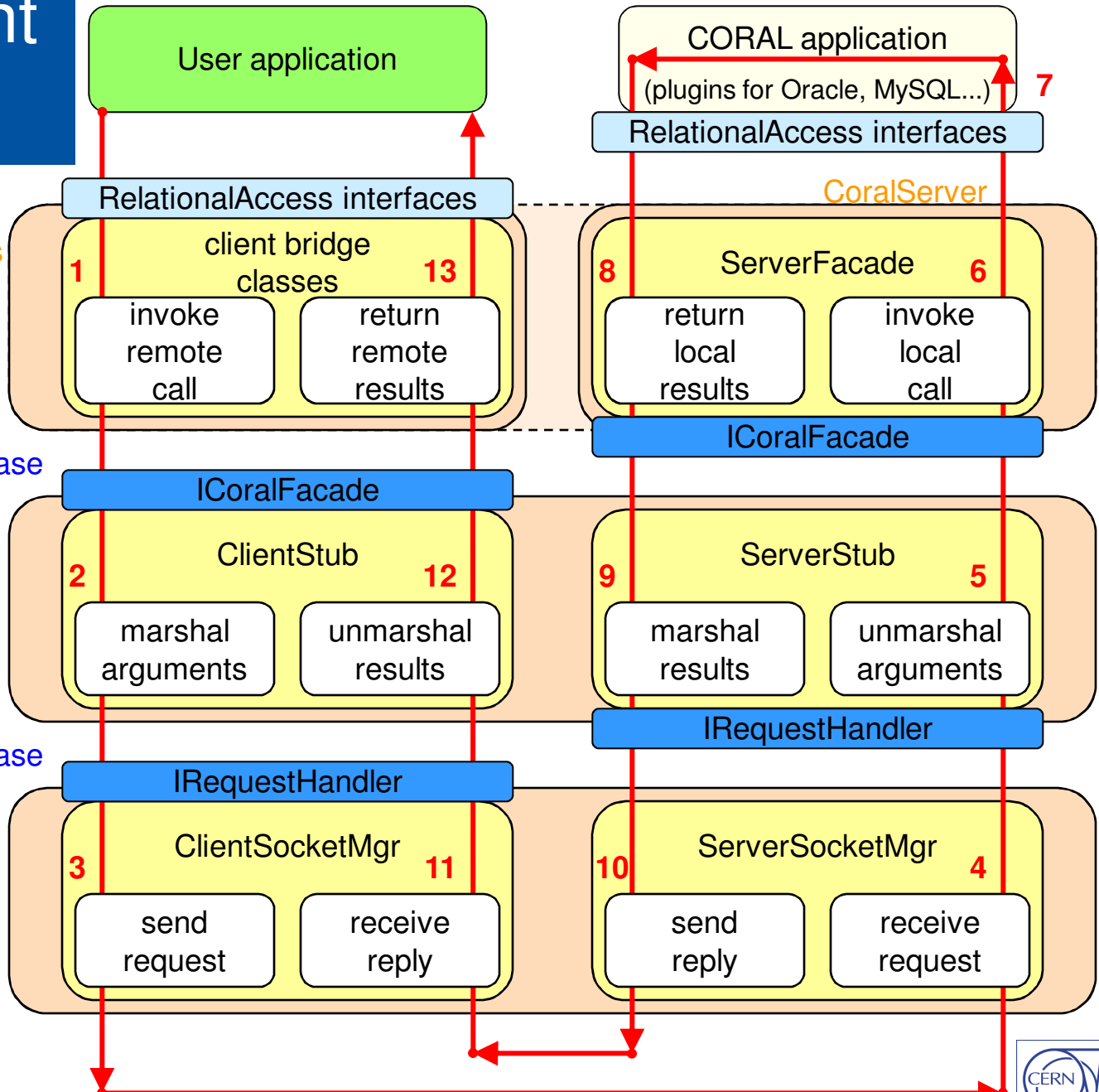
- **Joint design of server and client components**
  - Split system into packages ‘horizontally’ (each package includes both the server-side and the client-side components)
    - Proxy is now standalone but can be reengineered with this design
  - RPC architecture based on December 2007 python/C++ prototype
- **Different people may work in parallel on different packages**
  - Minimize software dependencies and couplings
  - Upgrades in one package should not impact the others
- **Strong emphasis on functional (and performance) tests**
  - Include standalone package tests in the design
    - Aim to intercept issues before they show up in system tests
  - System tests may be decomposed back into incremental subtests
- ***Decouple components using abstract interfaces***
  - Modular architecture based on object-oriented design
  - Thin base package with common abstract interfaces
  - Implementation details encapsulated in concrete derived classes

# SW component architecture

Package 1a / 1b  
[Andrea]

Package 2  
[Alex]

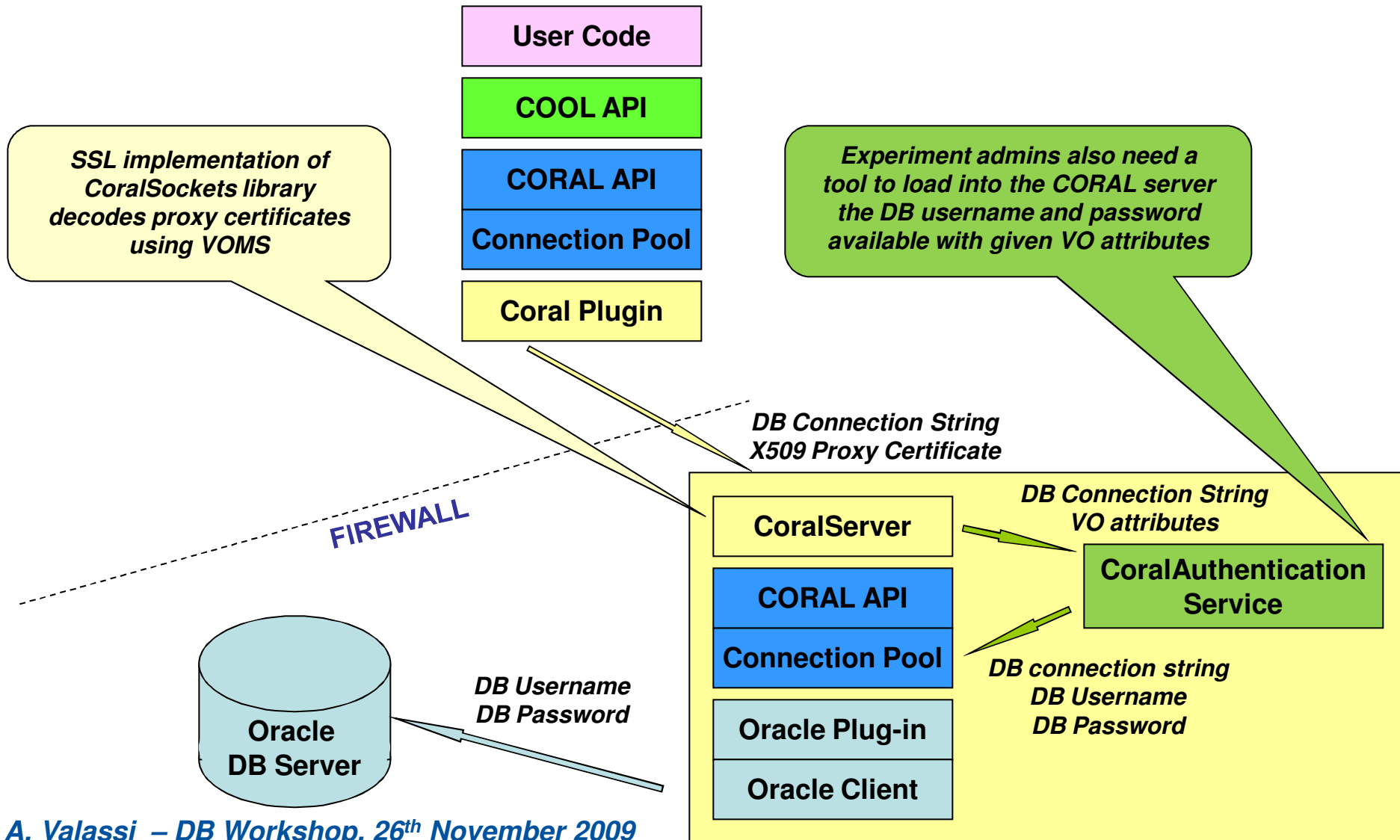
Package 3  
[Martin]



- **Software deployed and validated for Atlas HLT – Q3 2009**
  - Read-Only access with passwords (single authentication.xml file)
- **Complete secure authentication/authorization – Q4 2009?**
  - Proxy certificates supported in client/server – but not yet released
    - Dependencies on SSL and VOMS versions must be sorted out
    - SSL missing in proxy – might switch to common components
    - Separate SSL and non-SSL sockets for control and data packets
  - In progress: tool to load Oracle passwords into the CORAL server
- **Monitoring enhancements for ATLAS HLT – Q1 2010?**
  - Need at least equivalent features as M2O and DbProxy monitoring
- **Further performance tests and optimizations – Q1 2010?**
  - Measure and minimize the overhead w.r.t. Oracle direct access
  - Later: add disk cache to proxy for possible offline usage
- **Full Read-Write functionalities – Q1 2010?**
  - DML (e.g. insert table rows) and DDL (e.g. create tables)
- **Deployment of test server and proxy at CERN – Q2 2010?**
  - In parallel, discuss possible deployment at T1 sites

# Secure access scenario

Thanks to A. Frohner, J.P. Baud, V. Ciaschini, M. Alandes, S. Roiser for help with SSL and VOMS!



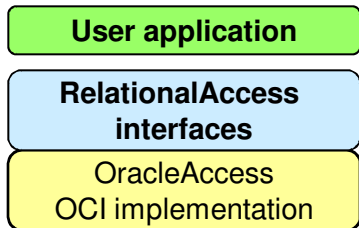
- **CoralServer is validated and fully operational for Atlas HLT**
  - Only 9 months after start of developments with present design
  - Read-Only access with passwords, plus caching proxy
  - Also adopted for Global Monitoring at Atlas Point1
    - Interest for Read-Write access at Point1 too
- **Several developments are planned in the future**
  - Secure authentication, Read-Write access, Monitoring...
  - Precise timescales for achieving them are hard to predict
    - Manpower is limited and needed also for mainstream PF support tasks
- **Deployment at CERN for offline users will follow**
  - Deployment at Tier1 sites could be useful too
- **Mutually beneficial collaboration of CERN/IT and ATLAS**
  - Effective and very pleasant too: thanks to everyone in the team!
  - I believe this software will be useful for other experiments too

# Reserve slides

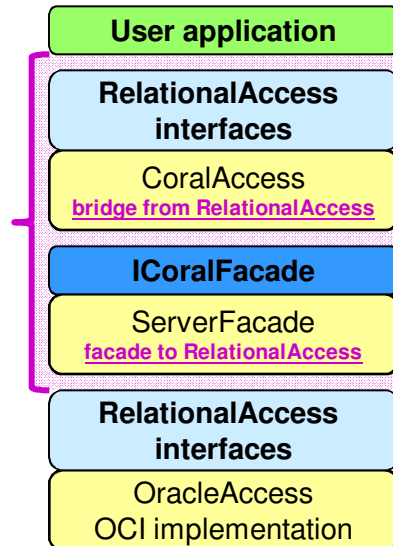


# Incremental tests of applications

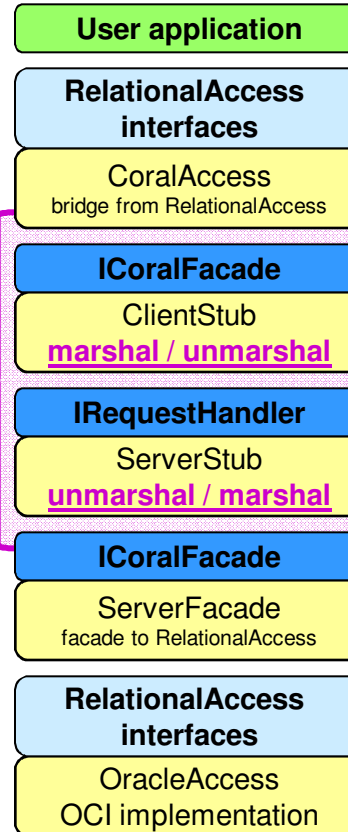
## Traditional CORAL "direct" (OracleAccess)



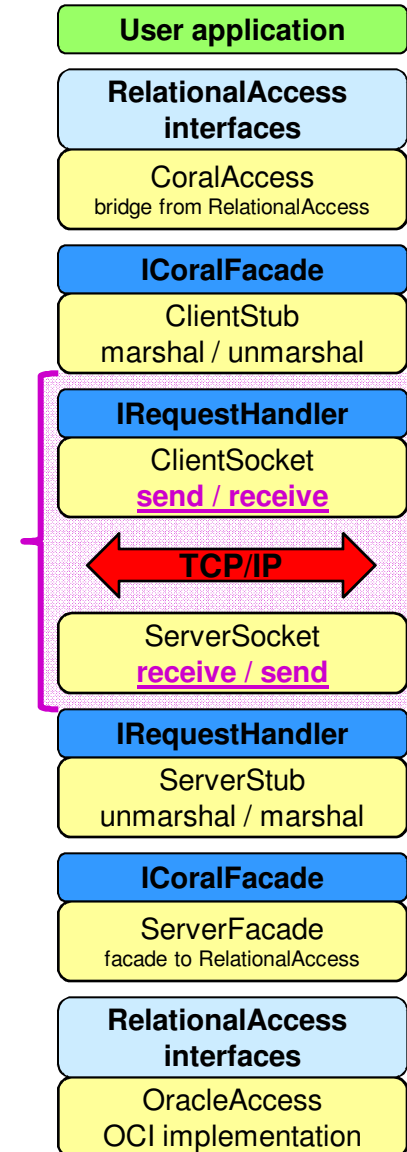
## Add package 1 (a/b) "façade only"



## Add package 2 "stub + façade"



## Add package 3 "server" (full chain)



- **Server is multi-threaded**
  - Threads are managed by the SocketServer component
  - One listener thread (to accept new client connections)
  - One socket thread per client connection
    - Pool of handler threads (many per client connection if needed)
- **Network protocol agreed with proxy developers**
  - Weekly meetings (ongoing, now used for regular progress review)
  - Most application-level content is opaque to the proxy
    - Proxy understands transport-level metadata and a few special application-level messages (connect, start transaction...)
    - Most requests are flagged as cacheable: only hit the DB once
  - Server may identify (via a packet flag) client connections from a proxy and establishes a special 'stateless' mode
    - Session multiplexing (cache connect and drop disconnect requests)
    - One R/O transaction spans whole session (drop transaction requests)
    - 'Push all rows' model for queries (no open cursors in the database)

- **Development has largely focused on ATLAS HLT so far**
  - Very complex, but also very simple, depending on the point of view
    - Some specific choices for HLT (mainly in the proxy layer) may need to be changed for the deployment of a general-purpose offline service
- **Oracle connection sharing was initially disabled**
  - Not necessary for HLT, thanks to session multiplexing in the proxy
    - But necessary for connection multiplexing in general-purpose server
  - Hang had been observed with connection sharing
    - Problem identified as Oracle bug in MT mode and solved in 11g client
- **HLT deployment needs non-serializable R/O transactions**
  - Requirement: see data added after the start of the R/O connection
    - Connections are potentially long
  - Presently handled by a 'hidden' env variable in OracleAccess
    - May need a cleaner way out (API extension: three transaction modes)

