CORAL Server - A Development Proposal

Dirk Düllmann, CERN IT

LCG Application Area Meeting,

23 January 2008

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Outline

- CORAL as foundation for Physics Database Applications
  - Package Layering
  - Recent developments
- Review of remaining deployment issues
  - Secure database access
  - Efficient use of server resources for many concurrent clients
  - Software distribution
- CORAL server design study
  - Integration with existing code base
  - Server connection and threading model
  - Network protocol structure
  - Results from early prototype studies

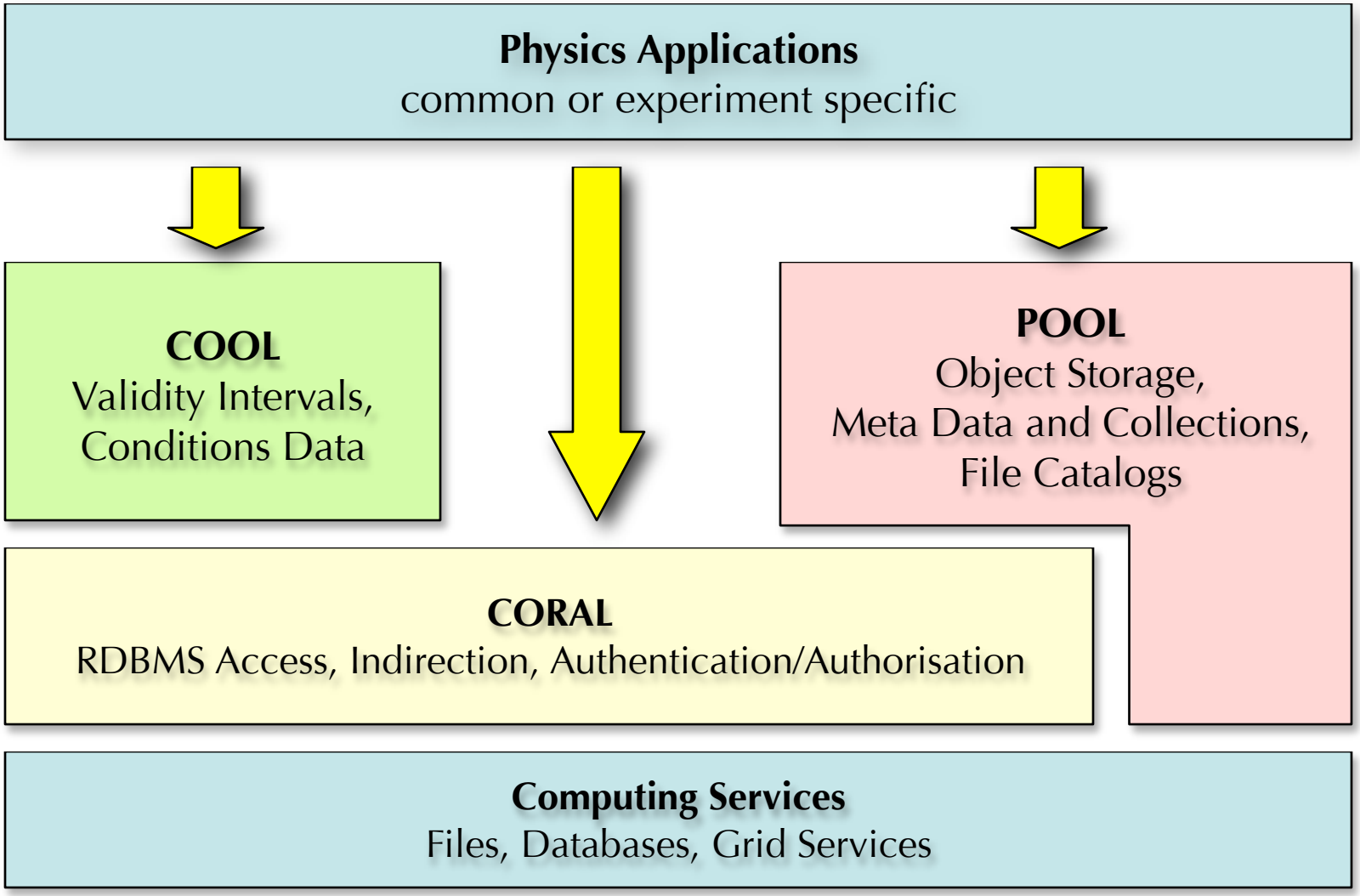
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Disclaimer

- This is a development proposal and does therefore contains in several areas:
  - vapourware, hand waving
  - implementation choices, which may still change
- Still, the aim of this presentation is to convince you that the development
  - is necessary to insure scalability and availability of DB services
  - design aspects are reasonably well understood
  - could be introduced on a time scale of end of this year without major disruption to existing CORAL apps/ services
  - can be done with rather limited resources in the Persistency Framework

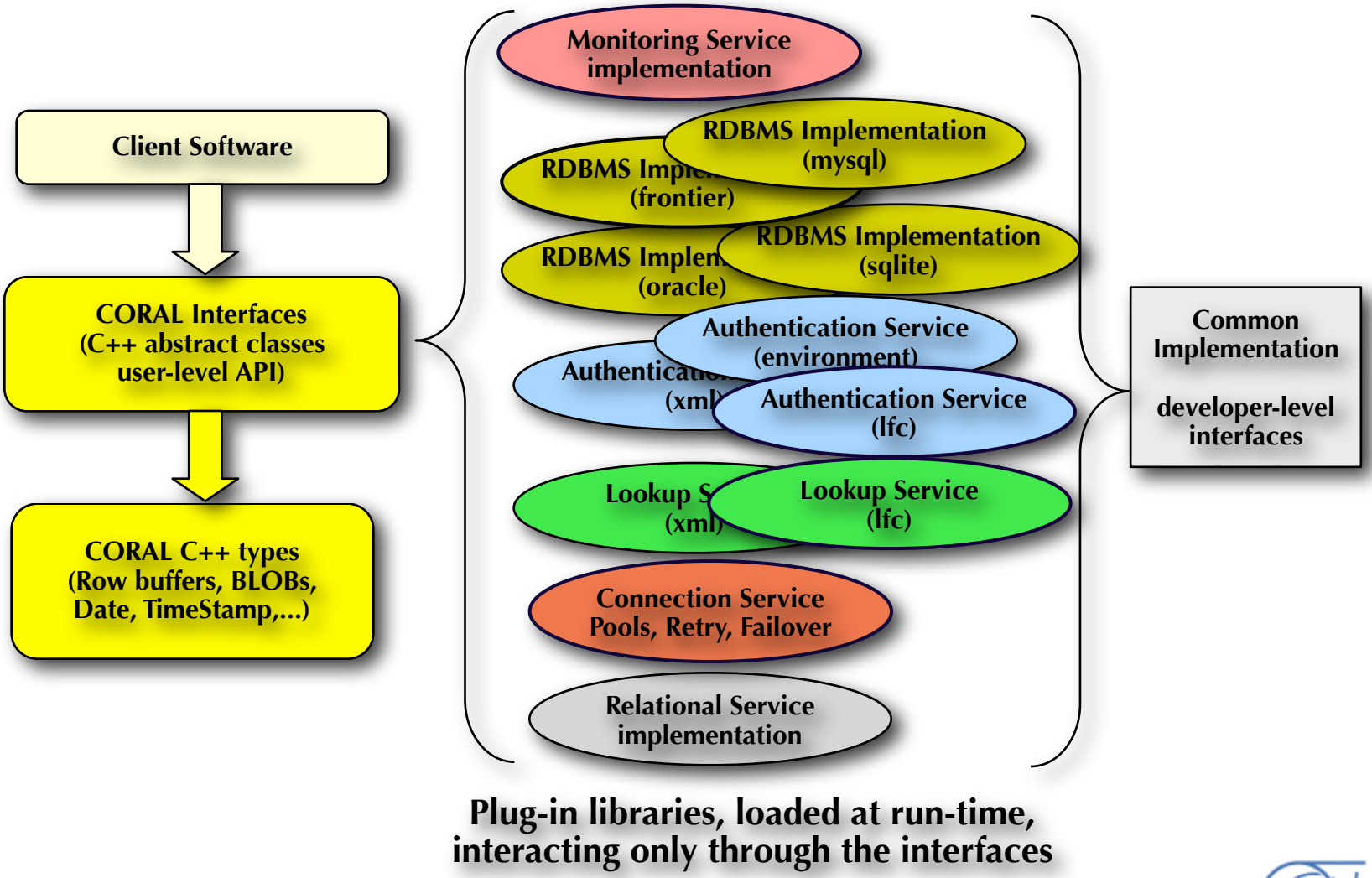
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

CORAL DB Foundation



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

CORAL Components



Plug-in libraries, loaded at run-time, interacting only through the interfaces

Data Management

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Remaining Issues - Access Security

- Oracle server does not support grid authentication (X.509 proxy certs) and grid authorisation (eg VOMS groups)
  - CORAL implemented a secure grid enabled store for database credentials (db user/pw pairs) based on LFC as a workaround
  - being picked up slowly
  - additional dependency on LFC service
- Many physics application still exposed to possible security related outages, which severely impact on production
  - disclosed user/password pairs
  - vulnerabilities/exploits of Oracle servers exposed to the internet
- Need progressively to
  - move to grid certificates for external connections
    - apart from applications for which the password distribution (and change - after eg an password exposure) is well controlled
  - minimise external exposure of Oracle server ports
    - eg by moving CORAL based applications to a secure protocol

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Remaining Issues - Efficient Server Use

- Physics apps tend to use databases with
  - very many concurrent connections/sessions
  - rather limited data traffic on each connection per time unit
- Result
  - Oracle server spawns/manages very many idle processes
  - waste of server memory and CPU cycles, which could be used for caching database blocks and executing queries

- Need to bundle connections in front of the database server
  - the existing Connection Service component does that, but is limited to pool only connection in the hosting process
  - need to move this component closer to the database server

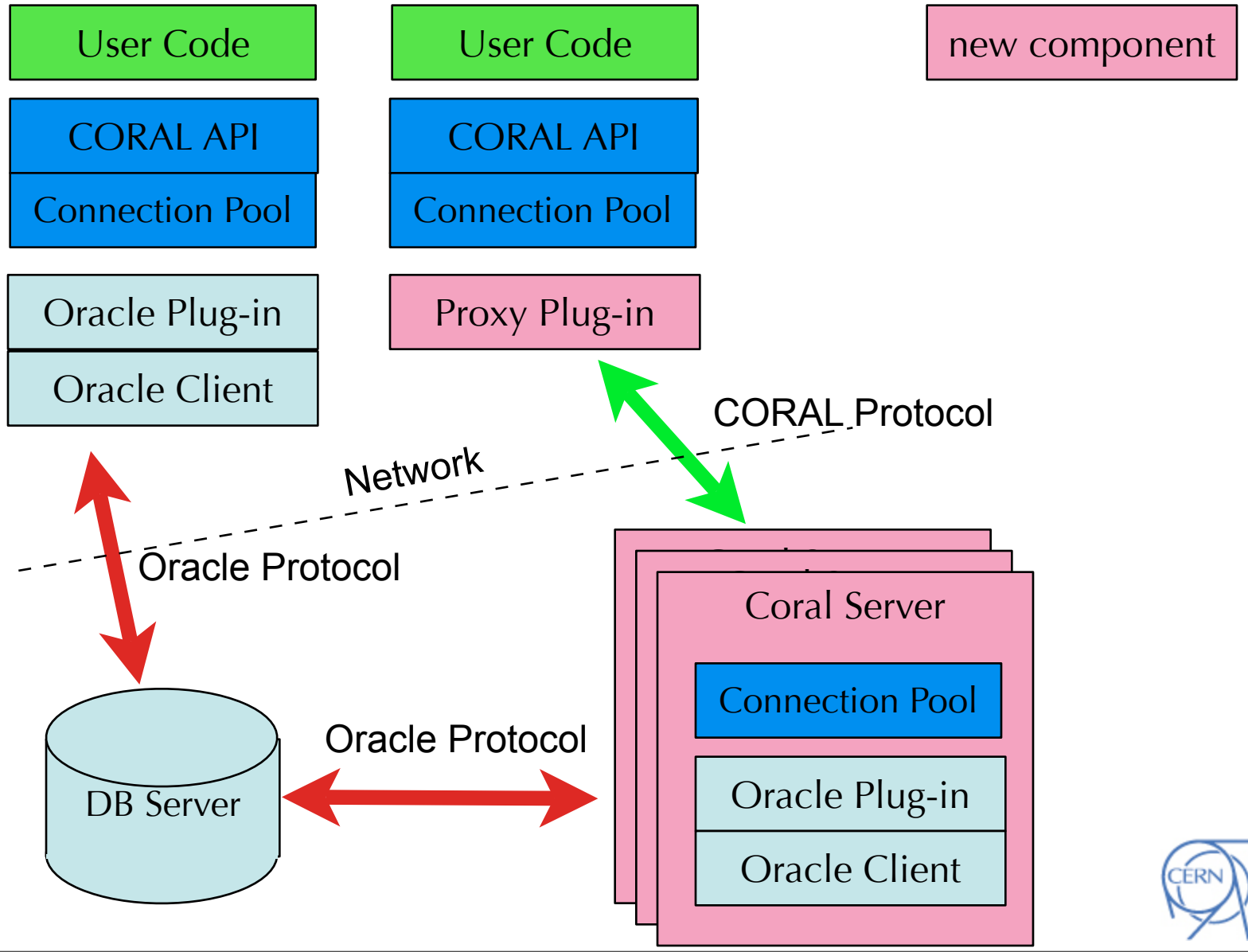
CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Remaining Issues - Client s/w Distribution

- CORAL clients link (at runtime) against the specific database access libraries
  - eg Oracle instant client, MySQL client
- Even though an agreement exists with Oracle to distribute the client software within LCG
  - configuration management issues because of compiler dependencies (eg for OCCI)
  - risk of failing to properly implement export constraints requested by the Oracle license
- Both could be avoided/reduced by making the CORAL client fully independent of any vendor provided client s/w

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

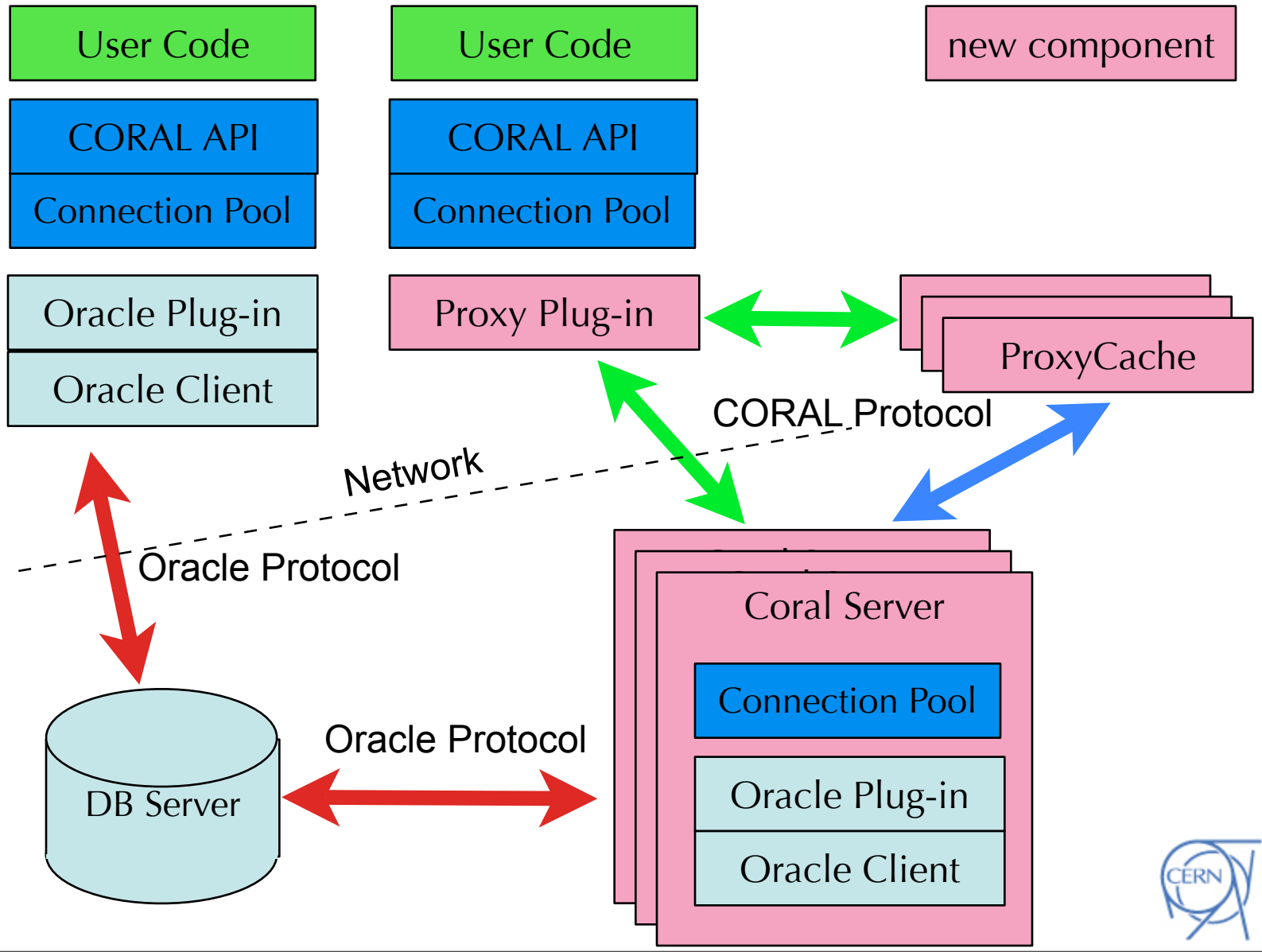
Component Reuse for the Server



Data Management

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Component Reuse for the Server



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Main risk areas

- Server component
  - needs stable multi-threading code
  - needs stable memory footprint
  - are existing components suitable for server side deployment?
- Client semantics and API stability
  - need to be fully transparent to existing clients
  - needs to coexist with existing services
- Protocol and it's maintenance
  - scalable protocol, which does not impair on performance
  - including a secure control channel for external access
  - manage protocol evolution
- Do resources and deployment time scales match?

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Stateless vs Stateful

- Database connections are intrinsically stateful
  - Transaction context spans many operations
  - Bulk queries/inserts are spread over several client server communications
  - Authentication and authorisation state is expensive to obtain and dominates CPU use of many existing services
- Stateless connection model would lead to
  - many repeated operations to recover existing state
  - problems with lifetime management of transaction/query/ security related data in the server if client programs fail
- CORAL server itself is stateless (beyond transactions)
  - several servers can run in parallel (eg DNS load balanced)
  - Individual servers can fail losing only the state of uncommitted transactions

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Server Prototype

- Implementation options
  - **multi-threaded** vs multi-processed
  - network layer s/w packages (ACE, **BOOST asio**)
    - for now direct TCP implementation (linux, osx)
  - **C++ server** or Tomcat/Java (eg extended FronNTier server)
- Threading Model (C++ prototype)
  - server runs from a thread pool
    - a listener thread on a well known port
      - establish client connections
    - a dedicated thread per client connection
      - multiple sessions per connections (eg for multiplexing cache proxy)
    - a worker thread for each database operation
  - shared information
    - connection table with authentication and session state
    - session table with cursor / bulk inserter state

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Secure External Connections

CERN IT Department

- Server protocol is split into
  - control path: connection requests, coral client requests
  - data path: server response eg query result sets
- For external connections
  - use GSI or SSL for control path
  - client certificate required to connect from outside the site network
  - data path can stay open (at least for HEP)
  - credentials will be evaluated once per session
  - VOMS groups from certificate define database user and privileges
  - Real database user and password used only by CORAL server
- Online environments may continue with open control path
  - coral://some-online-database (LAN connection)
    - plus existing coral authentication file
  - corals://some-offline-database (WAN connection)
    - no further information apart from certificate required

Data Management

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Prototype Studies - Functional Integration

- Python based server prototype
  - exercising client server communication and server side protocol implementation using existing CORAL code
  - identified minimal set of operations required to implement a functional client (some 10 different basic operations)
- C++ client prototype as a CORAL plugin
  - new plugin - (CoralAccess in CVS)
- Versioned protocol implementation
  - new package (CoralProtocol in CVS)
    - provides plugin to server (and client)
  - match-making based on client and server supported protocol versions during connection start
    - may allow protocol changes without server downtime

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Prototype Studies - Server Scalability

- As client side CORAL component will move to the server side
  - need to insure that existing implementation scales with significantly more threads and activity
- Possible Issues
  - scalability of CORAL thread locking
  - scalability of DB client s/w with many threads
- To test this, we used a CORAL application
  - with a large number of threads
  - several concurrent individual queries
- Result
  - No bottleneck observed with thread diagnostic tools from Intel wrt to the locking used in CORAL or the ORACLE client
  - Threads execute in parallel and use the available CPU and DB server performance

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Cacheable Network Protocol

- Several experiments have introduced caching layers to increase scalability by exploiting many identical queries (r/o)
  - CMS: FroNTier plug-in to CORAL
  - ATLAS online: caching MySQL proxy
- Propose to move caching layer in front of the CORAL server
  - expose caching relevant data in network protocol
- Goal:
  - allow to create a generic CORAL proxy cache (main user initially ATLAS online)
  - allow to multiplex connections in a hierarchical set of cache servers
  - CoralProxyServer under development by ATLAS contributors

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Caching / Multiplexing Proxy

CERN IT Department

- Core developers from ATLAS online team
  - authors of the MySQL caching proxy
  - used through CORAL in the ATLAS HLT tests
  - Motivation: move ATLAS' coral application base to a Oracle based service
- Weekly design meetings
  - clarified CORAL protocol aspects to enable generic caching/ multiplexing
  - de-coupled cache development from internal details of CORAL protocol (which may still change)
  - Propose to make ATLAS development available via CORAL CVS / releases
- Aggressive schedule as online environment will move to Oracle back-end soon
  - can live initially with reduced functionality set
    -> read-only, no cursors, no bulk ops, no ssl, single threaded, single protocol version etc...

Data Management

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

What does a cache server need to know about CORAL?

- Caching server is introduced between CORAL client and server
  - eg for online apps, HLT
- C++ server prototype
  - caching is done with the same stateful binary network protocol as in un-cached mode
  - in case of composite requests all individual exchanges are cached
- Application and CORAL provide information about
  - payload to cache (with optional expiration time)
  - key and request ID to identify the request/response chain
  - routing information to enable multiplexing (client ID)
  - protocol version and endianness
- Protocol is purely data based
  - no shared RPC infrastructure required

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Platforms and Dependencies

- Target database back-ends
  - at least Oracle and MySQL
- Target platforms
  - client : all LCG AA platforms
  - server: Linux/OSX
- Externals
  - BOOST: threads, possibly asio for networking
  - GSI security and VOMS integration (sharing code with LFC/DPM)

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Proposed Schedule

- April'08 - first release to ATLAS online community
  - reduced feature set
  - Goal: sufficient to support of proxy cache for HLT
- June'08 - functionally complete
  - Goal: passes existing CORAL and COOL test suites
  - start of performance optimisation
- October'08 - first full s/w release
  - start of experiment validation (validation clusters)
- December'08 - production deployment
  - in parallel to direct DB connections to prod clusters
- March'09 - phase out direct external access
  - only for validated applications

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Summary

- CORAL is widely used as foundation for online and offline database applications
- By introducing a CORAL server in front of the Oracle/MySQL databases we could remove several of the remaining deployment issues
  - security, scalability and s/w distribution
- Caching possibilities being investigated with contributors from ATLAS online
- The development would profit from significant reuse of existing components and could be integrated adiabatically with minimal impact on existing code.
- Production deployment by end of this year seems doable with existing resources

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it