EGI- FORUM Munich 2012

Network glitch issue in a CORAL client application

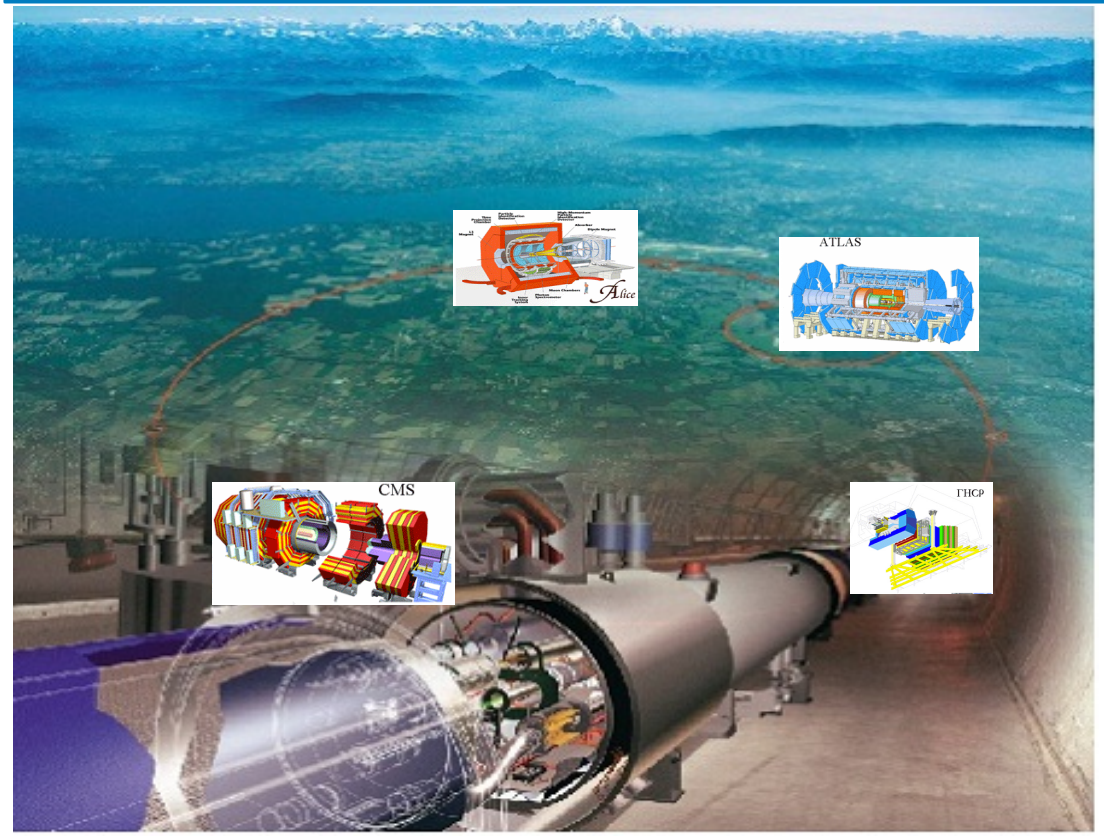
Raffaello Trentadue
On behalf of the
IT-ES Persistency Framework group

e-infrastructure

Outline

- CORAL INTRODUCTION
  - ✓ The context where CORAL was designed and implemented
  - ✓ What is CORAL?
  - ✓ CORAL back-ends
  - ✓ Oracle Call Interface
- NETWORK GLITCH ISSUE
  - ✓ Introduction: the experiment point of view
  - ✓ Some Oracle concepts
  - ✓ CORAL Session and Transaction isolation level
  - ✓ What is the Oracle/CORAL reaction
  - ✓ How to simulate the glitch
  - ✓ Test suite implementation and road map investigation
  - ✓ Proposal and implementation for a fix
- CONCLUSIONS

LHC challenge



✓ *Enormous amount of data to be processed and analyzed (hundreds of petabytes over the whole lifetime).*

✓ *Impossible to implement a unique CERN analysis facility*

✓ *Boost of the development of the grid computing infrastructure and technology.*

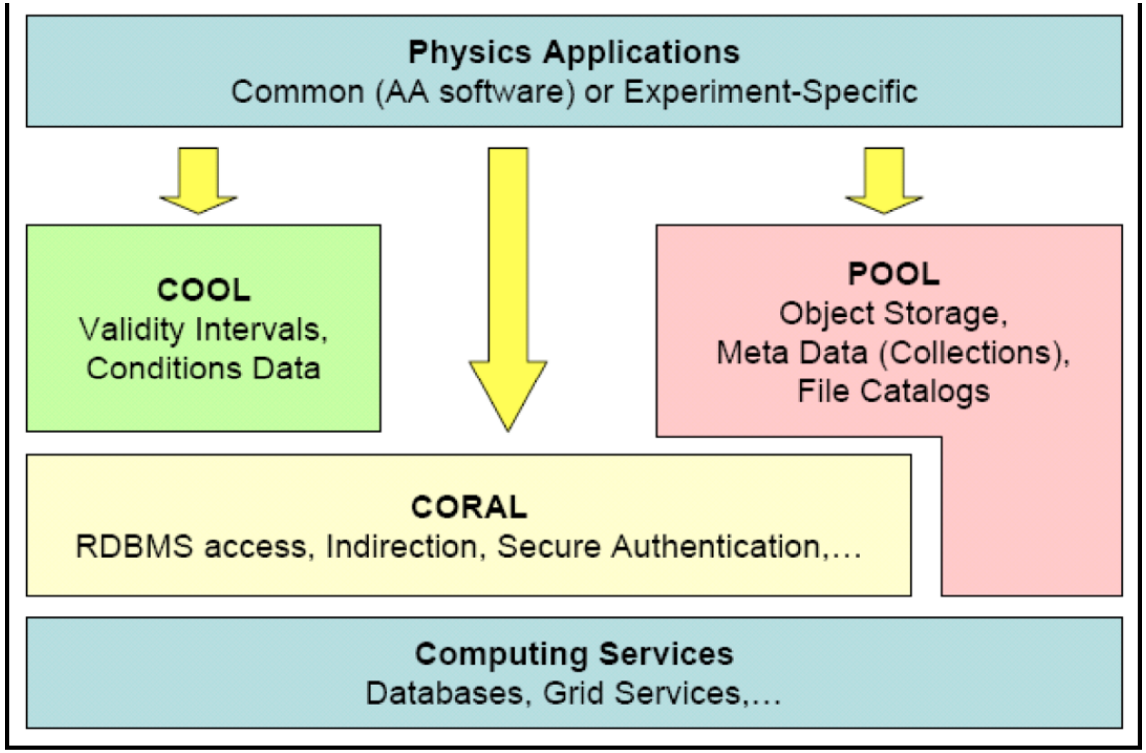
✓ *A distributed analysis model implies an in-homogeneity in the data storage infrastructure across the different institutes and across the long LHC lifetime.*

✓ *The computing infrastructure has to be easily maintainable and adaptable.*

Persistency framework

In three of the experiments (ATLAS, CMS and LHCb), some types of data are stored and accessed using the software developed by the Persistency Framework (PF) within the Application Area (AA) of the LHC Computing Grid (LCG) to find common solutions for the LHC experiments.

✓ POOL is a generic hybrid store for C++ objects, metadata catalogues and collections, using streaming and relational technologies.



✓ COOL provides specific software to handle the time variation and versioning of conditions data.

✓ CORAL is a generic abstraction layer with an SQL-free API for accessing relational databases.

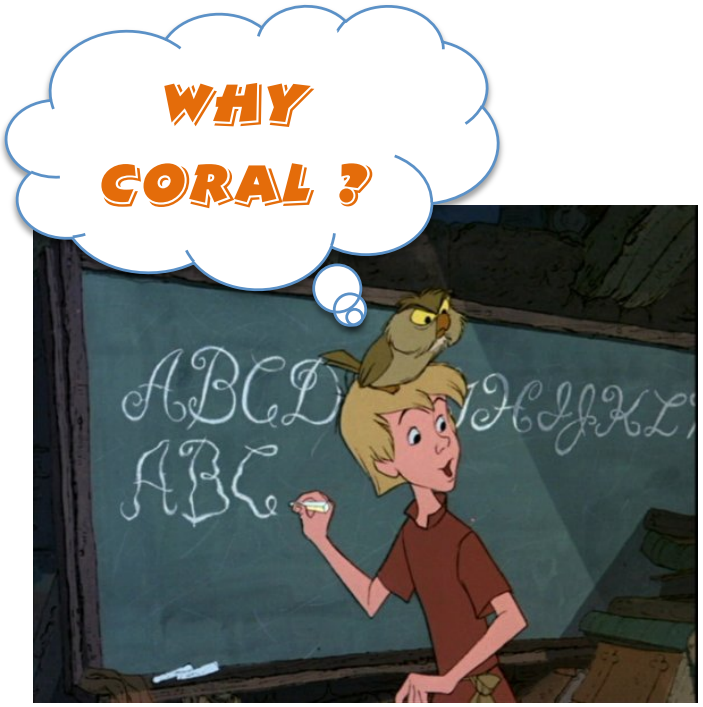
CORAL: introduction

The Common Relational Abstraction Layer (CORAL) is a C++ framework to access data in relational databases.

The C++ API of CORAL consists of a set of SQL-free abstract interfaces that isolate the user code from the database implementation technology.

Python bindings of the API are also available.

Why CORAL?



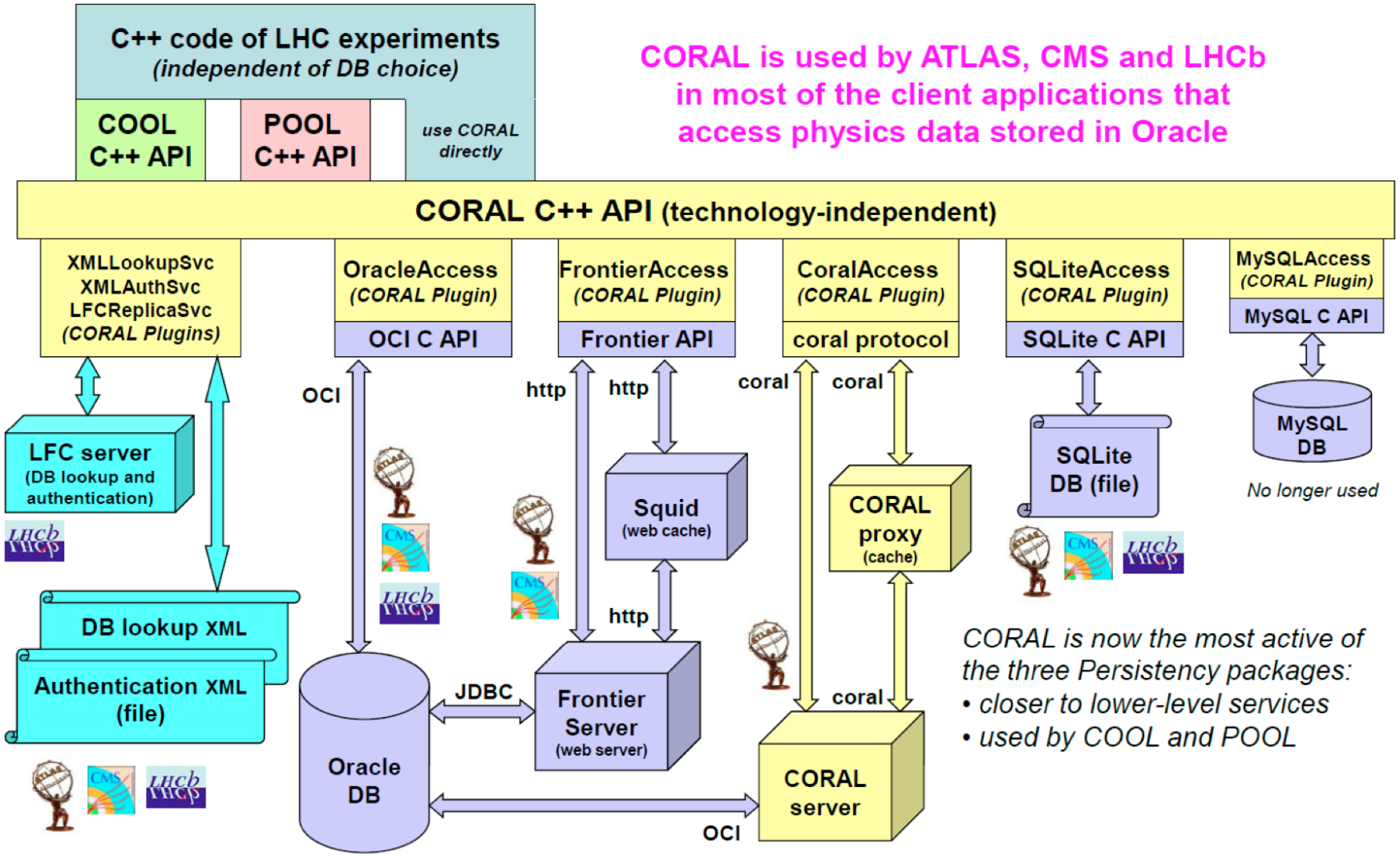
In the distributed data analysis model, there is an in-homogeneity in the data storage (database) infrastructure and security policies across the different institutes.

CORAL provides a set of C++ libraries for several database back-ends:

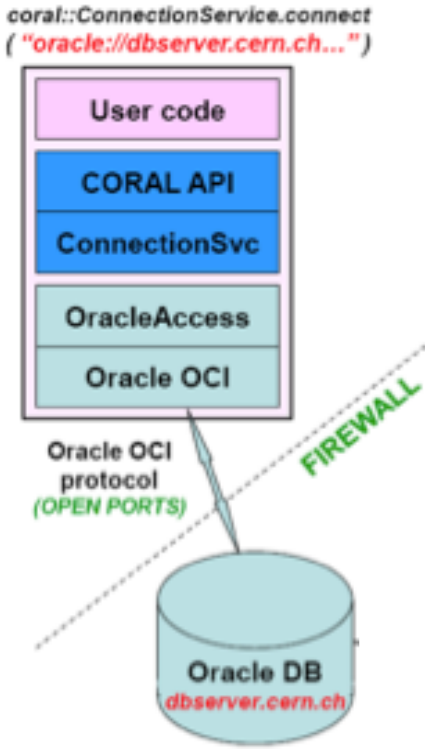
- ✓ *local access to SQLite files;*
- ✓ *direct client access to Oracle and MySQL servers;*
- ✓ *read-only access to Oracle through the FroNTier/Squid or CoralServer/CoralServerProxy server/cache system.*

- ✓ *Users write the same code for all back-ends*
- ✓ *A detailed knowledge of the many SQL flavors is not required*
- ✓ *The SQL commands specific to each backend are executed by the relevant CORAL libraries, which are loaded at run-time by a special plugin infrastructure.*

CORAL Backends



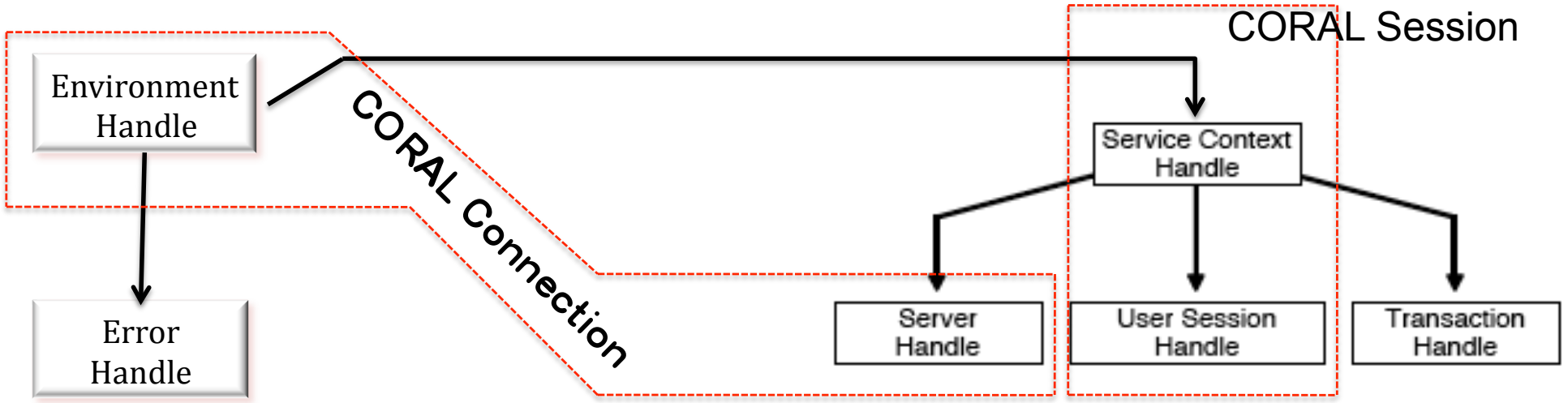
Oracle OCI Access



The Oracle Call Interface (OCI) is an application programming interface (API) and allows the user to avoid a complex OCI implementations.

CORAL uses OCI internally and allows the user to avoid a complex OCI implementations.

CORAL connection by Oracle Call Interface



✓ *The environment handle defines a context in which all OCI functions are invoked*

✓ *The service context handle contains three handles as its attributes, that represent a server connection, a user session, and a transaction.*

✓ *A server handle identifies a connection to a database*

✓ *A user session handle defines a user's roles and privileges (also known as the user's security domain), and the operational context in which the calls execute.*

Network glitch introduction

During the last two years the three experiments that make use of CORAL (ATLAS, CMS and LHCb) experienced a similar issue: an Oracle error appeared during the execution of some operations against the Oracle database, even though in different circumstances.

The oracle error found is : **ORA-03113**

In some case, this error triggered an infinite loop , causing an application hang.
As reported by the Oracle site:

"ORA-03113: end-of-file on communication channel
Cause: The connection between Client and Server process was broken.
Action: There was a communication error that requires further investigation."

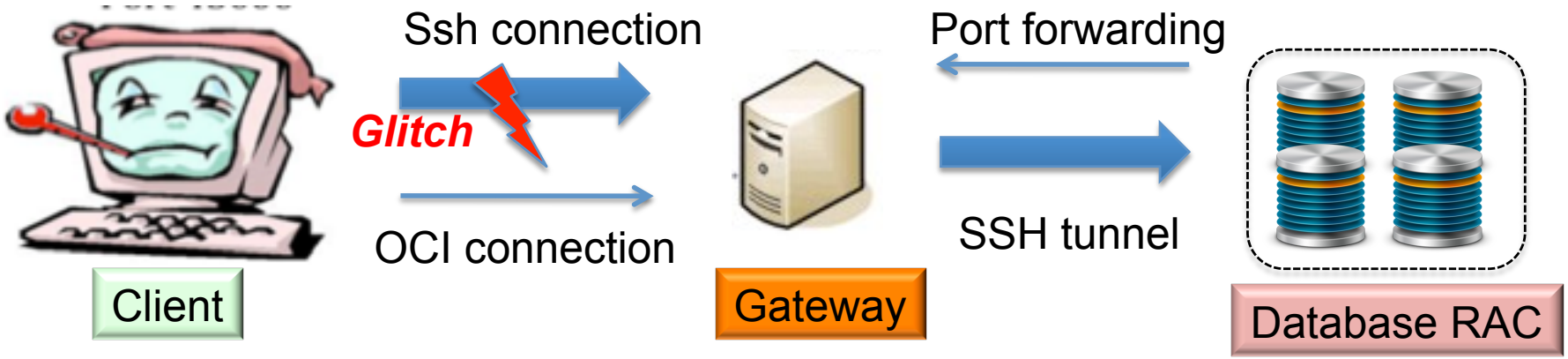
The most likely cause of this issue is **an instability of the network , leading to a temporary connection break between Client and Server.**

What is Oracle and CORAL reactions

In case of network glitch:

- **Oracle reaction:**
  - ✓ the server is not able to check if the connection with the client is still alive;
  - ✓ the client probes the server by means of the first instruction sent to the database after the glitch and discovers that the server is no longer reachable;
  - ✓ the client sends the error ORA-03113 to the user.
- **CORAL reaction:**
  - ✓ the current version of CORAL does not have any method to check the connection validity;
  - ✓ after a glitch, the first operation against the database (OCI call) triggers an exception that reports the Oracle error, as the OCI handles are no longer valid.

How to simulate a network glitch



✓ *The connection has been split in two branches.*

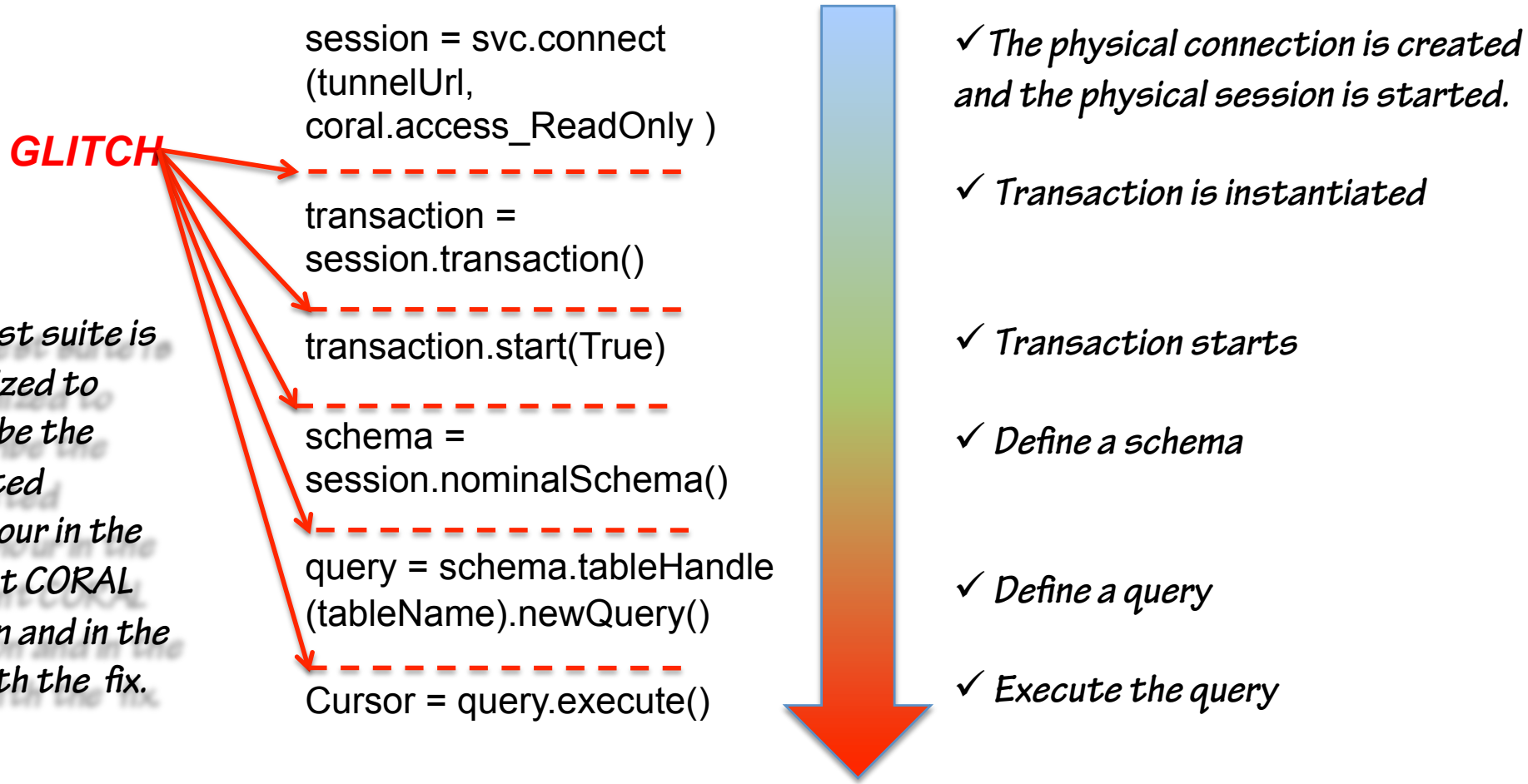
✓ *The Client use a free port to forward the traffic on the Gateway through the ssh.*

✓ *The Gateway is in contact with the listner of the database server.*

Our strategy consists in killing the ssh process to break the tunnel, simulating the break of the connection.
The method has been validated using SQL developer to monitor the sessions open for the client during the application work-flow (before, during and after the glitch).

Network glitch test suite

A python test suite has been implemented, including a set of tests aiming to kill the tunnel and simulate the temporary network interruption with the Oracle server.



The test suite is optimized to describe the expected behaviour in the current CORAL version and in the one with the fix.

What CORAL is supposed to do after a glitch 1

When a network glitch occurs, the reaction of CORAL should depend on the type of the session and transaction active.

SESSION

CORAL presently supports two types of sessions, Update and ReadOnly.

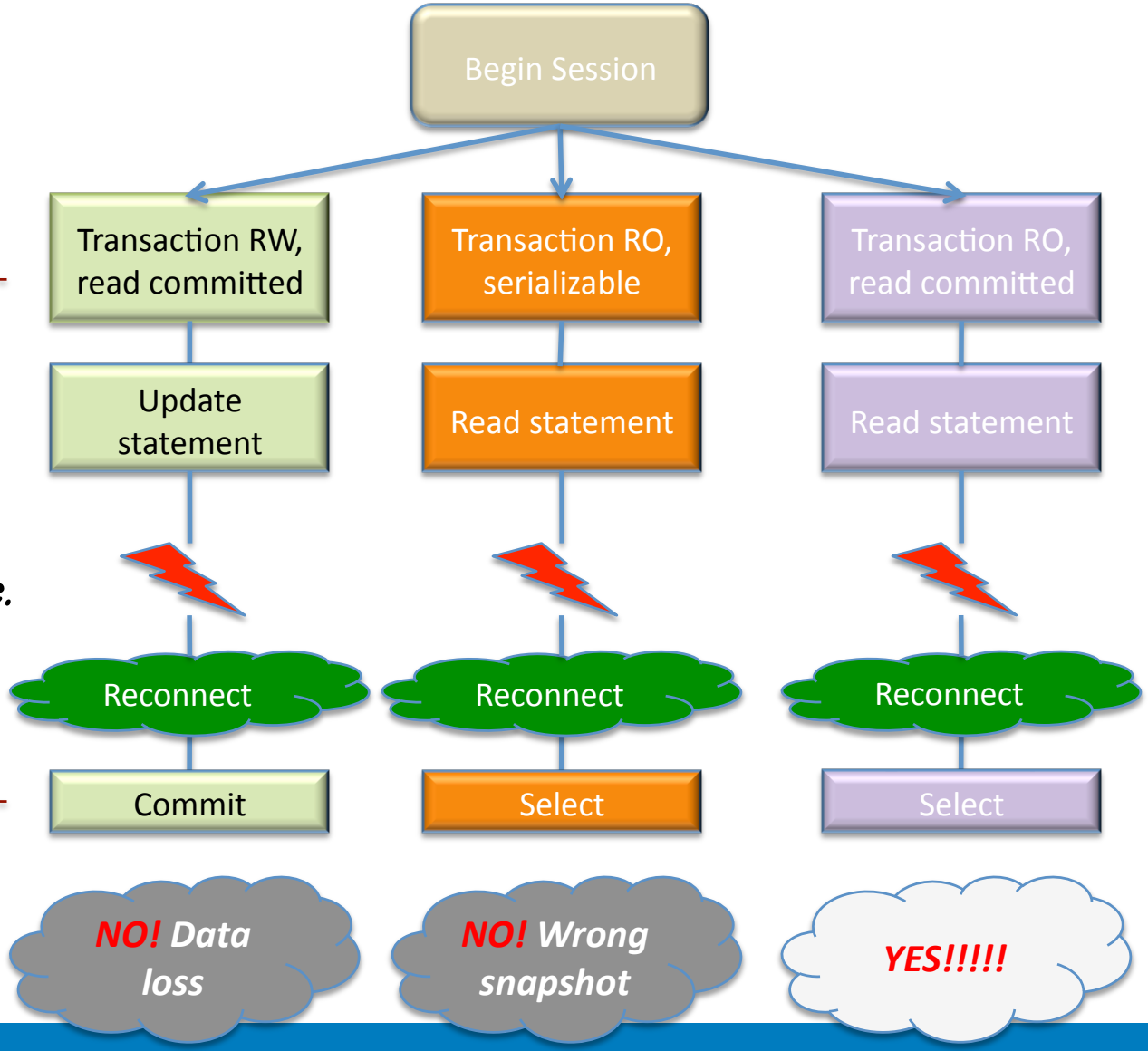
if in ReadOnly mode CORAL performs some checks to impose extra constraints, i.e. forbid DDL and DML and only allow SELECTs. For instance, an INSERT will throw a CORAL exception.

TRANSACTION

	Statement-level read consistency ("non serializable")	Transaction-level read-consistency ("serializable")
Updates allowed ("read-write")	Read Committed isolation level	Serializable isolation level
Updates forbidden ("read-only")	-	Read-Only isolation level

What CORAL is supposed to do after a glitch 2

When a network glitch occurs, the reaction of CORAL should depend on the type of the session and transaction active.



A possible solution in Oracle

✓ *Before any new implementation in CORAL, the reaction from Oracle has been thoroughly analyzed with the aim of understanding whether it is already able to perform somehow the reconnection, without any CORAL modification.*

Transparent Application Failover (TAF).
It is an Oracle feature that can automatically open a new instance on the same or a new node in case of instance crash.

Ex. *If the crash of the instance occurs during the loop of a select statement, TAF is able to restore the result of the query on the other node and the right position of the cursor used in the loop.*

However, is an instance crash the same as a network glitch?

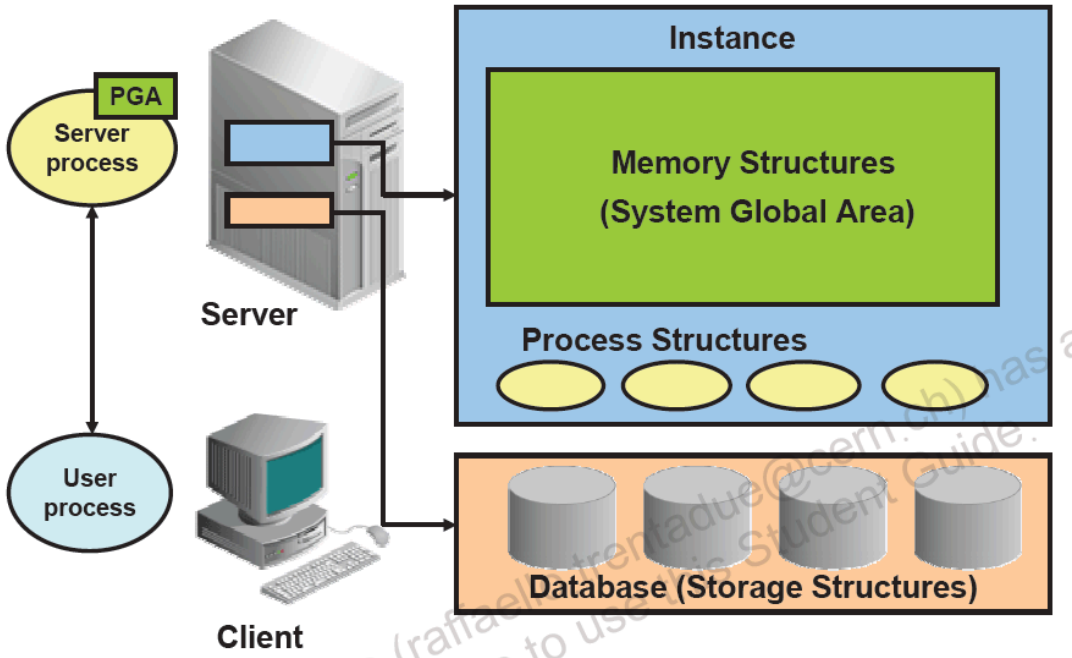
An instance is a set of processes and memory structures . It exists in the CPU and in the memory on the server node.

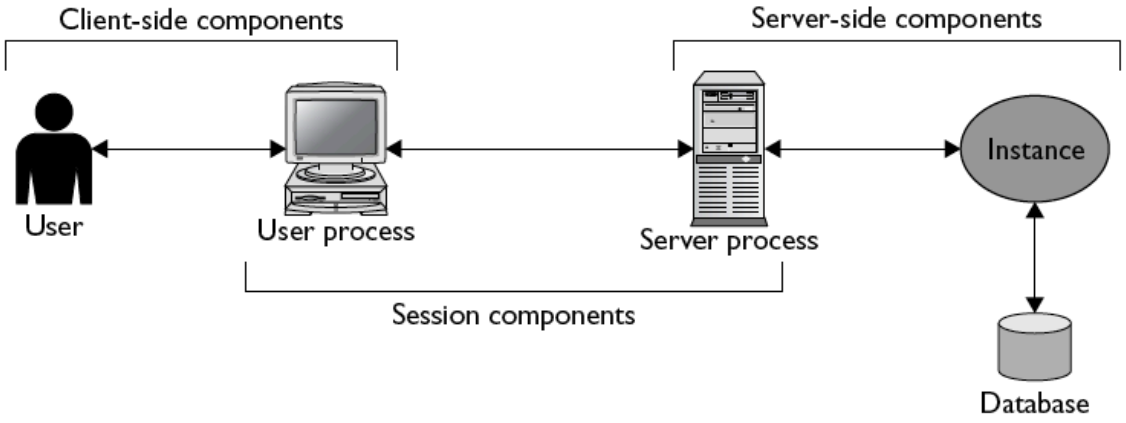
..... some Oracle definitions

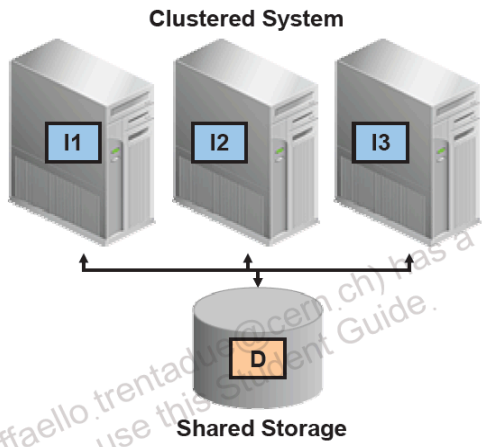
A *connection* is a "physical circuit", a pathway to a database.

A *session* is a user process in communication with a server process.

Users establish sessions against the instance, and the instance then manages the access to the database.







Conclusions about TAF

An instance crash is not a network glitch.

This conclusion could be also supported by another observation:

Thanks to the DBAs, we obtained the privileges to kill the session while the client application was running. The test showed that killing the session has the same effects as killing the ssh tunnel.

That is, the network glitch seems equivalent to the session kill.

Furthermore the TAF seems to work as expected in the case of the instance crash.
On the contrary, it does not work with the session kill (the behaviour is different depending on the Oracle version 10g/11g).

The TAF is not helpful to fix the network glitch issue in CORAL.

How to fix the network glitch issue in CORAL

There are two possible approaches to fix the network glitch issue:

✓ *To react to the error: this would be the safest solution as it would detect the connection loss at the level of any OCI call. However it needs a huge change of the code and the class structures.*

✓ *To prevent the error: this solution allows us to keep the present structure and fix the issue just adding few functions in a few classes.*

At the moment the second strategy has been chosen as it offers an easier implementation. However in the future also the first strategy could be re-considered.

How to prevent the error

Trigger a reconnection mechanism by means of the check of the connection and session validity at the beginning of few and crucial instructions (session, transaction, schema).

A probe function has been implemented using the OCIServerVersion function, that checks the server accessibility.

If a network glitch is detected, CORAL reacts in the following way:

- ✓ *If the transaction is not active yet CORAL triggers the reconnection for any type of session;*
- ✓ *If a transaction is already active, CORAL triggers the reconnection only if the transaction is defined as RO-read committed mode.*

The reconnection consists of a procedure to create a new physical connection and enable a new user session. From the point of view of CORAL this means just refreshing all the OCI handles without changing their pointers.

In all the other use cases an exception is thrown.

Conclusion

The network glitch issue has been reproduced by means of a simulation set-up and a suitable test suite.

This enabled a thorough analysis that allowed us to identify all the possible solutions.

An Oracle solution (TAF) has been evaluated and discarded

Two possible solutions to be implemented in CORAL have been carefully evaluated.

The strategy of preventing the error has been chosen. A reconnection procedure which consists of refreshing all the OCI handles is triggered in some cases if the connection is not valid.

The tests of the new implementation have been successfully executed in Oracle 10g and should be validated also in Oracle 11g.

Operation	Execution Time
Session = svc.connect()	2.45 s
session.transaction()	10.01 µs
transaction.start()	3.86 ms
session.nominalSchema()	0.92 ms
schema.tableHandle (tableName).newQuery()	41.54 ms
query.execute()	16.88 ms