Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

Re-engineering SAM or Changing the Engine in the Train While it is Running

R. Illingworth, M. Mengel, A. Norman, S. White

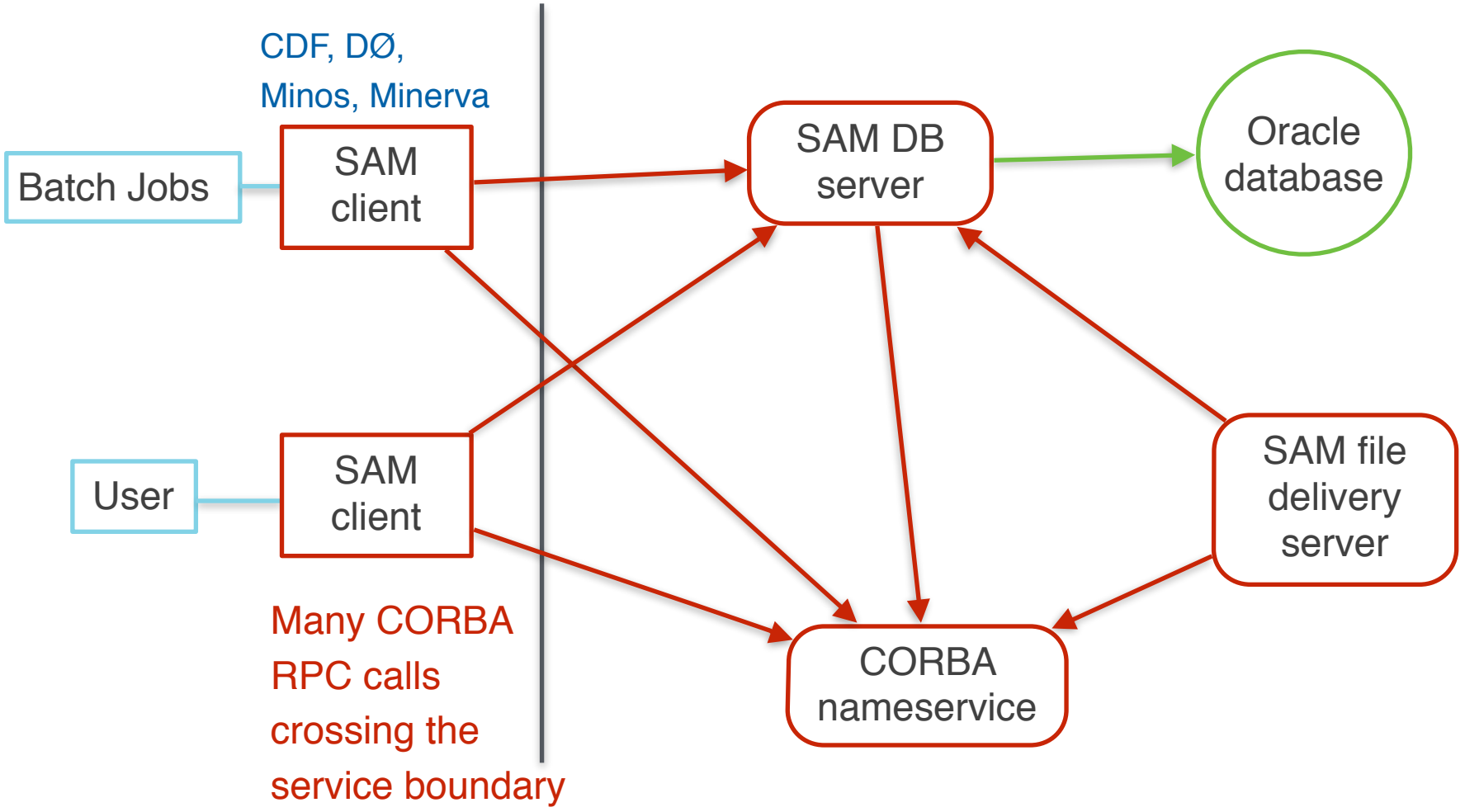
Fermilab, Scientific Computing Division
Scientific Data Management

Overview

- SAM was the data management system used during Run II of the Tevatron
  - Provides metadata cataloguing, file location catalogue, and managed the transfer and delivery of files to processing jobs
- The original design dates back to 1997 (pre-Grid) and relies heavily on legacy software components
- **HIGHLY Successful for Run II**
- Fermilab Intensity Frontier experiments needed a full featured data management system that could integrate with their tools
  - We decided that keeping the SAM system viable for these users
  - Required bringing it up to date w/ modern software tools/protocols
  - This had to be accomplished while experiments were already making use of the system

Original design

The original design for SAM used CORBA as the Remote Procedure Call layer and Oracle as the backend database

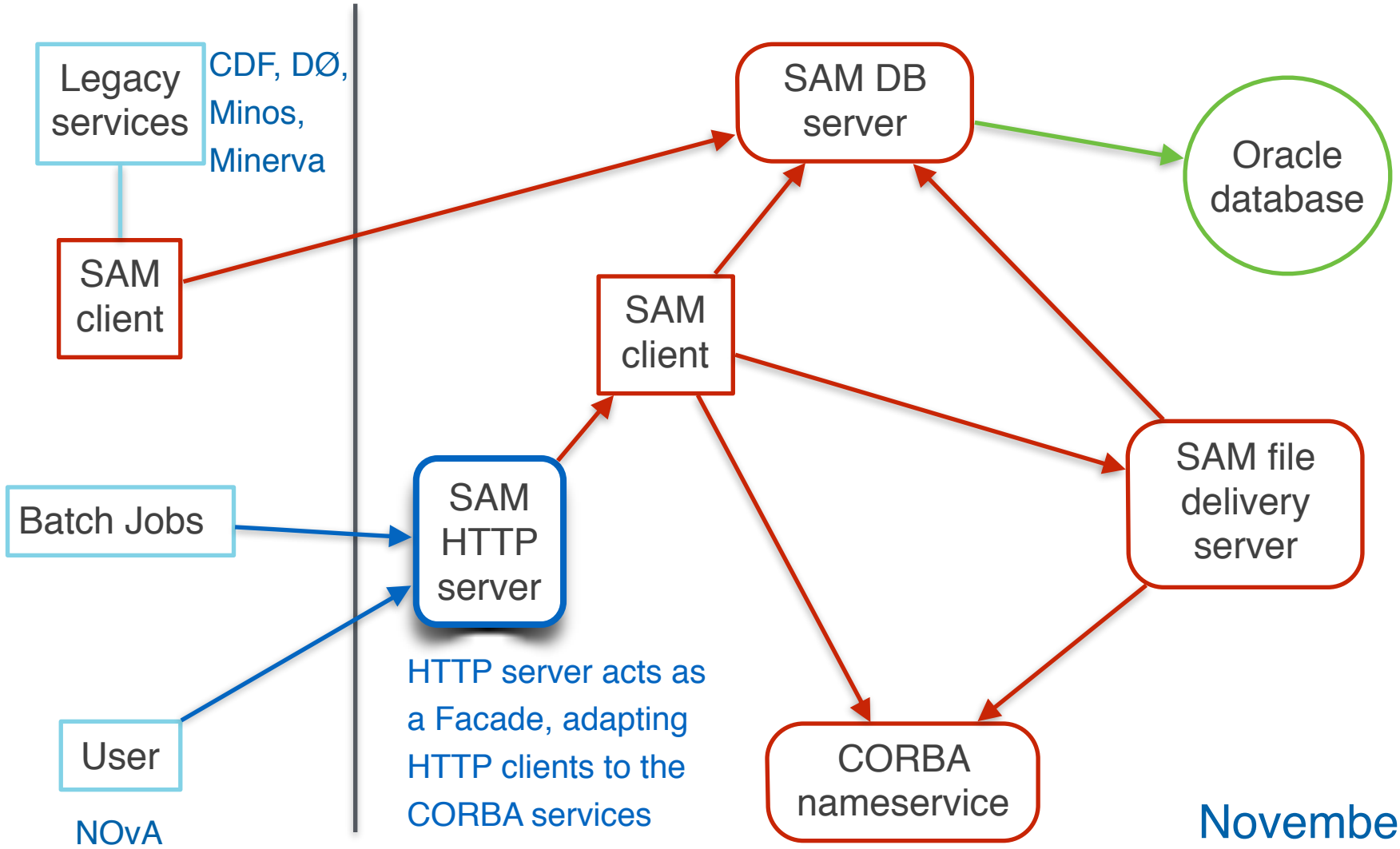


Problems with the old model and the initial solution

- CORBA is a complex protocol with a limited number of available implementations
  - Also very rigid and hard to modify interfaces
- Required distributing large client libraries along with jobs just to talk to the data management system
- We decided to replace the end user client with a REST style HTTP based interface
  - The HTTP server became a *Facade* with *Adaptors* for the CORBA services
  - Required clients to change their scripts once, but after that can be evolved in a backwards compatible way
  - Also took the opportunity to simplify some interfaces and remove others

Putting up the Facades

Add HTTP server to adapt the existing CORBA based services



November 2011

Updating the query language

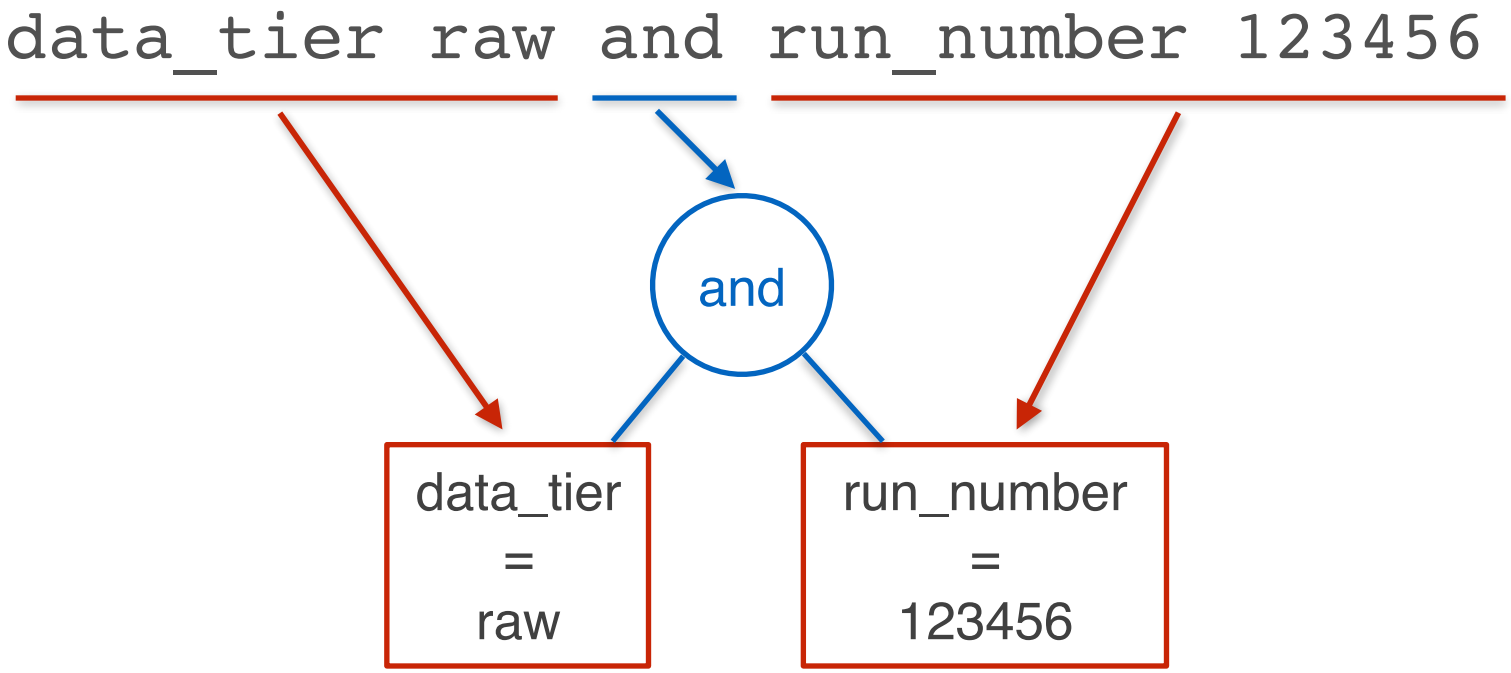
- SAM provides a simple query language that insulates users from having to deal with SQL
  - The original implementation was Oracle dependent and had numerous deficiencies

- The query language allows selecting data files based on their metadata properties
- More advanced features include selecting files based on file provenance (child or parent files)

- As part of the updating project (and new facade), we replaced it with an improved version that was non-Oracle dependent

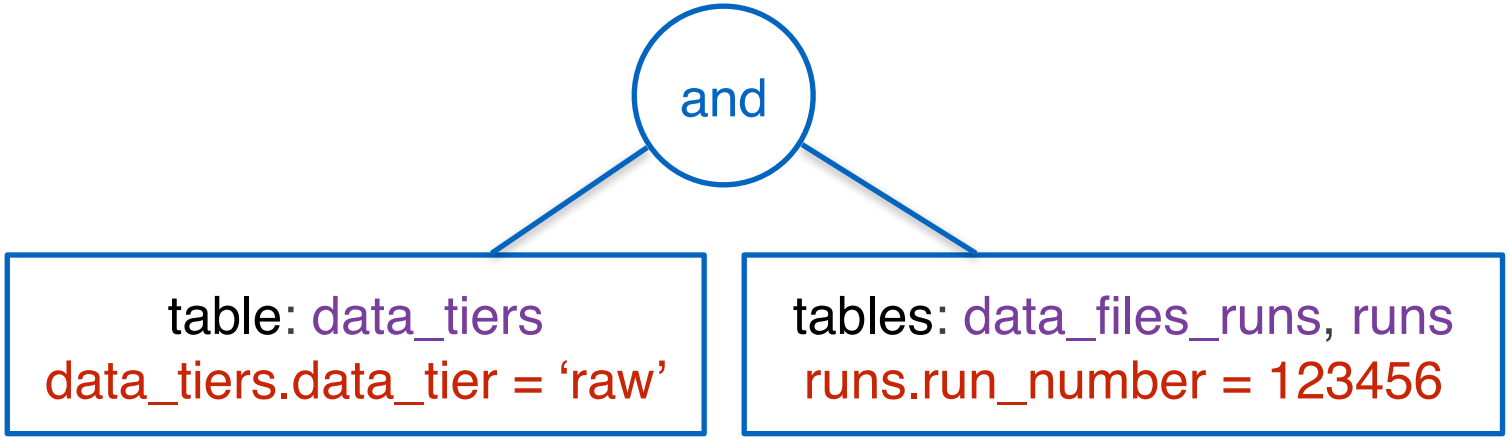
Example query evaluation

Parse query into component parts



Convert to SQL

Get database tables and constraints for each node



Combine tables and constraints into SQL expression

```
select … from data_files
join data_tiers join data_files_runs join runs
where
data_tiers.data_tier = 'raw' and runs.run_number = 123456
```

Benefits of the query evaluator

- This approach of building up SQL expressions from small pieces allows end users the flexibility to create complex queries without knowing the internal database schema

- It is then possible to alter the implementation, schema, or even the database engine without affecting the end users

- Preview: This was essential when migrating from the legacy Oracle backend to a new PostgreSQL database backend

Next stage

- With the HTTP interface in place the CORBA implementation is now hidden from the users
- The inclusion of the query builder also hides the database scheme from the user

- This means server components can be progressively replaced by new implementations that bypass the old layer and communicate directly with the database
- Some legacy services remained running in parallel using CORBA directly
  - These were migrated to HTTP as time permitted

Next stage

Start moving database operations to HTTP server

Legacy services converted to HTTP

Some database operations directly implemented in the HTTP server

Batch Jobs

User

SAM HTTP server

HTTP server acts as a Facade, adapting HTTP clients to the CORBA services

Oracle database

SAM DB server

SAM client

SAM file delivery server

CORBA nameservice

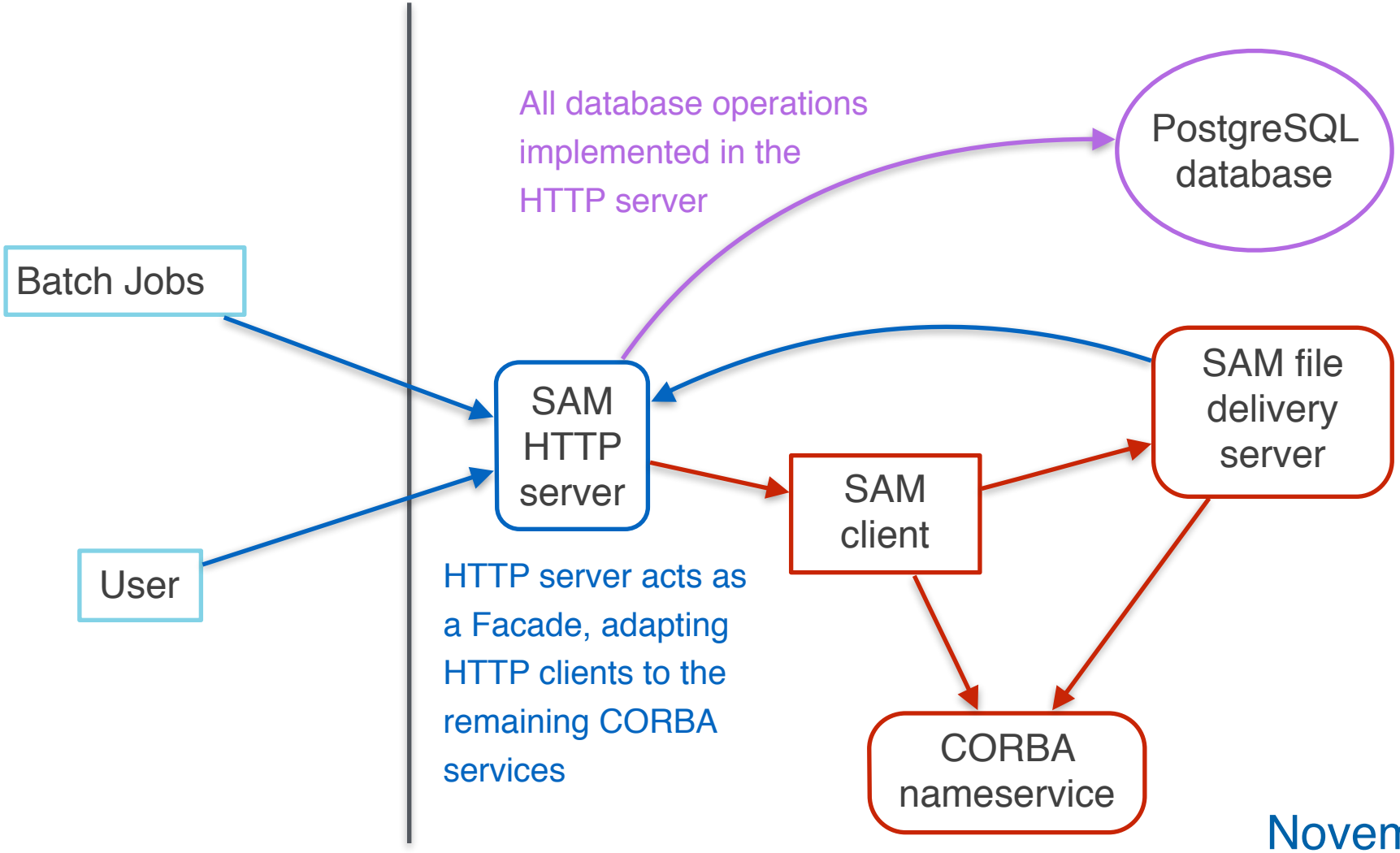
January 2013

Further evolution

- Once all required database components are implemented using HTTP the CORBA DB server could be removed completely
  - Since clients were only using the HTTP interfaces this was transparent to them
- We were also able, thanks to the simplification of the interface, to clean up the database schema and migrate the database from Oracle to PostgreSQL
  - Again this was transparent to clients as they did not access the database directly

Further evolution

HTTP server used for all database interactions; move to PostgreSQL



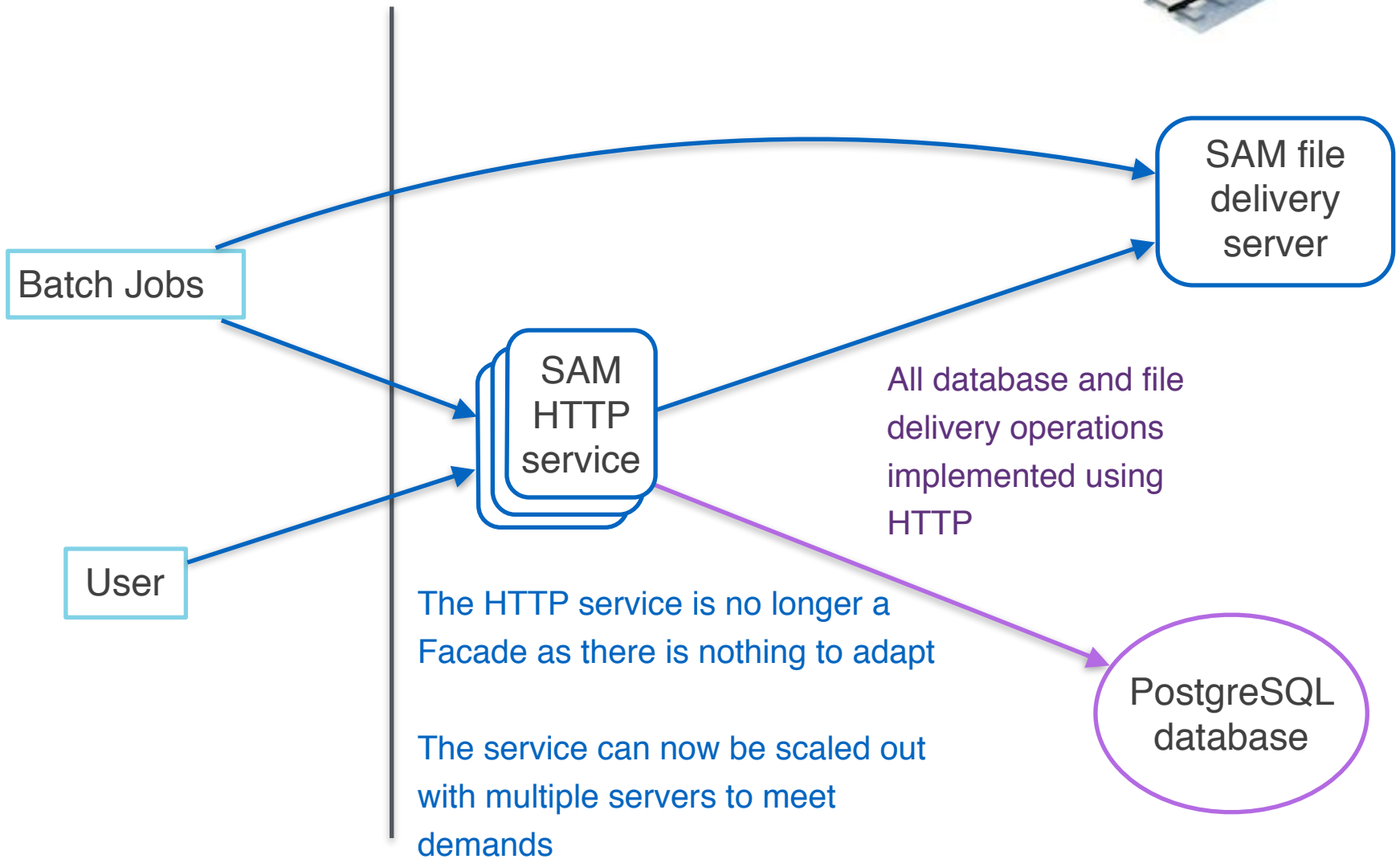
November 2014

Future configuration

- The final stage will be to replace the remaining CORBA services with HTTP
  - This will allow direct communication between clients and the file delivery components, improving scalability
  - As the original interface design allowed for URL redirection this will not require any client changes

Future configuration

Simplify by removing all remaining CORBA services



Conclusions

- The approach of defining a new interface as an adaptor to a legacy interface allowed
  - making major changes to the internal implementation of the service
  - without impacting users after the initial move to the new interface
  - while maintaining operations
  - spreading out the development effort
- Using this approach we successfully migrated SAM instances for 12 Intensity Frontier experiments without significantly disrupting their activities