Towards a new conditions data infrastructure

Evgeny Alexandrov, Luca Canali, Davide Costanzo, Andrea Formica, Elizabeth Gallas, Mikhail Mineev, Nurcan Ozturk, Shaun Roe, Vakho Tsulaia, Marcelo Vogel

On behalf of the ATLAS S&C community

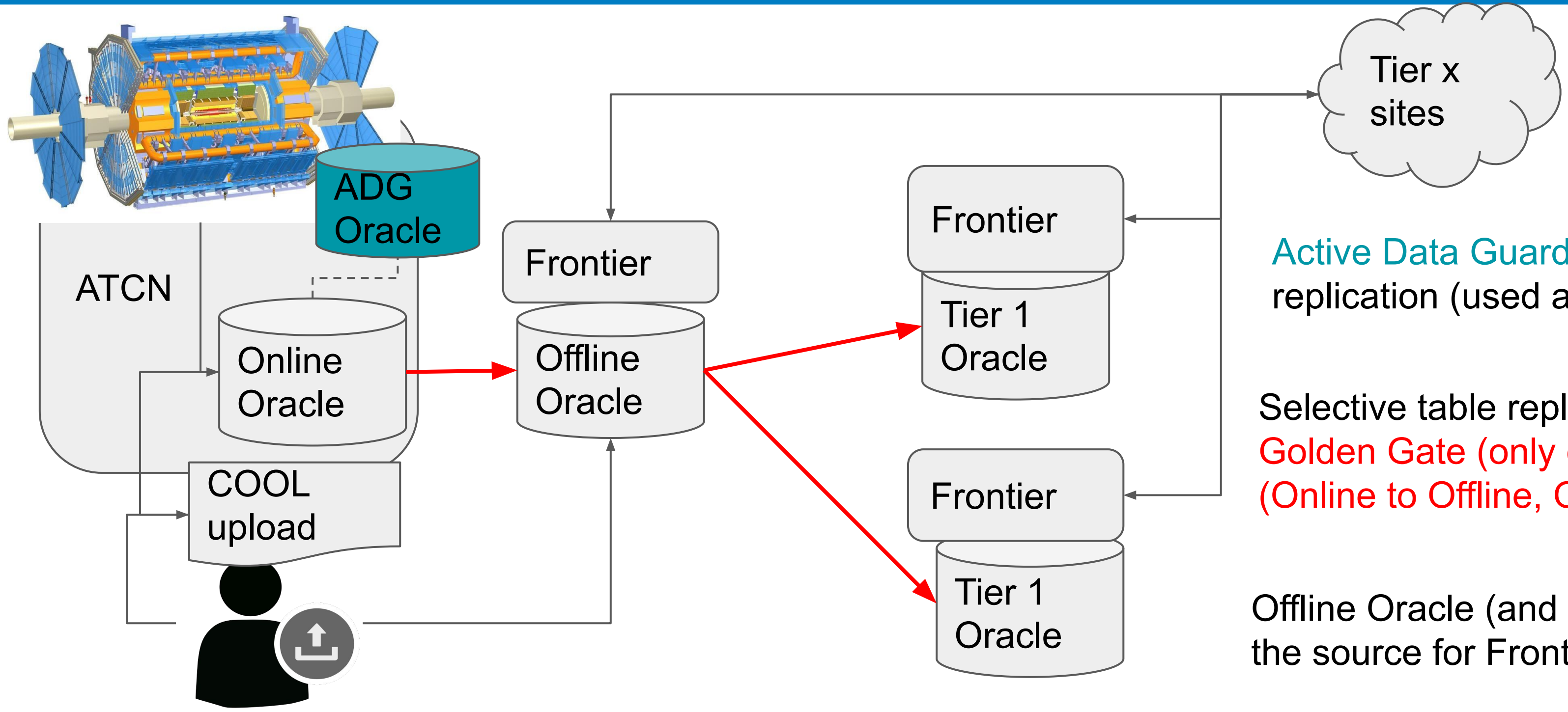
Outline

- ATLAS conditions data infrastructure
  - The architecture until now
  - Evolutions for Run 3

- The new architecture for CREST (ConditionsREST)
  - Motivations
  - CREST: Conditions as a REST microservice

- Ongoing activities
  - CREST ecosystem
  - CREST status and deployment of a demonstrator

Conditions Infrastructure: Run 2 -> Run 3



Active Data Guard is a binary DB replication (used also for backup)

Selective table replication via Golden Gate (only conditions) (Online to Offline, Offline to Tier 1)

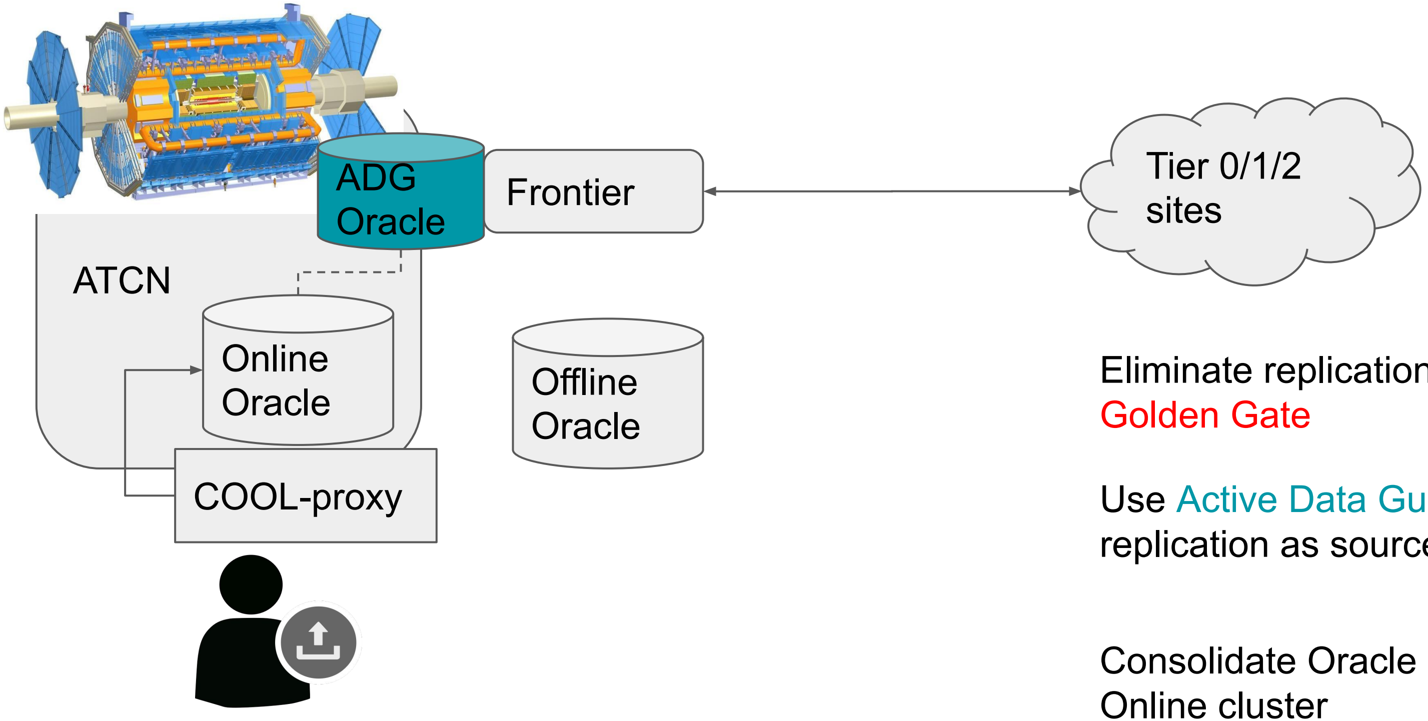
Offline Oracle (and Tier 1 replica) is the source for Frontier access

ATCN is the private ATLAS experiment network (access forbidden from General Public Network)

COOL (the in use Conditions DB) uploads via shared file system to access Online Oracle

Offline Oracle hosts many other Databases (metadata, authors, detector constructions, monitoring)

Conditions Infrastructure: Run 2 -> Run 3



COOL-proxy: web service allowed to access Online Oracle for conditions upload from General Public Network

Eliminate replication via Golden Gate

Use Active Data Guard binary replication as source for Frontier

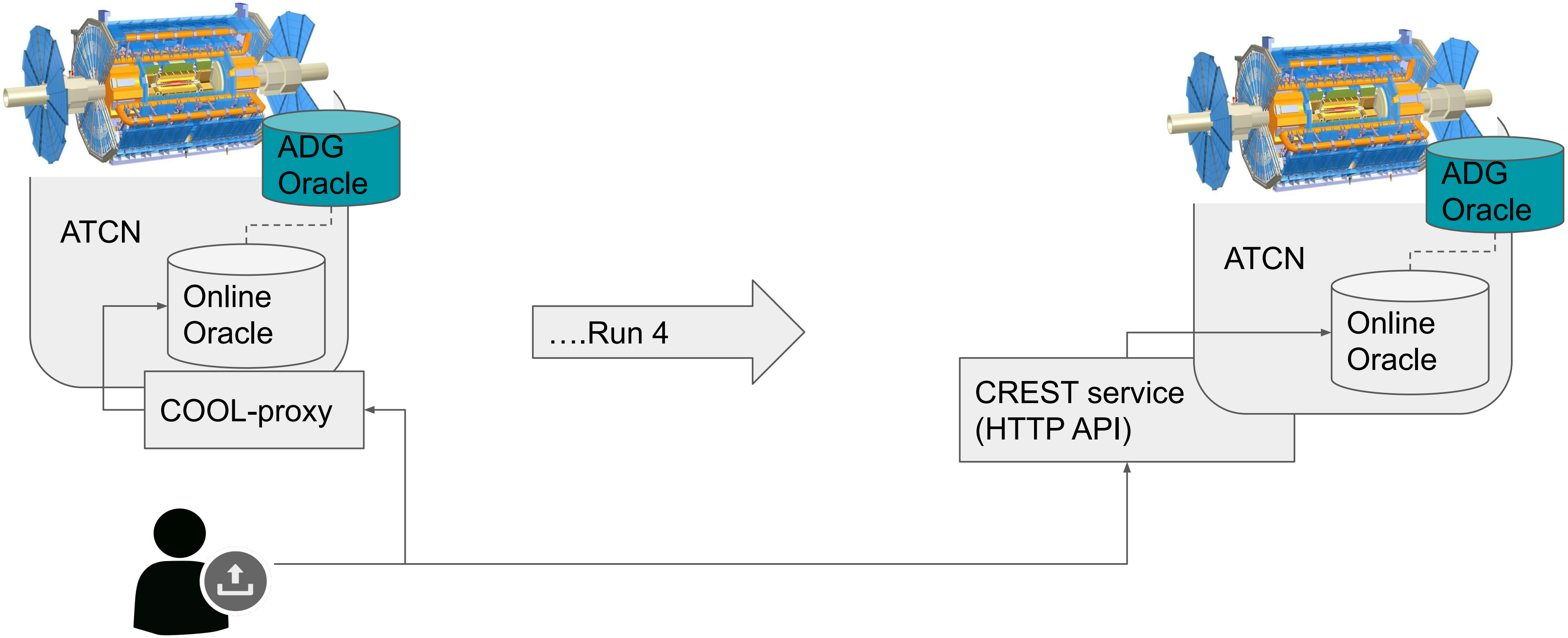
Consolidate Oracle resources in the Online cluster

Protect access for data uploads via a web proxy system (use of SSO for authentication)

The Run 3 Oracle/Frontier architecture

- Usage of Golden Gate technology for “selective” replication
  - Requires additional licensing and expertise
  - Used only by ATLAS
- Oracle resource consolidation
  - Optimize licensing costs
  - Provide an environment dedicated to “data processing” workflows
  - Avoid mixing **conditions, geometry**, … with many other applications which are using Oracle Offline cluster
  - Remove Tier 1s (cost optimization required for Oracle licensing after moving to *per-core* licensing model, respect to the previous *Campus* license model)
- Frontier
  - Simplify deployment by using only Oracle @ CERN

Evolution from COOL-proxy to CREST

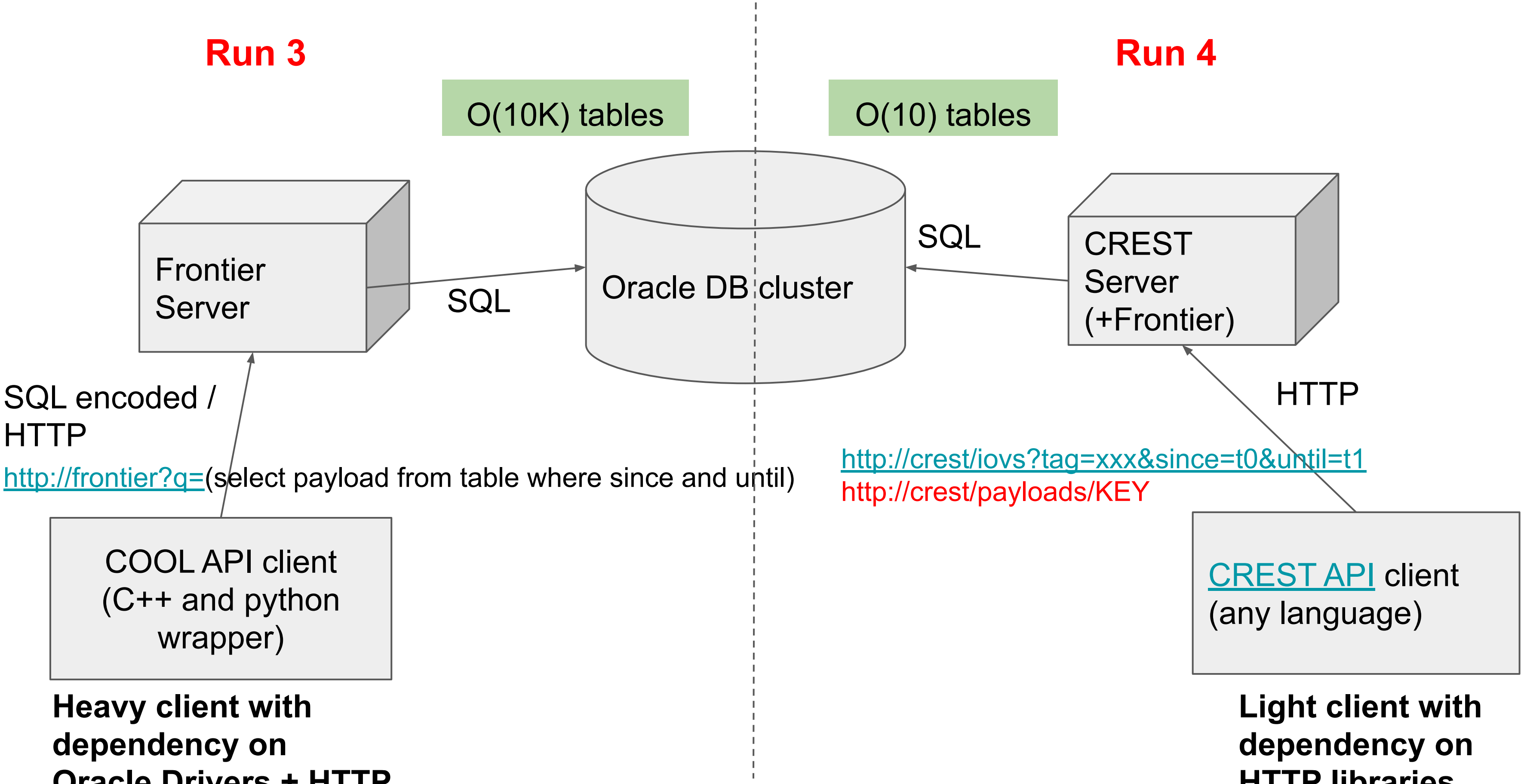


The present architecture will allow a smooth migration towards the Run 4 system (CREST)

CREST: Motivations for a new infrastructure

- Conditions data: volumes and access
  - Past experience: the volumes are almost constant during data taking periods, and they scale more with the number of "channels" in the detector than with Luminosity.
  - Read access: order of 10k jobs use these data during a normal processing.
  - Need for optimal caching of the data.
- Storage technology
  - ATLAS is using Oracle infrastructure provided by CERN (and Tier 1s until Run 3).
  - Evolution of DB versions (and costs) indicates that is wise to disentangle the clients from the storage solution.
- CREST data model and API
  - The simple data model design and REST API choice are based on a collaboration and prototyping between CMS and ATLAS conditions data experts that started back in 2016.
  - CREST API hides all technical aspects of the data model implementation: we used OpenApi specifications to describe and document our API.
  - HSF activities on conditions data aspects played a role to better define the scope of typical workflows consuming conditions.

Differences: COOL vs CREST



Run 3

O(10K) tables

Frontier Server

SQL

Oracle DB cluster

SQL encoded / HTTP

http://frontier?q=(select payload from table where since and until)

COOL API client (C++ and python wrapper)

Heavy client with dependency on Oracle Drivers + HTTP

Run 4

O(10) tables

SQL

CREST Server (+Frontier)

HTTP

http://crest/iovs?tag=xxx&since=t0&until=t1
http://crest/payloads/KEY

CREST API client (any language)

Light client with dependency on HTTP libraries

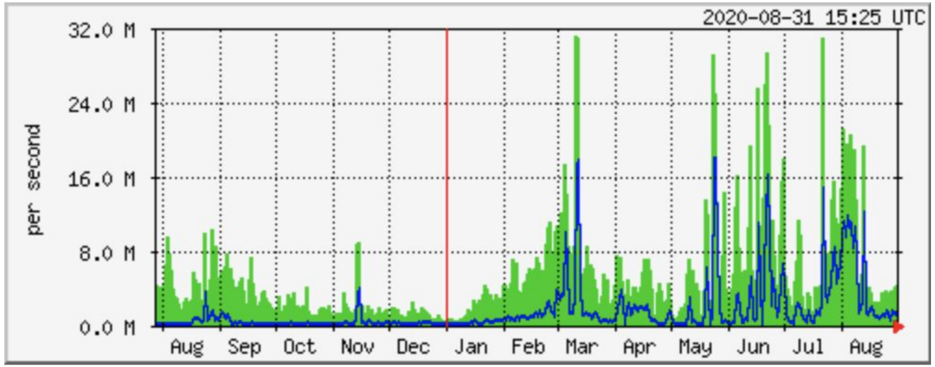
COOL vs CREST caching

- COOL caching
  - Limited support by design: even when using Frontier
  - **Plots: data loaded from squids - data taken from Frontier(Oracle)**
  - **Ratio: fetches/total**

ATLAS: ~1.5 MB/s / 5.1 MB/s => 30% (Avg)
~17 MB/s / 31 MB/s => 54% (Max)

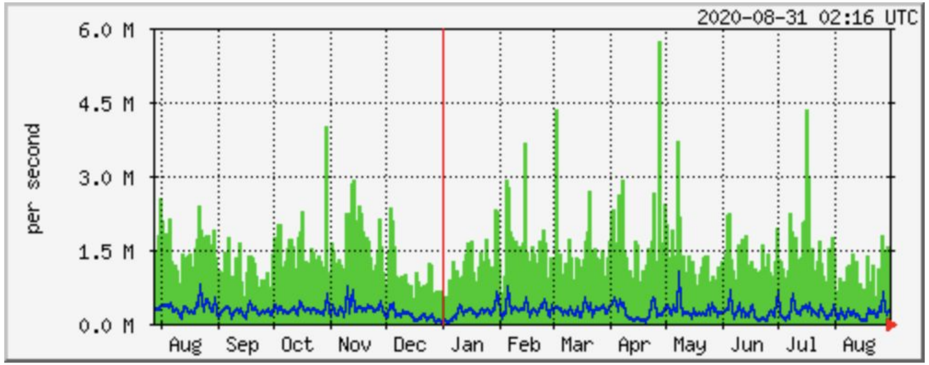
CMS: ~250 KB/s / 1350 KB/s => 20% (Avg)
~1 MB/s / 5.8 MB/s => 17% (Max)

`Yearly' Graph (1 Day Average)



	Max	Average	Current
Total	31.2 MB/s	5132.0 kB/s	4487.0 kB/s
Fetches	17.8 MB/s	1501.0 kB/s	1442.0 kB/s

`Yearly' Graph (1 Day Average)

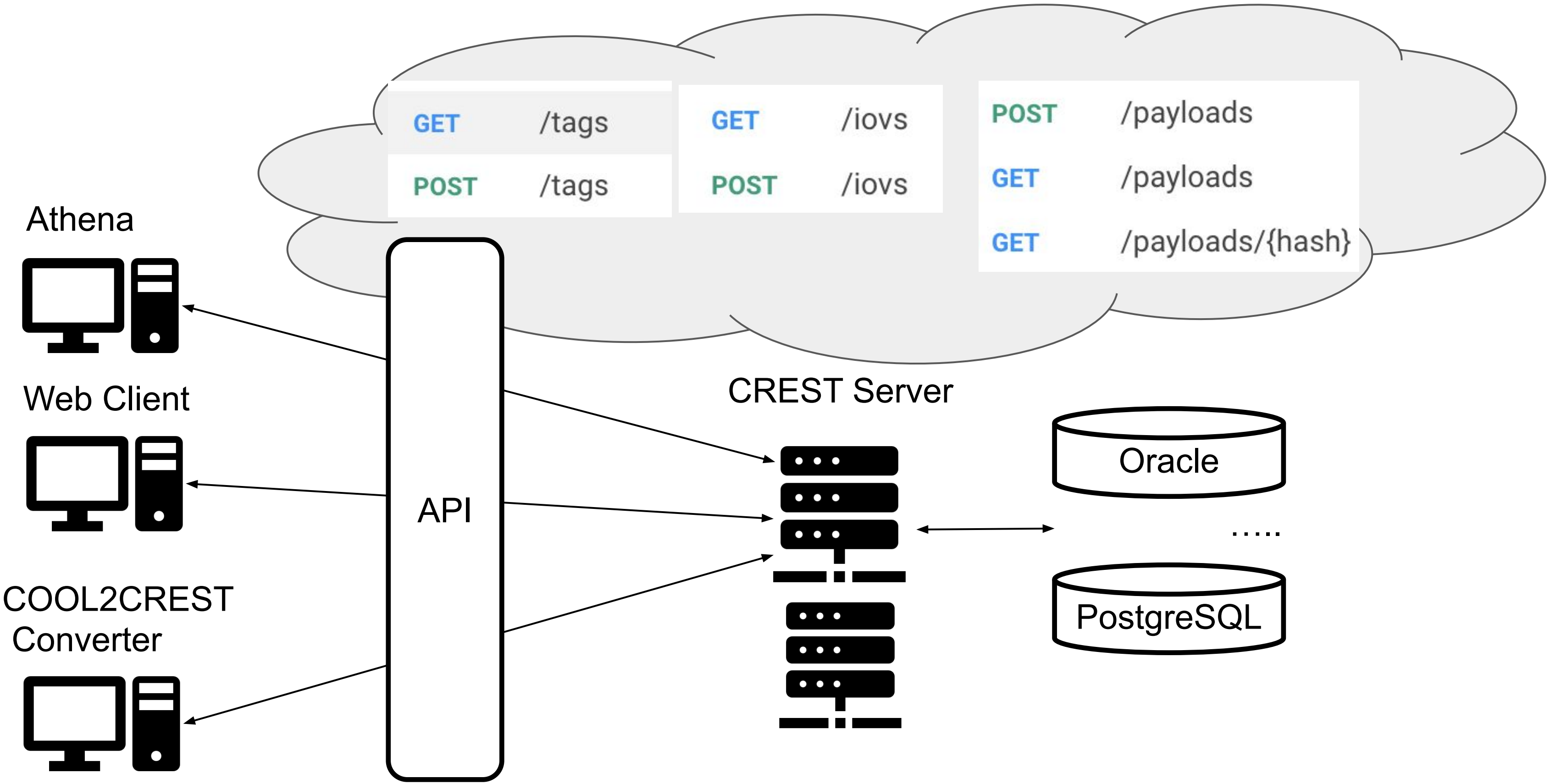


	Max	Average	Current
Total	5866.0 kB/s	1356.0 kB/s	1235.0 kB/s
Fetches	1042.0 kB/s	246.0 kB/s	246.0 kB/s

CREST architecture

- Data model
  - Simple data model using the main metadata concepts that are coming out from CMS and ATLAS experiences: ***Tags, IOVs, GlobalTags, Payloads***
  - Relational implementation: ~10 database tables
  - Optimal data caching by design (different payloads identified by unique *hash*)
  - RDBMS technology: many supported (Postgres, Oracle, H2, MySQL, …)
  - File-System: the simple metadata structure allows to easily dump into a filesystem
- CREST server
  - The CREST server exposes the REST API to manipulate the data
  - Implemented as a Spring Boot microservice
- CREST client
  - C++ client for ATLAS experts usage and Athena integration
  - Generated via OpenApi: *python, typescript etc*
  - Conversion tools: to migrate data from the present COOL conditions DB into CREST

CREST ecosystem



GET	/tags
POST	/tags

GET	/iovs
POST	/iovs

POST	/payloads
GET	/payloads
GET	/payloads/{hash}

CREST status

- Server and clients
  - Testbed for server deployed
  - Data migration tool: used to copy data from COOL into a standard JSON format
  - C++ client integration in Athena:
    - successful tests using simple jobs for specific subsystems
  - Gitlab workspace
  - CI workflows for automatic deployment in CERN Cloud (CERN SSO)
- Deployment of a demonstrator
  - Foreseen by the end of the year (2023)
  - Deploy in a dedicated *kubernetes cluster* @ CERN
  - Use the demonstrator for testing HLT workflows
- Goals
  - Test infrastructure over a large number of clients
  - Test specific workflows with conditions being updated in real-time
  - Verify that we refresh requests when new data are present (avoid stale cache)

Conclusions

- Evolutions of conditions data infrastructure for Run 3
  - ATLAS went through a big effort of reorganization of database infrastructure and related services
    - Simplification of workflows
    - Consolidation of resources
  - Removal of Golden Gate replication expected by the end of 2023

- Evolutions for Run 4: migrate to CREST service and data model
  - CREST is well underway (API, client library, tools)
  - Migration of data from COOL and preliminary tests using Athena ✓
  - Deployment of a CREST cluster in kubernetes ✓
  - Preparation of a demonstrator by the end of 2023 to validate HLT workflows