Data Bookkeeping Service 3
Providing event metadata in CMS

CHEP 2013, Amsterdam
17.10.2013

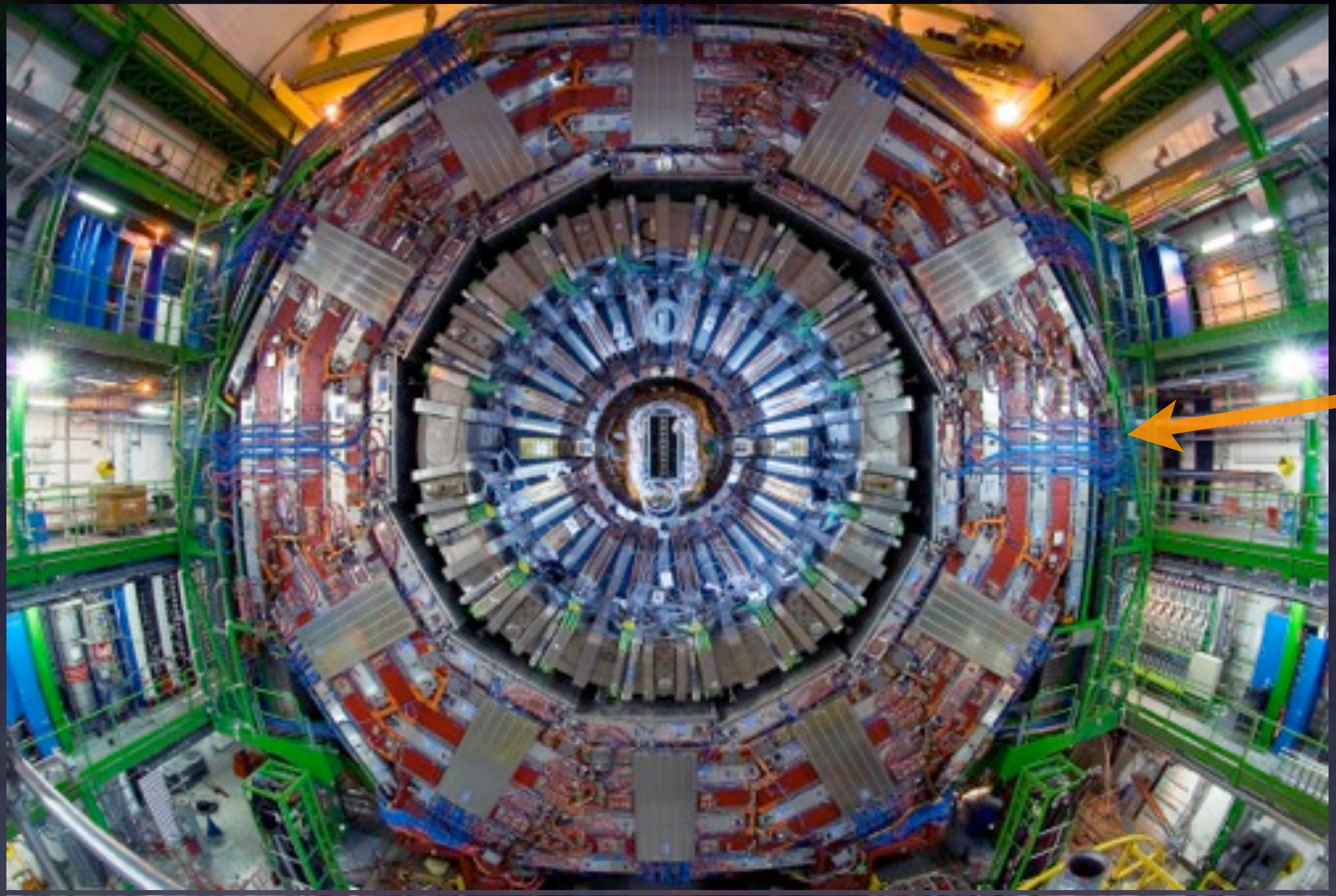
Manuel Giffels, Yuyi Guo, Daniel Riley
for the CMS Collaboration

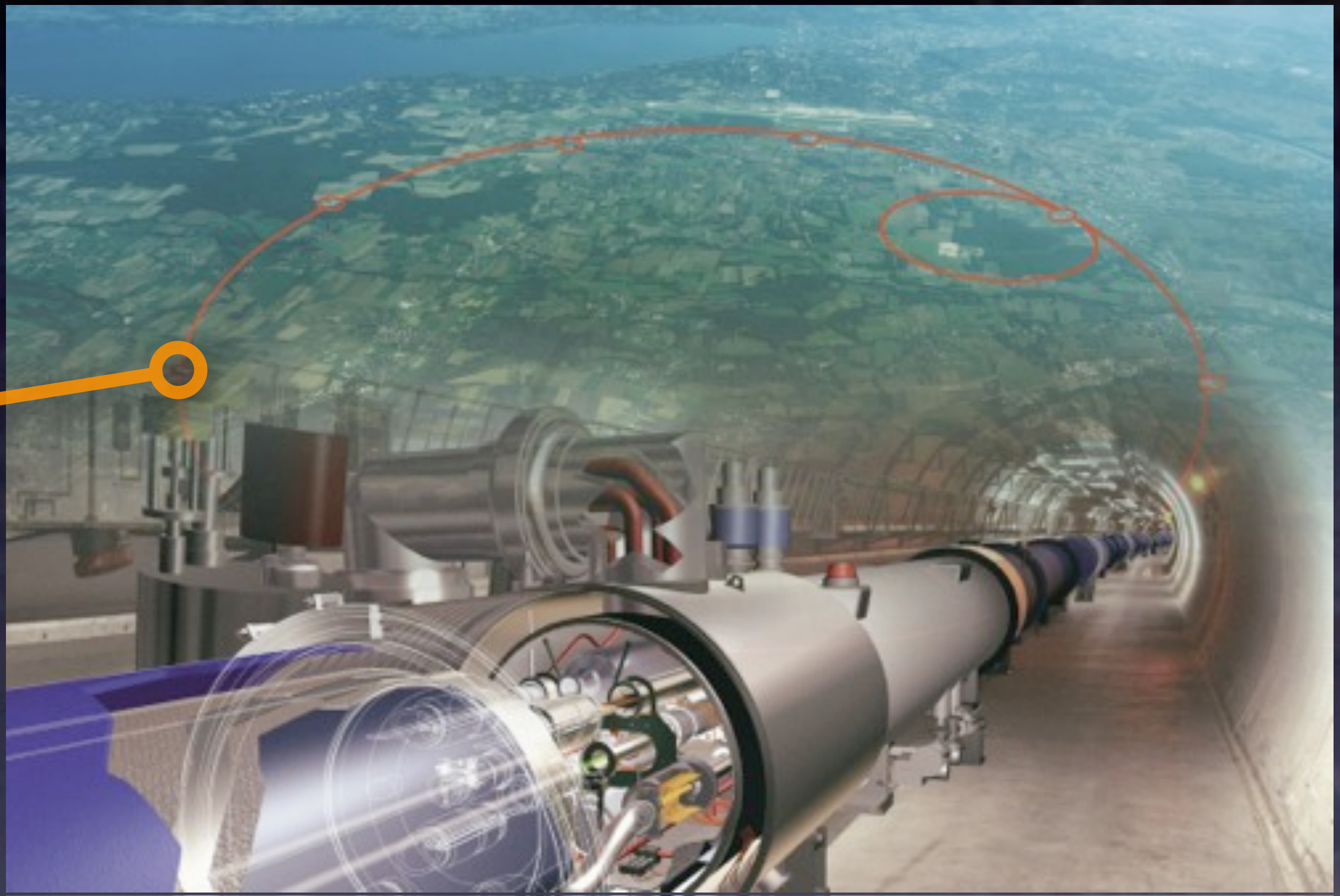
Outline

- Introduction
  - The CMS Experiment
  - The Data Bookkeeping System (DBS 2/3)
- Design of DBS 3
- Dependencies on DBS
- Transition DBS 2 to DBS 3
- Testing of DBS 3

The CMS Experiment

The **C**ompact **M**uon **S**olenoid is a general purpose detector at the Large Hadron Collider at CERN (LHC Point 5)



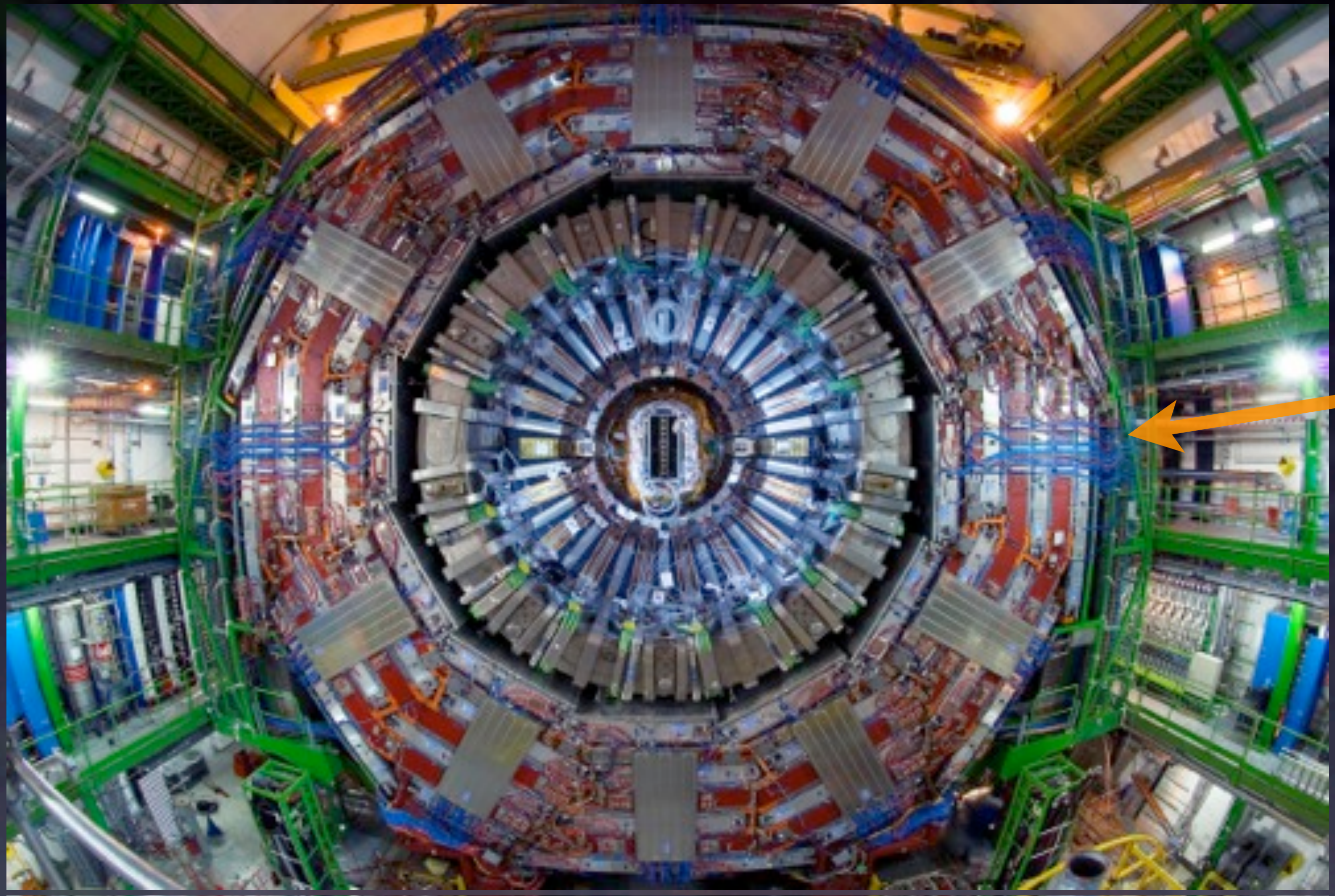


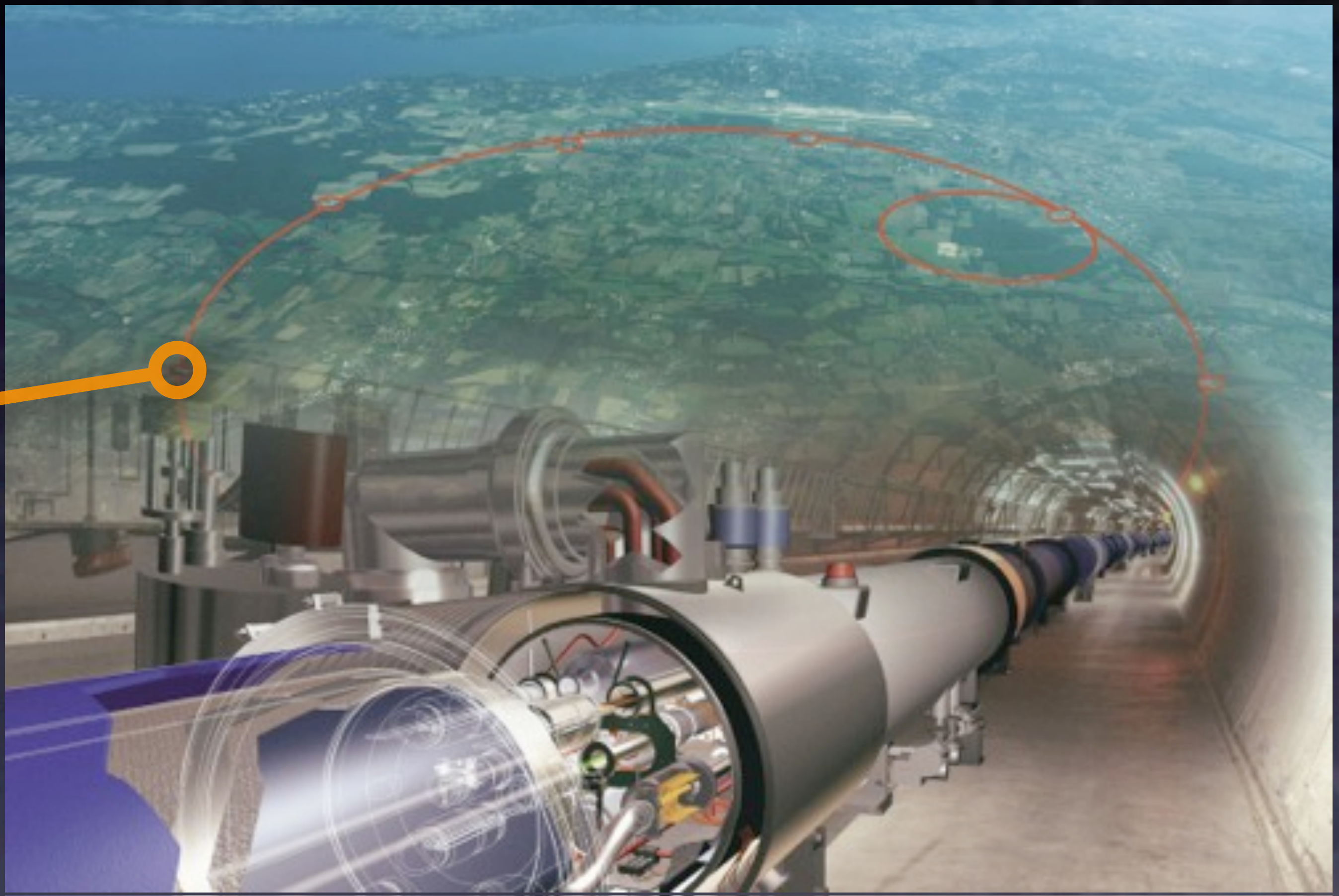
Huge Data Source:

- Around 200k official datasets
- About 40M official files
- 700 GB ofcl. metadata (DBS 2)

The CMS Experiment

The **C**ompact **M**uon **S**olenoid is a general purpose detector at the Large Hadron Collider at CERN (LHC Point 5)


CERN


CERN

Huge Data Source:

- Around 200k official datasets
- About 40M official files
- 700 GB ofcl. metadata (DBS 2)

- ▸A lot of data to cope with
- ▸A lot of metadata as well
- ▸Need an event data catalog

What is DBS?

- Acronym for the Data Bookkeeping Service
- Event data catalog for the CMS-Experiment at the LHC
- Information used for tracking datasets, data-processing history, associations between runs, files and datasets
- All data-processing in CMS relies on the information stored in DBS
  - Monte Carlo production
  - Data processing of recorded data
  - Physics analysis done by the users

Essential part of the Data Management and Workload Management Systems at CMS

Why DBS 3?

- DBS 2 designed/implemented 2006-2007 prior operation of LHC
  - CMS did not have a standardized service architecture
  - Implementation in JAVA servlets in a Apache Tomcat container
  - XML RPC for client-server communication
  - Some very „thick“ APIs
  - ▸API versioning problems, scalability issues
- CMS computing model evolved since design time of DBS 2
  - Many requests to store additional data not entirely consistent with its original purpose
  - CMS Data Management and Workflow Management group developed a common service and deployment architecture for web services (CMSWEB, RESTful Web services, etc.)

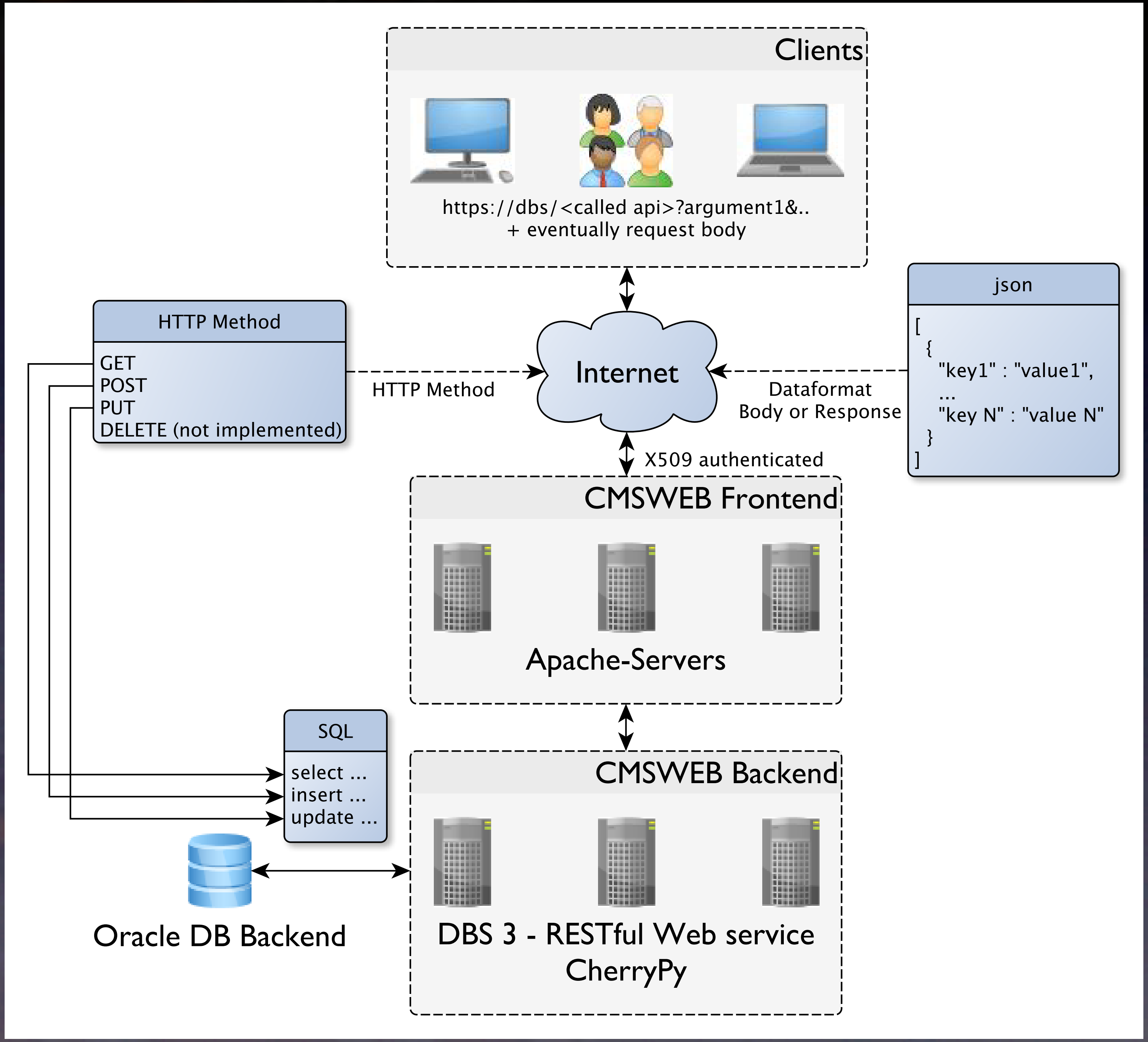
Led to the decision to re-design a new DBS

Design Goals

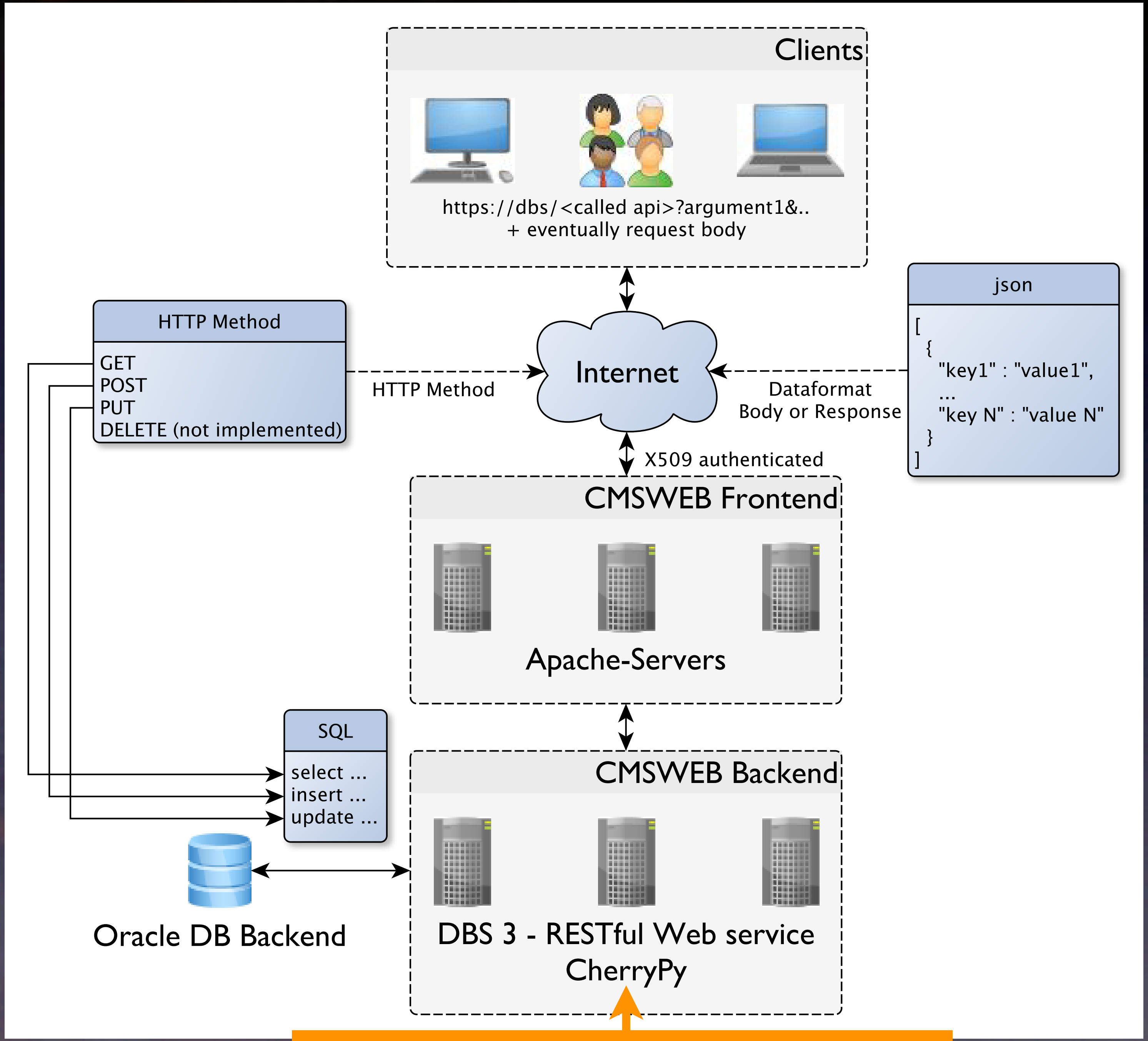
- Re-design with respect to the evolved processing model in CMS
- Spin-off any services outside the project scope (i.e. Web GUI)
  - Web GUI provided by Data Aggregation Service DAS
- Simplified and lightweight APIs
- Optimized DB schema
- Logical data-processing history (provenance, parentage)
  - No split and merge steps in processing history
    Since they have no impact on the data content

Implementation of prototypes using various architectures and technology led to the current DBS 3 design and implementations

Design of DBS 3

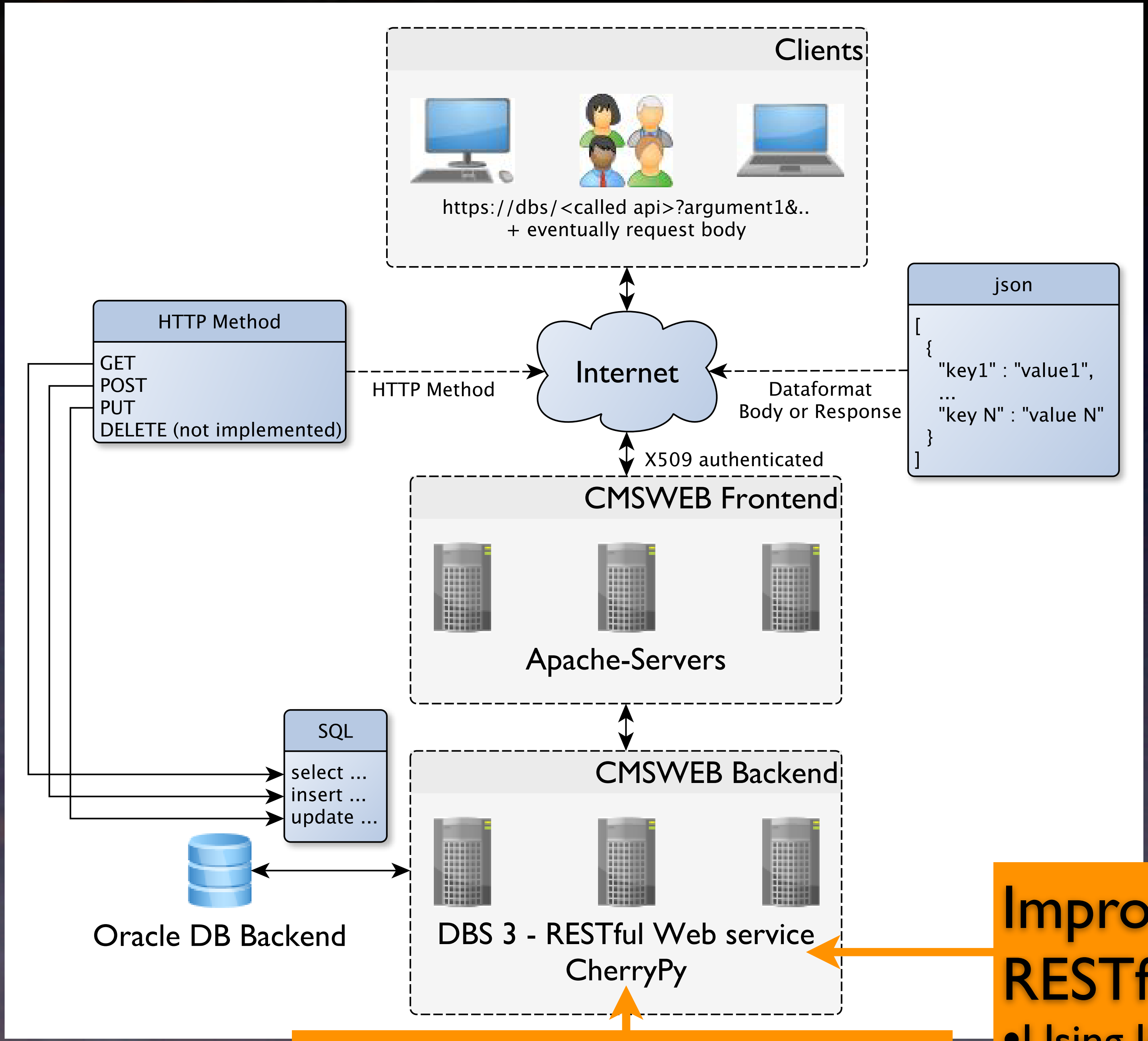


Design of DBS 3



Re-designed/implemented in Python (CherryPy)

Design of DBS 3



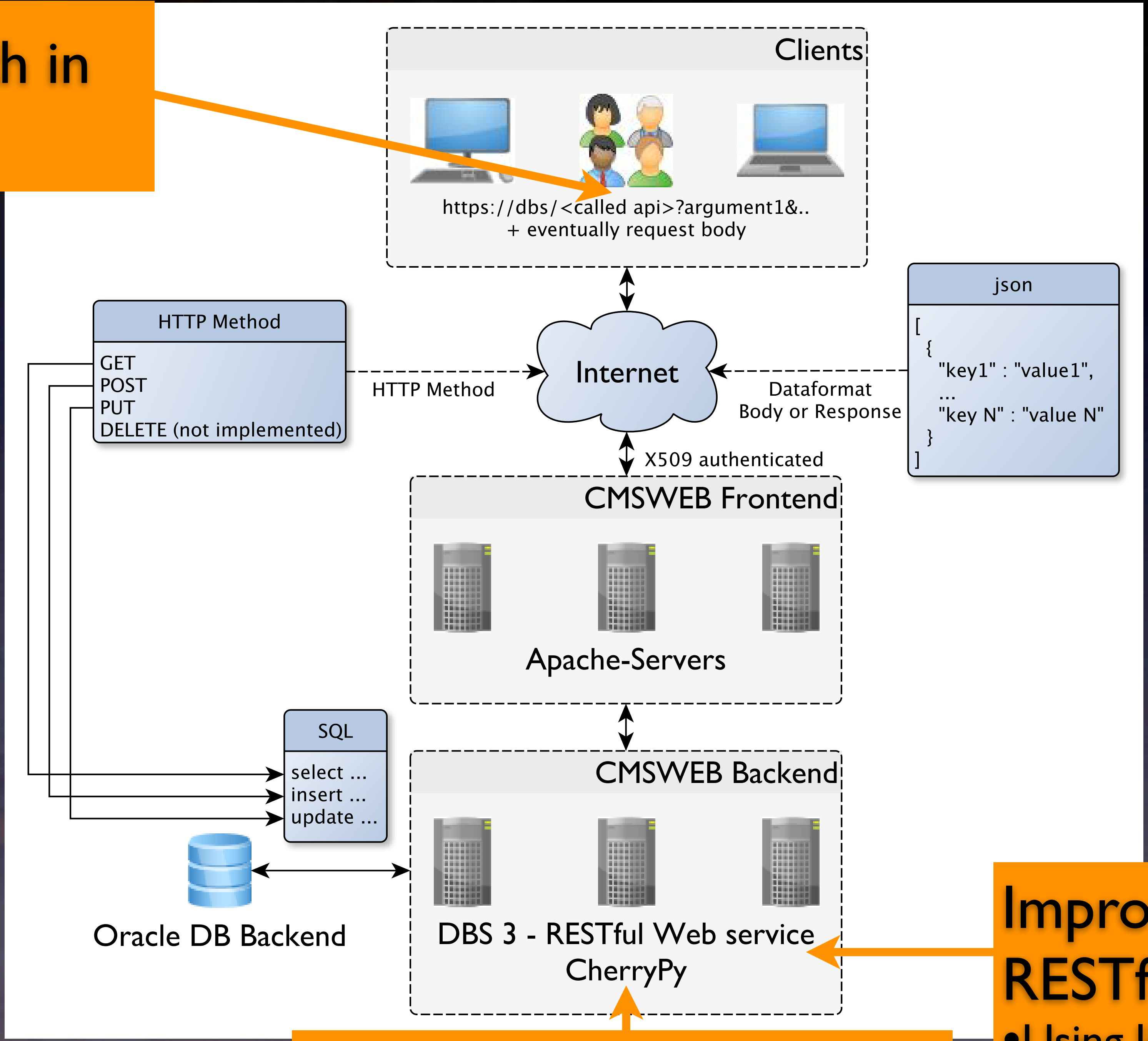
Improved scalability by RESTful design

- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Re-designed/implemented in Python (CherryPy)

Design of DBS 3

API chosen by path in URI



Re-designed/implemented in Python (CherryPy)

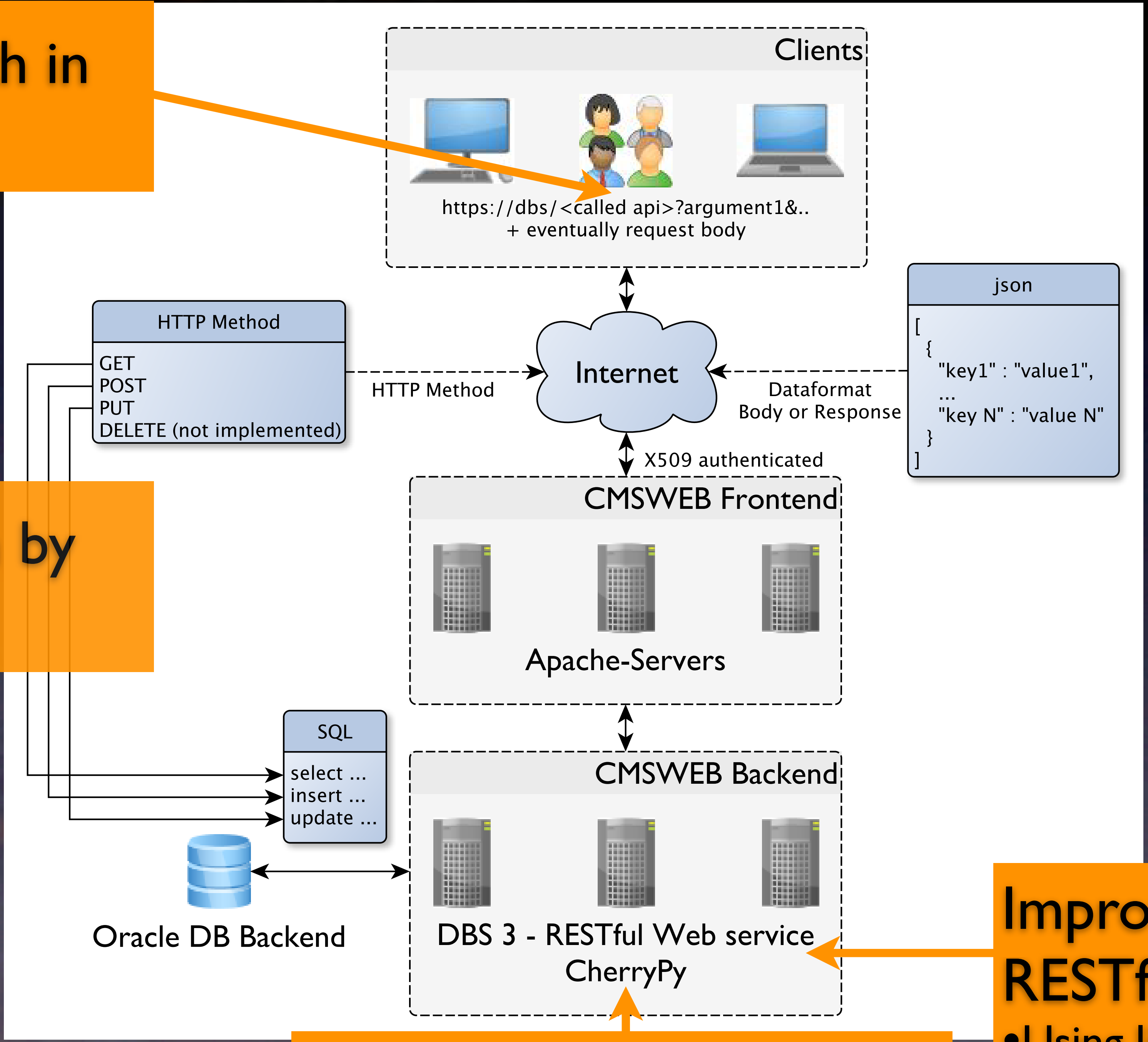
Improved scalability by RESTful design

- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Design of DBS 3

API chosen by path in URI

Operation chosen by HTTP method

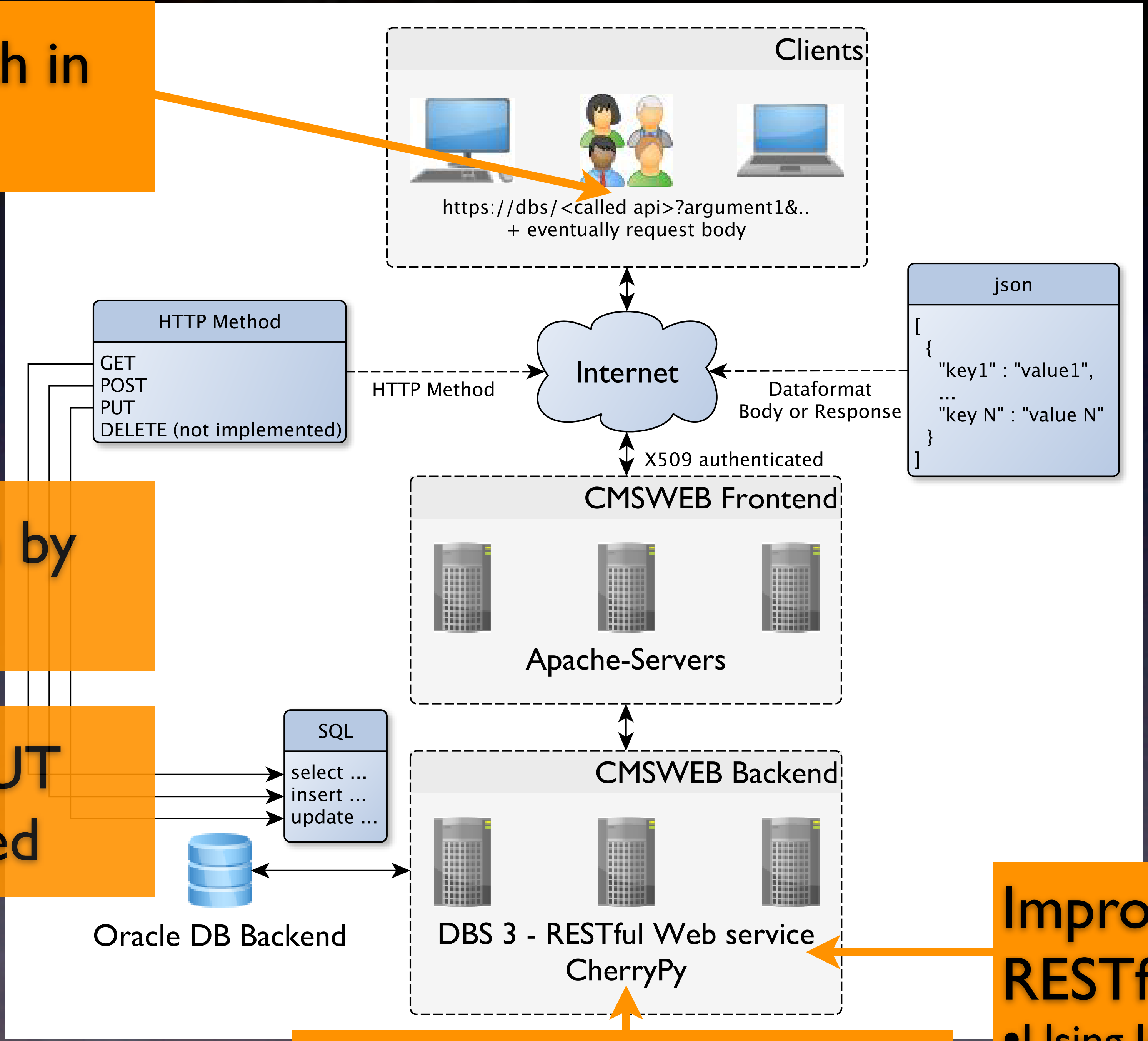


Re-designed/implemented in Python (CherryPy)

Improved scalability by RESTful design

- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Design of DBS 3



API chosen by path in URI

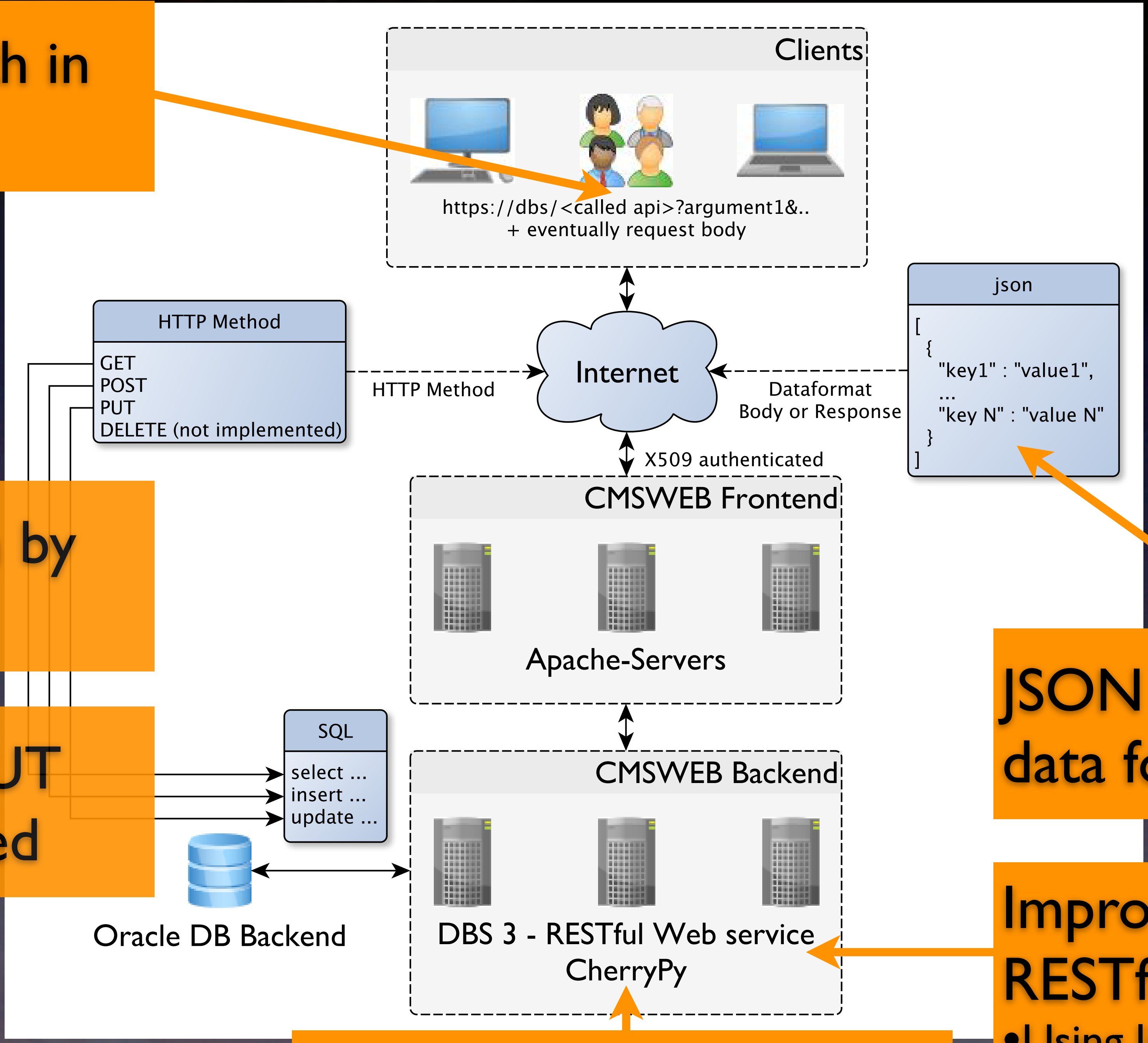
Operation chosen by HTTP method

GET, POST and PUT methods supported

Re-designed/implemented in Python (CherryPy)

Improved scalability by RESTful design
- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Design of DBS 3



API chosen by path in URI

Operation chosen by HTTP method

GET, POST and PUT methods supported

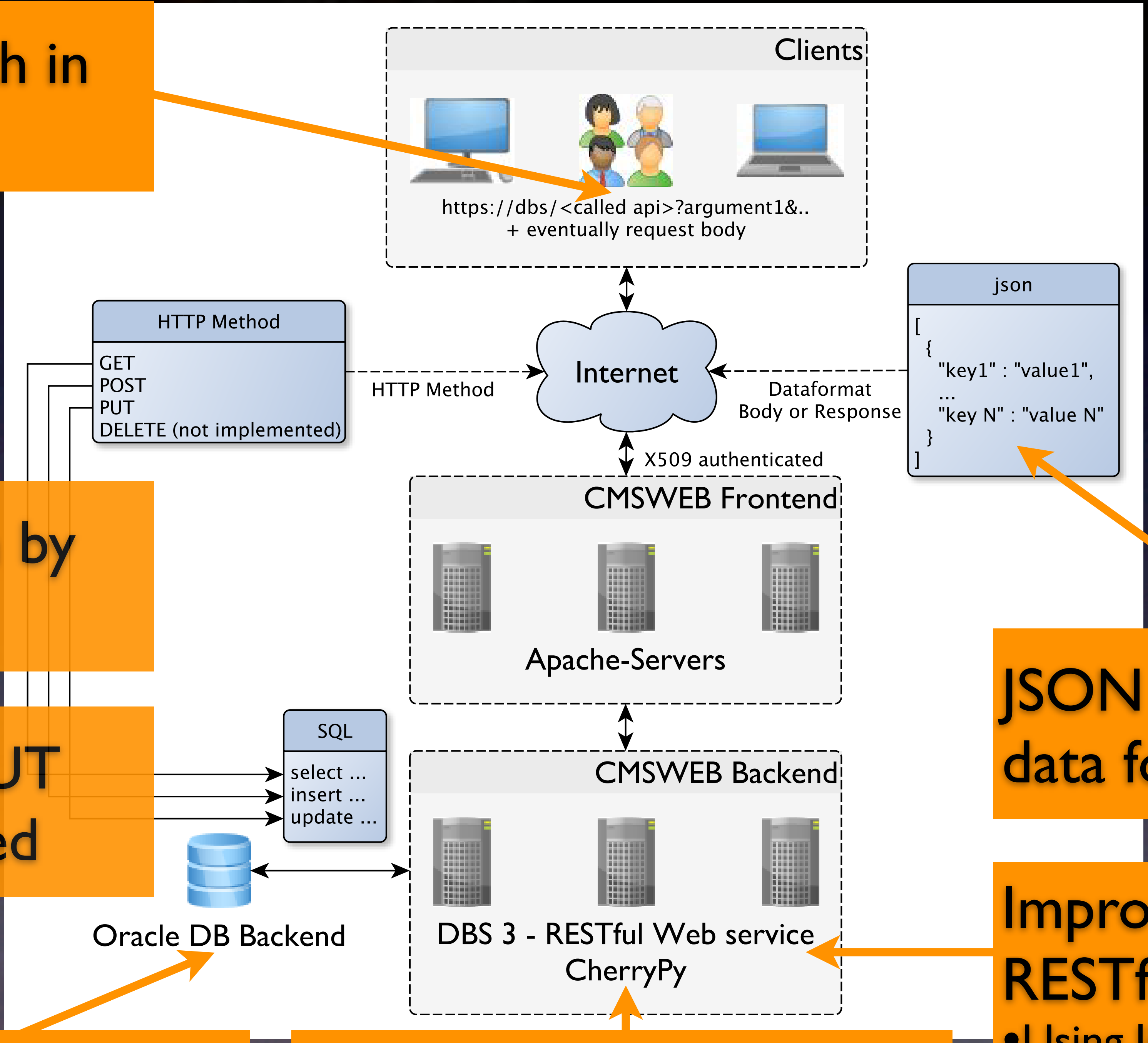
JSON input/output data format

Improved scalability by RESTful design

- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Re-designed/implemented in Python (CherryPy)

Design of DBS 3



API chosen by path in URI

Operation chosen by HTTP method

GET, POST and PUT methods supported

JSON input/output data format

Improved scalability by RESTful design
- Using lightweight APIs (Amdahl‘s law of scaling limits)
- Stateless client server communication

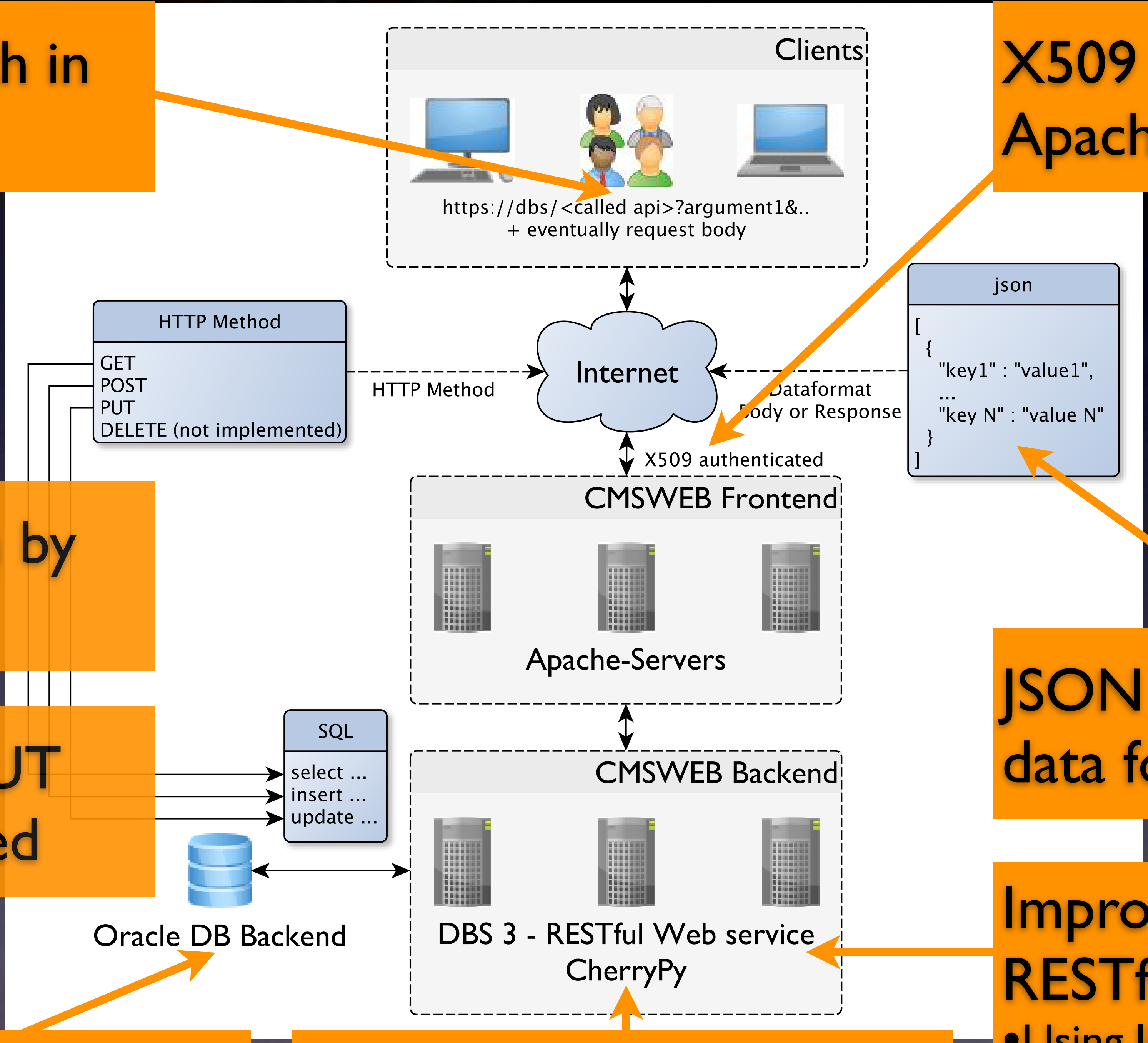
Oracle DB backend for persistent storage

Re-designed/implemented in Python (CherryPy)

Design of DBS 3

API chosen by path in URI

X509 authenticated by Apache HTTP frontend



Operation chosen by HTTP method

GET, POST and PUT methods supported

JSON input/output data format

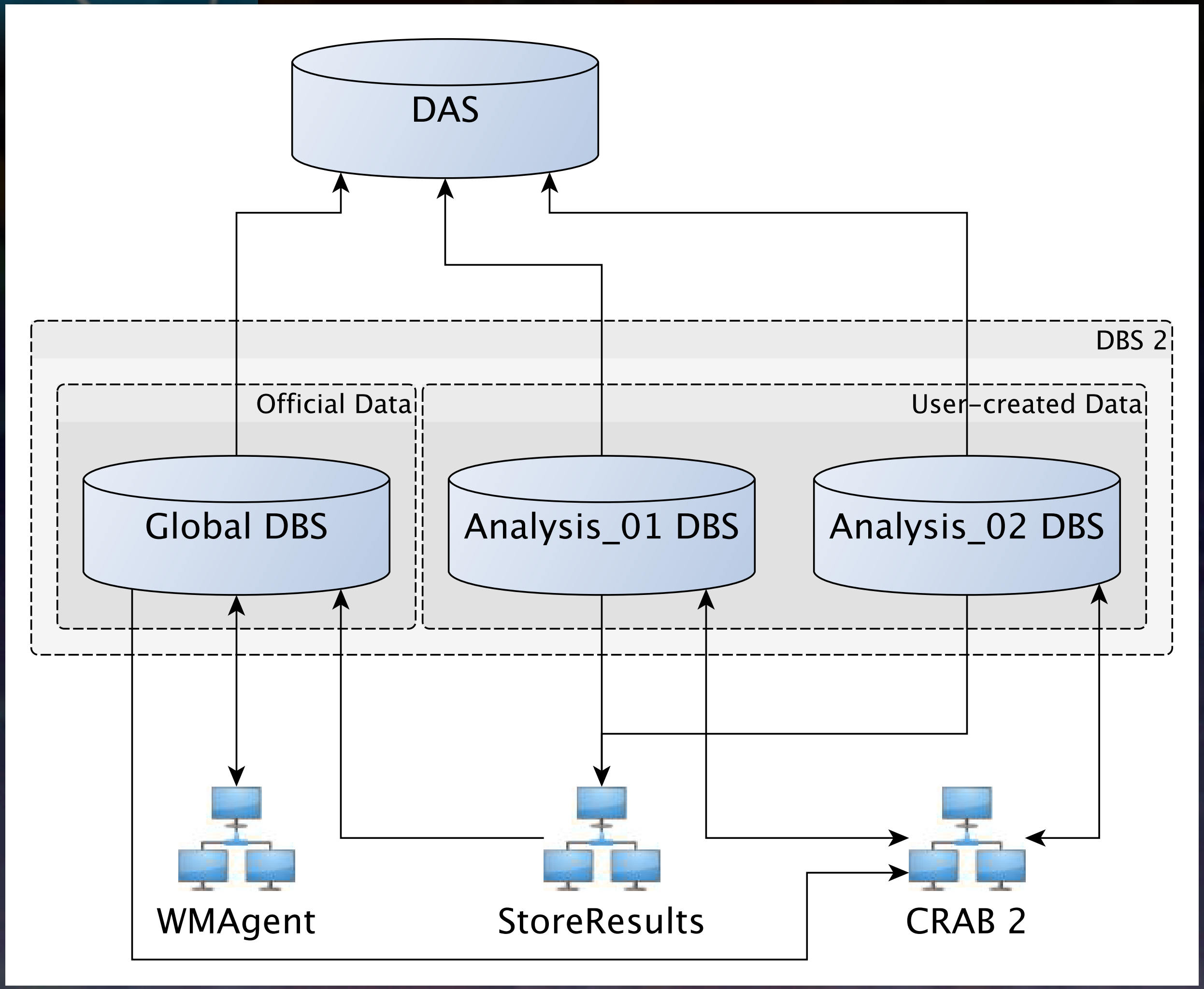
Improved scalability by RESTful design

- Using lightweight APIs (Amdahl's law of scaling limits)
- Stateless client server communication

Oracle DB backend for persistent storage

Re-designed/implemented in Python (CherryPy)

Dependencies on DBS

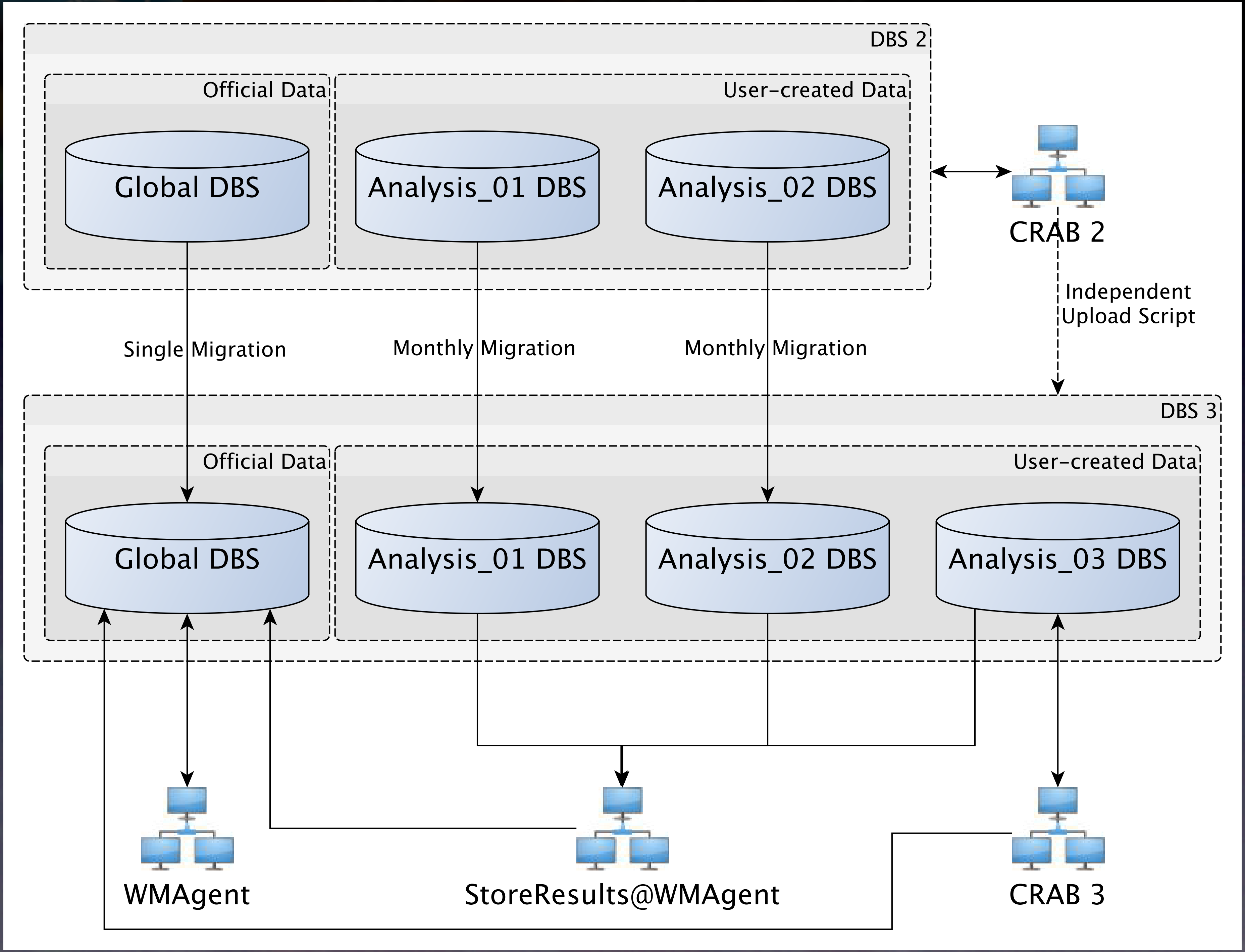


- Official and User-created data separated in CMS
- Splited in Global and Analysis DBS
- Data Aggregation Service (DAS) fetching and displaying information
- WMAgent distributed data processing tool in CMS (Official Data)
- StoreResults data-transfer of user-created data
- CRAB 2 distributed user analysis tool

- New versions for CRAB and StoreResults soon available
- Both old systems are not able to talk to DBS 3

▸ Transition to DBS 3 is a complex endeavor

Transition to DBS 3



- Global DBS will be migrated only once
- Global DBS 2 will be read-only
- Analysis DBS will be migrated periodically
- New Analysis DBS for CRAB 3 usage
- StoreResults will be replaced by SR@WM
- CRAB 2/3 used in parallel for a transition period
- WMAgents and CRAB 3 will use DBS 3 only

▸ Transition planed to happen end of the year

Migration to DBS 3

Challenges:

- DBS 2 is growing fast 40M files, 240k blocks and 200k datasets
- DBS 3 is completely different, data conversion is necessary

Migration & Validation:

- Adaption of data to the new schema is done using PL/SQL scripts
- Once data has been transformed, a one-to-one validation is done
- Validation is driven by a python script using standard SQL queries

Limitations:

- Duration of the migration fluctuates depending on DB load
- Migration happens en bloc - no incremental migration of data possible
  - All services need to be ready for DBS 3 before migration can happen

Migration Pilot Runs

Pilot Run Migration March/May 2013:

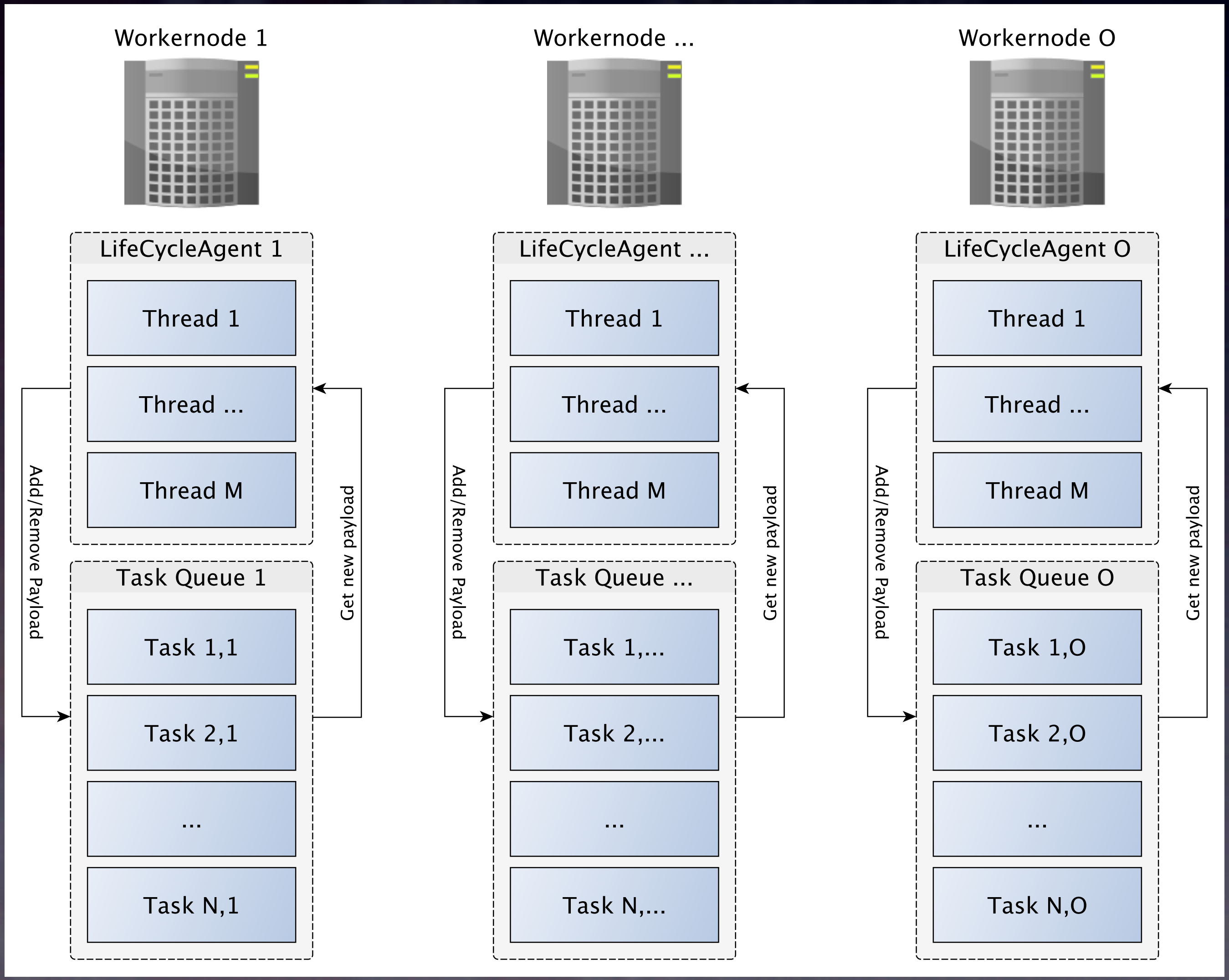
- DBS 2 was operated in read-only mode during the migration
- In March:
  - Data from Heavy Ion run was inserted in both DBS 2/3
- Since May:
  - All WMAgents are inserting all data in both DBS 2/3
  - Inserting around 18000 blocks per month
  - Monthly consistency checks between DBS 2/3 are done
  - Only minor differences found between DBS 2/3 data
    - Differences well understood

Conclusion:

- Migration and validation of data strongly depend on the DB load
- In both cases the migration finished in less than a day

DBS 3 Stress Tests

- Using PhEDEx LifeCycleAgent to drive the stress tests
- DBS 3 is using a dedicated queue on the CERN batch system
  (4 Hosts with 8 job slots each)



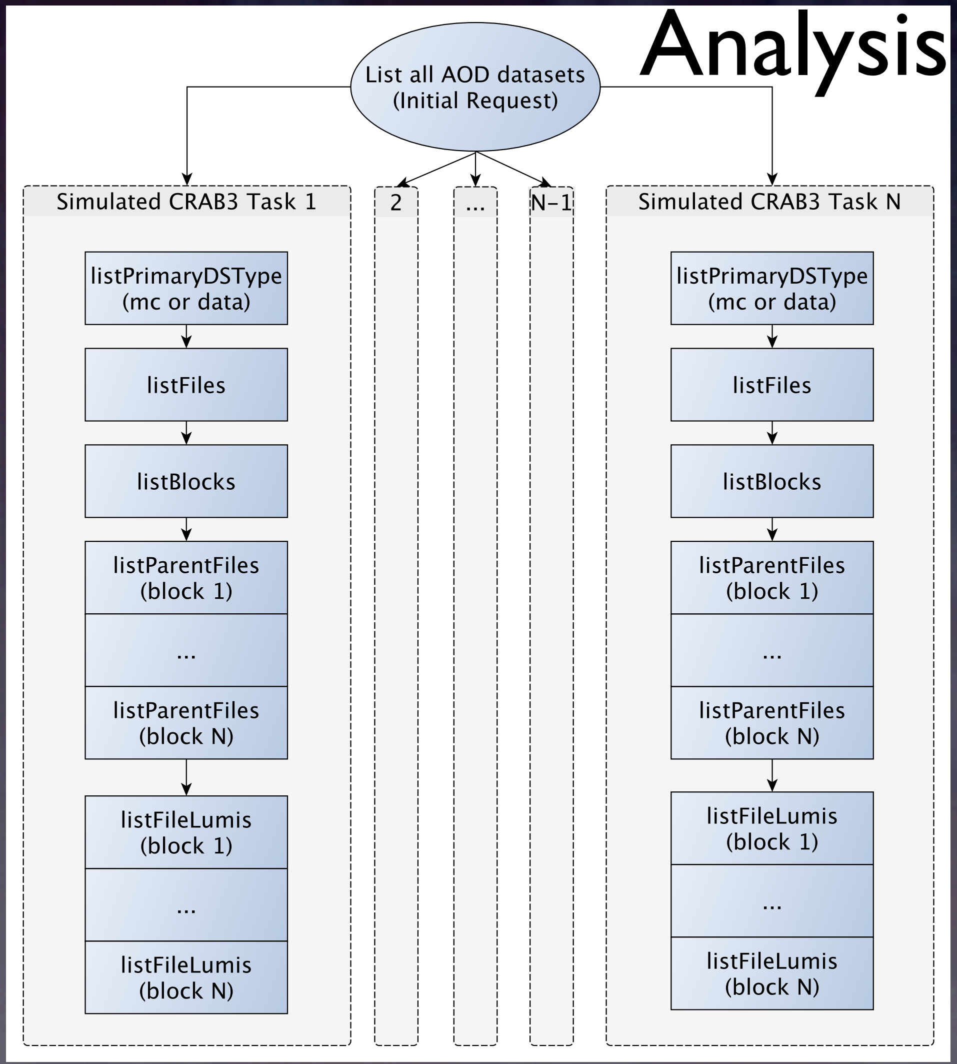
Poster at CHEP13: Integration and validation testing for PhEDEx, DBS and DAS with PhEDEx LifeCycleAgent

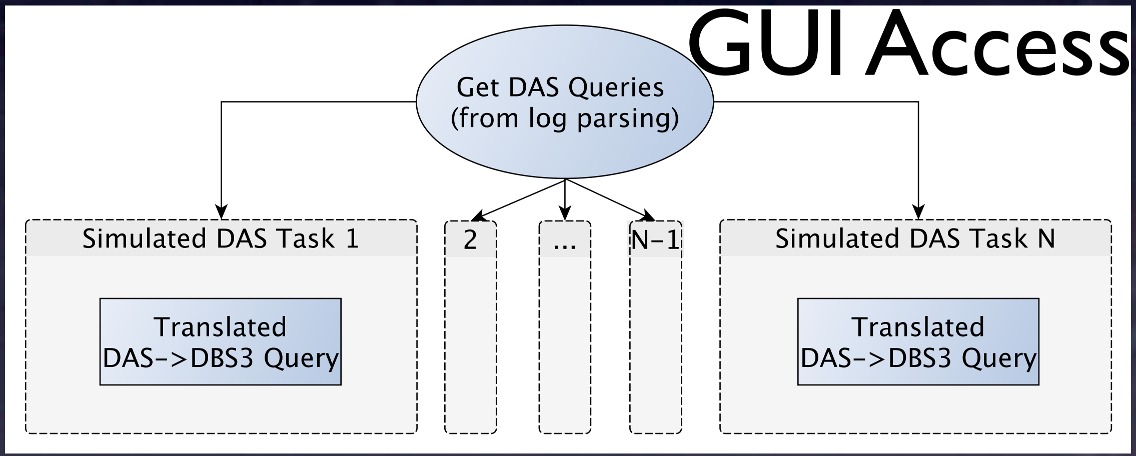
DBS 3 Stress Tests

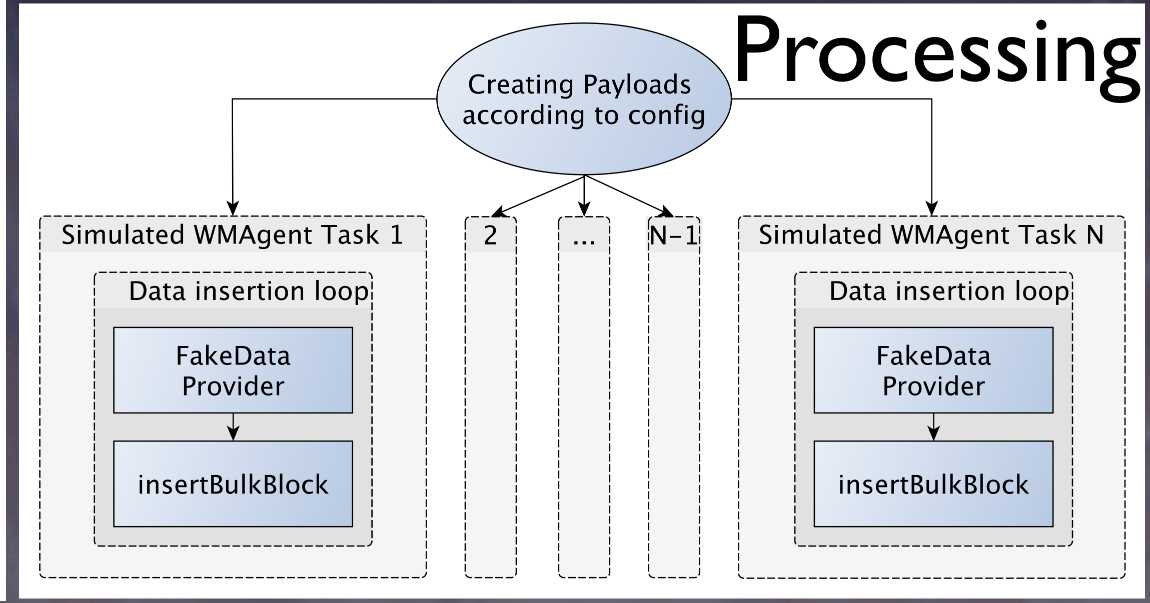
- Simulated CRAB 3 access pattern to DBS 3 (Distributed Analysis)
- Simulated DAS access pattern to DBS 3 (GUI Access from Users)
- WMAgent bulk block insertion simulation (Official Processing)

Run tests against cmsweb pre-production cluster

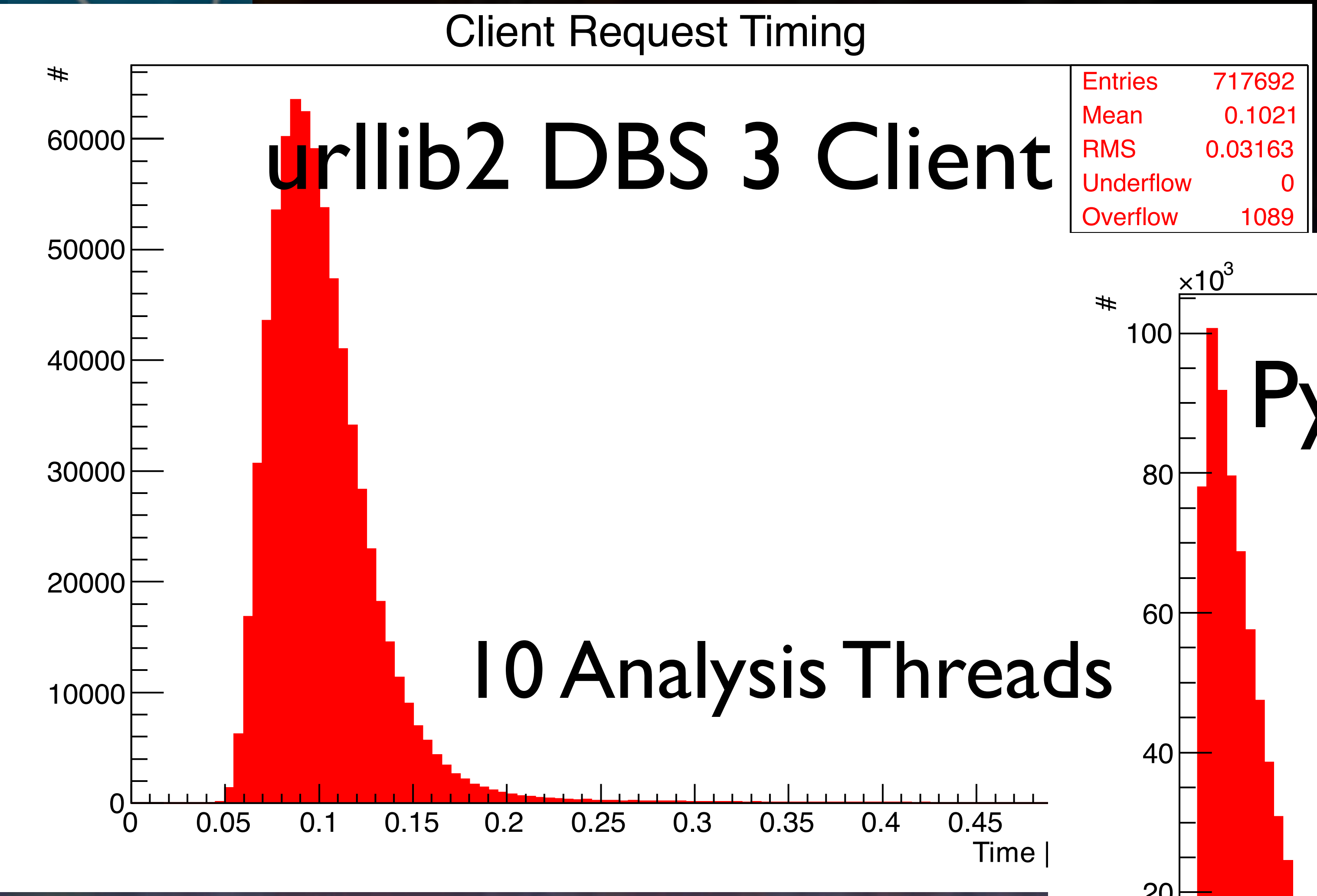
(2 Apache frontends, 2 cherrypy backends each running on a dual core VM having 4GB RAM)

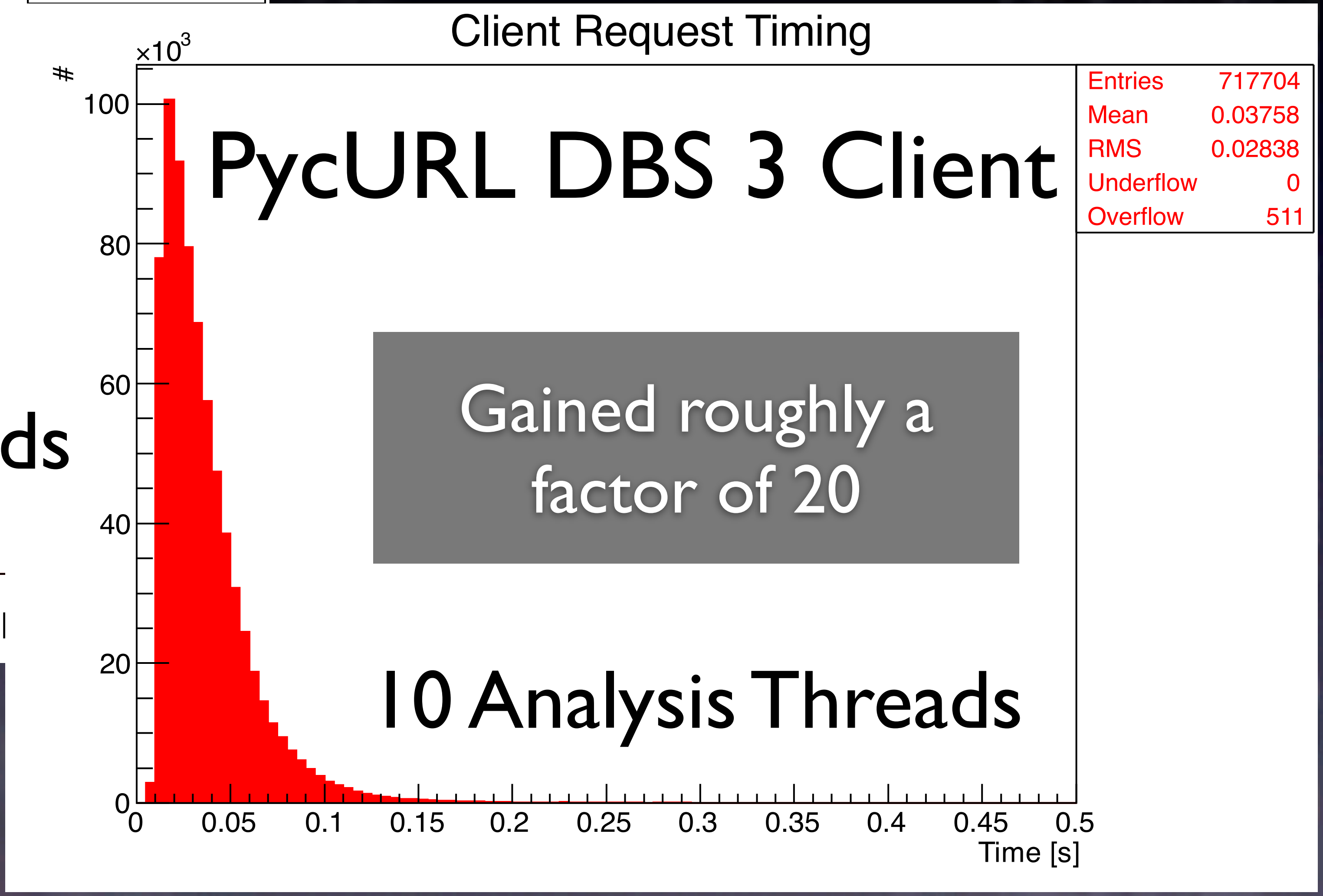



GUI Access
Get DAS Queries (from log parsing)
Simulated DAS Task 1
2
...
N-1
Simulated DAS Task N
Translated DAS->DBS3 Query



Results(Analysis Task)

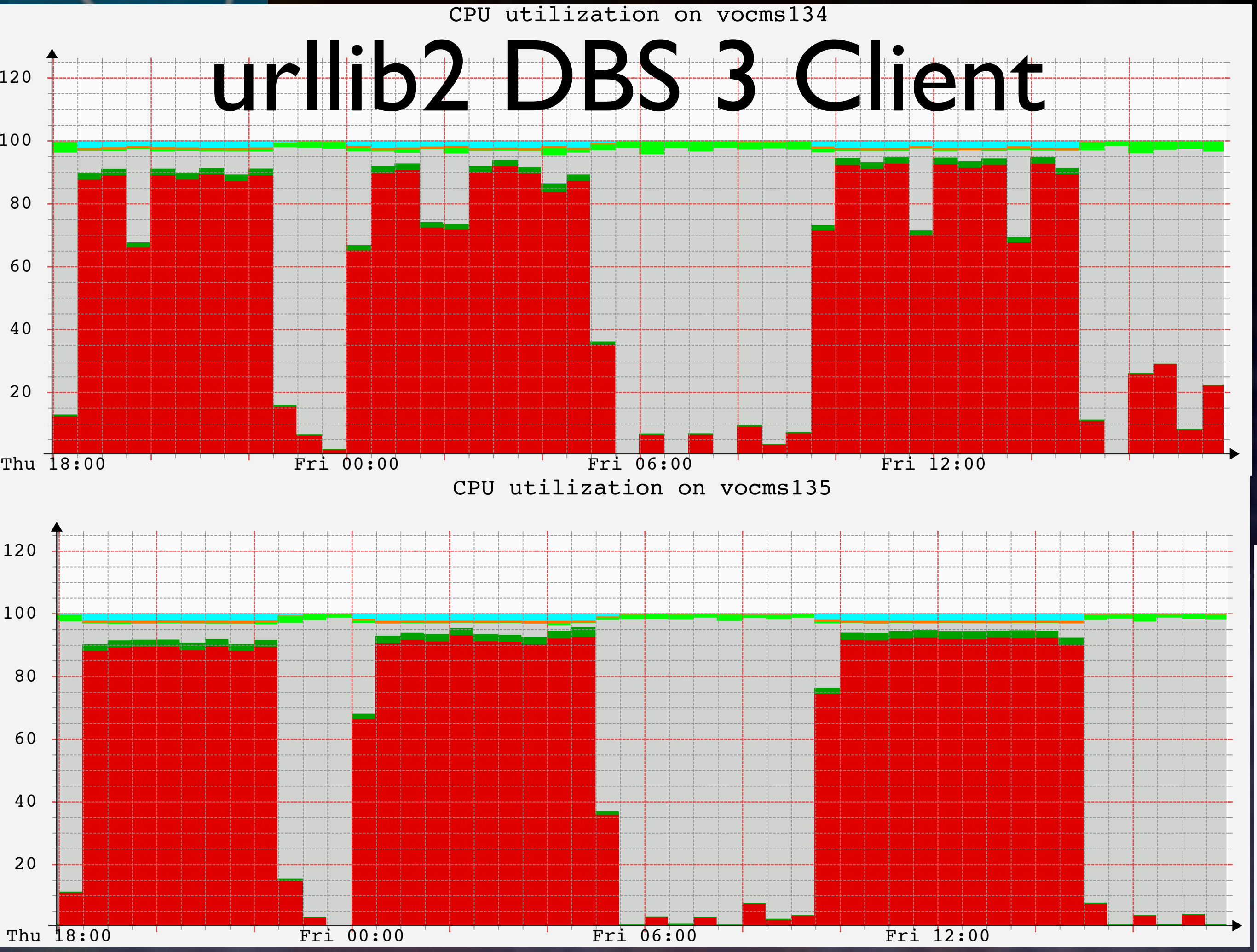




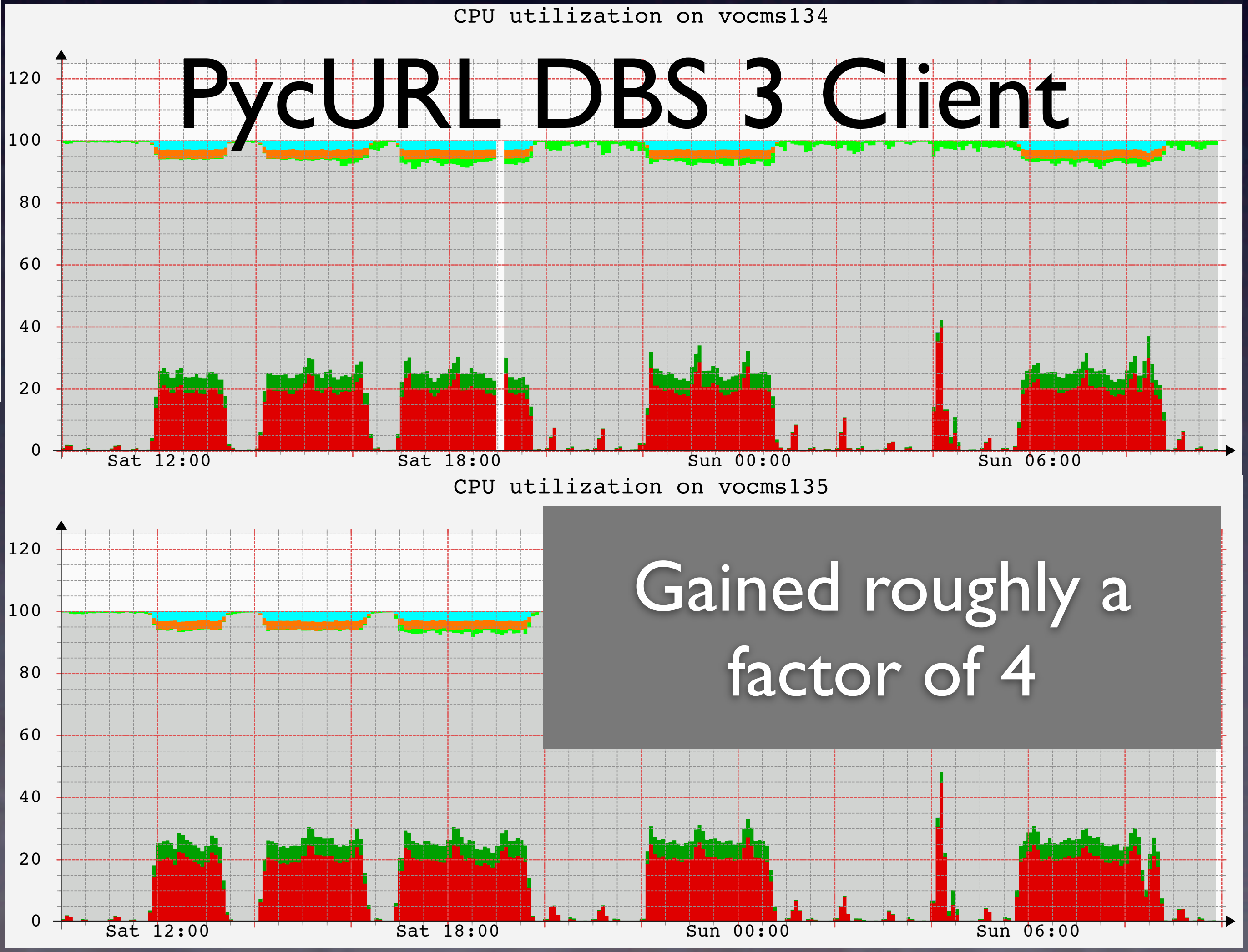
Gained roughly a factor of 20

Each client calls several DBS 3 APIs in a row. We can gain a lot by using PycURL ssl authentication caching

Results(Analysis Task)



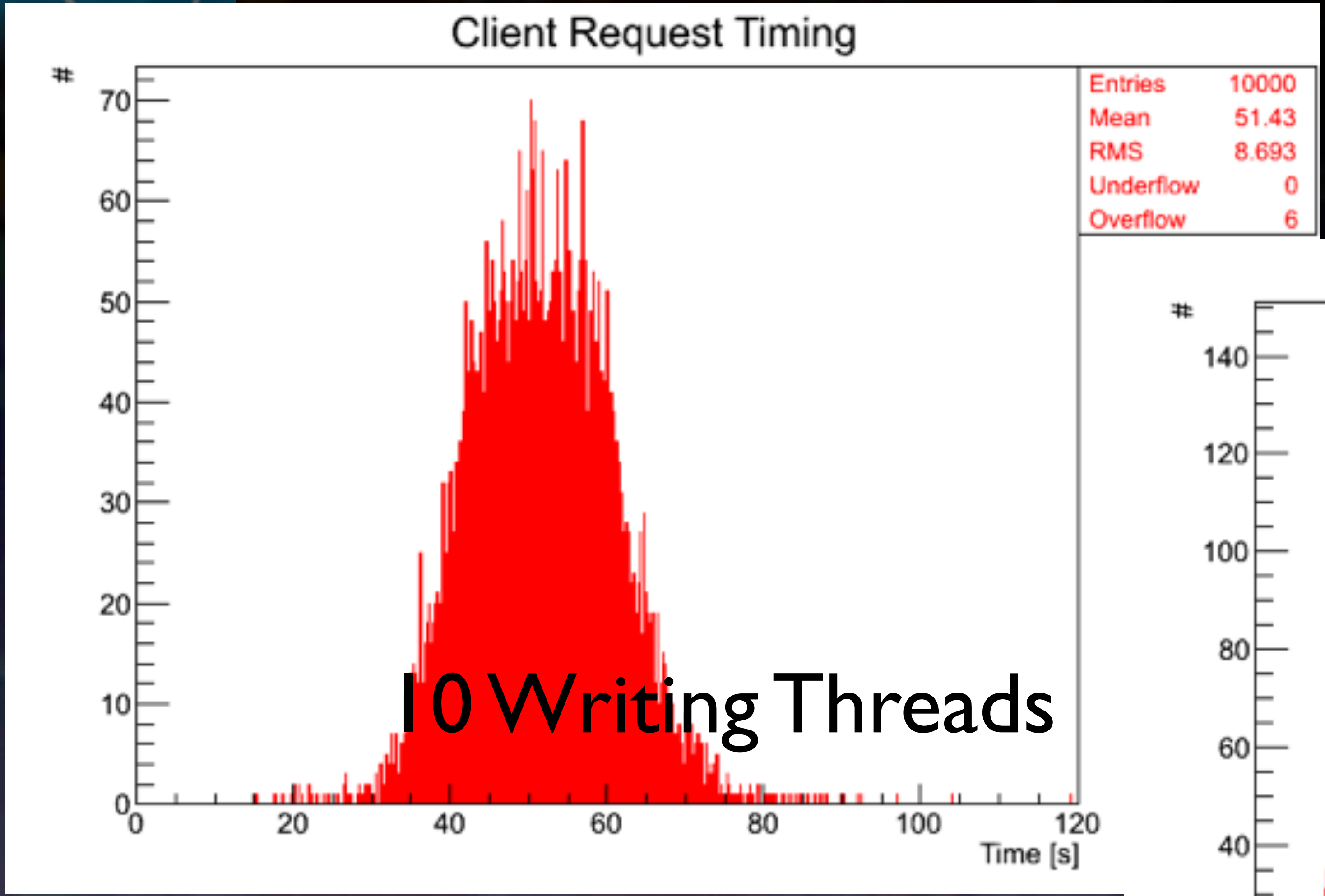
Using PycURL ssl authentication caching, reduces the number of authentications to the Apache Frontends of CMSWEB.



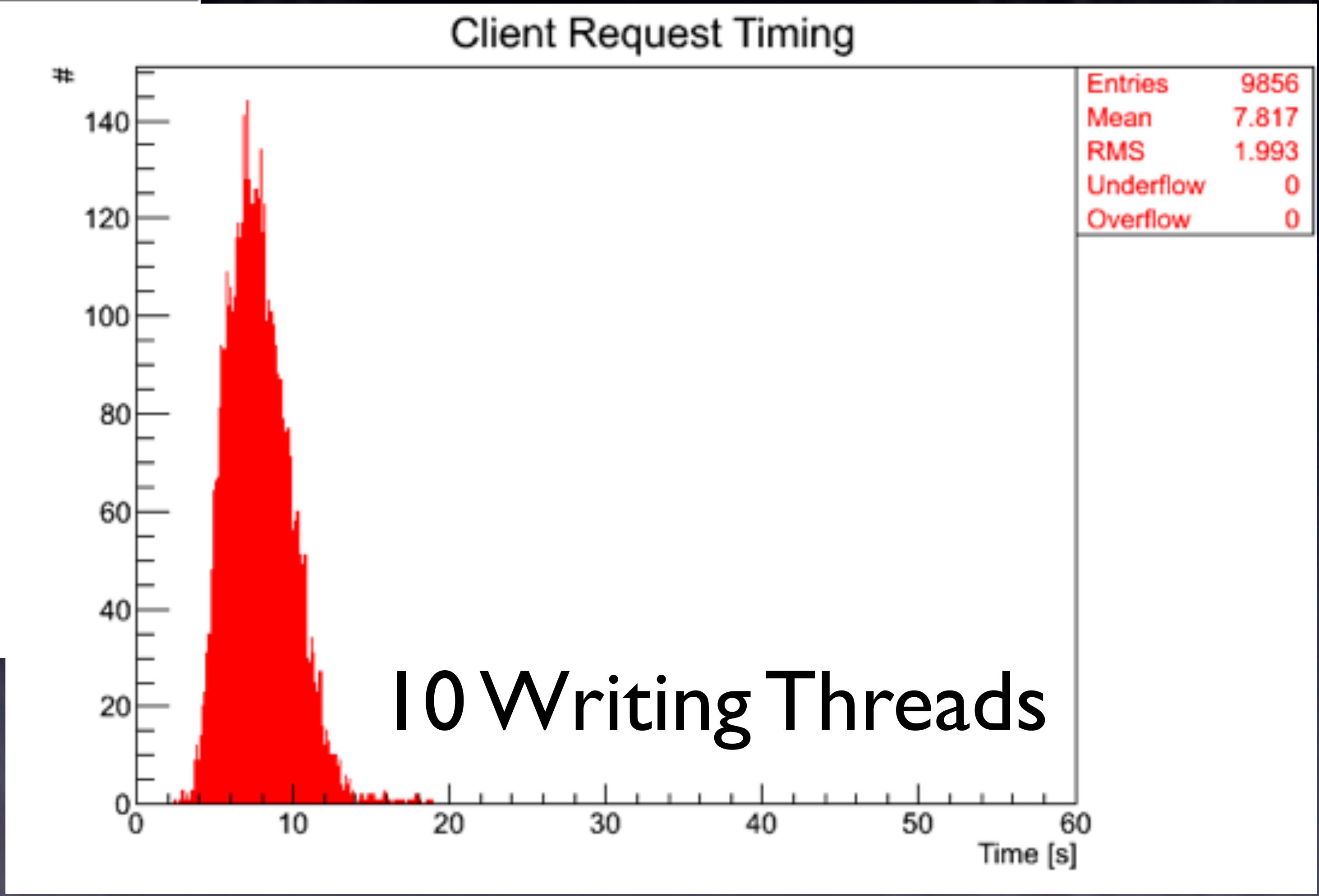
That means less load on the Apache Frontends of CMSWEB.

Gained roughly a factor of 4

Results(Writing)



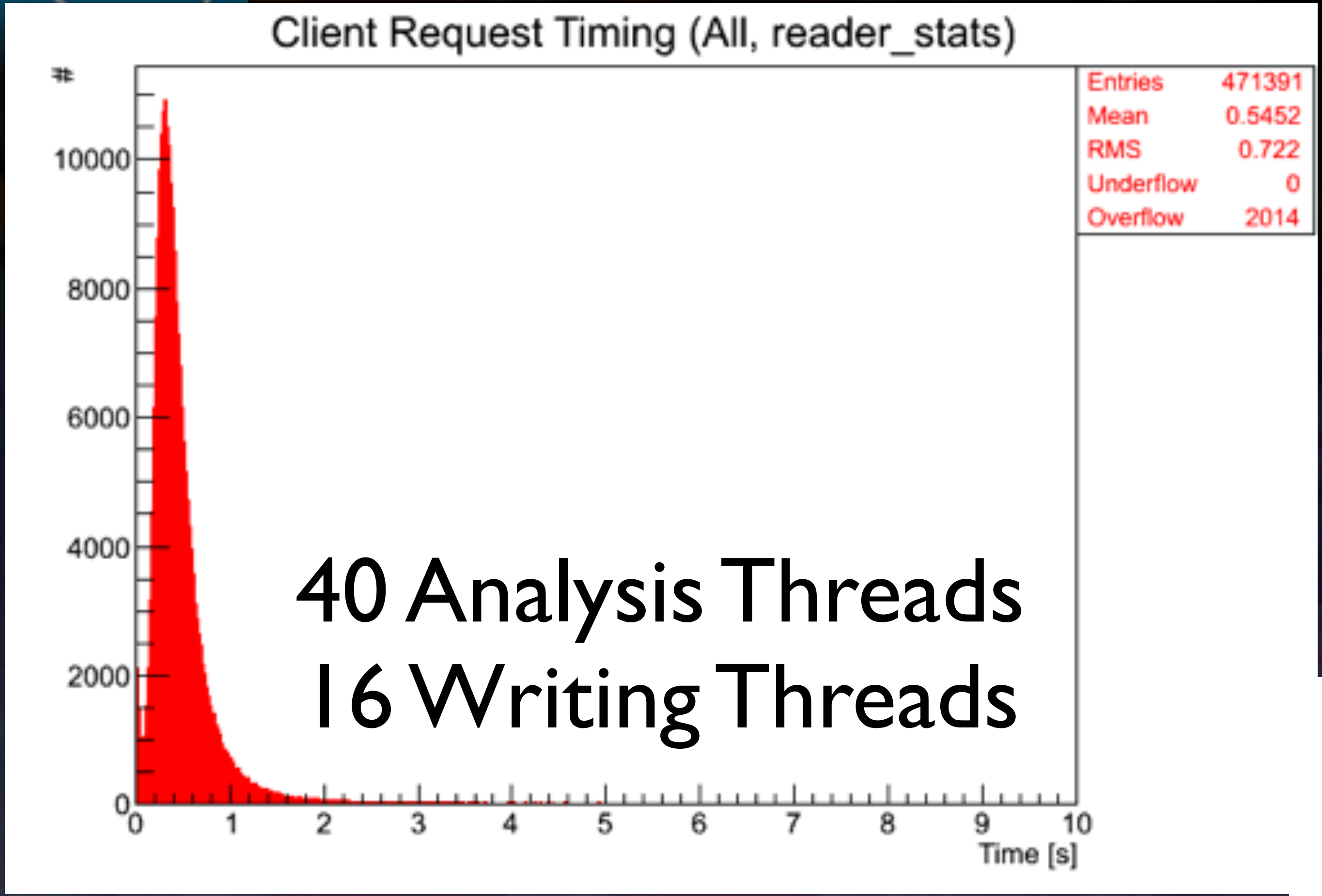
10000 Blocks, 100 Files each, 100 Lumisections each



10000 Blocks, 100 Files each, 14000 Lumisections each

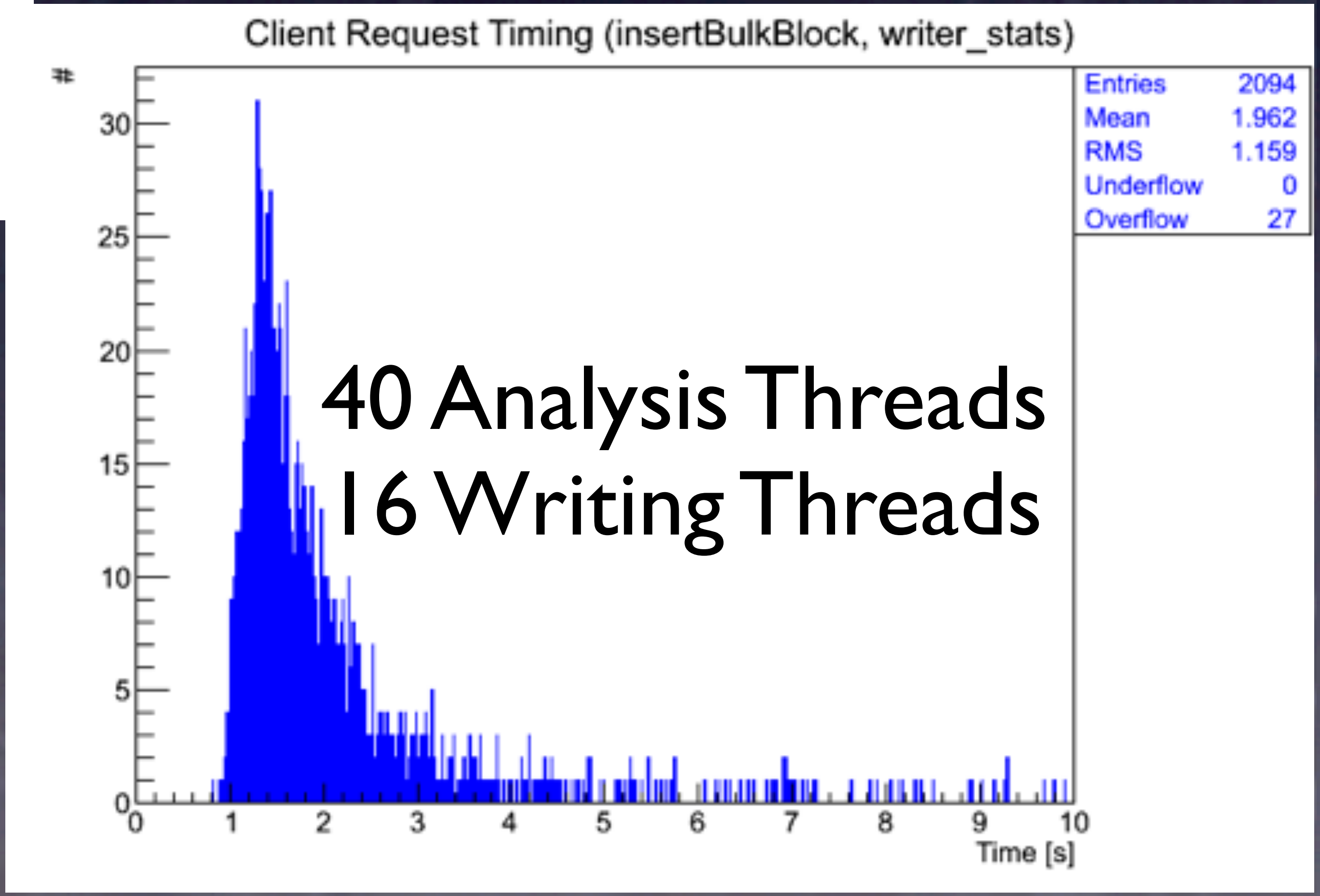
Even for the a large production task with 14000 Lumisections, the block injection in DBS 3 takes only 50s per block, compared to about 4 hours in DBS 2.

Results(Combined)



- Combined reading and writing test
- Even under high load DBS 3 behaves well

- The tests corresponds to 40 servers reading in parallel and 16 writing every 60s a block of 100 Files, having 100 Lumisections
- CMS currently has ~10 servers



Summary & Outlook

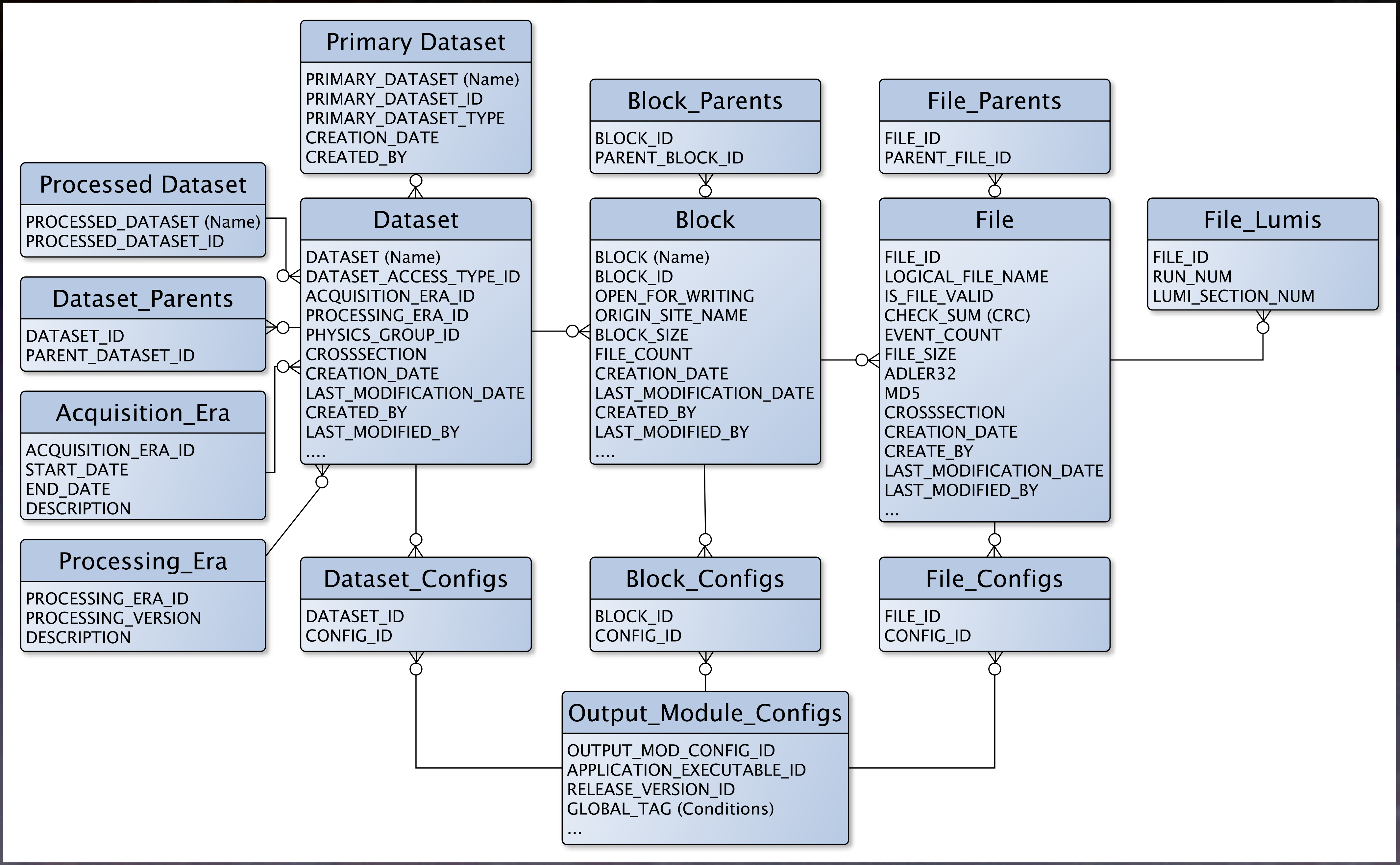
- DBS 3 is an essential part of the Data Management and Workload Management in CMS
- Without DBS MC production, data processing and user analysis are not possible
- The design of DBS is following the original design goals
- Transition from DBS 2/3 is a complex endeavour
- Migration pilot runs were successful, help to spot problems before going official to production
- Stress-tests showing that performance of DBS 3 is good

- Doing another migration pilot run before DBS 3 is going into production
- Make DAS querying both DBS 2/3 will move it a bit closer to production

Backup

Simplified DBS Schema

What information is stored in DBS?



Primary Dataset
PRIMARY_DATASET (Name)
PRIMARY_DATASET_ID
PRIMARY_DATASET_TYPE
CREATION_DATE
CREATED_BY
Block_Parents
BLOCK_ID
PARENT_BLOCK_ID
File_Parents
FILE_ID
PARENT_FILE_ID
Processed Dataset
PROCESSED_DATASET (Name)
PROCESSED_DATASET_ID
Dataset
DATASET (Name)
DATASET_ACCESS_TYPE_ID
ACQUISITION_ERA_ID
PROCESSING_ERA_ID
PHYSICS_GROUP_ID
CROSSSECTION
CREATION_DATE
LAST_MODIFICATION_DATE
CREATED_BY
LAST_MODIFIED_BY
....
Block
BLOCK (Name)
BLOCK_ID
OPEN_FOR_WRITING
ORIGIN_SITE_NAME
BLOCK_SIZE
FILE_COUNT
CREATION_DATE
LAST_MODIFICATION_DATE
CREATED_BY
LAST_MODIFIED_BY
....
File
FILE_ID
LOGICAL_FILE_NAME
IS_FILE_VALID
CHECK_SUM (CRC)
EVENT_COUNT
FILE_SIZE
ADLER32
MD5
CROSSSECTION
CREATION_DATE
CREATE_BY
LAST_MODIFICATION_DATE
LAST_MODIFIED_BY
...
File_Lumis
FILE_ID
RUN_NUM
LUMI_SECTION_NUM
Dataset_Parents
DATASET_ID
PARENT_DATASET_ID
Acquisition_Era
ACQUISITION_ERA_ID
START_DATE
END_DATE
DESCRIPTION
Processing_Era
PROCESSING_ERA_ID
PROCESSING_VERSION
DESCRIPTION
Dataset_Configs
DATASET_ID
CONFIG_ID
Block_Configs
BLOCK_ID
CONFIG_ID
File_Configs
FILE_ID
CONFIG_ID
Output_Module_Configs
OUTPUT_MOD_CONFIG_ID
APPLICATION_EXECUTABLE_ID
RELEASE_VERSION_ID
GLOBAL_TAG (Conditions)
...