Common Analysis Framework for ATLAS and CMS

Feasibility Study Report

Fernando H. Barreiro Megino
Mattia Cinquilli
Daniele Spiga
Daniel C. van der Ster
CERN IT-ES-VOS

The Feasibility Study

- The analysis frameworks of the ATLAS and CMS experiments have successfully coped with the analysis load during the first 2 years of data taking
- However a common infrastructure is a step in the direction of reducing development and maintenance effort

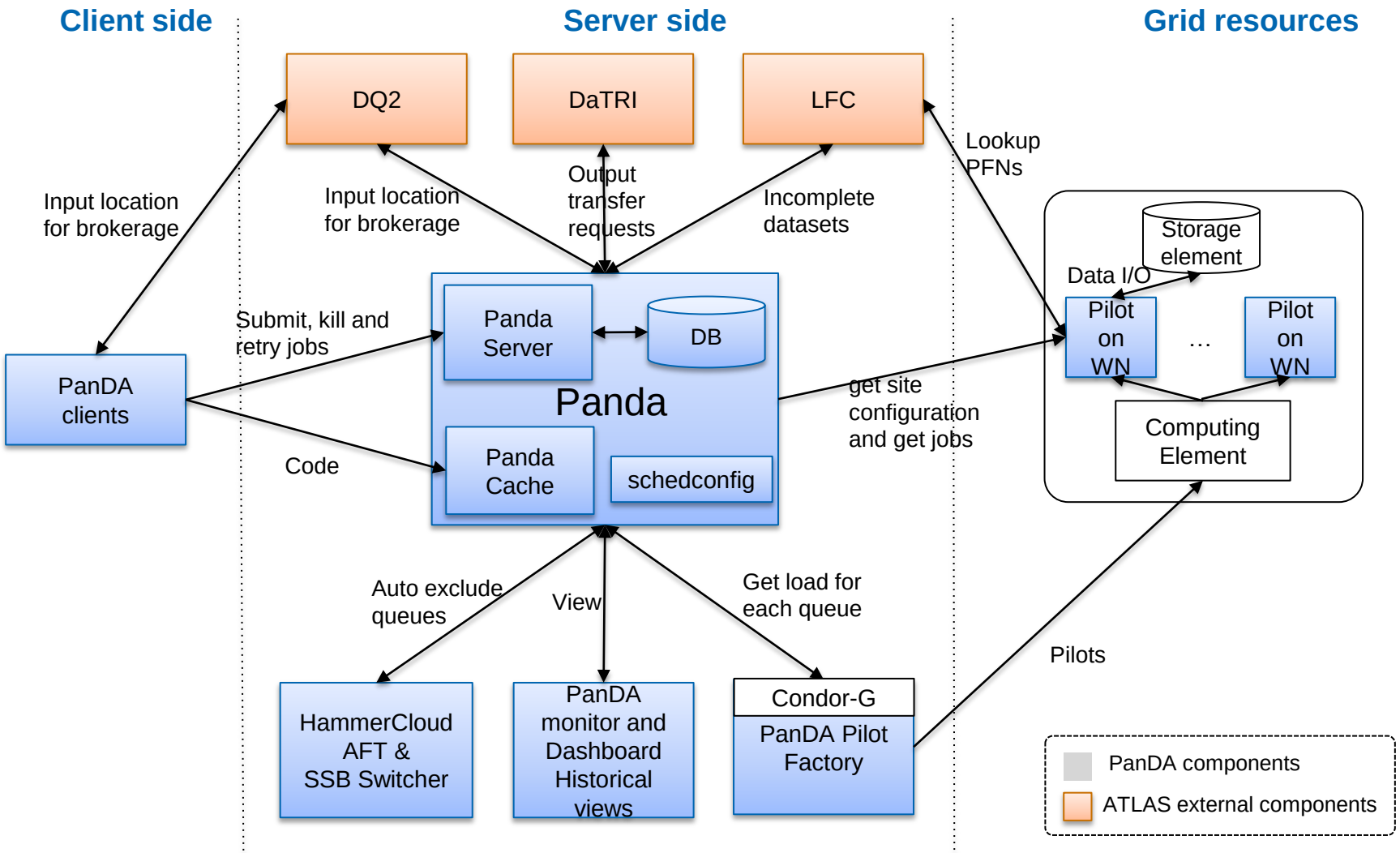
➔**Goal of the Feasibility Study:** assess the potential for using common components for distributed analysis, based on elements from **PanDA** and **glideInWMS**

- Review architecture and functionality of current analysis frameworks
- Identify interfaces to external systems
- Identify what can be reused
- Identify show-stoppers

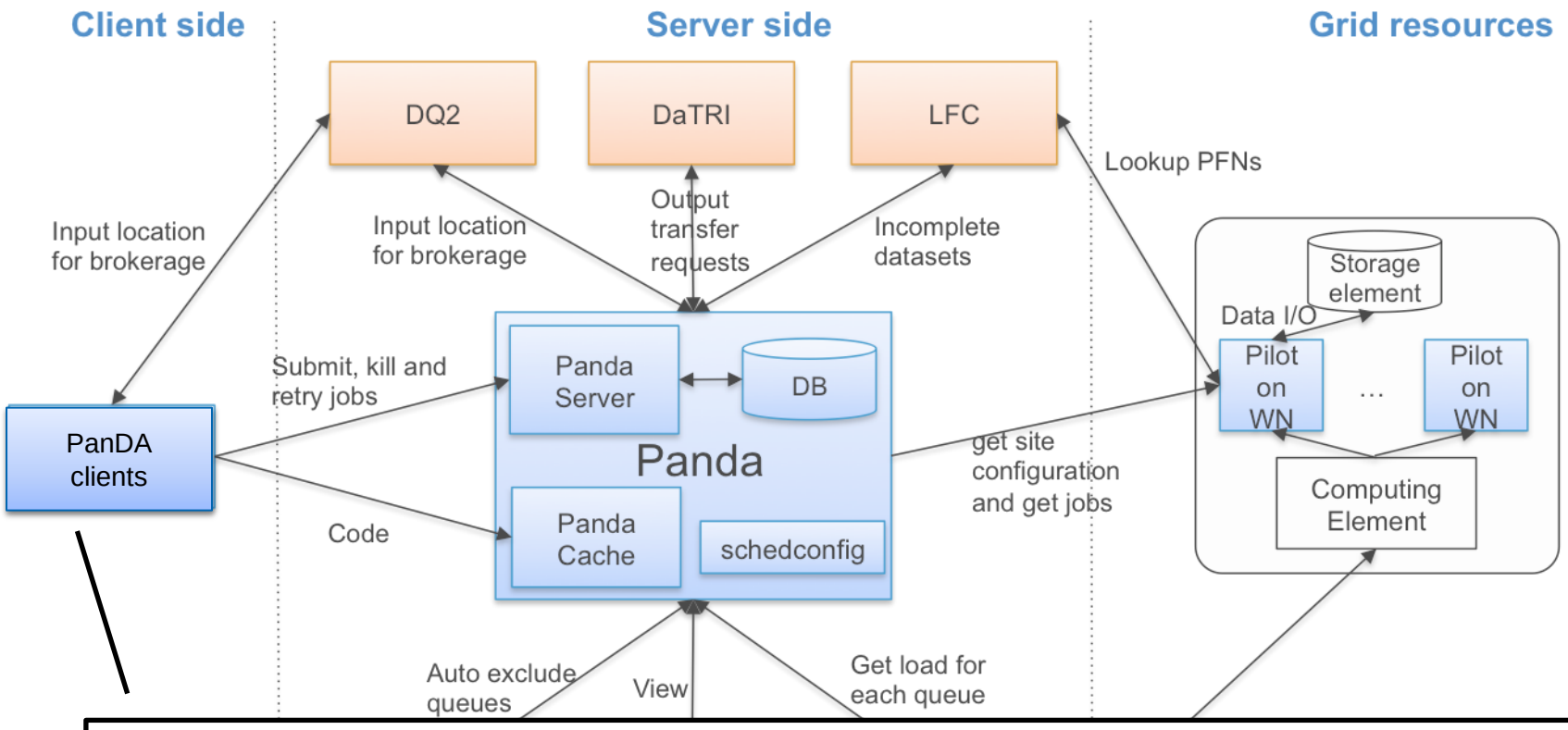
- The study was carried out by CERN IT-ES working group in collaboration with experts of different components and with continuous feedback from the experiments' computing management

Experiment analysis framework architectures

PanDA architecture

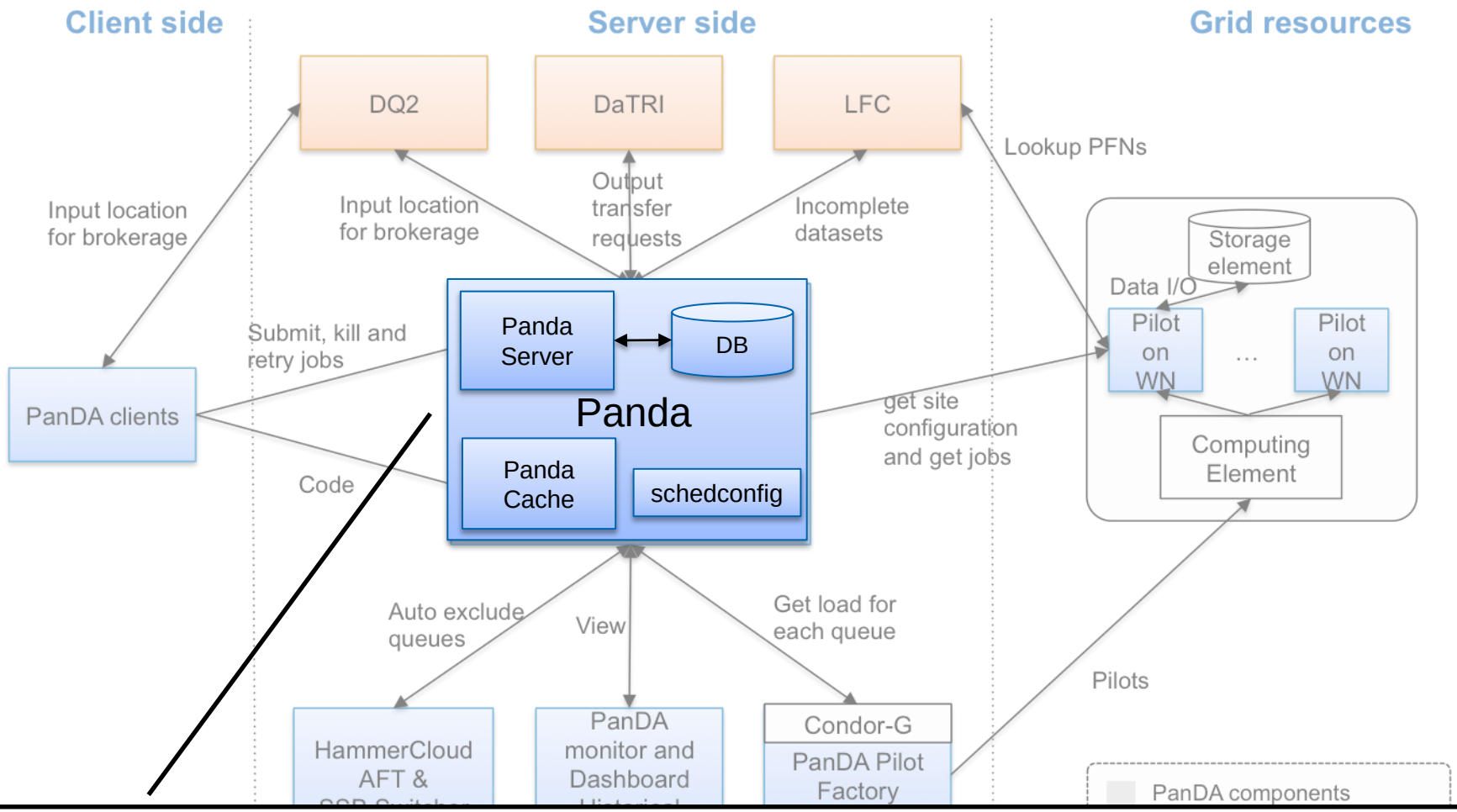


PanDA architecture



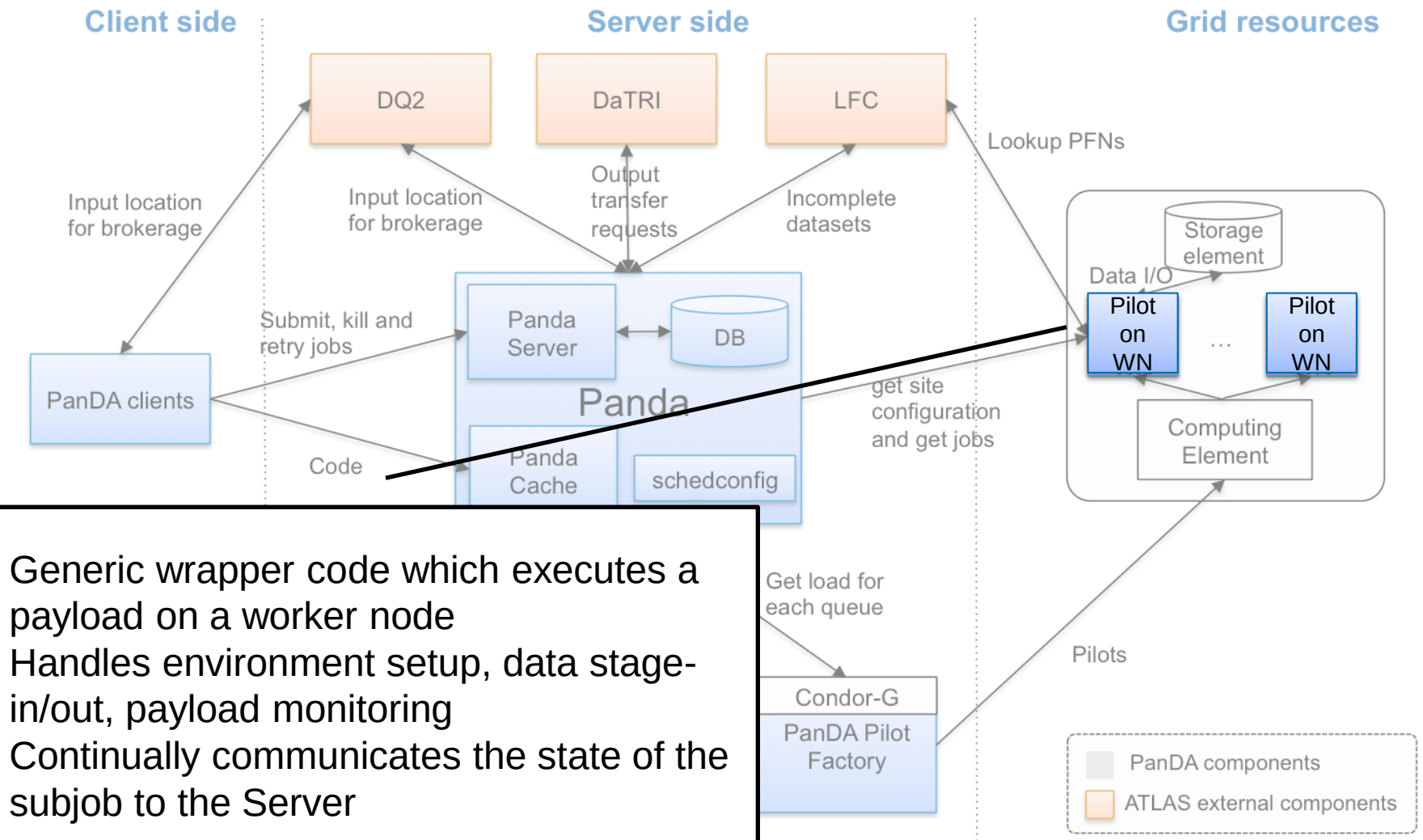
- CLI tools for physicists to submit, monitor and manage their analysis jobs
- Location lookup
- Job splitting into subjobs (i.e. individual work units)
- Inject jobs to the PanDA Server

PanDA architecture



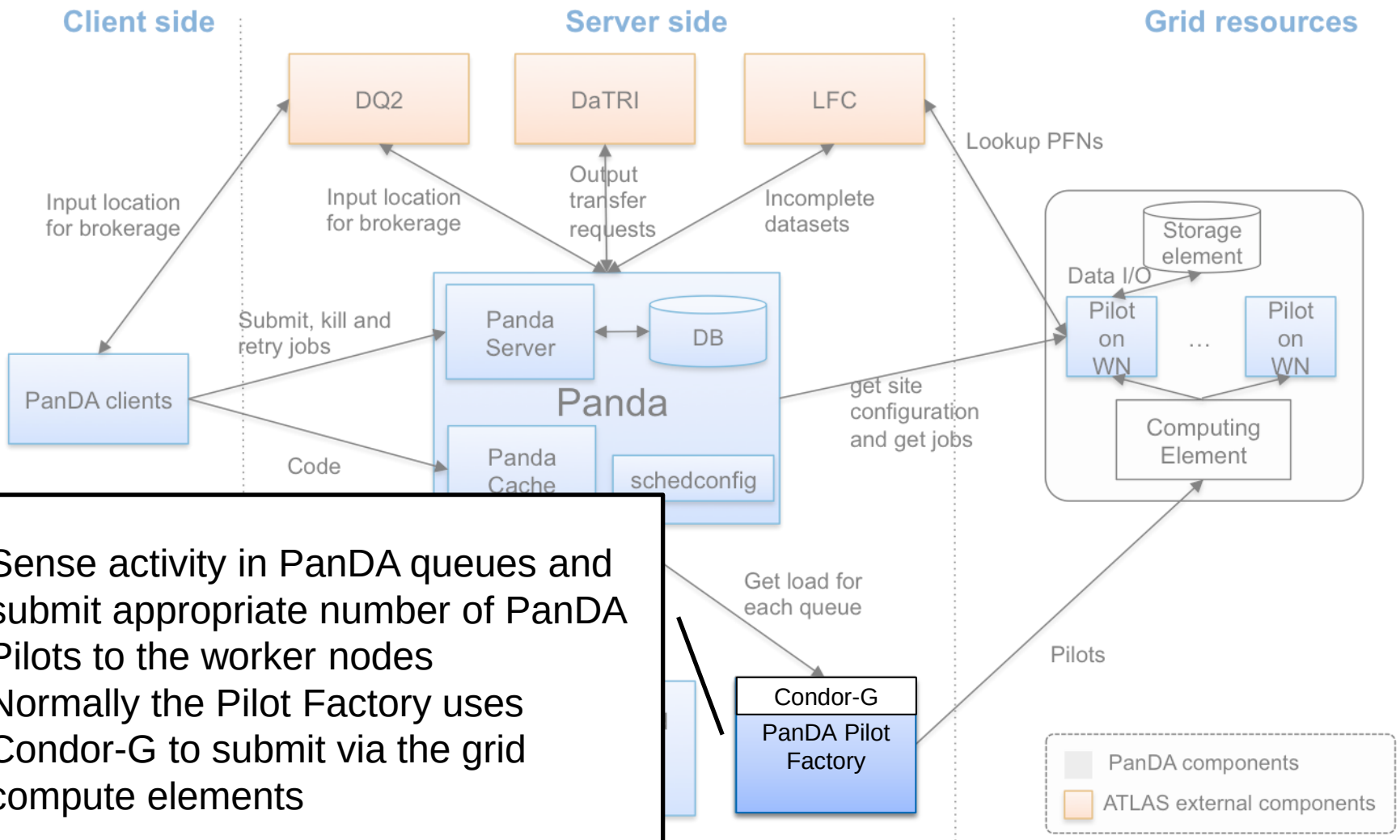
- Receives jobs from Clients
- Job brokering and re-brokering to grid sites
- Calculates job priorities based on various fair share mechanisms
- Communicates with other ATLAS services for data management operations

PanDA architecture



- Generic wrapper code which executes a payload on a worker node
- Handles environment setup, data stage-in/out, payload monitoring
- Continually communicates the state of the subjob to the Server

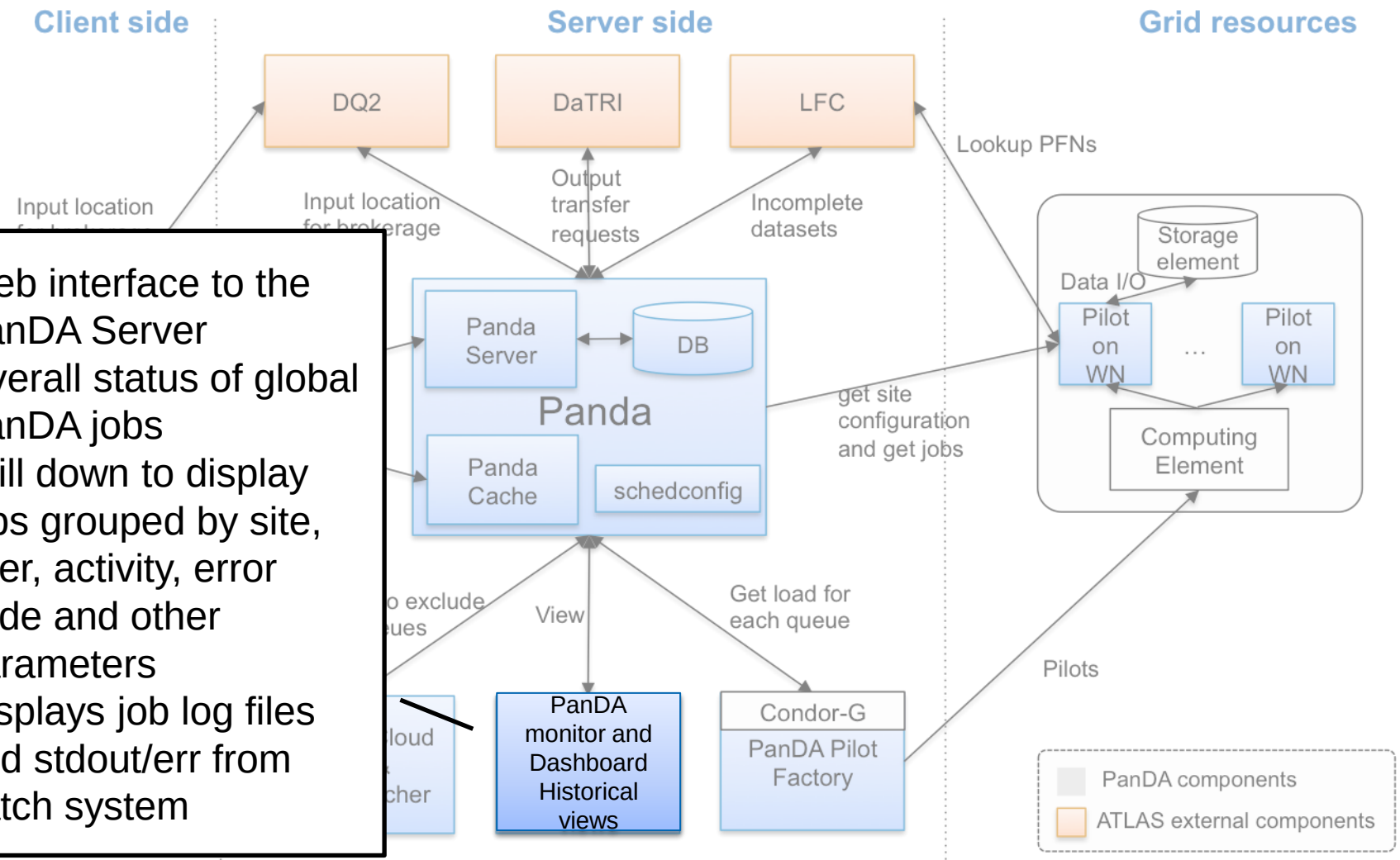
PanDA architecture



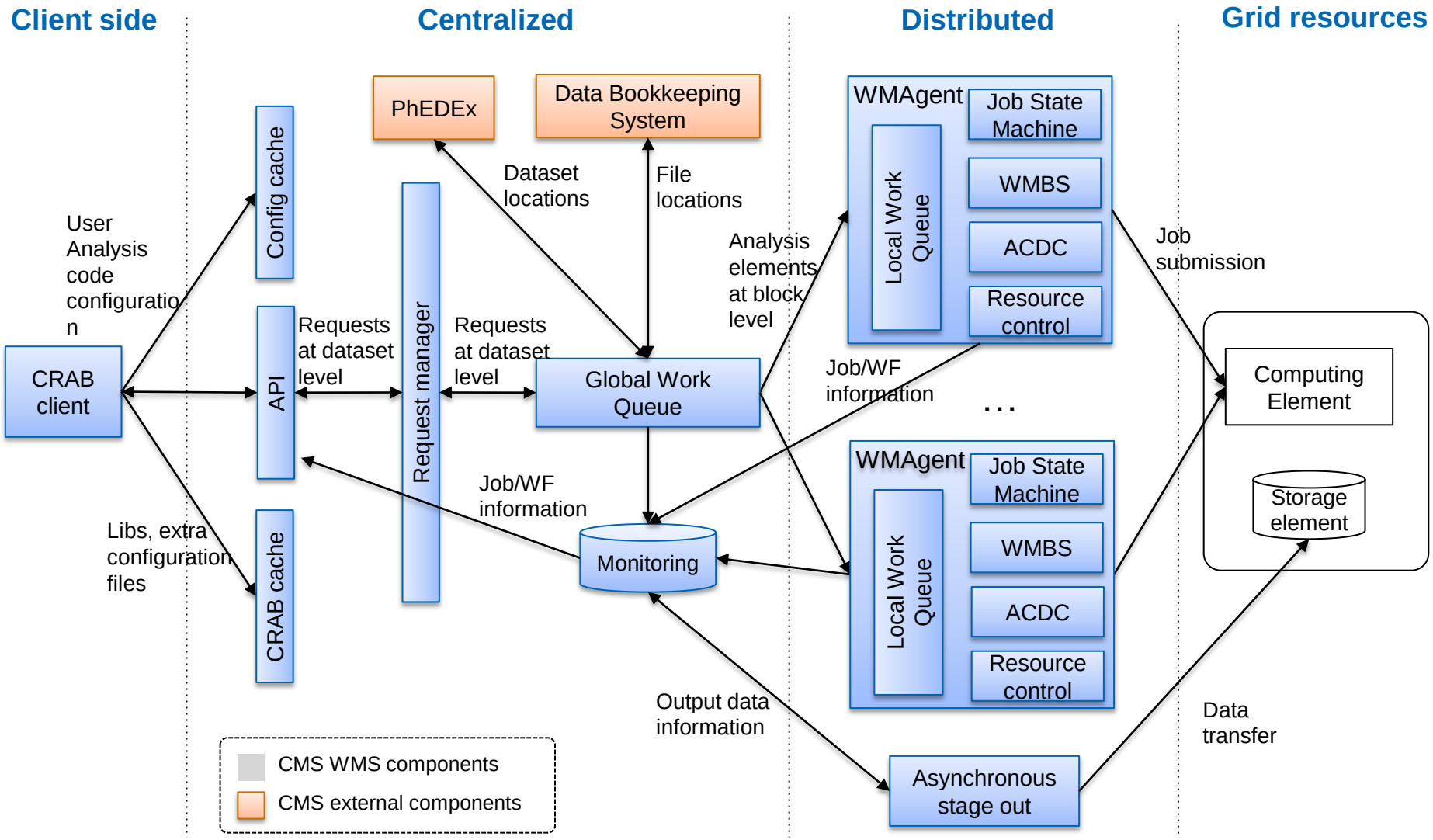
- Sense activity in PanDA queues and submit appropriate number of PanDA Pilots to the worker nodes
- Normally the Pilot Factory uses Condor-G to submit via the grid compute elements

PanDA architecture

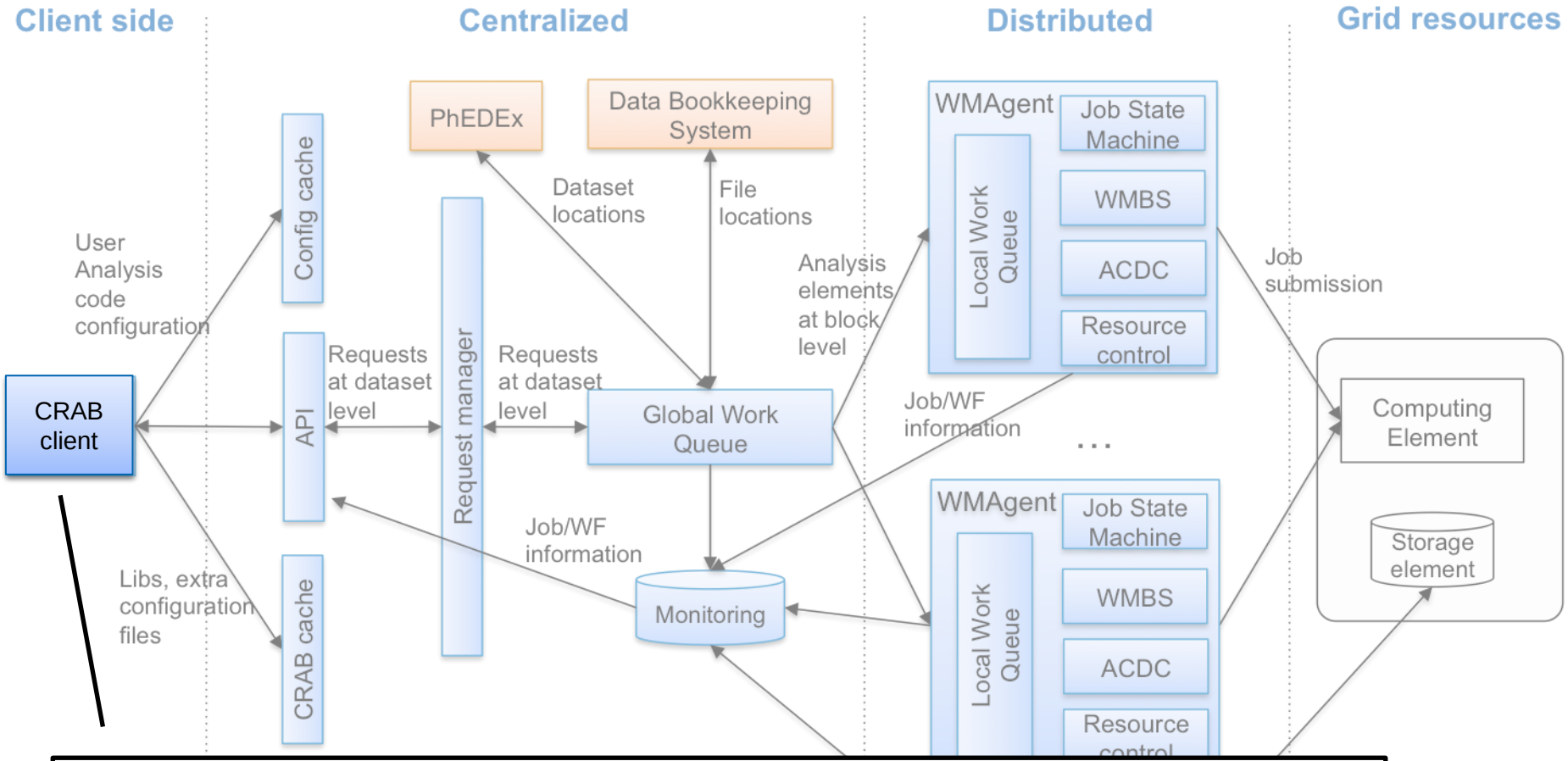
- Web interface to the PanDA Server
- Overall status of global PanDA jobs
- Drill down to display jobs grouped by site, user, activity, error code and other parameters
- Displays job log files and stdout/err from batch system



CMS analysis framework

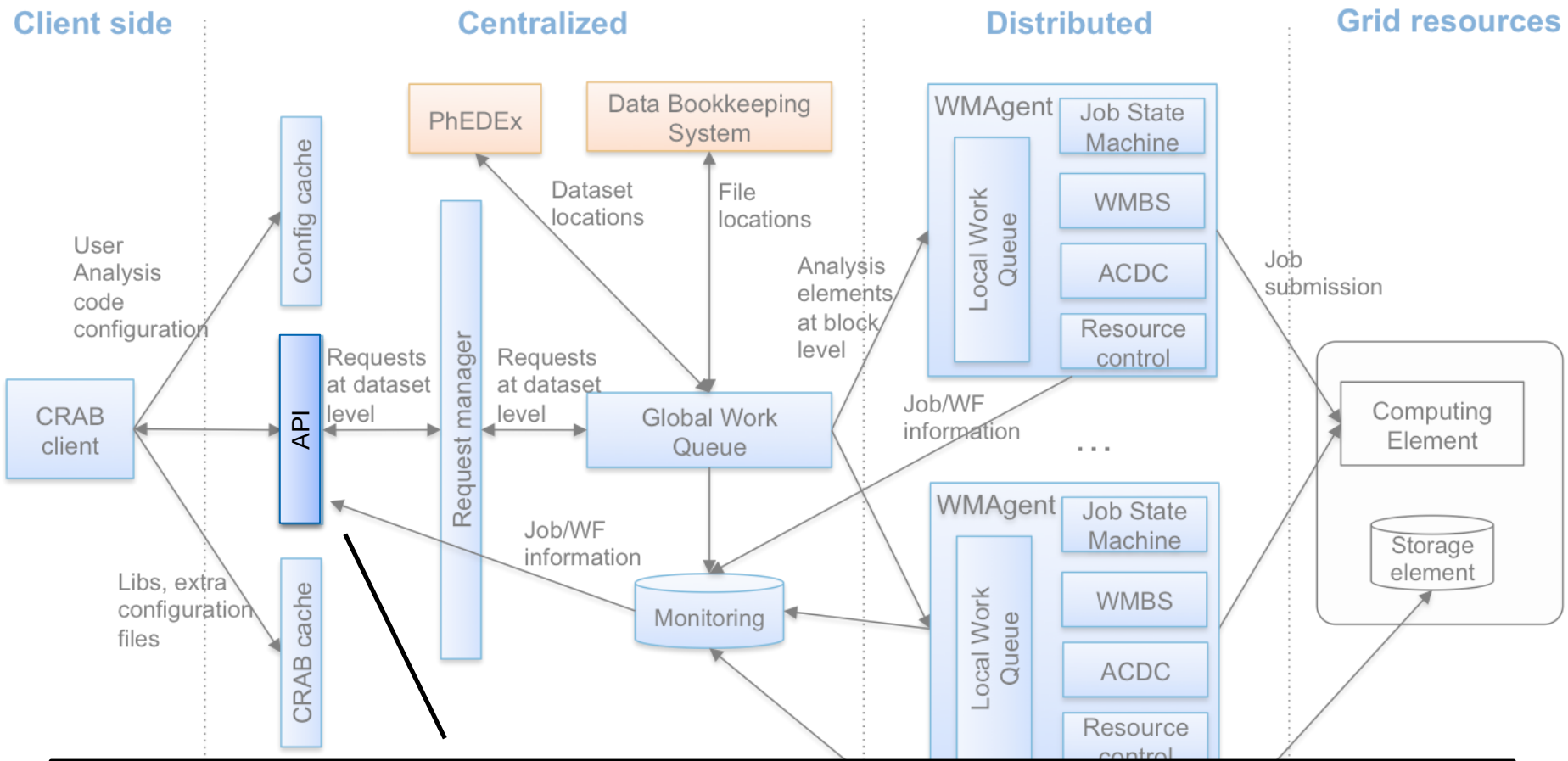


CMS analysis framework



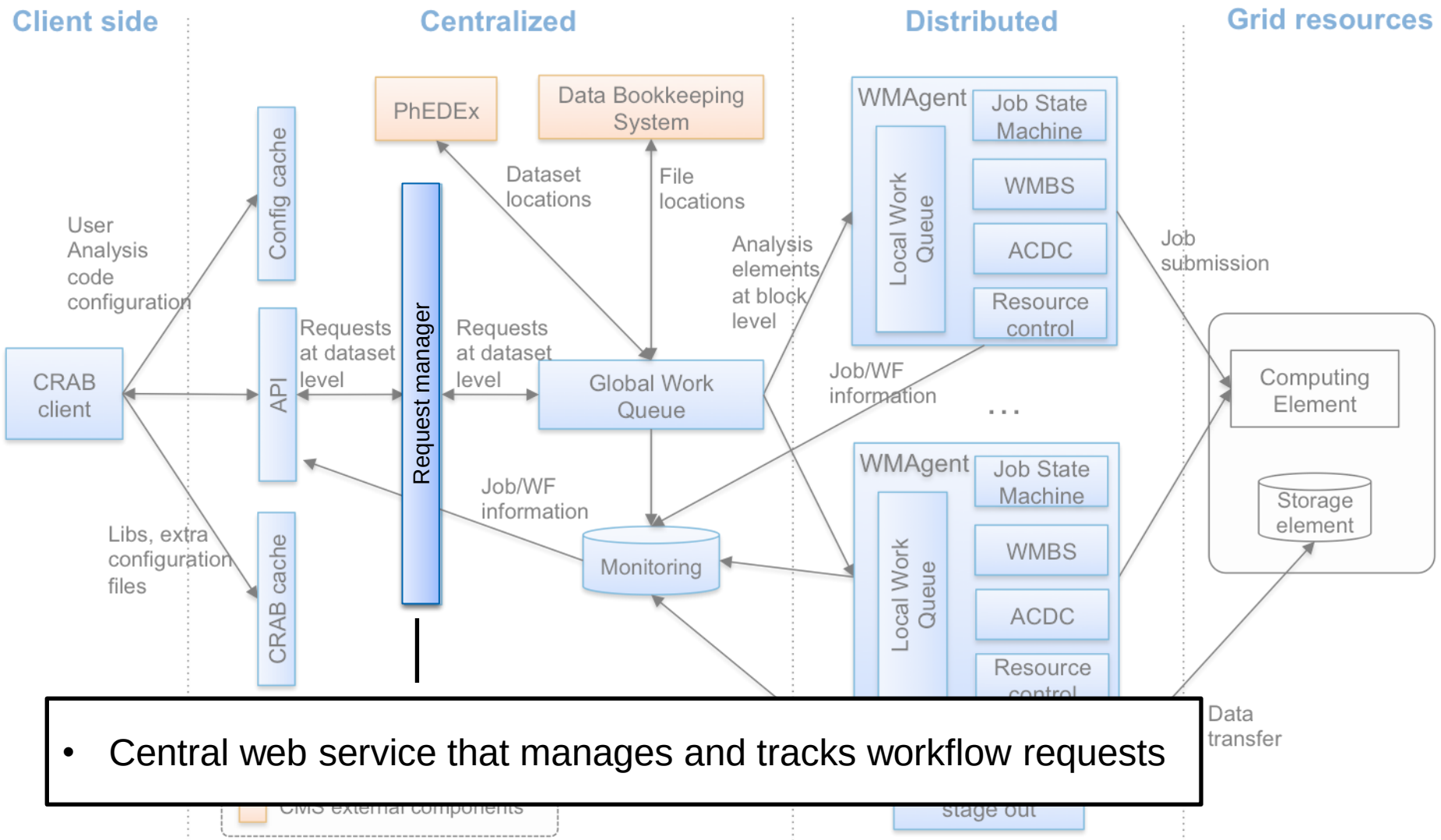
- CLI for physicists to submit, monitor and manage their analysis jobs
- Inject jobs into the WMSystem through the CRABInterface

CMS analysis framework



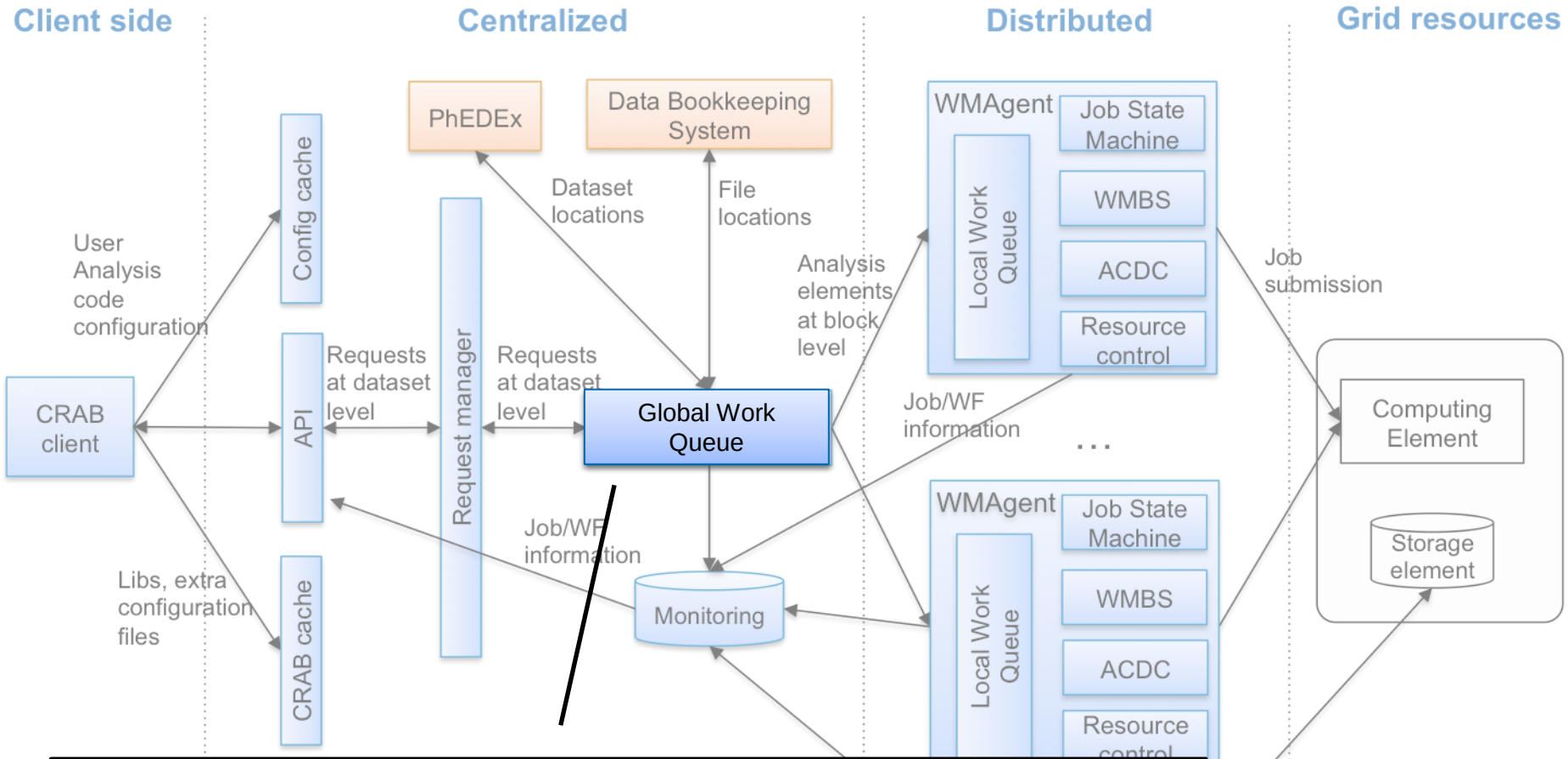
- Central RESTFul web interface that receives user requests and injects them in the system
- APIs to monitor and manage the submitted workflows

CMS analysis framework



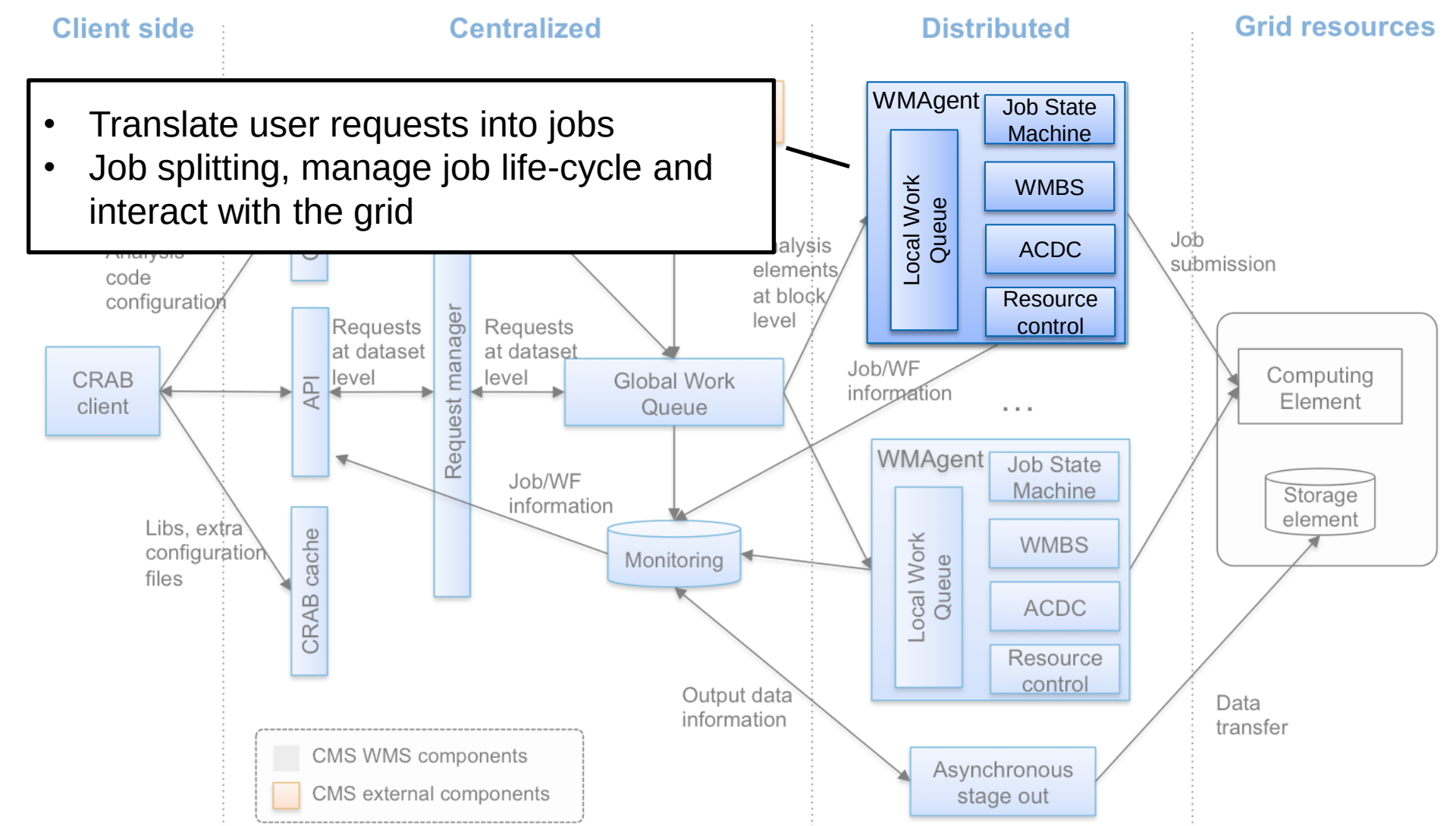
- Central web service that manages and tracks workflow requests

CMS analysis framework

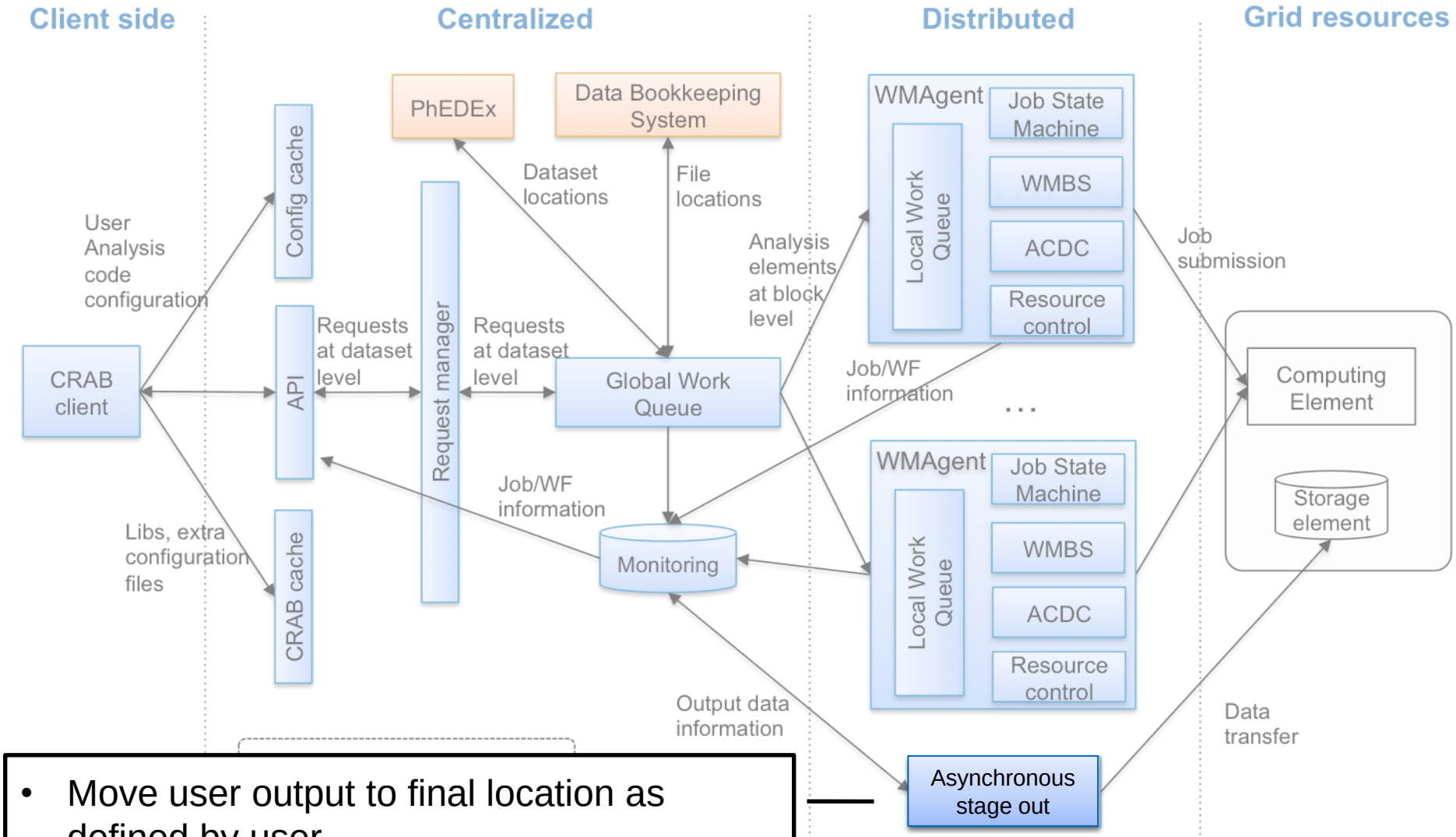


- Central queue of all user requested workflows
- Serializes requests and allows prioritization of requests

CMS analysis framework



CMS analysis framework



- Move user output to final location as defined by user

Coarse comparison

- Both experiments' analysis frameworks have historically evolved to adopt similar concepts
  - Client/server-based architecture
  - Usage of a central queue to manage jobs
  - Similar supported workflows
- Main differences between PanDA server and CMS analysis framework
  - Complexity of the systems and levels of queuing
  - Resource allocation
    - Dynamic brokerage in PanDA, more fixed in CMS WMSystem given distributed character

	Architecture	Upside	Downside
PanDA	Simple central architecture	Global view and control	Potential single point of failure
CMS WMS	Distributed 2-level queuing	High scalability & reliability	No global view

Additional features in PanDA

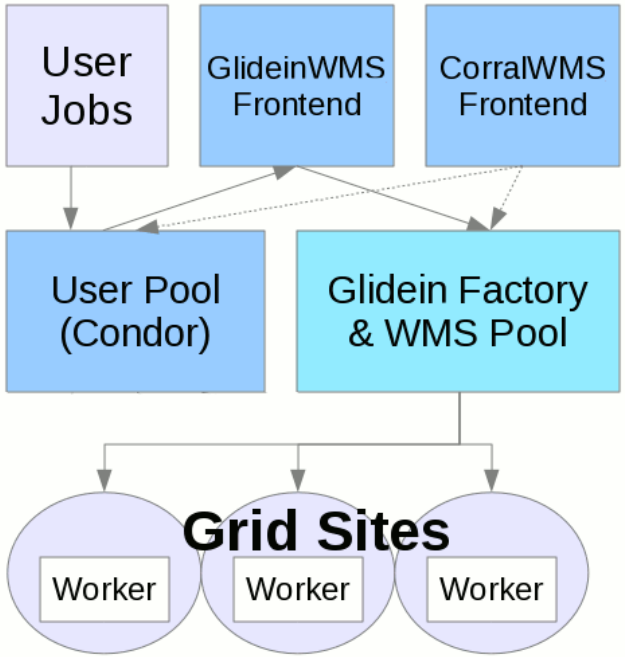
- There are other interesting/essential features in PanDA

- **Dynamic data placement and re-brokerage**
  - ATLAS has a data pre-placement model relying on dynamic data placement
  - When a jobset is submitted, PanDA can decide to trigger a replica request
  - Jobs waiting longer than *x* hours can be reassigned to another site

- **Priorities and Fairshare**
  - Users get *x* CPU hours per 24h
    - Additional jobs are de-prioritized
  - Priority boosts/beyond pledge for users and groups at particular resources
  - At submission time: Jobs in a jobset get decreasing priorities
    - a few jobs run right away to check for errors
  - Waiting jobs: Job priority increases while jobs wait to prevent starvation
  - Retried jobs get lower priority to delay slightly
  - Prod/analy balance set at site level

GlideinWMS

GlideInWMS Overview



- Build a distributed Condor pool which looks like a local batch system
- GlideinWMS automates submission of Condor Glideins according to user jobs
- Users (VOs) submit to a local Condor *schedd*; a frontend polls the user *schedd* and tells a Glidein Factory to send GlideIns via CondorG to the grid.
  - GlideIns run a condor *startd* on the WN which connects back to the user pool
- Features:
  - Credential management handled by Condor
  - gLExec id switching
  - Condor scheduling and fairshare between users and groups
  - Whole node scheduling
  - SSH-to-job
  - *Preemption*

Animation taken from
http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/index.html

GlideInWMS and CMS

- CMS is using GlideInWMS with CRAB 2 and testing with v3
  - Each CRAB 2 server / WMAgent has a local *schedd*
- CRAB server / WMAgent injects jobs (with full payload) to the *schedd*
- Using simple condor matchmaking: jobs run in FIFO order

- Condor itself has some scaling limits (provided by Igor, not definitive)

Component	Limiting factor	Observed limit
Schedd	Memory	60k jobs on 64GB node
Collector	Memory	90k jobs on 24GB node
Negotiator	CPU	40k jobs, depending on complication of matchmaking expressions

- CMS architecture allows to replicate the Agents to scale up:
  - Currently ~7 agents running up to ~20k jobs per schedd

GlideInWMS and PanDA

- ATLAS (Rod Walker) is evaluating GlideinWMS, its scalability and best way to interface PanDA and GlideInWMS
- Scaling tests don't use gLExec, only run production pilot
  - delay/avoid additional integration work, myproxy and per user pilots
  - Per-user pilots would override PanDA late-binding and fairshare mechanisms
  - (gLExec functionality has also been tested in the past)

- Schedd is ran on the current Pilot Factory machines
  - More RAM on machine allows the Scheduler to scale
    - schedd shadow-processes take 1M per running job
- VO-frontend watches the pilot factory schedd's
  - UCSD submit glideins to run the queued jobs

- Reached ~15k running jobs on over 30 sites and no show stoppers have been found
- But effort still needs to be invested in evaluating different scenarios that need to be followed up together with its side-effects (see next talk!)

Towards a common analysis framework

Towards a common analysis framework

- No show-stoppers found: ATLAS and CMS can work on common analysis framework
- PanDA found to be attractive due to its simple architecture and proven reliability
- GlideinWMS could bring in additional benefits, e.g. credential management and gLExec identity switching

Towards a common analysis framework

- If we continue towards a Common Analysis Framework
  1. Development effort to be invested in the adaptation of PanDA for CMS
     - The necessary changes are identified – an initial common architecture has been designed
     - Depending on the component the adaptation effort ranges from writing new adaptors to re-factorizing parts of the code
     - Some CMS specific components are still needed and ideally would be reused from current framework (e.g. Clients)
     - Proof-of-Concept evaluation is possible in the short term
  2. Different scenarios would have to be evaluated to interface PanDA to glideinWMS
     - Pay attention to limiting factors
     - Consider side-effects

Mattia's presentation has the details

Acknowledgements

We are very thankful to the experts for their time and ideas

- Jose Caballero
- Simone Campana
- Burt Holzman
- John Hover
- Steve Foulkes
- Claudio Grandi
- Tadashi Maeno
- Paul Nilsson
- Maxim Potekhin
- Igor Sfiligoi
- Eric Vaandering
- Rodney Walker
- Torre Wenaus

Backup slides

Common architecture