



The CNRS logo, consisting of the letters 'cnrs' in a white, lowercase, sans-serif font inside a dark blue circle.

**Centre de Calcul**  
de l'Institut National de Physique Nucléaire  
et de Physique des Particules

**BBQ – Beautiful Batch Query**

**FJPPL - January 2023**

**Guillaume Cochard - CC-IN2P3**

- Grid (WLCG)
  - About half a dozen big users (Atlas, Alice, etc.)
  - Batch scheduler: HTCondor
  - **765 workers, 37 000 CPUs**
  - Up to **50 000 standardized jobs**
- Local cluster
  - More than 100 users from the IN2P3
  - Batch scheduler: Slurm (replacing UGE/Grid Engine)
  - **423 workers, 22500 CPUs, 80 GPUs**
  - Up to **120 000 jobs** that can be very different



- High needs for monitoring
  - Nagios : incident monitoring
  - Grafana : timeseries
  - ?? : numbers, data and configuration settings in real-time or in the past
- Historically, several scripts used to monitor clusters
  - Clusters' usage
  - Ranking of users with the most jobs
  - Distribution of jobs by workers
  - Pending jobs

- Different data sources
  - Condor and UGE : parsing of command lines outputs
  - UGE : MySQL database updated every 2 minutes
- Scripts used only by clusters' administrators
- Scripts are not very practical
  - Heterogeneity
  - Requires connecting to machines
  - Limited functionalities (especially by the interface)

- Why not use a web interface?
  - Fix previous defects
  - More user-friendly
  - High potential for new features
  
- Questions
  - How to deploy it?
  - Data source compatibility
  - Technical overhead not too significant?
  
- Answer : a POC (Proof Of Concept)

## Tops

[Tops](#)

## Infos

Get informations about owner:

Get informations about job:

Get informations about worker:

- Written in Python
  - Flask: simple, powerful, efficient
  - Templating: Jinja2
  - Raw SQL queries
- Very simple, but effective
  - First conclusion: it's a good idea
  - First remark: the design is somewhat lacking
- POC validation



# Version 1 : adding Condor cluster

Beautiful Batch Query   Condor   UGE   GPU   UGE queues   UGE tops

condor

### Infos

Get info about

VO  
 owner  
 clusterid  
 routedfromjobid  
 worker

job id | owner | worker

Submit

### Search for job(s)

vo:

owner:

worker:

running after:

running before:

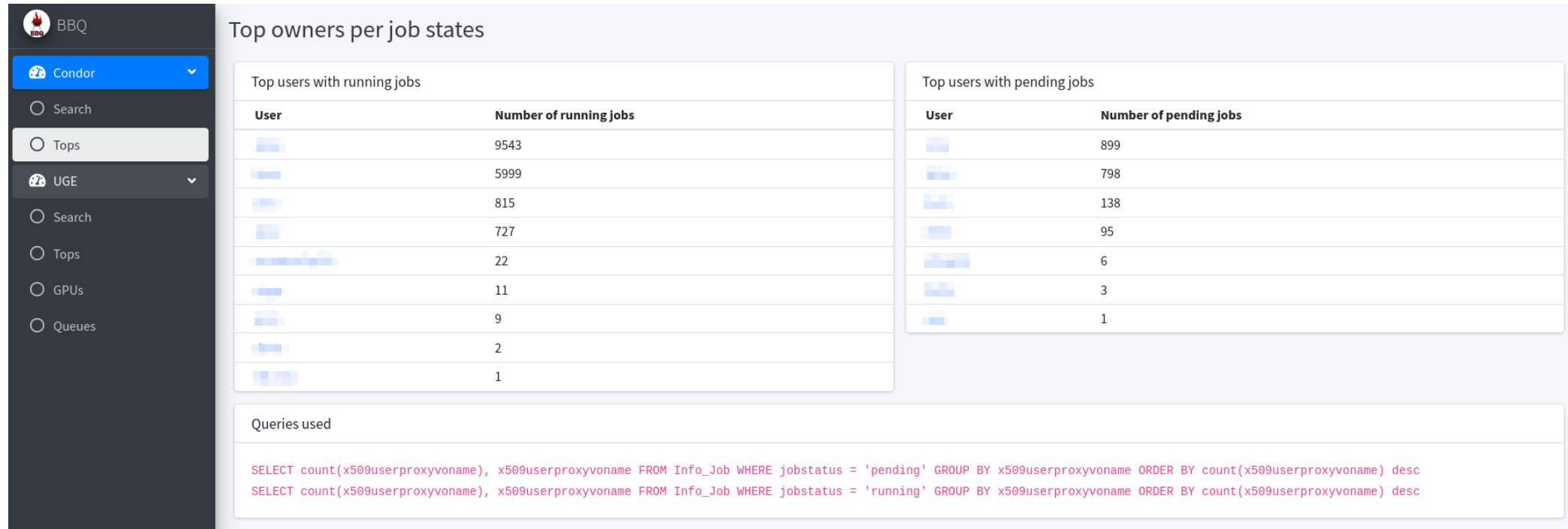
### State

Pending jobs  
 Running jobs  
 Ended jobs

Search

- Improved interface (not by much)
- Addition of features for UGE
- Creation of a database for Condor
  - Upfront work
  - Separation between retrieval/parsing and display
  - Necessary for history
- Choice to make BBQ as cluster-agnostic as possible internally

# Version 1 : mise à jour de l'interface



The screenshot displays the BBQ interface with a sidebar on the left containing navigation options: Condor (selected), Search, Tops, UGE, Search, Tops, GPUs, and Queues. The main content area is titled "Top owners per job states" and contains two tables and a code block.

**Top users with running jobs**

User	Number of running jobs
[User]	9543
[User]	5999
[User]	815
[User]	727
[User]	22
[User]	11
[User]	9
[User]	2
[User]	1

**Top users with pending jobs**

User	Number of pending jobs
[User]	899
[User]	798
[User]	138
[User]	95
[User]	6
[User]	3
[User]	1

**Queries used**

```
SELECT count(x509userproxyvname), x509userproxyvname FROM Info_Job WHERE jobstatus = 'pending' GROUP BY x509userproxyvname ORDER BY count(x509userproxyvname) desc
SELECT count(x509userproxyvname), x509userproxyvname FROM Info_Job WHERE jobstatus = 'running' GROUP BY x509userproxyvname ORDER BY count(x509userproxyvname) desc
```

- Interface with AdminLTE / Bootstrap
  - Why reinvent the wheel?



- Part-time development for 4 months
- Lots of changes since the beginning (data, features and interface)
- Still lots of new features ideas
- But the back-end hasn't evolved much: still based on raw SQL queries
  - Hard to make complex requests
  - Hard to add new features
- UGE is replaced by Slurm

- Cluster UGE → Slurm
  - End of compatibility between BBQ Condor and BBQ Slurm
- Switch to SQLAlchemy
  - Queries are no longer done "by hand"
  - Much easier to maintain
  - Very powerful
- However, local processing to limit the amount and complexity of SQL queries
  - Complex code based on multiple nested dictionaries
  - Equivalent of count() and group by
  - Less performant than SQL.

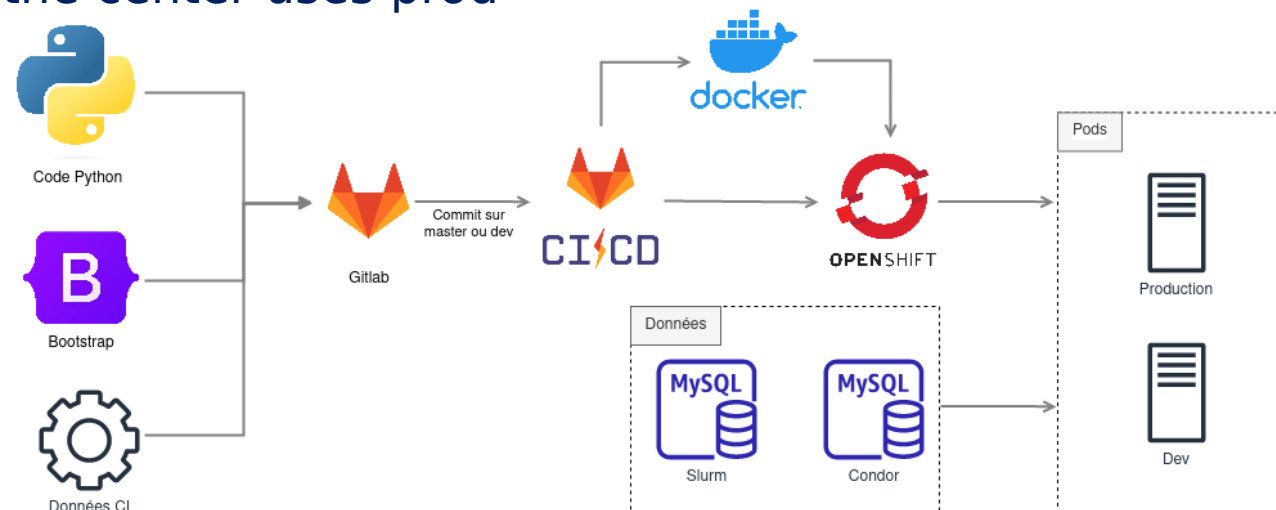
The SQLAlchemy logo, with "SQL" in a black serif font and "Alchemy" in a red, stylized, cursive-like font.

- Switch to Pandas for local processing
  - Data is processed in matrices
  - Fixes part of the complexity (no more dictionaries)
  - Pythonic solution
  
- But Pandas was not the answer
  - Performance issue once new features were added
  - It's actually quite complex too



- Python can't do everything
- SQL is still very relevant: it's powerful and efficient
- Most of the data manipulations are now done in SQL via SQLAlchemy
  - Multiple queries are still faster than local processing
  - Multi-threading for slowest pages
  - Data optimization in the database
- Page loading went from over 20 seconds to less than 5 seconds.

- Project on Gitlab
- Deployment via Gitlab-CI to push a Docker image to our Openshift platform
  - Just commit to automatically update
- Two instances: prod and dev
  - The team uses dev
  - The rest of the center uses prod



# Demo time

- Appearance matters
- Knowing what is the goal
  - BBQ started as a script but is now a complete product
- Beware of the novelty effect
  - It's not because it's old that it's bad
- Take the context into account
  - Practice  $\neq$  Theory


- Add actions directly in BBQ
  - Requires identity management
- Add testing
- Continue to improve code quality
- Adapt features to everyday usage



# Why not share BBQ (for now)

- Custom data sources
- Code is not fully independant from our implementation
- Need a bit of refactoring

Merci de votre attention

Beautiful Batch Query 

[HOME PAGE](#) [PREVIOUS PAGE](#)

Links

[TOPS](#) [UGE GPU INFO](#) [UGE USER QUEUES](#)

Infos

Get informations about owner:

Get informations about job:

Get informations about worker:

Search

owner:

*Owner of the job*

worker:

*Worker on which the job is running*

state:

*The state of the job*

running at:

*Time when the job was running. Format: YYYY-MM-DD [HH[:MM][:SS]] (omitted fields will be set as 00)*

running between:

*Format: DATE TIME,DATE TIME*

resource:

*Resource(s) declared by the job*

- Ajouts de quelques fonctionnalités
  - Validation du POC
- Mise en place du déploiement automatique
- Toujours que UGE
  - Modification des données en BDD
  - Pas d'accès aux scripts pour Condor
- Trouvaille du nom
  - Promis le jeu de mot n'est pas de moi

- Python
- Données
  - Requêtes : SQLAlchemy
  - Traitements secondaires : Pandas
- Web et affichage
  - Back-end : Flask
  - Front-end : AdminLTE / Bootstrap, un peu de Javascript
  - Templating : Jinja2 + HTML
  - Graphes : Bokeh
- Serveur web : Gunicorn

- 400 commits sur la branche dev
- Code :
  - 29 fichiers Python pour 4300 lignes de code
  - 38 fichiers HTML/Jinja2 pour 4500 lignes
- Refactoring : beaucoup trop de fois
- Développé sous VS Code