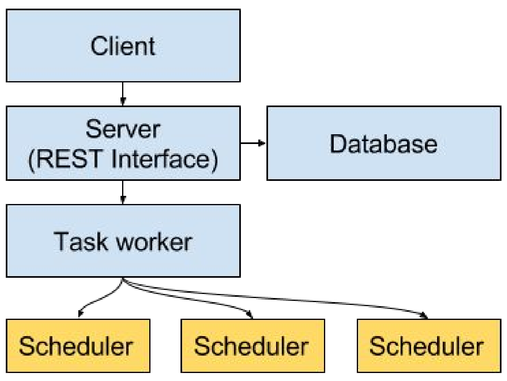
Efficient monitoring of CRAB3 jobs at CMS

What is CRAB3?

CRAB3 is a tool used by more than 500 users worldwide for distributed Grid analysis of CMS data. Users can submit sets of Grid jobs with similar requirements (**tasks**) with a single user request. CRAB3 uses a client-server architecture, where a lightweight client, a server, and ancillary services work together and are maintained by CMS operators at CERN.

- CRAB Server provides a **REST Interface** which it uses to communicate with the other components.
- **Task worker** is a queue system responsible for processing tasks.
- Information about tasks are stored in a **task database**.

Architecture

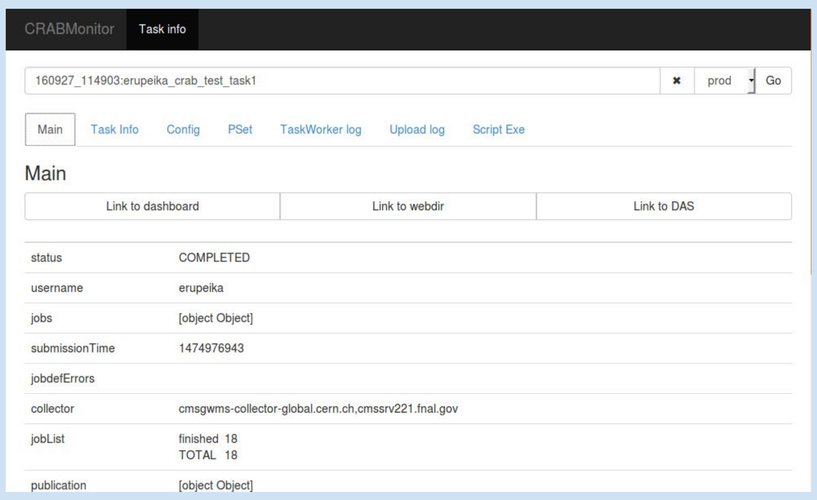


Number of tasks

One of CRAB3 operator's day-to-day responsibilities is debugging user tasks (upon user request). More than 8000 tasks are submitted every week.

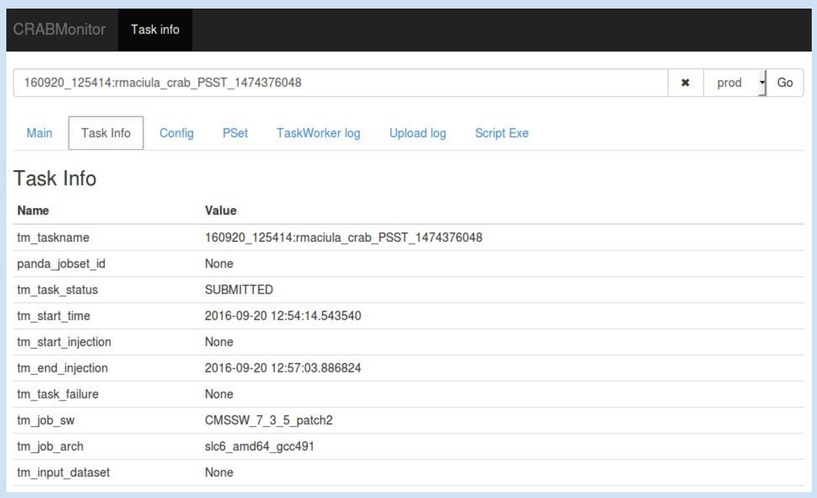
CRABMon

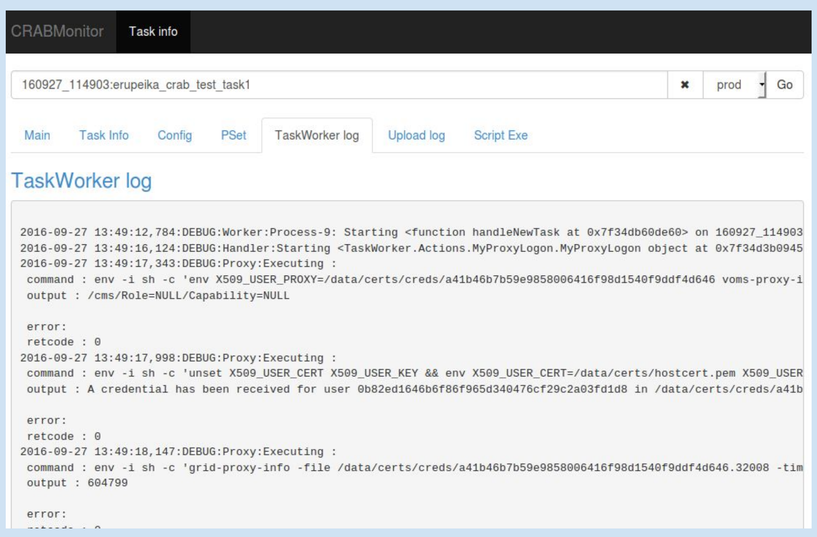
CRABMon is a javascript-based tool developed in-house to allow operators easier access to information related to a particular **task**. It is available on cmsweb.cern.ch to every CMS user.



Main view providing some general information about a task (such as task and job statuses) and links to external sites to complement available information.

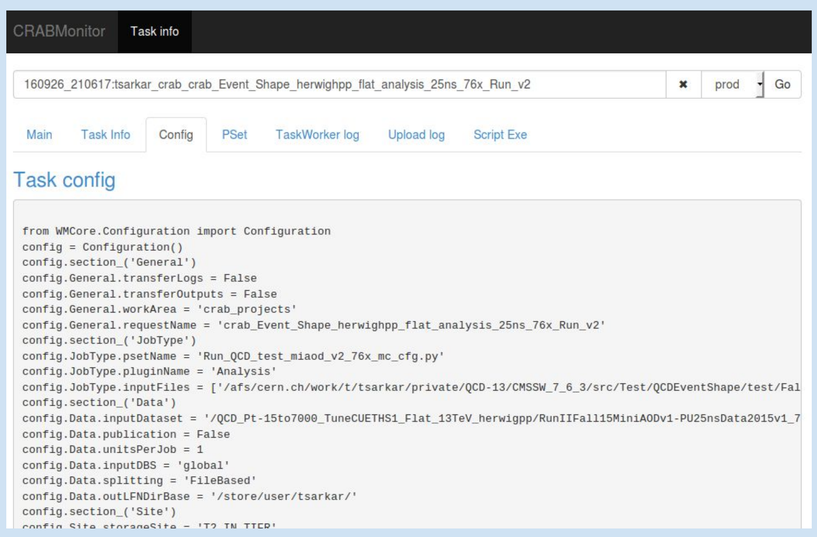
Provides an easy way to access the information stored in the task database (helps avoid writing SQL queries by hand).





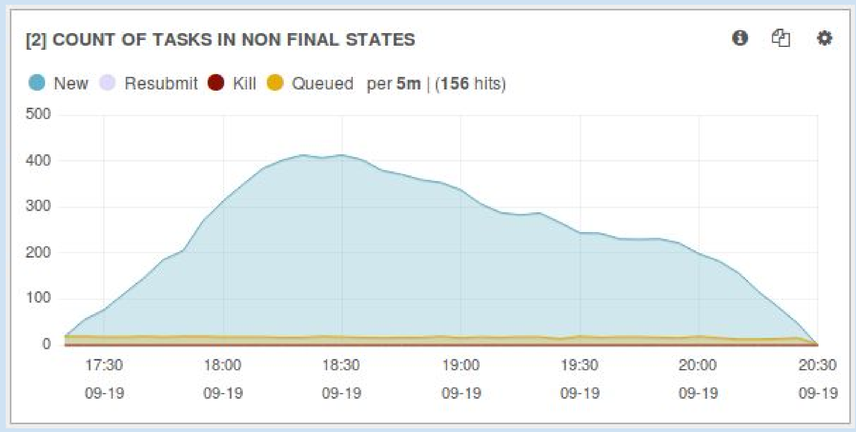
Task worker logs for a specific task. These allow debugging problems that happen during task submission.

Access to user configuration files (client configuration, framework configuration, ancillary files).

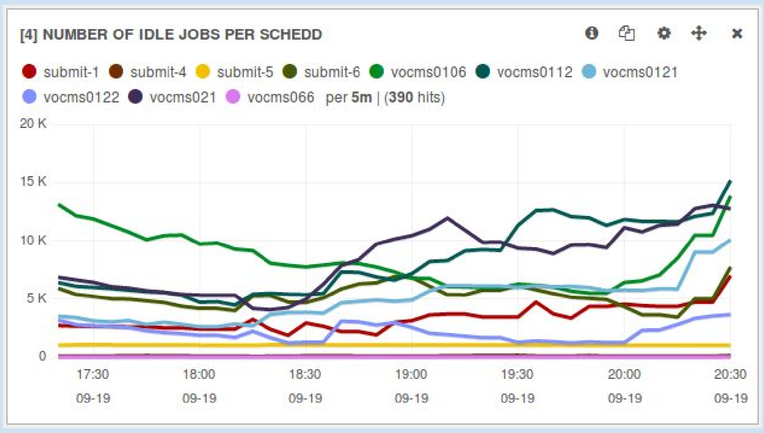


CRAB Kibana dashboard

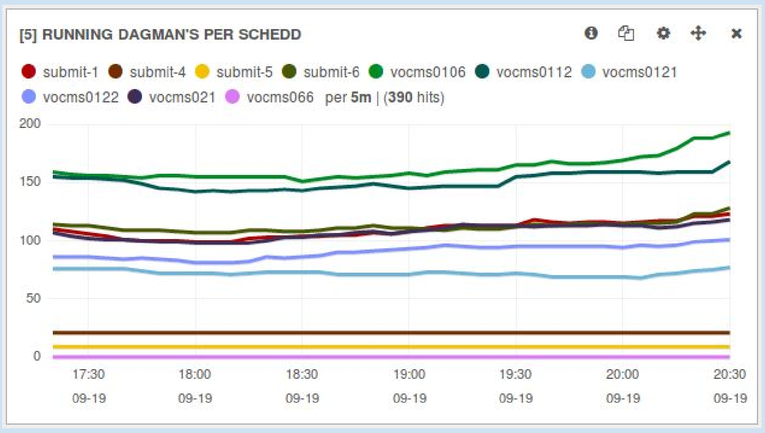
Set of kibana views deployed on the CERN Kibana instance that helps CRAB3 operators monitor the core components of the infrastructure (Task worker, schedulers, etc.) in real time.



Charts that show the number of tasks per state help identify problems with the task worker since a high number of tasks in non-final states indicates it is not processing tasks.



Monitor the number of idle jobs for each scheduler to make sure they are being processed.



Schedulers have finite resources (RAM, CPU, etc.) To prevent unexpected behavior, limits on the number of running jobs and tasks are set according to scheduler capabilities.

Marco Mascheroni (Fermi National Accelerator Lab. (US)), Jadir Marra da Silva (UNESP - Universidade Estadual Paulista (BR)), Emilis Antanas Rupeika (Vilnius University (LT))