To have the appropriate tools for the operation and maintenance of computing centers by 2020, the analysis of massive data produced by the supercomputer seems essential.

We propose here to analyze two types of data: the metadata of the calculations submitted to predict the power consumed and the text logs emitted by the computing nodes to check their state.

Power prediction

Inputs that can be used for prediction
- The inputs are the informations given by the user about his needs:
  - UID (User ID), GID (Group ID of the user)
  - submit date
  - number of nodes (asked, allocated) or number of CPU's (asked, allocated)
  - timelimit (duration before killing the job), QoS (Quality of Service, test or production job)...

Outputs, collected after computation
- time elapsed (duration of computation)
- consumed energy (in Joules)

Notice: one core consumes about 10 Watts when active, so 300 Watts/node.

COBALT’s data: the color is given by the GID, the vertical axis is the power by node.

Predict Power/nodes = \frac{\text{consumed energy}}{\text{nodes} \times \text{Y}}

Because this is a too simple model, we use also bagging and boosting which gives us respectively gradient boosting regression and random forest.

Reweighting and online computations for global power consumption monitoring

The GID and the number of tasks per nodes are the most characteristic inputs for estimation. We compute the classical estimators of statistics on the empirical probability distribution conditioned on the possible values of the inputs:

\mathbb{E}[Y|X = x_i] \quad \text{with} \quad \mathbb{P}(Y < y | X = x_i) = \alpha

Textual logs analysis

What is a log file?
- Logs: event dated as a slightly formatted string, displayed by a console program and saved in order to ensure its proper execution.
  - Difficulty reconciling textual and temporal informations
  - "bursty behavior": the intensity of event is spiked
  - message not very structured with more or less formal formats (syslog)
  - important messages are hidden in redundancy (i.e., inuites)
  - the logs are spread in many files (cf \"var\log\" folder)

Logs structure and templates

We need to gather the logs and one way to think that two logs are equal is to infer if they are emitted by the same line in source code. So they should share a template. To extract these templates, anonymizing the logs is a first step. Each template is then hashed to a number so that it is uniquely defined.

This gives use the steps bellow: brut data, anonymized and finally hashed.

Outlooks

We will focus on the events marked as shown on the graphs. This representation is the starting point for methods to create alerts or extract the underlying structure in order to detect anomalies or to perform predictive maintenance.