Summer Student Project

An alternative approach to configure permanent tasks in LHCb Online farm nodes

Krzysztof Wilczyński

Supervisors: Markus Frank, Beat Jost

August 28, 2018

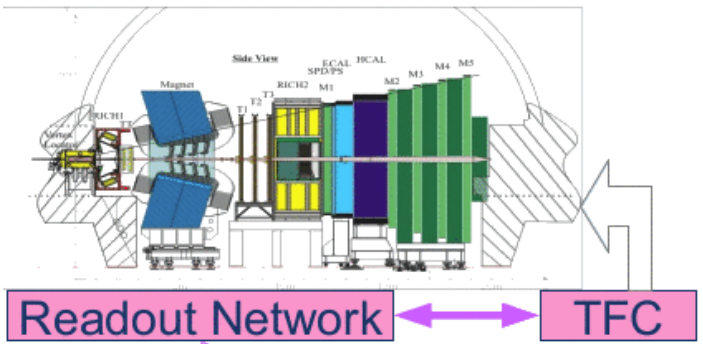
CERN, Meyrin

About me

- I am... an engineer!
  - Master student of Automatic Control and Robotics (spec. Robotics)
  - Faculty of Power and Aeronautical Engineering, Warsaw University of Technology
- I used to work at...
  - Bosch Rexroth
  - Airbus Military Defence and Space
  - Student Association for Vehicle Aerodynamics
  - Polish National Opera

Online farm computing nodes

Data Processing Apps > 80.000 Instances



Readout Network

TFC

Data Quality Cluster O(100)
- Process data from disk

CPU Reader

CPU CPU … CPU

Storage Cluster O(100)
- File and Stream handling
- Fork data streams

Storage System

Switch Switch Switch

HLT CPU Farm

High Level Trigger
HLT1: O(3000-25000)
HLT2: O(25000-50000)
Infrastructure: O(15000)

Partially replicated data streams:
parasitic, best effort

Monitoring Cluster O(100)
Low level monitoring

Reconstruction Cluster O(100)
High level monitoring with fully reconstructed events

The handling of the permanent processes on the data processing nodes is based on sending commands to pcSrv process running on each of the corresponding "Controls PCs".

LHCb Online Farm Process Controler on Researchgate

Farm boot script

In the current solution, all processes (scripts) started on the farm nodes are grouped in a single, huge python script that prints out ready to execute pcAdd commands for a given node name.

A command used to start a task on node(s):

pcAdd(regex, start parameters, script, script parameters)

Farm boot script

- The main disadvantages of the solution:
  - Modifications of task parameters are difficult
  - It is easy to make a modification that harms dependencies in the task sets (no error prevention mechanism)
  - Only a specialist who knows the boot script structure can use it (no high-level interface)
  - There is no easy way of knowing which tasks are running on given node (one has to analyze the boot script line by line)

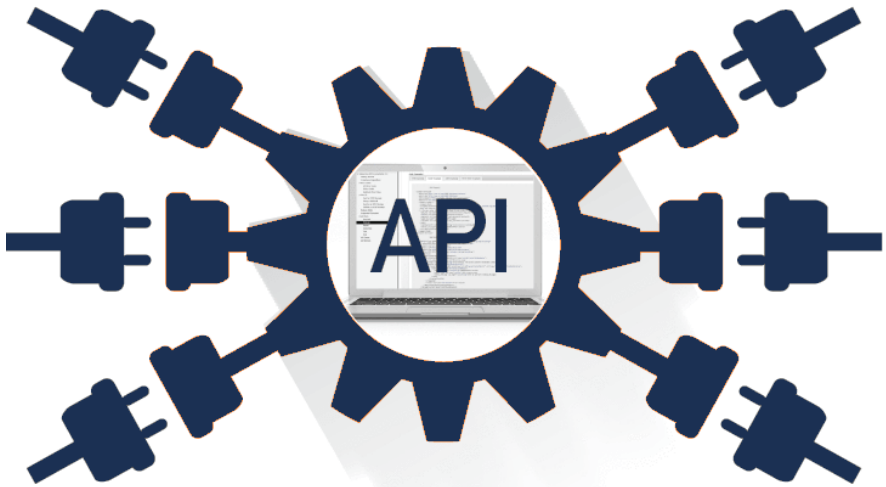
The boot script has been created as a "quick hack" about 10 years ago.
The time has come to upgrade it!

My project

The solution: create a system for the process controller infrastructure utilizing a database driven approach. The main goals were to:

- simplify the modifications of hierarchical structure of tasks running on the nodes
- prevent human errors breaking the system integrity
- create one source of information regarding processes running on a given node
- create a reliable and future-proof API for future developments

What do I mean by Main API?



API: Application Programming Interface

The created API is a Python class containing methods (add, delete, modify, get, assign, inSet) that allow safe access to the underlying database. It is a high-level connector providing an easy integration of different client applications.

Stages of development

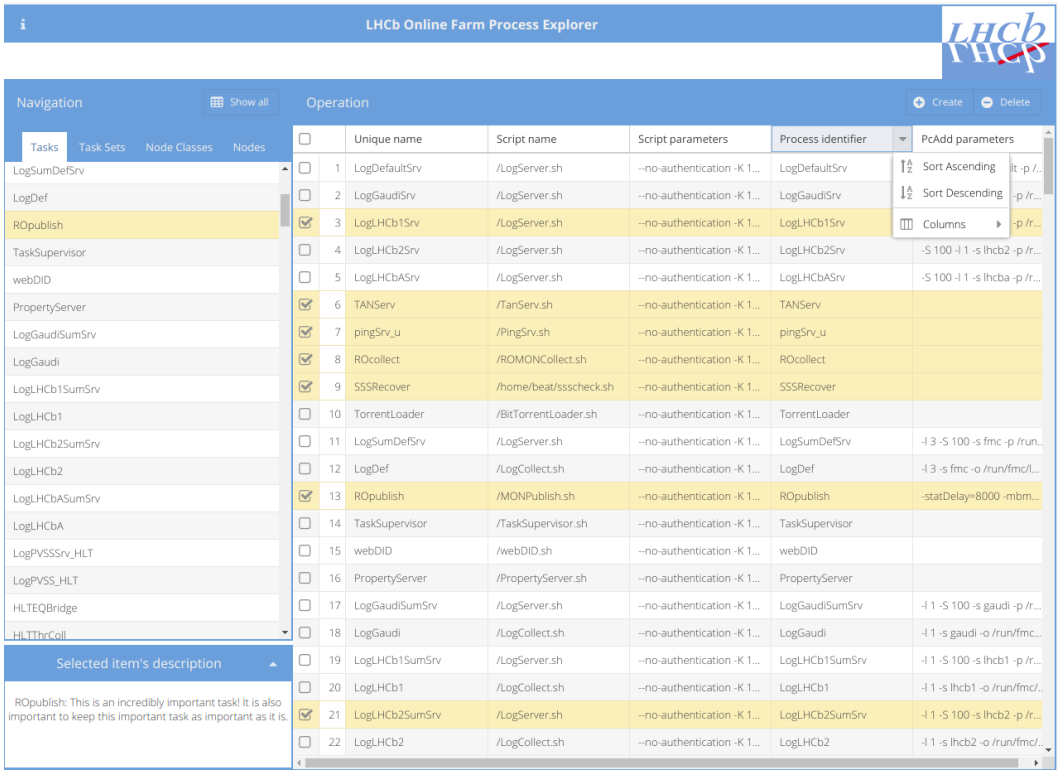
Back-end

- Database schema architecture
- Main database API
- New boot script
- Unit testing script (internal error prevention)
- Frontend connectors: JSONRPC, (REST, XMLRPC)

Front-end

- Command line user interface
- Graphical user interface (web application)

Current results



Open-source code repositories:

- Bitbucket Repository (K. Wilczynski)
- Will be moved to LHCb Gitlab Repository (M. Frank)

Tools used in the project

- Back-end programming language: Python 2.7.15rc1
- Front-end programming language: JavaScript
- Database engine: SQLite (+ python sqlite3, sqlalchemy)
- Front-end connector protocol: JSONRPC
- Front-end framework: Sencha Ext JS ver. 6.2.0
- Git version control: Bitbucket + GitKraken

Thank You,

Task grouping hierarchy