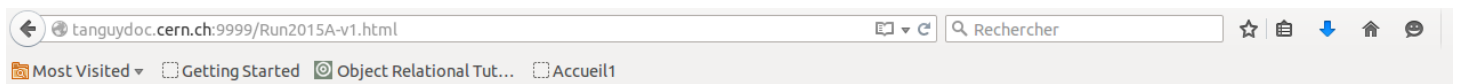


Web service for the monitoring of dataset processing in CMS



ALCaReco monitoring for Run2015A-v1

[\[index\]](#)

Opening data in a window when generating plots: ☐

ALCaLumiPixels: [LumiPixels](#)
Commissioning: [SiStripCalMinBias](#) [SiStripCalZeroBias](#) [TkAlMinBias](#)
Cosmics: [DtCalibCosmics](#) [HcalCalHOCosmics](#) [MuAlGlobalCosmics](#) [TkAlCosmics0T](#)
HcalNZS: [HcalCalMinBias](#)
MinimumBias: [SiStripCalMinBias](#) [SiStripCalZeroBias](#) [TkAlMinBias](#)
SingleMu: [DtCalib](#) [TkAlMuonIsolated](#)
ZeroBias: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias1: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias2: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias3: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias4: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias5: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias6: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias7: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)
ZeroBias8: [LumiPixelsMinBias](#) [SiStripCalZeroBias](#)

ALCaLumiPixels

LumiPixels
Number of events per run

/ALCaLumiPixels/Run2015A-LumiPixels-PromptReco-v1/ALCARECO
Selection efficiency per run

Generate plots

Link to [data](#) file used to build the plot.

Introduction

Event reconstruction is the act of transforming binary data from CMS into physically understandable variables (ie : angles, velocity, energy...). This reconstruction takes place with a dedicated software and infrastructure that we will explain later. Nonetheless physicists realize that some problems occur when the reconstruction is going on; due to imperfection of the procedures a part of the derived data get lost during the process. In 2012 they decided to create a web service from which we are able to see analysis of the reconstructed data and thus find the missing data. This website however needs updates with the new run of LHC in 2015.

Few definitions

Event:

Collision of protons developing multiple other particles.

Luminosity section:

This is the period of time (app. 23s) during which the CMS experiment is recording events. One run of the experiment is partitioned in time by lumisection, each lumisection containing events.

Run:

This is a period of time during which the experiment is recording events. When the experiment stops, the current run is stopped too and a new run is started when the experiment starts again.

Acquisition era:

This is a period of time corresponding to a set of runs, generally one year is cut in few parts. It appears in this explicit format : "Run2015A". (For the moment year 2015 is cut in 2015A, 2015B and 2015C).

Dataset:

A dataset is define by an acquisition era, the name of the event topology it contains (eg: singleMu, doubleElectron...) and the format of the data called data tier (eg : RAW,ALCARECO...). The format determine the type of the data stored (RAW is the binary data, the parent of every other data tier).

The dataset contains a variable amount of runs, each of them containing some lumisection.

Overall presentation

How reconstruction works ?

Our basic structure is several RAW datasets, each of them containing runs, containing lumisections, containing events. The reconstruction takes each lumisections one by one and

produces physically understandable data. We have thus the RAW dataset, called parent, and from it we create a lot of new other datasets like ALCARECO, RECO, AOD...; called children.

What is the problem ?

During this reconstruction it appears that some time a lumisection is not reconstructed well, and all its events are lost during the process. This might be due to atypical conditions of the detector or problems with the software used for the reconstruction.

What was the solution ?

A web service was developed. It was displaying for every acquisition era for the children ALCARECO the number of events in each run. We were able also to see a graph with the ratio of the number of events of the children dataset (ALCARECO) and the number of events of the parent dataset (RAW). This system allowed us to search for the missing events in the children.

Now this solution is outdated, the service does not have the new acquisition eras and it does not display all the options that could help physicist to solve this loss of data. We have to move to another service, more relevant, with more data and updatable.

The new service



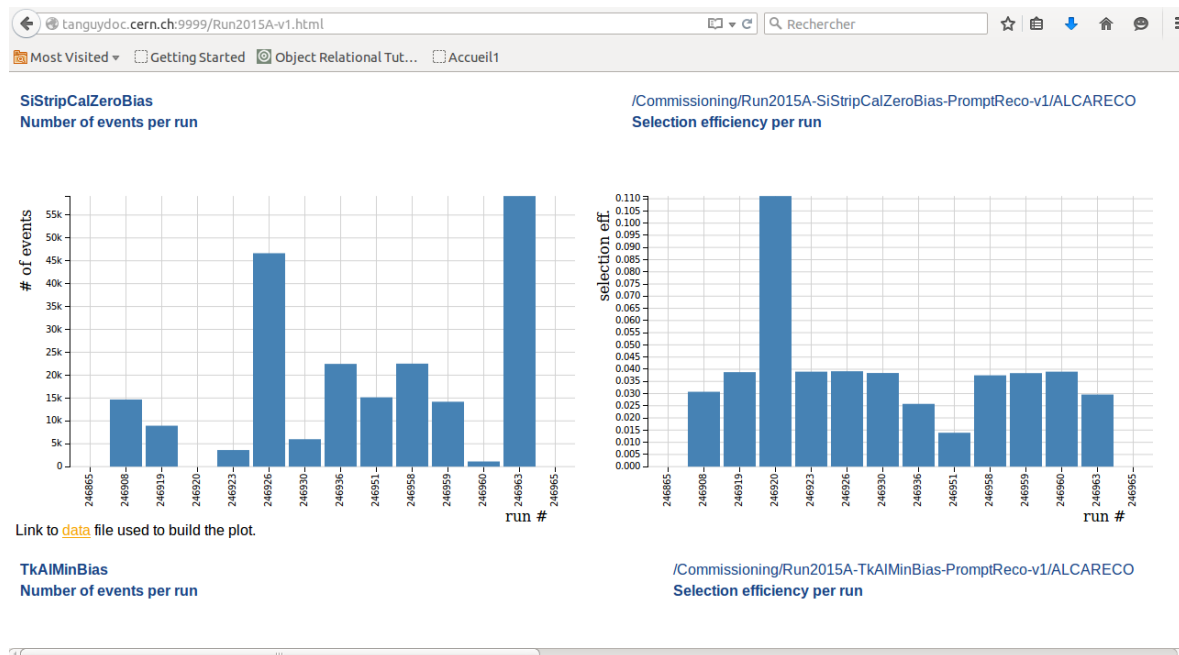
Data acquisition Era	Processing version	Link to plots
Run2015A	v1	plots
Run2015B	v1	plots
Run2015C	v1	plots

How is the new website ?

The new website has the same display than the old one and is really easy to use. Just by following links towards the website we have access to every acquisition era. The index page displays all the acquisition era for each kind of data tier (children dataset). Choosing a specific children and specific acquisition era leads the user to a dynamic webpage where one has access to every option to display histograms (see figure 1) and tables presenting data points.

This webpage is the heart of the analysis of the mentioned problem. The user can choose here to display the histograms he wants depending of the specific children he wants to zoom in. He can customize options on the website to make his work easier when analysing and searching data by an easy-to-use interface.

Figure 1 : histograms from the new web service



What is the infrastructure ?

A web site called DBS is the authoritative source of information about datasets and events from the experiment.

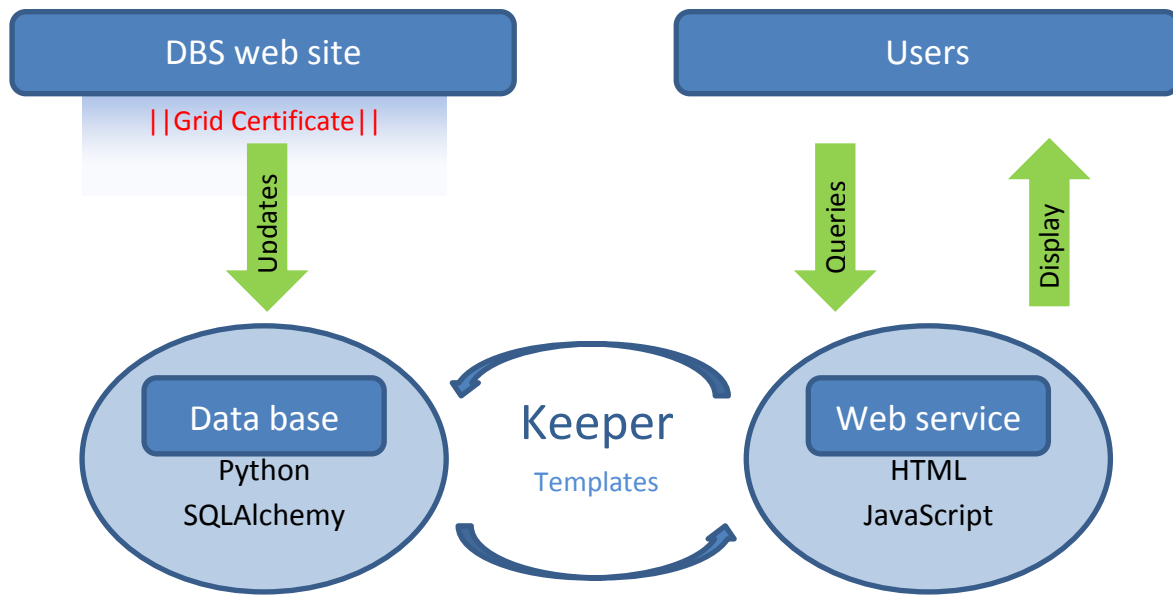
First we had to build the database for our service holding part of the information from DBS. We built it with Python, using SQLAlchemy. We create a program that is able to take specific data from DBS and put it on the database to update it (A grid certificate is needed to access DBS). With the database comes a program enables us to make queries in the database to find exactly what we are looking for. Thanks to SQLAlchemy the usage of queries is really simple and thus the database can be manipulated how exactly we want.

After the creation of the database we could start writing the interface of the web service. Using HTML and making it dynamic with templates files we created the structure of the web pages, how they are linked together and how they display their content. When displaying the content we used the templates to access the previous Python files and make the queries relative to the page. The already existing keeper service was our base to host the web service. Keeper is a service that generates all the basic files to be able to develop a secure web application at CERN. It enables such things than creating the templates file, generating the http address with port or study what could be wrong with a logs folder.

We used cache files containing part of the database to print the plots and generate data sheets, making the web pages quick and efficient.

Finally we used JavaScript to display the histograms, and make the interaction between the webpage and the user. Every move the user can make and its consequences on the display are handling by JavaScript (see figure 2 to have an overall view of how the process works).

Figure 2 : Process of the activity of the web service



Conclusion

The new service is now deployed and will start to be used soon. In the further weeks some will expand and generalize its functionality. Its new interface and the fact that it is easily updatable give it a lot of chance of success in its task to replace its old version. Because of its functionality this service can manage to be the solution of the problem physicists encountered and we really hope that it will help them to figure out the problem and solve it.