USE OF GRID AT CMS: SOFTWARE VALIDATION AND TRACKER ALIGNMENT

GREGOR BRUNS, University of Leipzig

Abstract. Here I will describe the results of my stay at DESY during the Summer Student Programme 2009. My project focused on two topics in connection with the CMS experiment: validation of new software releases and the use of the LHC Computing Grid in the alignment procedures. In the first case I wrote a Python automatization script for the validation procedure, including an easy-to-handle graphical user interface. This script makes use of the Grid by employing CRAB, the CMS Remote Analysis Builder.

For the alignment procedures, the use of CRAB was also tested succesfully. By modifying Perl scripts from the MillePede Production System (MPS) the embedding of CRAB into this system can be achieved.

1. Introduction

Annually, the Summer Student Programme of DESY gives students from all over the world the chance to actively participate in a research project in one of the important particle physics labs of the world. I was selected as a member of the CMS group for the eight weeks of my stay. While the CMS experiment is located at CERN, the DESY facility has its own group working in connection with the group at CERN.

When I arrived at DESY, there were a lot of application procedures to be fulfilled. Accounts at DESY and CERN had to be requested and a certificate for Grid Computing was to be validated. Since these steps took almost a week, I had time to familiarize myself with the working environment I was going to use. The main programming language that encompasses interaction with the CMS Software (CMSSW) is the Python scripting language. While I knew some aspects of this language at my arrival, there was still much to grasp and I spent most part of the first week getting to know Python better. For data analysis the ROOT toolkit, a C++ library for which a python wrapper is available, is used most of the time.

In the following sections, I will describe the basics of the Large Hadron Collider at CERN, the CMS dectector, the Grid computing network and of alignment procedures. I will also report on the work I have done and the results achieved.

1.1. The CMS detector.

The CMS (Compact Muon Solenoid) is one of the four detectors at the LHC. Besides ATLAS, it is one of the multi-purpose-detectors that are suited for different kinds of investigations like searching for the Higgs boson or supersymmetric particles. Figure 1 shows the general structure of the LHC with its preaccelerators and the relative location of the experiments.

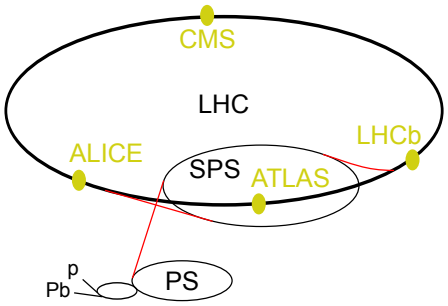

Figure 1. The general structure of LHC.

The LHC[1] will provide proton-proton collisions with a center of mass energy of 14 TeV and a design luminosity of $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$, which is a remarkable improvement in comparison with earlier hadron colliders. As usual for colliders, particles are accellerated in bunches, so that there is a high number of collisions every time a bunch crossing takes place. The energy levels that can be reached

with this machine will be high enough so that definitive answers to challenging problems of high energy physics can be given, for example whether there is super symmetry or a Higgs boson. Furthermore, there will probably hints on other extensions on the standard model of particle physics, for instance if there are extra dimensions on small spatial scales to be discovered. Especially important are answers in this area in regard to the formulation of a theory that unifies the strong, the weak and the electromagnetic force.

The CMS detector will play an important role in this search. Because of the high luminosity and energy, there will be high levels of radiation, especially in the collision area, which imposes the need of a very robust design. Another aspect is the selection of relevant events from the huge amount of about 10^9 inelastic collisions per second at the highest energy. A trigger system was therefore designed to select only about 100 events per second for further storage and analysis. The remaining events are discarded. Furthermore, the time between bunch crossings in the beam line will only be 25 ns, which calls for a fast read-out of the data. This means also that the time resolution of the detector has to be very good, so that later events do not pile up on their predecessors. Of course the resolution of energy, momentum and angles has to be very good, too.

As can be seen in the figure 2, the CMS detector consists of several detector sections: The tracker, electromagnetic and hadronic calorimeter, the solenoid and the muon chambers.

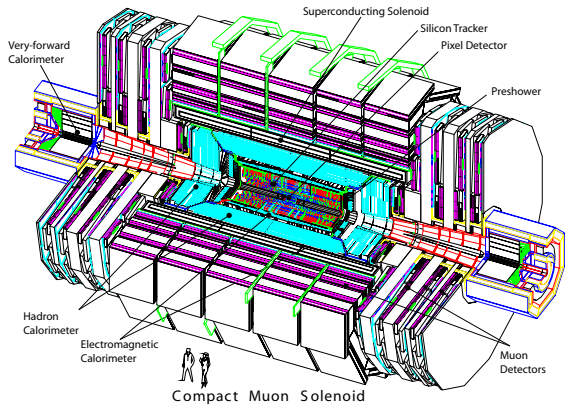

FIGURE 2. An overview of the CMS detector parts.

Each component is splitted into several subparts. The tracker is composed of a silicon pixel detector and a silicon strip detector, which are in turn again split in barrel, endcap or disc parts. More details on the CMS design, especially details about the materials and detector techniques used, can be found in the technical design report [2].

1.2. **Alignment.** While the parts of the detector can be put in place only with a precision in the order of 100 μm during the installation, the detector modules itself achieve a precision in the order of a few μm in detecting particle hits. This fact makes it necessary to accurately estimate the position of the detector parts by other means, a process called alignment. Of course, while LHC is running, there can be additional shifts and displacements for instance due to thermal expansion in the detector. The alignment task is therefore continuous and is not ending with the start of the experiments.

Three different kinds of alignment are used. First, the position of the detector modules can be more accurately directly measured as they are installed. Second, and even more accurate, a laser alignment system can be used, which consists of dedicated hardware installed into the CMS detector. Laser beams are deflected by mirrors with known orientations and the positions of the modules can be determined from shifts and deflecting angles. The third method is the most precise and is called track based alignment. We will consider this procedure in more detail.

Imagine a particle going through the detector, creating hits on its way. The modules will report the coordinates of the hit and based on these data the trajectory of the particle is reconstructed. Based on this trajectory a set of expected hits is obtained and the shift is measured. This is illustrated in figure 3, where r_{ij} is the distance between the reconstructed and the expected hit point.

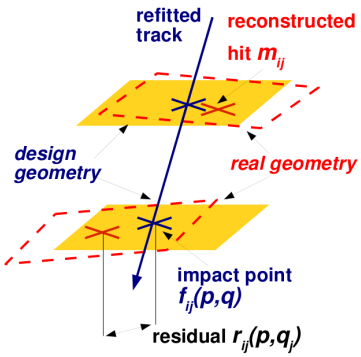

FIGURE 3. Sketch of a process involved in alignment. Shown are reconstructed and expected hit points.

As can be seen, the real geometry and the ideal design geometry differ somewhat. By shifting the coordinates of the computer internal representation of the detector this difference can be minimized. Of course the process is much more complicated in reality because the number of detector modules is high (in the order of 100000), each one characterized by three spatial coordinates and three orientation angles, and there are correlations between the shifts of the modules. For an alignment to produce sensible data the analysis of millions of tracks is required.

Currently, three algorithms are used at CMS. There is the so-called Hits and Impact method [3], which minimizes a χ^2-function constructed from the residuals mentioned above, but only locally. Since correlations are mostly not considered, this algorithm is comparatively fast and low in consumation of computing resources. The procedure is iterated, until a good level of convergence is reached. The second method in use is the Kalman filter algorithm [4], which is also an iterative method. This algorithm can take input from prior direct measurements or laser based alignment. The third one is MillePede [5], which has been previously used, for instance at H1 and CDF, and is therefore well tested. Here linear least-squares fitting is used without iterations, which requires the inversion of large matrices, but gives very precise results. Since MillePede can select non-correlated blocks from sparse matrices and invert them individually, the CPU power needed has drastically decreased.

Since CMS has not started its operation by now, the only real particles that can be used for alignment are cosmic muons. This has been intensively done, especially worth mentioning are the two CRAFT (Cosmic Run At Four Tesla) runs in 2008 and 2009 with the magnetic field at 3.8 T activated.

To design and improve the alignment algorithms instead, simulations with Monte Carlo data are used. In these simulations, which include all the detector parts and their hits by particles, the 'real' path of the particles is known and therefore the deviation between this and the reconstructed track can be accurately determined. Therefore algorithms can be tuned to minimize the shift either more precisely, in less time or with lower need of resources. More information about the alignment procedures at CMS can be found in [6].

1.3. **The Grid.** With the design and implementation of LHC arose some technical difficulties in the processing of the data:

- The LHC experiments will produce large amounts of output data, CMS alone will produce more than 3 PB per year,
- This data has to be stored and analyzed,
- There have to be at least ten times the amount of data in Monte Carlo simulations available to drastically reduce statistical fluctuations, this simulated data also has to be stored and processed.

The CERN facilities alone cannot cope with this amount of data and cannot provide the necessary means of storage and CPU power. For these reasons, the LHC Computing Grid was launched. It is a world spanning computer network, connecting all facilities participating in the LHC experiments and sharing their resources. Every facility provides mass storage systems and a computer farm that can both be globally accessed from every other research center. Of course, the distribution of jobs on available computers and the storage of data has to be managed in an efficient way. A basic concept of the Grid are the Tiers, ordered by size and importance (see figure 4). Tier-0 is the CERN facility and implements a link to the detectors, Tier-1 are all national labs and Tier-2 are regional centers (like DESY). This continues over Tier-3 (institutes) to the Tier-4 workstations.

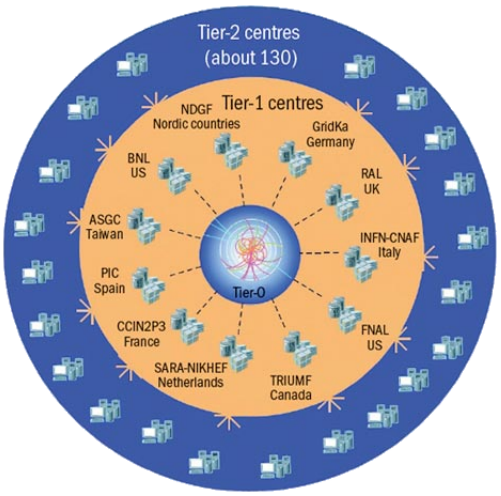

FIGURE 4. A schematic overview of the Grid structure.

The tool to access the Grid for CMS is called

CRAB (CMS Remote Analysis Builder) and is part of the CMS Software package. It provides means of locating data automatically via a database system, getting information about the datasets (e.g. the number of events), takes care of finding suitable Grid resources, transfers the job to the respective facility and processes it there. It additionally handles the transfer of the output files to the mass storage system, where it can be accessed later. This procedure is controlled via a highly adjustable configuration file. CRAB can either run a standard CMS analysis program (i.e. `cmsRun`) or execute a shell script supplied by the user that is run locally on the Grid computer.

2. Software validation

The CMS SoftWare (CMSSW) package is improved regularly. Changes in the software alter the way data is processed. To make sure that no errors are introduced and to confirm expected improvements, a standard set of data samples is processed with each software release and the results are plotted in some histograms in comparison with the results from earlier software releases.

This job is done by several persons, each responsible for a certain set of data. In my group physics processes like the creation of $t\bar{t}$ and $Z \rightarrow \mu^+\mu^-$ decay were considered. The validation was mainly done using two Python scripts: `harvestRelVal.py`, which takes a list of data sets and produces Python files on which a `cmsRun` analysis job can be run to get ROOT data files, and `ALCARECOTkAl_Validation.py`, which takes a ROOT file and produces the required histograms.

The idea was to automatize this process and to introduce the CMS Remote Analysis Builder (CRAB) to it, so that it is independent on the actual storage of the data. With the release of CMSSW 3.2.4 the directory structure within the ROOT files as well as the name of the files was changed. A change like this usually made a lot of changes for paths and histogram names in the scripts necessary. This was also compensated for in the automated Python script I worked on. It employs some navigation routines for looking for the right ROOT file and for the respective histogram inside, based on the name of the dataset considered. All parts of the program are located in small to medium sized subroutines so that the behavior of the script can be easily changed if necessary. This maybe makes it applicable for other purposes too.

When I had seen the interface for the MillePede Production System (MPS, see next section), I got the idea of building a small graphical user interface (GUI) for the release validation program too. Since Perl/Tk, the interface on which the MPS GUI is based, is in most part self-explainatory, I quickly found an approach on using Tkinter, the Python counterpart to Perl/Tk. Tk may not look very modern, but it is available on most platforms, is relieable, very easy to understand and highly configurable. Additionally, it takes very low system resources to run and even a remote X Window display runs almost smooth. This makes it the right choice for designing interfaces that probably more than one person will work on.

The user interface consists of a main window and two additional windows that can be opened by clicking the corresponding button. The main window (see figure 5) contains basic and most needed options, buttons to start the validation process, quit the program and save the options and to open the windows for more options and a non-automated validation process.

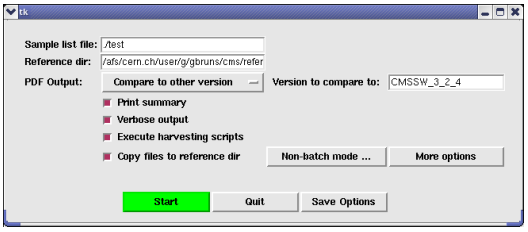

Figure 5. Main window of the validation GUI.

The user can select the button 'Non-batch mode' to open a window that grants more control over the execution steps (see figure 6). Buttons are listed from top to down in the order that the automated script would execute all of them.

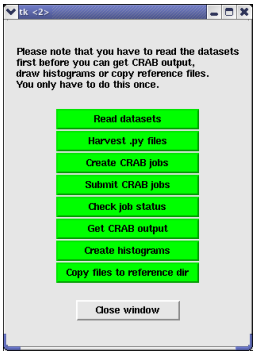

Figure 6. Execution window of the validation GUI.

When pressing sequentially all the buttons in this window, the same results as in the batch queue should be achieved. However, here the user can repeat a process with different settings or account for errors that have happened while processing.

Another window (figure 7) that can be opened contains additional configuration options that are not needed all the time. All these options are stored in an easy to handle configuration file, which can also be manually edited with every text editor. Since options are read and stored in a syntax-based manner, the program does not itself care about the names of the options it reads. Therefore, the implementation of additional options is quite easy and virtually only the GUI design has to be changed.

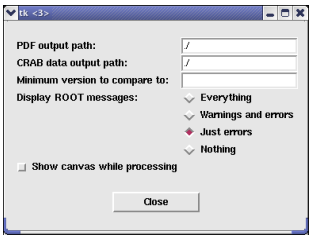

FIGURE 7. Option window of the validation GUI.

The results of a comparison from version 3.3.0.pre1 to 3.2.5 are shown in figure 8.

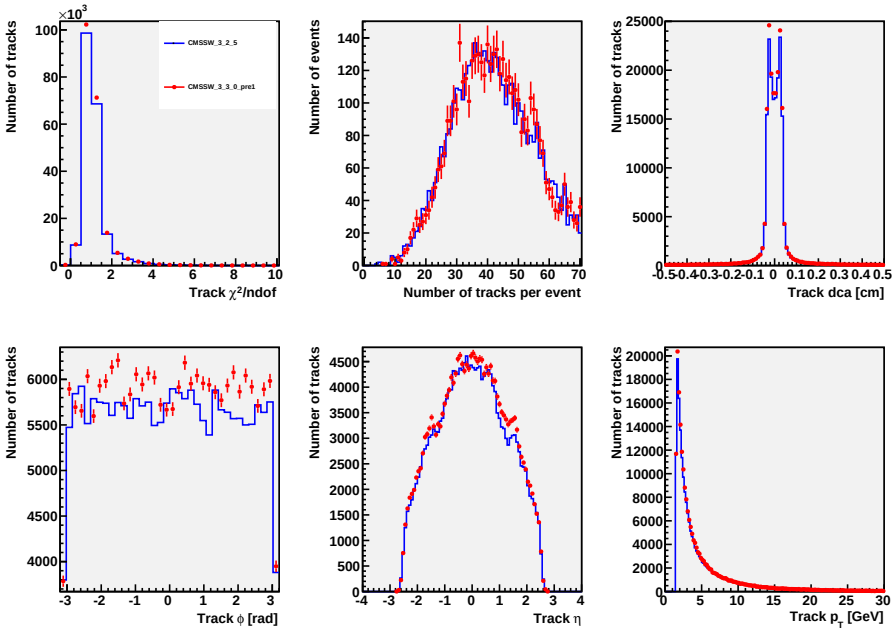

FIGURE 8. Comparison of some histograms from data created by CMSSW versions 3.3.0.pre1 und 3.2.5.

As can be seen, the general shape of the plots looks the same. However, there is a slight increase in the number of tracks observed in all histograms, especially in the lower left one, where a distinct elevation of the mean value can be seen. If something like this happens, it is reported to the software developers so that they can judge if this was intended (maybe they changed something in the cutoff routines) or if this was unexpected.

3. MillePede Production System

The MillePede algorithm is one of the algorithms used for track based alignment in CMS. It is performed in two steps: first there is the Mille step, where the input tracks are processed in order to produce the input for the alignment, followed by the Pede step that performs the actual tracker alignment. The configuration and setup is done via a `cmsRun` input file, where parts of the

detector can be fixed, artificial misalignments be introduced and so on.

To provide an easy-to-handle interface for the alignment procedure with MillePede, there exists the MillePede Production System (MPS) [7]. This interface consists of a set of Perl scripts that carry out various jobs like setup of the working directory, splitting jobs, submitting jobs to the batch queue and so on. Splitting jobs is possible at the Mille step, because tracks are processed independently from each other, and necessary because an alignment procedure usually consists of several millions of events and therefore the run time is way too high when running without parallelizing.

A general MillePede alignment run involves the setup of the jobs, submitting the splitted Mille jobs to a computing queue, checking the results, submitting the Pede merge job and reading the output data. It was implemented using the batch job queues locally available at CERN for computing. More details on the scripts can be found at the MPS web page.

To make things easier, a graphical user interface, written in Perl/Tk, which is called MPedeGUI, is available. This GUI was part of the project by last years summer student supervised by Andrea Parenti and Silvia Miglioranzi. This GUI (see figure 9) links several buttons to the MPS script calls and the entry fields to the relevant arguments. Modifying the scripts did therefore not pose any problem for the GUI, as changes have to be done only in the called subroutines.

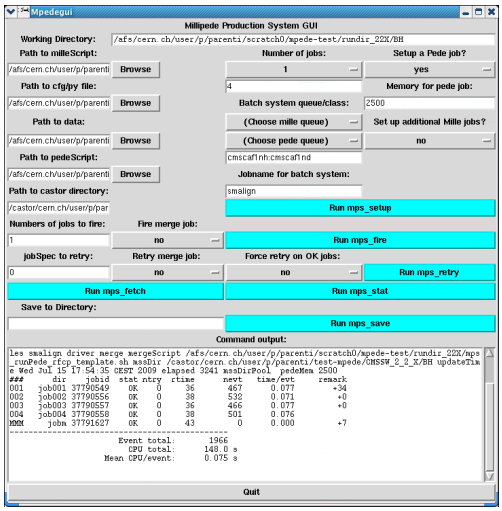

FIGURE 9. The main window of the MPS.

Since alignment will still be an important issue when the LHC is running and there will be a lot more alignment data to be processed than now, the implementation of Grid use is an important aspect. In principle, CRAB can take over some part of the job MPS used to do, including fetching the data and splitting the Mille jobs. Since this implies important design changes in the MPS, for instance a change of the directory structure, this has to be done carefully in the future.

However, the applicability of CRAB to submit Mille and Pede jobs was tested by me. Some adjustments had to be made in the run scripts, a fitting CRAB configuration file had to be created and the management of the storaged output files from the split jobs had to be considered. After all, this proved succesfull and I obtained the same results with CRAB as with MPS in doing a simple alignment job. For this, the local CERN queues (LSF and CAF) were used. Other Tier-2 facilities, to which users can submit jobs, did not carry the requested input files, so no test could be done there. However, in the future with the load of CMS data this will probably be changed and then the CRAB configuration can easily be adjusted.

It should be mentioned that some features the batch queue offered cannot be used anymore when working with CRAB. For instance, a report on the CPU time already consumed is not available anymore.

Finally, I started implementing MPS with CRAB, without using the CRAB features of data collecting and job splitting. Because this only involves changes in command processing and input/output parsing, this can be done without changing the design of MPS. The results of this work will be submitted to Andrea Parenti together with a description of changes and things still to be done, so that this implementation step can be continued.

ACKNOWLEDGEMENT

I would like to thank my supervisors, Andrea Parenti and Justyna Tomaszewska, for their observation of my work, their instructions and tips and for the time they spent explaining things and helping me out. I also would like to thank Gero Flucke for giving helpful comments and explanations, too. Last but not least I would like to thank all people that have made this stay at DESY possible, especially Joachim Meyer, Andrea Schrader and the lecturers.

References

[1] L. Evans and P. Bryant (editors). LHC machine. *JINST 3 S08001*, 2008.

[2] S. Chatrchyan et al. (CMS Collaboration). The CMS experiment at the CERN LHC. *JINST 3, S08004*, 2008.

[3] V. Karimaki et al. *CMS-NOTE 018*, 2006.

[4] W. Adam et al. *CMS-NOTE 022*, 2006.

[5] G. Flucke et al. CMS Silicon tracker alignment strategy with the MillePede II algorithm. *JINST 3, P09002*, 2008.

[6] The CMS Collaboration. Alignment of the CMS Silicon Tracker During Commissioning with Cosmic Ray Particles. *in preparation*, 2009.

[7] R. Mankel and A. Parenti. The MillePede Production System (MPS). https://twiki.cern.ch/twiki/ /bin/view/CMS/SWGuideMillePedeProductionSystem.