CMS Analysis Deconstructed

Journal of Physics: Conference Series

S Malik 1
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

E-mail: malik@fnal.gov

Abstract. The CMS Analysis Tools model has now been used robustly in a plethora of physics papers. This model is examined to investigate successes and failures as seen by the analysts of recent papers.

1. Introduction

After a brief technical stop the CMS experiment [1] started data taking again in April 2012 at a higher proton-proton total collision energy of 8 TeV and a higher instantaneous luminosity of \( \sim 10^{33} \text{ cm}^{-2} \text{ s}^{-1} \). During the 2011 run, a 5 fb\(^{-1}\) data was collected at the total collision energy of 7 TeV leading to 120 publications. The CMS computing structure and a flexible data model, designed to deal with Petabytes of data per year and a complex detector, have played a key role in this success. This year we expect to collect up to a 50 fb\(^{-1}\) with many more physics analysis, publications and possible discoveries. In order to cope up with distributing a large amount of data and efficiently reducing the time-span between its reconstruction and analyses, the computing model has been coherently organised into four layers or Tiers. The Tier-0, located at CERN, performs primary reconstruction of the data using the first calibration constants and the jet-energy corrections. This data is then distributed to the Tier-1 centres in seven countries around the globe via the worldwide Grid [2] infrastructure. The Tier-1 centres share the storage of raw data and perform data reprocessing. Data selections are applied to skim the data to manageable sizes and distribute it via the Grid to the Tier-2 centres. Tier-2 centres are dedicated to physics group data storage, production of simulated data and providing resources for user analysis. Tier-3 centres serve entirely as a local institutional user resource. The distributed computing structure is accompanied by a very efficient data model [3], which provides the data in different formats, each reflecting the amount of the reconstructed event information. Examples of collision data tiers include RAW (raw data) and RECO (reconstructed data). The RAW data is the output of HLT (High Level Trigger) with the event size of 1.5 MB. The most important data tiers from a physics analysis point of view are RECO and AOD (Analysis Object Data). RECO contains all reconstructed objects and hits with typical event size of 500 kB/event while AOD is a smaller subset of RECO with typical event size of 140 kB/event. CMS uses the Grid to provide data access to all the collaborators. The CRAB (CMS Remote Analysis Builder) [4] provides a user front-end to enable submission of physics analysis jobs to all the CMS datasets within the data locations driven by the distributed computing structure. In this paper we focus on the analysis model used by CMS in terms of its boundary

1 malik@fnal.gov
requirements, official analysis tools, analysis workflow from a users perspective and feedback from a survey of current analysis practices.

2. Boundary Requirements
CMS has set some boundary requirements to guide the physics analysis structure, given the geographical spread of the collaboration, high cost of the program and the potential of major discoveries and breakthroughs. It is very important that the physics results be of a very high quality and largely reproducible, not only by the other collaboration members but also by other competing experiments as well, long after they are published. The CMS analysis model has the above goal programmed in its structure where datasets [5] and skims [5] of reconstructed data are selected on the basis of persistent information and retention of provenance [5] information. Provenance means that the information about all the parameter values used in the job that created a data file are recorded in the file itself. For example, a provenance record would show that a file A.root was made by running cmsRun application on file B.root and configuration file C.py. Provenance record might also show which filters were applied to choose the stored events, e.g. filter trackfilter (events with at least two tracks with p, of 5 GeV/c) had to pass an event before it could be stored. The model is centred around the idea of easy access to the data samples, processed in a user friendly and easy to use format to quickly produce the physics results using the array of available physics analysis tools. While most of the tools are officially approved, the analysis model has enough flexibility to support those analyses that require special datasets and tools. The software platform used to produce the physics results is Scientific Linux on which CMS develops and runs its validation code. The Physics Objects and Algorithms that are used in skimming (the process of selecting events and saving them in output) the data are approved by the corresponding Physics Object Groups (POGs). The origin of the samples must be fully traceable by the provenance information and the analysis code fully reviewable by Physics Analysis Groups (PAGs). The PAGs consist of experts providing leadership in physics within the experiment.

3. Data organisation and analysis workflow
To extract a physics message for a high energy physics analysis, a user needs to combine a variety of information: reconstructed information from the recorded detector data, specified by a combination of trigger paths [5] and possibly further selected by cuts on reconstructed quantities (e.g., two jets [5]), Monte Carlo (MC) samples which simulate the physics signal under investigation, and background samples (specified by the simulated physics process). The physics abstractions used to request these items are called datasets [5] and event collections. An event collection is the smallest unit within a dataset that a user can select. Typically, the reconstructed information needed for the analysis would all be contained in one or a few event collections. The expectation is that a single primary dataset can be used to perform majority of analyses. Data are stored as ROOT files [6]. The smallest unit in computing space is the file block, which corresponds to a group of ROOT files likely to be accessed together. This requires a mapping from the physics abstraction (event collection) to the file location. CMS has a global data catalogue called the Dataset Aggregation Service (DAS), which provides mapping between the physics abstraction (dataset or event collection), and the list of file blocks corresponding to this abstraction. It also gives the user an overview of what is available for analysis, as it has the complete catalogue.

The primary dataset can be refined to a smaller subset called Central skim based on trigger and reconstruction information geared towards a physics case. The Central Skims are provided in the AOD data tier at the Tier-2 centres. Users have a direct access to the data only at Tier-2 where they perform high-level analysis steps. This includes preselecting events and writing user defined event content to a reduced dataset that can be copied to a private space at Tier-2/3. This entails the use of Grid tools to submit analysis jobs. These steps can be reiterated. Thereafter, one can run final analysis and make plots.
4. Analysis Tools

The full suite of CMS software is called CMSSW. It draws on experience from other HEP experiments like Babar, CDF, D0 and CLEO and is designed to meet the goals of accessing all data via the Event, allow modular testing of the software, track provenance and browse-able ROOT files. It is centred on the concept of an Event, called the Event Data Model [3]. In this model an event starts as a collection of the RAW data from a detector or MC event. As the event data is processed by series of algorithms (e.g., track fitter, track finder), products are stored in the event as reconstructed (RECO) data objects. Event contains the triggered physics event, the derived data and the metadata. Metadata comprises of information like software version, detector conditions and calibration data. The CMSSW consists of over two thousand packages and can deal with all the tasks at once - online data taking, simulation, primary reconstruction, and physics analysis. It uses C++ as the coding language and is steered via configuration files written in Python language. CMSSW code resides in code repository with open access via web interface. One can search details of a software package using the cross-reference browsing tool [8]. Doxygen [9] is the tool used to generate and maintain reference manual for data formats in CMS software. With this tool one can generate the documentation from the software code for each release version of CMSSW software.

Each event data products in the ROOT file are stored as independent TBranches within one TTree, the Events TTree. The event data products for one event sit at the same entry for all TBranches in the Events TTree. This structure also has an in-built provenance. The data files can be opened and inspected using a TBrowser giving access to members of the stored objects. TTree objects are used as a standard data structure in almost all HEP experiments. However, CMS specific dictionaries are needed by the member functions in ROOT to read the data in TTree objects. The TTree objects are connected via smart pointers to optimize memory space usage leading to a strong hierarchy. This kind of structure gives flexibility in the reconstruction but it is not optimized for usability. For example, “ECAL sub-cluster” information for Electron object is completely independent of its “Track hit” information. One can access “ECAL sub-cluster” information for electron only via “ECAL super-cluster” and not directly. Figure 1 shows this hierarchy for a Physics Object called Electron.

![Figure 1. The hierarchical TTree structure of Electron object](image-url)
To ease this situation for the ultimate goal of physics analysis, a special software layer called PAT (Physics Analysis Toolkit) [10] was developed where one can configure data formats using python scripts. It combines flexibility with user friendliness and maximum configurability, and provides easy interface to complicated set of algorithms approved by physics object and analysis groups. Intricate and complex tasks like generator level and trigger matching of objects, applying jet energy corrections, and cleaning of overlapping objects can be performed with a rich set of tools and information stored along with already existing data. Combined with Particle Flow [11] technique PAT can give a full interpretation of an event. Using PAT, one sticks to the provenance and uses all official tools. One powerful feature of PAT is to trim the event size to the level needed for a particular analysis by slimming event information or simply dropping objects from a collection. The following snippet is an example of selecting a jet object in PAT with $\text{et} > 40 \text{ GeV}$

```python
selectedPatJets = cms.EDFilter("PATJetSelector",
    src = cms.InputTag("patJets"),
    cut = cms.string("et > 40")
)
```

The job configuration (execution of software modules) system of the CMSSW is based on the Python language. A typical physics analysis uses a large number of configuration files. Therefore, it is essential to have a visualization tool to investigate and verify configuration parameters and the dependencies in the modular software. To ease inspecting and editing a python configuration file, CMS offers a tool called configuration browser [12]. One can not only learn how a configuration file is nested and structured but also look at the modules used, how they are related to each other and the set values of the parameters. It can also be used for the documentation of configuration files. Figure 2 gives a snapshot of the configuration browser.

![Figure 2. A snapshot of the configuration browser](image)
The final stage of an analysis is usually very interactive, does not require access to conditions, and is based on skinned, stripped (and possibly thinned) data samples. With interactive analysis on a laptop in mind rather than using the full EDM Framework, its lighter interactive version called FWLite (FrameWorkLite) is designed. FWLite is plain ROOT with data format libraries and dictionaries that provide the capability to read CMS data together with some basic helper classes. It allows the data classes to be used directly in ROOT. A compiled code using the FWLite libraries can be setup in such a way that the code between FWLite and the full CMS software framework is fully inter-changeable.

A 3D-accurate display to visualize proton-proton collisions is very essential to understand the physics outcome of the event. The CMS event display, called Fireworks [13] (also called cmsShow), is specially developed for this purpose. The core structure of Fireworks is built on top of the Event Data Model (EDM), the EVE subsystem [14] of the CERN ROOT project and the light version of the software framework (FWLite). The collision data is presented via a user-friendly graphical interface and textual views. The Fireworks software is easy to install and use. Data handling is greatly simplified by using only the reconstructed information and the ideal geometry. This allows a physicist to concentrate only on the data of interest.

5. Organising analysis workflow

Analysis workflow is an important and difficult decision. A typical physics analysis workflow for a user broadly involves the steps of comprehension, development and execution. The user needs the standard skims which are defined and created by the physics groups, the standard physics objects (reconstruction, identification, cleaning) and official analysis tools. A successful analysis involves selecting events, understanding the data and corrections, convincing others about the results, writing notes and papers. This implies that the official tools should be followed to as much extent as possible. One can analyse the data either by directly using the RECO/AOD objects or by producing and using the PAT objects.

Using the RECO/AOD objects directly not only needs expert knowledge of all available features but also a very up-to-date knowledge of POGs. This is very time consuming and an impractical path to follow.

For a non-expert, PAT provides the software analysis environment that not only aides in data analysis but at the same time incorporates all the official tools and features provided by the POGs and PAGs. One gets the latest algorithms from the same interface: the PAT objects. The solutions provided to aid analysis are fast and efficient as any self-made solution (an obviously very difficult task) and in addition are CMS compliant. Using PAT makes analysis easy to be approved by the collaboration.

There are several options used by CMS users to store the final dataset for their analysis:

- RECO/AOD - has full flexibility but needs lots of space and expert level knowledge to analyse
- PATtuple - full flexibility and saves space
- EDMtuple - not flexible, need to know exact set of objects, variables, provenance information is kept
- TNtuple – gives up the provenance information and not officially supported

While the user has the freedom to choose any of the four options listed above, it is very important to emphasize here that there is a decreased space usage, increased speed in terms of analyzing and decreased flexibility, as one goes from RECO/AOD to TNtuple.
Using RECO/AOD keeps provenance and the internal references within a data file makes the use flexible but the space consumption is more (for example: keeping all info: Electron -> Cluster -> Track -> Hits -> RecHits etc.). It is very complex for a user to keep track of references.

When working with PAT tuples the biggest challenge an analyst faces is that they want all relevant information from RECO/AOD stored in an intuitive and compressed way. However, RECO/AOD can only be skimmed by keeping or dropping complete branches and not by selecting objects from branches. PAT has a solution for this in the form of a feature called embedding [15] that saves space usage and allows one to keep only the relevant information. For example, Jets contain reference to the Calorimetric Towers (CaloTowers) they were constructed from. Instead of keeping the entire collection of CaloTowers one can embed and keep only those specific ones that are associated with the Jet. With PAT one can access the relevant information for each physics object from a single interface called PAT objects (C++ classes). PAT is fully compatible with FWLite and is not a data tier.

TNtuple is the fastest way to store and plot data with built-in types or simple C++ structs. These are usable by everyone; however, one needs to know names and types. Moreover, it is quite clumsy to use 4-vectors and their structure is not preserved. It also lacks EDM provenance information that is needed for a reproducible analyses e.g. for the proper calculation of Luminosity. There is a lack of common format among different analyses. It also lacks integration with Grid tools. Some TNtuples have private classes and nicer structures but many are complex to setup, not easily portable and, of course, lack provenance.

The tool developers came up with an alternative to TNtuple use by providing what is called an EDMtuple that stores a set of user-defined vector of doubles in the EDM format. It is as fast as TNtuple and at the same time keeps the provenance information. Some examples of usage can found in [16].

Given the above choices to access and store the CMS data while performing physics analysis, some typical analysis workflows are shown in Figure 3. In the development phase of an analysis one can keep all relevant information in a PAT tuple in order to be fully flexible for object and algorithm changes and do the final analysis in EDMtuple and make plots.

---

**Figure 3.** Typical analysis workflows using different choices to access and store the data.
6. Survey of analysis practices
The CMS Physics Support Group conducted a survey of analysis practices followed by CMS. A questionnaire was prepared to record information about the analysis workflow. The participants were asked to describe all the intermediate steps, the corresponding data tiers and the number of steps (tiers) beginning with the primary data set up to their final analysis plots. A total of 20 analysis participated from all CMS Physics Analysis Groups – 1 from B-physics, 6 from Exotica, 3 from Higgs, 2 from SUSY, 3 from Top physics and 5 from Electroweak. The questions specifically asked about the name of the data tier, its format, provenance, the tools used to produce it and if the subsequent data tier had a compact data format. It further asked if the software was tagged and its location in CMSSW code management repository (CVS), usage of PAT, and if, besides the analysis note, the technical details are documented and their location. We briefly discuss below some of the important result of this survey.

Each of the figures below has the step number (Step1, Step2 …etc.) on the horizontal axis. The results are shown for each step. Out of the 20 analyses, each analysis sometimes had multiple answers for a given step. Hence number of entries for step could be more than 20. Figure 4 shows the data formats used at each of the step of the analysis. It shows that most of the analysers used either RECO/AOD or PAT or both at Step1. Also many used Ntuples with private classes. At Step3 and Step4, as expected, most people are plotting histograms for their analyses. Figure 5 shows that a significant fraction of analysers used PAT. While most used PAT as a configuration tool to make private ntuples, others used PAT objects. Figure 6 indicates that most users submitted CRAB jobs on the Grid for Step1 and Step2. This is expected as one runs initially on a big dataset stored at Tier-2 centres and skims it down and stores on a local cluster and then work from there for the final stages of their analyses.

Figure 7 shows that almost no one used FWLite and a very few used official EDMtuplizer. Almost all analysers used official tools and PAT in first two steps of their analyses.

The survey shows that tools and software used for the analyses were stored mostly in the CVS. Using PAT as the main tuple turned out to be the best choice in several occasions. Only two analysers indicated that what they did was optimal and therefore would not change anything for their next analysis. Of 35 responses, 2 said their analysis workflow is not documented. The rest documented it mostly on twikis, some of which are hosted at their home institution instead of centrally at CERN.
7. Conclusion

CMS has a very flexible computing and data analysis model keeping in mind the long-term reproducibility of the analyses as well as comparison with other analyses within and outside CMS. It has provided guidelines and several official tools to the users so that their analysis stays CMS compliant. The survey indicates that most users tend to follow official tools and provenance as much as possible. They use Grid as a first step to reduce their dataset to a smaller skim and then analyse results on local clusters. A clear tendency to store analysis code in the central CVS repository and
document the steps is indicated. Given that CMS is a big mass of collaborators, the trend towards growing use of official practices is encouraging. It also shows that the analysis tools (especially PAT) are suited to the need of almost everyone.

8. References

[1] The CMS Collaboration, “The Compact Muon Solenoid Technical Proposal”, CERN/LHCC 94-38, 1994
[2] http://lcg.web.cern.ch/LCG/
[3] Elmer P et al 2010 J. Phys.: Conf. Ser. 219 032022
[4] Codispoti G et al 2009 IEEE TRANS NUCL SCI , vol. 56, no. 5, pp. 2850-2858
[5] https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookGlossary
[6] http://root.cern.ch/root/
[7] http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/
[8] http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/
[9] Doxygen is a documentation generator for the programming languages. More info on http://www.doxygen.org
[10] Hegner B 2011 J. Phys.: Conf. Ser. 331 032014
[11] http://arxiv.org/pdf/0810.3686v1.pdf
[12] Erdmann M 2010 J. Phys.: Conf. Ser 219 042008
[13] Kovalskyi D 2010 J. Phys.: Conf. Ser. 219 032014
[14] Tadel M 2010 J. Phys.: Conf. Ser 219 042055
[15] Adam W et al 2010 J. Phys.: Conf. Ser 219 032017
[16] https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideEDMNtuples