The architecture and operation of the CMS Tier-0

Dirk Hufnagel, for CMS Offline and Computing

E-mail: `Dirk.Hufnagel@cern.ch`

Abstract. The Tier-0 processing system is the initial stage of the multi-tiered computing system of CMS. It takes care of the first processing steps of data at the LHC at CERN. The automated workflows running in the Tier-0 contain both low-latency processing chains for time-critical applications and bulk chains to archive the recorded data offsite the host laboratory. It is a mix between an online and offline system, because the data the CMS DAQ writes out initially is of a temporary nature. Most of the complexity in the design of this system comes from this unique combination of online and offline use cases and dependencies. In this talk, we want to present the software design of the CMS Tier-0 system and present an analysis of the 24/7 operation of the system in the 2009/2010 data taking periods.

1. Introduction
The CMS (Compact Muon Solenoid) experiment is a general purpose particle physics detector at the LHC (Large Hadron Collider) at CERN. It is located in an underground cavern at LHC P5 (Point 5) in Cessy, France. When the experiment is running and collecting data, that data is first written to a large disk buffer at P5. In a secondary step, the data is then transfered to CERN, where it is further processed. When we talk about the CMS Tier-0, what we mean is all the immediate and automatic data handling and processing at CERN.

2. Dataflow
Figure 1 shows a summary of the dataflow for CMS data at CERN, starting from P5, showing the major Tier-0 tasks and PromptCalibration loops, feeding back conditions to the Tier-0 processing. In addition to the Tier-0, which includes the automatic data handling and processing, there is also the CAF (Calibration and Alignment Facility / CERN Analysis facility), which includes manual data handling and processing tasks. The Prompt Calibration workflows that are run manually will be run on the CAF.

3. Before data gets to the Tier-0
Before the Tier-0 there is the data acquisiton system, the High Level Trigger (HLT) and the StorageManager (system which writes data to the disk buffer) at P5. What is written to disk at P5 is the input to the Tier-0 and its structure determines to some extent how we have to handle it. The HLT output is separared into online streams, for proton-proton collisions they are

- Physics stream, about 200Hz
- Express stream, about 20Hz, a subset of Physics for fast monitoring/analysis and feedback
- Various calibration/monitoring streams, about 20Hz total

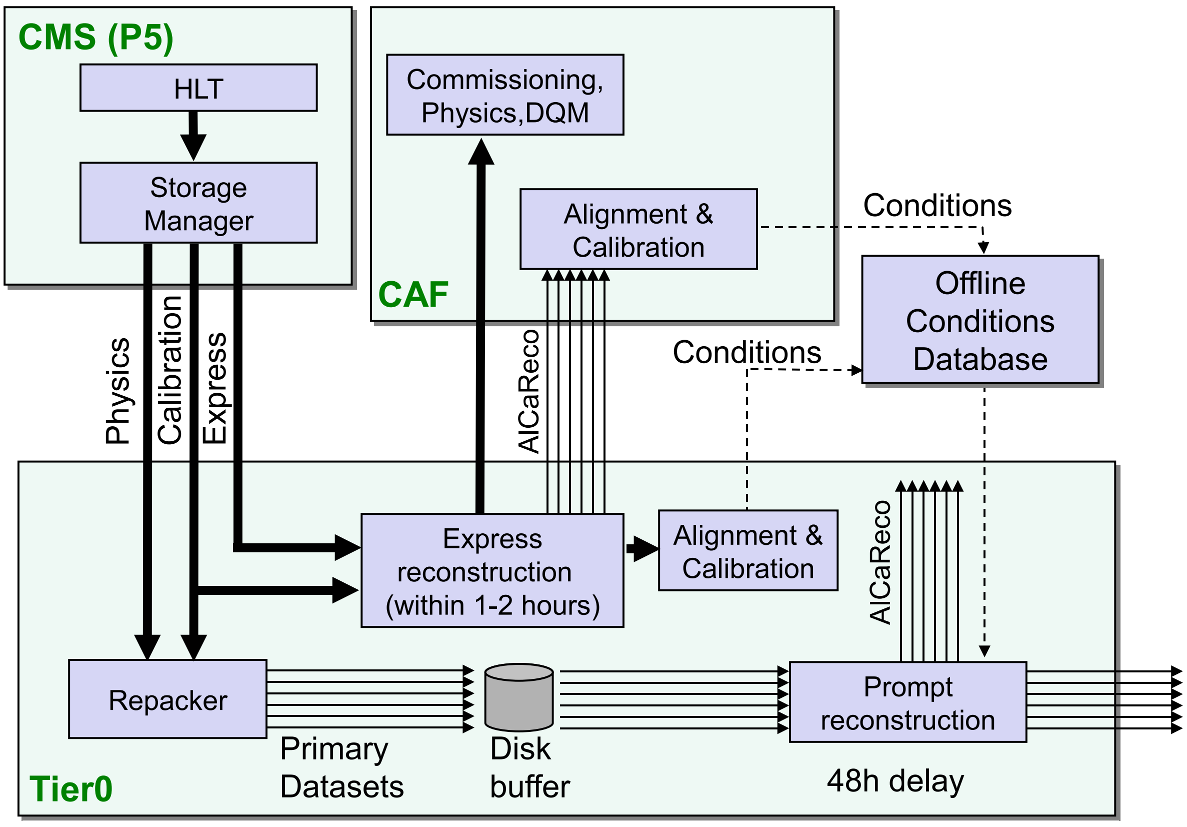

Figure 1. CMS Dataflow Summary

These numbers are not stable during data taking, the instantaneous rates at any given time can look quite different. But integrating over time and taking duty cycle into account we are trying to get the average close to these numbers.

Within an Online stream there is no sorting of events by physics/trigger classification. Events are handed from the HLT to the StorageManager and written to disk in a special streaming format (streamer files). There are multiple StorageManager instances to optimize the write rate to disk, which causes data taken within very small time windows to be split across multiple streamer files.

4. Tier-0 requirements
Before we talk about implementation, lets look at the functional requirements for the Tier-0.

(i) Repacking
First, the special data format of the streamer files is not software release independent. This means that there is no guaranteed forward or backward compatibility for streamer files, to first order they need to be read with the software release they were written. This makes them unsuitable for custodial long-term storage. Second, luminosity for CMS data can only be calculated for clearly defined short data segments (called a lumi section). At the moment this is set to about 23s. Due to the way the StorageManager writes out the data at P5, a lumi section is split across multiple streamer files. Having a lumi section split across multiple files leads to complications in the data handling as it introduces dependencies between files. And third, the data in the streamer files is not sorted by physics/trigger classification. This

leads to 3 requierements, which all will be addressed in a single workflow, the repacking.

- Convert streamer files into ROOT-based custodial data format
- Assemble lumi sections into files that only contain complete lumi sections
- Split data into subsamples according to trigger classification of events

(ii) PromptReconstruction
Reconstruct the data after a 1 to 2 day wait for updated conditions from the PromptCalibration. Once started, finish the reconstruction within 24 hours.

(iii) ExpressProcessing
Provide a fully reconstructed subset of the data after 1 hour for fast monitoring/analysis and feedback. The first stage of this is already done at the HLT level, writing a separate Express online stream, which is the input for the ExpressProcessing. The PromptCalibration loops that calculate conditions for PromptReconstruction also run on output of the express processing.

(iv) PromptCalibration
There are two PromptCalibration loops, one fully in the Tier-0, the other via the CAF. For the latter, output from the express processing is made available on the CAF, where Alignment and Calibration workflows can then be run manually or semi-automated. In constrast, the PromptCalib loop within the Tier-0 has to work automatically, without any manual steps.

5. Tier-0 general architecture

The software system that manages the CMS Tier-0 uses an Oracle database as its backend. The database backend is called T0AST (Tier-0 Activity State Tracker) and it keeps the current system state, all files and all jobs and associated metadata.

The software is written in python and leverages the ProdAgent architecture (CMS MC Production System). It consists of many individual components, each component having a well defined task. The components do not interact directly, but by changing state in T0AST. They trigger an action based on a certain input state, then change the state to reflect the action that was taken.

State consistency is guaranteed by database transactions. The action taken by the component and the state change are part of the same transaction. If the component fails to take an action, the state is not changed. If the state change fails, the action is not taken.

6. Tier-0 description

6.1. Merging

In order to optimize tape migration of data, we want to produce files that are reasonably large (about 2GB to 4GB). Sometimes this is not possible and we'll create (much) smaller files. Instead of keeping these directly then, they'll have to go through a merge step to produce larger files. So any processing workflow that follows has an optional merge workflow attached to it, which is triggered if the output files of the processing step are too small.

6.2. Repacking

Repacking is at its core a very simple workflow. Each job reads multiple streamer files (representing complete lumi sections) and writes out RAW files, splitting events into output files representing primary (Physics) datasets (Jets, Muons, etc). The complexity here is not in what the jobs do, but in configuring them correctly.

For the repacking configuration, the Tier-0 reads the HLT configuration from ConfDB, an Oracle database containing the authoritative copy of each HLT configuration. Each HLT configuration contains a list of online streams, the primary datasets for each online stream and the trigger paths for each primary dataset. Repacking is done separately for each online stream. Within each online stream, repacking jobs are scheduled to read one or more (depending on data volume in the online stream) complete lumi sections and write out one file for each primary dataset in the online stream. The filter criterion for sorting events to primary datasets are the mappings of primary dataset to HLT trigger paths as defined in the HLT configuration. It is matched against the list of HLT trigger paths passed by an event, which is available from within the event.

Job scheduling tries to take into account online stream data rates/sizes in order to write out reasonably large files that can be written directly to tape. This works very well for primary datasets that contain a high fraction of the events in the corresponding online stream. With a higher number of primary datasets per online stream and consequently primary datasets that only contain a small fraction of the events in the online stream, this eventually fails. If the repacking output is too small, an additional merge step is performed after the repacking.

At the other end of the spectrum, under normal conditions repacking can not schedule a job to process less than one complete lumi section. If the data volume in one lumi section is very large, this would lead to very large output files from the repacking, which can cause WAN transfer and other data handling problems. The Tier-0 will treat too much data in a lumi section as an error condition. To protect itself and to make the data available for debugging it will work around it, creating multiple files per lumi section. That data is considered invalid though and is routed to special error primary datasets. The protections are fully configurable (leaving the definition of what “too much data” means in the hands of Tier-0 operations). There are two separate mechanisms that come into play. First, if the sum of the sizes of the streamer files in a lumi section is larger than a configurable threshold, repack jobs processing fewer streamer files than are in the complete lumi section are scheduled. Second, repack jobs are configured to break output files if they go above a configurable threshold. Unfortunately, the second feature breaks all outputs if one of them goes above threshold, even the ones that have written very little data. Both features can be combined or activated independently based on how one configures the relevant threshold. No matter how complex, the combination of repack splitting on input and/or output and the merge step following the repacking tries to make sure that only primary datasets that really need it are split into multiple files per lumi section, everything else will be recombined in the merge.

6.3. PromptReconstruction

PromptReconstruction is performed on the RAW output of the repacking. It can be held for up to 48 hours to wait for new conditions from PromptCalibration, but this is a configurable setting. We are, at present, commissioning the PromptCalibration loop, for most of the time so far we have run without a delay for PromptReconstruction.

RAW files can be large and contain many events. Reconstructing them in one single job could potentially take a very long time. To optimize job performance and overall system latency, we can split up the reconstruction of single RAW files into multiple jobs. This can result in a resplit of lumi sections, but lumi section are recombined in a merge step performed after the PromptReconstruction.

PromptReconstruction writes multiple outputs. RECO files containing reconstructed objects, DQM files containing data quality monitoring information (like histograms), AOD files containing a subset of RECO (for physics analysis) and ALCARECO files containing input for Alignment and Calibration workflows.

6.4. ExpressProcessing

The design goal for ExpressProcessing is to provide a fully reconstruced subset of the data in 1 hour. In principle, just looking at the workflows, one could take the Express stream, repack and prompt reconstruct it and then have the output that is needed. Unfortunately, looking at the various latencies, job runtimes etc, it becomes clear quickly that with this approach the design goal of 1 hour is not achievable.

In order to reach the 1 hour design goal a completely different processing chain starting with streamer files and producing RAW, RECO etc output was designed. ExpressProcessing always works in two stages. The first is a processing part that is parallelizable even within a lumi section and schedules reconstruction jobs to run directly on one or more streamer files. The second part is a merge/split step which reads the output of the first step, combining complete lumi sections, and writes RAW, RECO, FEVT (RAW+RECO), DQM and ALCARECO output files. DQM and ALCARECO output is treated in a special way in the processing part, writing one combined output file, not splitting into primary datasets. For the other output this is a configuration option, the split into primary datasets can either happen in the processing part or the merge/split part. The way we have been operating the Tier-0 so far, we always split into primary datasets in the processing part. For technical reasons and due to the Express stream only having one primary dataset this leads to a lower latency for the whole express processing chain.

The combined ALCARECO output from the processing part goes through a special merge/split, which is called an AlcaSkim. We still combine complete lumi sections, but we do not split into primary datasets. Instead we split into samples requested by the various Alignment and Calibration groups. For instance, for alignment one needs tracks, so we produce a ALCARECO sample with just reconstructed track information.

6.5. PromptCalibration

The PromptCalib loop inside the Tier-0 is fully automated and uses the ALCARECO from the ExpressProcessing as its input. For that reason it is implemented as part of and extension of the ExpressProcessing. We designed again for low latency because operationally, the longer we have to delay PromptReconstruction, the higher our load on the Tier-0 resources.

Any PromptCalibration algorithm can potentially be split into 3 parts. First a part that can be run event by event, like selecting the appropriate input for the algorithm. This part is already run during the processing part of ExpressProcessing, writing out a combined ALCARECO file. Then a second part, which uses complete lumi section or multiple complete lumi section as input. We decided to add this to the AlcaSkim part of ExpressProcessing because we have complete lumi section of combined ALCARECO available in the input there. The PromptCalibration part of the AlcaSkim job will write a special ALCAPROMPT output file, containing the results of the per lumi section algorithm plus any potential per event data needed for the next step. Lastly, an AlcaHarvesting step which runs after ExpressProcessing for a run is complete and processes all ALCAPROMPT files for the run in a single job.

In a best case scenario, all the expensive calculations for a PromptCalibration algorithm would be per lumi section and therefore be split across many jobs that are scheduled as the run is ongoing. The AlcaHarvest job then would just be a cheap aggregation and condition creator, writing a sqlite output file. A perfect example and the first PromptCalib algorithm that was implemented (and so far also the only one) is the beamspot determination. There we do a per lumi section fit of the beamspot in the AlcaSkim job and then collapses IOV (interval of validity) in the AlcaHarvest job because the beamspot does not change often and we want to avoid a new condition payload per lumi section.

Type	Size	Events
Repack(RAW)	511TB	14.2B
PromptReco(RECO)	445TB	4.2B
Express(FEVT)	175M	45TB

Table 1. Output in 2010

Type	Size	Events
Repack(RAW)	90TB	4.1B
PromptReco(RECO)	99TB	505M
Express(FEVT)	15M	7TB

Table 2. Output in Run2010B period

7. Tier-0 Operational Experiences
All the parts of the Tier-0 described here are either deployed or in the process of being deployed.

- Repacking: deployed since summer 2008
- PromptReconstruction: deployed since fall 2008
- ExpressProcessing: deployed since summer 2009
- PromptCalibration: currently being commissioned

Table 1 shows the output data produced by the three major Tier-0 subsystem, Repacking, PromptReconstruction and ExpressProcessing, during the whole 2010 year up until October 12. Table 2 shows the same numbers just for the time period from after the LHC September technical stop to October 12. Table 3 shows the total number of jobs in the same time period and the number of failures, not taking into account jobs that succeeded on automatic or manual retry.

Table 3. Job count and failure rates in Run2010B period

Type	Total Jobs	Failed Jobs
Repacking	26544	0
PromptReconstruction	101896	14
ExpressProcessing(reco)	229891	2
ExpressProcessing(merge/split)	144458	0
Merging	42157	0

8. Summary
The CMS Tier-0 has been very reliable and worked well even beyond design specs. It has performed well for CMS, with relatively small operational effort. With the commissioning of the PromptCalibration loops, it will become feature complete soon.

References
[1] CMS Collaboration, The Computing Project, Technical Design Report, CERN/LHCC 2005-023