CLEO III Online Software: Pass1 and Level3

Sean McKinney

Physics/Computer Science, University of Missouri-Rolla, Rolla, MO, 65401

Abstract

Level3 and Pass1 are online programs used in CLEO III data acquisition for the purposes of filtering and detector diagnostics respectively. Being online programs, both require extensive testing in the areas of stability, CPU efficiency, and memory usage. This project was designed to aid in the testing and development of both of these programs. In the case of Pass1, improvements were made in speed and program stability. In addition, functionality, in the form of new monitoring algorithms, was added and tested to improve upon the existing Pass1. With Level3, two separate components were finally brought together, tested, improved, and fixed as needed.

Introduction

CLEO III is the name given to the next generation detector used in the interaction region of the CESR electron-positron collider. CLEO III data acquisition is straightforward. Any time something occurs to trigger data acquisition, the information in the data acquisition boards is read out into an "event". Events contain all of the information in every data acquisition board and are sent to the EventBuilder for processing and storage. Level3 is a filtering mechanism designed to stop useless events from ever going to the EventBuilder. Level3 makes its decisions from the "sparsified"[1] data given to it by the data crates and determines a classification for the event to be tagged with as well. Pass1 operates on the opposite side of the EventBuilder, taking data that is about to be written to tape and building diagnostic histograms from it in order to inform the operator monitoring the detector of problems as they occur.

Pass1

Pass1 serves as a CLEO III monitoring program that gives fast, near immediate diagnostics of all aspects of the detector when in operation. As with Level3, Pass1 consists of several algorithms (called "processors" and "producers") that can be added, removed, and replaced quickly. As Pass1 grew, new algorithms were added; as these algorithms were added, they had to be thoroughly tested. Oftentimes as something was

[1] Level3 does not use the full data set from any crate. To prevent it from having to process it "sparsified" versions of the data are formed in the crates and sent to Level3.

added it had bugs that had to be found and quickly fixed. Searching for problems was one of the tasks of this project; one specific instance of this search can be seen with the addition of the "Solo" algorithm.

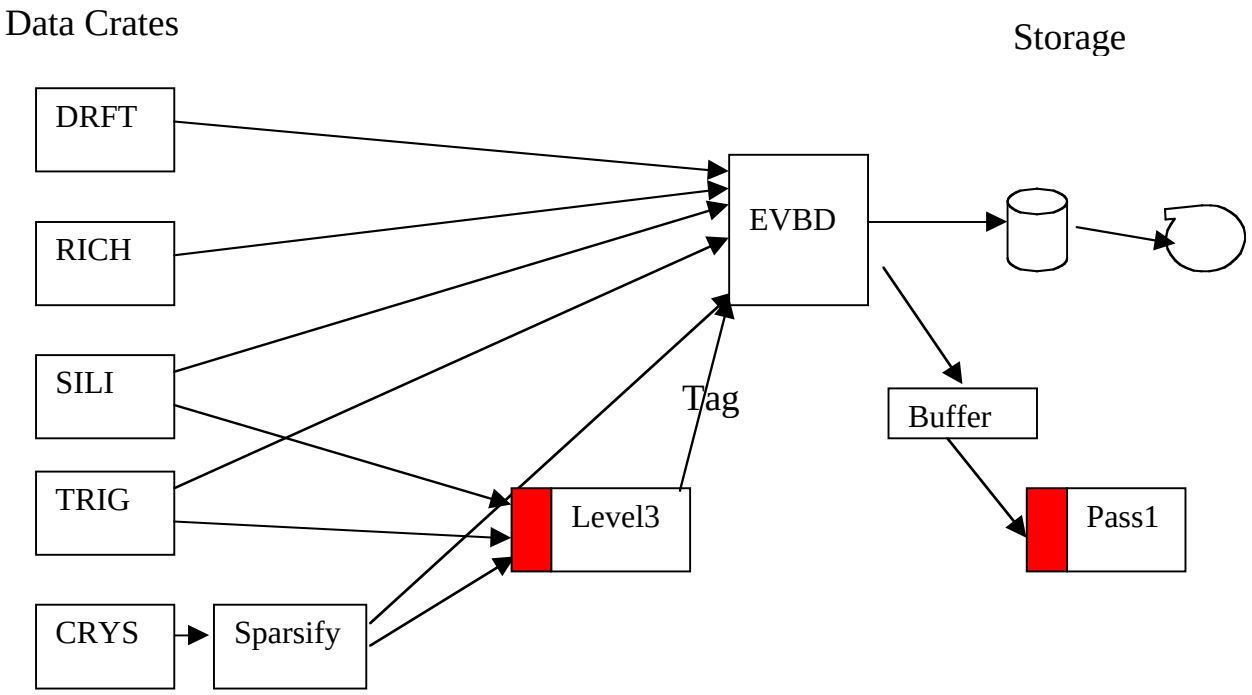

Figure 1. Level3 and Pass1 are shown with respect to data acquisition boards and EventBuilder (EVBD). The data crates consist of the following: drift chamber data (DRFT), Cheronkov radiation data (RICH), silicon wafer data (SILI), trigger data (TRIG), and crystal calorimeter data (CRYS). Note that the crystal data is "sparsified" before sent to Level3. Silicon will eventually come in the form of SIL3 (Level3 silicon sparsification).

In the past, Pass1 used a quick and dirty track finding algorithm that failed to take into account useful travel data. When a hit was found in the drift chamber, the algorithm did not use drift times to resolve the particle's position. This meant that particles were assumed to always pass through the actual point at which the wire was located, giving a poor idea of the particle's location. One of the project's first tasks was to make the switch from the old algorithm to the new one called "Solo". Given the additional complexity of "Solo", it was thought that CPU usage would rise with its addition. This was found to be true, but to a much greater degree than expected. The majority of the increased CPU usage was traced to useless, superfluous code. Once found, it was easy to pass on the problem specifics to the algorithm's author and have it remedied. Unfortunately, after adding "Solo" Pass1 began to crash rarely and unexpectedly while processing runs. Thus the new task was to determine the cause of the crash and remedy the situation. When the crash was traced to "Solo", specifics about the problem were again given to the algorithm's author, and the appropriate changes were made. In

addition to the author's adjustments, error-checking code was added to ensure that should the problem happen again, the results would not be fatal to Pass1.

Replacing older algorithms was not the only source of new Pass1 errors. When an upgrade was made to the histogram making software, problems arose. Specifically, histograms, the graphs Pass1 displays to the detector operator, no longer worked. This problem had to be traced back and was discovered to be the result of a Fortran/C++ interface problem that had escaped earlier testing.

Unfortunately, in some cases Pass1's problems only show up when run in an online environment. To simplify things, almost all testing is performed in offline mode; however when Pass1 moves to offline, network interface code can become a problem. When this happened, the added complexity of working in an online environment had to be tackled in order to track down the problem. Originally it was thought to be the result of data analysis code in Pass1 interfering with the network code. This was shown to not be the case, the network code was to blame and the problem was remedied.

Since Pass1 is a perpetually shifting and growing program, other incidents such as the ones above were common but not nearly as severe. However, in some cases Pass1 failed as the result, not of a change, but of external factors that had never been taken into account. As these were discovered, appropriate measures were taken. When a fatal error started occurring as the result of multiple events coming in with the same event number (a situation which should be impossible), integrity-monitoring code had to be written. A similar error showed up when calibration (no data) runs were processed, this required an expansion of the monitoring code. At one point it was discovered that one particular piece of data had stopped coming from its appropriate acquisition board and that Pass1 had failed to notice. Appropriate functionality was added to the monitoring code to prevent this possibility as well.

Though issues arose in the areas of stability and CPU efficiency, surprisingly no memory problems were found in Pass1. For a program of this size, written in C++, memory leaks (the allocation of memory and failure to deallocate when no longer in use) were to be expected. Leaks, however small, in an online, constantly running program such as Pass1 would eventually exhaust all sources of memory and terminate the program. Fortunately, leakages in Pass1 were not discovered. Pass1 maintains a relatively constant memory usage during operation at less than 150 megabytes of memory, which fits comfortably within the system's bounds.

Level3

Level3 is not nearly as large or complicated a piece of software as Pass1. There were only two authors and only one event processing algorithm ("processor), compared to Pass1's twelve. There is a simple reason for this: Level3 has to be trim. With peak data flow at around 1000 events per second, Level3 has got to determine whether an event is worthy of saving and what kind of event it is within a millisecond. Level3 also does not have Pass1's surge protection buffer, an intermediary storage area in between

itself and the incoming data. This means Level3 has no choice but to keep up with data flow, even at peak crate output. To make the critical decisions as quickly as possible, Level3 uses only three sources of input: the CsI crystal calorimeters, the trigger data, and the silicon wafer data. If any incoming event has energy deposition in the calorimeters that could be storage worthy, the event is saved. If any incoming event has tracks in the four layered silicon that line up with an appropriate hit in the trigger, the event is saved. These two criteria are all that Level3 uses to filter events. To aid in future analysis, Level3 makes a quick determination of the type of event under consideration by looking at the kinds of energy deposits in the calorimeters and the number of tracks found. The problem at hand was that both of the functional components of the Level3 processing algorithm (trackfinding and calorimeter analysis) had been developed independently and never brought together or tested. An additional problem was caused by the fact that an important piece of data, the sparsified silicon data, used by Level3 had not been put into the data crates yet. This sparsified silicon data therefore had to be produced using the full silicon data if any testing was to be performed of Level3. This integration of algorithms was accomplished and testing was completed. As with Pass1, the areas of primary interest were stability, speed, and memory usage.

Of primary interest was whether or not Level3 could survive hours of usage without failure. Not surprisingly the preliminary answer was no. Given the proper circumstances, Level3 would cycle endlessly. In other cases it would crash when exiting. When put into production it was discovered that calibration runs (runs without any real data) would confuse it and cause it to crash as well. Naturally, all these issues were tracked down and eliminated as discovered, and Level3 looks to be in good shape, in terms of sustained usage without crashing.

CPU efficiency is of even greater importance in Level3 than Pass1 for the reasons outlined above. Surprisingly, this was the area of least resistance. The addition of the trackfinding algorithm to the already existing crystal calorimeter code only increased CPU time per event by a factor of two or three, which was better than hoped.

Level3's only other real problem was its memory usage. In some runs memory usage could skyrocket into the 600 megabyte region after only a few thousand events, several times larger than the much more complicated Pass1. Memory usage on that scale would oftentimes lead to a crash simply because the system ran out of memory. Unfortunately memory leak detection software confirmed that there were no "memory leaks" (see above), so there had to be something fundamentally wrong with the tracking code. This was found to be a simple issue; the data unit used for storing silicon data was not being cleared out when it should have been. This was easily remedied. Now memory consumption by Level3 remains in the same realm as Pass1, again well within the constraints of the system.

Functionally, the new Level3 tracker was disappointing. When tested, Level3, which was expected to filter out somewhere between 20-40% of all events, actually only filtered about 10%. When examined it was found that raw silicon data was very noisy; appropriate thresholds had not been placed on what could be considered a valid hit. This

meant the trackfinding algorithm was finding tracks through noise and stamping bad events as good. With some appropriate cuts made in raw silicon data, the filtering was raised, but far too high. Higher filtering came at a drastically decreased efficiency in passing Bhabha events and adjustments are still being made.

Table I. Efficiencies and rejection rates for run 108797.

Level3Tracker Results (71434 events)			
Version	Bhabha efficiency (as compared to CC)	Events rejected (tracker)	Events rejected (total)
No pedestals	97%	17%	9%
Pedestals	14%	91%	24%
2-layer tracks, no pedestals	99%	13%	6%
2-layer tracks, pedestals	75%	55%	20%
Level3Proc, CC only	NA	NA	24%

In addition to the fixes outlined above, a few bits of functionality were added to Level3's trackfinding algorithm. When the algorithm was setup it was only supposed to look for tracks in the silicon wafers that involved all four layers, or three out of the four. As long as a trigger hit corresponded to a track through at least three of the silicon layers, the tracker would approve the event (for a more in depth look at Level3 tracking, see reference 1). However, some difficulties with the silicon led to decreased silicon efficiencies, so it was suggested that two layer tracks also be considered valid as well. These were added and tested. Naturally this made it even easier for Level3 to find tracks through the noise in the silicon data, and before the silicon cuts were made this dropped the filtering rate to below 5%. But, with the new cuts to silicon data made two layer tracks became more important, since in many cases only two layer tracks could be found in Bhabhas (an event type the tracker should easily find tracks for). With the cuts and two layer tracks in place, the filtering rate rose to the expected values of 20-40%, however the rate of passing Bhabha events was still down in the 75% region, which is considered unacceptable. Other methods of sparsification such as different silicon cuts and clustering algorithms met with little success. Fortunately when the tracker's shortcomings were investigated, they were found to mostly be the result of alignment issues between the layers of silicon. Ideally these issues would be resolved independently of the Level3 tracker, and once the silicon strips and constants reach readiness Level3's tracker should work just fine.

Conclusions

Pass1 is, for the moment, stable and seemingly bug free. Memory usage, CPU efficiency, and program stability are all fine and Pass1 continues to run on all data as it is

taken. Level3's tracker, however, is not performing as well as hoped because of problems with silicon data. Silicon wafers are not properly aligned and more inefficient than the original program was designed to handle. However the components such as the tracker, the internal sparsification, and the additional two-layer track option, have all been added and successfully tested. When channel pedestals and alignments are ironed out and finalized there will hopefully be no problems.

Acknowledgements

I would like to thank Professor David Kreinick, of Cornell University for his constant support and guidance throughout the duration of this project. This work was supported by the National Science Foundation REU grant PHY-9731882 and research grant PHY-9809799.

References

1. Kinney, Justin. "A Silicon Tracking Algorithm for Level3." http://www.lns.cornell.edu/public/reu/1999reports/kinney.pdf