Enabling continuous speedup of CMS Event Reconstruction through continuous benchmarking

Claudio Caputo, on behalf of the CMS Collaboration

E-mail: `claudio.caputo@cern.ch`

Abstract. The outstanding performance obtained by the CMS experiment during Run 1 and Run 2 represents a great achievement of seamless hardware and software integration. Among the different software parts, the CMS offline reconstruction software is essential for translating the data acquired by the detectors into concrete objects that can be easily handled by the analyzers. The CMS offline reconstruction software needs to be reliable and fast. The long shutdown 2 (LS2) between LHC Run2 and Run3 has been crucial for the optimization of the CMS offline reconstruction software and for the introduction of new algorithms reaching a steady CPU speedup. In order to reach these goals, a continuous benchmarking pipeline has been implemented; CPU timing and memory profiling, using the igprof tool, are performed on a regular basis to monitor the footprint of the new developments and identify possible areas of performance improvement. The current status and achievement obtained by a continuous benchmarking of CMS experiment offline reconstruction software are described here.

1. Introduction

The CMS experiment [1] archives excellent performance in both Run 1 and Run 2 thanks to a remarkable accomplishment of seamless integration of hardware and software. The CMS offline reconstruction software plays a crucial role in converting detector data into tangible objects for the analyzers, and its speed and reliability are essential. During the long shutdown 2 (LS2) between LHC Run 2 and Run 3, efforts were made to optimize the CMS offline reconstruction software and introduce new algorithms to achieve a steady CPU speedup. A constant benchmarking pipeline was established, which involves regular CPU timing and memory profiling using the igprof tool [2, 3, 4] to monitor the footprint of new developments and identify potential areas for performance improvement.

2. CMSSW releases

The collection of software referred to as CMSSW [5, 6], is built around a Framework, an Event Data Model (EDM), Services needed by the simulation, calibration and alignment, and reconstruction modules that process event data so that physicists can perform analysis. The primary goal of the Framework and EDM is to facilitate the development and deployment of reconstruction and analysis software. The CMSSW event processing model consists of one executable, called cmsRun, and many plug-in modules which are managed by the Framework. All the code needed in the event processing (calibration, reconstruction algorithms, etc.) is contained in the modules. The same executable is used for both detector and Monte Carlo (MC) data.

CMSSW development proceeds in "cycles": each cycle is intended for a specific goal, i.e. MC production campaign, data-taking campaign, etc. Before being deployed, each cycle is in a "pre" phase, which typically lasts three months, where developments that change reconstruction algorithms can be integrated. Integration, review and testing of the new developments are performed on GitHub [7]. On regular basis, a complete build of the CMSSW release (pre-release) is launched and validated.

3. Tools and flow

For each CMSSW full release, CPU and memory profiling measurements are performed. The measurements are repeated in three different scenarios:

- Run 3: code used for the Run 3 (2022-2025) data-taking and MC campaign
- Run 3 T0-like: code used for the Run 3 tested with conditions as T0
- Phase 2: code that will be used during HL-LHC

The measurement is performed using a Jenkins job on a dedicated machine: Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz. The tool used for profiling is igprof [2, 3, 4], the Ignominous Profiler. It is a nice simple tool for measuring and analysing application memory and performance characteristics. The igprof profiler requires no changes to the application or the build process and provides a web-navigable report. This is essential to easily navigate from one part of a call stack to another, spotting possible flaws or potential performance improvements. Profiling measurements are analyzed and propagated to a dedicated web-page by a Jenkins job. Figure 1 shows a schematic representation of the workflow used.

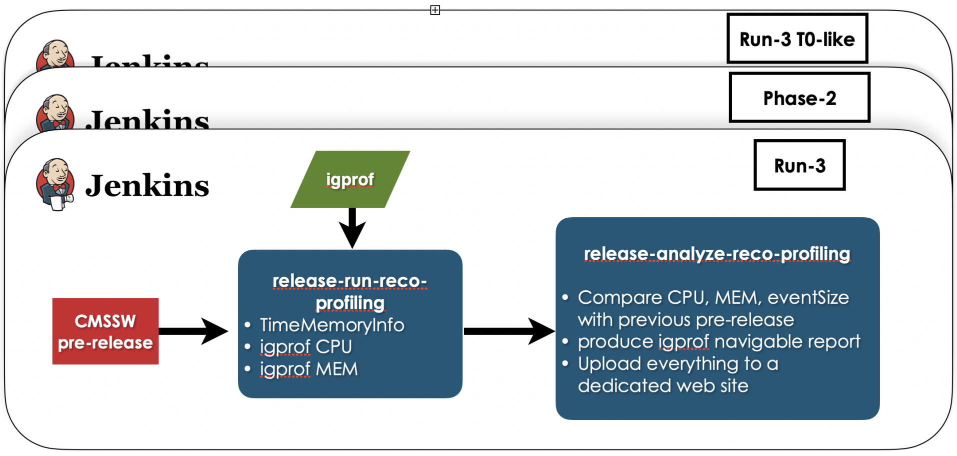
Jenkins
Run-3 T0-like
Jenkins
Phase-2
Jenkins
Run-3
igprof
CMSSW pre-release
release-run-reco-profiling
• TimeMemoryInfo
• igprof CPU
• igprof MEM
release-analyze-reco-profiling
• Compare CPU, MEM, eventSize with previous pre-release
• produce igprof navigable report
• Upload everything to a dedicated web site

Figure 1. Profiling Measurements Schema.

Detailed plots are produced to check the performance of all the modules involved during the reconstruction. An effective way to display and analyze this information is through a Circle-plot, an interactive navigable pie chart, shown in Figure 2. This plot provides a clear insight into the timing allocation among different modules.

Figure 3, 4, and 5 show the CPU reconstruction timing as a function of the main CMSSW releases, respectively for Run 3, Phase 2, and Run 3 T0-like. From the plots, a steady improvement is clearly visible in the reconstruction timing.

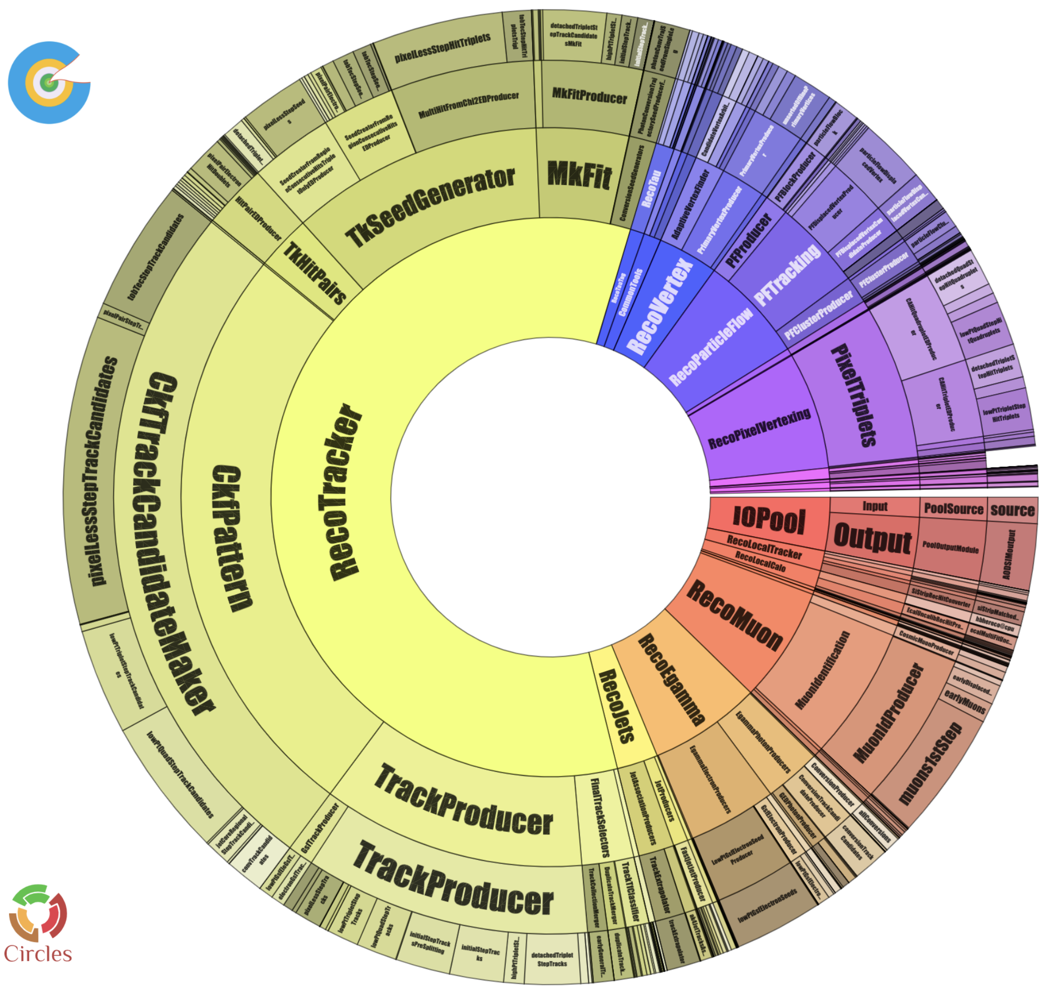

Figure 2. Run 3 CPU reconstruction timing (a.u.) Circle plot.

4. Conclusions

A continuous benchmarking of the CMS event reconstruction has allowed us to achieve a steady improvement in the reconstruction CPU timing. Measuring the performance (CPU timing, memory, output size) for each pre-release, both globally and locally, allows us to identify problems and define, in an objective and quantifiable way, the next steps. This approach enables us to identify potential areas for performance enhancement and develop actionable guidance for future plans.

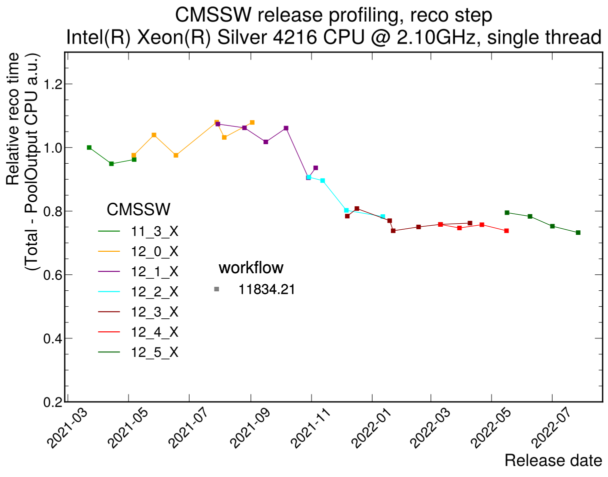

Figure 3. Run 3 CPU reconstruction timing (a.u.) as a function of the main CMSSW releases.

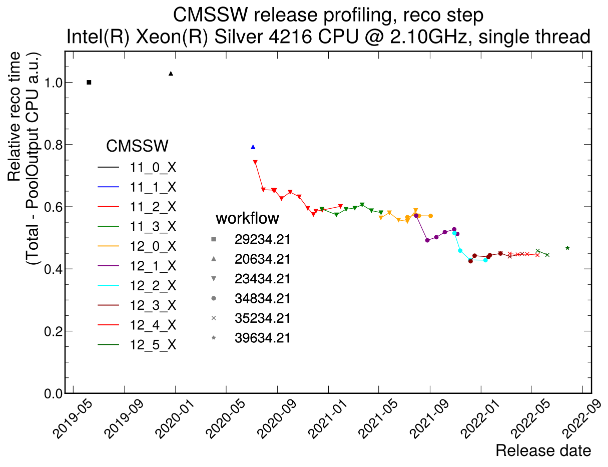

Figure 4. Phase II CPU reconstruction timing (a.u.) as a function of the main CMSSW releases.

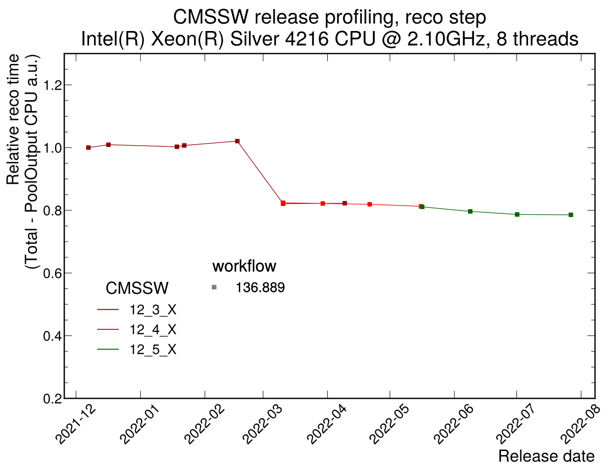

Figure 5. Run 3 T0-like CPU reconstruction timing (a.u.) as a function of the main CMSSW releases.

References

[1] CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3 (2008) S08004, doi:10.1088/1748-0221/3/08/S08004
[2] L. Tuura, V. Innocente, G. Eulisse: Analysing CMS software performance using IgProf, OProfile and callgrind, Proc. Computing In High Energy And Nuclear Physics (CHEP07), Victoria, Canada, 2007, doi:10.1088/1742-6596/119/4/042030 (paper, slides).
[3] https://igprof.org/
[4] https://github.com/igprof/igprof
[5] https://cms-sw.github.io/
[6] https://github.com/cms-sw/cmssw
[7] G. Benelli: The CMS software performance at the start of data taking, Nuclear Science Symposium Conference Record (NSS '08), 2008, doi:10.1109/NSSMIC.2008.4774926 (paper, slides).