CMS reconstruction improvements for the tracking in large pile-up events

[1]D Giordano and [2]G Sguazzoni

[1]CERN, Information Technology Department, Experiment Support Group, Geneva, Switzerland
[2]INFN, Firenze, Italy

E-mail: domenico.giordano@cern.ch, giacomo.sguazzoni@cern.ch

Abstract. The CMS tracking code is organized in several levels, known as 'iterative steps', each optimized to reconstruct a class of particle trajectories, as the ones of particles originating from the primary vertex or displaced tracks from particles resulting from secondary vertices. Each iterative step consists of seeding, pattern recognition and fitting by a kalman filter, and a final filtering and cleaning. Each subsequent step works on hits not yet associated to a reconstructed particle trajectory. The CMS tracking code underwent a major upgrade needed to make the reconstruction computing load compatible with the increasing instantaneous luminosity of LHC, resulting in a large number of primary vertices and tracks per bunch crossing. The iterative steps have been reorganized and optimized and an iterative step specialized for the reconstruction of photon conversion has been added. It is based on the innovative idea to use an existing track to build up a custom seed in the conversion hypothesis. For special event reconstruction applications, as the particle flow algorithm, it is necessary to test the possible association between a given reconstructed track and an energy deposit in calorimeters (cluster). The implementation of a k-dimensional tree in two dimensions allowed the combinatorics of links between tracks and clusters to be reduced from N*N to N*log(N), where N is the number of objects. The impact on reconstruction performances are promising and the prospects for future applications are discussed.

1. Introduction

The Compact Muon Solenoid, CMS, is one of the two general-purpose experiments installed at the Large Hadron Collider (LHC) at CERN [1]. The core of the CMS detector is the superconducting solenoid, 6 m in diameter and 13 m long, that produces a magnetic field of 3.8 T. The solenoid contains, from outside to inside, the calorimeter system and the silicon tracking system for the reconstruction of charged particles trajectories.

The silicon tracking system, shown in figure 1, is composed of a Pixel Silicon detector with three barrel layers at radii between 4.4 cm and 10.2 cm and two endcap disks at each end. Pixel sensors feature single pixel size of $100 \times 150\,\mu\text{m}^2$ for a total of 66M channels. The Silicon Strip Tracker covers the radial range between 20 cm and 110 cm around the LHC interaction point. The barrel region ($|z| < 110$ cm) is split into a Tracker Inner Barrel (TIB), made of four detector layers, and a Tracker Outer Barrel (TOB), made of six detector layers. The TIB is complemented by three Tracker Inner Disks per side (TID). The forward and backward regions (120 cm $< |z| <$ 280 cm) are covered by nine Tracker End-Cap (TEC) disks per side thus extending the overall acceptance to cover the region $|\eta| < 2.5$. In some of the layers and in the

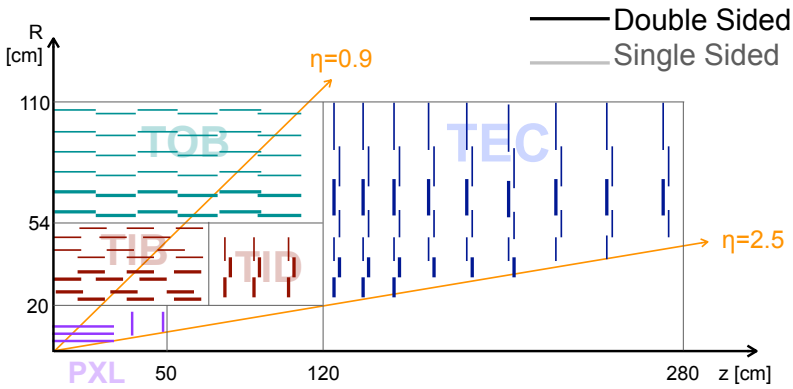

Figure 1. A simplified sketch of a quadrant of the Rz section of the CMS Tracker (bold lines represent double sided module assemblies).

innermost rings, special double-sided modules are able to provide accurate three-dimensional position measurement of the charged particle hits. The Silicon Strip Tracker is the world's largest silicon strip detector with a volume of approximately 23 m^3, instrumented by about 15,000 modules with different strip pitches ranging from 80 to 180 μm, for a total of 198 m^2 of Silicon active area and about 9.6 million channels with full optical analog readout [1][2][3].

The CMS track reconstruction [4] starts with the appropriate grouping of the hits in the innermost layers to build up *seeds*. The seed is an initial track estimate and consists of a pair or a triplet of hits, sufficient for a basic prediction of the trajectory parameters if the primary vertex is also used. Starting from a given seed, pattern recognition using a Kalman Filter is performed to build inside-out trajectories. Then each identified track undergoes a procedure to reject possible outlier hits and is refitted, also using a Kalman Filter. Finally, a quality selection is performed. Reconstruction efficiency relies on several iterations (steps) of the tracking procedure; every step, except the first, works on the not-yet-associated hits surviving the previous step. Each step is optimized with respect to the seeding topology and to the final quality cuts. This recursive procedure is referred to as *iterative tracking*.

The algorithms that build up the CMS event reconstruction software, CMSSW [5], output physics objects (e.g., tracks, electrons, jets, ...) from the raw data recorded by the detector. Ideally, all events collected by CMS are reconstructed by the CMS prompt reconstruction system quasi real-time, soon after being collected. The prompt reconstruction stream is of masterpiece importance for a discovery experiment and prompt reconstruction data samples have been the base of most of the CMS physics results so far. Clearly, they are also crucial for fast and accurate feedback on detector conditions.

The complexity and the granularity of the tracker system in connection with the large LHC instantaneous luminosity, resulting in a large number of interactions (pile-up) per bunch crossing, make the track reconstruction largely dominating the entire reconstruction chain memory-wise and in terms of CPU time.

The LHC instantaneous luminosity is steadily growing since the beginning of the run in 2010. In figure 2 and figure 3 the maximum LHC instantaneous luminosity per day delivered to CMS is shown as a function of time for 2011 and 2012, respectively. During the 2012 the LHC instantaneous luminosity is expected to reach a value of $7 \cdot 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{sec}^{-1}$ that corresponds to a number of primary vertices per bunch crossing of about 25.

The other worsening factor that was not foreseen at the design level of LHC and consequently not taken into account in writing first versions of reconstruction software, is the bunch crossing frequency. LHC was supposed to run with 25ns between subsequent bunch crossings. But as a consequence of the LHC operating conditions and beam optics, it is preferable to run with 50ns between bunch crossings. This allows for fatter bunches in the machine and, eventually, for a larger overall delivered luminosity. On the other hand, this results into a larger pileup

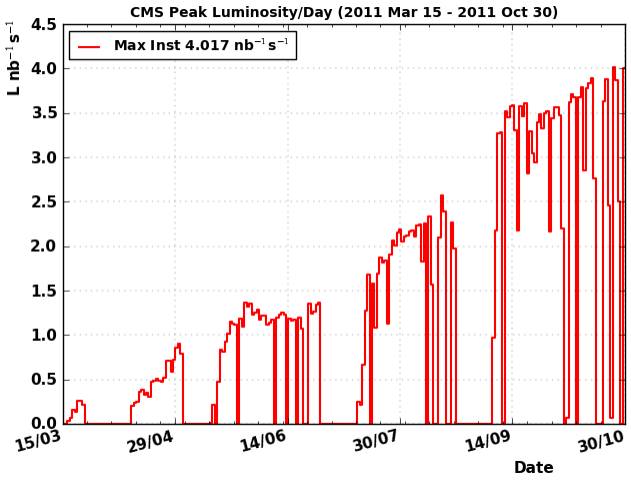

Figure 2. Maximum instantaneous luminosity per day delivered to CMS for pp running at 7 TeV centre-of-mass energy in 2011.

per bunch crossing and considerably increases the computing time per event that, being due to combinatorics, scales faster than purely linear dependence. During 2011 it was soon clear that the current version of reconstruction software was not performing well enough to withstand prompt reconstruction for last part of 2011 and 2012.

This paper describes the actions that have been put in place to improve the tracking reconstruction software, the major responsible of the CPU load at the reconstruction level, in order to be compliant with the expected luminosity in 2011 and 2012. This improvement campaign took place in two phases: the first during the 2011 and completed in September 2011 (Fall 2011), the second started at the beginning of 2012 and completed in March 2012 (Spring 2012).

2. Fall 2011

The modifications put in place and optimized in the first phase of the improvement campaign have been developed on top of CMSSW version 4.2.x and implemented in the CMSSW version 4.4.x.

A first group of improvements are purely based on smarter coding and better algorithm implementation and do not change the physics outcome of the tracking reconstruction workflow.

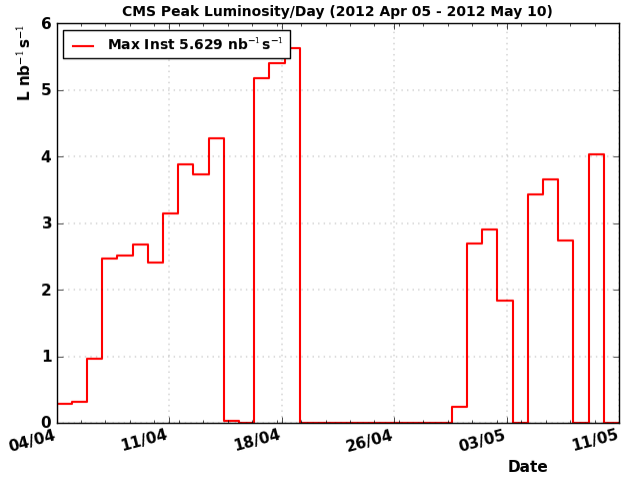

Figure 3. Maximum instantaneous luminosity per day delivered to CMS for pp running at 8 TeV centre-of-mass energy in 2012.

They are mainly targeted to reduction and better handling of the memory and in fact they allow for a 40% cut of the memory budget. More in detail, these modifications are described below.

Copy-less cluster masking within the iterative tracking. Each step of the iterative tracking, but the first, works on the hits not yet associated to any track. Technically this was implemented by creating a new collection of surviving hits at each step. To save memory, a masking algorithm has been implemented adding to the hit object an appropriate data member for the masking bits. Results are unchanged with a major reduction of the allocated memory.

Batch cleaning of track candidates. The track candidate results from a seed that has been successfully propagated. Before being declared as a reconstructed track the track candidate must undergo a filtering selection to reject fakes. To avoid storing too much track candidates in memory the cleaning procedure is done once a subsample of 1000 track candidates has been accumulated with large benefit on the overall required memory.

Efficient quality assignment. Each step in the iterative tracking assigns tracks to a quality tier. Old implementation of the algorithm just created a copy of the same track per each quality tier it was belonging to; this has been modified by removing the copying and adding an appropriate data member to store the quality tier bits with an obvious advantage on the memory consumption.

Efficient track merging. After all iterative steps, the resulting track collections have to be merged and further cleaned up by potential fake tracks and duplicated tracks. In fact, only hits associated to tracks with highest quality, know in CMS as *high purity tracks*, are not used in the following steps. But hits associated to lower quality tracks are retained, in the attempt to build better tracks out of them with different seed and propagation parameters. Old implementation of merging algorithm compared the collection created by the various steps in pairs creating intermediate collections to be further compared with other collections up to the end of the process. In the updated version all track collections feed a merging module that works without creating any intermediate collection. This is pictorially shown in figure 4.

The second group of ameliorations directly affects the algorithms and thus the outcome on observables and have to be evaluated also with respect to performances on physics. These modifications target the CMSSW modules related to tracking that are dominating, in terms of CPU time, the entire reconstruction chain and are described in the following.

Iterative tracking. A factor 2.5 of improvement in the CPU time has been obtained by optimizing the iterative tracking, as detailed in table 2 to be compared with table 1 that

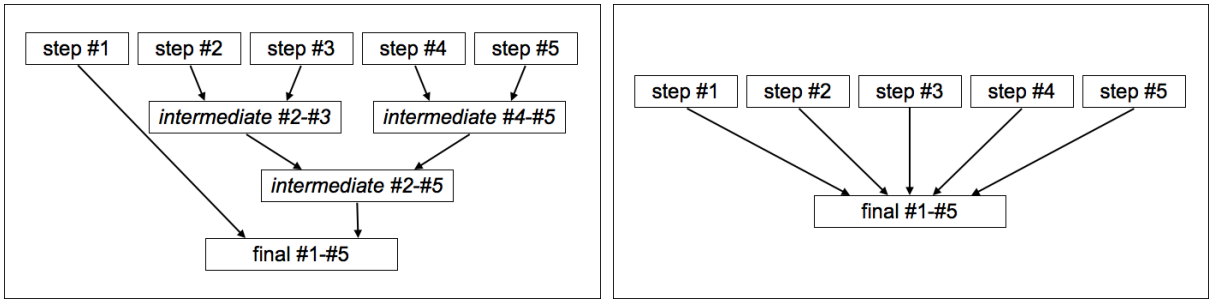

Figure 4. Schematical representation of the old (left) and the new (right) merging algorithm for an hypothetical iterative tracking with five steps; "intermediate" track collections are avoided in the new algorithm and this allows for consistent memory savings.

summarizes the baseline configuration of CMSSW 42x. As can be seen, the net effect is an increase of the effective P_T threshold for track reconstruction together with tighter constraint on impact parameter. This configuration results into a reduced efficiency for P_T lower than $300\,\mathrm{MeV}/c$ but a larger efficiency for P_T greater than $0.9\,\mathrm{GeV}/c$.

Reconstruction of photon conversions. Reconstruction of photon conversion in the tracker volume results heavily affected by the higher P_T threshold and by the tighter impact parameter cuts since conversion leg tracks are typically soft and displaced. To recover this loss, a dedicated seeding has been deployed [6] and the photon conversion reconstruction has been further optimized resulting in a factor 12 improvement of the CPU time for conversion reconstruction.

Reconstruction of primary vertices. The reconstruction of primary vertices in the event has been optimized by integrating into the same module all the different reconstruction methods; the removal of the overhead due to the module split we had beforehand was enough to gain a factor two in CPU time in this specific context.

Reconstruction of nuclear interactions. Similarly to photon conversions, also nuclear interactions are reconstructed for tracker material studies and to correctly estimate the hadronic energy fraction in jets within the Particle Flow, a sophisticated event reconstruction that consists in reconstructing and identifying each single particle with an optimized combination of all subdetector information. To avoid consuming CPU time in the heavy vertex fit with candidates that are very likely to be fakes, a very simple preselection

Table 1. Relevant parameters of the six tracking iterative steps in CMSSW 42x, i.e. before the reconstruction improvement campaign described in this paper; σ represent the beam spot size along the z axis.

#step	seed type	seed subdetectors	P_T^{min} [GeV/c]	d_0 cut	z_0 cut
0	triplet	pixel	0.8	0.2 cm	3.0σ
1	pair	pixel/TEC	0.6	0.05 cm	0.6 cm
2	triplet	pixel	0.075	0.2 cm	3.3σ
3	triplet	pixel/TIB/TID/TEC	0.25-0.35	2.0 cm	10.0 cm
4	pair	TIB/TID/TEC	0.5	2.0 cm	12.0 cm
5	pair	TOB/TEC	0.6	6.0 cm	30.0 cm

Table 2. Relevant parameters of the seven tracking iterative steps in CMSSW 44x, after the first phase of the improvement campaign in fall 2011; in bold the parameters changed with respect to the corresponding steps in CMSSW 42x (see table 1); step #1 is brand new with respect to CMSSW 42x; σ represent the beam spot size along the z axis.

#step	seed type	seed subdetectors	P_T^{min} [GeV/c]	d_0 cut	z_0 cut
0	triplet	pixel	**0.6**	**0.03** cm	**4.0**σ
1	**triplet**	**pixel**	**0.2**	**0.03** cm	**4.0**σ
2	pair	**pixel**	0.6	**0.01** cm	**0.09** cm
3	triplet	pixel	**0.2**	**1.0** cm	**4.0**σ
4	triplet	pixel/TIB/TID/TEC	**0.35-0.5**	2.0 cm	10.0 cm
5	pair	TIB/TID/TEC	**0.6**	2.0 cm	**10.0** cm
6	pair	TOB/TEC	0.6	2.0 cm	30.0 cm

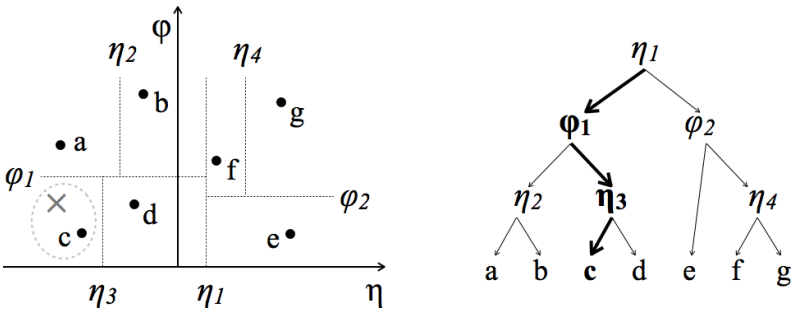

Figure 5. Pictorial representation of the kd-tree algorithm in the case of a very simple neighbor search problem in the (η, ϕ) plane: a track (represented by the "×" symbol) needs to be associated to one of the calorimetric clusters represented by the dots labelled with letters. On the left panel it is shown the way (η, ϕ) is split into domains and the resulting navigation tree is sketched on the right.

has been implemented: a nuclear interaction candidate track is kept only if P_T exceeds 800 MeV/c, in case of primary tracks, or if d_0 is larger then 2 mm for secondary tracks; the vertex candidate must have at least three tracks (one primary and two secondaries or three secondary) and, finally, candidates are discarded if the secondary vertex estimate falls within the beam pipe. These simple criteria are enough to reduce the combinatorics such that the nuclear interaction reconstructions gains a factor 5 in CPU time with no sensible degradation in physics performances.

Particle flow links. The Particle Flow algorithm needs to link tracks to calorimetric clusters in the (η, ϕ) parameter space. This problem of nearest neighbor search over a large number of objects in CMSSW 42x is implemented with nested loops and results to be rather time intensive. Moreover the complexity scale quadratically (N^2) as the object multiplicity N increases. In CMSSW 44x the well known linearization technique known as *kd-tree* [7] has been introduced to replace nested loops. The method consists in an algorithm that, starting from a collection of objects (calorimetric cluster, for example), dynamically splits the (η, ϕ) space into appropriate domains, each containing one single object, organized in a tree. The closest cluster to a given track can be found by exploring the (η, ϕ) space with a very fast binary search that ends up in the closest neighbor domain. This algorithm, schematically

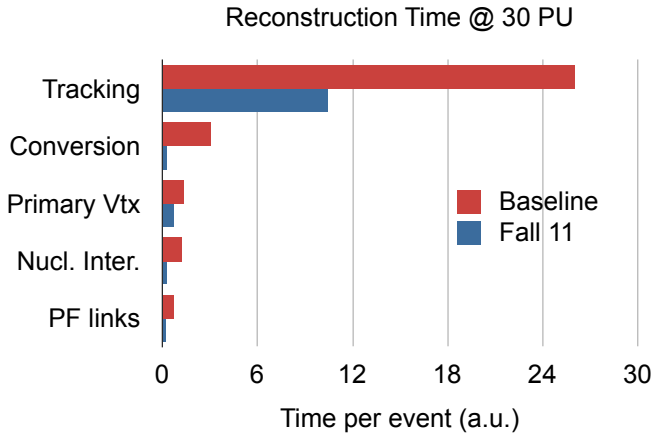

Figure 6. Breakdown of the average CPU time per event in arbitrary units before and after 'Fall 2011' for each improvement area of the tracking reconstruction software (simulated QCD events with 30 pile-up interactions).

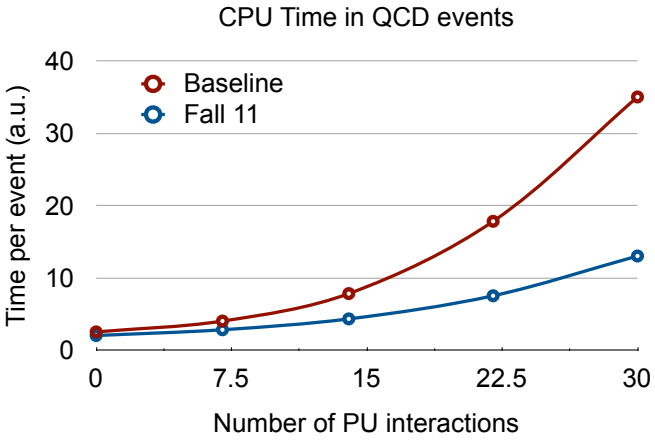

Figure 7. Total reconstruction CPU time per event (in arbitrary units) for as a function of pile-up events for simulated QCD events for the baseline CMSSW and the improved version.

represented in figure 5, has a complexity that scales as $N \cdot \log N$, thus more convenient with respect to standard nested loops especially for large multiplicity, and allows to gain a factor 4 in CPU time in this specific application. Its extension to other modules of CMSSW is being studied.

The results of the improvements just described are graphically represented in figure 6 and figure 7. In the former the breakdown of the CPU time for each improvement area before and after each improvement is reported for simulated QCD events with 30 pile-up interactions per event; in the latter the total CPU time is shown as a function of the number of pile-up interactions for simulated QCD events.

3. Spring 2012

The modifications put in place in the second phase of the improvement campaign have been developed on top of CMSSW version 4.4.x and implemented in the CMSSW version 5.2.x. Again a group of improvements are based on better coding and technological improvements and do not change the physics outcome. More in detail, these modifications are described below.

Change of compiler version. The implementation of CMSSW 5.2.x has been accompanied by the switch from `gcc` 4.3.4 to `gcc` 4.6.2 to produce binaries. This latter compiler version allows for faster code to be generated also thanks to some compiler specific optimizations. The net gain is up to 10% as shown in figure 8 where the reduction in CPU time is shown as a function of the number of pile-up vertices for simulated QCD events. Other features that came along the new compiler version are the C++11 standard support and autovectorization flags on by default.

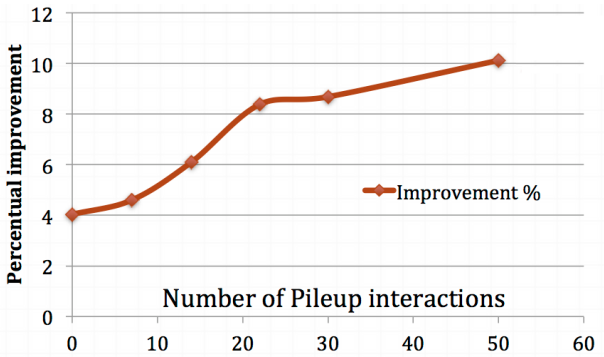

Figure 8. Relative CPU time reduction to be ascribed to the introduction of gcc 4.6.2 as a function of the number of PU vertices for simulated QCD events.

JEMalloc. Switch to the concurrent malloc implementation JEMalloc highly performant and able to better redeem memory.

Switch to improved ROOT version. The ROOT package version has been changed from 5.27 to 5.32 that features several improvements, especially in I/O with less memory required.

Several design modifications to improve speed and memory consumption. The code has been again carefully reviewed and many improvements have been implemented. Several of those are related to track reconstruction classes. For example, the devirtualization of the BasicTrajectoryState class (an ancillary class for track reconstruction) resulted into a 10% gain in speed and in some 100MB of RSS saved per event. Similarly the stereo hit class (the class that stores the double sided module hits) has been considerably slimmed down (a factor three in size) with a net decrease of RSS memory from 50MB to 150MB, depending on the event occupancy.

Another set of modifications directly affects the outcome on physics output. These are described in the following.

Offline vertexing. The offline reconstruction of primary vertices is based on a deterministic annealing algorithm to find the z coordinate of the vertices. Major improvements have been deployed for CMSSW 52x: loops have been autovectorized (thanks to the introduction of the new compiler) but, to further profit of autovectorization capabilities, the exponential functions heavily used in the algorithm have been replaced with a fast, autovectorizable inlined double precision version. Eventually the deterministic annealing algorithm has been further made more efficient by optimizing some configuration parameters with essentially no change in physics performances. The net increase in CPU time amounts to a factor 3 for large PU events.

Cluster shape based seed filtering The large CPU time needed by the track reconstruction is to be ascribed to the huge number of seeds due to hit combinatorics; in fact a propagation has to be attempted for each of them. A way to keep this number under control is to implement filters able to reject fake seeds. One of the most effective is based on the cluster shape. For example a track impinging a sensor with a large angle will generate a cluster wider than a track with normal incidence. This can be used to evaluate seed compatibility with the track hypothesis. Such a filter was used only in steps #0 and #1 in CMSSW 44x (see table 2); for CMSSW 52x it has been extended also to steps #2, #4 and #5 (see table 3) with a substantial CPU time benefit especially in some step. For example, the step #2, particularly prone to combinatorics since seeds are made up of hit pairs, sees a CPU time reduction of a factor 2.7. Overall the improvement in CPU time is of a factor 1.5.

Iterative tracking. After all the modifications above described, also the iterative tracking has been further optimized for CMSSW 52x. Nevertheless the changes, summarized in table 3, are tiny as a demonstration that upstream improvements are already almost sufficient to make CMSSW compliant with requirements. There is no need to modify deeply the iterative tracking, i.e. to to reduce combinatorics and match performance target by increasing effective P_T thresholds and/or by reducing efficiency for displaced tracks.

The overall result obtained with the "spring 2012" campaign improvements implemented in CMSSW 52x is shown in figure 9 where the dependence of RSS memory as a function of running time is plotted in CMSSW 44x and CMSSW 52x for a reconstruction job of 100 real data events from the 2011 special run with high PU. The substantial reduction either in memory load either in total running time is clearly evident.

4. A glimpse into the future

The challenge for the CMS reconstruction cannot be considered over with the deployment of the software for 2012 data taking, currently ongoing. After the first long shutdown,

Table 3. Relevant parameters of the seven tracking iterative steps in CMSSW 52x, after the second phase of the improvement campaign in 2012; in bold the parameters changed with respect to the corresponding steps in CMSSW 44x (see table 1); σ represent the beam spot size along the z axis.

#step	seed type	seed subdetectors	$P_T^{\min}$ [GeV/c]	d_0 cut	z_0 cut
0	triplet	pixel	0.6	**0.02** cm	4.0σ
1	triplet	pixel	0.2	**0.02** cm	4.0σ
2	pair	pixel	0.6	**0.015** cm	0.09 cm
3	triplet	pixel	**0.3**	**1.5** cm	$\mathbf{2.5}\sigma$
4	triplet	pixel/TIB/TID/TEC	**0.5-0.6**	**1.5** cm	10.0 cm
5	pair	TIB/TID/TEC	0.6	2.0 cm	10.0 cm
6	pair	TOB/TEC	0.6	2.0 cm	30.0 cm

foreseen for almost two years across 2013 and 2014, LHC will increase center-of-mass energy and instantaneous luminosity as well. This will require a major reengineering of the entire reconstruction software and of the tracking as well.

Two major areas of improvements are being outlined: implementation of tracking techniques never used in CMS up to now (like Hough transforms tracking); exploitation at any possible level of parallelization techniques. The latter, in particular, turns out to be necessary to better profit of the actual trend of increase of the computing power that is realized by an increase of the number of cores in the same monolithic CPU.

Parallelization can be implemented in several ways. At the level of the framework by allowing different modules to run in parellel taking appropriately into account all dependencies; this would be almost transparent for the final user and developer, i.e. it would require no to minor changes to user and reconstruction module code. Nevertheless this is not optimal as some modules of CMSSW need much longer time to run with respect to the others; track reconstruction is the typical example. In this case it is worthwhile to implement parallelization at the module and algorithm level. This requires code modifications but is also much more effective. Prototype implementations are already being studied and are very promising [8].

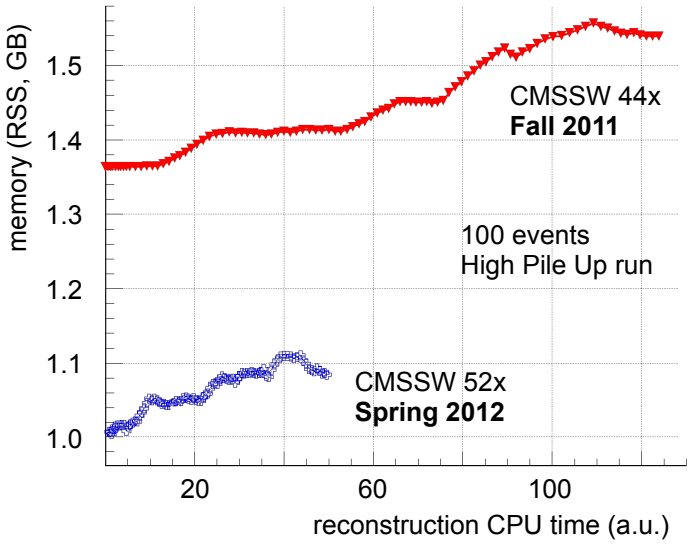

Figure 9. RSS memory as a function of running time in CMSSW 44x and CMSSW 52x for a reconstruction job of 100 real data events from the 2011 special run with high PU.

5. Conclusions

The CMS track reconstruction has been recently streamlined to allow for quasi real-time prompt reconstruction to be performed with the available computing resources during 2011 and 2012 data taking, despite the steady increase of instantaneous luminosity delivered by the LHC thanks to its impressive performance.

This goal has been achieved by a number of actions taken at any level of the track reconstruction workflow by means of smarter coding techniques and by using technological improvements as well. The overall gain can be approximately estimated in a factor 2 reduction of memory load and a factor 7 reduction of CPU time for the typical 2012 event.

The LHC operating conditions after the long shutdown, foreseen in 2013-2014, requires further performance improvements that will be object of future activities. New tracking techniques and parallelization will be implemented. Prototype applications are already being investigated.

References

[1] Adolphi R et al. [CMS Collaboration] 2008 The CMS experiment at the CERN LHC *JINST* **3** S08004

[2] CMS Collaboration 1998 The Tracker System Project Technical Design Report *CERN-LHCC* 98-6

[3] CMS Collaboration 2000 Addendum to the CMS Tracker TDR *CERN-LHCC* 2000-16

[4] CMS Collaboration 2010 Tracking and Vertexing Results from First Collisions *CMS-PAS* TRK-10-001

[5] CMS Collaboration 2006 CMS Physics Technical Design Report Detector Performance and Software *CERN-LHCC* 2006-001

[6] Giordano D and Sguazzoni G [CMS Collaboration] 2012 An innovative seeding technique for photon conversion reconstruction at CMS *these proceedings*

[7] Bentley J L 1975 Multidimensional binary search trees used for associative searching *Commun. ACM* **18** 9 (Sep. 1975) 509-517

[8] Hauth T, Innocente V and Piparo D [CMS Collaboration] 2012 Development and Evaluation of Vectorised and Multi-Core Event Reconstruction Algorithms within the CMS Software Framework *these proceedings*