Journal of Physics: Conference Series **513** (2014) 052010 doi:10.1088/1742-6596/513/5/052010

Parallel track reconstruction in CMS using the cellular automaton approach

D Funke[1], T Hauth[2], V Innocente[2], G Quast[1], P Sanders[1] and D Schieferdecker[1]

[1] Karlsruhe Institute of Technology, Karlsruhe, DE [2] CERN, Geneva, CH

E-mail: daniel.funke@cern.ch thomas.hauth@cern.ch

Abstract. The Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC) is a general-purpose particle detector and comprises the largest silicon-based tracking system built to date with 75 million individual readout channels. The precise reconstruction of particle tracks from this tremendous amount of input channels is a compute-intensive task. The foreseen LHC beam parameters for the next data taking period, starting in 2015, will result in an increase in the number of simultaneous proton-proton interactions and hence the number of particle tracks per event. Due to the stagnating clock frequencies of individual CPU cores, new approaches to particle track reconstruction need to be evaluated in order to cope with this computational challenge. Track finding methods that are based on cellular automata (CA) offer a fast and parallelizable alternative to the well-established Kalman filter-based algorithms. We present a new cellular automaton based track reconstruction, which copes with the complex detector geometry of CMS. We detail the specific design choices made to allow for a high-performance computation on GPU and CPU devices using the OpenCL framework. We conclude by evaluating the physics performance, as well as the computational properties of our implementation on various hardware platforms and show that a significant speedup can be attained by using GPU architectures while achieving a reasonable physics performance at the same time.

1. Introduction

The Large Hadron Collider (LHC) at CERN in Geneva is the most powerful particle collider ever built. One of the four major physics experiments at the LHC is the Compact Muon Solenoid (CMS), a general-purpose particle detector. Its research program is extremely ambitious and spans from the search of the Higgs boson to the quest for signals originating from processes beyond the established Standard Model of particle physics. In July 2012, the ATLAS and CMS experiment announced the discorvery of a new boson which is compatible with properties expected from a Higgs Boson, culminating in the Nobel Prize in Physics for François Englert and Peter Higgs in 2013.

In order to carry out further physics searches and investigation of the Higgs candidate, a tremendous amount of data has to be processed. Part of this procedure includes the so-called event reconstruction, which transforms the digital signals coming from the detector's read-out electronics into physics objects, such as leptons and hadronic jets. These procedures require a wide range of algorithms including cluster finding, pattern recognition, linear algebra computations and function minimization. For this purpose, the CMS Collaboration has developed an object-oriented C++ software framework, called CMSSW [1].

With the continuous increase of the luminosity delivered by the LHC machine, the treatment of the recorded data becomes more and more computationally expensive. The demand for computing capabilities will increase even further with the upgrade to the Super-LHC [2], which will raise the instantaneous luminosity by a factor of 10 to $10^{35} cm^{-2} s^{-1}$.

As the increase in clock speed of CPUs has been stagnating in recent years, this challenge must be addressed by exploiting the capabilities of the latest microprocessor architectures, such as vector units of growing size and an increasing number of cores per die.

At the same time, Graphics Processing Units (GPUs) – initially only intended for high-performance computer graphics and gaming – have become a possible tool for the reconstruction of physics measurement. While these devices might not be as versatile as classical CPUs, they offer the possibility of massive parallelism. To leverage this potential, data structures and algorithms must be tailored towards GPU programming models.

This paper describes a novel track finding algorithm for CMS, based on the cellular automaton approach [3]. Tracks are reconstructed by joining compatible triplets of hits. Effective and efficient criteria are presented to determine the compatibility of triplets. Furthermore, valid hit triplets need to be identified by inexpensively computable measures, which are presented prior to the triplet joining.

2. Parallel Computing in Heterogeneous Environments

The CMSSW application is executed in various computing centers around the world, featuring a wide range of hardware equipment [4]. Therefore, parallelism needs to be exploited in a transparent manner.

For multi-core programming, various frameworks exist, such as OpenMP or Intel TBB. In order to utilize the CPU's vector units, native commands – *vector intrinsics* – need to be used or vectorized libraries employed. General purpose GPU (GPGPU) programming on the other hand, requires GPU devolopment kits such as NVIDIA's CUDA.

The Open Compute Language (OpenCL) is an open framework for heterogenous massively-parallel processing, maintained by the Khronos Group [5]. It allows the transparent exploitation of GPUs, multi-core CPUs and vector units, with little performance penalty compared to less general frameworks [6, 7]. It partitions the application into *kernels* – performing the compute intense operations on the compute device (CD), e.g. a GPU – and the *host* application – steering the data flow to and from the CD as well as scheduling kernels for execution.

3. Algorithms and Data Structures

For efficient parallel track reconstruction, the algorithms and data structures need to be designed with the peculiarities of the chosen platform in mind. To avoid costly data transfers between host and CD, all processing is performed on the CD. The reconstruction proceeds in four steps: i) pair generation, ii) triplet prediction, iii) triplet filtering and iv) triplet joining. The first two steps generate triplet candidates out of hits in three detector layers. To avoid the $\mathcal{O}(n^3)$ combinatorial complexity of triplet formation, the physical properties of a charged particle's track through a magnetic field are exploited to predict search windows. A uniform grid data structure is used [8] to efficiently query for the hits within the predicted search range in a layer.

All algorithms need to address the lack of dynamic memory allocation within an OpenCL kernel. For instance, the number of feasible hit pairs and triplets within the respective search windows is unknown before kernel execution, however allocating ample memory to hold all possible combinations beforehand is infeasible. Thus, a *two-pass scheme* is used as main building block for all presented algorithms. A general two-pass scheme for a predicate p and a function f evaluated on a set of inputs $\{x_1, \ldots, x_n\}$ consists of a counting pass – determining the number of inputs x_i for which $p(x_i)$ is true – and a store pass – actually writing $f(x_i)$ for valid inputs x_i to their output memory. Both passes are realized as separate kernels in between which memory

is allocated by the host for the determined number of outputs. If the n inputs are processed by t threads then each thread i counts the number of valid inputs in its partition $\{x_{i\cdot\frac{n}{t}}, \ldots, x_{(i+1)\cdot\frac{n}{t}-1}\}$ and stores the result s_i in an array $\mathbf{s} = \{s_0, \ldots, s_i, \ldots, s_t\}$. Calculating the exclusive prefix sum of $\mathbf{s}$, $\{0, s_0, s_0 + s_1, \ldots, \sum_{0 \leq k < t} s_k\}$, yields the total number of found valid inputs in the last entry of the array. Furthermore, the offset into the output memory for each thread i is given by the ith entry in the prefix sum array. As the evaluation of predicate p is generally costly, the outcome of evaluating $p(x_i)$ is stored in an "oracle" bit-string $\mathbf{o} = \{o_1, \ldots, o_{\mathcal{O}(n^3)}\}$. The oracle bit o_i is used during the store phase to determine the validity of input x_i, reducing the computation in the store kernel to evaluating $f(x_i)$ for valid x_i.

All following algorithms employ the presented two-pass scheme, adapting the evaluated predicate and the use of the oracle bit-string to their particular needs. The approach can be employed for a wide range of problems beyond the presented triplet finding.

3.1. Grid Data Structure

A grid is a regular tessellation of space into contiguous cells and is independent of the distribution of data points within that space [8]. For two-dimensional detector layers in (ϕ, z), a grid partitioning the layer into $\#_z \cdot \#_\phi$ cells requires linear space and can be constructed ex-situ in linear time and space.

The construction follows the two-pass scheme outlined above. In the count pass, the number of hits within each cell of the grid is determined. If feasible, fast local memory of the CD is employed to store the counters. Subsequently, the prefix sum of the counter is computed according to the algorithm described by [9], yielding the bounds of each cell. In the store phase, the hits are copied to their appropriate positions in the grid data structure.

As each grid cell can be accessed in constant time, the data structure is very suitable to retrieve hits within the predicted search windows in the subsequent steps of triplet finding. Its uniform structure is well suited for GPGPUs, since branching introduces a significant performance penalty on these devices. The granularity of the grid is a tuning parameter. More fine-grained grids require more construction effort; the combinatorics, however, are limited in later steps due to fewer retrieved hits.

3.2. Pair Generation

Pairs of hits form the basis for later triplet building. However, hit pairs reveal limited information about a particle's trajectory through the detector. Therefore, only rough extrapolations can be employed to limit the search window and reduce the number of fake pairs.

Given a first hit, the admissable z- and ϕ-range of the second one is calculated based on the maximum longitudinal and transverse impact parameter– $z_{0,\max}$ and $d_{0,\max}$, respectively – and the minimum transverse momentum p_T of the particle track, similar to the current CMSSW implementation. Based on a hit $h_2 = (\phi_2, z_2)$ in the *outer* layer, the search range for z in the inner layer can be limited to

$$\left[r_1 \frac{r_2}{z_2 + z_{0,\max}} - z_{0,\max}, r_1 \frac{r_2}{z_2 - z_{0,\max}} + z_{0,\max} \right],$$

with r_1 and r_2 denoting the radii of the detector layers. The minimum p_T of a particle prescribes a maximum curvature – i.e. a minimum radius $r_{\min}$ – of its track in presence of a magnetic field. Therefore, Δ_ϕ can be limited to

$$|\Delta_\phi| \leq \left| \arccos\left(\frac{r_2}{2r_{\min}}\right) - \arccos\left(\frac{r_1}{2r_{\min}}\right) \right| + \arctan\left(\frac{d_{0,\max}(r_2 - r_1)}{r_1 r_2}\right).$$

Several expensive operations are required to calculate the prediction window, as listed in Tab. 1. A simpler approach, restricting the search window purely based on grid-based proximity measures, proved unviable due to the large number of fake hit pairs produced.

3.3. Triplet Prediction

Given a hit pair, restricting the admissible search window for third hit candidates is of paramount importance to cope with high track multiplicities. The trajectory "defined" by the hit pair is extrapolated in a similar fashion as in pair generation. When moving the origin of the coordinate system to the hit in the first layer, the calculations for Δ_ϕ are identical to the ones in pair generation except that the change in ϕ due to d_0 is already accounted for by the distance between hit h_1 and h_2 in the transverse plane. In the longitudinal plane, the z coordinate of the hit in the third layer is determined by a straight line extrapolation of the hit pair, taking into account the effects due to the transverse impact parameter d_0 [1].

Tab. 1 summarizes the number of expensive operations required by the algorithm.

3.4. Triplet Filtering

Triplet candidates produced in the previous step may include many *fake* triplets, composed of hits originating from the interaction of multiple particles with the detector material. In order to discard these fake triplets, efficiently calculable properties of *valid* triplets are employed.

Particles approximately follow a straight line in the longitudinal plane, therefore the polar angle θ of the track segment connecting the first and the second hit should be identical to the polar angle θ' of the second segment – between the second and third hit. Due to non-ideal magnetic field conditions, multiple scattering, and limited detector resolution of a hit's z-coordinate, small variations $d\theta$ between individual segments must be permissible, see (Eq. 1).

In the transverse plane, a charged particle's trajectory prescribes a circular path, whose radius depends on the transverse momentum p_T of the particle. Limiting the difference in ϕ between two track segments thus imposes a constraint on the minimum p_T. Furthermore, the admissible $d\phi$ must account for the energy loss of the particle, multiple scattering, and detector resolution, (Eq. 2).

A further discriminator for valid and fake tracks is the transverse distance of a trajectory's closest approach to the beam line, (Eq. 3). In order to determine d_0, a circle must be fitted to the hits of a trajectory. The problem of fitting a circle to points on a plane can be transformed to the more efficiently solvable problem of fitting a plane to points on a circular paraboloid [10]. The algorithm yields the radius and the center of the circle, thus the point of closest approach to the beam line can be easily calculated.

$$\left|\frac{\theta'}{\theta} - 1\right| \leq d\theta \qquad (1) \qquad \left|\phi' - \phi\right| \leq d\phi \qquad (2) \qquad d_0 \leq d_{0,\max} \qquad (3)$$

The cutoff values $d\theta$, $d\phi$ and $d_{0,\max}$ define the trade-off between the need for efficiency – i.e. the number of valid triplets found – with the desire to discard the majority of fake triplets. Suitable values are identified by studying simulated $t\bar{t}$ events. In order to account for the varying spatial resolution of the detector and the amount of material traversed between the layers, cutoff values are determined for each detector layer combination individually. The values are chosen to achieve an efficiency of $\approx 80\,\%$. Due to space constraints we refer the reader to [11] for an elaborate account on cutoff value determination.

3.5. Triplet Joining

To derive more information about actual tracks in the event, it is necessary to find criteria to join compatible triplets. Compatible triplets are a combination of two or more valid triplets, which all originate from the same particle track. A valid combination of triplets t and t' is identified

Table 1. Overview of expensive operations performed per first hit, hit pair and triplet candidate, respectively. The number of grid cells within the predicted z-range is denoted by $n_{\text{cells},z}$; n_{recall} marks the number of hits within the predicted grid cells; † applies only if grid does *not* fit in local memory.

	Pair Generation	**Triplet Prediction**	**Triplet Filtering**
division/modulo	$12 + 2n_{\text{recall}}$	12	7
square root	1	10	5
trigonometric	$5 + n_{\text{recall}}$	$4 + n_{\text{recall}}$	4
global memory	$9 + 4n^\dagger_{\text{cells},z} + 4n_{\text{recall}}$	$16 + 4n_{\text{cells},z} + 4n_{\text{recall}}$	13
atomic	n_{recall}	0	1

by i) the triplets having two hits in common, ii) the absolute difference between the triplets' charge – plus or minus one – divided by the magnitude of their momentum being smaller than dp (Eq. 4), and iii) the change in direction between the triplets – as given by the difference of their trajectories' normal vectors – being lower than dx, (Eq. 5).

$$\left| \frac{q}{|\vec{p}|} - \frac{q'}{|\vec{p}'|} \right| \leq dp \qquad (4) \qquad\qquad \left|\vec{n} - \vec{n}'\right| \leq dx \qquad (5)$$

The first criterion can be checked in a very fast fashion and without any additional computations on the input triplets. To have the best possible estimation of the triplet's parameters for the second and third criterion, a Kalman filter-based track fit [12] is performed for each triplet, using only the three hits contained in the triplet.

4. Results

The evaluation of heuristic algorithms needs to address two aspects: the quality of the algorithm's results and the time required to produce them. In the case of particle track reconstruction, the quality of the produced results is described by the efficiency and fake rate as well as the clone rate – valid tracks found multiple times due to overlapping detector modules:

$$\text{Eff} = \frac{n_{\text{valid reconstructed}}}{n_{\text{simulated}}} \qquad \text{Fr} = \frac{n_{\text{fake tracks}}}{n_{\text{total reconstructed}}} \qquad \text{Cr} = \frac{n_{\text{clones}}}{n_{\text{total reconstructed}}}.$$

The run-time behavior is characterized by the kernel time – the time spent by the compute device executing the kernel – and the wall time – including the kernel time, data transfers, scheduling overhead etc.

The physics quality is assessed with 2 000 simulated QCD and $t\bar{t}$ events each, simulated at $\sqrt{s} = 14\,\text{TeV}$ with a minumum p_T of $1\,\text{GeV}\,c^{-1}$ using Pythia 6, version 4.26. The run-time behavior is evaluated with specifically tailored muon samples, featuring 1 to 4 096 muon tracks, originating at the transverse origin with p_T uniformly in $[1, 10]\,\text{GeV}$ and η uniformly distributed in $[-1, 1]$. An Intel Core i7 3930K CPU (Intel OpenCL SDK 2012) and NVIDIA GTX 660 GPU (NVIDIA driver version 319.23) are employed as compute devices.

Fig. 1 depicts the physics performance in $t\bar{t}$ events for different detector layer combinations. For all layer combinations an efficiency of $\approx 80\,\%$ is achieved in the barrel region, with a slight drop in efficiency for the 4-5-8 layer combination due to the long distance traversed between layer five and eight and the lower resolution in the outer silicon strip layers. These efficiencies are of similar order as the ones obtained by the initial seeding step of CMSSW. The fake rate is about 10 % in the pixel layers and rises to around 50 % to 60 % in the silicon strip layers. Due to the lower resolution in the strip detector – particularly in the z-coordinate – much larger cutoff

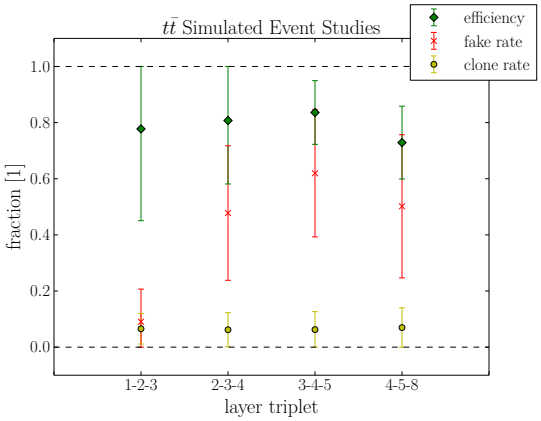

Figure 1. Physics performance in $t\bar{t}$ events.

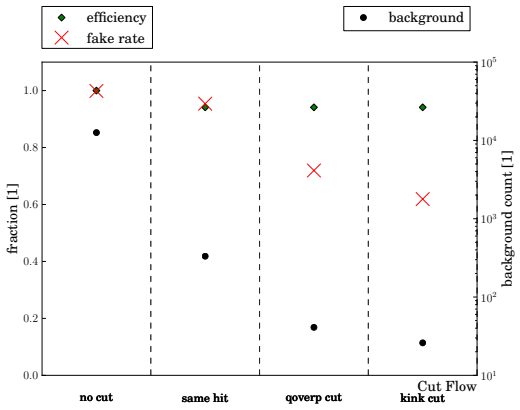

Figure 2. Physics performance of the triplet joining criteria.

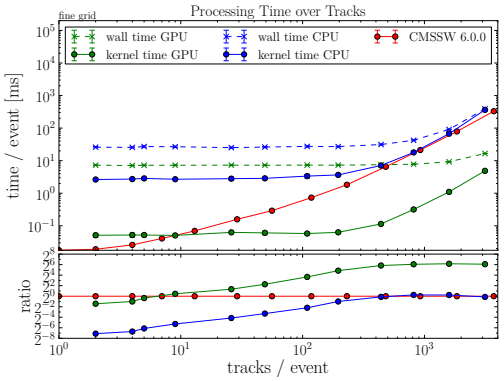

Figure 3. Run-time behavior over tracks per event in comparison to CMSSW.

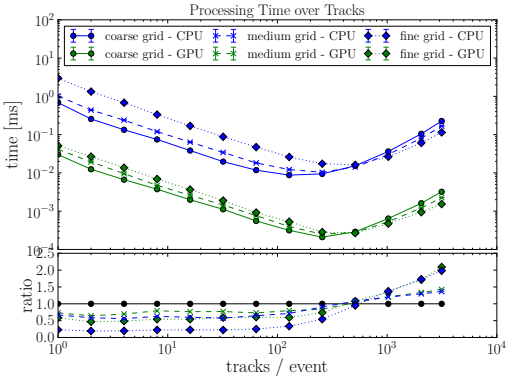

Figure 4. Run-time behavior over tracks per event depending on grid granularity.

values need to be employed for the filter criteria in order to maintain efficiency, refer to [11] for details.

For the investigation of the triplet joining, the criteria and fit method described in Sec. 3.5 have been implemented in C++ using the CMSSW framework. Cutoff values for criteria two and three have been derived from $t\bar{t}$ events. Fig. 2 depicts the efficiency, fake rate and the overall number of fake combinations passing each criterion (background count) for a combination of triplets resulting from a seeding step in layers 1-2-3 and layers 2-3-4. Using this fairly simple set of filters, a fake rate of 62 % can be achieved while preserving an efficiency for valid triplet combinations of 95 %.

Having established favorable quality results for the new algorithm, its suitability to address the performance challenge of 2015 is scrutinized. Fig. 3 depicts the run-time behavior over the number of tracks per event. The initial triplet seeding step of CMSSW 6.0.0 is used as a benchmark.[1] For events with only a few tracks, the OpenCL overhead due to scheduling, memory transfers and kernel invocation is very pronounced. Considering events with many tracks, the new algorithm achieves a similar performance on a CPU as CMSSW. On a GPU, a speedup of

[1] Note that CMSSW is a single-threaded application – thus only one CPU core is used – and performs many sophisticated calculations to address manifold influences on a particle's trajectory.

up to 64 can be achieved for events with more than 500 tracks, which is at the lower limit of the expected number of tracks per event for the increased LHC luminosity.

The granularity of the grid data structure is influencing the run-time behavior greatly, as exhibited by Fig. 4. The more fine-grained the grid, the more time is required for its construction – particularly as the data structure grows larger it does not fit in fast local memory of the CD anymore. However, this is mitigated by the reduced combinatorics in the pair generation and triplet prediction step, resulting in an individual sweet spot for every grid configuration. This suggests that the granularity should be adapted on the fly according to the expected number of tracks per event. Due to space constraints, the reader is referred to [11] for a more elaborate study of the presented algorithm's run-time behavior.

5. Conclusions and Future Work

The leap in luminosity of the LHC poses a tremendous challenge to track reconstruction algorithms due to increased number of simultaneous proton-proton collisions per event. This entails a higher combinatoric complexity, which must be met by exploiting parallelism on several levels. The presented algorithm performs triplet finding and joining for multiple events, processing the hits and detector layer combinations of them concurrently. Due to the diverse nature of the LHC computing grid, OpenCL is employed to exploit this parallelism in a transparent manner on a CPU as well as on a GPU.

The devised triplet finding algorithm achieves an efficiency of approximately 80 % on the studied $t\bar{t}$ sample, with fake rates ranging from below 10 to 60 percent, depending on the combination of detector layers under consideration. For events with high activity, performing the algorithm on a GPU attains a speedup of up to factor 64 compared to CPU execution. For the latter, a similar run-time as CMSSW is observed.

The studied criteria for triplet joining show promising fake rejection capabilities. As future work, these criteria need to be implemented in our OpenCL framework to assess their run-time behavior. Furthermore, the algorithm is currently limited to the barrel region of the detector and needs to be extended to also address the detector endcaps. Additional speedup and a significant reduction in fake rate can be expected by adapting CMSSW's iterative tracking approach – first reconstructing tracks from prompt, high-energetic particles with tight cutoff values and later addressing more difficult to find trajectories with looser criteria.

Acknowledgments

This work was supported by the German Doctoral Student Programme at CERN.

References

[1] Lange D 2011 The CMS Reconstruction Software Tech. Rep. CMS-CR-2011-002 CERN Geneva

[2] Klein K 2009 The Upgrade of the CMS Tracker for Super-LHC Tech. Rep. CMS-CR-2009-356 CERN Geneva

[3] ALICE Collaboration 2011 *IEEE Transactions on Nuclear Science* **58** 18451851 ISSN 0018-9499

[4] Eck C *et al.* (eds) 2005 *LHC computing Grid: Technical Design Report* Technical Design Report LCG (Geneva, CH: CERN)

[5] Khronos Group 2011 The Khronos Group Releases OpenCL 1.2 Specification press release

[6] Fang J, Varbanescu A and Sips H 2011 *ICPP* (New York, NY, USA: IEEE) p 216225 ISSN 0190-3918

[7] Shen J, Fang J, Sips H and Varbanescu A 2012 *ICPPW* (New York, NY, USA: IEEE) p 116125 ISSN 1530-2016

[8] Akman W, Franklin W, Kankanhalli M and Narayanaswami C 1989 *Computer-Aided Design* **21** 410420 ISSN 0010-4485

[9] Sengupta S, Harris M, Zhang Y and Owens J D 2007 *SIGGRAPH GH* (Aire-la-Ville, Switzerland, Switzerland: Eurographics Association) p 97106 ISBN 978-1-59593-625-7

[10] Frühwirth R, Strandlie A and Waltenberger W 2002 *Nucl Instrum Meth A* **490** 366378 ISSN 0168-9002

[11] Funke D 2013 *Parallel Triplet Finding for Particle Track Reconstruction* Master's thesis KIT Karlsruhe, DE

[12] Frühwirth R 1987 *Nucl Instrum Meth A* **262** 444450