ATLAS FTK challenge: simulation of a billion-fold hardware parallelism

J Adelman[1], L S Ancu[2], A Annovi[3], J Baines[4], D Britzger[5], W Ehrenfeld[6], P Giannetti[7], C Luongo[7], C Pandini[8], S Schmitt[5], G Stewart[9], L Tompkins[10], A Vaniachine[11] and G Volpi[7, 12] on behalf of the ATLAS Collaboration

[1] Northern Illinois University, DeKalb, Illinois, United States of America
[2] Université de Genève, Geneva, Switzerland
[3] Laboratori Nazionali di Frascati, Frascati, Italy
[4] STFC - Rutherford Appleton Laboratory, Oxford, UK
[5] Deutsches Elektronen-Synchrotron, Hamburg and Zeuthen, Germany
[6] Universitaet Bonn, Bonn, Germany
[7] INFN Sezione di Pisa, Pisa, Italy
[8] LPHNE, Paris, France
[9] University of Glasgow, Scotland, UK
[10] Stanford University, Stanford, California, United States of America
[11] Argonne National Laboratory, Argonne, Illinois, United States of America
[12] Università di Pisa, Pisa, Italy

E-mail: vaniachine@anl.gov

Abstract. During the current LHC shutdown period the ATLAS experiment will upgrade the Trigger and Data Acquisition system to include a hardware tracker coprocessor: the Fast TracKer (FTK). The FTK receives data from the 80 million of channels of the ATLAS silicon detector, identifying charged tracks and reconstructing their parameters at a rate of up to 100 KHz and within 100 microseconds. To achieve this performance, the FTK system identifies candidate tracks utilizing the computing power of a custom ASIC chip with associative memory (AM) designed to perform "pattern matching" at very high speed; track parameters are then calculated using modern FPGAs. A detailed simulation of this massive system has been developed with the goal of supporting the hardware design and studying its impact in the ATLAS online event selection at high LHC luminosities. We present the issues related to emulating FTK on a general-purpose CPU platform, using ATLAS computing Grid resources, and the solutions developed in order to mitigate these problems and allow the studies required to support the system design, construction and installation.

1. Introduction

In 2015 LHC Run 2 will reach centre of mass energies above 13 TeV and instantaneous luminosities exceeding $2 \cdot 10^{34}$ cm^{-2}s^{-1}. The physics goals of Run 2 will comprise high precision tests of the Standard Model and the Higgs sector and searches for physics beyond the Standard Model. These goals require detailed simulation of the expected physics and detector behaviour, in order to optimize the event selection at the trigger level and at the analysis level.

1.1. Run 2 Challenges

Discoveries and precision studies of rare physics events require rejection of “known” events by more than ten orders of magnitude. To meet the Run 2 challenges, the ATLAS experiment [1] will have a two-tier trigger system comprised of the hardware-based Level 1 trigger and the software-based High-Level Trigger (HLT) [2].

The Run 2 increase in instantaneous luminosities complicates the trigger rejection because of the presence of a large number of simultaneous collisions in the same event, overlapping the hard scatter of interest (such as $Z \rightarrow \mu\mu$, see figure 1). This phenomenon is usually called “pileup”. The Run 1 experience demonstrated that tracking performed within the full acceptance of the ATLAS silicon detectors in real-time is a powerful tool to limit pileup effects on the trigger performance. However, the use of software algorithms to perform track reconstruction in the full acceptance of the silicon detector in real-time on general-purpose CPUs would be too slow for the HLT. The Run 2 conditions therefore require a new approach to provide full acceptance tracking at the Level 1 trigger rate.

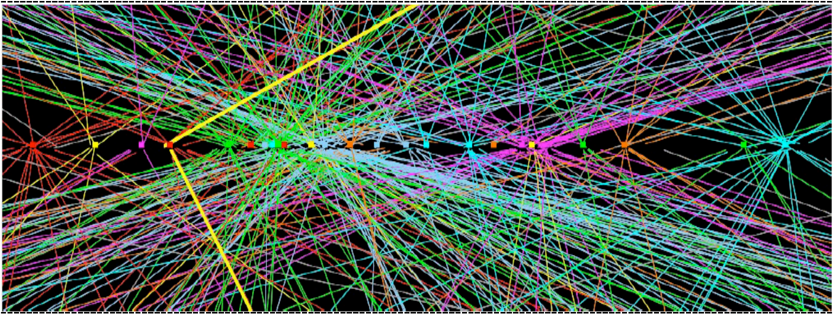

Figure 1. A visual example of pile-up in the ATLAS tracker: a Run 1 $Z \rightarrow \mu\mu$ event collected at an instantaneous luminosity $L = 0.5 \text{x} 10^{34}$ cm^{-2}s^{-1} in 8 TeV *pp* collisions. Two thick yellow lines show muon tracks from the Z final state, triggered among pileup events.

1.2. Silicon Detectors

In real-time, tracks will be measured in ATLAS using silicon detectors: the silicon Pixel detector and the silicon strip "SemiConductor Tracker" (SCT). To better handle the high-luminosity environment of the LHC Run 2, the Pixel detector is being upgraded with an additional Insertable B-Layer (IBL) [3]. The IBL brings the total number of layers in ATLAS silicon detectors to twelve: four Pixel detector layers and eight SCT layers.

2. Fast TracKer (FTK)

To provide reconstructed tracks to the High Level Trigger, ATLAS will use the Fast TracKer (FTK), implemented between the Level 1 trigger and the HLT. The FTK is a hardware device built with custom electronics [4] to reconstruct charged particles with $p_T > 1$ GeV in the full silicon-detector acceptance for every event accepted by the Level 1 trigger.

FTK allows the HLT to use global track properties for trigger selection. Additionally, because tracks are available at the start of HLT processing, they can be used for pre-selection of events or objects that more sophisticated algorithms will further refine for ultimate selection. FTK tracks will be widely used in most triggered events, as selectively refitted FTK tracks in HLT allow high quality tracking at a reasonably low CPU cost and low latency. The selections based on FTK tracks will be one of ATLAS’s best mitigations against pileup and the prohibitive time of full acceptance tracking at Run 2 luminosity.

2.1. Match and Fit

Conceptually, the FTK system reconstructs charged tracks in *pp* collisions in two steps: first comes “pattern matching”, i.e. the identification of suitable detector hits combinations; then track parameters are fitted (“track fitting”) based on the coordinates of the identified hit patterns. Pattern matching is performed using the capability of finding correlations in the data using Associative Memory (AM) ASICs loaded with about one billion of possible patterns, while the track parameters are calculated by implementing fast fit on an FPGA. In both steps a high degree of parallelism is used. The data flow of the pattern-matching step has preliminary sub-steps of hits clustering (with dedicated input FPGAs) and parallel distribution of oncoming tracker data.

3. FTK Simulation

The entire FTK data processing pipeline has to be simulated to support the hardware design and to develop trigger strategies at high luminosity. These two goals provide complementary requirements. The FTK performance studies require many millions of events, therefore, the efficient use of CPU is important. The FTK design studies require a smaller data sample and accurate, but flexible, emulation so that the impact of FTK design choices is matched by appropriate performance studies.

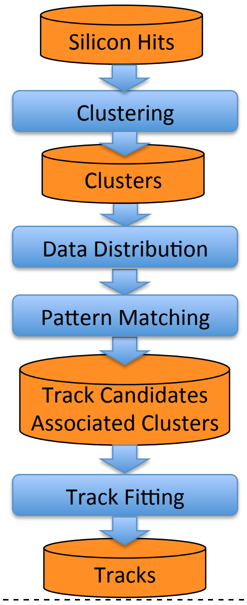

Figure 2. The FTK simulation structure reproduces all steps of the entire FTK data processing pipeline. Orange containers illustrate the information being passed to the next stage.

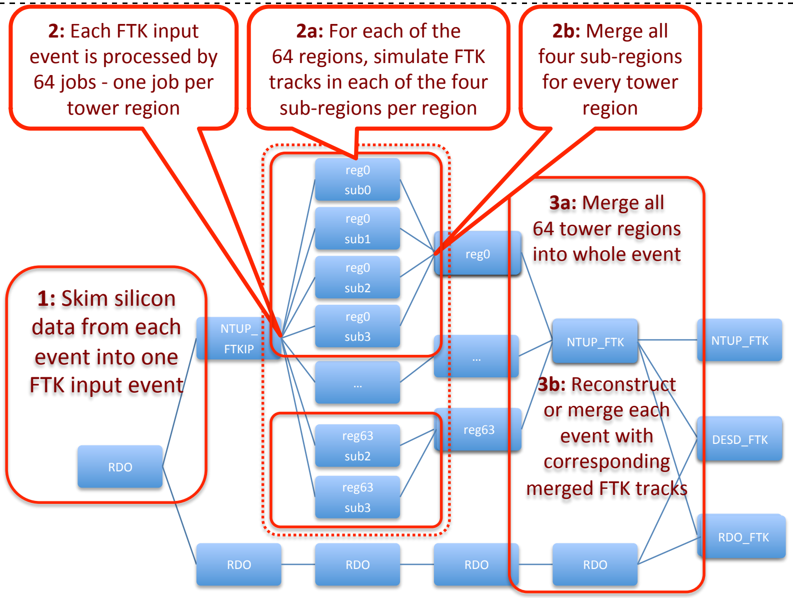
2: Each FTK input event is processed by 64 jobs - one job per tower region
2a: For each of the 64 regions, simulate FTK tracks in each of the four sub-regions per region
2b: Merge all four sub-regions for every tower region
3a: Merge all 64 tower regions into whole event
1: Skim silicon data from each event into one FTK input event
3b: Reconstruct or merge each event with corresponding merged FTK tracks
reg0 sub0
reg0 sub1
reg0 sub2
reg0 sub3
...
reg63 sub2
reg63 sub3
reg0
...
reg63
NTUP_ FTKIP
RDO
NTUP_FTK
NTUP_FTK
DESD_FTK
RDO_FTK
RDO
RDO
RDO
RDO

Figure 3. The FTK simulation splits every event into 256 η-ϕ sub-regions of each event. In the three-step workflow (1-3), each event is processed by 64 jobs. To keep the required resources below practical Grid limits, each job simulates FTK tracks in four sub-regions (sub0-sub3) one after another (2a). The sub-region merging is done in two steps: (2b) producing whole regions reg0-reg63 and (3a) whole events in the NTUP_FTK files. The final step (3) uses FTK tracks in trigger simulations producing the reconstructed data in DESD_FTK files or adds FTK tracks to the simulated events in the RDO_FTK files.

3.1. Algorithms

To achieve both goals, the FTK simulation closely emulates the hardware. All steps in the FTK data flow are reproduced with the possibility of evaluating the impact of changes in the FTK design and/or parameters. To provide input data for the FTK simulation, the silicon detector hits are skimmed, either from real data (RAW) or from simulated events (RDO), into custom ROOT files (NTUP_FTKIP). Figure 2 shows the FTK simulation data flow steps:

- Clustering: identify the hit coordinates based on the firing detector elements (strips or pixels), then simulates the custom clustering algorithm running on input FPGAs [5].

- Data Distribution: simulate geometrical distribution of clusters to the parallel processing units (FTK towers) to be matched with the track patterns in the AM.
- Pattern Matching simulation: read the FTK patterns and find coincidences with clusters in an event. Due to hardware constraints, FTK pattern matching is restricted to eight silicon detector layers: three Pixel, four SCT axial and one SCT stereo layers. The step produces the list of track candidates and the associated clusters.
- Track Fitting simulation: read found track candidates and search for good tracks among the sequence of clusters selected by the previous stage. Extrapolate the good tracks in the additional four silicon detector layers. Find nearby hits in those layers, refit using clusters in at most twelve layers improving the fit and minimizing the chance of track duplication. Collect tracks in the whole event and reject duplicates.

3.2. Challenge

The Associative Memory (AM) ASIC uses a content addressable memory architecture where any data inquiry is broadcast to all memory elements simultaneously. As a result, each incoming hit reaches all one billion patterns in the whole FTK AM system within the same 10 ns clock cycle. This feature cannot be matched on a system utilising general-purpose CPUs. Similarly, the track fitting simulation on a single general-purpose CPU is at least one million times slower than the 1 GHz rate obtained in the FPGAs on the FTK fit boards. The storage - in simulation - of one billion patterns requires about 35 GB of memory. There are also two million fit coefficients that require about 2 GB of simulation storage, in addition to smaller configuration data. For these reasons, the simulation of the FTK is resource intensive and has inherent performance bottlenecks due to the limited parallelism on general-purpose CPUs and low bandwidth for memory access[1].

3.3. Workflow

In ATLAS Monte Carlo simulation workflow, the FTK simulation is a part of the trigger simulation step integrated with the data reconstruction step that produces reconstructed data in ESD files. Use of Grid computing for ATLAS Monte Carlo simulations imposes several practical constraints on the implementation of the FTK simulation:

- The FTK simulation application has to be single-threaded and/or single-process.
- The memory available to the application is limited to 2 GB per CPU-core.
- The application execution (the batch job duration) is limited to hours.
- No more than a hundred input files and/or several GB of configuration data can be used per job.

To overcome these practical constraints we took advantage of the η-ϕ segmentation of the FTK hardware into 64 tower regions. Each of the tower regions then splits into four sub-regions (thus we have 256 sub-regions in total). This resulted in a natural parallelization of the FTK simulations into 256 independent steps, with each step processing one of the η-ϕ sub-regions of each event (figure 3). The configuration files for each step are also split into 256 parts, corresponding to the η-ϕ sub-regions, and thus reducing the data required by each step. In practice, the application processes all four sub-regions of the same tower sequentially in the same job. Thus, 64 jobs are used to process the same event in parallel.

The split workflow requires merging of the sub-regions simulation output in the whole event. To reduce the number of input files per merging job, the merging is done in two steps:

- in the first step, four files from each sub-region of one tower are merged into one file;
- in the second merging step, the 64 files for every tower are merged to produce the whole event.

To reduce the overall number of steps in the FTK simulation workflow, both merging steps are joined with adjacent workflow steps (figure 3): the first merging step is done at the end of the FTK

[1] In contrast to the best DRAM transfer rates of 25 GB/s, the AM system benefits of a bandwidth of about 25 TB/s.

simulation step, while the second merging step is done at the beginning of the data reconstruction step, which includes the trigger simulation using FTK tracks. Thus, the data reconstruction step uses 64 files from each tower region of the same event plus the RDO or RAW data file of the same event (i.e. the file that was used to skim the silicon detector hits as inputs to the FTK simulation).

As a result, single workflow jobs can be sequentially executed on a single Grid node or distributed concurrently to different Grid nodes. The workflow reduces the resources required by the FTK simulation application through increasing parallelism. The FTK simulation has been integrated into the Monte Carlo chain run by the ATLAS production system [6, 7]. The simulation has run smoothly and has so far produced 9.6 M fully simulated events. The next production of a multi-million event sample is in progress.

3.4. Performance

FTK simulation increases the ATLAS full Monte Carlo chain CPU usage by 15-30%. For a typical Grid node the total execution time is about 300 s/event. The AM simulation takes two thirds of the execution time. The execution time increases linearly with increasing LHC luminosity. The memory required for a single job fits below 1 GB.

4. Conclusions

The FTK is an important upgrade of the ATLAS trigger and data acquisition system. The hardware, based on custom AM ASICs and FPGAs, enables fast track reconstruction at Level 1 trigger rate. As the LHC luminosity increases, the availability of tracks from the FTK to the ATLAS High Level Trigger will ensure that the excellent physics performance of the ATLAS detector can be maintained.

Full simulation of the FTK is important for both the development of the hardware design and for the development of trigger algorithms making use of tracks provided by the FTK. For the FTK simulation we have developed solutions to run the FTK simulations on the Grid, overcoming many practical limitations, such as CPU power and memory limits. To cope with these limitations, event processing has been split into the parallel processing of detector sub-regions. This solution has allowed us to make use of Grid computing resources to produce millions of physics events to study the expected performance of the FTK system.

Acknowledgments

We thank all our colleagues who contributed to the FTK development. We are grateful to A. Cerri for fruitful discussions. This work was funded in part by the U. S. Department of Energy, Office of Science, High Energy Physics under Contract No. DE-AC02-06CH11357.

References

[1] The ATLAS Collaboration 2008 The ATLAS experiment at the CERN Large Hadron Collider *J. Inst.* **3** S08003

[2] Bartoldus R *et al.* 2013 *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*, CERN-LHCC-2013-018, ATLAS-TDR-023

[3] The ATLAS Collaboration 2010 *ATLAS Insertable B-Layer Technical Design Report*, CERN-LHCC-2010-013, ATLAS-TDR-019

[4] Annovi A *et al.* 2013 *ATLAS Fast TracKer (FTK) Technical Design Report*, CERN-LHCC-2013-007, ATLAS-TDR-021

[5] Sotiropoulou C-L *et al.* 2014 A Parallel FPGA Implementation for Real-Time 2D Pixel Clustering for the ATLAS Fast TracKer Processor, *Preprint* ATL-DAQ-PROC-2014-022, submitted to *Proc. Int. Conf. Technology and Instrumentation in Particle Physics (TIPP'14)*

[6] Golubkov D *et al.* 2012 ATLAS Grid Data Processing: system evolution and scalability *J. Phys.: Conf. Ser.* **396** 032049

[7] Borodin M *et al.* 2014 Multilevel workflow system in the ATLAS experiment, *Preprint* ATL-SOFT-PROC-2014-005, in *these proceedings*