# Table of Contents

# ATLAS Edinburgh GPU Computing

# High Level Trigger Studies

## Porting the HLT Z-finder algorithm to GPU

**Chris Jones, Andrew Washbrook**

### Project Overview

With around a billion particle collisions occurring every second within the detector, it is impossible to store the data collected by ATLAS in its entirety. It is responsibility of the trigger to reduce this data, selecting out events of potential scientific interest for storage and further analysis. The works in three main stages: level 1 running on customised hardware, and levels 2 and 3 in software on server farms. IDScan is a track-reconstruction program running in level 2 of the trigger, and taking its input from the inner detector. The first algorithm used by IDScan is the Z-finder algorithm, the purpose of which is to produce a good estimate for the z-coordinate of the collision event along the axis of the detector. This reduces the execution time of the track-reconstruction algorithms later in the chain.

The aim of this project will be to port the Z-finder algorithm to GPU using NVIDIA's C for CUDA, and investigate the performance.

### Project Report

- **GPUs and the ATLAS experiment** (pdf)

### Project Code

- Z-Finder GPU version (tar.gz)

## SIMT design of the High Level Trigger Kalman Fitter

**Maria Rovatsou, Andrew Washbrook**

### Project Overview

The track reconstruction in the ATLAS trigger is achieved with the utilization of the Kalman filter technique. The current implementation of the filter is a distributed sequential implementation. An acceleration of the filter would boost the whole process and would allow even more significant data to be available for processing. Apart from that, the proposed upgrade of the LHC in order to increase its performance, results in luminosity increase of one order of magnitude higher and would require a more efficient implementation of the Kalman filter. A possible way for acceleration of the filter is by porting it on a Graphics Processing Unit (GPU), which has immense computational power that can be exploited through programming models as CUDA. The proposed approach is a parallelization on thread level of the Kalman filter for execution on an

NVIDIA GPU. The proposed programming model is CUDA and the analysis of its performance will be done with the CUDA profiling tools and with benchmarking tools.

### Project Report

- **Analysis Acceleration on GPUs for the ATLAS Experiment at CERN** (pdf)

### Project Code

- Kalman Fitter SIMT Design (tar.gz)
- Kalman Fitter for CUDA (Dimitry Emeliyanov) (tar.gz)

## HLT Project Presentations

- **Processing Petabytes per Second with the ATLAS Experiment at the Large Hadron Collider in CERN**, Nvidia GPU Technology Conference September 2010 (stream, pdf)
- **Algorithm Acceleration from GPGPUs for the ATLAS Upgrade**, Computing in High Energy and Nuclear Physics October 2010 (pdf)
- **GPU-based tracking for ATLAS Level 2 Trigger** Parallelism in Experimental Nuclear Physics Workshop January 2011 (pdf)

## HLT Resources

### Software

- IDScan CERN wiki page
- IDScanGPU CERN wiki page
- IDScan Doxygen docs

### Articles and Preprints

- **Determination of the z position of primary interactions in ATLAS prior to track reconstruction** (pdf) Nikos Konstantinidis and Hans Drevermann ATLAS-DAQ-2002-014
- **Tracking at level 2 for the ATLAS high level trigger** (pdf) M Sutton, Nuclear Instruments and Methods in Physics Research A 582 (2007)
- **Fast SIMDized Kalman filter based track fit** (pdf) S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, W.F.J. Miller Computer Physics Communications 178 (2008)
- **IDSCAN: LVL2 Tracking for the ATLAS Trigger** (pdf) Janice Drohan UCL 1st Year Transfer Report June 20, 2003
- **Fast Tracking for the ATLAS LVL2 Trigger** (pdf) J. T. M Baines, H. Drevermann, D. Emeliyanov, N. Konstantinidis, F. Parodi, C. Schiavi, M Sutton, Computing in High Energy Physics and Nuclear Physics 2004, Interlaken, Switzerland, 27 Sep - 1 Oct 2004, pp.246
- **HLT Track Reconstruction Performance** (pdf) S. Ask, J.T. Baines, A. Coccaro, D. Emeliyanov, I. Grabowska-Bold, J. Kirk, N. Konstantinidis, J. Masik, E. Ozcan, F. Parodi, C. Schiavi, S. Sivoklokov, M Sutton Detectors and Experimental Techniques, 17 Mar 2009. - 19 p.

- **A Probabilistic Data Association Filter for fast tracking in the ATLAS Transition Radiation Tracker** (pdf) D. Emeliyanov Nucl. Instrum. Meth. A566 (2006) 50-53.
- **ATLAS Technical Design Report: High-Level Trigger, Data Acquisition and Controls** (pdf) ATLAS HLT/DAQ/DCS Group ATLAS-TDR-016 2 October, 2003
- **Commissioning the ATLAS Inner Detector Trigger** (pdf) Mark Sutton on behalf of the ATLAS Collaboration ATL-DAQ-PROC-2009-015
- **Overview of the High-Level Trigger Electron and Photon Selection for the ATLAS Experiment at the LHC** (pdf) IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 53, NO. 5, OCTOBER 2006
- **Fast tracking in hadron collider experiments** (pdf) Nikos Konstantinidis and Hans Drevermann
- **Vertexing strategy and algorithms at ATLAS** (pdf) F. Parodi, Nuclear Instruments and Methods in Physics Research A 560 (2006)
- **ATLAS L1 track trigger for super-LHC** (pdf) Nikos Konstantinidis ATL-UPGRADE-PROC-2010-003
- **ATLAS Track Trigger for SLHC - Ideas & Plans** (pdf) Nikos Konstantinidis ATL-UPGRADE-SLIDE-2009-243
- CERN Document Server search ATL-DAQ-PROC
- CERN Document Server search ATL-DAQ-SLIDE
- Trigger Wiki
- Trigger Workshop 2009 Wiki

# Tracking Studies

## Tracking of particles in Electromagnetic Field using Runge-Kutta method

**James Henderson, Phil Clark**

### Project Overview

Over recent years Graphical Processing Units (GPUs) have demonstrated their great ability in scientific calculations. They have the capability to carry out tasks in parallel, enabling huge speed-ups of calculations. This project investigates the potential speed-up using GPUs when calculating a particle's trajectory through a magnetic field. The results show that, when considering many particles, a speed-up of 32x can be achieved on the GPU versus a serial processor.

### Project Report

- **An Investigation Into Particle Tracking and Simulation Algorithms Using GPUs** (short version) (pdf)
- **An Investigation Into Particle Tracking and Simulation Algorithms Using GPUs** (long version) (pdf)

### Project Code

- RK4 stepper code used to compare timings (.cpp)
- CUDA code for RK4 stepper (.cu)

## Tracking Resources

- Geant4 Home Page
- G4 Tutorials
- G4 Source Code Browser
- ParGeant4 Home Page - Parallel Geant4 Version make by Gene Cooperman (Institute for Complex Scientific Software)
- CERN's ParGeant4 page - clear explaination
- Mixing C and C++ - useful for combining G4 (C++) and CUDA (C)
- G4 (4.9.3.p01) Directory Reference
- G4 Application Developers User Guide
- G4 Software Reference Manual

- All source code for the exercises is available here: tar.gz

## Simple

- Source code: simple.cpp
- To compile: g++ -o simple.exe -fopenmp simple.cpp

- Modify simple.cpp to use OpenMP directives to enable multiple threads to execute the cout statement in parallel.
- Modify the cout statement to print the ID of the calling thread in the parallel region.
- Fix the number of threads to 4 using the appropriate OpenMP environment variable.
- Repeat the above but instead use an appropriate OpenMP runtime function.
- Why is the output always mangled? (Note: you can fix this in a later example)
- Why is the thread index not in sequential order?

## Schedule

- Source code: src/main.cpp
- To compile: run make
- To execute: bin/schedule

- This code generates a configurable sawtooth-like workload (i.e a load imbalance between threads) to allow the relative performance of OpenMP loop schedule types to be evaluated.
- Use OpenMP directives to parallelise the loop contained in the main section of the code.
- Compile and measure the execution time using 1, 2, 4 and 8 threads (using the timer provided)
- Capture the thread allocation pattern by iterating the number of calls made by each thread into a suitable array. Display the array contents to show the distribution of work amongst threads for each schedule type
- Apply different schedule types and chunksizes to the loop construct to demonstrate the effect on execution time. Which type did you find to be optimal?
- Modify the SAW_LENGTH and SAW_WIDTH variables to study how schedulers perform under different load imbalances.

## Triangular

Triangular matrix addition is performed on 2 square equal matrices of equal size 20 x 20

- Source code: src/main.cpp
- To compile: run make
- To execute: bin/triangular

- Parallelise the outermost loop and run with an increasing number of threads.

- How does the code perform compared to the serial version? Experiment with larger matrix sizes and observe the relative timing measurements.
- Enable the choice of schedule used in the loop to be determined at runtime. How does dynamic compare to static (and guided)?
- Add a new variable in the loop to sum up all the array elements of Result. Ensure that the total value is consistent with repeated execution of the code.

## Mandlebrot

- Example from EPCC Advanced OpenMP course

The code generates a grid of points in a box of the complex plane containing the upper half of the (symmetric) Mandelbrot Set. Then each point is iterated using the equation above a finite number of times (2000). If within that number of iterations the threshold condition |z| > 2 is satisfied then that point is considered to be outside of the Mandelbrot Set. Then counting the number of points within the Set and those outside will give an estimate of the area of the Set.

- Source code: area.c
- To compile: g++ -o area -fopenmp area.c
- Parallelise the outer loop using a parallel for directive and declare all shared, private and reduction variables
- Check results consistency and alter schedule clause to measure performance across an increasing number of threads
- Rewrite this example using OpenMP tasks. You could try any of the following methods:
    - Make the computation of each point a task, and use one thread only to generate the tasks.
    - Each row of points is a task
    - All threads generate tasks

## Simple revisited

- Get a copy of simple.cpp from the previous session and ensure that the hello statement is printed out one at a time (but not necessarily in order).

## ATLAS use case: Clusterisation example

- Provided by Ben Wynne
- Apply OpenMP methods discussed in the slides and the exercises above to a simple clusterisation algorithm (i.e. a method of assembling adjacent deposits of charge in a detector into space points)
- The example algorithm and optimisation methods are discussed in detail here: slides

# Analysis Tools Studies

## Analysis Acceleration in TMVA using GPU Computing

**Hazel McKendrick, Andrew Washbrook**

### Project Overview

This project forms a feasibility study into the use of GPU (Graphics Processing Unit) devices to parallelise TMVA, the Toolkit for Multi Variate Analysis, to determine whether such techniques might lead to future performance gains for the framework.

In particular, the Multi-layer perceptron, a class of neural network, is ported to the GPU programming platform CUDA. Performance when training single networks is generally comparable to CPU performance but show promise for future improvement. However, this parallelism of the GPU also allows multiple networks to be trained simultaneously, and this is leveraged to show significant performance gains over under- taking such a task in serial. The challenges and potential for these results to be applied across the TMVA framework is then considered and discussed.

### Project Report

- **Analysis Acceleration in TMVA for the ATLAS Experiment at CERN using GPU Computing** (pdf)

### Project Code

- Modified TMVA code containing GPU-based MLP method (.tar.gz)

# General GPU programming resources

Links to CUDA and other GPU parallisation resources:

- NVIDIA's CUDA training page
- Illinois University Parallel Course

This university course link has some very helpful video lectures, linked in 'Syllabus/ Lectures' and under 'Materials'.

- NVIDIA's Online Seminars Page Previously recorded seminars can be found at the bottom of the page.
- Italian GPU HPC course suggested by Gene Cooperman
- OpenCL Video Lectures As far as I can tell, to date OpenCL only runs on Apple Macs as it was developed by Apple.
- Advice on integrating C++ with CUDA
- CuPP Home Page - Framework for C++/CUDA integration
- Jens Breitbart paper on CuPP

---

This topic: Main > AtlasEdinburghGPUComputing
Topic revision: r23 - 2014-09-12 - AndrewWashbrook