Updates

Jyoti Prakash Biswal

Rutherford Appleton Laboratory

SWIFT-HEP #6

Fry Building, University of Bristol

21-22 November 2023

Science and Technology Facilities Council

Rutherford Appleton Laboratory

SWIFT HEP

GridPP UK Computing for Particle Physics

ATLAS EXPERIMENT

CPU vs GPU throughput studies w/ FastTrackFinder

Introduction

Set-up

- Athena/23.0.11 stable release for both athena and **athenaHLT**.
- CUDA configuration: /cvmfs/sft.cern.ch/lcg/cuda/11.7.1-d8e95/x86_64-centos7-gcc11-opt.
- **Package(s) checked out:** TrigFastTrackFinder.
- **GPU** – 2x NVIDIA TITAN V; *hepacc01* (@RAL); 64 GB RAM; 5120 CUDA Computing Cores.

FastTrackFinder (FTF) acceleration

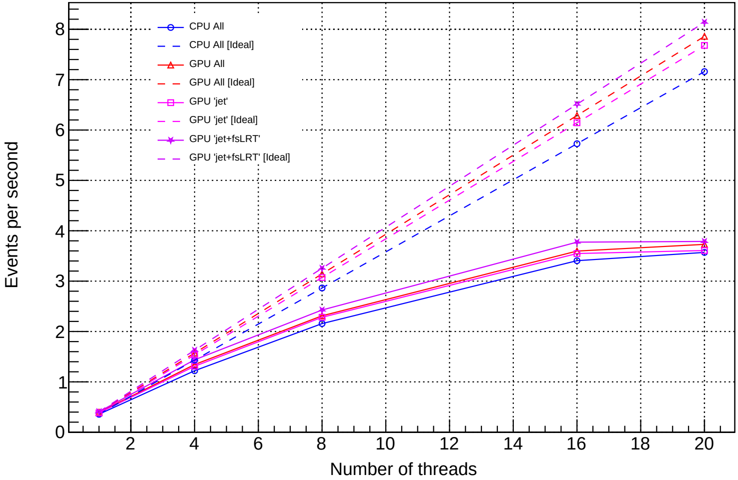
- FastTrackFinder is an HLT algorithm for reconstructing tracks in the detector.
- Works in several steps:
  - Search for triplets of space points (track seeds) in regions of interest (RoIs) from L1.
  - The combinatorial *Kalman filter* extends seeds in search roads and builds track candidates.
  - Another step to remove duplicate tracks sharing more than a certain number of hits.
- SeedMaker has been implemented on GPU – enabled by FastTrackFinder property, *useGPU=True*.
- At the time of this study, there were code differences between CPU and GPU implementations: **phi0 filtering**.
  - phi0 filter only selects triplets inside RoI.

    $\Rightarrow$ Reduces no. of seeds for RoI, hence the seed-making timing.
  - CPU code (already implemented).
  - GPU code (*to be implemented*) – not done yet.

Throughput plots

- Maximum events: 1000.
- Number of threads: 1, 4, 8, 16, 20.
- Parameter: *events per second*.
- Tests configured:
  - All FastTrackFinder algorithm instances for CPU: CPU All.
  - All FastTrackFinder algorithm instances for GPU [*i.e.*, w/ useGPU=True]: GPU All.
  - Only 'jet' FastTrackFinder algorithm instance for GPU [*i.e.*, w/ useGPU=True]: GPU 'jet.'
  - Both 'jet' and 'fullScanLRT' algorithm instances for GPU [*i.e.*, w/ useGPU=True]: GPU 'jet+fsLRT'.
- All these tests are performed for **athena** and **athenaHLT**.

Ideal: number of threads × events per second for thread=1.



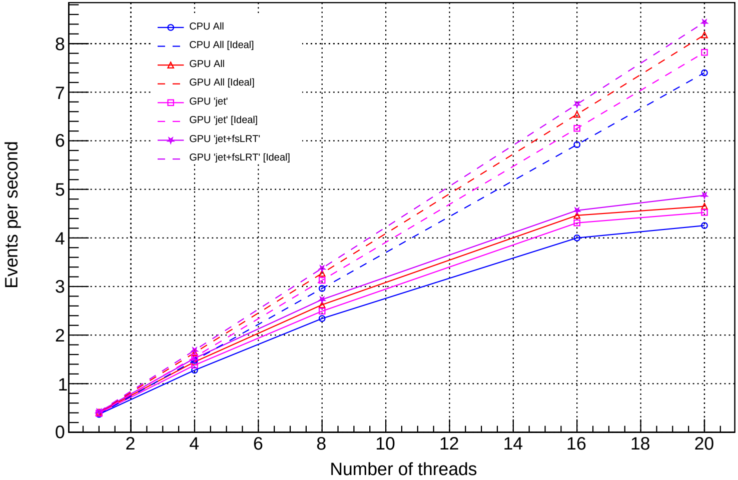
Speed up is observed for GPU. After thread=16, the jump is not that high. All cases are quite below w.r.t. their ideal scenarios for thread>4.

Total time(s): athena

TIME_Total [ms]; thread=1; TrigFastTrackFinder_				
Algorithm	**CPU All**	**GPU All**	**GPU 'jet'**	**GPU 'jet+fsLRT'**
DJetLRT	0 [not currently understood]	20.0 ± 0.0	17.0 ± 0.0	15.0 ± 0.0
DVtxLRT	171.806 ± 16.932	169.343 ± 13.042	173.963 ± 17.076	174.231 ± 17.054
bjet	4.711 ± 0.091	11.309 ± 0.09	4.772 ± 0.091	4.819 ± 0.092
bmumux	117.462 ± 12.065	143.625 ± 12.144	118.231 ± 12.245	117.872 ± 12.103
electron	1.837 ± 0.036	13.498 ± 0.098	1.869 ± 0.036	1.888 ± 0.036
electronLRT	7.822 ± 0.931	22.911 ± 1.945	7.889 ± 0.936	7.889 ± 0.928
fullScanLRT	**503.837 ± 11.792**	**339.683 ± 7.491**	**508.734 ± 11.919**	**338.216 ± 7.41**
jet	**671.869 ± 13.006**	**449.179 ± 8.231**	**448.573 ± 8.302**	**449.111 ± 8.34**
jetSuper	36.923 ± 1.696	48.787 ± 2.445	37.087 ± 1.702	37.206 ± 1.717
muon	5.233 ± 0.469	23.876 ± 1.243	5.256 ± 0.463	5.264 ± 0.469
muonIso	4.083 ± 0.54	11.542 ± 0.651	4.458 ± 0.556	4.25 ± 0.554
muonIsoMS	2.667 ± 0.72	10.0 ± 0.943	2.333 ± 0.544	2.667 ± 0.72
muonLRT	8.627 ± 1.091	20.441 ± 1.883	8.78 ± 1.074	8.881 ± 1.113
tauCore	8.352 ± 0.307	20.818 ± 0.421	8.436 ± 0.307	8.416 ± 0.308
tauIso	5.811 ± 0.148	11.916 ± 0.176	5.863 ± 0.148	5.9 ± 0.151
tauIsoBDT	5.691 ± 0.312	11.135 ± 0.321	5.736 ± 0.316	5.697 ± 0.31
tauLRT	23.834 ± 1.14	33.143 ± 1.387	23.897 ± 1.148	23.931 ± 1.138
Total	**1580.565 ± 27.348**	**1361.206 ± 21.444**	**1382.877 ± 25.686**	**1211.238 ± 23.862**

Throughput: athenaHLT

Ideal: number of threads × events per second for thread=1.



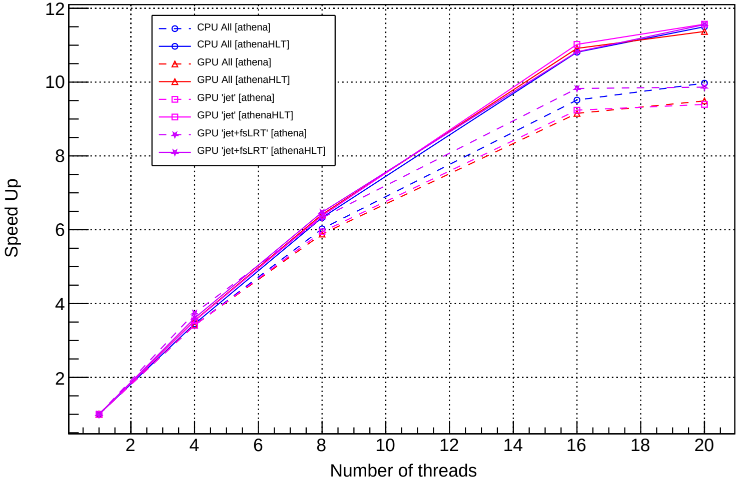
Speed up is observed for GPU. After thread=16, the jump is not that high. All cases are quite below w.r.t. their ideal scenarios for thread>4. Similar observations as of athena.

Total time(s): athenaHLT

TIME_Total [ms]; thread=1; TrigFastTrackFinder_				
Algorithm	**CPU All**	**GPU All**	**GPU 'jet'**	**GPU 'jet+fsLRT'**
DJetLRT	0 [not currently understood]	20.0 ± 0.0	15.0 ± 0.0	16.0 ± 0.0
DVtxLRT	171.852 ± 16.968	170.657 ± 13.086	177.287 ± 17.462	179.056 ± 17.696
bjet	4.758 ± 0.091	11.455 ± 0.098	4.901 ± 0.094	4.934 ± 0.095
bmumux	117.487 ± 12.028	147.4 ± 12.858	119.333 ± 12.341	119.333 ± 12.216
electron	1.887 ± 0.036	13.582 ± 0.105	1.928 ± 0.037	1.951 ± 0.038
electronLRT	7.889 ± 0.946	23.978 ± 1.929	8.133 ± 0.979	7.956 ± 0.964
fullScanLRT	502.263 ± 11.72	346.132 ± 7.638	519.364 ± 12.175	343.848 ± 7.545
jet	666.223 ± 12.893	451.492 ± 8.275	450.266 ± 8.369	450.435 ± 8.344
jetSuper	36.869 ± 1.686	48.593 ± 1.53	37.381 ± 1.707	37.521 ± 1.716
muon	5.295 ± 0.467	24.395 ± 1.27	5.403 ± 0.472	5.558 ± 0.502
muonIso	4.167 ± 0.542	11.692 ± 0.538	4.25 ± 0.544	4.25 ± 0.541
muonIsoMS	2.667 ± 0.72	10.0 ± 0.943	2.667 ± 0.72	2.333 ± 0.544
muonLRT	8.915 ± 1.107	21.0 ± 1.855	8.915 ± 1.095	8.949 ± 1.111
tauCore	8.377 ± 0.306	21.226 ± 0.444	8.626 ± 0.322	8.534 ± 0.31
tauIso	5.871 ± 0.148	12.105 ± 0.186	5.996 ± 0.155	5.997 ± 0.152
tauIsoBDT	5.719 ± 0.31	11.219 ± 0.351	5.837 ± 0.315	5.848 ± 0.314
tauLRT	23.714 ± 1.137	34.16 ± 1.409	24.406 ± 1.201	24.543 ± 1.177
Total	1573.953 ± 27.270	1379.086 ± 21.064	1399.693 ± 26.139	1227.046 ± 24.423

Speed Up: athena and athenaHLT

Speed Up: $\frac{\text{events per second for a particular number of threads}}{\text{events per second for thread=1}}$



In most cases, athena speed up (broken lines) $<$ athenaHLT speed up (solid lines).

Outcome

- A **steady increase** in events per second w/ number of threads for **both CPU and GPU**.
- Throughput: [GPU 'jet+fsLRT'] > [GPU All] > [GPU 'jet'] > [CPU All].
- Not much is gained in throughput and speed up **after thread=16** as the no. of cores available for the machine in use is 16.

Activities @RAL

1. Running FTF GPU prototype (CUDA) as a test on grid – [Stewart Martin-Haugh].
    - Runs regularly at ATLAS GPU sites – Manchester, SLAC, etc.
2. Implementing FTF seeding (a Graph-based track seeding) in ACTS is **ongoing** – [Rosie Hasan].
3. Developed ACTS stand-alone full-chain example for the ATLAS ITk – [Tim Adye].
4. Integration of ACTS Combinatorial Kalman Filter (CKF) into Athena – [Tim Adye].
    - Now validating, improving, and optimising.

3. & 4. – Being done by the ACTS-ATLAS team (including RAL).

News from *GPU usage survey 2023*

(WLCG link – the survey will be conducted again in one year to know what has changed.)

Questions and responses

	ATLAS	CMS	LHCb	ALICE
What is the current status of the integration of GPUs for offline computing at the software level?	**Several R&D projects** are ongoing investigating GPU usage for the HL-LHC era. Athena offline software supports the integration.	All the components for the **usage of NVIDIA GPUs** are **present in the CMSSW**. Submission of **workloads on Grid worker nodes** equipped with NVIDIA GPUs is **fully supported**.	Allen framework is used in Run3 on the **event building farm GPUs**, HLT1. **Offline GPUs** could be used to emulate the HLT1 in sim.	Uses a **common O2 software framework** for online and offline computing, capable of offloading certain reco. steps to GPUs. Codes can be run on different **GPU backends**.
Are GPUs already being used for offline computing activities?	GPUs are already **integrated into offline computing**. Used for **limited analysis use cases**, *e.g.,* ML training,...	Uses **GPUs offered opportunistically by sites** *e.g.,* large-scale validation of HLT reconstruction code,... **Very positive experience**.	**A few analysts use GPUs** for, *e.g.,* maximum-likelihood fits,... **Not Possible to comment** on actual usage.	A substantial part of the **2022/2023 pp data** was reconstructed on the EPN computing farm using the **farm's GPUs** (AMD MI50 and MI100).
If these resources are used offline, then clarification on the GPU usage is needed.	GPU usage is measured **similarly to CPU usage**. No benchmarks yet; **unpledged**.	Presently, **do not account for GPU usage** and consider all accelerators are opportunistic.	Do not account for GPU usage (see above).	The GPU usage is accounted for in CRSG reports. Use **equivalence accounting** between the GPU-equipped EPN nodes and CPU-only nodes **with known HEPScore**.
Are there any plans on GPU resource demands or future utilisation at sites – mid-term and long-term objectives?	Towards the **end of Run 3**, prototypes of evgen, sim., and/or reco. software able to use GPUs may become available. **Large-scale GPU deployment** at sites by the start of HL-LHC (∼2028).	**To follow WLCG guidance** for how to include GPUs. Goal: **offloading ∼10%** of the offline reconstruction to GPUs by the end of 2023.	**There are no established plans** on GPU resource demands or for future utilisation at sites. **November 2023 workshop** to understand how to use GPUs/accelerators.	**To optimise the use of various GPU models** through the resources made available from various labs and computing centres. Includes the optimisation of memory use. Also investigating the possibility to use **HPCs equipped with GPUs** for offline reco. tasks.
Any other plans, *e.g.,* on FPGAs?	**Not for offline**. For online, FPGAs are being seriously considered on the **timescale of HL-LHC**.	In the short-term, GPU resources must be considered **unpledged**. It will be good to start **developing a pledge** for possible future use cases.	Performing **some R&D work on FPGAs** and TPUs for **online** – targeted for LHCb Run 4/Run 5 upgrades.	**There are no plans to use FPGAs** in the offline reconstruction.

Takeaways from the survey

- Outside HLT farms, GPU usage is marginal. There are no short-term plans to include GPUs at scale from WLCG sites (in general), but sites offering GPUs help ongoing R&D activities.
- FPGAs are only for online; plans for offline still need to be made.
- There are no benchmarks for GPU at the moment, which affects the accounting $\Rightarrow$ Benchmark WG on it!
- Developing a GPU pledge framework within WLCG (related to the previous item) is good.
- All of these resources are treated as opportunistic; for the moment, we are waiting for guidance from WLCG, though the usage still needs to be at scale.

Backup

Details of the machine

- **hepacc01** (@RAL).
- CPU – DUAL Intel Xeon E5-26702.6GHz 8-core processors.
- RAM – 64 GB.
- GPU – 2x NVIDIA TITAN V.
- GPU memory – 12 GB HBM2.
- GPU cores – 5120 CUDA Computing Cores (1.5 GHz Boost Clock).

Job Options

The *job option* is obtained via test_trig_data_v1Dev_build.py.

- **athena** job option:

  athena.py -c
  "setMenu='Dev_pp_run3_v1_TriggerValidation_prescale';doL1Sim=False;doWriteBS=False;doWriteRDOTrigger=False;
  fpeAuditor=True;forceEnableAllChains=True;" - -imf - -pmon=perfmonmt **- -threads=1 - -evtMax=1000**
  - -filesInput=/cvmfs/atlas-nightlies.cern.ch/repo/data/data-art/TrigP1Test/data22_13p6TeV.00431885.physics_EnhancedBias.merge.RAW._lb0545._SFO-15._0001.1,/cvmfs/atlas-nightlies.cern.ch/repo/data/data-art/TrigP1Test/data22_13p6TeV.00431885.physics_EnhancedBias.merge.RAW._lb0545._SFO-17._0001.1
  /scratch/jbiswal/Rel23p0p11/run/run_GPU/runHLT_standalone.py >**athena.log 2**>**&1**

- **athenaHLT** job option:

  athenaHLT.py -c
  "setMenu='Dev_pp_run3_v1_TriggerValidation_prescale';doL1Sim=False;doWriteBS=False;doWriteRDOTrigger=False;
  fpeAuditor=True;forceEnableAllChains=True;" - -imf **- -threads=1 - -evtMax=1000**
  - -file=/cvmfs/atlas-nightlies.cern.ch/repo/data/data-art/TrigP1Test/data22_13p6TeV.00431885.physics_EnhancedBias.merge.RAW._lb0545._SFO-15._0001.1
  - -file=/cvmfs/atlas-nightlies.cern.ch/repo/data/data-art/TrigP1Test/data22_13p6TeV.00431885.physics_EnhancedBias.merge.RAW._lb0545._SFO-17._0001.1
  /scratch/jbiswal/Rel23p0p11/run/run_GPU/runHLT_standalone.py >**athenaHLT.log 2**>**&1**

- **forceEnableAllChains=True** option is used to force all the chains to run.

- Instead of using the default runHLT_standalone.py, the corresponding file is downloaded and modified locally (see backup). The default is also release-dependent.

Modifications to runHLT_standalone.py

- Addition to the default:

```
algList = ["TrigFastTrackFinder__BeamSpot", "TrigFastTrackFinder__electron",
"TrigFastTrackFinder__electronLRT", "TrigFastTrackFinder__muon", "TrigFastTrackFinder__muonLRT",
"TrigFastTrackFinder__muonIso", "TrigFastTrackFinder__muonIsoMS", "TrigFastTrackFinder__bmumux",
"TrigFastTrackFinder__tauCore", "TrigFastTrackFinder__tauLRT", "TrigFastTrackFinder__tauIso",
"TrigFastTrackFinder__tauIsoBDT", "TrigFastTrackFinder__jetSuper", "TrigFastTrackFinder__jet",
"TrigFastTrackFinder_fullScanLRT", "TrigFastTrackFinder__bjet", "TrigFastTrackFinder__DJetLRT",
"TrigFastTrackFinder__DVtxLRT", "TrigFastTrackFinder__muonFS"]

for TrigAlg in algList:
    from AthenaCommon.CFElements import findAlgorithm,findSubSequence
    ftf = findAlgorithm(topSequence, TrigAlg)
    ftf.TripletDoPPS = False
    ftf.useGPU = True ## for GPU mode
    ##ftf.useGPU = False ## for CPU mode
    ftf.UseTrigSeedML = 0
```

- Addition to the default while running 'jet' exclusively for GPU:

```
algList = ["TrigFastTrackFinder__BeamSpot", "TrigFastTrackFinder__electron",
"TrigFastTrackFinder__electronLRT", "TrigFastTrackFinder__muon", "TrigFastTrackFinder__muonLRT",
"TrigFastTrackFinder__muonIso", "TrigFastTrackFinder__muonIsoMS", "TrigFastTrackFinder__bmumux",
"TrigFastTrackFinder__tauCore", "TrigFastTrackFinder__tauLRT", "TrigFastTrackFinder__tauIso",
"TrigFastTrackFinder__tauIsoBDT", "TrigFastTrackFinder__jetSuper", "TrigFastTrackFinder__jet",
"TrigFastTrackFinder_fullScanLRT", "TrigFastTrackFinder__bjet", "TrigFastTrackFinder__DJetLRT",
"TrigFastTrackFinder__DVtxLRT", "TrigFastTrackFinder__muonFS"]

for TrigAlg in algList:
    from AthenaCommon.CFElements import findAlgorithm,findSubSequence
    ftf = findAlgorithm(topSequence, TrigAlg)
    ftf.TripletDoPPS = False
    ftf.UseTrigSeedML = 0
    if TrigAlg == "TrigFastTrackFinder__jet":
        ftf.useGPU = True ## for GPU mode
```

- Modification to the above block while running 'jet' and 'fullScanLRT' exclusively for GPU:

```
if TrigAlg == "TrigFastTrackFinder__jet": ⟶ if TrigAlg == "TrigFastTrackFinder__jet" or TrigAlg ==
"TrigFastTrackFinder_fullScanLRT":
```

FastTrackFinder algorithm instances

1. **TrigFastTrackFinder__BeamSpot** → full scan
2. **TrigFastTrackFinder__electron**
3. **TrigFastTrackFinder__electronLRT**
4. **TrigFastTrackFinder__muon**
5. **TrigFastTrackFinder__muonLRT**
6. **TrigFastTrackFinder__muonIso**
7. **TrigFastTrackFinder__muonIsoMS**
8. **TrigFastTrackFinder__bmumux**
9. **TrigFastTrackFinder__tauCore**
10. **TrigFastTrackFinder__tauLRT**
11. **TrigFastTrackFinder__tauIso**
12. **TrigFastTrackFinder__tauIsoBDT**
13. **TrigFastTrackFinder__jetSuper**
14. **TrigFastTrackFinder__jet** → full scan
15. **TrigFastTrackFinder_fullScanLRT** → full scan
16. **TrigFastTrackFinder__bjet**
17. **TrigFastTrackFinder__DJetLRT**
18. **TrigFastTrackFinder__DVtxLRT**
19. **TrigFastTrackFinder__muonFS**

full scan: tracking over the entire Inner Detector; processes large number of seeds ⇒ longer processing time relative to the standard tracking.