HOW2019 Workshop - HSF Sessions Live Notes

Following our usual traditions to democratise notes from the meetings, please use this notebook to record key points, ask questions and make suggestions for the HSF parellel sessions. The only rule is don't delete anyone else's comments.

Software on Accelerators

Overview of Accelerator Technology

- bfloat16 - very poor precision
- Can change techniques when using GPUs - more parallel algorithms
- With Aurora having Intel GPUs, is there a programming API announced?
    - Investment in Exascale Projects, so do they have to provide the solution in this way.
    - There does seem to be investment in Kokkos, RAJA, SYCL as part of ECP - need to explore these
        - C++ libraries, compile down to CUDA (and other targets)
    - AMD have HIP

Accelerator Integration in CMSSW

- Plugin mechanism is quite flexible
- Trying not to have state in modules, but if needed can be supported (through the multiple paths, if required)

ALICE GPU Algorithms

- Limited by C++ standard that CUDA supports for common code (currently C++14)
- Irreproducibility is mainly from concurrency itself, not from any GPU specific feature.

LHCb HLT1 GPU Project

- Dynamic memory allocation slow on GPU - big static allocation at start, simple manager hands out pointer to where to write.
- Batching is an extra layer you need to bookkeep.
- Algorithms really try to avoid branches - much simpler flow.
- T4 might be best bang per buck, but pricing a but uncertain.
- Always have to speak to DAQ people - optimise flow of data into the system (turns out to match the way subdetectors readout anyway)
- Does each developer need to have an nVidia GPU?
    - CERN people share a lab of machines.
- Would have liked to use a generic API that targets CPU and GPU, but there wasn't really anything suitable at the time the project started.
    - Although, don't use very sophisticated CUDA calls, so not so tightly bound…
- Find that 32bit FP performance is the most important factor. Memory not so bad overall. Some bottlenecks moving memory around the GPU blocks.
- Limiting factor is really the calculations.
- Older CPU code had good performance, but in one phase space corner. Porting that code to the GPU was horrible - really had to rewrite it all. Physics performance looks very comparable.
- Will have to port back to CPU for running simulation. Would also like to be able to use GPU HLT1 farm for other things when not data taking.

CMS Patatrack Project

- What is the comparison to the CPU physics performance, that used more complex algorithms?
    - Don't have all features, but do get same efficiency and fake rates as CPU tracking (only pixel tracker). Other groups are still working on full tracking.
- Do you use CPU cycles to push data to the GPU?
    - Want to test unified memory technique as well, but for the moment easier to do explicit transfers.
- Seems that for most developers, porting to a low level language like CUDA is too hard - need a higher level abstraction.
    - Yes, but need to check that the abstraction doesn't introduce big overheads, e.g. by allocating memory each event

MadGraph on GPU

- Is the CUDA code generator available in MadGraph now?
    - In a private branch, not yet in main repository.
- What is the speedup compared to CPU?

    - Needs to be measured further - for multi-jet processes could be high.
- Could we factor common pieces between generators to help with porting?
    - Code generations could be more commonly used.
    - SM: Phase space integration could be done, but
- Practical use also involves matching event with Pythia and hadronisation, which can be very inefficient. Can that whole workload be sped up?
    - Would be very desirable, but not clear how to do it.

Geant4 and effective use of Accelerators

- Speed-up comparisons are vs. a single CPU core (slide 9).
- Do you care about backward compatibility?
    - Not at the moment - try for the best proof of speedup first; then think about more portable solutions.
    - Frameworks like Kokkos might be a solution for an eventually more maintainable mode.
    - Andrei - worry that the experience is that things become more and more complex in practice; hard to move away from the prototype.
- Isn't it a huge amount of work to really rewrite this code? Would it be better to be even more radical? Not traditional particle-by-particle transport, but, e.g., an ML problem.
    - Can use existing code for now, even if inefficient, then look at the places where there are bottlenecks.
    - More radical approaches might come to light as limits are explored.
    - Projects to look into ML run in parallel.
- Paolo - experience shows that porting to GPU is not actually that hard, but it will run at about 10% performance. Maybe that's good enough.

Discussion

- Kokkos workshop next week. "Bring your kernel". But what would we have in HEP that would be considered a kernel? Could we have a library of pieces of code that we use.
    - Kokkos developers will come to you on request.
    - Kokkos worth trying, it's well funded and connected to ECP. However, we can't yet say it's a winner.

Detector Simulation

Wednesday Session

Physics requirements for simulation

-

Neutrino experiments simulation overview

- CAD2GDML is interesting - does it translate to tessellated solids?
    - Yes. Little gaps and overlaps are a problem.
- Optical photons simulation is also something to understand better

Modernisation of simulation code

- It is nice to have a docker for performance evaluation but it depends very much on what to run in G4: a benchmark for HEP EM calorimetry will not necessarily be useful for neutrino physics or tracking. Shouldn't there be a set of reference benchmark? How should they be chosen?
- Relation with HSF-simulation and Geant4 task force for R&D?
- Difficult for existing project to do maintenance/support and R&D at the same time:
    - Indeed we need to ensure both are addressed. Maybe a way is to also ensure better communication with interested parties

Fast simulation

- Are the speedups really orders of magnitude?
    - Yes

Geometry

- There are two 'issues': the in-memory representation (e.g. VecGeom) and the persistent representation (e.g. GDML). In fact DD4Hep is an 'hybrid'.
    - Yes but we should keep in mind that you can insert 'error' in moving from one representation to another.

Vectorization

- How much the code depends on a specific architecture?
    - All code designed in a way no problem to move from one architecture to another by using VecCore. Was a requirement from the beginning.
- Effects of lowered clock speed when vector registers are used is part of further investigation.

Pileup

- CMS have tested reduced ranges for out of time pile-up [-3,+3]. Found the reduced window had very little noticeable effect.
- ATLAS pre-mixed pileup is a bit larger than CMS, a few caches of datasets and then file-by-file distribution as needed.

Code integration

- Wanting to see final integration with G4.
- Are wrappers stable? As far as known, yes. CMS have dome some careful testing of that.
- Is the ATLAS FastCaloSim offload using APE? Probably not.

Data Analysis

Paul

- Paul highlighted Jupyter notebooks as a frontend for analysis. I'd like to hear more about how that works with teams and analyses that last ~1 year.
    - Jupyter team has moved to jupyter-lab, which is more like an IDE on the web (look, mom, I'm doing analysis on a Chromebook). If it is backed by a git repro, then this makes more sense to me, but this also means something really different from "notebook" that we keep referring too.
    - Has anyone built an interface where only notebooks are the interface?
- Is "turbo analysis" (analysis in trigger) perhaps best understood as a socially-acceptable way of agreeing to throw away the raw data after a certain amount of calibration? We could of course in principle do this even post-trigger.
    - Yes but it doesn't have to be, one of the big advantages is avoiding reprocessing and the cascade effect that creates. Lots of time is then spent following that data reduction cascade, the fewer people do that the better, and the less often people do that the better. You could always keep the raw data, either just in case or with a view to reprocess at the end of data-taking.
        - Sure, I was thinking mostly in terms of whether we can extend the paradigm ("you get one chance to process the data, so do it right the first time").
            - No need to be dogmatic, unless you have to throw away the raw data then you don't need to. But if you accept the prompt calibration quality is good enough then maybe reprocessing is really seen as the luxury it is, and isn't done.

Johannes

- Analysis is predominantly single-threaded at the moment. It may be an advantage to introduce MT for better exploitation of resources such as HPC.

- CVMFS seems to be used profitably at scale and in the future this can be complemented by containers, e.g. Singularity.
- Question: is the size of the distributed containers not an issue?
    - Yes and No - centralized produced containers O(10-100) can distributed be distributed via CVMFS
    - Unsolved/work-in-progress: If the containers are layered, parts can be distributed by CVMFS and the remaining parts pulled via caches - this is currently being investigated
- Containers for shipping analyses prototyped locally. Comment: different analysis phases have different turnaround times, containers may work ok for some stages, less for others..
- Quickly prototyped analysis containers would seem to not quite match the CVMFS model - do we need to think about the distribution network?
    - No and Yes - see the previous comment above
    - The issue here is that analysis containers aren't centralized, there will be thousands of them, and most collaboration members can't modify CVMFS… but OK if the additional "analysis layer" is thin (comparable to the current lib datasets) then it's probably fine.

Andrea

- The absence of a common interface to non-event data (or metadata) query is an issue we need to address to be more efficient.
- Caching intermediate analysis results requires very tight coordination of the analysis code and the data management system (so you actually know which intermediate results are still valid relative to your current code) - your cache key is really complex!
- Abstraction can make it more complicated for students to start analysis
- Are common analysis "frameworks" realistic?
    - Only if abstraction level is correct..
    - Should perhaps reframe the discussion to concentrate on toolkits
- HSF is a discussion forum, not a decision making body!
- The analysis models, formats and workflows differ between the experiments. E.g., ATLAS does use (d)xAOD as the primary analysis format for analysers, produced in trains. xAOD was replacing the previous ntuples. The xAOD allows to decorate and skim/slim the information stored in the objects. ATLAS does provide analysis releases with calib and recommendation tools to act on this. The analysis strategy of e.g. CMS differs, in that the compact CMS formats are predefined ntuples, leading to different work models.
    - The ATLAS analysis releases is an important point omitted from my (Paul's) talk. This was a huge improvement, organising analysis tools and providing some convergence on the interfaces for these tools.
- Life is changing - older methods don't scale today and certainly won't scale tomorrow.
- Distinguish between expert skills and what an analyst needs to know (we need experts)
- On the question of how to communicate resource limits to users - the ATLAS way is to consider the opportunity costs (you run x amount more data derivation, you get y times less Monte Carlo).

    - In general having a cost model is a very interesting idea. What we do with the information can come later, i.e. currency conversion will need work.
- Python ecosystem demonstrates how powerful toolkits and factorisation work.

Data analysis day 2:
Danilo

- Emphasising programming model - simplicity; performance and parallelism; infrastructure
- Declarative approach an old concept but powerful, separate frontend from backend
    - Not a natural fit for C++ for example
- Language choices: C+++python (C3P?); python for physics, C++ for power under the hood
    - Analysis description languages is an emerging concept, may be powerful for analysis preservation
- Are we exploring everything we should?
- Can common HEP concepts be identified?
- Can we classify those tools?
- Create a set of benchmarks?
- Analysis description languages - do we risk encouraging students to learn something that's completely useless outside of HEP?
    - Good point that we should keep in mind
    - If the language concepts are expressed in e.g. python, maybe this is a good solution
    - This concept has already started in the theory domain, with a different approach, there are papers
    - Notebooks aren't good for collaboration, and not composable (similar experience with Jupyterlab)
    - What does it translate to / code generation?
- How many real users are actively working on jupyter? Could be misleading to just see people who've clicked, taken a look and then logged off again.
- PyHEP have a very clear connection when thinking about python tools, we should ensure cross-coordination and take advantage of the commonalities here

Nick

- Columnar analysis, declarative - implicit inner loops
- Maybe less imperative is more accurate, array programming, you still have to define the sequence of operations
- Awkward arrays allow array programming on jagged arrays
- Coffea framework or toolkit glues these things together
    - Aim or hope is that the useful parts will end up as python modules
- Focus on speed of the tools themselves should be balanced against something that is simple and scales well (just use more condor cores)
- User support is google as the tools (e.g. scipy) are well used in the wider community
- Challenge is that it's not intuitive, easier to read than write, but maybe once you're over the learning curve then the power of the concept can be used

- Expressing systematics is supported in a way that looks intuitive
- Not applicable everywhere but powerful where it can be used
    - Example given deterministic annealing
- Coffea is being used for real analysis, some important lessons
    - Need to be able to do fast turnaround for debugging, and seamlessly submit to your powerful backend
    - Handle non-event data, physics book-keeping
    - Higher memory footprint but can be solved by repartitioning
- Impressive throughput numbers for multiple backends
- Q. On performance, what is the corresponding performance for doing the same thing in C++, some doubt if this is a fair comparison but does seem fair, can maybe be checked independently
    - Great case for defining a benchmark where we think we have something that is representative of analysis
- nano-AODs are compressed with LZMA, 36% time spent in decompression
    - Balance of optimising disk vs compression CPU
    - This is also a simple analysis so little real CPU
- Priority can be simplicity which allows people to spend more time doing physics
- Nick asserts that the user experience is just as important as speed - do we agree with this? If we can achieve parity with a well written C++/ROOT analysis is that sufficient?
    - Ben G. asked a question on this about technologies that improve ease of training and best productivity. This question is exceptionally pertinent as we set our sights on the HL-LHC, where funding for people will be tight.

Stefan

- Easy to use correctly, hard to use incorrectly and benefit transparently from parallelism
- See slides for a nice tutorial on definitions and not to use RDataFrame
- A really nice aspect of RDataFrame is lazy evaluation. What tools exist in the Python ecosystem that work like this? Do they interoperate well with the rest of the ecosystem (i.e. are they a “drop-in” replacement)?
    - Dask? Xarray? Can uproot interact with these?
- Linked examples, seems the Higgs example is broken
- Scaling works out to 100 cores and then problematic nose dive, under investigation
- Zero copy with adoption RVec vector class, numpy look and feel
- We know you love python
- pyROOT code looks exactly the same as C++
- Translation of results into numpy, use pandas to analyse that, seamless integration of these tools
- Python decorators to C++erise functions
- REALLY seamless integration of numpy back and forth, zero copy with adoption
- Q. Can the rdf/numpy interface be extracted from the full root package, is that technically possible?
    - Vs uproot solving this problem?
    - Performance would be better in rdf/numpy

    - Seems to be difficult to really extract given the interface is much bigger in scope
- Python3 support question for the decorator solution
- How about direct export to spark?
- Since RDataframe makes Root files look like a relational database, could we put a database interface in front of it. Turn RDataframe into a database server. Then we create a JDBC driver and use that to pull data into Spark. That way all of the C++ dependencies can stay on a server and keep Spark pure java and python
-

Reconstruction and Software Triggers

Session 1: Real-time analysis

Acronym glossary:

RTA = Real-time analysis
HLT = High Level Trigger
TLA = Three Level Acronym
TLA = Trigger Level Analysis
HTT = Hardware Track Trigger
HGTD = High Granularity Timing Detector
FTK = Fast TracKer
ALP = Axion-like particle

14:00-14:20: Real-time analysis model at the LHCb experiment - PEARCE, Alex

LHCb-DP-2019-002, describing evolution of RTA model during Run-2
RTA / Turbo = what does it mean? Explanation, plus challenges/discussion

What is RTA?
Definition in the paper: interval between a collision occurring and point in which one discards the event (total trigger decision)

RTA: set of action applied to events that may be later discarded forever, OR analysis of that information later (in principle could have done it earlier).

Why is RTA interesting?

- Bandwidth (limited) is proportional to accept rate x event size
- We want to maximise signal efficiency and reduce bias
- The only handle we generally had was accept rate -> reduce event size

This can and should to be done in the trigger: it's already computing useful information. LHCb makes that information good enough to use offline.

LHCb's environment: 1.6 average interactions per bunch crossing in Run2, will be ~7.8 in Run3 (pileup is still an issue, not as much as ATLAS and CMS).

Challenge of LHCb:
Every interaction is signal -> try to record as much as possible.
RTA buys offline-quality physics objects for the whole experiment

Turbo = persist objects from 2nd level trigger and only analyse these offline
Individual trigger chains can select what they want to persistify, configurable
Can also enable a flag to save everything from the trigger reconstruction, when you have no clue of what else you need

Internal details: see talk for how things are serialized/persistified/written out in application called Tesla. Same tools can be used to analyse the

Rewards of RTA:
Turbo - selective persistence size = complete persistence size = Raw event size
[add numbers]
Rewards for physics analysis: enormous dimuon rate can be used for searches, as well as measurements

Lessons learned:
- Shouldn't be afraid of losing information
- Improve constantly (e.g. selective persistence)
- Help users as new features were introduced

What is coming for Run-3:
Enormous increase in event rate (Nx), increase in event size (Mx)
70% of the physics program has to go into RTA model to fit in the output bandwidth is 10 GB/s. Comes with challenges.

Q&A

How many different different streams you have?
One trigger stream, then we separate in 5-6 streams with different files. Format is identical (save the same stuff), but trigger selection is different to help analysers.

Identicalness of HLT2 and offline -> how did you do it? You made the offline faster?
Yes.

Q How does the calibration work? There is a calibration running in the trigger itself, what are the differences?
A We don't perform calibrations offline.
Q Then you don't have corrections that needs a large data sample
A We only need to do efficiency corrections using that, and can be applied to the final products

14:20-14:40: Real-time analysis model in ALICE - ROHR, David

ALICE will record all the collisions on Run-3.
High data compression online is mandatory - based on event reconstruction, not for analysis but for compression.
Also, speed up online reconstruction with GPUs (see yesterday's talk)

Computing farm detail: see slide.

Turnaround time is different wrt LHCb: store 3 weeks of data taking and have 1 year to process.

Synchronous step/asynchronous steps for calibration, one online and one offline. There are distortions in the TPC that need to be corrected already online. Two strategies, a mix of both.

The real-time part is the drift time calibration, needed for the tracking. Feedback loop, needs to be stable over a period of 15 minutes. Take the first few events, use them for calibration, then use the following minute, etc…

Data taking and computing scheme:
- First detector sees the event
- Then FLP farm (125-150 nodes). This is where the event building happens.
- Then EPN farm, main processing farm. Has 30 seconds to process.
- Can also be used asynchronously, together with the grid
- This is where GPU reconstruction happens
    - GPU reconstruction on the grid
    - Reco code has same results on GPU/CPU

Data rates: TPC biggest contributor to the data rate (compression critical, assumed 20x). During the 50 kHZ lead-lead the peak rate is 80 GB/s [check slide]

How compression is done: reduction of entropy, entropy encoding (gets to 9x) but then the rest is removal of tracks done for physics and must go from 10x to 20x.

What is planned for removal of tracks: remove clusters from background and looping tracks, but not adjacent to a physics track -> various challenges for tracking.

GPU tracking used as well (1 GPU replaces 800 CPU cores running old software / 40 cores with new software).

Q&A

Q: Drift calibration already running in Run-2: how do you validate that / how reliable has that been?
A: compare the drift velocity with offline, better than 1 permil.
Q: running in some specific architecture?
A: in HLT computing farm.

14:40-15:00: Real-time analysis model in ATLAS - KALDERON, William

What are the limits of a trigger system?

- Readout electronics -> caps L1 accept at 100 kHz
- Hard limit of CPU available within HLT -> can run limited reconstruction
- Soft limit of storage and processing -> can only average max 1 kHz
    - Jets are 15% of the total (150-250 Hz)

This means there is an area of about 300 GeV in jet pT (between L1 and HLT) where we're discarding events because we don't have space to keep them, but we reconstructed them at the HLT.
Solution: record these partial events, not throw them away.

Store only HLT jets and 4-vectors + summary info -> go to 0.5% of the total event. This can be used for physics analysis for dijet resonance searches (Trigger Level Analysis, TLA), otherwise limited by the trigger thresholds and blind to jets below 1 TeV. ATLAS TLA / CMS scouting have the best reach between 450 GeV and ~600-700 GeV.

How real-time is real-time? Triggers rely on pre-existing calibrations that are not updated easily -> err on the side of caution, and improve afterwards, so analyse and recalibrate HLT jets written out. This is important for searches that have a lot of statistics and rely on smoothness of distribution for background estimation to better than 1 permille as they look for a very small bump. Final calibration: 0.005% wrt offline.

Run-2 / future developments:

- Hard limitation: L1 rate, but that decreases with instantaneous luminosity. Can record more TLA rate as online farm gets less busy.
- Run-2 proof of concept, run-3 full commissioning, run-4 can make more changes to help.

New things discussed in HSF whitepaper (ATLAS PUB note that went along that):
- L1 hardware (jFex, gFex)
    - Could in principle write out histograms at 30 MHz
- L1.5 hardware (FTK, HTT, HGTD) gives tracking to expand beyond dijet resonance searches
- Could do calibration for offline using HLT objects

Q&A

Q: Are people looking at this for leptons?
A: People are interested, there are low dimuon resonances.

Q: You write out an enormous rate. Do you come across specific challenges that others don't?
A: we broke the metadata because event number > INT. Our event size looks different (small event) and we are the most sensitive to detector effects. Carefully checked run by run and found out differences that would matter for us and not for others. Fed back into how we do the trigger.
Q: Real-time DQ is more important than for others. Is there a dedicated effort?
A: Not a particular effort. We have good trigger monitoring for a fraction of the data, compare offline and online. However we need a lot of the data to spot subtle problems.

15:00-15:20: Real-time analysis model in CMS - SPERKA, David

RTA in CMS: more physics side than technical side.

CMS trigger system: 30 MhZ to 1 kHZ, 3 GB/s output.

Different strategies:
Data parking - take a lot more than 1 kHZ data and reconstruct later.
Data scouting - reduce the event size to collect events at a higher rate.

Why should we go to lower pT?

- We expected new physics at TeV scale, but none yet -> we spent the center of mass energy increase, now it's time to do more with lumi.

- LHC experiments are the only active experiments that can search for new physics above 12 GeV
    - Axion-like particles
    - Dark matter/dark sectors
    - Flavor anomalies

Results:

- Dijet resonances
    - Using “traditional” scouting stream to go as low as 500 GeV in dijet mass (standard can only go to 1 TeV)
    - Calibration done with dijet balancing technique so that they have the same response as offline jets have the same response
        - Q: how?
        - Resolution is 18% worse for dijet mass of 2 TeV
    - Same technique as the standard analysis, place constraints on models
- Three-jet resonances
    - Looking for resonances decaying to 3j, pair produced
    - Key: pile-up combinatorics so need to have tracking information -> particle flow jets used
- Low-mass dimuon search (not yet public, coming soon)
    - Dark photon models: resonances decaying into muons
    - But: standard thresholds 17/8 GeV, cannot search below 30 GeV.
    - Collected dimuon events using variety of L1 and processed at the HLT, using three of them to do the analysis.

Extensions for Run-3 / HL-LHC:

- PF scouting for all L1 events? Now 400 ms / event, can only do 250 ms
- Target specific scenarios for low-mass scalars or ALPs in photons
    - Q: how? Thresholds
- Scouting at 40 MHz with track triggers
    - Boundary conditions: profit if limited by L1 trigger rate budget, but trigger is improving
    - Physics cases: dark photons, H->phi+gamma or rho+gamma, low mass W’, hidden sector hadronic physics

Q: how to go to feedback loop between monitoring/calibration and analysis
A: we had problems that made us lose some data, so we should improve there. Calibrations: for muons it’s critical that calibration is stable. If you do a diphoton search then it’s critical.

Q: CMS looking at putting part of tracking on GPU. How does that help?

A: Tracking is ~30% of the CPU time, so probably not enough on its own but this is the right direction to go in.

15:20-15:30: Discussion

You have a few new analyses moving to this model. In LHCb, people know that it's nice and you can't do otherwise. How does that work in CMS and ATLAS?
CMS: it's driven by analysts. No push from the trigger group at least so far.
ATLAS: first implementation was people who wanted to, but it's been recognized that we want to do more. So we recently had a push of trigger group asking "think about this", and there are some ideas internally.

Session 2:

16:00-16:20: Machine Learning for the Primary Vertex reconstruction - SCHREINER, Henry Fredrick

Efficiency as a function of nTrks for Primary Vertices (PV) - fewer tracks means lower PV efficiency. This is the figure of merit we are trying.

Vertices and tracks can be used as input for ML:

- Events contain on average 7 PV (5+ tracks) and Secondary Vertices (SV, 2+ tracks)
- This is a 3D dataset that can be turned to 1D and fed into a convolutional neural network (CNN)

Use tracking -> kernel -> CNN -> predictions come out and can be compared to truth

How to generate the kernel: see picture in the slides.
Architecture of NN designed together with computer scientists.
Cost function is symmetric, asymmetric term helps keeping the false positive (FP) rate down vs having a good efficiency. This can be tuned using the asymmetry parameter.
Predictions can be compared to target, and false positives are generally understood.
Results: PVs found with efficiencies and false positive rates similar to traditional algorithms.

One could apply to other detectors, using a second dimension (e.g. z of beamspot). Interesting place to try things in ML.

Code available with Conda on macOS, including event generation.

- Health&Safety warning: don't use the GPU environment on mac

Q&A

Q: Do you have ML baseline compared to other approaches?
A: Not on this dataset. But targeting Allen framework (see yesterday's talk)

Q: [missed question by A. Pearce, go ask later]

Q: Thanks for conda-root package
A: Come to the talk tomorrow

Q: how long is the training taking?
A: 240k events takes some time to generate, simplest network 20s/epoch, or 1 minute/epoch. To get to the final result, procedure is more involved as we make changes as we go. Generally 800 epochs.

16:20-16:40: Using FPGAs to accelerate machine learning inference - PEDRO, Kevin

Collaboration between ATLAS and CMS.

CPU needs will increase a lot in the next decade. What to do?

Use coprocessors, but generally geared towards ML. So can either:
- Rewrite physics algos for new hardware
- Recast physics problems as ML problems

DL in science and industry: very successful in industry, HEP is a really small player in that.

How do we do? Examples:

- ResNet-50, convert jets into images and retrain on top tagging dataset, then run it on FPGA. Obtaining results that are similar to state-of-the-art.
- Image reconstruction to reject CR background, using CNN inference (NOvA)
    - Since this takes a long time in terms of neutrino reconstruction, candidate for acceleration with coprocessors

Why should we accelerate inference and not training?
- DNN happens 1/year/algorithm, so can use huge resources then
- Once DNN is in use, inference happens billions of time

- Use algorithm as a service to minimise disruptions to computing model & dependence on hardware (just connect to a service that gives you a result back)
- Metrics:
    - Latency
    - Throughput

Coprocessors (FPGA/ASIC) are an industry trend - this project uses / collaborates with Microsoft.

CMSSW feature: ExternalWork has asynchronous task-based processing. Features:
- non-blocking (can do other tasks while waiting)
- Can be used on different architectures

Approach: SONIC = Services for Optimized Network Inference on Coprocessors using it. What it does:
- Convert data into NN input
    - Eg Tensorflow
- Send NN input to coprocessor
    - Using communication protocol
- Use ExternalWork for async requests

Latency metrics:

What service? Cloud vs edge? First used Cloud but it has latency, so use as edge resource (install a box of FPGAs near your computing cluster). Latency reduced from 60 to 10 ms, which takes 1.8 ms for inference and rest for classifying (on CPU) and I/O.

How does it scale? Fairly stable when running N simultaneous processes (simulating the fact that a std reconstruction process would have many things running at once)

Throughput metrics:

Load balancer works well, can compute inferences per second. N=50 occupies FPGA, so throughput is 600 inferences/second.

Comparison with CPU performance: a lot of the time goes into importing the algorithm, and then 1.75 seconds of inference. Doing huge inference takes time! GPU is a lot better and close to FPGA with batch=32. Caveats though, e.g. need direct connection.

Conclusion: as networks get larger inference takes longer, so FPGAs are a good option for acceleration. Need to understand scalability and cost.

Q: batching in GPUs
A: would be nice to batch problems.

Q: are huge NN going to be bread and butter for everyone in reconstruction?
A: trend seems to go that way

Q: FPGAs are hard to program, how easy is it to make changes or ask for what we want?
A: that's why we partner with industry so that they can do it for us as long as we use architectures/networks they know about.

Q: Does this scale to our use case or do we have to have industry people work with non-image inputs?
A: HEP.TrkX tracking seemed to be successful. CMS trying HGCal clustering with that. Active area of research.

Q: How proprietary is the scheduling?
A: quite.

Q: can you send your brainwave to many machines, and do inference, and then send to grid jobs
A: you could but you don't want to do that event by event

16:40-17:00: Event reconstruction goals and challenges at JLab - ZIEGLER, Veronique

Challenges - data rates, data distribution, backgrounds and tracking efficiency, detector inefficiency understanding and modeling
Tracking in magnetic field is the big time consumer. Physics configuration differ depending on the groups taking data. 97% tracking for CLAS12, 94% for Glue-X
CLAS12 - Parton distributions in deep inelastic scattering.
GlueX - Origin of quark gluon confinement - exotics, production asymetries. GlueX-II is using the babar Dirc detector
CLAS12 data volumes 15kHz, 25 kB/event - thanks to tracking trigger
Analysis trains to make skimmed results
Java toolset, CLARA framework with microservices (multithreading naturally)
Compute resources - OSG, MIT, NERSC
Software - CVMFS, Docker
Tracking via road search algorithm - MC to find dictionary of possible track patterns

Cellular automaton algorithm for seedings (a la Hera-B CATS algorithm)
Efficiency measurement via track embedding (MC signal into data events)
1-2% level agreement in data mc comparison
Reco time- track propagation in torus B field in forward detector
Caching B field done by service
Looking at ML for track pattern recognition / noise rejection

Questions
Replace time-critical components in java with C++? Haven't tried yet - but have the capability given the micro-service architecture
OSG for simulation, Nersc for reconstruction [opposite split of the way Atlas does it..] - this is historical , no specific reason
Alignment - how is the excellent precision reached - straight tracks with field off data. Millepede software is used for some of the cross checks (default is in-house written)

17:00-17:20: GPUs for IceCube (reconstruction/simulation/deployment) - RIEDEL Benedikt

Broad science program - Events in IceCube - up-going tracks, isolated energy deposit
Ice is one of the clearest materials on earth - 300m absorption length
Complexities due to tilt, bore hole, polar angle variance
Other unknowns - eg, birefringence
Algorithm - brute force algorithm via iterative procedure to account for more physical effects
Threading parallelization by photon (queue of photons)
GPUs via OpenCL
Single precision is good enough-
300 GPUs running on average
Reconstruction- Max likelihood technique with timing, position, noise, ice model
Ice models can introduce large systematic effects
Recent improvements brought 10x increase in precision of location

Questions -
Lines of code in OpenCL - 300 (!) - scattering function plus absobion function
KNL performance comparisons - 1 Skylake node = 1 KNL node
Data from detector to computing - 3 TB/day to 80 GB/day reduced on site.
Transfered by satellite. Then high fidelity via ship once per year.

17:20-17:30: Discussion

Pre-notes for discussion sessions, from pre-workshop chats with the session speakers:

RTA Session:
===

Topic 1:

For LHCb, RTA is pretty straightforward as an identification between offline and trigger is possible. In CMS and ATLAS there is a tradeoff between RTA (more data, less information/precision) and offline analysis (less data, full quality). How do we implement a good feedback loop between trigger and offline?

Topic 2:

How to find more motivations for doing this kind of things beyond searches, and match them to the work needed for objects.

Topic 3 (from ALICE):

We will have out-of-bunch pile-up (only 10% of our continuous readout data is interesting). How do you control pile-up in ATLAS and CMS in real-time analyses?

Reco/ML Session (more heterogeneous):
===

Topic 1 (from JLab): what are the tools used for reconstruction in LHC experiments? How helpful is machine learning for you, in bulk reconstruction mode rather than testing/analysis?

Topic 2 (from IceCube): can you give some details on the reconstruction chain of LHC experiments?

Topic 3 (from IceCube/CMS):
- what is done in terms of I/O (see yesterday's session)
- are there tools to access distributed GPUs that are already usable?

Topic 4 (from LHCb):
- would it be useful to do ML vertexing in 2D for ATLAS and CMS given the beamspot width? It could be used to reject pile-up.
- what about a fourth dimension (timing detectors for HL-LHC)?

Education and Training

StarterKit

- Why did you disconnect from the Carpentries?
    - The level of the material was not right for our community. The level was too low and the rate was a bit slow. Carpentries actually encouraged us to learn from them, but to adapt the material.
    - Could we re-engage with them and contribute back?
- Financial support and career recognition needs addressed at the funding agency level, not inside each experiment.

HSF Training Survey

- Debugger training - learning how to use those tools are crucial. Also editors are now much more powerful and we don't use them enough.
    - IDEs are greatly hampered by the size of out software stacks.
- People do seem to like to learn from written material.
- Need to distinguish between overall training programme in a language and use of HOWTOs that give an answer to a very specific question (Carpentry vs. StackOverflow).
- Do we as a community need to invest in suitable C++ material? The online material here seems much less generally useful (or at the right level) than in, e.g., python or git.

First-HEP training program

- What about careers? How do we improve this.
    - This is an aspect, albeit not well highlighted in the slides.
- Careers
    - Giving students appropriate and transferable skills (90% will leave the field)
    - For people who stay, labs do have career paths. Universities are many, many local battles to win. In the US some discussion with FAs is possible.
    - Recognition of career path in hardware has been done in CMS - extend to software matters?
- FIRST-HEP is about developing sustaining and scalable training, not specific pieces of material.

Software and Computing Training at Jefferson Lab

- What about Java training?
    - Not done formally, but neither is C++
- Complex tools like RooFit get picked up by students, but without a full understanding, so easy to make mistakes. Can discuss some of this with universities.

- We do ask students to do things they don't know how to do (nature of the field). Students will seek help, but informal passing on of knowledge can propagate mistakes as wekk.
- An induction initial training would be a good way to establish a baseline. Synchronise this with new students starting their PHDs?
    - It's a good idea to consider, but right now it's more 'on the job' training.
    - If material is well prepared in advance it's much easier to run training (lowers costs).

PyHEP

Coffea

- If the data doesn't fit in memory chunking is done - is this manual or automatic?
    - At the moment it's manual, but no inherent reason it could not be automated.
- x1.5 faster in ROOT, but what are you comparing? Isn't LZMA the bottleneck?
    - Some questions about how much data is read in the two cases, block arrays, etc.
    - To be followed up offline - important technical details to be understood.
- Can compare with ROOT standard analyses that use open data.
- Any plans to use Dask or Xarray, particularly with lazy evaluation?
    - Started investigations, but not conclusive yet.

Anaconda - numba, packaging

- Community likes to use very modern C++, e.g. C++17, but needs a much more modern compiler.
    - Conda5 does package compilers, currently gcc7.3.
- Is there support for unusual environment variable settings, that we often need.
    - Activating an environment will also source scripts, which can then set environment variables (and deactivate them later)
- If your code only uses numpy, would numba be able to speed it up?
    - Depends on the code - sometimes it can give you better optimisations. But if pure BLAS calls already, probably not much. Both systems now try to avoid intermediates.
- Can mini-conda be slimmed down even more?
    - Tricks are employed: hard links, strip binaries, delete package caches.
    - "conda clean" will clean caches.

Conda: a complete reproducible ROOT environment in under 5 minutes

- Can you have multiple ROOTs available?

    - Yes, in each environment you can have different ROOTs. At the moment limited to the versions of ROOT that have been compiled (6.14 and 6.16).
- Windows support will come when ROOT supports 64bit builds on Windows.
- More discussion in the packaging session...

The zfit package

- There are some problems in fitting that it's not possible to do with TensorFlow.
    - At the moment haven't found any hard blockers (not tested everything!), but the graph model can't express everything.
- Has there been a comparison between minimisers?
    - Minuit is better than TF minimisers in our cases (second order); we just interface to these tools.
- Have you considered other backends, like PyTorch?
    - For the user this is not explicit. At the moment we use TF in the back, but in the future it could be done.
- Long term support?
    - There is a team of people supporting it - no immediate concerns (at least as good as RooFit...?)

lichen and h5hep: mimicking ROOT functionality with python-only libraries

-

Software Development Tools

Performance Monitors/Profilers

- TAU: Tuning and analysis utilities from University of Oregon. Promising but needs more work
- Attila: Do we want to mostly talk about x86 linux? Gperftools practically useless on Mac.
- There was a comment that Vtune has very poor performance with locks.
- Have you tried "open speed shop (oss)" which the Geant4 collaboration settled on?
- Combined python/C++ profiler? TAU tries to do exactly that.
- Any open source or free tools that handle multi threaded applications? TAU, perf,
- Iprof had problems on CMS as well but could be fixed easily, follow up offline
- There was a comment that Valgrind doesn't work with jemalloc anymore.
- Gordon asked about finding coding problems: Recommendation to look into sanitizers (ubsan) shipped with clang and gcc
- Discussion sounds like a good use case for a tools survey. Also prmon and trident

- Profiling of GPU and communication overhead? TAU claims to be able, so are GPU vendor proprietary ones.
- Linux perf pretty good these days in conjunction with a GUI called osbat?

Static Analyzers

- Coverity: Problems keeping up with C++ standard. Free for open source and travis integration. Last version reports C++17 support.
- ATLAS looking to rewrite package dependencies in the build system which got lost when switching to CMake. Looking for collaboration
- Conda does over/underlinking tests.
- Clang static analyzers from CMS on github: https://github.com/cms-sw/cmssw/tree/master/Utilities/StaticAnalyzers

Packaging

- What makes Conda less suitable? For production build you want everything locked to exactly these versions in this way.
    - At build time you can set a yaml file of the exact versions
    - At install time you can have an environment file to specify the exact same
- Pere: We might not want to have two tools, we want full control for all use cases
- If we go to spack, how would that affect e.g. ATLAS?
    - LCG changes will try hard to not interrupt experiments
    - Maybe some additional yaml/python files to build on top of that.
    - One could base ATLAS on LCG but it would allow to replace one package in the LCG stack
- Why did Spack not use Conda or the other way round
    - Mostly historic
    - Conda was rewritten and needed more platforms and architectures
    - Spack was looking around at the time and didn't find anything usable.
- How easy is it to modify a recipe from upstream?
    - Spack: install root with this version of clang with this and this options.
    - Yaml: define aliases in yaml file to specify concrete option for the stack
    - Conda: parameters for recipes are possible.
- HepOS libs part of the taskforce?
    - Conclusion was either build very deep or very shallow.
    - Production should probably build deep but the tools allow to make exceptions
    - One of the features: They don't setup LD_LIBRARY but use RPATH.
    - ATLAS doesn't use RPATH because then you cannot patch/override something so all RPATH were removed.
    - RPATH has big advantages like not interrupting normal work with system binaries
    - LD_PRELOAD can overload RPATH
    - General agreement that RPATH is better than LD_LIBRARY_PATH