HSF WLCG Virtual Workshop on New Architectures, Portability, and Sustainability

Workshop Notebook

The notebook here is for comments, discussions and any issues that should be followed up after the workshop. There are assigned note-takers for each session, but anyone is welcome to add their comments and observations.

If you wish to raise a discussion point here ***please give your name*** so that we know who asked the question.

General Items

Monday

Code Portability (Charles Leggett) [Participants: ~175]

How much tension is there between portability and efficient code? To what extent can toolchains and frameworks help and to what extent will dedicated code be needed? [Roel Aaij, Nikhef]

A: There is a lot of tension between them. Code portability is more important than efficient code: the goal is to get access to HPC systems and we'll not get it if we cannot use accelerators. As our code doesn't map well onto GPUs, hard to eke out last 5% of performance.

The portability solution adopted by ALICE and LHCb focusses on C++14 (for device code) combined with minimal #ifdef in a few or even single header. Backed up with good practices for the code this supports CPU, CUDA and HIP builds without additional layers. The focus has been on GPU performance of the code, but perhaps the reduced complexity is worth some reduction in performance on some architectures in certain cases?

A: Reducing complexity is great. To run on HIP, the CUDA code would require translation (needs more investigation), but you won't be able to run on Intel GPUs. It may be a great way to start. Will include details in metric document.

I always get worried about the schedules for these efforts. It is obviously too late for these studies to have a real impact on LHCb, and I would think there was time for Run 4 for ATLAS and CMS. But I worry we sometimes get artificially constrained by the computing TDR schedules. Has someone worked backwards from big simulation production for Run 4, likely to be the first really big loads, to see how well this schedule matches? I know this isn't a technical issue, so sorry! [Gordon Watts, UWashington]

A: Too late have impact on LHCb as they have decided to use GPUs for the next run and have already decided on hardware and software. We're looking at Run-4 (HL-LHC) timetables. We're getting constrained (artificially) by the computing TDR. We know that the sooner we come to a conclusion the better for the community.

You list "backend decided at compile time" in almost all of the things you mention. Since the goal is C++, I would assume this was almost always the case. Otherwise we'd move to a dynamic system, and have to build something like ROOT's JIT. Which brings me to distribution & testing: this is going to be awful. I just think about testing when we move to a new version of the OS (how long CentOS validation took)... And distribution - will all of this be accessible from a container or similar? You reference this in your "metrics" slide - but if there is a way to evaluate any differences in these layers on distribution, testing, etc., explicitly, I think you should (e.g. Validation - not just validating the code, but how hard is it to validate). [Gordon Watts, UWashington]

A: Many require compile time optimisation. Sycl is very nice because it requires little. The whole workflow may need to change. In many cases, HPUs are simpler because you know the architecture well. Running on the GRID will be much more challenging - may require changes to protocols. I'm hoping that as time goes by all the various backends get integrated into a single compiler. More likely for Kokkos, Alpaka and Syckl than openML which requires more manual intervention.

Has any thought been given to having a scheduler running on the GPU? My understanding of the LHCb Allen project is that they have something that plays the same role as Gaudi in the ATLAS experiment - dispatching algorithms etc., to build a full event. Would any of these backends affect our ability to do this (this question came to me as I was looking at the CMS Patatrack backup slide - and realizing this was an isolated algorithm and we are going to want to run many… and perhaps eventually move a chunk of the reco over there if it makes sense). Of course, Charles L's simulation work may tell us a bit about if this idea is worth it for these workloads. [Gordon Watts, UWashington]

A: You don't tend to run schedulers on GPUs -- they are run on CPUs, e.g. allow memory to be reused. The various experiments have looked at this and some experiments have made inroads into doing this. It will be done during the optimization process rather than at the start.

Please don't confuse "ATLAS" with the "HEP community" [Andrea Bocci, CERN]

A: We have a broad range of experiments that we are looking at.

I have a question about the various solutions being developed outside HEP. So far, you seem to indicate we should wait and see what happens -- which library can support which backend. Do you think it would be worthwhile to try to contribute to push in one direction or another? [Andrea Bocci, CERN).

A: I have been pushing for some time that we should be providing input to vendors, etc, to make sure that we have solutions for all our problems. Sycl has been making progress into integrating NVIDIA backends because people have said they won't use the code unless it provides that functionality. We have tried to encourage NVIDIA to provide cross-platform solutions (including support from the DOE). I would like to see more pressure from the DOE at the various labs, which are currently choosing different solutions. I would like to see more effort from the funding agencies to ensure solutions will work at all labs and across the world. We see a lot of involvement from HEP physicists and developments (through bug reports and merge requests). There is good communication between developers and the engineers.

Follow up: I was mostly talking about direct contributions rather than pushing.

A: I agree and think this has a lot of impact.
Attila: I hope that eventually it becomes the C++ standard, but we should try to figure out which one we would like to become the standard.

Q. on presentation, slide 7. Given that Kokkos already has abstractions for CUDA and HIP, how interesting is the SyCL support? Is it likely to be used in your opinion? [Ed Moyse, UMass]

A: I think SyCL is very interesting because it has broad adoption of backends. I like how you can target the various backends at runtime. I hope that one day we'll see the AMD backend as well. A lot of time is spent optimising Kokkos and its performance on various architectures, but I don't think it'll be the defining metric. It does read easily so if it gets adopted as a C++ standard, we should follow it carefully.

Comment on the availability of effort. Students and postdocs need up-to-date skills in order to advance and get their next jobs. Technologies that are likely to still be around in the future are more likely for them to invest time to learn and get good at so they can still be relevant in their next job. And skills in technologies that are too old also tend not to be as marketable unless they are very stable.

A: It's good to have the languages in use on your resume. It's not hard to pick up these languages as they are C++ based. Once you learn one of them deeply (e.g. how to move data back and forth) the rest is just syntactics. I don't like it would look bad to have one on your resume because it would show that you know how to use one of them.

Q. How will the use of containerization help or hurt this (various CPU- accelerator combinations)? [Doug Benjamin, ANL]

A: I think it will help and make life a lot easier. Won't require needing to understand how to compile at the site. It depends on how much you have to include in the container -- don't want to distribute 50GB of data each time. I'm hoping that this won't be the case. As these libraries are rather interoperable, e.g. can compile root stack with gcc. Can compile specific modules with SyCL. Can do the same thing with Kokkos. The amount is probably not as bad as the worst case projects.

Q: Very nice, but one general statement that concerns me - that this community has to use these machines and its the only way forward. The ATLAS computing CDR shows that we know how to solve the problem with CPU but we don't know how to use GPUs. There is also a risk if we give up on the development path that we know how to succeed on. I don't know how to weigh these two risks and the way forward.

It's premature to make a selection. While we may be able to get away with not using HPCs, they are the wave of the future. The online environment is very different. The hardware ecosystem is changing and more will be available in a few years. I think it would be a mistake from our community point-of-view if we can only put an NVIDIA GPU into a GRID node. If someone is going to start working right now, I think it's best if they start by using CUDA. Easy to port to another parallel language. You don't want to ignore the entire accelerator ecosystem at this point. It's not an easy problem.

Heterogeneous Architectures and Detector Simulation (Guilherme Amadio) [Participants: ~175]

Could simulation benefit more from zero-cost abstraction techniques (instead of virtual calls)? Compile time optimisation might even be worthwhile, given the huge CPU budget we use. [Graeme Stewart, CERN SFT+ATLAS]
→ *comment postponed to dedicated simulation meeting*

Yes. The first thing would be to just not compile in physics models and other functionality not needed in HEP simulations. This has been discussed before and may be possible in Geant4 in the future. Consolidating several physics models into a single one is another approach that is being tried to simplify class structure (with a generic class for most EM models, written by Mihaly Novak). The other approach, if a new code is developed, is to try to follow a model similar to how OpenGL and Vulkan drivers work, by having a well defined set of API symbols that get populated at runtime (i.e. assigned to a function pointer) with the implementation that you need/want (the implementation from your GPU driver in the case of OpenGL/Vulkan, assigning a physics model implementation to a symbol representing a particular process in the case of simulation).
(G. Amadio)

Are there hidden costs in offloading computation to accelerators, in hybrid code that runs part on CPU and part on GPU? Is there some kind of threshold to make offloading convenient? [Claudio Grandi, INFN]

The model that we want to have is that the CPU is a big orchestrator and calls many parts of the code simultaneously. This hides the cost. You might send the entire code for electrons and muons to GPU, while the rest of the CPU is dealing with other parts (i.e. hadronic physics models). You run most on the GPU and hide the communication cost by running things concurrently.

Slide 10 - the G4 Microarchitecture Utilization Overview - can it be understood what parts of the code are causing the different stalls? For example, the I/O code vs the material propagator, vs the magnetic field calculations, etc.? This is, I suppose, a follow on question to Graeme's question above. [Gordon Watts, UWashington]

- Your comments on each thread following a single particle, and then many threads, is at least a partial answer of this.

→ *comment postponed to dedicated simulation (?) meeting*

Yes, we know that most of the instruction and data cache misses happen in functions that are trying to access geometry data (like G4Navigator::LocateGlobalPointAndSetup) or physics data (various functions that access cross section data). More details can be seen here: https://indico.cern.ch/event/809405/contributions/3629814/

Geant4's work with CUDA as a backend - will the limit the places that G4 can run? Could the G4 work and the work from the previous talk be integrated at some point? [Gordon Watts, UWashington]
→ *comment postponed to dedicated simulation (?) meeting*
CUDA is being used for prototyping. Evaluation of portability frameworks is a planned future work item. (J. Apostolakis.)

On the final thoughts slide - "hardware evolution save us again" - I think for the initial HL-LHC timeframe the answer is no - the HPC's that will be running in that era have been spec'd now. So those will have to be supported - even if A64FX machines (or whatever) become available. [Gordon Watts, UWashington]. Sorry to be negative in these comments - but the huge task ahead here seems rather daunting…

- **The comments about ray tracing scare me, for example - as I think those are not going to be in any large machines in the near future.**

Even though the V100s are not supported yet in hardware, the same thing can be done in software so its supported in the current architecture but at the price of performance. I don't know if there won't' be a A64FX machine in 10 years or not. The first machine will come online this year so there is the hope that more machines like this will exist by the time that we start. We could separate the full CPU budget by machine. e.g. use machines more suitable for simulation and other machines for other workflows.

Raytracing doesn't require special hardware to start with, this is a super well studied field. R&D projects should focus on the existing libraries and start with simple steps, e.g. creating a triangular mesh geometry representation for Geant4 shapes and test these with existing libraries. [Andreas Salzburger, CERN/ATLAS].

- [Gordon] - I'm totally in favor of R&D projects. Don't get me wrong. But it feels like we need more R&D on just the straight G4… Given where we are now, and how long we have to get G4 "ready" I'm not sure where these will fit in. For Run 4++ these make more sense, certainly!

→ *comment rather than question, noted*

Andreas, I fully agree with you, if converting our geometries to triangles becomes possible (it involves development by itself as no GDML reader can do it now), this would probably be the best way to take advantage of hardware support for ray tracing. (G. Amadio)

Is ray tracing feasible for curved trajectories, *i.e.* from charged particles ? [Andrea Bocci, CERN/CMS]

There is a short segment that can be checked for intersection with the detector and then you can go and do the physics processes. It means that we can exploit the ray tracing hardware in particle transport. We can't use the triangle geometry. The key is the traversal of the complex geometry structure.

Andrea: Are the segments straight or curved?

A: They are straight. Even G4 right now does short steps using a short line. (More: The curved trajectory is broken into straight sections, each one limited by a maximum sagitta. High momentum tracks can go far, if there is no boundary.)

Note: Tesla T4 have hardware ray-tracing, and next NVIDIA server-grade GPUs are likely to integrate it as well.

I understand that the particle transport part is now available in CUDA (slide 15), which is great. What is the main technical difficulty in compiling also some physics model with cuda? (basically extending your shading model to something more realistic) [Davide Costanzo, Sheffield, ATLAS]

A: The complexity is in managing the result of the physics and also happens in general ray tracing problems. Interactions need to be accounted for by changing the state of the renderer. In physics, you can produce new particles and this needs to be managed (does it go back to the CPU? Push to dynamic stack on GPU). We need to learn how to deal with these things as part of putting the code on GPUs. Once you have one model adding more models is not a problem. The challenge is learning to deal with secondaries.

Followup to the two previous comments: When suggested so far, the response has been to veto it from a validation point of view, multiple code paths would all need full physics validation for all variants of software that could run. This is the reason why there is a grid discussion on forcibly decommission the oldest hardware in order to use faster CPU features, see for instance: https://indico.cern.ch/event/813757/contributions/3698522/attachments/1969803/3276878/2020-01-15-GDB-CPUsupportInWLCG-v2.pdf [Mattias Wadenstein, NDGF]

→ *comment postponed to Wednesday's validation discussion*

Need to clarify that the decision to pursue the vector CPU target and 'sideline' the GPU target was not influenced by external funding from a company, but taken due to manpower issues (which Guilherme did mention) and the feedback received from the reviewers (including from LHC/HEP experiments) during the GeantV review of October 2016 aka "HEP Software Community Meeting on GeantV R&D" in order to focus the effort on delivering a working prototype on vector CPUs. [J. Apostolakis, CERN]

Thanks for the clarification, John, I didn't want to imply that decisions were based on funding. (G. Amadio)

TensorFlow as a Compute Engine [Participants: ~100]

Could TF, JAX, etc., fit into a wider class of algorithms we have to deal with in HEP? I've seen this in zFIT, pyhf, etc. - which are fitters, and obviously a good match. [Gordon Watts, UWashington] (sorry, I had to leave for a second meeting - will read answer here when I have a moment)

- → *comment to be answered offline*

As DUNE has begun to incorporate TF into part of the reconstruction, we have seen that it by default wants to make use of any and all available CPU on the worker node unless you explicitly cap it, while HEP jobs have traditionally requested a specific number of cores. Have there been any studies so far on what the optimum choice might be (number of cores per job, whole node pilot for TF jobs, etc.) between restricting such jobs to a certain number cores vs. giving them the entire worker node? [Ken Herner, FNAL]

A: The card is shared between 2 or 3 users usually and limit the RAM. I don't know if this is optimal sharing. I haven't done any thorough benchmarking. It depends on the application and how it is parallelised. You can have threads and also how you split threads. If you're using the GPU, the main load is on the GPU if it is doable and the CPU usage is very low. With GPUs you limit the RAM and then you can run several jobs in parallel and they share the CUDA cores, but it's not obvious to me how this is done -- I think code only runs on a single GPU core with no sharing between the GPU cores.

In Bonn, we have also seen really hefty inefficiencies of TF 2.0 when it uses a significant number of CPU cores (>=8, for example, all cores it sees), since a lot of inter-thread locking and yield() calls are observed and almost all CPU time goes into context switching. Using all available configuration parameters and environment variables, it can be reduced to a few cores, and becomes much faster then (and uses less memory). Only capping it via cgroups however still causes a lot of context switches. With GPUs, it does not suffer from this problem too much, but we should still study such issues in-depth. Long-standing upstream bug reports on this exist (e.g. #29968) and this would likely be very harmful especially in Grid-like or HPC-like environments, and spoils any direct comparison between CPU and GPU. [Oliver Freyermuth, Uni Bonn]
→ *Thanks for the comment, kept for notes*

We have found in using Tensorflow as a performance portability library that if you stray too far from the types of operations used for Machine Learning you may find it has never been optimized for your architecture. This includes sparse math operations, fast fourier transforms, and likely others. By depending on these libraries you may find yourself either implementing the optimizations yourself or contributing pull requests to Tensorflow. It may make more sense for HEP to develop portable math libraries that are useful for processing lists of events composed of particles, etc. This may have improved in TF 2, but we found TF 1.14 to be lacking. [Taylor Childers, Argonne Nat. Lab, US]
→ *Thanks for the comment, kept for notes*

In the amplitude analysis, "fit once, publish the paper, and more or less forget about it" invites a question: Should we preserve the ability to refit in the future, especially with the addition of new data or other experiments' results. To what extent do the DOE-required data management plans include final results extraction steps? The bare minimum is to provide machine-readable versions of published plots, but there is a lot more use to go back a step. [Tom Junk, FNAL]

A: You need to preserve the ability to refit and be reproducible. I don’t think this ability is too dependent on the performance. I would like to have different backends for this library to switch between tensorflow and numpy. You must have the ability to reproduce the calculations on something that is stable (even if tensorflow support might be dropped). Whenever you want real reproducibility you need containerisation of your whole environment. There could be effects from changing the hardware, e.g. floating point is slightly different.

Comment on slide 19: the increase in memory usage when dealing with large datasets can be compensated by more sophisticated IO pipelines enabled by TF data API, where in essence one keeps in memory only a batch of the total data. This has the price of lower processing speed as there is a constant data transfer between storage, memory and CPU/GPU. [Vangelis Kourlitis, ANL]

Tuesday

Application Frameworks [Participants: up to 150]

Talk: *Heterogeneous Experimental Frameworks (Speaker: Attila Krasznahorkay)*

Notes

Basics of accelerator programming to build GPU applications: starts on the CPU, then passed to the GPU. The GPU is idle when the CPU part is running and the CPU idle when the GPU is running.

Categories of framework organisation: a) separate processes share data in memory, each process expected to use one CPU core/thread b) framework designed to use the accelerator, and c) hybrid use of CPU and GPU where a multithreaded CPU process try to optimize both the use of the CPU and the GPU by appropriately scheduling algorithms (CMS approach, ATLAS goal)

Separate processes: used by ALICE-FAIR (ALFA) for online reco and data analysis. Pros: async. API for offloading; simpler way; Cons: dynamic load-balancing is hard to achieve

Primary Accelerator Processing: used by LHCb for 1st trigger level. All mem allocated at the start of the application. Steps config happens at compile time. **[finish]** Pros: Code only works on specific accelerator. High performance achieved. Cons: code only runs at maximum efficiency on a certain type of HW; learning curve is steep (knowledge of the architecture is needed)

Hybrid approach: ATLAS and CMS process large events in many different and long running steps. These can be run in threads [finish]

Hybrid Async. Execution: separation of modules in pre and post execution steps, with the goal to keep CPU efficiency as high as possible. **[finish]** Pros: can be used with very large apps. to use efficiently all of the underlying resources. TBB or HPX can be used to maximise CPU usage. Cons: **[finish]**

Summary: different ways to include accelerators in the frameworks.

Questions / Discussion #1

Q: Slide 7 “The learning curve can be steep” - after listening to Charles’ talk yesterday - is there anyway we can avoid this? Will this work always be specialized? Will there ever be a task a starting graduate student can pick off early in their career (50% of their time for 1 year type of task). Could we modify things so that we don’t get every last bit of performance (say

lose 10% of the GPU's performance) and get a x2 ease in "programming use"? **[Gordon Watts, UW Seattle]**
A: We should use clever portability solutions, which are available in the market. But, this might not take the full performance of the accelerators, however, we don't use CPUs fully efficiently as well. We need to learn how to use these resources in the most efficient way, but we can use portability solutions.

→ Very nice talk, thanks Attila! [Gordon]

Q: **Paolo:** shouldn't we first convince ourselves that accelerators can run our modules in the simplest possible setup from the fwk point of view, and then try complex scheduling schemes to extract the last 10% of performance? Shouldn't experts like Attila work on algorithms now?
A: Good algorithms might not be found soon, so scheduling schemes can be tried, indeed the benefit in the performance might be a bit better than 10%.
Comment : we probably also have to deal in the future with very heterogeneous sites (different CPU/GPU mixes) and there the semi-smart scheduler is really needed (very different from the HLT). [**Graeme**]

Comment: Synchronous offloading shouldn't be off the table: NVidia has made significant progress in its drivers since the beginning of the year to allow the offloading thread on the CPU to do other work while it's waiting for the GPU to finish. **[Charles Leggett]**
A: APIs cannot be used for multiple threads. Calculations need to be serialized. Sync. waiting at the end of calculations, we need to work on this, to maximize the efficiency.

A. Bocci: CUDA use seems not very much complex as writing CMSSW code. What is difficult is to identify the patterns (mem usage, scheduling eff.). Implementing algorithms seems doable. Attila: learning the basics is needed, once the framework is understood, it is not a big step to jump and re-write things using CUDA.

Charles Leggett: how difficult is it to jump into, it depends on the previous expertise. For newcomers, the hard work would already be done. What's needed to be modified will come later.

Liz: from survey (early days before the WG was properly established) - different directions to address different domains. Could this be quantified? How to use metrics to know how to evolve? Attila: LHCb solution seems very promising and can be used as a guidance, at some point one has to start writing the code, but it depends on the solution/direction it might imply an overhead **(polish later - jfm)**

Talk: *Support for heterogeneous Computing in CMSSW (Speaker: Matti Kortelainen)*

Notes

CMSSW implements multi-threading using Intel TBB
Threads blocked until the process their task
External worker concept: to interface with anything external to the fwk - block waits replaced to callback-style solution. The system is capable of delivering exceptions. Two stages: acquire() and produce()
WFs should run at any site; same configuration for all jobs in a WF (hw agnostic / hash mechanism to separate WFs). Keep CPU and non-CPU algorithms separated. Event by event processing.
Switch mechanism for producers: specifying multiple producers associated to the same data label.
Supporting CUDA algorithms for CMS code: some considerations based on the CMS code context and performance shown; allow using CPU when the GPU is running an algorithm. Mechanism enabled to share resources for a chain of algorithms. Transferring data GPU-CPU only when needed. → The model appears to work pretty well (presented at CHEP). Limiting factors identified: many small operations overheads, explore scheduling mechanisms to improve performance.
Performance measurements: external worker vs. blocking wait on the event throughput.
Support for remote services -

Building blocks to explore non-CPU resources available in CMSSW
Developed support for CUDA and ML inference on external services using generic building blocks.

Questions

Q: **Paolo**: how will cmssw (or others) handle offloading of non-event data (e.g. alignment), including when a GPU is processing data from multiple events?
A: Simple - plain arrays to GPUs. Pointers to memory - calibration and alignment data. Threads do not have to wait *the end of the synchronisation: GPUs will deal with it.*

Q: Does CMS have to handle merging the event histories in the case where a switch producer is used (as on page 6)? **[Kyle Knoepfel, FNAL]**
A: Merging is not done, *per se*, as events are not required to have consistent event histories.

Liz: CUDA interface for developers could it become lock free? A. Unlikely and CMS are hammering the mutex at 100kHz.
Attila: memory managers and allocators are implemented at runtime in the most efficient way by the developers. There are solutions that are less efficient but faster, it might help. Could this be a common piece of work for multiple experiments to avoid everyone having to reinvent this wheel? (something similar to what Google did with tcmalloc on CPUs).

A. Bocci: the latest version of CUDA includes more efficient memory management.
See https://devblogs.nvidia.com/introducing-low-level-gpu-virtual-memory-management/ .
This will need to be wrapped in a more user-friendly library before it can be widely used.

Discussion (Part 2)

For the LHCb use-case one of the developments that enabled efficient use of the GPU was to move to processing events in batches of O(1k) events and O(100) MB in size. This necessitated development outside of Gaudi. How can this efficiently (in terms of development effort) be taken into account in frameworks? **[Roel Aaij, Nikhef]**

We have long had CPU frameworks for orchestration of modules/algorithms and provision of common services. Is there scope for similar developments for GPUs? **[Graeme Stewart, CERN SFT+ATLAS]**

Workload Management [Participants: ~ 145]

Talk: CMS WMS heterogeneous resources access status (Speaker: Antonio Pérez-Calero Yzquierdo)

Notes
How to include heterogeneous resources in the CMS SI.
GPUs: offline and online → HLT farm during Run3
Limited number of GPUs available.
How these resources are allocated and how to schedule work on them? Target or discovering the GPU resources (preferred)?
The execution times can span an order of magnitude, depending on the resource in which the job run
How CMS creates and handles a WF is shown (in a late binding model)
How/when to use GPU resources? Simulation steps are chained in a single task, which minimizes the complexity of the job and alleviates the management of the WF. How can this be integrated when using the GPUs?
Several ways to include HPCs in the Global Pool in CMS, but Antonio shows the strategy on how to include the GPUs in CMS pilots
Auto-discovery of GPUs is probably a bad idea: which layer? If GPUs are shared? The GPU needs to negotiated with the local systems: HTCondor-CE & BS at sites
Payloads executed in singularity containers; specialized SW distributed in images in CVMFS; --nv option as well
+ ..

Talk: ATLAS WMS heterogeneous resources access status (Speaker: Doug Benjamin)

Notes
PanDa ecosystem presented (ProdSys): Orchestrates all of the ATLAS WFs. Harvester to access HPC resources
Harvester structure: can be sitting at the edge of the HPC, or centrally - ASGC using it for other purposes, as well

Jumbo jobs: large scale fine grain simulation WFs. Event service, can be stopped at any time. The work can be then fit into an HPC.
N nodes for T time → Yoda → G4 sim at NERSC efficient running. These techniques can be used beyond, for example for ML approaches
Panda was adapted to work with Jumbo Jobs
New WFs: active learning / ML / iDDs / Madgraph and GPUs / G4 w/GPUs
Active Learning: process data based on old selections and results
iDDs: service to transform and deliver data to consumers - not an specific ATLAS project. Gives flexibility to support new WFs → first test was to do Data Carousel with iDDs. New component in PanDa: reactor.
GPU resources are available, not only at HPCs. Accessible via PanDA. Running times are widespread. All NVidia GPUs.
Opportunistic HPCs: GPU in NERSC Cori - small allocation for testing. Constraints are very different from those in HTCs: job running time limits depend on the number of jobs submitted.
Challenges: wide variety of GPUs. Scheduling policies vary. Whole node scheduling.
Conclusions: developed tools and techniques to exploit HPCs. New tools (iDDS) and changes to PanDA will allow for a variety of new workflows; A wide variety of CPU/GPU/other accelerator combinations will require additional tools and monitoring for accurate brokerage of work to these resources

---

Talk: k8s heterogeneous resources access (Speaker: Ricardo Rocha)

Notes

CERN-IT resources available. How to access them: bare metal or VMs.
They are integrated in CERN Openstack. K8s integration straightforward: managed by a helm chart, or can be done with the GPU operator. Interest in including them in K8s for ML purposes: sharing not possible.
Evaluation of the performance penalty of vGPUs ongoing
AMD/Intel - using SR-IOV for non Nvidia vendors?

Questions / Discussion

Q: **[A. Krasznahorkay]** What about AMD devices in the future? Could they be integrated in a similar way to NVidia's vGPU offering?
A: Ricardo → using SR-IOV for non Nvidia vendors, this is the protocol, supported for AMD and Intel.

Q: **[Paul Laycock, BNL]** The event processing frameworks and workflow management systems are both trying to optimise CPU/non-CPU usage but there was nothing on interplay between the two - does this interplay not exist? Testing independent optimisation strategies is good, but at some point the throughput and "keep your funding agency happy plot" will rely

on efficient interplay between the two. Is whole-node processing the right interface between the two?
A: Optimal usage with the interplay → **Doug (ATLAS):** it depends on the WF. Analysis or RECO, it depends on what you try to do in the WF. WF management systems will need info to properly make decisions. Certain types of running in HPCs do not charge much, others prefer to get big jobs, whole-nodes, … Multithreaded fwks in whole-nodes might be degraded in efficiency, which seems an issue | → **Antonio (CMS):** resources and requests well defined: tags. Each job might not use the entire node. But we can run several tasks simultaneously in a pilot: filling the node, and with different types of tasks inside. This functionality could be extended, the whole node can be used in this partitionable model. The condor matchmaking will be helpful for this. This helps in the overall throughput and efficiency of the tasks being executed in the machines | → Paul: event info passed to WF management systems will be needed | → Graeme: at the task level info, this is needed

Q and comments: Alessandra: type of GPU is typically considered, do the fwk be tied to a type of GPU? A: Lukas: with K8s one can define scalable WF to be scaled to 1 or N-GPUs. | Chris: in CMS. WM tells the pilot to tell how many cores to be used. For GPUs this might be simpler, in a pilot we can handle that (you have N GPUs - use 1 per payload) | Alessandra: but different GPUs have different features. WMs and app. Fwk interaction to get a better description of these resources? Is this needed? | Antonio: # of CPUs can change in a slot. The information can be passed from the pilot and incorporated in the condor autodiscovery, the info can be passed from the slot to the job, this is already happening with CPUs. This principle can work on GPUs | Alessandra: if there are AMD GPUs, then it should work transparently? AMD and NVidia GPUs are different in use. | Attila: portability will mature, now this can't be done, but the goal is that this can be done. | Doug: compute power, how to handle that? Different execution times. | Chris: switch producer (can be used or not - CMSSW code) → Could this be made experiment agnostic? | Matti: the idea can be transferred, but this is embedded in CMSSW. A. Bocci: CMS has made a decision and chose that the job config does not change if accelerator is present or not. But this can be adapted. Ricardo: granularity of info needed? Chris: which binary we are loading? Antonio: CUDA version can be obtained. Doug: time limits fix which type of task you want to execute there. Alessandra: this info needs to be known by the WMs beforehand. Antonio; yes, this is a problem for operations. We also have dispersion in CPU power. Alessandra: the spread might be bigger than in CPUs. This is a dramatic difference. Antonio: KNLs are very different as other CPUs, this is monitored and the runtime is predicted. Benchmarking and then predict and select to exploit the resources.

Simone: CMS online - 20% of the code meant to run in GPUs to cut the costs. If this is achievable offline, the same applies. Heterogeneous environments might be difficult to handle. Sites won't deploy the same GPUs for several years ahead.

Q [Ricardo, CERN]: Is there a requirement for multi-GPU jobs and any specific interconnects? Charles: which level of PCI? PCI-4. Ricardo: Which are the requirements for the WLs? This can be followed offline. Charles: WLs will change, currently just prototypes. We shouldn't lock ourselves taking the existing WLs as reference.

From yesterday's session:

Comment on slide 10: "`Tesla V100` works great, but $8000…" and sites are already thinking nvidia licence restrictions imposing to buy high end GPUs in data centres are problematic. They are now also imposing licences on virtual environments. TF also tends to occupy all the resources it sees and this is problematic with some batch systems set ups maybe this also for tomorrow WMS session. **[Alessandra Forti, Manchester/ATLAS]**

Comment: the fact that the minimum level of hardware available on the grid does not support AVX2 is not a good reason to avoid using it. There are various possibilities to check the hardware capabilities at runtime: function multiversioning and hardware dispatch at the individual function level (e.g. GCC's FunctionMultiVersioning), at the library/plugin level, *etc*. **[Andrea Bocci, CERN/CMS]**

- This is also matter of brokering or runtime detection which we don't have at the moment maybe worth talking about it tomorrow as we will have the same problem with GPUs spectrum **[Alessandra Forti, Manchester/ATLAS]**
- → *comment postponed to tomorrow's WMS discussion*

Follow-up to the previous comment: this seems like a situation where tools like Spack could play an important role. One could imagine doing some kind of "lowest common denominator" build, and then one or more additional builds that would incorporate the more modern instruction sets. Then the job could choose the appropriate build for whatever worker node it lands on. **[??]**

- → *comment postponed to tomorrow's WMS discussion*

What kind of distributed computing resources are needed/wished for, for a wider use of TF from e.g. analysers? **[Caterina Doglioni, Lund University+ATLAS]**

- → *comment answered in talk & postponed to tomorrow's WMS discussion*

Not only matter of power and memory but certain GPU features like vGPU are specific only to certain brands. Is it something we need to look into from the WMS point of view?

Has NVidia actually sued anyone for using a "gamer" card in a data center / server? I wonder how many teeth there are in their regulations. [**Charles Leggett, LBNL**]

- If you buy only the cards and separately the servers and you install them yourself they don't know but you also do not get the support (i.e. if your card breaks there is no replacement) and you have to be sure the machines you buy are compatible. Rack mounted machines are harder to match. So in other words you can do it, but not for large orders. **[Alessandra Forti, Manchester/ATLAS]**

Wednesday

Benchmarking and Accounting Metrics (~90-110)

Talk: Hep Benchmarks for CPUs and beyond (speaker: Domenico Giordano)

Notes:
Spec17/Spec06 and HEP jobs use different performance counters there is almost not overlap. That's why the HEP Benchmark work started to try to create a benchmark based on our (LHC experiments) applications. Based on standalone containers which encapsulate only what is needed (can run both in docker and singularity). Alice hasn't been included yet because simulation software slightly older. Images are few GBs. HEPscore calculation is similar to HS06 the geometric mean of the WLs speed factors. Main advantage is that it is not proprietary and results can be submitted to sites and collected. WG started to work on CPU+GPU. Problems at HPC is that the container runtimes are incompatible with this workflows in particular singularity version is ancient and doesn't support nested containers, while docker is still a nono until it becomes rootless.

Alessandra: version 3 of singularity works for nested container but HPC have earlier
Domenico: Yes, that is true but how can we request the change
Maarten: HPC centres largely do their own thing (so version is "anyone's guess")
Doug: need to go through active members to contact the HPCs

Talk: Assigning Value to Heterogenous Pledges (speaker: Markus Schulz)

Notes: Heterogeneous means anything but x86 and this is mostly to integrate these resources in the pledges because that is where the big investments will be. HEPscore will integrate accelerators naturally when the code to run on them will become available in the experiments' workflows. Also number of resources is limited compared to the grid sites. An important outcome is to understand if certain resources are worth to use (metric: Realised Potential) because they have a very poor throughput and they might require a lot of effort to use. Usability must be taken into account both in terms of accessibility and software redesign. Far away from accepting them as pledges. There are hidden assumptions in the definition of a pledged resource (grid site) that are not true for HPCs (or commercial cloud services).

Questions / Discussion

Assigning Value Talk: "Usable capacity" seems like it gives a score that is very dependent on the workload (e.g. can it actually use all the resources it is given). This also means that a machine measured one week might get a very different score the next week as a new workload that makes more efficient use of a GPU comes online. From the experiment's POV, this makes a lot of sense, but from a funding agency POV this might not make a lot of sense. [Gordon Watts, UWashington/Seattle - apologies, I'll not be at this talk in person due to a conflict I can't avoid]

Markus: These are not general purpose resources, if the experiment doesn't have a workflow that can exploit them, it means the experiment loses the capacity. Porting common

code (event generators and detector simulation) would make them much more useful to all experiments.

Liz: you would provide a number and a weighted number? On what is based the weighted number?
Domenico: The idea is that the measure is based on the throughput. If we need a single various we can use HEPscore which is an average of the different workflows. In HS06 we take the unweighted average of the scores. In HEPscore it's WLCG deciding which application should have a higher weight.
Liz: worried about the efficiency of the applications if wallclock is used.
Markus: sites use this benchmarks for procurements. Even if one of the experiment is inefficiency the number of events has to be maximized.
Liz: Site has to provide for all the customers, but there is still a worry about wrong incentives.
Markus: we need to expose the inefficiencies so the funding agencies are aware on HPC a
Graeme (on chat): There are other mechanisms for oversight of the efficiency of the experiments' codes and they should not be part of a benchmarking process.

Helge: we want 1 number that gives the processing value. I don't see that yet, but we are going in the right direction.
Markus: for procurement we can measure just the throughput but for pledges we need to evaluate also other factors (hidden assumptions)
Domenico: the way we are going to run is not going to consider only the "power" of the CPU but also the efficiency of the site
Paolo: WLCG worry about the usability but shouldn't take decisions for the experiments. Experiments can decide different resources (CMS only CPU and ATLAS only FPGA for example)
Markus: we have a different record of different workflows per experiments and for example ATLAS runs better at ATLAS sites. This is already visible.
Maarten: we should also profit from the fact that we have measurements consistent with how user communities are going to use the resources. Network problems should be separated, (network efficiency is part of the site efficiency Domenico was talking about before)
Markus: doesn't agree with the assessment. The only important thing is the amount of work a resource can do, so the system balance has to be measured.
Michel: How do you assess a system changing as fast as the GPU landscape?
Markus: these things don't move that fast. We can re-evaluate once a year.
Michel: drivers do evolve quite fast.
Domenico: we are agile enough to adjust to the changes.
Markus: the limiting factor is if what new version do is what we expect compare to compiler
Alessandra: drivers are site dependent
Doug: drivers don't change as often at HPC we have examples of usability already with some sites working for ATLAS but not CMS and you have to include human resources as a cost.
Domenico: you agree with us, we said exactly that.
Markus: it is an economic argument. You need a benchmark to "evaluate the potential"

Helge: an important difference between the past and the current discussion about HPC. The

LHC experiments were a major part of the community and co-evolved. Sites were incentivised if they weren't used much. With HPC sites we don't have this.
Markus: agree

A-1: The usable capacity shall be evaluated on the basis of a set of pre-defined workloads (on which the community will agree), that will be part of the benchmark suite. This set of benchmarks will be representative of the average job mix that will land on these resources as production jobs. Even if there is variation of the "usability" on a weekly basis, because different production workloads are submitted, the average usability shall match the benchmark average job mix [Domenico Giordano, CERN IT]
A-2: With improved adaptation of experiments' workloads to a machine with very different architecture the value of the pledge will increase. This is actually in the interest of the resource owner. If we use the current workloads that are in HEP-score on one of the next generation Exascale machines the throughput figures will be very low to zero, because there aren't many workloads that can utilise the accelerators efficiently.
The "week" as a scale for changes is very optimistic, the time between starting a port of a workload to GPUs to it's introduction in production is on the scale of many months. This means that this re-evaluation will happen at most on a yearly basis. [Markus Schulz, CERN IT]

(semi-related to the talk) Are there (composite?) metrics that can also account for the power-efficiency and climate sustainability of a given machine? This is not an easy point to solve / disentangle from the software and from the place where the machine is placed (inefficient software can make efficient machines wasteful / electricity is produced in different ways in different places / lifetime of the machine may come in as well....) but it goes in the direction of more climate-friendly trigger and computing farms. This has probably been discussed at length before, I'm new to the topic and I was curious to see what the status is. [Caterina Doglioni - Lund/ATLAS]
→ *to be left for dedicated follow-up meeting Michel: it was my plan!*

When will countries be able to pledge non-dedicated resources (the current x86 ones) using the process described on Markus appendix 1 slides? {Paolo Calafiura, LBL]

Markus: good point we should actually look into this and run on these resources
Simone: valuable to sketch how pledging and procuring with these benchmark. We have the tool, but someone should give a dry run on an HPC site.
Paolo: US ATLAS would be happy to give it a go
Simone: a concrete example would simplify the discussion

Already addressed above. I would think that the usability of a platform is a concern of the users (experiments), not WLCG. If ATLAS decides to run analysis on FPGA HPC systems and CMS simulation on GPU HPC systems, they port their codes and they provide the new benchmarking described in your slides, why is WLCG concerned with user costs? [Paolo Calafiura, LBL]

We are isolated from GPU hardware by more layers than exist for CPUs, such as the various compilers, intermediate representations, drivers, etc. These are changing pretty rapidly. How often will benchmarking for heterogeneous systems be re-run? Are the benchmarks going to be run for all the different versions of these layers? [Charles Leggett, LBL]

Validation

Talk: Physics validation procedures (Speaker: Ines Ochoa)

Notes How validation is done in practice (in ATLAS). The goal is to check the effects of software changes on the physics output. Different levels of validation on progressively larger sets of events CI O(10s) -> physics validation O(100k). Physics validation has in itself several steps and also has a framework to compare histograms against results from previous verified software. Sources of non reproducible numbers are listed (for example random seeds, maths libraries… this is to discriminate when different output is expected and when not. Analysis group participate to the final steps of validation. Architecture and environment (compiler, os and hardware) are expected to give differences. More complicated environments like MP have a technical validation too. Is it possible to automate according to certain criteria? Only for certain changes, but a level of manual intervention is expected for major changes.

Michel: have there been validation effort to evaluate heterogeneous resources in ATLAS and have you seen specific challenges?
Ines: I'm not aware of any large scale validation to study these effects
Michele: when you expect differences in terms of intel and AMD do you mean different numbers or efficiency?
Attila: in terms of results
Michel: is this something we should also expect on GPUs?
Attila: very little experience but from what we have seen in the past on GPUs it's much worst because it depends who is running on the GPUs
Maarten: this new phase space makes the problem much larger.
Attila: yes, the hope is that if we add few more dimensions to the matrix we will be able to handle it.

Talk: Patatrak Heterogeneous Validation(Speaker: Adriano Di Florio)

Notes Patatrak is a software R&D incubator, targeting at first the CMS HLT. Pixel based track reconstruction done on the HLT, minimise data transfers. Patatrak does the validation. It runs on GPU and CPU. Validation is done in 3 stages and compared to reference results. It is fully automatised. Workflow starts in github with a PR. Final plots have different values for Legacy, Reference, Development and new PR. The effects of each PR are traced and recorded and then automatically uploaded. Calorimeter local reconstruction validation has a similar workflow but it is not as structured yet or automatised but it is evolving.

Michel: how long does it take to produce this PR validation and do you need anyone to review the code.
Adriano: it takes few hours, it is not triggered automatically. Someone still has to do the PR.
Maarten: you don't do that on every single PR you do a merge of at least few single PRs.
Adraino: yes
Maarten: And like in the previous presentation some differences are perfectly ok
Adriano: yes

Physics Validation Procedures

For generic physics performance, could an analysis that had been previously published and archived with a tool like RECAST play a role in verifying a new version of reconstruction? I can see downsides: you only archive an analysis after it has finished: so late, you'd have to check just the MC (the dataset reprocessing would likely be too expensive), if there are data level API changes, calibration changes, it wouldn't work. But it might also become an increasing source of (automated) physics result that could be used when applicab**le. - Gordon Watts UW/Seattle**.

Adriano: no for us.
Ines: we didn't consider this explicitly. If there is a single workflow that can be reproduced it would be useful.

How much infrastructure is required for the mechanical comparison of distributions (getting chi^2, showing the histograms)? Are there developments that could be envisaged to make this better? Would a common infrastructure help here? (Even beyond the experiments I think that would be useful for event generators and Geant4). [**Graeme Stewart, CERN+ATLAS**]

Adriano: not sure a common would help. The idea is to be as production like as possible and this can change a lot.
Ines: first question yes, we have an infrastructure but no we wouldn't be able to share.
Maarten: let's reverse the question: are some of these technologies exportable? Can part of these be reused?
Adriano: the machinery could be adapted for other experiments. Everything is public on github.
Maarten: it is a start. We can share ideas, but if libraries can be re-utilised instead of starting from scratch and incorporated in their frameworks
Graeme: the question was triggered talking to the generators developers who don't have an infrastructure of their own.

- Putting these resources in scale of % efficiency gains on the grid, there are lots of CPU hours available to spend and still coming out ahead. Can we minimize the number of expert hours needed for validation so we can validate tens (or hundreds) of binary versions of a release? This is probably needed work anyway for the heterogenous world, where each GPU model and driver version is a different target with different numerical properties. [**Mattias Wadenstein, NDGF**]

Graeme: for the numerical differences expected we need to put some brain power at the beginning to find a method, but eventually it should be possible to integrate in the automated process
Maarten: also this process is going to be driven by an economic argument as Markus said earlier.

Detector Simulation could be made (potentially much) more robust towards ordering and many arithmetic changes, but at a cost in development time and with non-zero CPU overhead. Method is to add an RNG state to a track - ensuring that a change in one interaction or boundary crossing only affects 'downstream' interactions, and not ones in a different shower/tree branch. A prototype of this was undertaken in a proof of concept prototype as a GSoC project in 2017 using Geant4 as a testbed, with intended application in the GeantV Vector Sim R&D. Non-trivial changes in Geant4 would be needed. And it would not be expected to to become the recommended way to run production, but could be created for the purpose of making validations more robust. It was explored also in GeantV as a way to have the same results between runs in which scalar and vector implementations were used for the same events/tracks. The approach was chosen to be portable to GPUs as well - although different constraints exist from RNG usability on GPUs, adaptability of some sampling algorithms to vectors/GPUs, ... [**John Apostolakis, CERN**]

Michel: since this is a non trivial change is it worth it?
Ines: I wasn't aware this was possible, actually this would be really interesting.
John: a simpler/cheaper version of what I suggested is possible and (if I understand correctly) is being evaluated in the ATLAS simulation 'user' code.

Patatrack Heterogeneous Validation

Very impressed with the GitHub PR integration for validation. Two technical questions: how much time/person power did that take? And, when a PR is updated/submitted, how long does it take validation to produce results. And my second second question: do you have your own runners that execute the code (I assume you are not using github actions). - **Gordon Watts UW/Seattle**

I see you also validate the runtime performance of the code at the same time as validating the output, just to check that you don't get a regression in the performance. Are the CI resources always suitable for that? (Shared machines or a variety of hardware) [**Graeme Stewart, CERN+ATLAS**]

Graeme: different aspects of the same problem that can be checked at the same time on occasion