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

Outcomes and Summary

This document is a summary of the outcomes and follow-up items identified at the HSF-WLCG workshop in May 2020. It was prepared referring to the workshop notebook and to the recordings posted to Indico.

## Outcomes

### Code portability

HSF can act as a forum for the presentation of results, to understand and influence what will become the standard (e.g. highlighting pros/cons for our own use cases in a discussion with developers and engineers, with help from HEP-CCE). (This is complementary to training in parallelisation techniques for algorithms.)

This is an extremely active area and we expect that there will be a lot of activity: e.g. openlab have a close relationship to key industry players; UK ExCALIBER project speaking to Intel and to CodePlay. Regular feedback should be encouraged.

Many of the HSF code testbeds being used for benchmarking in HEP-CCE have a connection with working groups, e.g. Trigger and Reconstruction.

Training for specific software able to select different targets at compile time e.g. Kokkos, Alpaka (being organised), SYCL. There are training opportunities that we can take advantage of (see also Day 2, frameworks), where the training WG is working on ways to make training *sustainable*.

### Heterogeneous architectures and detector simulation

Potential follow ups on specific topics in Simulation WG meeting(s) include evaluation of portability frameworks (linked to PPS work above).

For detector simulation R&D a specific investigation of ray tracing, and how it can be used effectively on different devices; compilation of (modern) physics models with CUDA; how to manage producing new particles and following them, including secondaries on accelerators.

These topics have started to be looked at since the workshop.

Optimal computation offload in hybrid CPU/GPU environments for simulation will also need to be integrated into the experiments' frameworks (link to Frameworks WG).

## Tensorflow as a compute engine

Establish communication between different TF-based packages (TFA/AmpliTF, zfit, ComPWA/tensorwaves, VegasFlow for Monte-Carlo techniques, internal BESIII/LHCb library by UCAS Beijing). This is to facilitate finding common solutions and ironing out issues / migration to TF2. Best placed under the Scikit-HEP umbrella / analysis or PyHEP WGs. There may be a use to develop/collect/document portable math libraries useful for processing lists of events.

## Experiment Frameworks

Frameworks area has a lot of R&D going on (async offloading, event batching, interface exposed to developers) all topics for the Frameworks WG.

Allied issues, particularly for the reconstruction WG (algorithms). There will be implications for generators and simulation as R&D develops in this area. Training continues to be an issue, with progress here (e.g. upcoming Alpaka workshop).

## Workload Management

Interactions between WMS and Framework, is a cross-cutting topic with more complex information needing to be transmitted to help broker jobs. Further discussions needed, but in which forum? Each experiment has different combinations of software. Interaction with resource providers to help standardise resource descriptions (GDB) and methods of probing.

Code bases being flexible to different devices will help a lot (back to HSF WGs and generic coding approaches - know what targets you can use).

Generalising the deployment of containers as the generic way of packaging workloads (WLCG GDB) helps sites and experiments. Containers are also becoming more interesting as a way to manage sites, using Kubernetes.

## Validation

Assess technical needs within experiments (results could be discussed in an HSF Forum meeting).

No common projects unless there's some up-front interest/commitment from experiments and projects - not clear there is sufficient interest in this.

Assessing numerical stability is a topic for both algorithmic design and technical improvements (some work on this with the verrou tool, HSF Tools group).

## Benchmarking and Accounting

How to deal between these somewhat different approaches between benchmarking for procurements and pledges based on the usable capacity? This is something new: until now pledges were based on the benchmark execution on a given resource.

Usable capacity: test the proposed approach for pledging usable capacity on a guinea-pig HPC site using the tools we have currently and learn from it. How to deal with the possibly evolving “usable capacity” for a given resource? It is particularly true for machines with GPUs. The changes brought in the SW layers by new versions of drivers, CUDA and other portability layers, can increase by an order of magnitude the usable capacity in some cases…

Future benchmark: work needed to agree on the one number (score) that will reflect the potential capacity of a resource. Do we have one and only one number or do different experiments use a different mix? With what consequences on sites supporting multiple experiments? Which rules for evolving the workloads that are part of the benchmark?

Is it possible to have a (composite?) metric that can also account for power efficiency and climate sustainability in addition to  computing power?

Questions to be followed up in HEPiX Benchmark Group and GDB (technical), then WLCG MB for approval.

## Workshop Summary

### Monday

#### Code Portability

Ultimate goal: make (reconstruction, but also other kinds of) code run without too much customization on current and near-future heterogeneous HPCs (CPU/accelerators, including GPU and FPGA). This is for HL-LHC timescales.

Projects like HEP-CCE (HEP Center for Computational Excellence) can aid HEP-specific benchmarking using current software, and will produce a document with different kinds of metric.

There are portability trade-offs to keep in mind (code complexity vs performance, portability vs efficient code). Generally simple and more portable code is privileged. Containerization is a possibility but it depends on how much data needs to be shared.

Liaising with non-HEP developers is important, and having a common view could help influencing the standards. We should not lock ourselves into a single technology/software for the time being.

We should not ignore the fact that HPCs are becoming more and more prominent in research environments and things may move on without HEP if we aren't able to stay on track (also for career development purposes).

#### Heterogeneous architectures and detector simulation

Simulation needs for HL-LHC are substantial.

Many lessons learned from GeantV project (e.g. vectorization of individual components may not be enough and bring complexity, but re-writing and modernizing large CPU code can bring performance improvements). Maintainability and hardware portability are important, since hardware and software are evolving.

Geant 4 R&D TF focused on topics such as improvement of current code (including tracking R&D with ray tracing), integration of fast simulation, accelerator use for particle transport.

#### Tensorflow as a compute engine

The talk covered the use of Tensorflow as a computational library / engine in case of heavy computations on small datasets (e.g. amplitude analyses of 10-100 millions of events in flavour-physics with more after upgrades with hundreds of thousands of fits). Led to the creation of a dedicated TF analysis package as a framework for ML fits + libraries, which can be used by higher level fitters (eg zfit). This shows that TF can be used for uses beyond machine learning.

## Tuesday

### Application Frameworks

Goal is the optimal use of heterogeneous resources. This is easier on owned resources (HLT) as opposed to HPCs or other sites. A number of approaches: separate process spaces (ALICE-FAIR approach in O2 - message-passing; code separation, dynamically balance CPU resource use at process level); accelerator only approach (ideal for R&D projects; optimal performance for the target devices; can leave CPUs idle when they could do useful work, may be balanced by other tasks); hybrid approach with asynchronous execution (most complex for framework, but maybe the biggest prize, smart underlying schedulers (TBB, HPX) help maximise CPU usage). CMSSW is a good example of the latter and can switch between CPU/GPU version(s) according to resource availability. In all cases, there are hurdles for the experiment developers and some steep learning curves (cf. portable parallelisation strategies talk)

### Workload Management

Challenge here is to seamlessly incorporate heterogeneous resources. Outside of HLT farms this can be really heterogeneous - many different GPU and CPU combinations. Total heterogeneous general (non-HPC) resources are small at the moment but will grow. Workloads are CPU only (today the majority), jobs that need a GPU and jobs that can take advantage of a GPU, but can run CPU only (HEP might require a lot of these jobs for high-efficiency use of our global resource pool). Issues include probing/discovering resources correctly (negotiate with the LRMS). Tag resources properly for matchmaking, which is more complex than current CPU cases, but CPUs will also become more heterogeneous with software starting to use more specific instruction sets. Currently ATLAS and CMS users can submit payloads that require GPUs via PanDA and CMS Connect. Use of containers for the software stack is ubiquitous (with more and more sites also using k8s to manage their resources). At the moment we have no large scale grid production workflows using GPUs, with an open question as to how much machine learning HEP will use.

## Wednesday

### Benchmarking

A new CPU benchmarking suite is available and ready to replace HS06 for procurements and pledges

- Based on different categories of workloads from all LHC experiments
- Able to target multiple architectures

This new benchmarking suite is a first and important step to help characterize resources during procurements but is not enough to assign a value to pledges. This value must reflect the “usable capacity”, not only the potential capacity, and take into account more than the CPU. For computing the value, we need to take into account only the workloads appropriate to the non-general purpose resources. The reasoning is that an experiment has a mix of workflow to run and will use the most appropriate resources in its pledges to run each workflow, maximising the usable capacity.

## Validation

From ATLAS, we learned that physics validation is a well-known process and most of the issues are identified. Only a small part of the work can be and is automatized. A specific source of complication with validation is the non reproducible numbers, in particular those from random seeds in simulation. Maybe some improvements in this area could be discussed (in particular for Geant4, it may be technically feasible to achieve a better reproducibility).

CMS implemented a physics validation step implemented in the development lifecycle of their Patatrack framework (R&D framework for high granularity reconstruction). Patratrack supports both CPU and GPU and the validation is run in both environments. This validation runs as part of the GitLab CI and final plots showing the effects of a PR are attached to it, allowing further tracing. The physics validation is rather expensive (a few hours) and thus is not run on every PR: someone has to decide to run it or not. In addition to physics output, the runtime performance is also checked to avoid any significant regression. An experiment-agnostic infrastructure of this kind doesn’t seem feasible.