FERMILAB-PUB-20-467-SCD

HSF-DOC-2020-01
10.5281/zenodo.4009114
31 August, 2020

HL-LHC Computing Review: Common Tools and Community Software

HEP Software Foundation*: **Aarrestad, Thea[3]; Amoroso, Simone[5]; Atkinson, Markus Julian[31]; Bendavid, Joshua[3]; Boccali, Tommaso[14]; Bocci, Andrea[3]; Buckley, Andy[34]; Cacciari, Matteo[21]; Calafiura, Paolo[18]; Canal, Philippe[7,a]; Carminati, Federico[3]; Childers, Taylor[1]; Ciulli, Vitaliano[12]; Corti, Gloria[3,a]; Costanzo, Davide[47]; Dezoort, Justin Gage[43]; Doglioni, Caterina[39,a]; Duarte, Javier Mauricio[30]; Dziurda, Agnieszka[8,a]; Elmer, Peter[43]; Elsing, Markus[3]; Elvira, V. Daniel[7]; Eulisse, Giulio[3]; Fernandez Menendez, Javier[41]; Fitzpatrick, Conor[40]; Frederix, Rikkert[39]; Frixione, Stefano[13]; Genser, Krzysztof L[7]; Gheata, Andrei[3]; Giuli, Francesco[15]; Gligorov, Vladimir V.[21]; Grasland, Hadrien Benjamin[10]; Gray, Heather[18,26]; Gray, Lindsey[7]; Grohsjean, Alexander[5]; Gütschow, Christian[29]; Hageboeck, Stephan[3]; Harris, Philip Coleman[22]; Hegner, Benedikt[3]; Heinrich, Lukas[3]; Holzman, Burt[7]; Hopkins, Walter[1]; Hsu, Shih-Chieh[46]; Höche, Stefan[7]; Ilten, Philip James[27]; Ivantchenko, Vladimir[24]; Jones, Chris[7]; Jouvin, Michel[10]; Khoo, Teng Jian[36,35,a]; Kisel, Ivan[6]; Knoepfel, Kyle[7]; Konstantinov, Dmitri[17]; Krasznahorkay, Attila[3]; Krauss, Frank[33]; Krikler, Benjamin Edward[28]; Lange, David[43]; Laycock, Paul[2,a]; Li, Qiang[42]; Lieret, Kilian[20]; Liu, Miaoyuan[45]; Loncar, Vladimir[3,16]; Lönnblad, Leif[39]; Maltoni, Fabio[11,38]; Mangano, Michelangelo[3]; Marshall, Zachary Louis[18]; Mato, Pere[3]; Mattelaer, Olivier[38]; McFayden, Joshua Angus[18,a]; Meehan, Samuel[3]; Mete, Alaettin Serhan[1]; Morgan, Ben[49]; Mrenna, Stephen[7]; Muralidharan, Servesh[3,1]; Nachman, Ben[26]; Neubauer, Mark S.[31]; Neumann, Tobias[29,9]; Ngadiuba, Jennifer[4]; Ojalvo, Isobel[43]; Pedro, Kevin[7]; Perini, Maurizio[3]; Piparo, Danilo[3]; Pivarski, Jim[43]; Plätzer, Simon[48]; Pokorski, Witold[3,a]; Pol, Adrian Alan[3]; Prestel, Stefan[39]; Ribon, Alberto[3]; Ritter, Martin[20]; Rizzi, Andrea[14,a]; Rodrigues, Eduardo[37]; Roiser, Stefan[3]; Schulz, Holger[32]; Schulz, Markus[3]; Schönherr, Marek[33]; Sexton-Kennedy, Elizabeth[7]; Siegert, Frank[25]; Siódmok, Andrzej[8]; Stewart, Graeme A[3,a]; Sudhir, Malik[44]; Summers, Sioni Paris[3];**

*hsf-editorial-secretariat@googlegroups.com

This document was prepared by HEP Software Foundation collaboration using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE-AC02-07CH11359.

Thais, Savannah Jennifer[43]; Tran, Nhan Viet[7]; Valassi, Andrea[3,a]; Verderi, Marc[19]; Vom Bruch, Dorothea[21]; Watts, Gordon T.[46]; Wenaus, Torre[2]; Yazgan, Efe[23,a]

[1] High Energy Physics Division, Argonne National Laboratory, Argonne, IL, USA

[2] Physics Department, Brookhaven National Laboratory, Upton, NY, USA

[3] CERN, Geneva, Switzerland

[4] California Institute of Technology, Pasadena, California, CA, USA

[5] Deutsches Elektronen-Synchrotron, Hamburg, Germany

[6] Frankfurt Institute for Advanced Studies, Johann Wolfgang Goethe-Universität Frankfurt, Frankfurt,Germany

[7] Fermi National Accelerator Laboratory, Batavia, IL, USA

[8] The Henryk Niewodniczański Institute of Nuclear Physics, Polish Academy of Sciences, ul. Radzikowskiego 152, 31-342 Kraków, Poland

[9] Illinois Institute of Technology, Chicago, USA

[10] IJCLab, CNRS, Université Paris-Saclay and Université de Paris, Orsay, France

[11] INFN Sezione di Bologna, Università di Bologna, Bologna, Italy

[12] INFN Sezione di Firenzea, Università di Firenze, Firenze, Italy

[13] INFN Sezione di Genova, Genova, Italy

[14] INFN Sezione di Pisa, Università di Pisa, Scuola Normale Superiore di Pisa, Pisa, Italy

[15] INFN Sezione di Roma Tor Vergata, Roma, Italy

[16] Institute of Physics Belgrade, Pregrevica 118, Belgrade, Serbia

[17] Institute for High Energy Physics of the National Research Centre Kurchatov Institute, Protvino; Russia

[18] Lawrence Berkeley National Laboratory and University of California, Berkeley, CA, USA

[19] Laboratoire Leprince-Ringuet, École Polytechnique, CNRS/IN2P3, Université Paris-Saclay, Palaiseau, France

[20] Fakultät für Physik, Ludwig-Maximilians-Universität München, München, Germany

[21] Laboratoire de Physique Nucléaire et de Hautes Energies (LPNHE), Sorbonne Université, Université Paris Diderot, CNRS/IN2P3, Paris, France

[22] Massachusetts Institute of Technology, Cambridge, MA, USA

[23] National Taiwan University (NTU), Taipei, Taiwan

[24] P.N. Lebedev Physical Institute of the Russian Academy of Sciences, Moscow, Russia

[25] Technische Universität Dresden, Dresden, Germany

[26] Physics Division, Lawrence Berkeley National Laboratory and University of California, Berkeley CA, USA

[27] University of Birmingham, Birmingham, United Kingdom

[28] H.H. Wills Physics Laboratory, University of Bristol, Bristol, United Kingdom

[29] Department of Physics and Astronomy, University College London, London, United Kingdom

[30] University of California, San Diego, La Jolla, CA, USA

[31] University of Illinois Urbana-Champaign, Champaign, Illinois, IL, USA

[32] University of Cincinnati, Cincinnati, OH, USA

[33] IPPP, Durham University, Durham, United Kingdom

[34] SUPA - School of Physics and Astronomy, University of Glasgow, Glasgow, United Kingdom

[35] Institut für Physik, Humboldt Universität zu Berlin, Berlin, Germany
[36] Institut für Astro- und Teilchenphysik, Leopold-Franzens-Universität, Innsbruck, Austria
[37] Oliver Lodge Laboratory, University of Liverpool, Liverpool, United Kingdom
[38] Université Catholique de Louvain, Belgium
[39] Fysiska institutionen, Lunds Universitet, Lund, Sweden
[40] School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom
[41] Universidad de Oviedo, Instituto Universitario de Ciencias y Tecnologías Espaciales de Asturias (ICTEA), Oviedo, Spain
[42] Peking University, Beijing, China
[43] Princeton University, Princeton, NJ, USA
[44] University of Puerto Rico, Mayaguez, USA
[45] Purdue University, West Lafayette, USA
[46] University of Washington, Seattle, WA, USA
[47] Department of Physics and Astronomy, University of Sheffield, Sheffield, United Kingdom
[48] University of Vienna, Austria
[49] Department of Physics, University of Warwick, Coventry, United Kingdom

[a] Editor

Contents

1 Introduction

Common and community software packages, such as ROOT, Geant4 and event generators have been a key part of the LHC's success so far and continued development and optimisation will be critical in the future [1–3]. The challenges are driven by an ambitious physics programme, notably the LHC accelerator upgrade to high-luminosity, HL-LHC, and the corresponding detector upgrades of ATLAS and CMS. The General Purpose Detectors describe their specific challenges elsewhere; in this document we address the issues for software that is used in multiple experiments (usually even more widely than ATLAS and CMS) and maintained by teams of developers who are either not linked to a particular experiment or who contribute to common software within the context of their experiment activity. We also give space to general considerations for future software and projects that tackle upcoming challenges, no matter who writes it, which is an area where community convergence on best practice is extremely useful.

ATLAS and CMS will undergo major detector upgrades and will also increase their trigger rates for HL-LHC by about a factor of 10; event complexity will rise, with peak pile-up of 200, far higher than in Run-2. This places an enormous burden on storage and processing resources. Current CMOS microprocessor technology is clock speed limited (due to the failure of Dennard scaling) and, while processors per-integrated circuit still keeps rising, Moore's Law is expected to stall during the 2020s. More importantly, the effective runtime related improvements in computing from CPU servers at sites is likely to be only about 10% per year, making the shortfall in processing resources more severe. As manufacturers struggle to deliver ever-more effective computing through CPUs the drive to different architectures intensifies, with GPUs becoming commonplace and increasing interest in even more specialised architectures, such as TPUs, IPUs and developments that make FPGA devices more user friendly [4]. These pose huge challenges for our community as the programming models and APIs vary widely here and possible lock-in to a particular vendor's devices is a significant problem for code sustainability and preservation. Huge work has been done already to adapt to this changing landscape, e.g. multi-threading software and GPU codes have been used in production by some experiments for years now [5, 6]. In other areas migration is still ongoing. In yet others it has not even started. Generic heterogeneous programming models have existed for some time, and new ones are arising, but there is, as yet, no clear winner, in-part because the performance obtained can be highly application-dependent. HEP itself will not drive the success of one model or another, so even if the community converged on a preferred model, its long term success would not be assured. The C++ standard itself is likely to lag for many years behind what is required by us in the area of heterogeneous or even distributed computing. Further, experiment frameworks (and with knock-on effects to systems like workload management) will need to adapt to sites that provide heterogeneous resources. How to do this, and make effective use of different heterogeneous resources across different sites, remains far from settled.

For storage systems (see the DOMA and WLCG documents) the pressure on software is to store as much physics information in as few bytes as possible, but also to be able to

read at very high rates to deliver data from modern storage technologies into the processing hardware. This requires critical developments in the storage formats and libraries used in HEP, e.g. developments like RNTuple for ROOT I/O is likely to be of great importance for the community [7]. The likelihood of finding an off-the-shelf compact and efficient storage format for HEP data is remote, so investment in smart software to support our PB sized science data is simply cost effective. Particularly for analysis, which is usually I/O bound, we have an end-to-end problem from storage technology, through the software layers, to processing resources that may well span multiple nodes. Other workflows, which are less dependent on I/O rates will, nevertheless, have to be adapted to using remote data where the I/O layer must optimise data transfers and hide latency, e.g. taking advantage of XRootD's single column streaming ability [8].

In this increasingly complex environment in which to write software, there are important problems where sharing information at the community level is far more efficient. Providing a high level of support in the build environment for developers, sharing knowledge about how to measure, and then improve, performance (especially on multiple different architectures) and sharing best practice for code development can have a large integrated benefit [9]. This requires improved training and the development of a curriculum for all developer levels. In the field, such initiatives are often warmly received, but real support is needed for those who can put this into practice, also ensuring that their work in training contributes to their career development and a long term future in the field [10]. HEP software stacks are already deep and wide and building these consistently and coherently is also an area where knowledge can be shared. Support is needed for multiple architectural targets and ensuring the correct runtime in heterogeneous environments.

This brings up the important question of validation and the need to improve the security of physics results, which is even more complicated on heterogeneous platforms, when exact binary compatibility often cannot be assured. Currently almost every experiment and project has its own infrastructure for this.

Once software is built, it needs to be shipped worldwide so that the production workflows can run. CernVM-FS was a huge leap forward for software distribution and has even been widely adopted outside HEP [11, 12]. However, new challenges arise, with container based payloads, scaling issues and disconnected supercomputer sites. So maintenance and development needs to be undertaken to support and extend this key supporting infrastructure for software.

Finally, over the multi-decade lifetimes of HEP experiments, we need to preserve both the core and analysis software so that results can be confirmed and updated as the field moves on. There are many exciting developments based around CernVM-FS [13, 14], containers and things like analysis description languages [15], but these are not yet at the stage of being settled nor integrated into our day-to-day workflows.

In the rest of this document the main issues associated with the key parts of the software workflow in high-energy physics are presented, focusing on those that dominate current resource consumption: physics event generation, detector simulation, reconstruction and analysis.

2 Physics Event Generators

2.1 Introduction

Physics event generators are essential in HEP. All of the LHC scientific results, both precision measurements or searches for new physics, depend significantly on the comparison of experimental measurements to theoretical predictions computed using generator software.

Using Monte Carlo (MC) techniques, generators allow both the generation of unweighted events for experimental studies of differential distributions and the prediction of total cross sections. The large-scale event generation campaigns of the LHC experiments have significant computational costs, mainly in terms of CPU resources. The limited size of simulated samples is a source of major uncertainty in many analyses and is therefore a limiting factor on the potential physics output of the LHC programme. This situation will get significantly worse for HL-LHC. The fraction of the CPU resources used for event generation today is approximately 5-15%. As is the case for the other big consumers of CPU (detector simulation and reconstruction), speedups in generator software are needed to address the overall resource problem expected at the HL-LHC, compounded because more accurate predictions, requiring more complex calculations will be needed (e.g. beyond NLO or with higher jet multiplicities). Many other issues, both technical and non-technical exist, e.g. funding, training, careers for those working in this area [16, 17].

A HSF Working Group (WG) on generators [18] was set up at the beginning of 2019. The main focus of the WG so far has been on gaining a better understanding of the current situation, and identifying and prioritising the areas where computing costs can be reduced. In particular, the WG has been active in analysing the ATLAS and CMS compute budgets in detail, in profiling MG5_aMC [19], Sherpa [20] and Powheg [21], in discussing the possible sharing of common parton-level samples by ATLAS and CMS, and in reviewing and supporting the efforts for porting generators to modern architectures (e.g., MG5_aMC to GPUs). This last activity is particularly important, as it has become increasingly clear that being able to run compute-intensive WLCG software workloads on GPUs would allow the exploitation of modern GPU-based supercomputers at High Performance Computing (HPC) centres, and generators look like a natural candidate for this, as they are smaller code bases without complex dependencies.

This section gives an overview of the many technical and non-technical challenges in the generator area and of the work that can be done to address them. This is finally condensed into a list of a few high-priority items, for the next 18 months. A more detailed version of this contribution, including a more complete list of references, is uploaded to arXiv and will be submitted for publication [22].

2.2 Collaborative Challenges

2.2.1 Generator Software Landscape

The landscape of generator software is extremely varied. Different generators are used for different processes. Generating a sample also involves choices of precision (e.g. Leading Order, LO, or Next-to-Leading-Order, NLO), hadronisation and Parton Shower (PS) models, underlying event tunes, prescriptions for matching/merging and simulating particle

decays, and other input parameters, chiefly among them the parton density functions, for which different interfaces exist. Various combinations of software libraries are thus possible, often written by different authors and frequently many years old. For a given process the LHC experiments often use different software packages and settings from each other, and a single experiment can generate events using more than one choice. Many different packages and configurations may therefore need to be worked on to get cumulative CPU cost reductions. The large number of external packages also complicates their long-term maintenance and integration in the experiments' software and workflows, sometimes leading to job failures and computing inefficiencies. Other packages are also absolutely critical for the whole generator community and must be maintained, even if their CPU cost is relatively low (LHAPDF, Rivet, HepMC, etc.).

2.2.2 Skills and Profiles

A very diverse combination of skills and profiles are needed for generator software. Theorists (who create fundamental physics models, and design, develop and optimise most generator code), experimentalists working in research (who request different samples) and in computing (who implement, monitor and account execution of workflows on computing resources), software engineers and system performance experts. This is a richness and opportunity, as some technical problems are best addressed by people with specific skills, but it also poses some challenges:

Training challenges. Theorists and experimentalists often lack formal training in software development and optimisation. Software engineers and experimentalists are often not experts in the theoretical physics models implemented in MC codes.

Communication challenges. It is difficult to find a shared terminology and set of concepts to understand one another: notions and practices that are taken for granted in one domain may be obscure for others. As an example, there are many articles about the physics in generators, but software engineers need papers describing the software modules and overall control and data flow.

Career challenges. Those working in the development, optimisation or execution of generator software provide essential contributions to the success of the (HL-)LHC physics programme and it is critical that they get the right recognition and motivation. However, theorists get recognition on published papers, and may not be motivated to work on software optimisations that are not "theoretical" enough to advance their careers. Generator support tasks in the experiments may also not be enough to secure jobs or funding for experimentalists pursuing a career in research.

Mismatch in usage patterns and optimisation focus. The way generators are built and used by their authors is often different from the way in which they are deployed and integrated by the experiments in their software frameworks and computing infrastructure. The goals and metrics of software optimisation work may also differ. Theorists are mainly interested in calculating cross sections and focus on minimising the phase space integration

time for a given statistical precision. The LHC experiments run large scale productions of unweighted event generation, and mainly need to maximise the throughput of events generated per unit time on a given node.

Programming languages. Attracting collaborators with a computer science background to work on generators, especially students, may also be complicated by the fact that critical components of some generator packages are written in Fortran, which is rarely used in industry and less popular among developers than other programming languages. Some of the generators also do not use industry standard version control systems, making it harder to contribute code.

2.3 Technical Challenges

The event generation workflow presents several challenges and opportunities for improvement.

2.3.1 Inefficiency in Unweighted Event Generation

Phase space sampling inefficiency. Efficient sampling is the most critical ingredient for efficient unweighted event generation. Many generic algorithms exist (e.g. VEGAS [23], BASES / SPRING [24], MINT [25], FOAM [26]), as well as others developed specifically for a given generator (e.g. MadEvent [27], itself based on a modified version of VEGAS, in MG5_aMC, or COMIX [28] in Sherpa). In general, the larger the dimensionality of the phase space, the lower the unweighting efficiency that can be achieved: in W+jets, for instance, the Sherpa efficiency is 30% for W+0 jets and 0.08% for W+3jets [29]. This is an area where research is very active, and should be actively encouraged, as significant cost reductions in WLCG compute budgets could be achieved. Improvements in this area start from physics-motivated approaches based on the knowledge of phase space peaks and are complemented by machine learning (ML) algorithmic methods [29–32].

Merging inefficiency. Merging prescriptions (e.g. MLM [33], CKKW-L [34] at LO and FxFx [35], MEPS@NLO [36] at NLO) imply the rejection of some events, to avoid double counting between events produced with n_{jets}+1 matrix elements and with n_{jets} MEs plus parton showers. The resulting inefficiencies can be relatively high, depending on the process, but they are unavoidable in the algorithmic strategy used by the underlying physics modeling [37]. However, a method like shower-kt MLM can reduce the merging inefficiency of MLM [38].

Filtering inefficiency. An additional large source of inefficiency is due to the way the experiments simulate some processes, where they generate large inclusive event samples, which are then filtered on final-state criteria to decide which events are passed on to detector simulation and reconstruction (e.g. CMS simulations of specific Λ_B decays have a 0.01% efficiency and ATLAS B-hadron filtering has ˜10% efficiency for V+jets). This inefficiency could be reduced by developing filtering tools within the generators themselves, designed for compatibility with the requirements of the experiments. Filtering is an area where LHCb has a lot of experience and already obtained significant speedups through various

techniques. The speed of colour reconnection algorithms is a limiting factor for simulating rare hadron decays in LHCb.

Sample sharing and reweighting. In addition to removing inefficiencies, other ways could be explored to make maximal use of the CPU spent for generation by reusing samples for more than one purpose. Sharing parton-level samples between ATLAS and CMS is being discussed for some physics analyses. Event reweighting is already commonly used (e.g. for new physics searches and some systematic uncertainty calculations) and could be explored further, though this may require more theoretical work for samples involving merging or NLO matching [39].

Negative weights. Matching prescriptions, like MC@NLO, are required in (N)NLO calculations to avoid double counting between (N)NLO matrix elements and parton showers, leading to the appearance of events with negative weights. This causes a large inefficiency, as larger event samples must be generated and passed through the experiment simulation, reconstruction and analysis codes, increasing the compute and storage requirements. For a fraction r of negative weight events, the number of events to generate increases by a factor $1/(1-2r)^2$: for instance, with $r = 25\%$ (which may be regarded as a worst-case scenario in top quark pair production [40]), one needs to generate 4 times as many events. Negative weights can instead be almost completely avoided, by design, in another popular matching prescription, POWHEG; however, this is only available for a limited number of processes and describes the relevant physics to a different degree of precision than MC@NLO (see [40] for a more in-depth discussion). Progress in this area can only be achieved with physics knowledge: for instance, a new approach for MC@NLO-type matching with reduced negative weights has recently been proposed [40] and some recent papers show how to lessen the impact of negative weights in a secondary step [41, 42]. It should be noted that negative weights can also happen at LO because of not-positive-definite parton density function sets and interference terms, e.g. in effective field theory calculations.

2.3.2 Accounting and Profiling

While progress has been made to better understand which areas of generator software have the highest computational cost, more detailed accounting of the experiment workloads and profiling of the main generator software packages would help to further refine R&D priorities.

Accounting of CPU budgets for generators in ATLAS/CMS. Thanks to a lot of effort from the generator teams in both experiments, a lot of insight into the settings used to support each experiment's physics programme was gained within the WG, and it is now clear that the fraction of CPU that ATLAS spends for event generation is somewhat higher than that in CMS. More detailed analysis of the different strategies is ongoing. These figures had to be harvested from logs and production system databases, which was particularly difficult for CMS, requiring significant person hours. It is important to establish better mechanisms to collect this information, to allow for an easy comparison between different experiments.

Profiling of the generator software setups used for production. Another area where the WG has been active is the definition and profiling of standard generator setups, reproducing those used in production. Detailed profiling could also be useful to assess the CPU cost of external parton distribution function libraries [43], or the memory requirements of the software (which may motivate a move to multithreading or multiprocessing approaches).

2.3.3 Software Modernisation

More generally, as is the case in other areas of HEP, some R&D on generator software is certainly be needed to modernise it and make it more efficient, or even port it to more modern computing architectures [16, 44]:

Data parallelism, GPUs and vectorisation. The data flow of an MC generator, where the same (matrix element) function is computed repeatedly at many phase space points, lends itself naturally to the data parallel approaches found in CPU vectorised code and in GPU compute kernels. Porting and optimising generators on GPUs is essential to be able to use modern GPU-based HPCs. The work done in this direction in the past on MG5_aMC, that never reached production quality, is now being reinvigorated by the WG, in collaboration with the MG5_aMC team, and represents one of the main R&D priorities of the WG. This work is presently focusing on Nvidia CUDA, but abstraction libraries will also be investigated. GPUs may also be relevant to ML-based sampling algorithms and to the pseudo-random number generation libraries used in all MC generators.

Task parallelism, multithreading and multiprocessing. Generators are generally executed as single threaded software units. In most cases, this is not a problem, as the memory footprint of unweighted event generation is small and usually fits within the 2 GB per core available on WLCG nodes. However, there are cases where memory is higher than 2 GB (e.g. DY+jets using Sherpa); this leads to inefficiencies as some processor cores remain unused, which could be avoided using multithreading approaches. The fact that some generators are not even thread safe may also be a problem, for instance to embed them in multi-threaded event processing frameworks, such as that of CMS. Multi-processing approaches may also be useful to speed up the integration and optimisation step for complex high-dimensional final states. In particular, a lot of work has been done to implement MPI-based multi-processing workflows for generators in recent years. For instance, the scaling of LO-based generation of merged many-jet samples has been successfully tested and benchmarked on HPC architectures using both Alpgen [45] and Sherpa [46]; in the latter case, new event formats based on HDF5, replacing LHEF, have also been instrumental in the success of these tests. MPI integration has also been completed for MG5_aMC [47]. It should also be noted that, even if HPCs offer extremely high-speed inter-node connectivity, it is perfectly ok for WLCG workflows to use these systems as clusters of unrelated nodes.

Generic code optimisations. A speedup of generators may also be achievable by more generic optimisations, not involving concurrency. It should be studied, for instance, if data caching [43] or different compilers and build strategies may lead to any improvements.

2.4 Physics Challenges

In addition to software issues, important physics questions should also be addressed about more accurate theoretical predictions (above all NNLO QCD calculations, but also electroweak corrections) and their potential impact on the computational cost of event generators at HL-LHC. Some specific NNLO calculations are already available and used today by the LHC experiments in their data analysis. With a view to HL-LHC, however, some open questions remain to be answered, in particular:

NNLO: status of theoretical physics research. The first question is, for which processes NNLO precision will be available at the time of the HL-LHC? For example, when would NNLO be expected for $2 \to 3$ or even higher multiplicity final states? And for lower multiplicity final states, where differential NNLO predictions exist but the generation of unweighted NNLO+PS events is not yet possible. It is important to clarify the theoretical and more practical challenges in these calculations, and the corresponding computational strategies and impact on CPU time needs.

NNLO: experimental requirements for data analysis at HL-LHC. The second question is, for which final states unweighted event generation with NNLO precision would actually be required? and how many events would be needed? One should also ask if reweighting LO event samples to NNLO would not be an acceptable cheaper alternative to address the experiment's needs.

Size of unweighted event samples required for experimental analysis at HL-LHC. Another question, unrelated to NNLO, is in which regions of phase space the number of unweighted events must be strictly proportional to the luminosity? For example, in the low p_{T} regions of W boson production it is probably impossible to keep up with the data, due to the huge cross section. Alternative techniques should be investigated, to avoid the generation of huge event samples.

3 Detector Simulation

3.1 Introduction

In HEP, data analysis, theory model evaluation and detector design choices rely on detailed detector simulation. Since the start of the first run the LHC experiments have produced, reconstructed, stored, transferred and analysed hundreds of billions of simulated events. This effort required a major fraction of WLCG's computing resources [48–56]

The increase in delivered luminosity in future LHC runs, in particular with the HL-LHC, will create unique research opportunities by collecting an order of magnitude more data. At the same time, the demand for detector simulation will grow accordingly so that statistical uncertainties are kept as low as possible. However, the expected computing resources will not suffice if current detector simulation is used; the availability of simulated samples of sufficient size would soon become a major limiting factor as far as new physics discoveries, for example through precision measurements, are concerned. Development of

faster simulation, therefore, is of crucial importance and different techniques need to be explored under the assumption that they will provide a sufficient gain in time performance with a negligible or acceptable loss in physics accuracy.

The Geant4 simulation toolkit has been the de facto standard for HEP experiment simulation for physics studies over the last two decades [57–59]. Designed in the late 1990s to cope with the simulation challenges of the LHC era, Geant4 introduced flexible physics modelling, exploiting layers of virtuality. This fostered progress in physics by comparison of competing models on the same energy ranges, and allowed complementary models to be combined to cover large energy ranges. The simulation applications of the LHC experiments make use of this toolkit and, as such, most of the LHC physics program relies on it. It has delivered a physics precision that allows the experiments to perform precision analysis of the collected data and to produce new results pushing the boundaries of our understanding in HEP. At the same time, the code of the toolkit is becoming old, with some parts written almost thirty years ago, making it more and more difficult to run efficiently on modern computing architectures. Any improvement in the critical elements of Geant4 can be leveraged by the applications that use it. Improvements of physics models need to continue to take place to prevent systematic uncertainties from simulation from becoming a dominant factor, however, their exact influence on the precision of physics analysis is not always easy to quantify. On the other hand, the overall CPU performance of the current code will not improve drastically just by relying on modern hardware or better compilers.

Taking all those elements into account, three main development paths have emerged to be undertaken simultaneously and these have started to be pursued. However, effort in any one of them is far from sufficient to exploit their individual potential.

Firstly, it is necessary to continue to modernise and refactor the simulation code by using more optimal, modern programming techniques.

Secondly, different fast simulation techniques, where tracking of individual particles in given detectors is replaced by some sort of parameterisation and precision is traded to achieve higher event throughput.

Finally, the use of compute accelerators (like GPUs or FPGAs) is the third emerging path that needs to be undertaken and that requires intense R&D effort.

We will discuss more in detail these three development axes in the following sections, outlining the developments identified in and since the Community White Paper [3, 60].

3.2 Geant4 R&D

The Geant4 particle transport toolkit contains advanced models for electromagnetic and hadronic physics processes, as well as the geometrical description of the detectors. The toolkit design principles were to allow flexibility in future extensions of its own code, as well as choices for particular applications. It is clear that this flexibility comes at the price of a performance penalty. It was observed that writing more compact and more specialised code can lead in certain places to up to several tens of percent speed-up [61, 62]. The technical improvements to explore here consist of avoiding too many virtual layers and improving the instruction and data locality, leading to better cache use.

Going even further, as far as specialisation is concerned, studies suggest that implementing compact, self-contained transport libraries with a minimal set of physics models (for instance only electromagnetic interaction for selected particles) with predefined scoring (to simulate, for instance, the response of an EM calorimeter) can also bring speed-ups to the complete application [61, 62].

As far as architectural modifications are concerned, the design of the Geant4 track-status machinery prevents the introduction of fine-grained track parallelism. It is still not clear whether this kind of parallelism can bring any substantial speed-up, due to the overheads; however, making the Geant4 kernel and physics processes stateless would certainly provide the necessary starting points for further R&D in that direction.

Most of these investigations have started and it is hoped to see some first results on a year long time-scale. More effort, however, will be required to integrate any eventual changes in the production code in a timely manner.

The GeantV Vector prototype has allowed the assessment of the achievable speedup using a novel, vectorised approach to particle transport. The demonstrator of a full electromagnetic shower in realistic (CMS) geometry has been implemented and the comparison to a similar Geant4 application has been performed. The conclusion of that work showed that the main factors in the speedup seem to include better cache use and tighter code, while the vectorisation impact was much smaller than hoped for. This unimpressive impact is due to a set of factors, including the fraction of the current algorithms that could be vectorised, unresolved challenges for the gather/scatter and tail handling (to keep the number of events in flight within bound) needed for large number of shapes, and of course Amdahl's Law applied to vectorisation where some bottleneck (for example geometry navigation) could not be tackled without a major additional long term effort [61, 62].

At the same time libraries developed in the context of the GeantV R&D, e.g. VecGeom, have been successfully integrated in Geant4 and ROOT benefiting the whole HEP software community [63]. The cost-over-benefit of more vectorisation of the code has not been deemed worth the necessary investment also due to the speculation that a complete rewrite of the physics base could become necessary to fully exploit it. As a result investing in the optimisation and modernisation of the Geant4 code has gained even more relevance.

3.3 Experiment Applications and Optimised Use of Geant4

All LHC experiments have dedicated simulation frameworks making use of the Geant4 toolkit for modelling physics processes in their detectors. Every experiment configures and uses what is provided by Geant4 for their specific needs. Effort has been spent by each experiment since the publication of the Community White Paper to benchmark their simulation code and to maximally exploit the options provided by the Geant4 toolkit to optimise the performance of their applications [64]. All of the various handles provided are being continuously explored and adopted when deemed useful: range and physics cuts have been reviewed and customised by all experiments, shower libraries adopted as baseline in the forward region by ATLAS and CMS, stepping granularity optimised, magnetic field caching investigated, neutron Russian roulette used for dedicated simulations. All of these

can have major impacts on simulation time and sharing knowledge and experience between experiments can be of high benefit even if each experiment has to find its optimal solution.

ATLAS, CMS and LHCb have integrated the Geant4 event-based multi-threading features within their own multi-threaded frameworks, harmonising their different architectural choices with the Geant4 implementation [65–67], and have used, or are planning to use, them in their production environment to gain access to a larger set of distributed resources. ATLAS has been running their multi-process simulation framework on HPCs systems for production in the last years [68, 69] while CMS has pioneered the use of their multi-threaded simulation on available resources including HPCs [70]. Both approaches have allowed the exploitation of additional computing resources by enabling access to lower memory systems.

3.4 Fast Simulations

Fast simulation techniques consist of replacing the tracking of individual particles through the detector (or part thereof), including all the physics processes they would undergo, by a parameterisation, where the detector response is produced directly as a function of the incoming particle type and energy.

An example of such a parameterisation was implemented many years ago in the context of the H1 experiment [71]. This parameterisation, available within the Geant4 toolkit under the name of the GFLASH library, became the starting idea for several custom implementations, specific for other calorimeters. Those were interfaced to experiments' simulation applications through a hook in the Geant4 toolkit. The LHC experiments implemented, for example, dedicated parametrised response libraries for some of their calorimeters. Recently, an effort has been invested in Geant4 to generalise the parameterisation formulae, and to implement automatic procedures of tuning the parameters for specific detector geometries. This kind of 'classical' fast simulation, which describes the shower shape using some complex mathematical functions, with several parameters, will remain an important tool; however, it is also clear that their overall precision for different energy values, especially for highly granular and complex calorimeters will always be relatively low.

Recent developments of deep learning-based techniques have opened up an exciting possibility of replacing those 'classical' parameterisations by trained neural networks that would reproduce the detector response. This approach consists of training generative models, such as Generative Adversarial Networks (GAN), Variational Auto-Encoders (VAE) or Autoregressive Generative Networks on the 'images' of particle showers [72–75]. The energy deposition values in the calorimeter cells are considered as 'pixels' of the images that the network is supposed to reproduce. The first studies and prototypes have shown very promising results, however, the generalisation of the developed models (for different calorimeters, with different particles incident at different angles with different energies), still requires further effort. Detailed physics validation of those tools and understanding their capability to reproduce fluctuations is a prerequisite towards production quality fast simulation libraries. This work is already ongoing, but will require, over the next few years, more effort to be invested, combining physics and Machine Learning expertise. Given that the training of the deep learning algorithms usually requires using sufficiently large MC

samples produced using standard techniques, the development of these novel tools does not necessarily remove the need to speed up Geant4.

It is worth pointing out that different fast simulation techniques have already been extensively used for LHC simulation campaigns. Effort has been spent in the experiments frameworks to combine as much as possible fast and detailed implementations. In particular, ATLAS and CMS are using shower libraries in production for the modelling of calorimeters in the forward region, in combination with the detailed simulation of the rest of the sub-detectors. In LHCb the re-use of part of the underlying events for specific signals of interest has been established in production of samples when appropriate, with particular care paid to ensure keeping under control bias on statistical uncertainties [76]. Applicability of this technique in ATLAS and CMS could be explored for specific cases. Similarly, the re-use of simulated or real events to mimic the background to hard-scatter events is also being exploited, where additional challenges exist concerning storage and I/O due to the handling of the large minimum bias samples needed to model the additional interactions.

All LHC experiments have explored the use of multi-objective regression and generative adversarial networks (GANs) [74, 75, 77], they are now in the process of investigating the use of fast simulations for other types of detectors than calorimeters, e.g. Cerenkov based systems [78] and Time Projection Chambers [79]. While implementations of fast simulations of given sub-detectors is specific to their technology and experimental environment and as such the responsibility of each LHC experiment, an area of potential common investigation is frameworks for fast tuning and validation.

The use of fully parametric response of experimental setups at the level of reconstructed objects (tracks and clusters) for rapid evaluation of physics reach is also exploited by the experiments with generic frameworks for fast simulation of typical collider detectors being used [80, 81]. The advantage of such frameworks is that they allow simultaneous studies for different experiments as well as their potential use by the theory community. Experiment specific fully parameterised simulations providing reconstructed objects compatible with the experiment's analysis frameworks for systematic verifications have also been emerging [78] .

It is reasonable to expect that the full variety of simulation options, from 'smeared reconstructed quantities' to parameterisation for specific detectors to detailed Geant4, will need to be exploited by the experiments for different tasks. Seamless ways of providing them in a transparent and integrated way in the experiments' simulation frameworks should continue to be pursued with Geant4 support.

3.5 Technical Challenges

The hardware landscape has always set directions as far as software evolution is concerned. The recent adaptation of the simulation code to multithreading (MT) turned out to be relatively straightforward, but still took several years to adopt and validate. Geant4 can now run in MT mode efficiently, distributing the events between threads and sharing resources, like the geometry description or physics data, and thus reduce the memory footprint of a simulation while maintaining its throughput performance. The efficient utilisation of Single Instruction Multiple Data (SIMD) architectures, on the other hand turned out to

be more challenging. Not only was a limited part of the code vectorisable but also, as the already mentioned GeantV R&D has shown, overheads related to gathering the data for vector processing presented a significant challenges (tail handling, work balancing across threads, etc.) to efficiently (development time wise and run time wise) profit from it [61, 62].

The most recent revolution in computing hardware is the use of Graphics Processing Units (GPUs) for general purpose computing. Several attempts have been made to port specific simulation code to GPUs. A few of them turned out to be successful, leading to factors of several hundred or a thousand speedup [82–84]; however, they were always limited to a very specific simulation problem, far from what is addressed by a general HEP simulation. This application of GPUs has been very successful in restricted domains like medical physics simulation, neutron transport [85] or optical photon transport [86]. In those applications, the type of particles considered is very limited (often just one type, with no secondary particle production), the set of physics processes is reduced and the geometry is often extremely simple compared to the LHC detectors. A natural extrapolation of those approaches to a general HEP simulation is very difficult to imagine, because the stochasticity of the simulated events would immediately lead to divergences as far as different GPU threads are concerned, not to mention, simply the feasibility of porting the whole simulation to GPU code, for which specific expertise would be needed.

On the other hand, running only very small pieces of the simulation application on GPUs does not seem to be efficient either, as gathering data and transferring it from the host to the device and back again may strongly limit any possible performance gain, similar to what was seen with the GeantV vectorisation R&D. The most plausible approaches seem therefore to lead in the direction of specialised libraries (like those described above in the context of speeding up Geant4 simulation on the CPUs) that would perform the complete simulation of specific sub-detectors (for instance EM calorimeter, for specific incoming particles or Cherenkov-based detectors for the optical processes) on the device. Such libraries could be the right compromise between the complexity of the algorithm that needs to be ported and the overall time that is now saved in by the full simulation on the CPU.

An investigation of these directions is starting now, but it will certainly require considerable effort to be invested before being able to judge the real impact. The first steps that are currently being undertaken consist of implementing, on GPUs, prototypes based on VecGeom geometry and navigation capabilities [87]. If those prototypes turn out to be successful, the next challenge will consist of adding tracking and eventually a limited subset of physics models. While Geant4 is a toolkit capable of addressing different modelling and simulation problems and contains many features and capabilities allowing for user access to detailed information for a large variety of use cases and scientific communities, this GPU development might turn into specialised transport modules, stripped of some features which would be expensive to implement or support efficiently on GPUs.

Another interesting avenue consists of exploring the application of some vendor libraries (like Nvidia Optix). Those libraries, originally developed for ray tracing, have several similarities with general particle transport and, if applied successfully, could lead to major

speed-ups. All these efforts certainly require a major investment and new developers, expert in those technologies, to join the R&D work.

3.6 Other Activities

Common digitisation efforts would be desirable among experiments, with advanced high-performance generic examples, which experiments could use as a basis to develop their own code. Nevertheless the large variety of detector technologies used reduces its applicability. Digitisation is not yet a limiting factor, in terms of CPU requirements, so developments in this area have not been a priority.

Simulated samples are often processed through the reconstruction as real data. As such in some cases they require the emulation of hardware triggers. Hardware triggers are based on very specific custom devices and a general approach does not seem very feasible, even if some parametrisation could be generalised.

3.7 Outlook

It is important to realise that there is a considerable risk that simulation becomes a major limiting factor as far as new physics discoveries are concerned if no serious effort is invested in R&D. Concerted effort of different experts in physics, Machine Learning and GPUs is needed. There are too many unknowns to focus on only one direction, so a lot of different prototyping is needed. Development of Geant4, while working on fast simulation algorithms and doing extensive R&D on leveraging compute accelerators is required. While the R&D activities are taking place, one can not forget that it is still necessary to support the existing code and make sure that sufficient funding and staffing is provided for maintenance and development of physics algorithms, as well as for adapting the code to any updated CPU hardware, operating systems and new compilers.

It is also important to stress once again the need for a continuous integration activity of new runtime performance improvements and solutions into the Geant4 code base and experiments applications to profit from any improvement as quickly as possible. An effort needs to be invested into making it possible to evolve it to meet the HL-LHC needs without compromising the production quality of the Geant4 code used by the experiments. We need to ensure that new developments resulting from the R&D programs can be tested with realistic prototypes and, if successful, then integrated, validated, and deployed in a timely fashion in Geant4 and adopted by the experiments. The strategy adopted in the successful integration in Geant4 of VecGeom libraries developed in the context of the GeantV R&D can provide a working example of how to proceed to provide some incremental improvements to existing applications.

No single solution appears at the moment to provide by itself the processing gains required by HL-LHC, nevertheless if a novel approach emerges that could do so it should be pursued and carefully evaluated in terms of gains against the disruption of a complete re-implementation of how events are simulated.

4 Reconstruction and Software Triggers

Software trigger and event reconstruction techniques in HEP face a number of new challenges in the next decade. Advances in facilities and future experiments bring a dramatic increase in physics reach, as well as increased event complexity and rates.

At the HL-LHC, the central challenge for high-level triggers (e.g. software triggers) and object reconstruction is to maintain excellent efficiency and resolution in the face of high pileup values, especially at low transverse momenta. Detector upgrades, such as increases in channel density, high precision timing and improved detector layouts are essential to overcome these problems. The subsequent increase of event complexity at the HL-LHC also requires the development of software algorithms that can process events with a similar cost per-event to Run-2 and Run-3. At the same time, algorithmic approaches need to effectively take advantage of evolutions in computing technologies, including increased SIMD capabilities, increased core counts in CPUs, and heterogeneous hardware.

This section focuses on the challenges identified in the Community White Paper [6, 16, 88] and the development of solutions since then. It also includes mentions of open source software efforts that have developed within or outside LHC collaborations that could be adapted by the LHC community to improve trigger and reconstruction algorithms.

4.1 Evolution of Triggers and Real-Time Analysis

Trigger systems are evolving to be more capable, both in their ability to select a wider range of events of interest for the physics program of their experiment, and their ability to stream a larger rate of events for further processing. The event rates that will be processed by experiments at the HL-LHC will increase by up to a factor 10 with respect to Run-3, owing to upgraded trigger systems and expanded physics programs. ATLAS and CMS plan to maintain a two-tiered trigger system, where a hardware trigger makes use of coarse event information to reduce the event rate to 10x over the current capability, up to 1 MHz [89, 90]. The high level trigger system, implemented in software, selects up to 100 kHz of events to be saved in full for subsequent analysis*.

Experiments have also been working towards minimising the differences between trigger (online) and offline software for reconstruction and calibration, so that more refined physics objects can be obtained directly within the HLT farm for a more efficient event selection. This is also in-line with enhancing the experiment's capabilities for real-time data analysis of events accepted by the hardware trigger system, driven by use cases where the physics potential improves when analysing more events than can be written out in full with traditional data processing.

Implementations of real-time analysis systems, where raw data is processed into its final form as close as possible to the detector (e.g. in the high-level trigger farm), are in use within several experiments. These approaches remove the detector data that typically makes up the raw data kept for offline reconstruction, and keep only a limited set of analysis objects reconstructed within the high level trigger or keepig only the parts of

*LHCb [91] and ALICE [92] will both stream the full collision rate to real-time or quasi-real-time software trigger systems.

an event associated with the signal candidates, reducing the required disk space. The experiments are focusing on the technological developments that make it possible to do this with acceptable reduction of the analysis sensitivity and with minimal biases. The HSF Reconstruction and Software Trigger group has been encouraging cross-talk between LHC experiments and beyond on real-time analysis, as these kinds of workflows increase the physics output for selected physics cases without adding significant overhead to the current resources.

Active research topics also include compression and custom data formats; toolkits for real-time detector calibration and validation which will enable full offline analysis chains to be ported into real-time; and frameworks which will enable non-expert offline analysts to design and deploy real-time analyses without compromising data taking quality. Use cases for real-time analysis techniques have expanded during the last years of Run 2 and their use appears likely to grow already during Run 3 data taking for all experiments owing to improvements in the HLT software and farms. Further ideas include retaining high-rate, but very reduced data, from selected regions and sub-detectors, even up to the 40 MHz bunch crossing rate [93].

4.2 Challenges and Improvements Foreseen in Reconstruction

Processing and reducing raw data into analysis-level formats, event reconstruction, is a major component of offline computing resource needs, and in light of the previous section it is relevant for online resources as well. This is an essential step towards precision reconstruction, identification and measurement of physics-objects at HL-LHC.

Algorithmic areas of particular importance for HL-LHC experiments are charged-particle trajectory reconstruction (*tracking*), including hardware triggers based on tracking information which may seed later software trigger and reconstruction algorithms; jet reconstruction, including the use of high-granularity calorimetry; and the use of precision timing detectors.

4.2.1 Tracking in High Pile-up Environments

The CPU needed for event reconstruction in Runs 2 and 3 is dominated by charged particle reconstruction (tracking). This is still a focus for HL-LHC triggering and event reconstruction, especially when the need to efficiently reconstruct low transverse momentum particles is considered.

The huge increase in the number of charged particles, and hence the combinatorial load for tracking algorithms, at future colliders will put great strain on compute resources. To minimise the memory footprint, tracking software needs to be thread-safe and to support multi-threaded execution per core. At the same time, the software has to be efficient and accurate to meet the physics requirements.

Since the CWP, experiments have made progress towards improving software tracking efficiency in HL-LHC simulation, e.g. [94]. A number of collaboration and community initiatives have been focusing on tracking, targeting both offline and online, and these are listed below.

The **ACTS** project [95, 96] is an attempt to encapsulate the current ATLAS track reconstruction software into an experiment-independent and framework-independent tracking software, designed to fully exploit modern computing architectures. It builds on the tracking experience already obtained at the LHC, and targets the HL-LHC as well as future hadron colliders. ACTS provides a set of track reconstruction tools designed for parallel architectures, with a particular emphasis on thread-safety and concurrent event reconstruction. ACTS has active collaborators from other HEP experiments, such as FASER, and provides support to Belle-II, EIC and CEPC.

The aim of the **mkFit** [97] project is to speed up Kalman filter (KF) tracking algorithms using highly parallel architectures and to deliver track building (and possibly fitting) software for the HL-LHC. Recent activities focused on delivering a production quality software setup to perform track building in the context of LHC Runs 2 and 3. The implementation relies on single-precision floating point mathematics and is available for multicore CPUs. It achieves significant gains in compute performance from the use of vector instructions, extending to AVX-512, based on Matriplex library (a part of the mkFit project).

Exa.TrkX [98] is a cross-experiment collaboration of data scientists and computational physicists from ATLAS, CMS and DUNE. It develops production-quality deep neural network models, in particular Graph Neural Networks, for charged particle tracking on diverse detectors employing next-generation computing architectures such as HPCs. It is also exploring distributed training and optimisation of Graph Neural Networks (GNN) on HPCs, and the deployment of GNNs with microsecond latencies on Level-1 trigger systems.

4.2.2 Adding Timing Information to Reconstruction

Physics performance in very high pileup environments, such as the HL-LHC or FCC-hh, may also benefit from adding timing information to the reconstruction. This allows the mitigation of the effects of pile-up by exploiting the time-separation of collision products [99, 100]

Experiment communities have been working on timing detector reconstruction and object identification techniques in complex environments. Since the CWP, initial tracking and vertexing algorithms that include timing information have been developed and incorporated into experimental software stacks [101][†].

4.3 Enhanced Data Quality and Software Monitoring

At HL-LHC, the development, automation, and deployment of extended and efficient monitoring tools for software trigger and event reconstruction algorithms will be crucial for the success of the experimental physics programme.

HEP experiments have extensive continuous integration systems, including code regression checks that have enhanced the monitoring procedures for software development in

[†]Tracking algorithms where events overlap in time within time slices are employed by the CBM experiment [102, 103]. The use of a cellular automaton makes the algorithm less dependent on the specific detector geometry, and there may be the possibility of cross-talk with LHC experiments that are investigating 4D reconstruction.

recent years. They also have automated procedures to check trigger rates as well as the performance of the low- and high-level physics objects in the data.

Since the CWP, experiments have started making limited use of machine learning algorithms for anomaly detection [104, 105].

4.3.1 General Reconstruction Software Improvements: Vectorisation

Improving the ability of HEP developed toolkits to use vector units on commodity hardware will bring speedups to applications running on both current and future hardware. The goal for work in this area is to evolve current toolkit and algorithm implementations, and develop best programming techniques to better use the SIMD capabilities of current and future computing architectures. Since the CWP, algorithm development projects have demonstrated success in increasing the use of vector units [97, 106]. In addition, HEP has increasingly benefited from industry developed software, including machine learning toolkits (e.g., TensorFlow), that typically make excellent use of vector units. Challenges remain in this area: for example, full exploitation must also account for many generations of hardware that experiments must exploit on the grid. In addition, to realise full performance gains, a large fraction of the application must be improved to use SIMD processing capabilities (partly due to turbo-boost capabilities of processors).

4.3.2 Use of Machine Learning

It may be desirable, or even necessary, to deploy new algorithms that include advanced machine learning techniques to manage the increase in event complexity without increasing per-event reconstruction resource needs.

Work is already ongoing in the collaborations to evolve or rewrite existing toolkits and algorithms focused on their physics and technical performance at high event complexity (e.g. high pileup at HL-LHC), and efforts in this area have expanded. Cross-collaboration developments are focusing on the availability of ML algorithms for experimental software, especially in persistifying models and running inferences in production and in enhancing the training capacities of collaborations by submitting jobs to the grid and to facilities offering large CPU/GPU resources [107, 108]. One can also foresee that HEP will increase the benefit from techniques and cross-talk from industry.

4.3.3 Algorithms and Data Structures

Computing platforms are generally evolving towards having more cores or to adopt different architectural models (GPUs, FPGAs) to increase processing capability [4]. The goal for HEP is to improve the throughput of software trigger and event reconstruction applications, so current event models, toolkits and algorithm implementations have to evolve to efficiently exploit these opportunities. The first algorithms that should be targeted are those which are most time consuming.

Since the CWP, HEP now has a number of algorithmic projects that are designed for, or have successfully adapted to, hardware accelerators, most notably to NVIDIA GPUs using CUDA. Two projects targeting Run-3 and with prospects for use in Run-4 are Patatrack (from CMS) [109] and Allen (from LHCb) [110].

Patatrack is a CMS initiative aimed at using heterogeneous computing for charged particle reconstruction. Within the CMS codebase (CMSSW), this project has demonstrated that physics reconstruction code can be written to leverage heterogeneous architectures, like GPUs, and achieve a significant speed-up, while reducing the overall cost and power consumption of a system. An example is the CMS Pixel local reconstruction and the track and vertex reconstruction: these algorithms can run on an NVIDIA T4 GPU with the same performance as 52-56 Xeon cores, at a fraction of the cost and power consumption of an equivalent CPU-only system; further improvements may come pairing one or more high end GPUs with a low power ARM system.

Allen is a data processing framework for GPUs, as well as a specific implementation of a first-level trigger for LHCb Run-3 data-taking. Allen is optimised for sustaining the rate required by real-time processing in LHCb, but can be used as a more general GPU data processing system as it includes a scheduler and a memory manager that can be integrated into Gaudi [111]. This permits the extension and implementation of Allen within other experimental software using the same underlying framework.

ATLAS is also investigating GPU solutions for reconstruction software and its acceleration at HL-LHC, in particular in terms of highly parallel algorithms [112] and to enable the use of machine learning algorithms in object reconstruction and identification.

The experience acquired with these projects shows that GPUs can be used for increasing the throughput in cases of large-size (ALICE) [113] and small-size events per second (LHCb), and to speed up specific parts of the data processing (CMS Patatrack).

It is clear that the technology is improving rapidly in this area. The adoption of portability toolkits is needed to avoid vendor lock-in and/or the need to evolve algorithms by hand to adapt to each new type of computing architecture, as well as to increase the sustainability of the codebase avoiding multiple implementations of the same algorithm.

The **HLS4ML** project [114–116] targets the implementation of machine learning algorithms on FPGA for low-latency applications useful for e.g. Level-1 triggers. The aim of the HLS4ML project is the translation of trained ML models into FPGA firmware. HLS4ML also extends to the production of coprocessor kernels for FPGAs for longer latency applications, and can be used as a tool to design AI-powered ASICs. The **SONIC** application [117] is being developed in parallel, with the goal of facilitating and accelerating the inference of deep neural networks for triggering, reconstruction, and analysis, by providing software to use heterogeneous computing resources as a service targeting next-generation facilities at the energy and intensity frontier (HL-LHC, LBNF).

4.4 Trigger and Reconstruction Software Sharing

Nearly all software solutions presented in this chapter are tailored to a specific experiment, and would require additional work to be adapted to different environments. Nevertheless, it is still useful to maintain open communication channels to discuss design choices and compare performance assessments of different solutions to guide future software and reconstruction design choices.

The use of open-source elements of the LHC software stacks for trigger and reconstruction by smaller experiments, especially in case of common experimental design[‡], is still worthwhile This benefits smaller experiments and increases the return on investment for LHC experiment software. For this reason, we also encourage the addition of common open software projects to the ESCAPE software catalogue.

5 Data Analysis

5.1 Key Analysis Computing Challenges at the HL-LHC

Examination of collision data is, in essence, the primary objective of the experiment collaborations, coming at the end of the data preparation, simulation and reconstruction chain. The stage of analysis, starting in most cases from a standard data format generically referred to as "Analysis Object Data" (AOD) containing reconstructed physics object data, and producing physics results as the end product, poses unique computing challenges. The scope of analysis data processing is broad, encompassing the production of derived data formats as well as end stage analysis actions, such as producing histograms from those intermediate datasets and visualising the final results. In the following, attention is focused on the computing challenges related to analysis for the ATLAS and CMS experiments.

Today, ROOT format AOD files and derived datasets take up the lion's share of disk resources, filling approximately 80% of the total data volume for both ATLAS and CMS [118, 119]. To serve precision analyses that require large event statistics experiments will increase the recorded event rate by an order of magnitude, compared with a projected ~10% annual growth of storage resources. The large pile-up in HL-LHC collisions will compound the storage challenge by inflating the data size per event. Reducing the storage footprint of data analysis is therefore of paramount importance.

The effective CPU needs for end-stage analysis payloads are typically orders of magnitude lower than those in simulation and reconstruction. When scaled up to O(100) analyses per year, each processing the input data dozens of times a year, the total CPU consumption at the HL-LHC is projected to be around 10% of total experimental computing [118, 119]. Central production of analysis data formats may account for another 10-30%. It is worth noting at this point that analysis data access patterns tend to be more chaotic than preceding stages, which can increase the effective resource needs by large factors.

More significant than the global storage and computational costs is the job turnaround time, closely tied to the "time to insight", which is a strong limitation on the speed of analysis publication. One significant constraint is the need for full coverage of the input data, which is inextricably linked to weaknesses in book-keeping tools that make it difficult, if not impossible, to make meaningful studies on subsets of the data. This full coverage requirement is clearly limited by computing infrastructure load and downtime. Another challenge is the multiplication of computing resource demands from the assessment of

[‡]An example of software sharing happens in the case of the FASER experiment, whose offline software (https://gitlab.cern.ch/faser/calypso) is based on a derivative of the ATLAS Athena open-source software, which in turn uses the Gaudi framework. FASER also shares tracking detector components with ATLAS, facilitating the adoption of software relying on similar detector descriptions.

systematic uncertainties, which involves processing many alternate configurations and input data, inflating both the CPU load and storage footprint. Improvement of community standards for metadata handling and uncertainty handling will be very important for HL-LHC.

A last concern is that, while simulation and reconstruction code is mostly optimised and developed under greater scrutiny, analysis code is often produced with emphasis on quick results and less emphasis on code quality or performance. Over time, this may result in inefficient, under-documented code that exacerbates the disk and CPU shortfall and poses a major difficulty when software is re-purposed for a new analysis. Add to this a growing array of alternative toolkits from the data science community, in particular for machine learning, and it is clear that the entropy of the analysis code ecosystem has become a major challenge in itself.

The following sections expand on the major challenges for analysis software, including those identified above. An overview is given of ongoing work on common tools that alleviate these problems, followed by an outlook on R&D prospects that would more significantly transform the HL-LHC analysis model for major resource savings.

5.2 Analysis Processing and Workflows: Current Problems and Solutions

To make data analysis tractable, AOD data are typically reduced both by saving only relevant information for each event, which may require transformations (object calibration, computation of derived variables) beyond simply discarding information, and by skimming out only events that are interesting. In post-reconstruction data formats, lossy compression is also in use and being optimised. Coordination of the data reduction phase is a key point for the organisation of data analysis processing at HL-LHC.

Two approaches have been followed during LHC Run 2 by ATLAS and CMS: ATLAS analysis trains [120] and the CMS mini-AOD, a highly reduced and standardised event content. Analysis trains place software payloads tailored for specific analyses (carriages) into a periodically scheduled centralised execution, amortising data staging costs and sharing some common processing steps [121]. Standardised event contents are instead common reductions that satisfy the needs of the bulk of analyses, avoiding duplication of commonly selected events in multiple datasets. Neither of these approaches provides exhaustive coverage of all the analysis use cases for the experiments and alternative data formats are needed for the small but significant (10-20%) fraction of analyses requiring custom reconstruction that may be CPU-intensive. These exceptions to the rule are expected to remain at the HL-LHC.

It is worth considering the viability of the arguably more flexible option to have analysis trains run over the reconstruction output at the HL-LHC. Extrapolating using a simple model, ATLAS finds that they would produce 35 PB/year of data AOD and 213 PB/year of MC [118]. In comparison, a reduced DAOD_PHYSLITE format (10 kB/event) made from the AOD would result in 0.5 PB/year for data and 2 PB/year for MC. Production of this reduced format using a Data Carousel model [118] allows the much larger AOD to reside on tape, providing significant savings in disk storage. Experience from CMS shows that further reduction (down to ˜2 kB/event for the nano-AOD) could be possible while still

supporting a majority of physics analyses [122]. File size reductions also permit multiple copies to be held on disk for more efficient parallel processing. It is clear that as many analyses as possible need to be fed into this standardised analysis event-content pipeline if analysis computing resources are to be kept under control. Nevertheless, analysis trains or any similar means of synchronising data access could be applied to this highly reduced format. It is worth noting the LHCb analysis model already in Run 3 has to forego the luxury of reprocessing raw data. This constraint from sheer event rate is not without merit as it greatly simplifies the data processing and analysis models and removes one tier of offline data products.

Careful attention needs to be paid to the assessment of systematic uncertainties, particularly with regard to event content standardisation, to avoid creating many derived outputs that differ only minimally. Analysis models that do not require all event information to be present in a single file, instead leveraging ROOT "friend trees" or columnar storage, may be a way to reduce this duplication both in the case of uncertainty calculations and reconstruction augmentations, but will require development of robust strategies for keeping the augmentations in sync with the core events. It is noted that the ROOT data format is quantifiably very efficient for the processing model in HEP [123], and any competitors would have to achieve a similar level of performance.

Access to metadata, defined as cross-event information, is clearly a weak point in the handling of the multiple analysis datasets originating from the LHC. Metadata may include information such as book-keeping of processed events, detector conditions and data quality, a posteriori measured scale factors, and software versioning. This information is often scattered in multiple locations such as: in-file metadata, remote database resources, shared-area text or ROOT files, TWiki pages or even in e-mail. The lack of proper metadata integration and a coherent interface leaves analyses exposed to any problems in data processing. This in turn results in data completeness demands that simply will not work at the HL-LHC, where the huge datasets all but guarantee that some fraction of the data will be unavailable at any given point in time.

At the HL-LHC, as energy and luminosity conditions are quasi stable across the years, datasets from multiple years with different conditions will need to be analysed in a coherent way. The analysis software will then need to be able to fetch and use the proper metadata automatically while today's analysis software often requires dedicated configuration and tuning for each data-taking period. Belle II have taken some steps to address this by using the "global tag" concept commonly used in reconstruction and applying it to analysis, allowing users to better organise and store their analysis metadata. Nevertheless, until all metadata is organised under the same global tag umbrella and accessed through a coherent interface, which is extremely challenging, the problem remains. Improvements here are necessary both to permit efficient studies on partial datasets, and to reduce the risk of user error in metadata access.

The growing complexity of analysis codes makes them extremely fragile (i.e. bug-prone), hardly reusable, and unsuited for analysis preservation needs. The current best efforts at analysis preservation are based on a snapshot of the full analysis setup that can ensure the possibility to re-run the code on the original dataset in a few years. This may

allow a future analyser to reproduce previous results, but will likely not provide any understanding, for the future analyser, about what the analysis was effectively doing. Another use case is to reuse the analysis code on new data, be that the same dataset with improved calibrations or additional data that will improve the precision of the measurement. This brings significant additional challenges, not least related to metadata.

Nevertheless, these preservation efforts encourage better organisation of the analysis as a workflow, and promote the use of version control tools together with continuous integration, which offers a natural route to improving analysis code quality. In addition to these code quality measures, a longer-term solution for code complexity may be the adoption of a Domain-Specific Language (DSL) [15], discussed in more detail later, a model that would have physicists write logical descriptions of their analyses, leaving low-level code generation and hardware optimisation to a backend. Thus analysis design could be quickly understood and shared, sidestepping the usual dearth of documentation, while simultaneously isolating analysis design from implementation and hardware, providing a natural means of analysis preservation. An interesting development in this direction is the REANA platform [124] that supports multiple backends and uses a DSL to provide a framework to run an analysis. Already a valuable resource as a tool for analysing open data, it is interesting to ask what would happen if every analyst first had to structure their analysis and capture their workflows before they started submitting jobs.

5.3 Plans and Recent Progress

Most of the problems identified above were already identified in the community white paper [125] from 2017 and since then some progress was made prototyping new technologies to enable data analysis at the HL-LHC.

A first branch of new technologies is that of efficient data analysis processing in the last steps of the analysis, i.e. when event data needs to be selected, derived and aggregated into histograms. Several platforms developed by industry or data science have emerged in recent years to quickly aggregate large datasets with parallel execution (either on clusters of computers or simply on multi-core servers). Many of those tools have been tested to understand the feasibility of usage in the HEP context providing fresh ideas and new paradigms that have stimulated development within the HEP toolkit.

In particular, the Python ecosystem, implementing many such solutions (e.g. numpy, pandas dataframes), is emerging as a possible alternative to HEP's traditional C++ environment. This is thanks largely to a combination of its versatility as a language, allowing rapid prototyping, and the ability to out-source compute-intensive work to more performant languages like C and C++. This is similar to the experimental software frameworks where Python is used as steering code and C/C++ is used as an efficient backend implementation. The typical Belle II analysis starts with Python using a custom DSL to specify particle selections and algorithms implemented in C++ and define output ntuples. The ntuples are typically analysed by the PyHEP toolkit [126], and packages from the wider data science community. Deeper integration with external Python tools is also particularly important for enabling state-of-the-art machine learning frameworks such as PyTorch and TensorFlow, whose relevance in analysis is likely to grow in the long term. Meanwhile Py-

ROOT, which allows the use of any C++ class from Python, has the potential to provide a coherent analysis ecosystem with an effortless integration of HEP specific tools written in C++ with industry Python big data toolkits.

The ROOT package, which is the foundation and workhorse for LHC analysis, has been the focus of substantial development. On the data storage front, the ROOT team demonstrated better data compression and accelerated reading for a wide variety of data formats. Information density drives compression and this can vary massively between experiments and analyses. This new RNTuple format [7, 127], an evolution of the TTree file format, shows robust and significant performance improvements ($1-5\times$ faster), which could potentially save significant storage space ($10-20\%$) for the HL-LHC. Another area of progress, and potential consolidation in ROOT I/O, is lossy compression, where the number of significant bits for a quantity may be far fewer than a standard storage type's mantissa, enabling further savings in storage space [7].

Inspired by data science tools, the event processing framework was extended with DataFrame-like functionality (RDataFrame) implementing a declarative approach to analysis processing in either Python or C++ (but always executed in C++), which natively exploits multi-core architectures [128] and could be extended to accelerators. Used from Python, it allows pre-processing data in C++, and exporting to numpy. Flexibility in interfacing RDataFrame to non-ROOT data formats has allowed the ALICE collaboration to prototype its Run-3 analysis model on this new technology. Growing use of Machine Learning was also anticipated, and the TMVA toolkit has been improved with substantial optimisations, more complex deep learning models and enhanced interoperability with common data science tools [129, 130], which is of particular importance for inference.

A complementary approach provided in Python is the toolchain of uproot, awkward arrays and coffea [131] for efficient columnar operations and transformations of the data. These function as a lightweight end-stage analysis environment that provides only the elements necessary for pre-processing ROOT inputs and converting them into formats used by the standard ML frameworks, typically numpy. Lightweight distribution using package managers such as pip or conda allows for rapid setup and extension with other Python tools. Good performance appears to have been achieved within the scope of these projects, which has been kept focused, but for the purposes of fair comparison with other existing or emerging options, defining clear benchmarks for I/O and processing speed is essential.

A further feature linked to the growing use of Python is the use of "notebook" technology such as Jupyter. This permits quasi-interactive exploration where the annotated history, including formatted outputs and graphics, can be saved and shared with collaborators for reproduction and adaptation. Although not a substitute for command line scripts in well-tested and complex workflows, notebooks make an effective vessel for software education and have been leveraged as a powerful frontend for access to facilities including CERN's SWAN [132].

5.4 Prospects and R&D needs

As the complexity of analysis data handling and processing grows, developing efficient and robust code using a steadily increasing number of tools running on ever more heterogeneous

hardware becomes more and more difficult. In addition, analysis code is typically re-implemented by tens of individuals, in different experiments, analysis teams or institutes mostly performing the same kind of operations, i.e. data reduction, plotting, variation of systematic uncertainty, fitting, etc. It is clear that tools and frameworks for performing these repeated operations need to provide very efficient interfaces to avoid analysis code bloat. Declarative rather than imperative programming is becoming visibly more prevalent both in and outside of the field as it naturally provides efficient syntax.

DSLs are a generalisation of this declarative concept and have already demonstrated their ability to simplify analysis code. Not only do they allow users to express operations concisely, improving comprehension and reusability, DSLs provide a natural layer of insulation against hardware evolution as low-level, optimised code generation for multiple different platforms and architectures is delegated to the tool and framework experts. Prototypes of DSLs have been developed in recent years [15] for the event processing and histogram production parts of the analysis workflow, and even built into experimental frameworks to perform high-level analysis [133]. Similar efforts can also be investigated in the context of data interpretation and statistical analysis and an interesting example here is the so called "combine tools" developed by CMS and based on RooFit for the Higgs discovery. Descriptions of a full analysis in terms of workflows could then tie together the analysis at the top level [124, 134]. With some effort, the HEP community may be able to converge on effective tools that provide a common solution, ideally across experiments.

There is an obvious need for good integration with analysis backends, whether these be local CPU, batch or grid resources. At some stage in the analysis process, interactivity or fast turn-around is needed. While typical grid task completion has a time scale of one day to one week, fast turn-around exploration requires answers within a few seconds up to perhaps a few hours in order to keep people productive. As analysis tasks are often I/O-intensive and not necessarily-CPU intensive, the right trade off between CPU and high bandwidth storage should be carefully studied. More detailed studies of resource usage and its evolution, especially as pertains to the need for heterogeneous compute, is needed to ensure the right resource balance is available in the HL-LHC computing infrastructure. This will require more interaction between the analysis and infrastructure communities.

As a proof of principle, an unoptimised Higgs discovery analysis has been rerun on Google Cloud in a few minutes as the infrastructure was able to quickly provide tens of thousands of CPUs with guaranteed bandwidth to storage of 2 Gb/coresec [135]. Scaling this to sustained, diverse analysis payloads will be far from trivial, so the question becomes how we can compare different approaches to provisioning analysis. Eight initial benchmark challenges have been defined to set the scope that should be addressed by analysis systems [136]. What remains to be demonstrated is the capability of such systems to scale to full analysis complexity, including tasks such as the handling of systematic uncertainties and data-driven background estimation, and these are yet to be incorporated into the benchmarks.

In contrast to the activity on analysis languages and infrastructure, metadata is a topic that is almost entirely neglected by the analysis community, even while computing experts have long understood its importance. That is likely due to the absence of a single coherent

stake-holder for and provider of analysis metadata. Event data comes in a ROOT file, but already within a single experiment there are numerous sources of metadata that operate on vastly different timescales and this poses a significant challenge to standardisation. As analyses and detectors grow in complexity, further divergence may amplify the challenge, making it critical to begin addressing this problem promptly. Nevertheless, the drive to capture a complete analysis must drag metadata with it, and the gleeful adoption of the declarative paradigm will make database-like interfaces seem less scary than they once were and may finally allow a more complete integration with analysis code.

One final comment in reviewing recent developments is worth noting. Exposure of the HEP community to industry-standard data science tools has undoubtedly been a force for good and a net gain for the community, providing fresh ideas and new ways of working - RDataFrame is an excellent example. It is clear that the larger and better-supported data science community will continue to innovate and produce tools that HEP will benefit from, both by using directly and by borrowing their ideas for our own tools. Taking machine learning as an example, it is extremely likely that industry will blaze a trail that HEP would do well to follow and we should continue to build bridges to those tools. Equally though, HEP will continue to have its own unique challenges and it is less likely that ultra-performant machine learning inference will be top of the data scientists wishlist. Given the challenges presented by extreme-throughput analysis at the HL-LHC, the right balance must be found between software development within the community and relying on external toolkits.

Finally, attention must be given to ensuring the longevity of any projects that the HEP community decides to adopt en masse. Whether by specific institutes, funding schemes or other collective programmes, responsibility for key software will need to be ensured in the long term. The development of a secure support base will therefore need to be considered for emerging projects that show significant promise.

6 Summary

The main challenges and R&D lines of interest have been discussed above. There are clearly identified priority topics for the different areas in the next few years[§].

6.1 Physics Event Generators

1. Gain a more detailed understanding of current CPU costs by accounting and profiling.
2. Survey generator codes to understand the best way to move to GPUs and vectorised code, and prototype the port of the software to GPUs using data-parallel paradigms.

[§]Longer term R&D, into high-risk, high-reward blue sky areas, such as quantum or neuromorphic computing, should happen at a low level, but are almost certainly too far from production to deliver improvements for Run 4 of the LHC.

3. Support efforts to optimise phase space sampling and integration algorithms, including the use of Machine Learning techniques such as neural networks.
4. Promote research on how to reduce the cost associated with negative weight events, using new theoretical or experimental approaches.
5. Promote collaboration, training, funding and career opportunities in the generator area.

6.2 Detector Simulation

1. Undertake a detailed performance analysis of current production detector simulation code, to better understand where performance limitations in data and cache locality arise from.
2. Improve current codes through refactoring and specialised HEP models to avoid bottlenecks identified.
3. Develop further fast simulation models and techniques, including machine learning based options in addition to optimised parametric approaches as well as common frameworks for their tuning and validation.
4. Undertake R&D into GPU codes for tackling very specific, time consuming, parts of the simulation for HEP. Specifically calorimetry, including geometry, field and electro-magnetic physics processes.
5. Develop integration prototypes to exercise and benchmark the simultaneous use of GPU specific libraries with CPU-based software (e.g. Geant4) to cover all particles and processes required in experiments simulation frameworks.

6.3 Reconstruction and Software Triggers

1. Develop fast and performant reconstruction software, making optimal use of hardware resources. A particularly crucial development for HL-LHC is that of pattern recognition solutions (including timing information, if available) that can withstand the increase of pile-up at HL-LHC. Wherever possible, solutions should be uniform for online and offline software, so the experiments can benefit from consistent event selection between trigger and final analysis and from more effective real-time analysis results.
2. Develop solutions that can exploit accelerators (FPGA, GPU) for trigger and reconstruction, including general-purpose interoperability libraries with a focus to generalising and mitigating their dependencies on architecture specific codes.
3. Further promote activities in the area of data quality and software monitoring, as the quality of the recorded and reconstructed data is paramount for all physics analysis.
4. Support efforts towards the implementation of machine learning models in experimental framework (C++) to enable more widespread use in trigger and reconstruction software.

6.4 Data Analysis

1. Identify a CPU-efficient data reduction and storage model that serves the majority of analyses with a footprint of O(1kb/event), retaining the flexibility to accommodate the needs of specialised analyses.
2. Streamline analysis metadata access and calibration schemes, and provide effective book-keeping for fractional datasets.
3. Develop cross-experimental tools establishing declarative syntax and/or languages for analysis description, interfaced with distributed computing backends.
4. Define and improve schemes for interoperability of end-stage experimental software with data science and machine learning frameworks.

Undertaking this programme of priority topics and R&D requires, first and foremost, investment from funding agencies to support developments. A new generation of developers is also needed to refresh and regenerate the long lived software projects on which the field so heavily relies. The transition from R&D to production ready software is usually long and so it is urgent to undertake this investment now, with the positive outcomes ready to become integrated in time for HL-LHC. Regenerating the cadre of software developers in HEP also requires support for realistic long term career prospects for those who specialise in this vital area. Collaboration with industry can certainly be fruitful and is to be encouraged to allow early access to technology and communication of HEP problems and priorities. R&D that is successful will become the next generation of production software, but this will require lifetime support for maintainance and further evolution.

Almost all the domain areas identify the use of compute accelerators, particularly GPUs, as being a priority item, which matches our expectation of how computing hardware will evolve. As a common problem area, this is an obvious area in which expertise should be developed.

The stakeholders in HEP must be able to give their priorities and feedback on R&D areas. Each project has mechanisms for doing this, while the HSF can help to provide overall input on prioritisation and coordination of common cross-experiment activities.

References

[1] The HEP Software Foundation. *The Importance of Software and Computing to Particle Physics*. Dec. 2018. DOI: 10.5281/zenodo.2413005. URL: https://doi.org/10.5281/zenodo.2413005.

[2] Richard Keith Ellis et al. *Physics Briefing Book: Input for the European Strategy for Particle Physics Update 2020*. Tech. rep. arXiv:1910.11775. 254 p. Geneva, Oct. 2019. URL: http://cds.cern.ch/record/2691414.

[3] Johannes Albrecht et al. "A Roadmap for HEP Software and Computing R&D for the 2020s". In: *Computing and Software for Big Science* 3.1 (Mar. 2019), p. 7. ISSN: 2510-2044. DOI: 10.1007/s41781-018-0018-8. URL: https://doi.org/10.1007/s41781-018-0018-8.

[4] John L. Hennessy and David A. Patterson. "A New Golden Age for Computer Architecture". In: *Communications of the ACM* 62.2 (Feb. 2019), pp. 48–60. DOI: 10.1145/3282307.

[5] Paolo Calafiura et al. *HEP Software Foundation Community White Paper Working Group - Data Processing Frameworks*. 2018. arXiv: 1812.07861 [physics.comp-ph]. URL: https://arxiv.org/abs/1812.07861.

[6] Johannes Albrecht et al. *HEP Community White Paper on Software trigger and event reconstruction: Executive Summary*. 2018. arXiv: 1802.08640 [physics.comp-ph]. URL: https://arxiv.org/abs/1802.08640.

[7] ROOT Team et al. *Software Challenges For HL-LHC Data Analysis*. 2020. arXiv: 2004.07675 [physics.data-an]. URL: https://arxiv.org/abs/2004.07675.

[8] *XRootD Website*. URL: https://xrootd.slac.stanford.edu/.

[9] Benjamin Couturier et al. *HEP Software Foundation Community White Paper Working Group - Software Development, Deployment and Validation*. 2017. arXiv: 1712.07959 [physics.comp-ph]. URL: https://arxiv.org/abs/1712.07959.

[10] HEP Software Foundation et al. *HEP Software Foundation Community White Paper Working Group - Training, Staffing and Careers*. 2018. arXiv: 1807.02875 [physics.ed-ph]. URL: https://arxiv.org/abs/1807.02875.

[11] Jakob Blomer et al. "Distributing LHC application software and conditions databases using the CernVM file system". In: *Journal of Physics: Conference Series* 331.4 (Dec. 2011), p. 042003. DOI: 10.1088/1742-6596/331/4/042003. URL: https://doi.org/10.1088%2F1742-6596%2F331%2F4%2F042003.

[12] J. Blomer et al. "The Evolution of Global Scale Filesystems for Scientific Software Distribution". In: *Computing in Science Engineering* 17.6 (2015), pp. 61–71.

[13] Jakob Blomer et al. "Delivering LHC Software to HPC Compute Elements with CernVM-FS". In: *High Performance Computing*. Ed. by Julian M. Kunkel et al. Cham: Springer International Publishing, 2017, pp. 724–730. ISBN: 978-3-319-67630-2.

[14] Blomer, Jakob et al. "Towards a serverless CernVM-FS". In: *EPJ Web Conf.* 214 (2019), p. 09007. DOI: 10.1051/epjconf/201921409007. URL: https://doi.org/10.1051/epjconf/201921409007.

[15] *ADL Workshop at Fermilab*. URL: https://indico.cern.ch/event/769263/timetable/?view=standard.

[16] Johannes Albrecht et al. "A Roadmap for HEP Software and Computing R&D for the 2020s". In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 7. DOI: 10.1007/s41781-018-0018-8. arXiv: 1712.06982 [physics.comp-ph].

[17] CERN HSF Physics Event Generator Computing Workshop. Nov. 2018. URL: https://indico.cern.ch/event/751693.

[18] HSF Physics Generators Working Group. URL: https://hepsoftwarefoundation.org/workinggroups/generators.html.

[19] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *Journal of High Energy Physics* 2014.7 (July 2014). DOI: 10.1007/jhep07(2014)079.

[20] Enrico Bothmann et al. "Event generation with Sherpa 2.2". In: *SciPost Physics* 7.3 (Sept. 2019). DOI: 10.21468/scipostphys.7.3.034.

[21] Stefano Frixione, Paolo Nason, and Carlo Oleari. "Matching NLO QCD computations with parton shower simulations: the POWHEG method". In: *Journal of High Energy Physics* 2007.11 (Nov. 2007), pp. 070–070. DOI: 10.1088/1126-6708/2007/11/070.

[22] HSF Physics Generators Working Group. *Challenges in Monte Carlo event generator software for High-Luminosity LHC*. CERN-LPCC-2020-002. arXiv: 2004.13687.

[23] G. P. Lepage. *VEGAS: an adaptive multi-dimensional integration program*. Cornell report CLNS-447 (1980). URL: https://cds.cern.ch/record/123074.

[24] S. Kawabata. "A new Monte Carlo event generator for high energy physics". In: *Computer Physics Communications* 41.1 (July 1986), pp. 127–153. DOI: 10.1016/0010-4655(86)90025-1.

[25] P. Nason. *MINT: a Computer Program for Adaptive Monte Carlo Integration and Generation of Unweighted Distributions*. 2007. arXiv: 0709.2085.

[26] S Jadach. "Foam: A general-purpose cellular Monte Carlo event generator". In: *Computer Physics Communications* 152.1 (Apr. 2003), pp. 55–100. DOI: 10.1016/s0010-4655(02)00755-5.

[27] Fabio Maltoni and Tim Stelzer. "MadEvent: automatic event generation with MadGraph". In: *Journal of High Energy Physics* 2003.02 (Feb. 2003), pp. 027–027. DOI: 10.1088/1126-6708/2003/02/027.

[28] Tanju Gleisberg and Stefan Höche. "Comix, a new matrix element generator". In: *Journal of High Energy Physics* 2008.12 (Dec. 2008), pp. 039–039. DOI: 10.1088/1126-6708/2008/12/039.

[29] Christina Gao et al. "Event Generation with Normalizing Flows". In: (2020). DOI: 10.1103/PhysRevD.101.076002.

[30] Joshua Bendavid. *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*. 2017. arXiv: 1707.00028.

[31] Enrico Bothmann et al. *Exploring phase space with Neural Importance Sampling*. 2020. arXiv: 2001.05478.

[32] Matthew D. Klimek and Maxim Perelstein. *Neural Network-Based Approach to Phase Space Integration*. 2018. arXiv: 1810.11509.

[33] Michelangelo L. Mangano, Mauro Moretti, and Roberto Pittau. "Multijet matrix elements and shower evolution in hadronic collisions: -jets as a case study". In: *Nuclear Physics B* 632.1-3 (June 2002), pp. 343–362. DOI: 10.1016/s0550-3213(02)00249-3.

[34] Leif Lönnblad. "Correcting the Colour-Dipole Cascade Model with Fixed Order Matrix Elements". In: *Journal of High Energy Physics* 2002.05 (May 2002), pp. 046–046. DOI: 10.1088/1126-6708/2002/05/046.

[35] Rikkert Frederix and Stefano Frixione. "Merging meets matching in MC@NLO". In: *Journal of High Energy Physics* 2012.12 (Dec. 2012). DOI: 10.1007/jhep12(2012)061.

[36] Stefan Höche, Frank Krauss, and Marek Schönherr. "Uncertainties in MEPS@NLO calculations of h+jets". In: *Physical Review D* 90.1 (July 2014). DOI: 10.1103/physrevd.90.014012.

[37] J. Alwall et al. "Comparative study of various algorithms for the merging of parton showers and matrix elements in hadronic collisions". In: *The European Physical Journal C* 53.3 (Dec. 2007), pp. 473–500. DOI: 10.1140/epjc/s10052-007-0490-5.

[38] Johan Alwall, Simon de Visscher, and Fabio Maltoni. "QCD radiation in the production of heavy colored particles at the LHC". In: *Journal of High Energy Physics* 2009.02 (Feb. 2009), pp. 017–017. DOI: 10.1088/1126-6708/2009/02/017.

[39] Olivier Mattelaer. "On the maximal use of Monte Carlo samples: re-weighting events at NLO accuracy". In: *The European Physical Journal C* 76.12 (Dec. 2016). DOI: 10.1140/epjc/s10052-016-4533-7.

[40] R. Frederix et al. *On the reduction of negative weights in MC@NLO-type matching procedures*. 2020. arXiv: 2002.12716.

[41] Jeppe R. Andersen et al. *A Positive Resampler for Monte Carlo Events with Negative Weights*. 2020. arXiv: 2005.09375 [hep-ph].

[42] Benjamin Nachman and Jesse Thaler. *A Neural Resampler for Monte Carlo Reweighting with Preserved Uncertainties*. 2020. arXiv: 2007.11586 [hep-ph].

[43] D. Konstantinov. *Optimization of Pythia8*. EP-SFT group meeting, CERN (January 2020). URL: https://indico.cern.ch/event/890670.

[44] Christian Bauer et al. *Computing for Perturbative QCD - A Snowmass White Paper*. 2013. arXiv: 1309.3598.

[45] J.T. Childers et al. "Adapting the serial Alpgen parton-interaction generator to simulate LHC collisions on millions of parallel threads". In: *Computer Physics Communications* 210 (Jan. 2017), pp. 54–59. DOI: 10.1016/j.cpc.2016.09.013.

[46] Stefan Höche, Stefan Prestel, and Holger Schulz. "Simulation of vector boson plus many jet final states at the high luminosity LHC". In: *Physical Review D* 100.1 (July 2019). DOI: 10.1103/physrevd.100.014024.

[47] O. Mattelaer. *MG5aMC status and plans*. In Ref. [17].

[48] ALICE Collaboration. *ALICE technical design report of the computing*. Tech. rep. CERN-LHCC-2005-018. ALICE-TDR-12. Geneva: CERN, 2005. URL: http://cds.cern.ch/record/832753.

[49] ATLAS Collaboration. *ATLAS Computing: technical design report*. Tech. rep. CERN-LHCC-2005-022, ATLAS-TDR-17. Geneva: CERN, 2005. URL: http://cds.cern.ch/record/837738.

[50] CMS Collaboration. *CMS computing: Technical Design Report*. Tech. rep. CERN-LHCC-2005-023, CMS-TDR-7. Geneva: CERN, 2005. URL: http://cds.cern.ch/record/838359.

[51] LHCb Collaboration. *LHCb computing: Technical Design Report*. Tech. rep. CERN-LHCC-2005-019, LHCb-TDR-11. Geneva: CERN, 2005. URL: http://cds.cern.ch/record/835156.

[52] I Bird et al. *Update of the Computing Models of the WLCG and the LHC Experiments*. Tech. rep. CERN-LHCC-2014-014. LCG-TDR-002. 2014. URL: http://cds.cern.ch/record/1695401.

[53] ALICE Collaboration. *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Tech. rep. CERN-LHCC-2015-006. ALICE-TDR-019. Apr. 2015. URL: http://cds.cern.ch/record/2011297.

[54] LHCb Collaboration. *Computing Model of the Upgrade LHCb experiment*. Tech. rep. CERN-LHCC-2018-014. LHCB-TDR-018. Geneva: CERN, 2018. URL: http://cds.cern.ch/record/2319756.

[55] ATLAS Collaboration. *ATLAS Computing update*. Feb. 2019. URL: https://indico.cern.ch/event/754731/contributions/3127500/subcontributions/262952/at

[56] CMS Collaboration. *Report from CMS Offline Software and Computing*. Sept. 2019. URL: https://indico.cern.ch/event/754734/contributions/3127510/subcontributions/262959/at

[57] S. Agostinelli et al. "Geant4—a simulation toolkit". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (July 2003), pp. 250–303. DOI: 10.1016/s0168-9002(03)01368-8. URL: https://doi.org/10.1016/s0168-9002(03)01368-8.

[58] J. Allison et al. "Geant4 developments and applications". In: *IEEE Transactions on Nuclear Science* 53.1 (Feb. 2006), pp. 270–278. DOI: 10.1109/tns.2006.869826. URL: https://doi.org/10.1109/tns.2006.869826.

[59] J. Allison et al. "Recent developments in Geant4". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 835 (Nov. 2016), pp. 186–225. DOI: 10.1016/j.nima.2016.06.125. URL: https://doi.org/10.1016/j.nima.2016.06.125.

[60] HEP Software Foundation et al. *HEP Software Foundation Community White Paper Working Group - Detector Simulation*. 2018. arXiv: 1803.04165 [physics.comp-ph].

[61] A. Gheata. *Design, implementation and performance results of the GeantV prototype*. Outcome of the GeantV prototype - HSF meeting, CERN (October 2019). URL: https://indico.cern.ch/event/818702.

[62] G. Amadio et al. *GeantV: Results from the prototype of concurrent vector particle transport simulation in HEP*. 2020. arXiv: 2005.00949 [physics.comp-ph].

[63] Vladimir Ivanchenko and Sunanda Banerjee. "Upgrade of CMS Full Simulation for Run 2". In: *EPJ Web of Conferences* 214 (2019). Ed. by A. Forti et al., p. 02012. DOI: 10.1051/epjconf/201921402012. URL: https://doi.org/10.1051/epjconf/201921402012.

[64] Miha Muskinja et al. *Geant4 performance optimization in the ATLAS experiment*. ATL-SOFT-SLIDE-2019-811, Talk given at CHEP2019, Adelaide, November 2019. URL: http://cds.cern.ch/record/2696432.

[65] M Hildreth et al. "CMS Full Simulation for Run-2". In: *Journal of Physics: Conference Series* 664.7 (Dec. 2015), p. 072022. DOI: 10.1088/1742-6596/664/7/072022. URL: https://doi.org/10.1088/1742-6596/664/7/072022.

[66] Dominik Müller. "Adopting new technologies in the LHCb Gauss simulation framework". In: *EPJ Web of Conferences* 214 (2019). Ed. by A. Forti et al., p. 02004. DOI: 10.1051/epjconf/201921402004. URL: https://doi.org/10.1051/epjconf/201921402004.

[67] Marilena Bandieramonte et al. *Multithreaded simulation for ATLAS: challenges and validation strategy*. Tech. rep. ATL-SOFT-PROC-2020-030. Geneva: CERN, Mar. 2020. URL: https://cds.cern.ch/record/2712940.

[68] Steven Farrell et al. “Multi-threaded ATLAS simulation on Intel Knights Landing processors”. In: *J. Phys. Conf. Ser.* 898.4 (2017). Ed. by Richard Mount and Craig Tull, p. 042012. DOI: 10.1088/1742-6596/898/4/042012.

[69] Douglas Benjamin et al. “Large scale fine grain simulation workflows (”Jumbo Jobs”) on HPC’s”. In: (Oct. 2019). URL: https://cds.cern.ch/record/2696330.

[70] M Hildreth, V N Ivanchenko, and D J Lange and. “Upgrades for the CMS simulation”. In: *Journal of Physics: Conference Series* 898 (Oct. 2017), p. 042040. DOI: 10.1088/1742-6596/898/4/042040. URL: https://doi.org/10.1088/1742-6596/898/4/042040.

[71] G. Grindhammer, M. Rudowicz, and S. Peters. “The fast simulation of electromagnetic and hadronic showers”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 290.2 (1990), pp. 469–488. ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(90)90566-0. URL: http://www.sciencedirect.com/science/article/pii/0168900290905660.

[72] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters”. In: *Physical Review Letters* 120.4 (Jan. 2018). ISSN: 1079-7114. DOI: 10.1103/physrevlett.120.042003. URL: http://dx.doi.org/10.1103/PhysRevLett.120.042003.

[73] Gul Rukh Khattak et al. “Particle Detector Simulation using Generative Adversarial Networks with Domain Related Constraints”. In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, Dec. 2019. DOI: 10.1109/icmla.2019.00014. URL: https://doi.org/10.1109/icmla.2019.00014.

[74] Aishik Ghosh et al. *Fast simulation methods in ATLAS: from classical to generative models*. Nov. 2019. DOI: 10.5281/zenodo.3599705. URL: https://doi.org/10.5281/zenodo.3599705.

[75] *Deep generative models for fast shower simulation in ATLAS*. Tech. rep. ATL-SOFT-PUB-2018-001. Geneva: CERN, July 2018. URL: https://cds.cern.ch/record/2630433.

[76] D. Müller et al. “ReDecay: a novel approach to speed up the simulation at LHCb”. In: *The European Physical Journal C* 78.12 (Dec. 2018). DOI: 10.1140/epjc/s10052-018-6469-6. URL: https://doi.org/10.1140/epjc/s10052-018-6469-6.

[77] Fedor Ratnikov. *Generative Adversarial Networks for LHCb Fast Simulation*. LHCb-TALK-2019-403, Talk given at CHEP2019, Adelaide, November 2019. URL: http://cds.cern.ch/record/2699549.

[78] Adam Davis. *Fast Simulations at LHCb*. LHCb-TALK-2019-404, Talk given at CHEP2019, Adelaide, November 2019. URL: https://cds.cern.ch/record/2699550.

[79] Benedikt Volker. *Fast Simulation Overview from ALICE*. HSF Simulation Working Group meeting, CERN (January 2019). URL: https://indico.cern.ch/event/782507/contributions/3293465/.

[80] J. de Favereau and and C. Delaere and P. Demin and A. Giammanco and V. Lemaître and A. Mertens and M. Selvaggi. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment". In: *Journal of High Energy Physics* 2014.2 (Feb. 2014). DOI: 10.1007/jhep02(2014)057. URL: https://doi.org/10.1007/jhep02(2014)057.

[81] J. Conway, R. Culbertson, R. Demina, B. Kilminster, M. Kruse, S. Mrenna, J. Nielsen, M. Roco, A. Pierce, J. Thaler, T. Wizansky. *Pretty Good Simulation*. URL: http://conway.physics.ucdavis.edu/research/software/pgs/pgs4-general.htm.

[82] Shogo Okada et al. "MPEXS-DNA, a new GPU-based Monte Carlo simulator for track structures and radiation chemistry at subcellular scale". In: *Medical Physics* 46.3 (Jan. 2019), pp. 1483–1500. DOI: 10.1002/mp.13370. URL: https://doi.org/10.1002/mp.13370.

[83] Julien Bert et al. "Hybrid GATE: a GPU/CPU implementation for imaging and therapy applications". In: Oct. 2012. DOI: 10.1109/NSSMIC.2012.6551511.

[84] Simon Blyth. "Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA® OptiXTM". In: *EPJ Web of Conferences* 214 (2019). Ed. by A. Forti et al., p. 02027. DOI: 10.1051/epjconf/201921402027. URL: https://doi.org/10.1051/epjconf/201921402027.

[85] Steven P. Hamilton and Thomas M. Evans. "Continuous-energy Monte Carlo neutron transport on GPUs in the Shift code". In: *Annals of Nuclear Energy* 128 (June 2019), pp. 236–247. DOI: 10.1016/j.anucene.2019.01.012. URL: https://doi.org/10.1016/j.anucene.2019.01.012.

[86] Simon Blyth. *Meeting the challenge of JUNO simulation with Opticks GPU Optical Photon Accelerator*. Talk given at CHEP2019, Adelaide, November 2019. URL: https://indico.cern.ch/event/773049/contributions/3581370/.

[87] Sandro Wenzel and Yang Zhang and. "Accelerating navigation in the VecGeom geometry modeller". In: *Journal of Physics: Conference Series* 898 (Oct. 2017), p. 072032. DOI: 10.1088/1742-6596/898/7/072032. URL: https://doi.org/10.1088%2F1742-6596%2F898%2F7%2F072032.

[88] Johannes Albrecht et al. *HEP Community White Paper on Software trigger and event reconstruction*. 2018. arXiv: 1802.08638 [physics.comp-ph].

[89] ATLAS Collaboration. *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*. Tech. rep. CERN-LHCC-2017-020. ATLAS-TDR-029. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2285584.

[90] CMS collaboration. *The Phase-2 Upgrade of the CMS Level-1 Trigger*. Tech. rep. CERN-LHCC-2020-004. CMS-TDR-021. Draft version. Geneva: CERN, Apr. 2020. URL: https://cds.cern.ch/record/2714892.

[91] R. Aaij et al. "A comprehensive real-time analysis model at the LHCb experiment". In: *JINST* 14.04 (2019), P04006. DOI: 10.1088/1748-0221/14/04/P04006. arXiv: 1903.01360 [hep-ex].

[92] P Buncic, M Krzewicki, and P Vande Vyvre. *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Tech. rep. CERN-LHCC-2015-006. ALICE-TDR-019. Apr. 2015. URL: https://cds.cern.ch/record/2011297.

[93] Hannes Sakulin. *40 MHz Level-1 Trigger Scouting for CMS*. Nov. 2019. DOI: 10.5281/zenodo.3598769. URL: https://doi.org/10.5281/zenodo.3598769.

[94] *Fast Track Reconstruction for HL-LHC*. Tech. rep. ATL-PHYS-PUB-2019-041. Geneva: CERN, Oct. 2019. URL: https://cds.cern.ch/record/2693670.

[95] Xiaocong Ai. "Acts: A common tracking software". In: *Meeting of the Division of Particles and Fields of the American Physical Society*. Oct. 2019. arXiv: 1910.03128 [physics.ins-det].

[96] C Gumpert et al. "ACTS: from ATLAS software towards a common track reconstruction software". In: *Journal of Physics: Conference Series* 898 (Oct. 2017), p. 042011. DOI: 10.1088/1742-6596/898/4/042011. URL: https://doi.org/10.1088%2F1742-6596%2F898%2F4%2F042011.

[97] Giuseppe Cerati et al. *Speeding up Particle Track Reconstruction in the CMS Detector using a Vectorized and Parallelized Kalman Filter Algorithm*. 2019. arXiv: 1906.11744 [physics.ins-det].

[98] Xiangyang Ju et al. "Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors". In: *33rd Annual Conference on Neural Information Processing Systems*. Mar. 2020. arXiv: 2003.11603 [physics.ins-det].

[99] ATLAS Collaboration. *Technical Proposal: A High-Granularity Timing Detector for the ATLAS Phase-II Upgrade*. Tech. rep. CERN-LHCC-2018-023. LHCC-P-012. Geneva: CERN, June 2018. URL: https://cds.cern.ch/record/2623663.

[100] Collaboration CMS. *A MIP Timing Detector for the CMS Phase-2 Upgrade*. Tech. rep. CERN-LHCC-2019-003. CMS-TDR-020. Geneva: CERN, Mar. 2019. URL: https://cds.cern.ch/record/2667167.

[101] Lindsey Grey. *First Steps Towards Four-Dimensional Tracking: Timing Layers at the HL-LHC*. Mar. 2018. URL: https://indico.cern.ch/event/658267/contributions/2870293/.

[102] Valentina Akishina and Ivan Kisel. "4-Dimensional Event Building in the First-Level Event Selection of the CBM Experiment". In: *J. Phys. Conf. Ser.* 664.7 (2015), p. 072027. DOI: 10.1088/1742-6596/664/7/072027.

[103] Valentina Akishina. *Four-dimensional event reconstruction in the CBM experiment*. 2016. URL: https://indico.gsi.de/event/6198/contributions/28511/attachments/20671/26101/Akishi

[104] M Adinolfi et al. "LHCb data quality monitoring". In: *J. Phys. : Conf. Ser.* 898.9 (2017), 092027. 5 p. DOI: 10.1088/1742-6596/898/9/092027. URL: https://cds.cern.ch/record/2298467.

[105] Pol, Adrian Alan et al. "Anomaly detection using Deep Autoencoders for the assessment of the quality of the data acquired by the CMS experiment". In: *EPJ Web Conf.* 214 (2019), p. 06008. DOI: 10.1051/epjconf/201921406008. URL: https://doi.org/10.1051/epjconf/201921406008.

[106] "Evolution of the upgrade LHCb HLT1 throughput". In: (July 2019). URL: https://cds.cern.ch/record/2684267.

[107] *Lightweight Trained Neural Network*. URL: https://github.com/lwtnn/lwtnn.

[108] *Open Neural Network Exchange*. URL: https://onnx.ai.

[109] Andrea Bocci. *Heterogeneous online reconstruction at CMS*. Nov. 2019. DOI: 10.5281/zenodo.3598824. URL: https://doi.org/10.5281/zenodo.3598824.

[110] Roel Aaij et al. "Allen: A high level trigger on GPUs for LHCb". In: (Dec. 2019). arXiv: 1912.09161 [physics.ins-det].

[111] Guy Barrand et al. "GAUDI- A Software Architecture and Framework for building HEP Data Processing Applications". In: *Computer Physics Communications* 140 (Oct. 2001), pp. 45–55. DOI: 10.1016/S0010-4655(01)00254-5.

[112] Attila Krasznahorkay et al. *GPU Usage in ATLAS Reconstruction and Analysis*. Nov. 2019. DOI: 10.5281/zenodo.3599103. URL: https://doi.org/10.5281/zenodo.3599103.

[113] David Rohr. *GPU-based reconstruction and data compression at ALICE during LHC Run 3*. Nov. 2019. DOI: 10.5281/zenodo.3599418. URL: https://doi.org/10.5281/zenodo.3599418.

[114] Javier Duarte et al. "Fast inference of deep neural networks in FPGAs for particle physics". In: *JINST* 13.07 (2018), P07027. DOI: 10.1088/1748-0221/13/07/P07027. arXiv: 1804.06913 [physics.ins-det].

[115] Sioni Summers et al. "Fast inference of Boosted Decision Trees in FPGAs for particle physics". In: (Feb. 2020). arXiv: 2002.02534 [physics.comp-ph].

[116] Vladimir Loncar et al. "Compressing deep neural networks on FPGAs to binary and ternary precision with HLS4ML". In: (Mar. 2020). arXiv: 2003.06308 [cs.LG].

[117] Javier Duarte et al. "FPGA-accelerated machine learning inference as a service for particle physics computing". In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 13. DOI: 10.1007/s41781-019-0027-2. arXiv: 1904.08986 [physics.data-an].

[118] *Evolution of the ATLAS analysismodel for Run-3 and prospects for HL-LHC* *ATLAS Collaboration. *ATL-SOFT-PROC-2020-002*. URL: https://cds.cern.ch/record/2708664/.

[119] CMS Collaboration. URL: %5Chref%7Bhttps://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults%

[120] ATLAS Collaboration. "A new petabyte-scale data derivationframework for ATLAS." In: *Journal of Physics: Conference Series 664.072007* ().

[121] ALICE Collaboration. "The ALICE analysis train system". In: *J.Phys.Conf.Ser. 608 (2015) no. 012019* (). URL: http://arxiv.org/abs/arXiv:1502.06381.

[122] CMS Collaboration. "A further reduction in CMS event datafor analysis: the NANOAOD format". In: *EPJ Web of Conferences. 214. 06021.* ().

[123] J Blomer. "A quantitative review of data formats for HEP analyses". In: *Journal of Physics: Conference Series. 2018. p. 032020.* ().

[124] *Reana homepage*. URL: http://reanahub.io/.

[125] HEP Software Foundation Collaboration. "HEP SoftwareFoundation Community White Paper Working Group - Data Analysis andInterpretation". In: *arXiv:1804.03983* (). URL: https://arxiv.org/abs/1804.03983.

[126] *PyHEP*. URL: https://github.com/hsf-training/PyHEP-resources.

[127] J Blomer et al. "Evolution of the ROOT Tree I/O". In: (). URL: https://arxiv.org/abs/2003.07669.

[128] D Piparo et al. "RDataFrame: Easy Parallel ROOT Analysis at 100 Threads". In: *EPJ Web of Conferences 214, 06029 (2019)* (). URL: https://www.epj-conferences.org/articles/epjconf/abs/2019/19/epjconf_chep2018_06029/

[129] L Moneta et al. "Machine Learning with ROOT/TMVA". In: *EPJ Web of Conferences;Proceedings of CHEP 2019 (in print). 2020.* ().

[130] K Albertsson et al. "Fast Inference for Machine Learning in ROOT/TMVA". In: *EPJ Web ofConferences; Proceedings of CHEP 2019 (in print). 2020.* ().

[131] *scikit-hep*. URL: https://github.com/scikit-hep/uproot.

[132] *SWAN Homepage*. URL: https://swan.web.cern.ch/.

[133] T Kuhr et al. "The Belle II Core Software". In: *BELLE2-PUB-TE-2018-002* (). URL: https://docs.belle2.org/record/1044.

[134] M Rieger et al. "Design and Execution of make-like,distributed Analyses based on Spotify's Pipelining Package Luigi". In: *arXiv:1706.00955* (). URL: https://arxiv.org/abs/1706.00955.

[135] L Heinrich & R Rocha. *Reperforming a Nobel Prizediscovery on Kubernetes*. URL: https://indico.cern.ch/event/773049/contributions/3581373.

[136] *ADL benchmarks index*. URL: https://github.com/iris-hep/adl-benchmarks-index.