High Energy Physics Simulations using HPCs

Zachary Marshall,[1,*] Miha Muskinja,[1,†] Vincent R. Pascuzzi,[1,‡] and Vakho Tsulaia[1,§]

[1] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

(Dated: September 1, 2020)

We present general ideas for running large-scale detector simulations on high-performance computers (HPC). Some properties of HPC systems are considered, e.g. , and several studies to benchmark and evaluate the use of these machines for high-energy physics purposes are proposed.

Keywords: CompF2, theoretical calculations, simulation

Significant funding and effort is being invested in the construction of High-Performance Computing (HPC) systems worldwide [1–6]. These systems present an excellent opportunity for large-scale computing, but they are best suited for problems that satisfy several constraints:

- Workflows should not demand low latency. HPCs are generally high-efficiency, and therefore do not have resources available on short notice.
- Workflows should have minimal input and output (I/O), including access to external services. Several solutions to avoid I/O constraints, including high-speed buffer systems, have been developed, and clever caching and merging systems can work around some of these constraints. This also means that database access is normally limited, so that (for example) LHC data reconstruction is not well suited to run on HPCs, while MC reconstruction which uses a limited subset of conditions that might be more easily distributed could run well on HPCs.
- Workflows should be able to take advantage of modern hardware, including accelerators. Many modern HPCs integrate accelerators like GPUs. Cost limitations have resulted in inhomogeneous systems in some cases, so that code able to run on traditional CPU resources as well as accelerators is ideal.

Detector simulation for experiments at the Large Hadron Collider (LHC) [7] is a workload ideally suited to run on these HPCs, as it is able to satisfy all of these constraints. Several groups are actively working on ways to use accelerators, such as GPUs, within simulation workloads [8, 9], including fast simulation [10]. Any use of machine learning in simulation can freely incorporate accelerators thanks to tools like TensorFlow [11]. Scheduling frameworks like Ray [12] have already been adopted for the purpose of distributing detector simulation work over a large HPC [13].

High-performance computers generally have high-speed interconnects between their computing nodes, an expensive feature ideal for large-scale n-body simulations [14–16], material simulations [17], and other problems that are easily distributed over many cores. LHC simulation tends to run in an 'embarrassingly parallel' event-per-thread mode, which inherently does not make use of these interconnects. Some inhomogeneous HPCs might have GPUs available on only a fraction of nodes, so that the distribution of work could be re-imagined; instead of sending one event to one computing core, different parts of an event (or even individual heavy computations) could be scheduled to run on hardware where they are most efficient. As long as simulations retain the concept that all particles are independent of one another, there would be no loss in precision from such an approach. This could have the potential side-benefit of having all the data a calculation requires residence in a low-level cache. There are latencies associated with moving data between nodes, and with the serialization and offload of data to a GPU which could in some cases become substantial. These overheads can be mitigated somewhat with a sufficiently large workload – that is, with a large enough volume of work that no resources are left idle.

One particularly data-intensive aspect of current LHC simulation is the modeling of "pile-up", the background of relatively low-energy interactions in addition to the one collision of interest. In current LHC conditions this means overlaying about 50 additional interactions for every one of interest; during the HL-LHC [18] that number will increase to 200, and in some FCC-hh [19] scenarios it increases further to 1000. Simulating each of these interactions is generally 5-10 times less expensive than simulating one of the "interesting" interactions. However, once 50 must be simulated this dominates the total event simulation time. Historically, therefore, the experiments have modeled pile-up by overlaying pre-simulated interactions from a library [20, 21]. This has been done either by overlaying individual interactions one

* zlmarshall@lbl.gov
† mihamuskinja@lbl.gov
‡ vrpascuzzi@lbl.gov
§ vtsulaia@lbl.gov

at a time or by overlaying a pre-constructed set of background events (e.g. 50 at a time). These libraries of pile-up events represent a significant data distribution challenge, particularly because many simultaneous simulation processes on an HPC might attempt to pull background events from the same source. On standard Worldwide LHC Computing Grid [22] resources, accessing these data causes significant wear on the local disks. On an HPC system, therefore, one of several alternative solutions might be preferable. If the simulation is sufficiently fast (e.g. for a fast simulation) and data movement is sufficiently expensive, simulating these additional events may again become practical. This could include the possibility of using a faster flavor of simulation for the pile-up events than for the collision of interest. Alternatively, the library of background events could be held in memory on a small number of nodes that are used specifically to model pile-up, either serving data to the processing nodes or executing the pile-up modeling algorithms locally and returning results to the simulation nodes.

The question then arises: Given the latencies associated with pushing data between nodes, serialization, and offload to a GPU, are there any rules that can be derived to determine when such a re-imagining of the distribution of simulation work in an HPC would provide some real benefit? If not, future HPCs hoping to support workflows like LHC simulation may be able to save significant costs. If so, can modern scheduling software adapt to these latency issues to efficiently make use of GPUs when available, but continue to run algorithms on CPUs if the latency would be too large?

---

[1] "Perlmutter: Nersc's next supercomputer," https://www.nersc.gov/systems/perlmutter.
[2] "Cygnus supercomputer," https://www.ccs.tsukuba.ac.jp/eng/supercomputers/.
[3] "Marenostrum supercomputer," https://www.bsc.es/marenostrum/marenostrum.
[4] "Taihu supercomputer," http://en.jitri.org/yanjiuyuan75.html.
[5] "Aris supercomputer," https://hpc.grnet.gr/.
[6] "Pratyush supercomputer," http://pratyush.tropmet.res.in/.
[7] L. Evans and P. Bryant, Journal of Instrumentation **3**, S08001 (2008).
[8] S. Blyth, EPJ Web Conf. **214**, 02027 (2019).
[9] T. Evans *et al.*, "Celeritas - a nascent gpu detector simulation code," (2020 (accessed August 31, 2020)).
[10] C. Leggett *et al.*, "Evaluation of performant portable solutions for computing on heterogeneous architectures using atlas fast calorimeter simulation," (2020 (accessed August 31, 2020)).
[11] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.
[12] P. Moritz *et al.*, in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18 (USENIX Association, USA, 2018) p. 561–577.
[13] M. Muskinja, P. Calafiura, C. Leggett, I. Shapoval, and V. Tsulaia, *Raythena: a vertically integrated scheduler for ATLAS applications on heterogeneous distributed resources*, Tech. Rep. ATL-COM-SOFT-2020-026 (CERN, Geneva, 2020).
[14] T. Ishiyama, K. Nitadori, and J. Makino, in *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (2012) pp. 1–10.
[15] M. Baldi, Monthly Notices of the Royal Astronomical Society **422**, 1028 (2012), https://academic.oup.com/mnras/article-pdf/422/2/1028/3451057/mnras0422-1028.pdf.
[16] P. Jetley, L. Wesolowski, F. Gioachin, L. V. Kalé, and T. R. Quinn, in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (2010) pp. 1–11.
[17] F. Xu, R. Cai, Q. Zeng, C. Zou, D. Wu, F. Li, X. Lu, Y. Liang, and R. Fu, J. Mater. Chem. **21**, 1970 (2011).
[18] G. Apollinari, A. I. Béjar, O. Brüning, P. Fessia, M. Lamont, L. Rossi, and L. Tavian, *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1*, CERN Yellow Reports: Monographs (CERN, Geneva, 2017).
[19] A. Abada *et al.* (FCC), Eur. Phys. J. ST **228**, 755 (2019).
[20] T. Novak (ATLAS Collaboration), (2017).
[21] T. Novak (ATLAS Collaboration), (2018).
[22] "Worldwide lhc computing grid," https://wlcg.web.cern.ch/.