Analysis Systems: From Future Facilities to Final Plots

2-3 November 2019

The future of HEP data analysis will need to mine mountains of data gathered from ever more complicated experiments. How to retrieve the pebble of data that each analysis needs from that mountain is a question that the wider data analysis community is also trying to answer. In the meantime the hardware landscape is changing quickly and that evolution will continue, heterogeneity of resources will be the new standard and efficient usage of resources is an expectation of funding bodies. Already in Run3 ALICE and LHCb are having to find ways to adapt to huge rate increases. Then at HL-LHC ATLAS and CMS will undergo a similar step-change. How can the HEP community expend a reasonable and sustainable amount of effort on provisioning facilities and software that runs on it in such a challenging landscape?

Analysis Today

For a student joining an LHC or any other HEP experiment today, one of the first things they need to do is sit through a tutorial explaining their experiment's custom analysis software. While there are some commonalities, not least the prevalence of ROOT, the second thing that the student will often learn is that there isn't really a well-defined way to do analysis, but a rather bewildering array of analysis framework choices built on C++ and Python, integrating metadata somehow. Most frameworks mix the logical definition of what is required with the implementation of how it should be extracted from data, often relying on an explicit knowledge of the underlying data structures. This presents itself as a complex solution to the student who will either use copy and paste if they're task-driven and want results, or start on their own, similar framework from scratch if they're process-oriented. After a little natural selection, we're left with several frameworks, none of which can be easily ported to new hardware and are very experiment specific.

Declarative Analysis

A new approach is also a rather old approach - declarative analysis. Strongly favoured in big data science, declarative analysis techniques focus on being explicit about what is wanted without defining exactly how the result is obtained. In principle this paradigm allows several different "how"s to be implemented, each optimised for different use cases and/or hardware. HEP data

usually has more complicated topologies than big data, at least until the very final stages of analysis, making the direct applicability of big data solutions problematic. Recent innovations both within ROOT and using Python have closed this gap. With more emphasis on declaring what the output should be, the use of intermediate datasets potentially becomes an implementation issue rather than a fundamental part of the analysis model.

Machine Learning

Machine Learning (ML) algorithms are changing daily life (e.g. Siri, Amazon, Google), so they are pervading HEP, providing ample opportunity for improvement if used effectively. However, in contrast to traditional physics algorithms, ML algorithms need careful training and require different software strategies and resource allocation to the historical computing model - training can be very long and inference extremely fast. Fortunately, many of the tools that have enabled ML's widespread adoption in industry are freely available to scientists, although these state-of-the-art ML toolkits sometimes interface awkwardly with HEP frameworks and analysis code. Another challenge is that the "embarrassingly parallel" CPU model of batch computing caters poorly for certain ML workflows, GPU acceleration being one case and high throughput conditional job scheduling being another. Finally, we may need to consider the fundamental issue of training: how can we ensure that physicists do it correctly and achieve optimal results?

Facilities

Up until now, the HEP community has largely enjoyed being able to specify what computing hardware it would like, and how it would like to be arranged, with direct requests to funding bodies leading to heterogeneous centres optimised for our high throughput computing needs. However, computer processing architectures are changing. The processing power of a single core has plateaued, processing power is increased by adding more processing units (that has direct implications for the budget of available memory per process). Hardware acceleration promises to perform mathematics-heavy calculations intelligently and extremely efficiently, so long as they can be fed with input data quickly enough. Finally, funding bodies no longer see HEP as having extraordinary computing requirements, computing is computing and if a big computing centre can provide the number of FLOPs required, the HEP community had better know how to use it… efficiently!

Challenges

- Facilities
    - How should facilities be organised for the HL-LHC?
    - Should there be dedicated facilities for analysis?
    - Should there be a costing model for analysis?
    - Are there other dedicated facilities needed (e.g. ML training)?
    - If we need dedicated facilities, how will we keep funding bodies happy?

- Machine Learning
    - What is the scale of training and who does it?
    - What are the requirements for training (interactivity, visualisation)?
    - Does training need to be integrated with analysis?
    - How do we bridge between training, inference and analysis?

- Analysis Model
    - How can we bridge between interactive and full/final analysis?
    - How can we bridge between heavy lifting and making the final plots?
    - What is the role of interactive analysis? How much data will be needed - from the laptop to the facility scale? What level of latency is tolerable?
    - Do we need intermediate datasets?
    - Are containers enough for analysis preservation?
    - What about metadata?