A) Compute Architectures, platforms and software performance

Panel: Jeff Hammond (Intel), Tom Gibbs (NVIDIA), Chris Jones (FNAL), G.Stewart (U.Glasgow), G.Eulisse (CERN Alice), Pere Mato (CERN SFT)
Moderator: David Abdurachmanov (U.Nebraska-Lincoln)

Panel questions:

1. In your view, considering future hardware product announcements from Intel, Nvidia, their partners and other companies, what are the most important trends for the next 5 years and how will those change software development?

2. Software engineering follows computing architectures evolution and optimizations. What best practices could you suggest for HEP community to follow and what should we avoid? Note that we [HEP community] develop software that will be used for a long period, e.g a decade or more.

3. In your view what computing languages and computing paradigms we [HEP community] are not benefitting from that we should be looking at from the standpoint of performing more work on the same amount of physical hardware?

4. Common software developments: examples and discussions on the HEP community needs. How can industry help HEP?

5. How do we grow software development skills in HEP community and how we can retain expertise?

Live minutes:

Potential institute themes/efforts that come out of this panel:

- Education of developers (potentially with software carpentry?)
- Discovery of and promotion of coding best practices (work with vendors?)
-

Jeff (Intel): Moore's law continues and will continue meaning a die will have more and more transistors. But the power budget is limited meaning not all transistors can be powered at 3 Ghz at the same time. Open the possibility for specialization: "dark silicon" can be used for very specific applications, being powered only when needed (the other part of the processor being temporarily switched off).
Tom (Nvidia): Moore's law easy ride is over. Performance requires code optimization and this cannot be done by every postdoc. Need to rely on optimized libraries.
Jeff: moving from coarse grain parallelisation (MPI) to fine grained (vectorisation,GPUs…), properly handling the memory hierarchy is now a requirement for SW performance.

Recommended SW engineering practices

- Jeff: rely on “standardized” languages, isolate through layering/libraries the scientific apps from the performance optimizations
- C. Jones (FNAL): educate developers that global state is a bad thing
- Graeme: our current code is too complex, both because the processes are complex (like reconstruction) and because of the sociology (a few experts and a long tail of contributors whose contributions are needed). We need to find a way to make our code simpler.
- Jeff: we need new tools that combine the productivity of tools like Python and MatLab with the performance of HPC languages like Fortran. Julia is a good example with its JIT compilation approach (using LLVM JIT compiler), verified on several concrete HPC-friendly use cases.
- Jeff: constraining people about their programming practice is not efficient: constrained people starts a counter culture (or do crazy things to work around standard practices that were imposed to them)! We need, through appropriate tools, to let people be creative and “guide them” so that the resulting code is efficient.

Gaining a 10x in performance

- Jeff/Tom: there is room for a 10x improvement of software performance with the improvements/features on roadmaps. But this will require relying on highly optimized low-level libraries. Consolidation required for having highly efficient low-level libraries: fragmentation of the limited low-level expertise is an obstacle. Example of deep learning: basically only one kernel, optimizing it for every architecture is easier.
- Graeme: there is unfortunately no magic for a major improvement of performance in things like tracking for example. Already a lot of optimization done between run1 and Run2. Looking at a revolutionary way of doing tracking that will dramatically improve performance but we have not found the solution yet! At the end we are constrained by the requirement of maintaining physics performance!
- Jeff: take into account end-to-end optimization. Sometimes concentrating on the compute optimization is not the right approach. It may happen that the bottleneck is somewhere else: network, storage…