Aurora / A21 ESP Status

"Simulating and Learning in the ATLAS Detector at Exascale"

Charles Leggett
D.Benjamin, P. Calafiura, T. Childers, P. van Gemmeren, W. Hopkins, **J. Proudfoot,** V. Tsulaia

US ATLAS HPC Meeting
September 26 2019

ESP Goal

- "This project will enable new physics discoveries by developing exascale workflows and algorithms that meet the growing computing, simulation and analysis needs of the ATLAS experiment at CERN’s Large Hadron Collider Update RR in EDMS."

- Work items explicitly addressed:
  - Workflow and cache coherence
  - Theory simulation using Madgraph
  - GPU Offloading

Milestones

- 1. 2019/07/31: Characterize the work required to modify the ATHENA framework to support an algorithm able to run on GRID and efficiently on an HPC (such as through vectorization and off-loading to accelerators)
- 2. 2019/07/31: Identify pre-existing algorithms in the ATLAS software code base, suitable for running on an HPC machine (we are aware of a number of such potential algorithms)
- 3. 2019/09/31: Prepare a draft developer document for discussion with ATLAS developers which describes the approach to be used to convert algorithms in the ATLAS codes base from GRID-style to HPC-style (we will be searching for generic solutions applicable to A21 but not specific to the RSNDA).
- 4. 2020/01/06: Initial deployment on THETA of ATHENA framework suitable for the development and testing of algorithms developed for HPC
- 5. 2020/03/31: Deployment and testing of HPC-style ATLAS/ATHENA algorithm on JLSE, testing for example off-loading
- 6. 2020/03/31: Deployment and testing of HPC-style ATLAS/ATHENA algorithm on GRID computing, testing for example the ATLAS validation procedure

Workflow and Cache Coherency

- ATLAS reconstruction is roughly 5x slower on Theta than a typical GRID machine
  - similarly for KNL nodes on Cori
- Suspicion: ATLAS reco has huge binaries. Possibly too large to fit in caches, especially when many concurrent events are being executed. Xeon Phi / KNL processors have relatively small caches.
  - If we segregate Algorithms by type to different cores, so that caches are not being constantly overwritten / refreshed, can we increase performance?
- For full scale implementation, this would require a significant redesign of framework to pin Algorithms to different cores and distribute data to them
  - we can test hypothesis by only reconstructing a very small part of the event, with just a few Algorithms (eg CaloCellMaker)
- Result: no significant improvement observed.
  - possibly even this Algorithm is too large.

Madgraph on GPU

- There exists an old (2013) version of Madgraph that was ported to GPUs
  - the GPU version was never fully integrated: no CLI integration, no event generation
  - needs external programs: gVEGAS, gBASES, gSPRING

- Resurrected it, and brought it into the modern era
  - cuda 4 → cuda 9
  - updated gVEGAS
    - gVEGAS could also be used by Sherpa and other generators

- Next steps:
  - udate gSPRING and gBASES
  - benchmark performance

GPU Offloading

- A GPU enabled version of the standalone FastCaloSim was recently developed by BNL
  - uses CUDA to target NVidia GPUs
- This offers an excellent testbed to prototype GPU offloading mechanism, and (performant) portability solutions

- Intel is pushing SYCL / llvm for the general purpose compiler and offloading mechanism on Aurora
  - SYCL is single source, based on llvm and OpenCL, and can target multiple backends
    - (Intel) CPU/GPU/FPGA/etc
    - other vendors have SYCL compilers / backends that can target other hardware (NVidia, AMD)
  - Intel wants to push its SYCL extensions into the llvm mainline
    - create an open standard
  - since it's using llvm, it should be ABI compatible with gcc, and can compile our C++ code without changes (in theory)

GPU Offloading

- Attempted to compile FastCaloSim with Intel SYCL and run it on the JLSE IRIS nodes
  - JLSE has limited build tools: gcc 4.8, cmake 2.2, old singularity, etc
  - rebuilt modern versions of compiler stack and build tools
  - built FastCaloSim without issue

  - but can't run due to ROOT build issues
- Can't build full ROOT as not all necessary libraries are installed on JLSE
  - mainly X11
- Can't run in a container as singularity is too old

- Next steps:
  - try to remove unnecessary ROOT libX11 dependencies (ongoing at BNL GPU hackathon)
  - port FastCaloSim CUDA to SYCL
  - get ROOT in a singularity container on IRIS once JLSE IT upgrades singularity
  - get Gaudi running to do Reco simulation w/ Alg CPU/GPUCruncher Offloading benchmarks