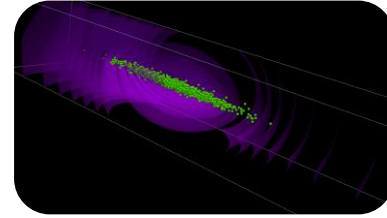# Achieving Perlmutter Readiness for the Synergia Accelerator Modeling Framework with Kokkos

Eric G. Stern, Qiming Lu, Marc Paterno, James Amundson

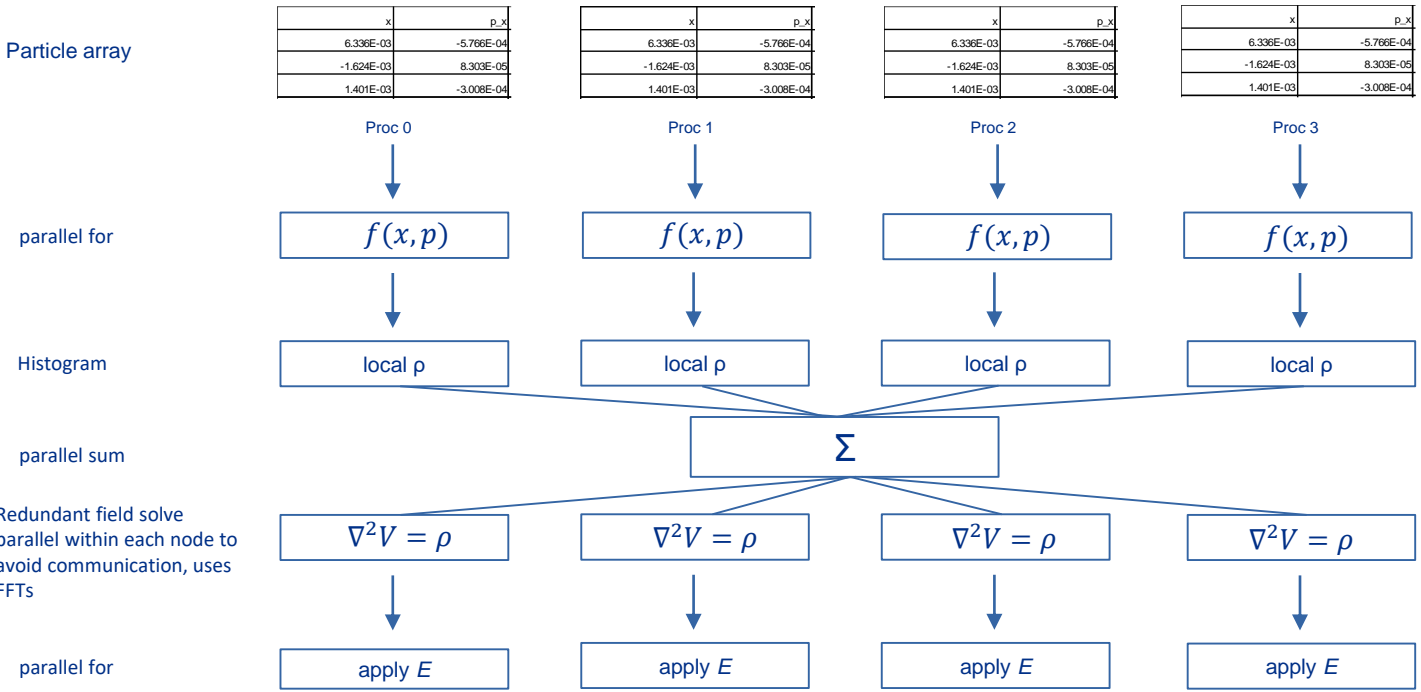NUG2020 Lightning Session

17 Aug 2020

Synergia Modeling Framework

- C++ library with Python wrappers to perform particle-in-cell accelerator simulations.
  - Most simulations are written in Python and import modules to perform the heavy calculation. Main processing loop is in C++.
- Uses MPI/OpenMP parallel processing to scale to large problems.
- Runs on Desktop/laptop, small/medium clusters, supercomputers including NERSC/Cori.
  - Small problems can be run on small systems (number of particles, size of accelerator, etc.)
  - Code scales well for large problems on large systems.

🐭 **Fermilab**

Synergia computational ingredients

Particle array		

x	p_x
6.336E-03	-5.766E-04
-1.624E-03	8.303E-05
1.401E-03	-3.008E-04

x	p_x
6.336E-03	-5.766E-04
-1.624E-03	8.303E-05
1.401E-03	-3.008E-04

x	p_x
6.336E-03	-5.766E-04
-1.624E-03	8.303E-05
1.401E-03	-3.008E-04

x	p_x
6.336E-03	-5.766E-04
-1.624E-03	8.303E-05
1.401E-03	-3.008E-04

Proc 0   Proc 1   Proc 2   Proc 3

parallel for

$f(x,p)$ $f(x,p)$ $f(x,p)$ $f(x,p)$

Histogram

local ρ   local ρ   local ρ   local ρ

parallel sum

$$\Sigma$$

Redundant field solve parallel within each node to avoid communication, uses FFTs

$\nabla^2 V = \rho$ $\nabla^2 V = \rho$ $\nabla^2 V = \rho$ $\nabla^2 V = \rho$

parallel for

apply E apply E apply E apply E

Fermilab

New systems will not work like the old ones

The bulk of computing power at the new large facilities will be provided by some kind of GPU or other co-processor. Synergia must adapt!

What do we want for an upgraded Synergia?

- Keep broad accessibility across computing platforms.
- Use "standard" languages and programming techniques as much as possible.
- Avoid architecture lock-in for code maintainability and execute-everywhere capability.
- Minimize architecture specific code and algorithms.
 - (a previous CUDA specific version was unmaintainable and rotted into uselessness)

🎇 **Fermilab**

kokkos

- Part of the Exascale Computing Project (`https://kokkos.org`).
- Supports both regular CPUs and CPUs with attached co-processors (GPU, etc.) employing back-ends, e.g. OpenMP or CUDA.
- Hardware agnostic: supports NVIDIA (now), AMD and Intel GPUs (promised)
  – Perlmutter nodes will use the NVIDIA A100 GPU
- Templated C++ library maintained as an open-source repository on Github.
- Abstracts array data as a *View* that may reside either in host or device memory.
- Application invokes parallel dispatchers that connect to the device specific backend to implement parallel operations.
- For Synergia, after initial setup that requires CPU resources, data resides in device memory.

Fermilab

Benchmark accelerator simulation results
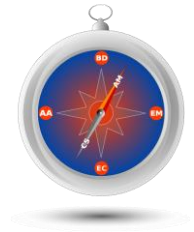
After about a year…

- – Comparison between PowerPC + one
  or four 5120 thread NVIDIA V100 GPUs
  (similar to Summit nodes)

  and

- – 1 or 8 AMD 32 core Opteron nodes

- – Perlmutter will have next generation
  A100 GPUs

iterations/second

🧲 Fermilab
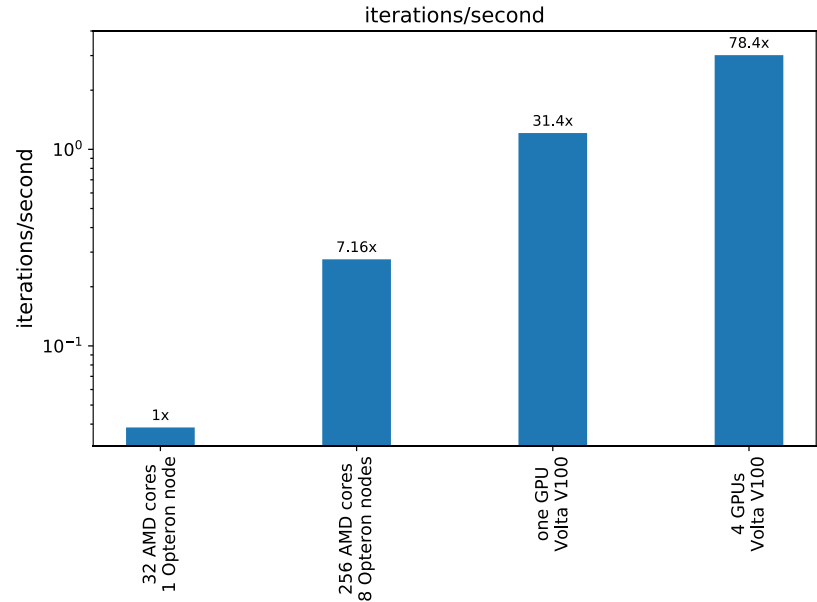
Caveats

- It took a year of work.
- Some algorithms and operations would not operate efficiently on attached processor memory and had to be redesigned for the GPU.
- Some device-specific code was necessary:
  - We had to rely on a CUDA specific FFTW implementation for GPUs.
  - The Kokkos primitive used for charge deposition is not efficient on CPUs.

🎇 **Fermilab**

We are ready for Perlmutter!!

🐝 **Fermilab**