Community Petascale Project for Accelerator Science and Simulation (ComPASS)
&
Synergia

Qiming Lu, Fermilab

A) Project Overview

- Community Petascale Project for Accelerator Science and Simulation (ComPASS)
  - Develop a comprehensive computational infrastructure for accelerator modeling and optimization
  - Advance accelerator computational capabilities from the terascale to the petascale to support DOE priorities for the next decade and beyond
  - Components for beam dynamics, electromagnetics, electron cooling, and advanced accelerator modeling, etc.

- The participants, description of team
  - ANL, BNL, FNAL, LBNL, SLAC, Tech-X, TJNAL, UCLA, U Maryland, USC

B) Science Lesson

- What does the application do, and how?
  - Restrict myself on my sub-project of beam dynamics simulation code called ***Synergia***
  - Single-particle dynamics
  - Multi-particle dynamics where the particle-particle interactions are important
    - Space charge: interaction of the beam with itself
    - Beam-beam interaction
    - Electron cloud
    - Wakefields: fields in accelerator components generated by passing beam
  - Parallel, 3D space charge Particle-in-cell (PIC) code with circular machine modeling capabilities

C) Parallel Programming Model

- MPI, OpenMP, Hybrid, Pthreads, etc
  - Plain MPI parallelization
  - (Particle manager) distribute particles among processors
  - (Poisson-Vlasov) distribute grid across all processors
    - communication avoidance scheme with multiple redundant grids

- Languages
  - C++ with Python wrapper

- Runtime libraries, build requirements
  - Boost, FFTW, HDF5, CHEF(single-particle accelerator physics library), etc.
  - Portable and cross-platform build system based on CMake

C) Parallel Programming Model

- What platforms does the application currently run on?
  - Linux desktops (with or without GPU)
  - Linux clusters (with or without GPUs)
  - BG/P and BG/Q (ALCF)
  - Cray XE6 (NERSC)

- Current status & future plans for the programming model
  - MPI-OpenMP hybrid for multi-core architectures
  - Single GPU / Multi-GPU / GPU cluster

D) Computational Methods

- What algorithms and math libraries do you use? (PDE, FFT, etc)
  - The fast Poisson-Vlasov solver involves intense FFT calculations

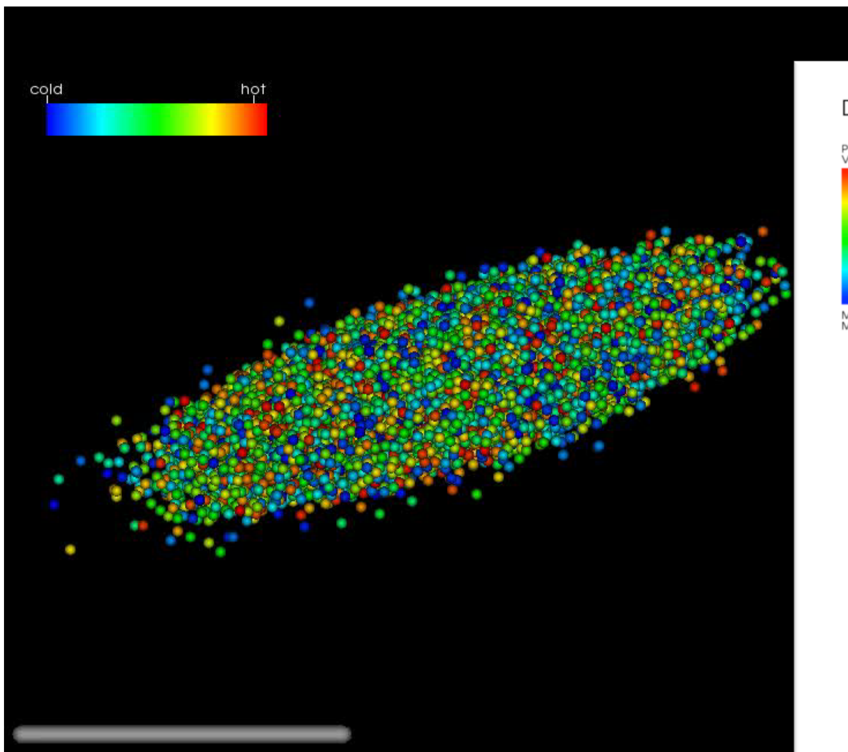
- Current status and future plans for your computation
  - Weak scaling with multiple bunches works very well
  - Strong scaling limited by FFT
    - Pure MPI FFTW performs/scales poorly on multi-core/multi-node platforms when crossing the node boundary
    - MPI/OpenMP hybrid FFTW is even worse
    - Seeking better parallel FFT routines, or implement our own
    - Communication avoidance helps, but does not solve the problem

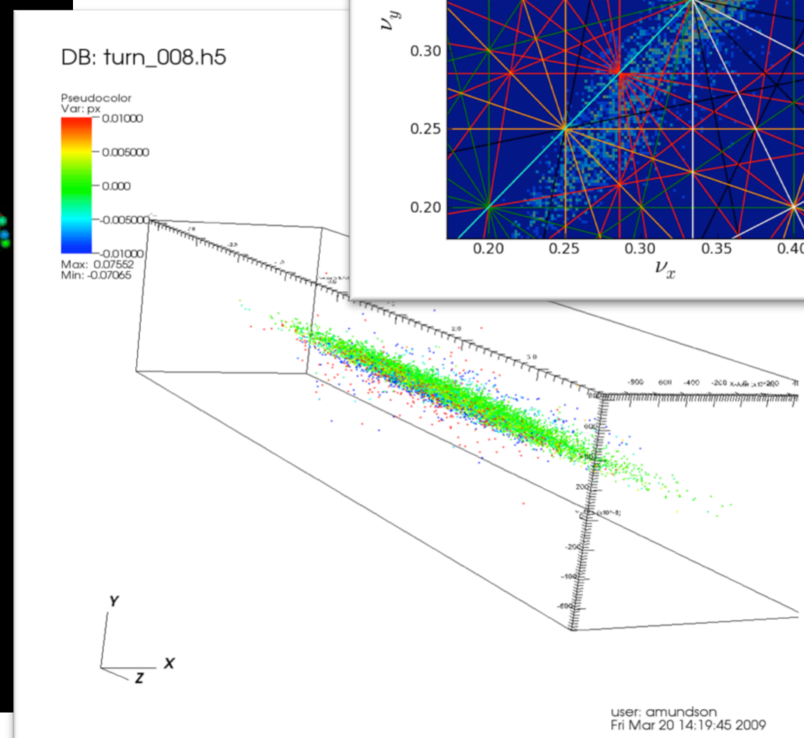
E) I/O Patterns and Strategy

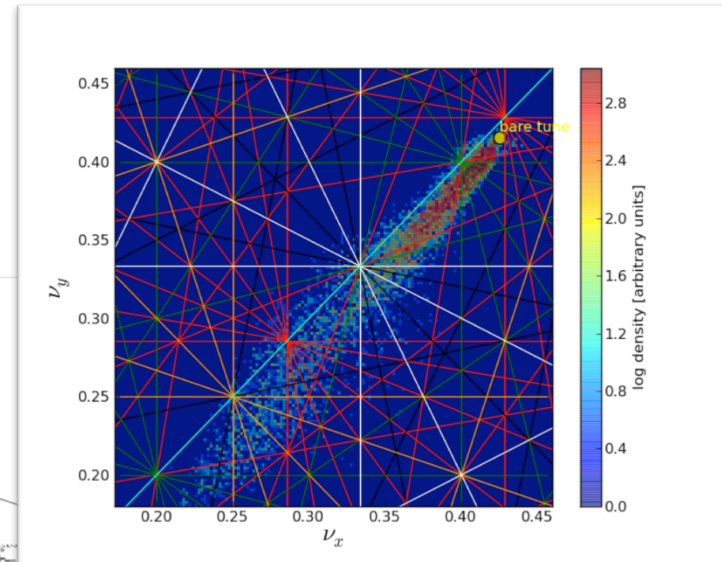
- Input I/O and output I/O patterns (one file per MPI process?, pNetCDF? HDF5?, etc)
  - One file per MPI process
  - Uses HDF5 for storing particle information when available
- Approximate sizes of inputs and outputs (before, during, and after computation)
  - Typical inputs 200KB
  - Typical outputs hundreds of MB to hundreds of GB
- Checkpoint / Restart capabilities: what does it look like?
  - Configurable checkpoint/restart frequency
  - One file per core + one copy of each open output file
- Current status and future plans for I/O
  - Plan to do platform specific optimizations

F) Visualization and Analysis

- How do you explore the data generated?
  - Post-processing of HDF5 data usually involves Python, Matplotlib, and VisIT
    - Scientists can pick their tools
    - Synergia uses VizSchema metadata with VisIT





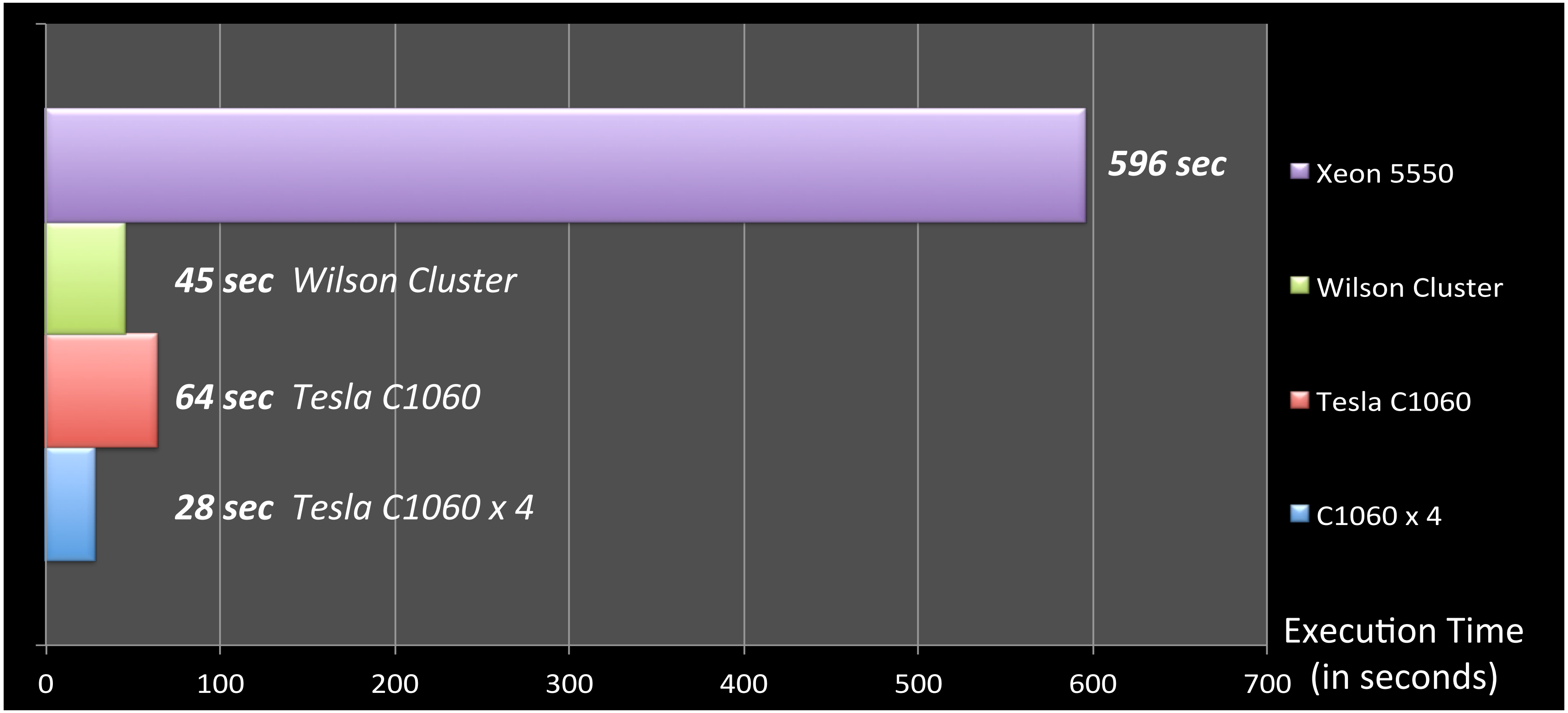


G) Performance

- What tools do you use now to explore performance (Tau, DynInst, PAPI, etc)
  - TAU and a custom simple timer
- What do you believe is your current bottleneck to better performance?
  - parallel FFT for small-to-medium sizes
- What do you believe is your current bottleneck to better scaling?
  - Single particles and independent operations can be scaled almost infinitely
  - The relatively small grid size has limited the strong scaling
  - Global communications, e.g., distribute charge density, distribute fields

G) Performance

- Current status and future plans for improving performance
  - Experimented using single/multi-GPU to accelerate the Synergia simulation



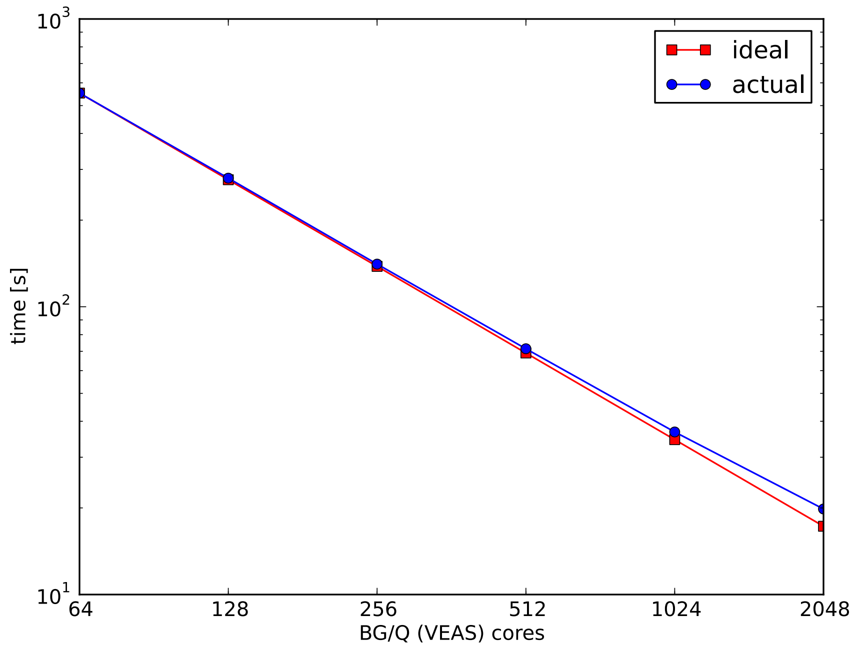
H) Tools

- How do you debug your code?
  - TotalView and std::cerr
- What other tools do you use?
  - CodeAnalyst
- Current status and future plans for improved tool integration and support

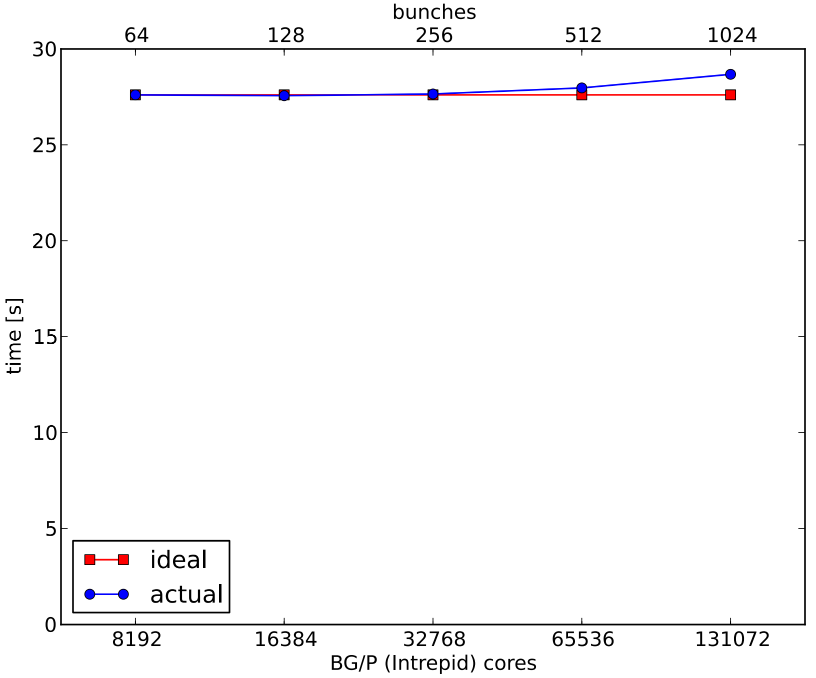
I) Status and Scalability

- Current status and future plans for improving scaling
  - Large problem sizes on large machine

strong scaling



weak scaling



I) Status and Scalability

- Current status and future plans for improving scaling
  - Medium problem sizes on clusters
  - For both large and small problems we would like to be able to run for many time steps so strong scaling is important

- Future plans
  - Better FFTs with multi-core architectures
  - Platform specific optimizations for BG/Q
    - Will build on previous works on GPUs and multi-core