Hybrid MPI/OpenMP/GPU Parallelization of XGC1 Fusion Simulation Code

Achieving excellent weak scaling out to 18,624 GPU-CPU XK7 nodes, enabling science studies that have not been possible before

E. D'Azevedo[1], J. Lang[2], P. Worley[1], S. Ethier[2], S-H Ku[2], C.S. Chang[2] and the EPSi Team

[1]Oak Ridge National Laboratory, [2]Princeton Plasma Physics Laboratory

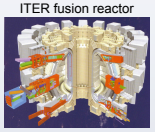
SC13
Denver, Colorado
November 17-22, 2103

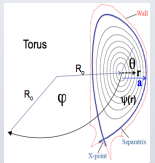
Background

Science:

- Edge simulation (near outer wall) is critical for tokamak-based fusion reactor design because the steady fusion yield is determined by the edge plasma properties.
- Accurate simulation is difficult and requires effective exploitation of leadership computing systems:
  - Edge plasma is in contact with wall and density has a very steep gradient, invalidating fluid approx. and requiring treatment of non-equilibrium physics
  - Complicated geometry, currently addressed using unstructured triangular mesh
  - Multiscale physics: background physics, microturbulence, neutrals and atomic physics

ITER fusion reactor





Poloidal cross-section

Computing System Target:

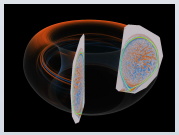
Cray XK7 sited at the Oak Ridge Leadership Computing Facility ('Titan'), consisting of a Gemini interconnect and 18,688 computational nodes. Each node contains

- One 2.2GHz 16-core AMD Opteron 6274 processor (with 8 floating point units, 1 per every two cores)
- One NVIDIA K20 Kepler GPU with 6 GB memory
- 32 GB (non-GPU) memory

XGC1

- Full-function (as opposed to perturbative delta-f) gyrokinetic particle-in-cell code designed for simulating edge plasmas in tokamaks
- Solves 5D gyrokinetic equations via
  - ordinary differential equations for time advance of particles in unstructured triangular physical space grid
  - finite difference discretization of partial integro-differential Fokker-Planck collision operator on rectangular velocity-space grid
  - Maxwell's (partial differential) equations on unstructured triangular physical space grid, solved using PETSc
  - interpolation of particles between physical and velocity space grids
- The whole volume is simulated with a coarse mesh in the core to capture the large scale turbulence interaction between core and edge.

Large scale turbulence in the whole volume simulation in DIII-D geometry

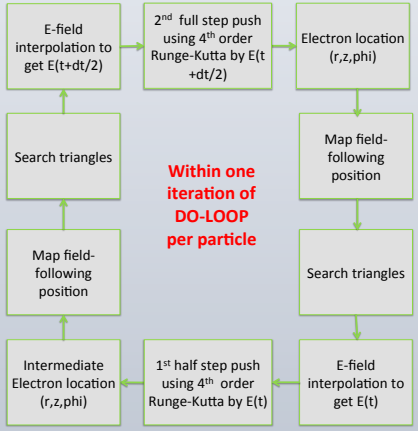


- Key features:
  - diverted magnetic geometry
  - gyrokinetic ions
  - drift-kinetic electrons: small gyroradius limit
  - Monte-Carlo neutral particle with wall-recycling coefficients
  - multi-species impurity particles
  - plasma heating in the core
  - torque input in the core
  - fully non-linear Coulomb collisions
  - logical Debye-sheath: code determines wall sheath from ambipolar loss constraint
  - reads in an experimental geometry and plasma data

Electron Push Computational Kernel

- The target problem in these experiments was a simulation for the DIII-D fusion reactor using a 3mm resolution grid with ~110,000 cells per poloidal-radial plane (fixed toroidal coordinate) and 32 planes. For XGC1 and for this problem it is adequate to use 7000 ions and 7000 electrons per computational grid cell, or ~25 billion particles of each kind.
- Majority of time is spent in particle push subroutines, calculating new particle positions.
- The electron mass is much smaller than the ion mass, and the timestep for each electron push needs to be correspondingly smaller. XGC1 uses an electron sub-cycling method and ~60 electron pushes per ion push. The actual cost of the electron push is even larger than this implies, reaching more than 90% of the computation time in CPU-only runs.

Code structure of electron push (PUSHE):

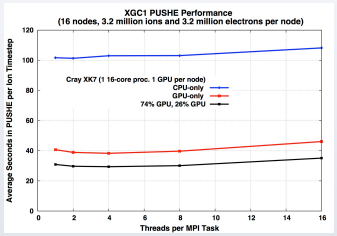


Whole Electron Push Routine Ported to GPU

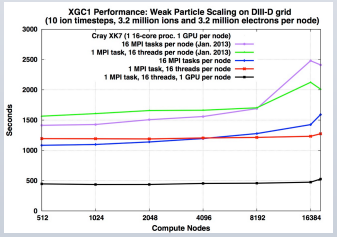
- All routines and module data structures used in CPU-version of PUSHE were extracted into a single Fortran module. PGI CUDA Fortran was then used to generate the GPU kernels.
- A separate stream is used for each MPI task:
  - Call function cudaStreamCreate(stream_id)
  - Use a stream option in kernel launch and data copy
- Only 64 threads per thread block are used, to reduce the amount of register spills to global memory. Total number of threads per MPI process varies from 384*64 (1 MPI process per GPU) to 64*64 (16 MPI processes sharing 1 GPU).
- Since communication between CPU and GPU is slow, all data required by electron sub-cycles is copied into the GPU. After electron subcycling is complete, the updated arrays are copied back to the CPU.
- To work with the PGI compiler –fast option, structured data structures are copied to "flat" arrays before copying to GPU. Indices are also permuted (to improve GPU internal accesses). Cost of all copying is ~5% of PUSHE time.
- Particles are sorted (or renumbered) using geometric hashing into a background rectangular grid so that particles are grouped based on their locality in the rectangular cells, improving PUSHE performance by between 10% and 25%.
- Since the CPU returns to the host program instantly upon launching GPU kernels, the electron push work load can be split between CPU and GPU. In XGC1 code, this split is determined at runtime and performance can be optimized by assigning work proportional to the empirically-observed capabilities of the CPU and GPU.

Performance and Weak Scaling

- Target is DIII-D simulation using 3mm grid and running on 8192 nodes, or 3.2M electrons (ions) per node.



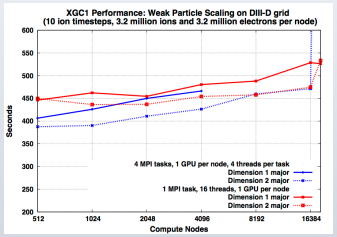
- Average electron push (PUSHE) runtime for main (ion) timestepping loop using 16 nodes and 3.2M electrons per node for different MPI process and OpenMP thread counts. Using the GPU increases performance by a factor of 2.5 compared to using the CPU only. Using both CPU and GPU increases performance by a factor of 3.3. 4-way OpenMP parallelism is optimal in this small case, but 16-way is only < 10% slower.



- Average runtime for main timestepping loop for 3mm DIII-D grid (strong scaling) and 3.2M electrons (ions) per node (weak scaling), comparing Jan. 2013 version of code with current version. GPU acceleration in conjunction with further optimization of MPI and OpenMP parallelism improved performance by a factor of 4 for target 8192 node experiment. Weak scaling in particles is also excellent.

MPI and OpenMP

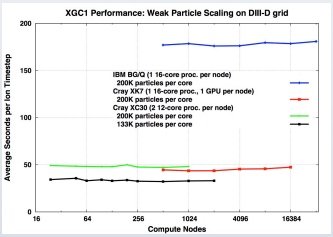
- Load imbalances and MPI communication overheads must be minimized and OpenMP parallelism exploited effectively even in weak scaling studies. GPU acceleration only increases the importance of this.
- OpenMP works very well in XGC1.



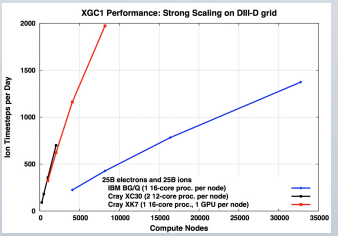
- XGC1 uses a logical 2D processor grid. MPI communication in Maxwell equation solution, charge deposition onto the physical grid, and particle collisions are in one dimension only, and concurrent across the other dimension. MPI communication when moving particles within physical grid and gathering data for copying to the GPU are in the other dimension only. XGC1 supports mapping processes to nodes using major ordering for either dimension. For these runs, there is a consistent preference for "Dimension 2 major".

Performance Comparison Across Platforms

XGC1 has also recently been ported to a 5214-node XC30 system at NERSC (24 cores per node, 125K cores total) and to a 49,152-node IBM BG/Q at the ALCF (16 cores per node, 786K cores total). Existing tuning knobs were used to optimize performance, but as of yet little effort has been made to modify source code to exploit more effectively special architectural features (e.g. "quad hummer" on BG/Q). For BG/Q, experiments did examine using 1, 2, and 4 hyperthreads per core.



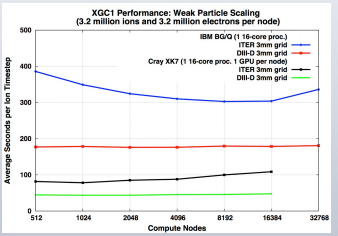
- Weak particle scaling for 3mm DIII-D grid, both 3.2M electrons (ions) per node and 200K electrons (ions) per core. (These are the same on the XK7 and BG/Q, but not on the XC30.) Weak scaling is excellent on all systems.



- Strong particle scaling for 3mm DIII-D grid using 25B electrons (ions) total. Given disparity in node capabilities, strong scaling results are preferred when comparing these systems.

Next Steps

New capabilities introduce new issues. New collision operator (not used in results reported here) can be as expensive as PUSHE. It needs serial and OpenMP optimization, and is a candidate for porting to GPU. Larger grids, e.g. 3mm ITER grid (~1M grid cells per plane), require additional MPI and OpenMP optimization.



Acknowledgements

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Fusion Energy Sciences, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC and No. DE-AC02-09CH11466 with Princeton University.

This work used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, of the Argonne Leadership Computing Facility at Argonne National Laboratory, and of the National Energy Research Scientific Computing Center, which are supported by the Office of Science of the Department of Energy under Contracts DE-AC05-00OR22725, DE-AC02-05CH11231, and DE-AC02-06CH11357, respectively.