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

SC13, Denver, Colorado, November 17-22, 2013

Extended Poster Abstract

Eduardo F. D'Azevedo
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6164
dazevedoef@ornl.gov

Jianying Lang
Princeton Plasma Physics Laboratory
P.O. Box 451
Princeton, NJ 08543-0451
jlang@pppl.gov

Patrick H. Worley
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6173
worleyph@ornl.gov

Stephane A. Ethier
Princeton Plasma Physics Laboratory
P.O. Box 451
Princeton, NJ 08543-0451
ethier@pppl.gov

Seung-Hoe Ku
Princeton Plasma Physics Laboratory
P.O. Box 451
Princeton, NJ 08543-0451
sku@pppl.gov

Choong-Seock Chang
Princeton Plasma Physics Laboratory
P.O. Box 451
Princeton, NJ 08543-0451
cschang@pppl.gov

ABSTRACT

By exploiting MPI, OpenMP, and CUDA Fortran, the FORTRAN fusion simulation code XGC1 achieves excellent weak scalability out to at least 18,624 GPU-CPU XK7 nodes, enabling science studies that have not been possible before.

XGC1 is a full-f gyrokinetic particle-in-cell code designed specifically for simulating edge plasmas in tokamaks. XGC1 was recently ported to and optimized on the 18,688 node Cray XK7 sited in the Oak Ridge Leadership Computing Facility, making use of both the 16-core AMD processor and the NVIDIA Kepler GPU on each node. XGC1 uses MPI for internode and intranode parallelism, OpenMP for intranode parallelism, and CUDA Fortran for implementing key computational kernels on the GPU. XGC1 also uses the CPU and GPU simultaneously for these computational kernels. The optimized version achieves a *four times speed-up* over the original CPU-only version.

1. INTRODUCTION

Effective simulation of burning plasmas is required for the development of commercially viable fusion power. In particular, a prediction capability for the edge plasma, that is the plasma near the outer wall of the reactor, and an improved understanding of the associated physics is critical for tokamak-based fusion reactor design because the steady fusion yield is determined by the properties of the edge plasma state. Although this criticality has been recognized for a long time, understanding the tokamak edge physics remains elusive. The edge plasma is in contact with the wall and the plasma density exhibits a very steep gradient near the wall. The distribution function of the plasma is far from equilibrium (non-Maxwellian), and the fluid approximation is invalid. The geometry of the edge plasma is complicated, especially the existence of the magnetic X-point on the separatrix surfaces and an arbitrary wall shape in the tokamak device. Also, the edge plasma has multiscale physics phenomena, including (small scale) microturbulence, (large scale) background plasma dynamics, nonlinear Coulomb collisions, and atomic physics. Due to the complexities and nonseparable multiscale nature of the physics, accurate simulation requires effective exploitation of leadership computing systems.

XGC1 is a particle-in-cell code designed specifically for simulating edge plasmas in tokamaks. It is a parallel code exploiting both MPI and OpenMP parallelism [2] that in 2010 demonstrated excellent weak and strong scalability out to the full system size (over 220,000 processor cores) of the Cray XT5 sited in the Oak Ridge Leadership Computing Facility (OLCF) [5]. Since then XCG1 has changed significantly. Among the many changes, it now simulates kinetic electrons. Kinetic electrons represent as much as 90% of the computational cost for simulations of interest, increasing the cost by a factor of 10 over simulations without kinetic electrons.

The capabilities of leadership computing systems have also increased over the same period, but much of the recent increase has come from the inclusion of accelerators such as the GPGPU and the Intel Phi. In this poster we outline a (successful) investigation into porting the primary kinetic electron computational kernel to a GPU and then evaluating the performance of XGC1 on a hybrid CPU/GPU system. The changes in XGC1 and in the target architectures also motivated MPI and OpenMP algorithm and implementation optimizations.

The target system for this work is the Cray XK7 sited in the OLCF (Titan), consisting of a Gemini interconnect with a modified 3D torus topology and 18,688 compute 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, and 32 GB (non-GPU) memory.

The target problem in these experiments was a simulation for the DIII-D fusion reactor using a 3mm resolution grid with approximately 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 approximately 25 billion particles of each kind.

2. XGC1

XGC1 [3, 4] is full-function (as opposed to perturbative delta-f) gyrokinetic particle-in-cell code designed for simulating edge plasmas in tokamaks. It solves the 5D gyrokinetic equations via (1) ordinary differential equations for time advance of particles in an unstructured triangular physical space grid, (2) Maxwell's (partial differential) equations on an unstructured triangular physics space grid, solved using PETSc, (3) finite difference discretization of partial integro-differential Fokker-Planck collision operator on a rectangular velocity grid, and (4) interpolation of particles between physical and velocity space grids.

3. ELECTRON PUSH KERNEL

The majority of the computation time is spent in the 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 implementation of the electron subcycling method [1] and approximately 60 electron pushes per ion push. The actual cost of the electron push is even greater than this would imply, reaching as high as 90% of the computation time in CPU-only runs.

To port the electron push kernel (PUSHE) to the GPU, all routines and module data structures used in the 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 was used for each MPI task, calling the function `cudaStreamCreate(stream_id)` and using a stream option in kernel launch and data copy. Only 64 threads per thread block were used, to reduce the amount of register spills to global memory. The total number of threads per MPI process varied from 384*64 (1 MPI process per GPU) to 64*64 (16 MPI processes sharing 1 GPU). Atomic increment intrinsics were used to ensure correct concurrent updating of critical data arrays. No shared memory was used and the L1 cache was configured to be 48 KBytes per thread block.

Since communication between CPU and GPU is slow, all data required by electron subcycles 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 were copied to "flat" arrays before copying to GPU. Indices were also permuted (to improve GPU internal accesses). The cost of all copying is approximately 5% of PUSHE time in these experiments.

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 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.

4. RESULTS AND FUTURE WORK

Detailed performance results are presented in the poster. In summary, using the GPU improved performance of PUSHE by a factor of 3.3. In combination with additional MPI and OpenMP optimizations, full code performance increased by a factor of 4 and excellent weak and strong scalability was demonstrated. New capabilities being introduced into the code will have a similar impact to PUSHE, and future work will evaluate porting this new code to the GPU. Other systems are also being evaluated. Preliminary performance results for a Cray XC30 and an IBM BG/Q are also presented in the poster, and excellent performance scalability was observed on these systems as well.

5. ACKNOWLEDGMENTS

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.

6. REFERENCES

[1] J. C. Adams, A. G. Serveniere, and A. B. Langdon. Electron sub-cycling in particle simulation of plasma. *Journal of Computational Physics*, 47(2):229–244, 1982.

[2] M. F. Adams, S.-H. Ku, P. H. Worley, E. F. D'Azevedo, J. C. Cummings, and C.-S. Chang. Scaling to 150K cores: recent algorithm and performance engineering developments enabling XGC1 to run at scale. *Journal of Physics: Conference Series*, 180(012036), 2009.

[3] C. Chang, S. Ku, P. Diamond, Z. Lin, and et. al. Compressed itg turbulence in diverted tokamak edge. *Physics of Plasma*, 16(056108), 2009.

[4] S. Ku, C. Chang, and P. Diamond. Full-f gyrokinetic particle simulation of itg turbulence with a strong central heating in realistic tokamak geometry. *Nuclear Fusion*, 40(115021), 2009.

[5] P. H. Worley, M. F. Adams, E. F. D'Azevedo, C.-S. Chang, S.-H. Ku, and C. McCurdy. XGC1: Performance on the 8-core and 12-core Cray XT5 systems at Oak Ridge National Laboratory. In R. Winget and K. Winget, editors, *Proceedings of the 52nd Cray User Group Conference*, Eagan, MN, May 2010. Cray User Group, Inc.