PERFORMANCE PORTABILITY OF XGC CODE AT DOE SUPERCOMPUTING FACILITIES

DOE Performance, Portability and Productivity Annual Meeting

Brian MacKie-Mason (Argonne National Laboratory) & XGC Team
April 2-4, 2019 | Denver, CO

ABSTRACT

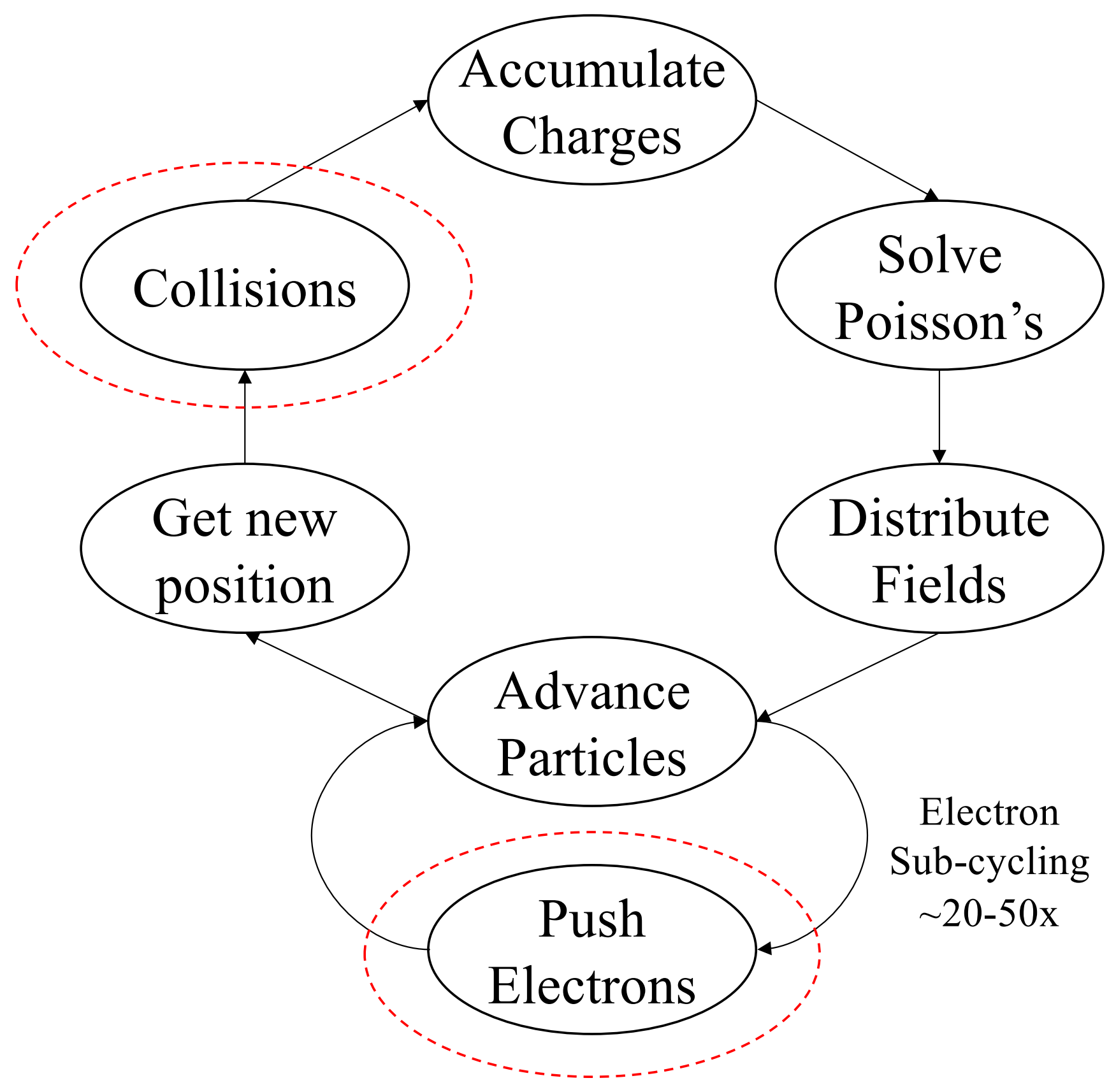
For the successful operation of a nuclear fusion Tokamak reactor, such as ITER, a complete understanding of the underlying physics in the energy loss mechanisms is important. A large source of energy loss occurs in the edge of the plasma near the boundary wall of the reactor. In order to have a good statistical model of the behavior in a particle-in-cell simulation, a large number of simulation particles is required, and therefore the use of exascale computing. In particular, exascale computing will allow for a better understanding of confinement near the plasma boundary as well as the transport of unconfined plasma to the divertor plates. This will be enabled by two key improvements to the underlying algorithms: i) a highly optimized electron push kernel; ii) a multi-(ion-)species collision operator.

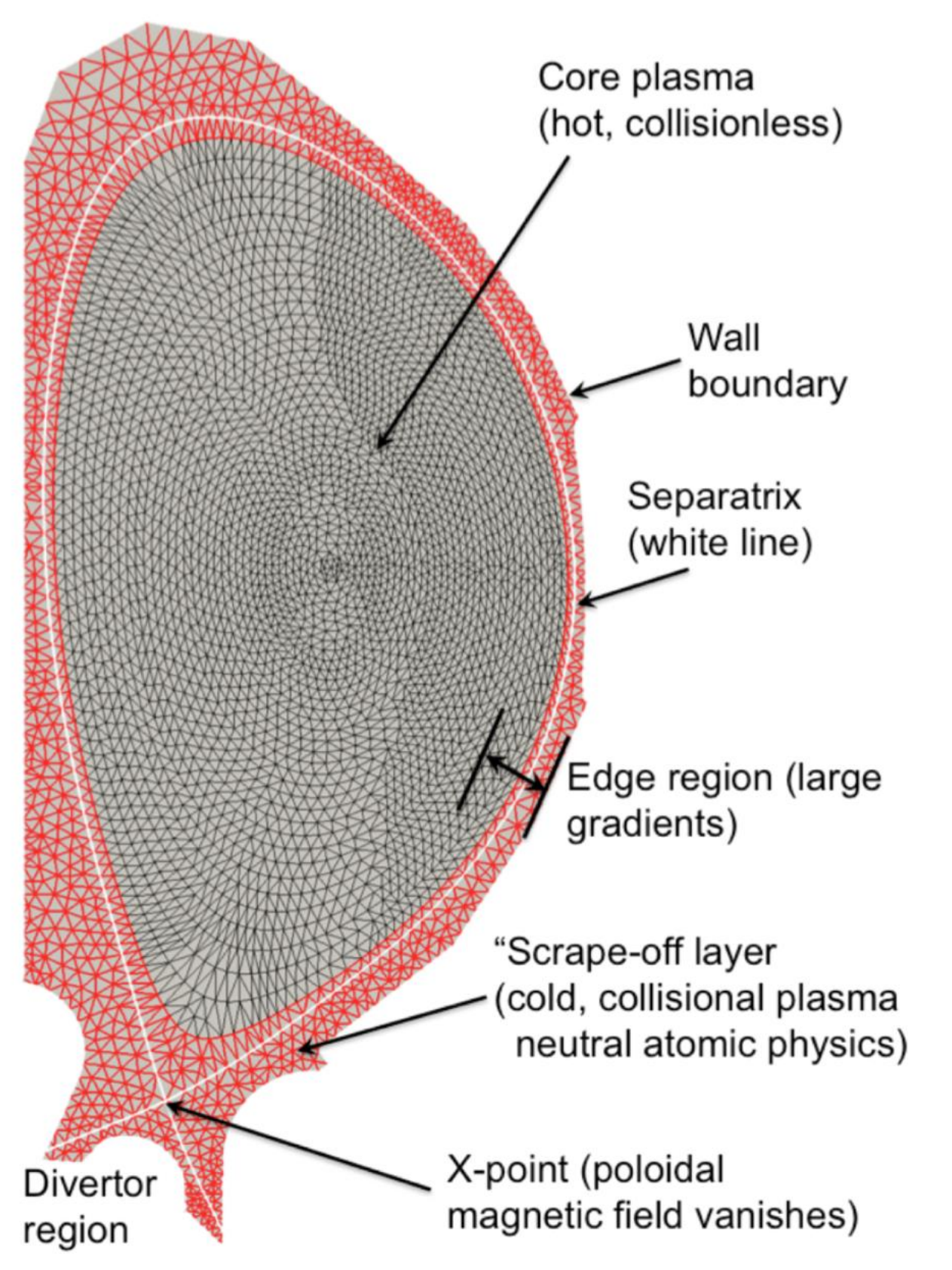
The need for exascale is primarily driven by the inclusion of multiple species, but also by the expensive electron push routine as the number of electrons will increase as the number of ions increases.

In order to have a diversified approach to answering the fundamental questions of nucelar fusion the XGC team has computing time at many different DOE supercomputing facilities whose systems each have their own microarchitecture. Optimizing the code for top performance on these systems has led to a multi-branched development of the code to match the “system of the day”.

XGC OVERVIEW

- X-point included Gyrokinetic Code (XGC) solves the gyrokinetic equations of motion to study the plasma turbulence in the boundary of a Tokamak plasma.
- Key algorithms: Hybrid Lagrangian δ-f scheme, Runge-Kutta hybrid 4th-2nd order, triangle searching, particle sorting, Poisson solve, interpolation routines.
- Performance-critical libraries: PETSc, pspline, ADIOS
- Electron push is ~70-80% of compute time, Poisson, collisions, and charge accumulation make up the remainder.
- Currently only one ion species is studied due to computational limitations. The collision kernel is expected to grow appreciably as ion species are added.
- Implementation details: Fortran 90 + MPI + openMP, CUDA Fortran electron push for GPUs, vector instructions for Intel architecture. OpenACC for collisions on GPUs. Some architecture specific mark ups for further performance optimization.



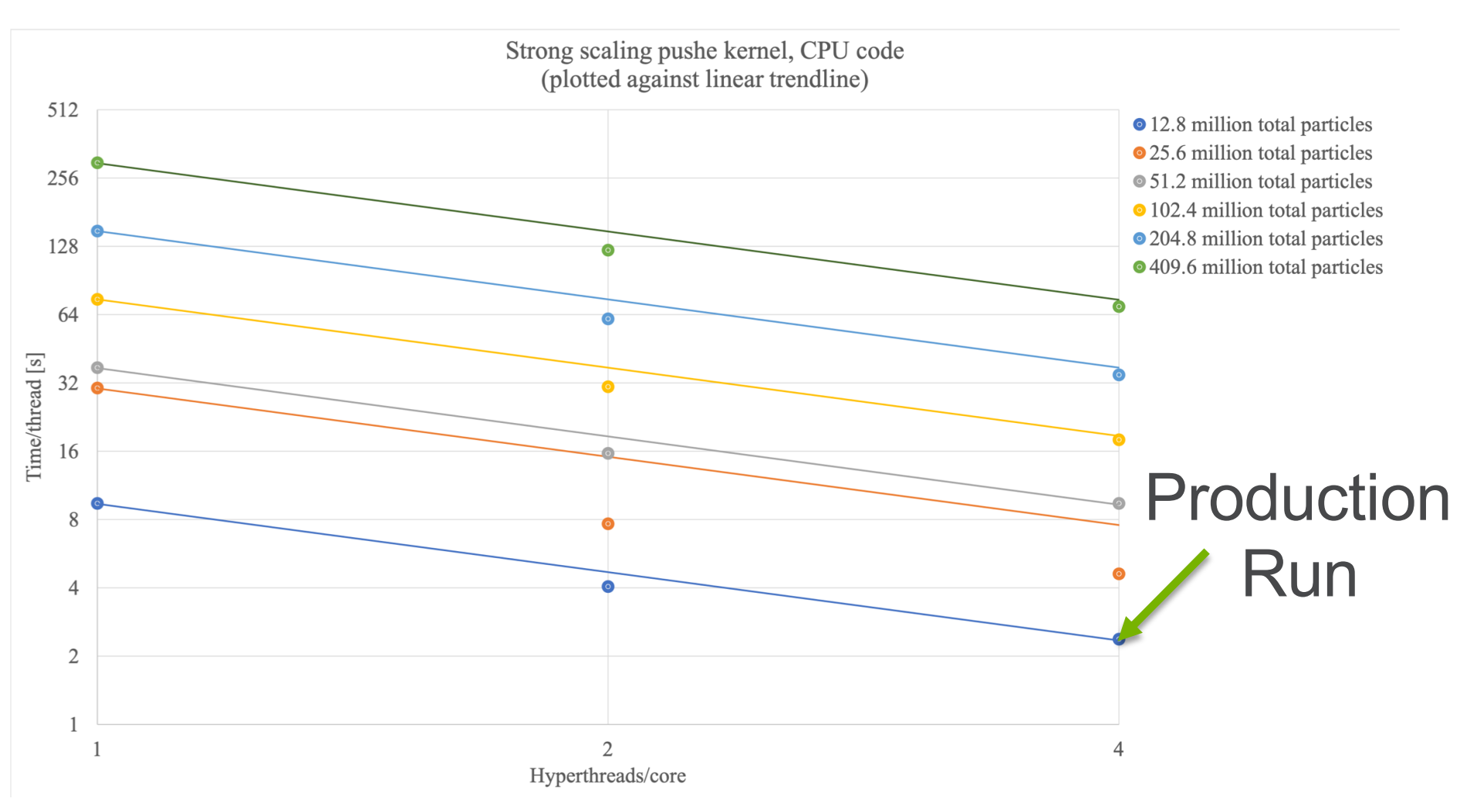

Core plasma (hot, collisionless)
Wall boundary
Separatrix (white line)
Edge region (large gradients)
“Scrape-off layer (cold, collisional plasma neutral atomic physics)
X-point (poloidal magnetic field vanishes)
Divertor region

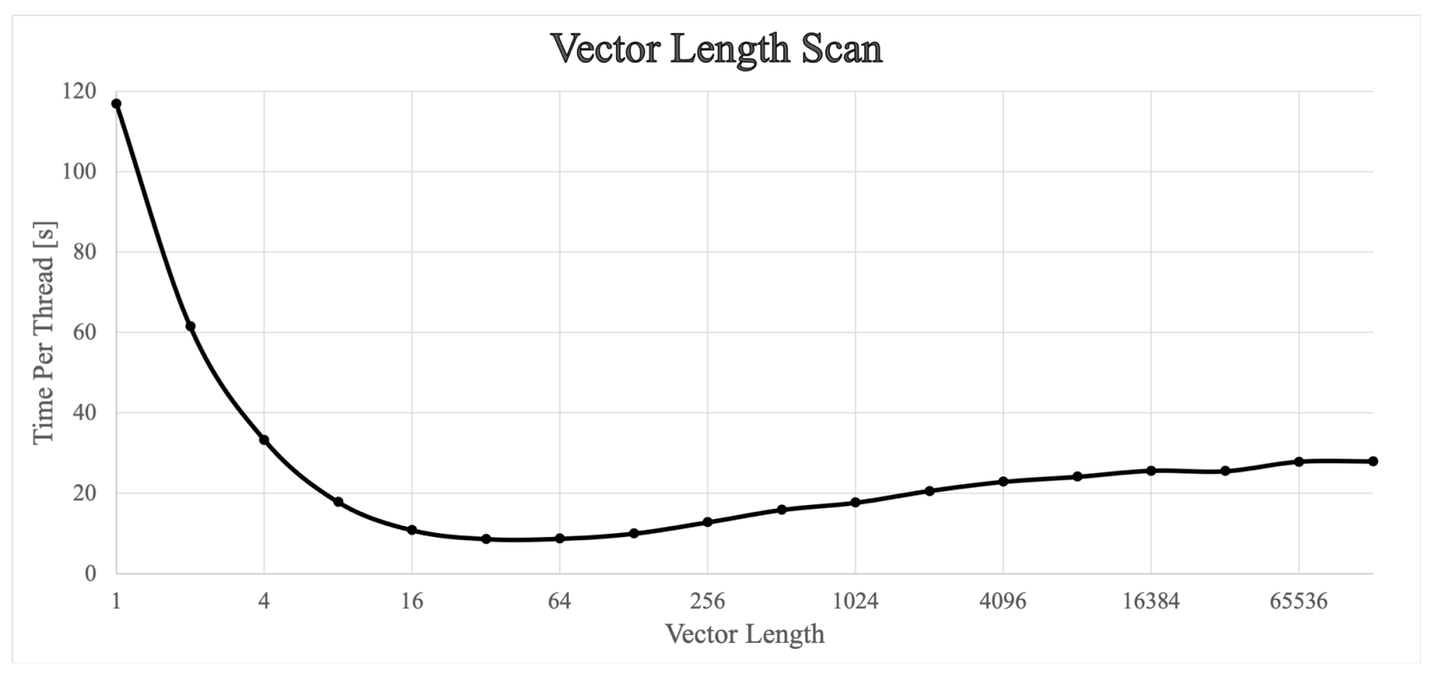
Chang, et. al. “The Fusion Code XGC.” *Exascale Scientific Applications* (2018).

CUF -> OPENMP

- Main challenge posed is CUDA as a proprietary language, and much architecture specific optimization has been performed using CUDA directives.
- Strategy:
  - Build standalone push kernel for GPU based from CPU push kernel
  - Use this as a basis for porting to OpenMP 4.5

KNL PERFORMANCE





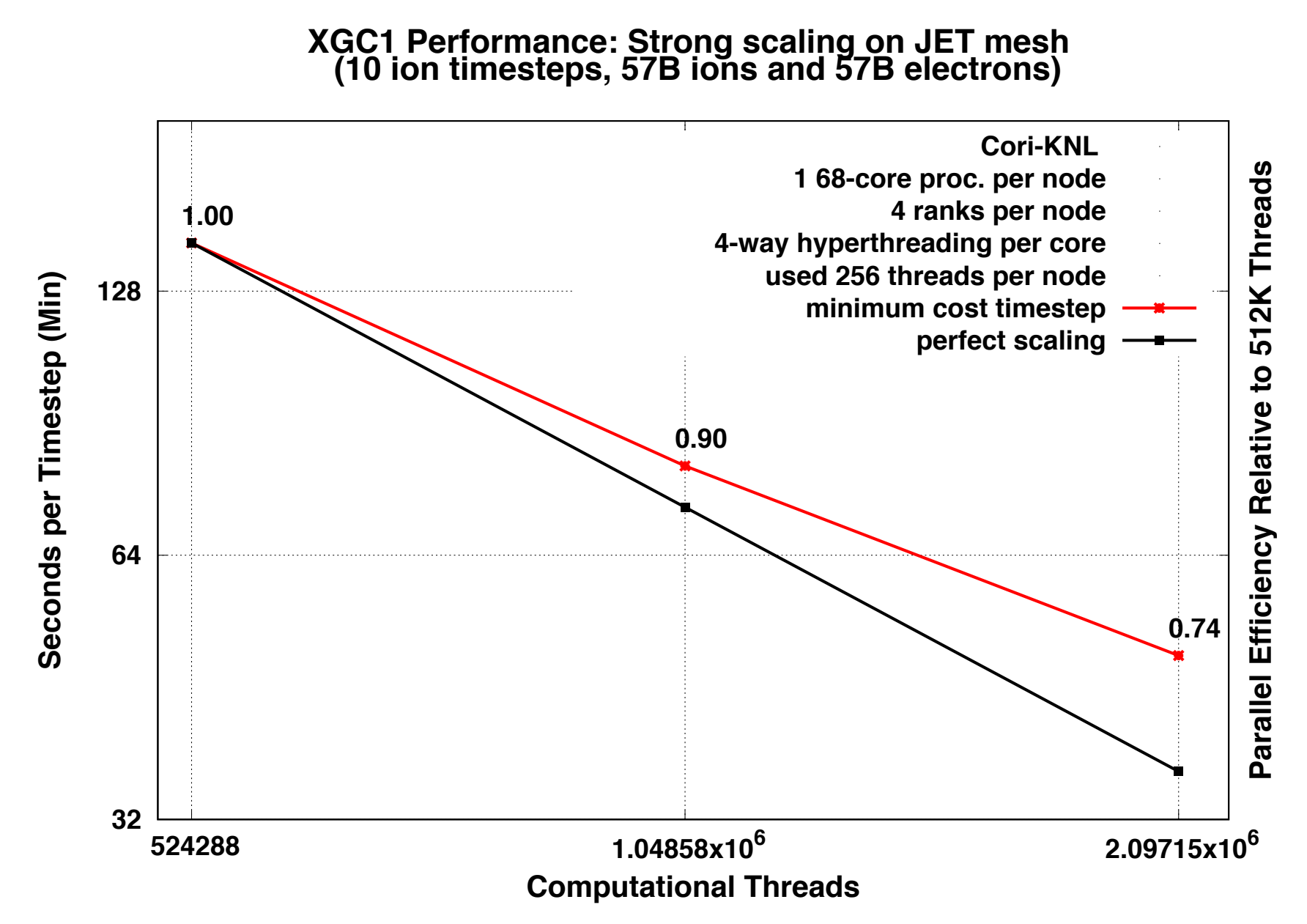
BENCHMARK PROBLEM

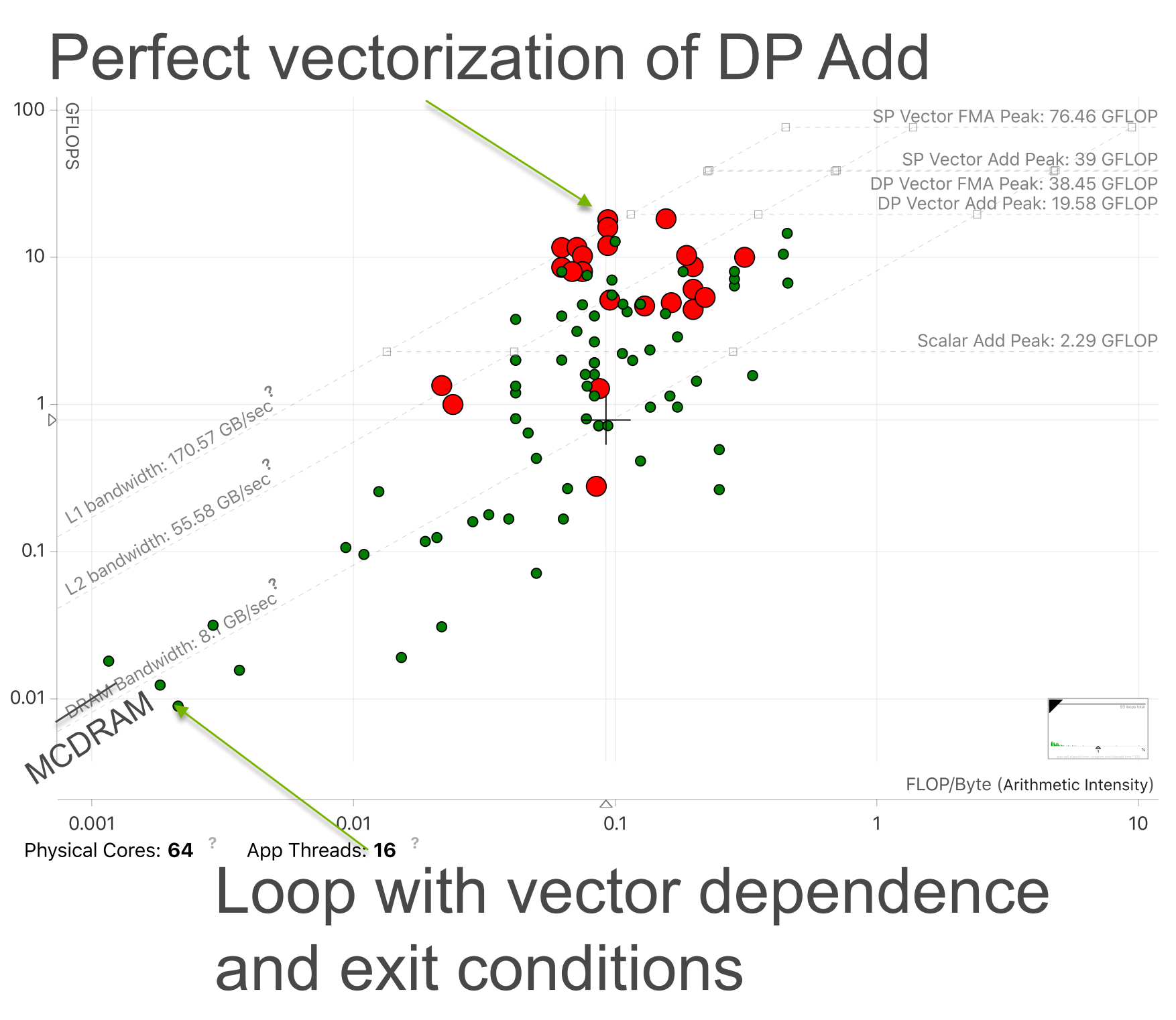
- 1/500th of toroidal transit time, 4mm mesh size
- Meshing: 75k grid points
- Particles: 12.8 million, 800k/rank, 50k/thread
- Tanh equilibrium profile
- R = {1, 2, 4} m; Z = {-0.67, 0.67} m

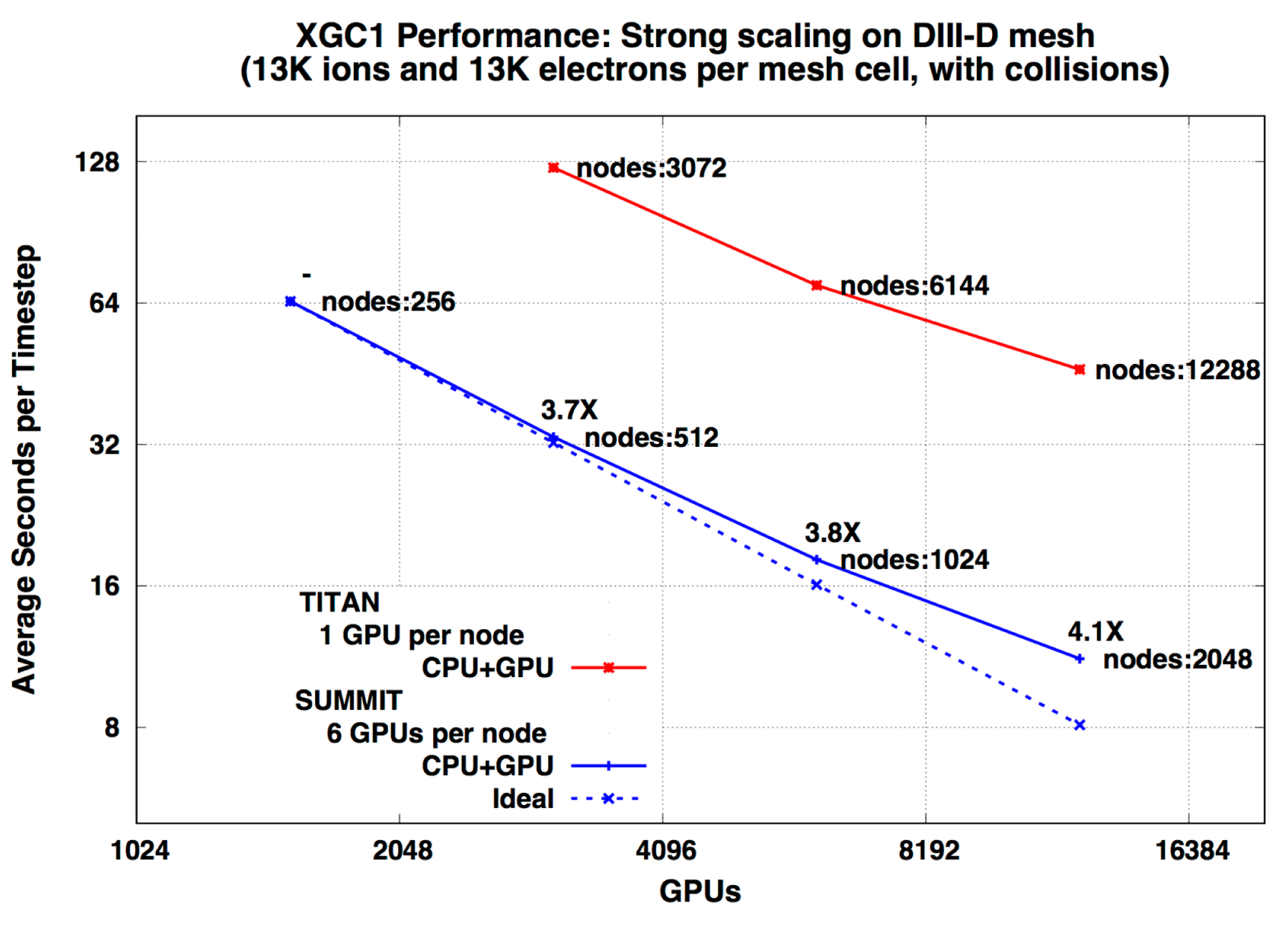
IMPROVEMENTS

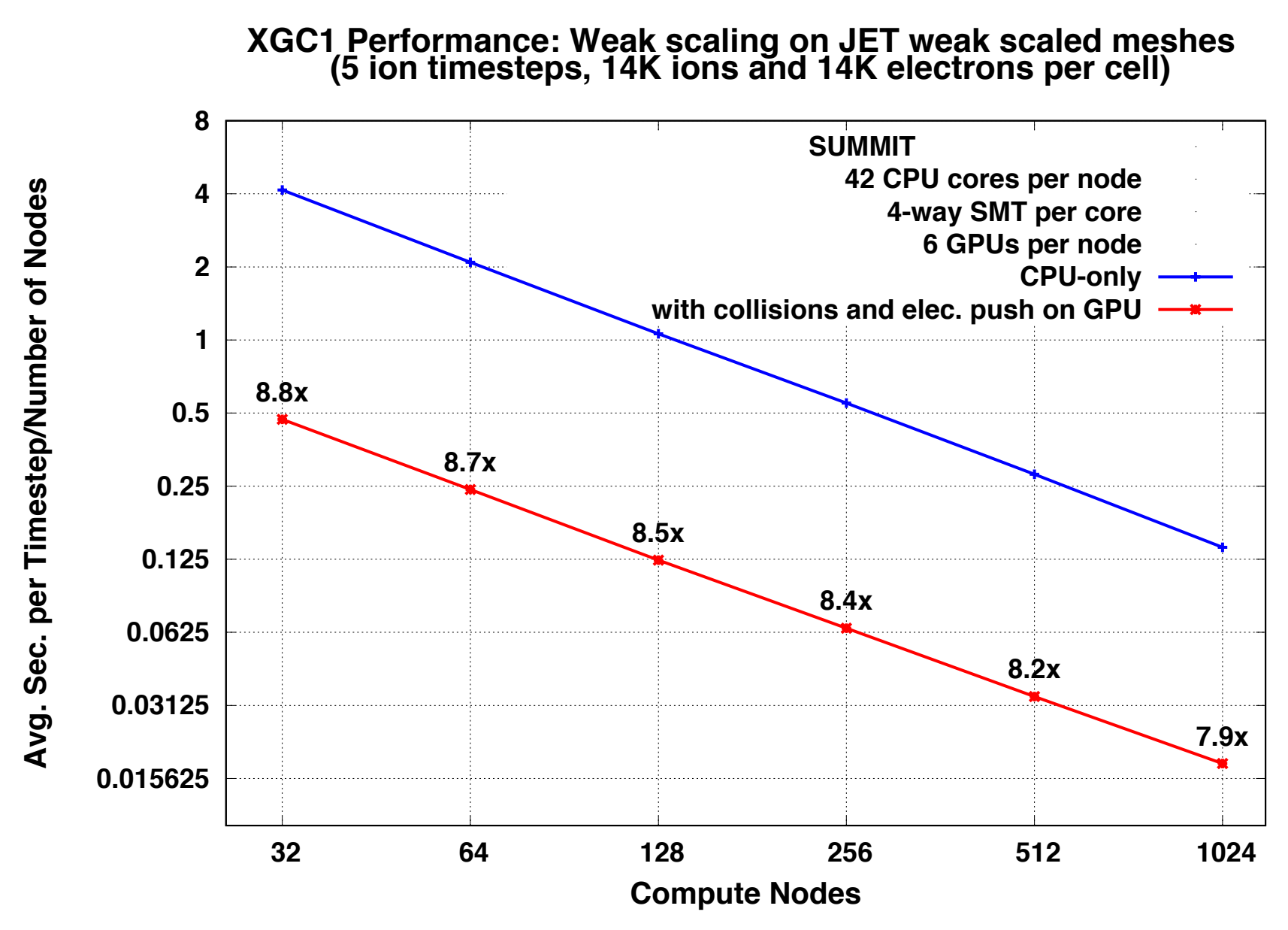
- Re-ordering of loops
- Data structure re-arrangements
- OpenMP pragma statements (loop unrolling, SIMDization)
- Nested OpenMP
- Abstraction for different cache sizes

SYSTEM PORTABILITY









KNL SKU	System	“PUSHE” Time [s]
7210	JLSE	9.32
7230	Theta	8.53
7250	Cori	8.35

OPENACC -> OPENMP

- !$acc kernels pbcopyin(varsIn) pbcopyout(varsOut) → !$omp target map(varsIn, varsOut)
- !$acc loop independent collapse(2) vector(256) → !$omp parallel for collapse(2) simd safelen(256)
- !$acc loop independent worker
  !$acc loop independent vector
  →
  !$omp parallel for
  !$omp simd
- Easy one-to-one mapping for most common features

NEXT STEPS

- Portability with Cabana
- Architecture-specific performance enhancements
- Platform independent software engineering (more loop re-ordering, data structure tiling, etc.)

REFERENCES

- C. S. Chang et. al., Physics of Plasmas **15**, 062510 (2008).
- S. Ku et. al., Nuclear Fusion **49**, 115021 (2009).
- S. Ku, R. Hager, C.S. Chang et al., J. Comp. Physics, **315**, 467 (2016)
- R. Hager. E.S. Yoon. S.Ku et al., J. Comp. Physics, **315**, 644 (2016)
- R. Hager, J. Lang et al., Phys. Plasmas **24**,054508 (2017)
- N. Sultana et. al., XSEDE ‘16 **44** (2016)

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.