

# Experiences in Porting XGc to Summit

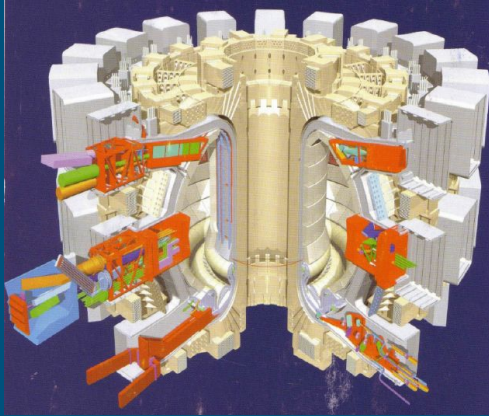
Ed D'Azevedo (ORNL)  
presenting for the XGC Team

ORNL is managed by UT-Battelle, LLC  
for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

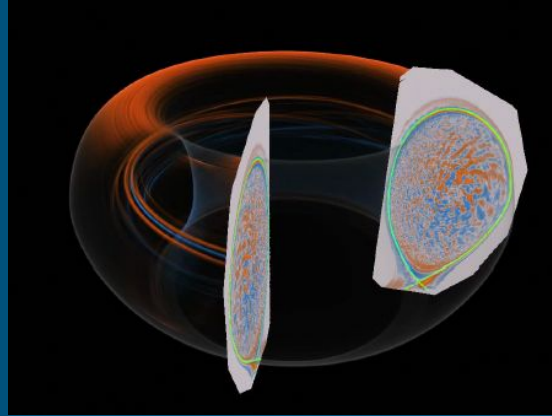
# 6/5D Simulations of Plasmas in “Toroidal” Tokamak Geometry



60m,

**ITER**

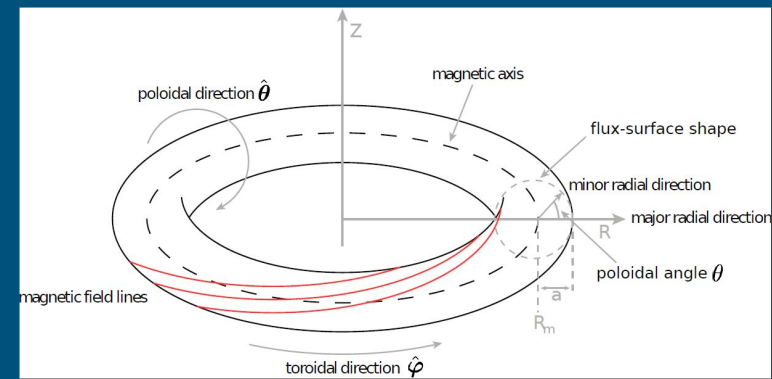
~\$20B



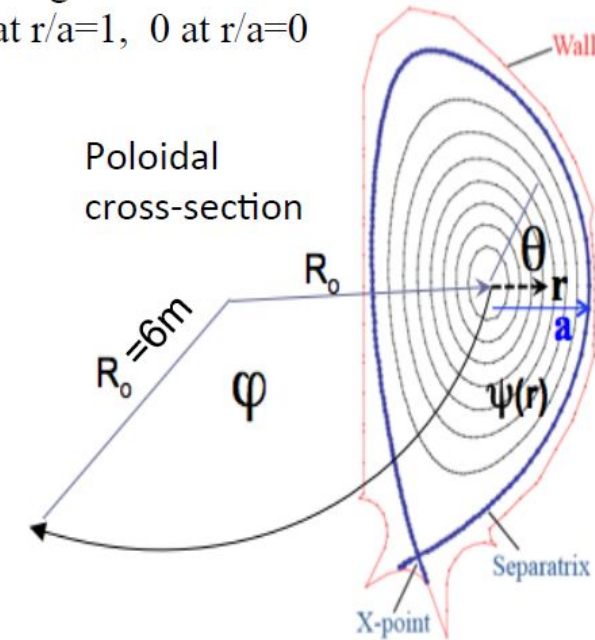
**XGC1 Simulation**

Torus, not a straight cylinder: physics and math become more complicated through spatial inhomogeneity and toroidal mode coupling.

- > Simplified geometry approaches have limited applicability.
- > Requires ~billion-trillion particles to describe important physics
- > Extreme scale simulation

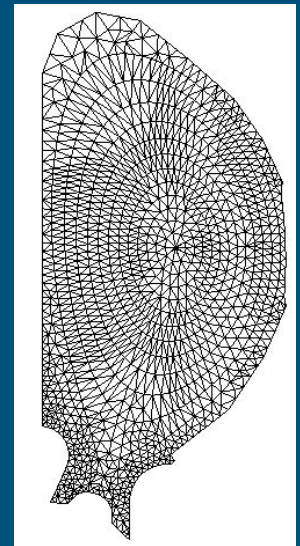
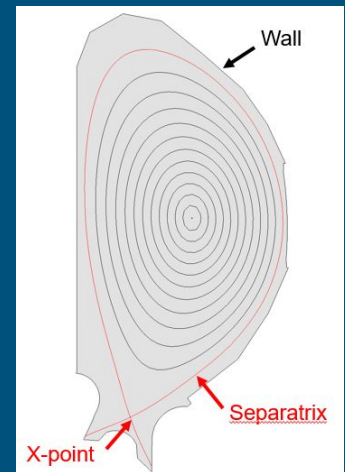


Poloidal magnetic flux label  
 $\psi(r) = 1$  at  $r/a=1$ ,  $0$  at  $r/a=0$



# XGC1 Particle-in-cell Code

- Gyrokinetic approximation reduces 6D problem to 5D
- XGC1: Gyrokinetic particle-in-cell (PIC) code for modeling plasma in tokamak, especially in edge region and near separatrix
- Unstructured triangular grid aligned to flux surfaces to resolve edge geometry and separatrix,  $O(10^6)$  triangles per plane for ITER
- NERSC Exascale Science Applications Program (NESAP) for Cori Phase II (Intel Xeon Phi) and Oak Ridge Center for Accelerated Application Readiness (CAAR) for Summit (Nvidia GPU)
- Performance portability achieved with OpenACC and OpenMP



# XGC

- Center for High Fidelity Boundary Plasma Simulation  
<https://hbps.pppl.gov/computing/xgc-1>
- Mostly F90, Petsc, NTCC Splines, LAPACK, ADIOS for parallel I/O
- Key Kernels: electron push and multi-species collision
- Cuda Fortran in electron push to take advantage of texture cache
- OpenACC version of electron push and multi-species collision
- Separate versions of push kernel for vectorization on Intel KNL

# Nonlinear multi-species collision

[Robert Hager et al., J. Comput. Phys. '16 ]

- Fokker-Planck-Landau collision operator

$$\begin{aligned}
 \frac{\partial f_\alpha}{\partial t} &= \sum_\beta C_{\alpha\beta}(f_\alpha, f'_\beta) \quad \underline{\mathbf{U}} = \frac{u^2 \underline{\mathbf{I}} - \mathbf{u}\mathbf{u}}{u^3}, \text{ where } \mathbf{u} = \mathbf{v} - \mathbf{v}' \text{ and } \underline{\mathbf{I}} = \text{identity tensor} \\
 &= - \sum_\beta \frac{e_\alpha^2 e_\beta^2 \ln \Lambda_{\alpha\beta}}{8\pi \epsilon_0^2 m_\alpha} \nabla \cdot \int d^3 v' \underline{\mathbf{U}} \cdot \left( \frac{f_\alpha}{m_\beta} \nabla' f'_\beta - \frac{f'_\beta}{m_\alpha} \nabla f_\alpha \right) \\
 \frac{\partial}{\partial t} \int f_\alpha \phi d^3 v &= \sum_\beta \frac{e_\alpha^2 e_\beta^2 \ln \Lambda_{\alpha\beta}}{8\pi \epsilon_0^2 m_\alpha} \int d^3 v \nabla \phi \cdot \int d^3 v' \underline{\mathbf{U}} \cdot \left( \frac{f_\alpha}{m_\beta} \nabla' f'_\beta - \frac{f'_\beta}{m_\alpha} \nabla f_\alpha \right) \\
 &= - \sum_\beta \nabla \cdot (\mathbf{E}_{\alpha\beta} f_\alpha + \underline{\mathbf{D}}_{\alpha\beta} \cdot \nabla f_\alpha) = - \sum_\beta \nabla \cdot \mathbf{J}_{\alpha\beta}
 \end{aligned}$$

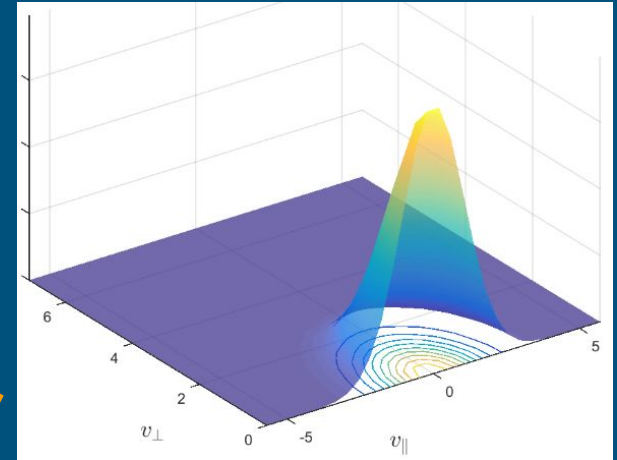
Weak form  
(implemented)

$$\langle \langle M_{\text{self/multi}} \rangle \rangle \equiv \int_0^{2\pi} \int_0^{2\pi} M_{\text{self/multi}} dv_\theta dv'_\theta, \text{ where } M_{\text{self/multi}} = \nabla \phi \cdot \underline{\mathbf{U}} \cdot \begin{pmatrix} \nabla' \\ \nabla \end{pmatrix}$$

$\phi$  : test function (1: density,  $v$ : momentum,  $v^2$ : energy)

# Nonlinear Multi-species Collision Algorithm

- Density functions  $f_{a/b}$ ,  $M \times N$  velocity grid
- Backward Euler for time integration
- Implicit Picard fixed-point iteration to solve nonlinear equation.
- Finite difference/volume discretization
- Computation matrix coefficients E and D  
 $O(M \times N)$  complexity for each velocity grid point :  **$O(M^2 \times N^2)$  - very expensive**
- Spectral methods and FMM investigated



**N**

**M**

$$\frac{f_a^{n+1} - f_a^n}{\Delta t} = \sum_b C_{ab}(f_a^{n+1}, f_b^{n+1})$$

$$\frac{f_a^{(k+1)} - f_a^n}{\Delta t} = \sum_b C_{ab}(f_a^{(k+1)}, f_b^{(k)})$$



# Collision Kernel

- Generate distribution from discrete particle data in Voronoi polygon of vertex (may need to merge subdomains to obtain sufficient number of particles)
- Solve non-linear equations by Picard fixed-point iteration in 2D velocity phase-space ( $M \times N$ )
- **Independent system** on each mesh vertex, 5-point stencil on rectangular grid leads to banded matrix, and solved by LAPACK band solver (on CPU)
- New development in conservative resampling in mapping distribution back to particles may require solving least squares optimization problem with algebraic constraints

# Collision Kernel (2)

- High cost in matrix construction:
  - high memory usage ( $O(M^2 N^2)$ ) may limit number of concurrent kernels
  - large arrays preallocated (for each thread)
  - vectorized evaluation of elliptic functions to fill large arrays
  - efficient AVX vectorization on KNL and OpenACC on GPU
  - nested OpenMP parallelization on CPU
- Sparse matrix transferred and solved by LAPACK band solver (CPU)
- OpenMP threads launch OpenACC kernels
  - concerns about race conditions on Summit
  - disable OpenMP, run single kernel on GPU
- Future needs in multi-species collision can significantly increase the computation cost



# OpenACC Memory Pool

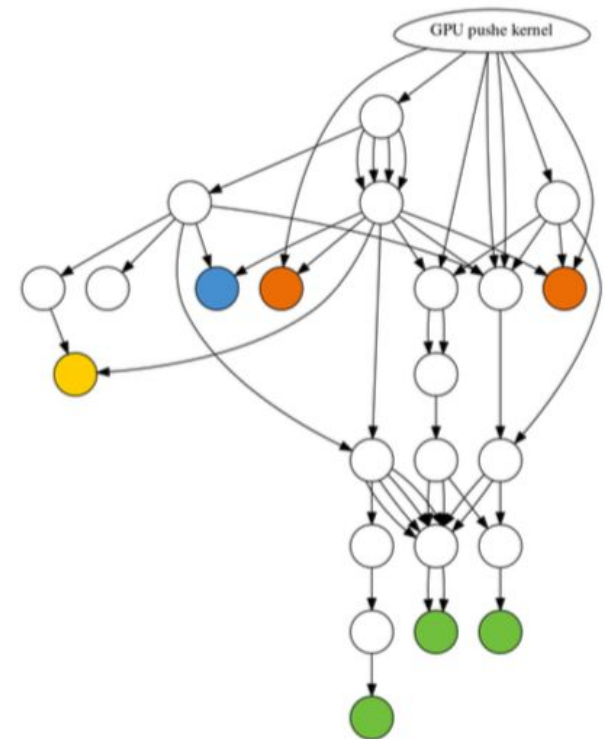
- [https://www.olcf.ornl.gov/wp-content/uploads/2018/03/PGI\\_OpenACC\\_ORNL\\_March\\_2018-final.pdf](https://www.olcf.ornl.gov/wp-content/uploads/2018/03/PGI_OpenACC_ORNL_March_2018-final.pdf)
- OpenACC memory pool as optimization feature  
`acc_malloc()`, `acc_free()`
- Less memory available for CUDA library or CUDA memory allocation
- `PGI_ACC_POOL_ALLOC=0` to turn off this feature
- Other environment variables such as  
`PGI_ACC_POOL_SIZE`, `PGI_ACC_POOL_THRESHOLD`
- call `acc_clear_freelists()` to release memory to CUDA

# Electron Push Kernel

- Electron **sub-cycling** requires about 50 time steps per ion time step, particles can travel cross many planes on torus
- Expensive operations (MPI communication, data movement, allocate device memory) to **replicate** field information to all GPUs
- ***Push performed on GPU without further communication***
- *GPU kernel is limited by memory access, not FLOPS*
- Solve initial value ODE by Runge-Kutta method
- Heuristics to balance work load and device memory

# Electron Push Kernel (2)

- **Deep call graph** of Fortran module routines and data structures:
  - push one particle to completion
  - essentially embarrassingly parallel
  - interpolation from cubic splines
  - locate particle in unstructured triangle mesh (heuristic to check same triangle)
  - CUDA Fortran to use texture cache
  - OpenACC version
- On Titan, push particles on both GPU ( $\geq 70\%$ ) and CPU ( $\leq 30\%$ )



# Optimization for GPU

- Array of Structure (AoS) on CPU, Structure of Array (SoA) on GPU, so data **transpose** required
- Tuning parameter for periodic particle bin **sorting** to improve locality and cache reuse. Note rearrangement of data structure is expensive.
- Particle search by geometric bin hashing into 2D uniform Cartesian grid holding short list of triangles.
- *Heuristic*: First check whether particle is still in **same triangle**.
- Particle binning or **sorting** (by triangle number) → require custom allocator for optimized **prefix sum** scan by Nvidia Thrust library

# Optimization for GPU (2)

- Asynchronous data transfer between CPU/GPU
  - Expect particle data to fill GPU device memory
  - dynamic pinning of host buffers
  - preallocated (small) device buffers to perform transpose operations on GPU
  - employ multiple streams to overlap
  - Implemented using F2003 abstract type, abstract procedures customized for transpose operation of different data structures
  - CUDA events for **synchronization** and timing
- Asynchronous (non-blocking) electron push on GPU
- Concurrently push ion particles on multi-cores using OpenMP
- Dynamic load balancing of particles assigned to GPU and GPU. However, nearly all particles assigned to Volta GPU.

# Future Development

- Texture cache may not be necessary for Volta with combined L1 cache
- Avoid **transpose** by using data structure optimized for GPU
- **Over-subscribe** device memory for particle data, concurrently overlap data movement with GPU push
- new particle-mesh library under development to **avoid replicating** all field data



# Vectorized Push Kernel for KNL

- Developed by NESAP Postdoc (Tuomas Koskela) for Cori/KNL and still developed by ALCF Postdoc
- Vectorized version for KNL (with SoAoS) to push groups of particles
- Vectorized version of code performs poorly when ported using OpenACC on GPU:
  - long loops split up as **simpler loops** to be recognized for vectorization by compiler
  - **temporary vectors** increase register usage and decrease parallelism on GPU

# Atomic Update

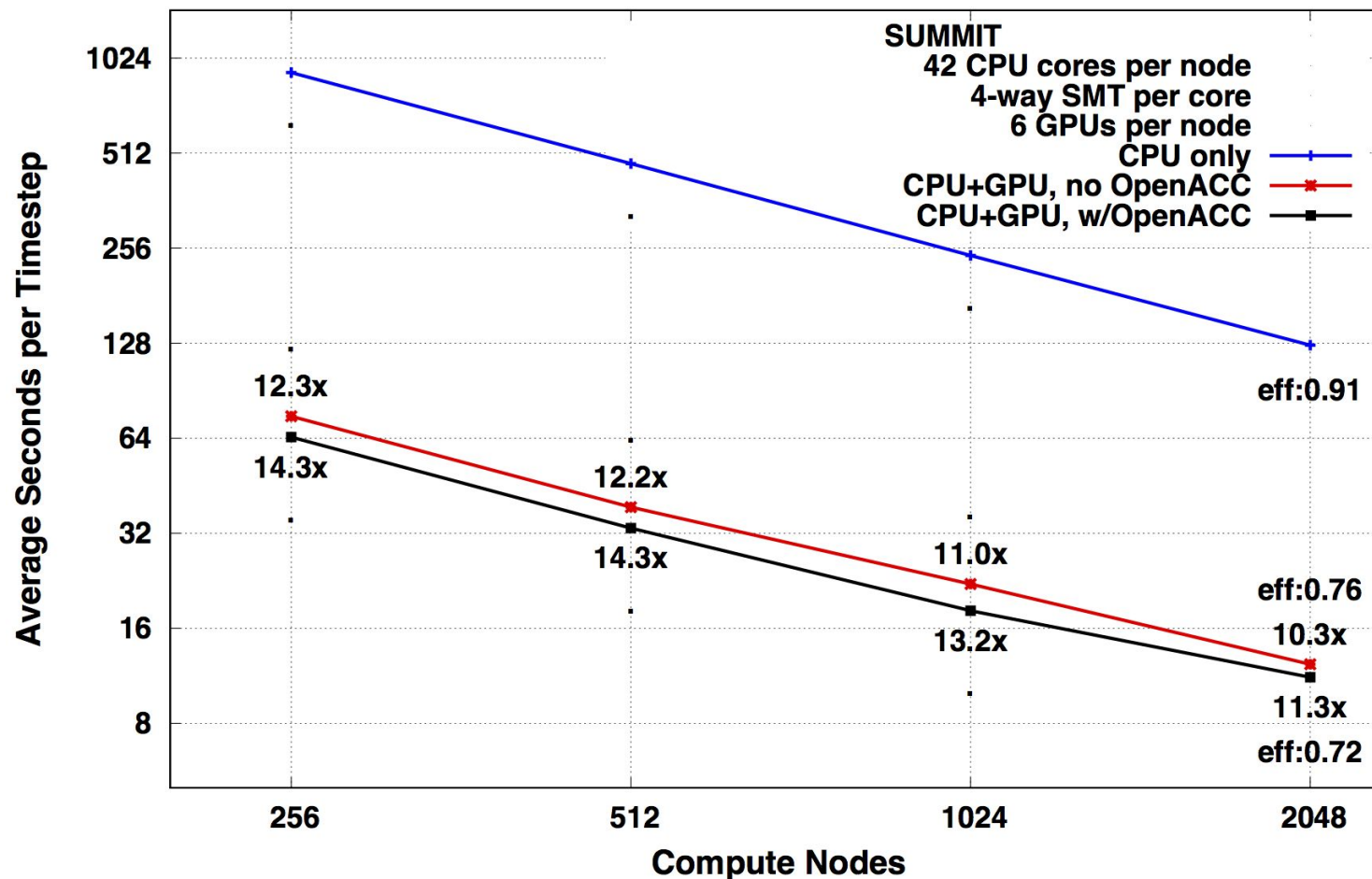
- Need atomic 64-bit floating point (FP64) updates for particle charge deposition on background grid and for diagnostics
- Avoid collision hazard by replicating arrays for OpenMP threads
- Kepler GPU has hardware support for atomic FP32 updates, but uses atomic 64-bit compare-and-swap to **emulate** atomic update for FP64
- Pascal and Volta GPUs have **hardware** support for atomic update of **FP64** with significantly improved performance

# Strong Scaling of XGC

- XGC has been scaled to 2048 nodes (over 40% of full machine) on Summit with about 90% parallel efficiency
- The CPU + GPU version is over 10X faster compared to not using the GPU
- Using same number of GPUs on Titan (12288 nodes), the Summit version is over 3X faster

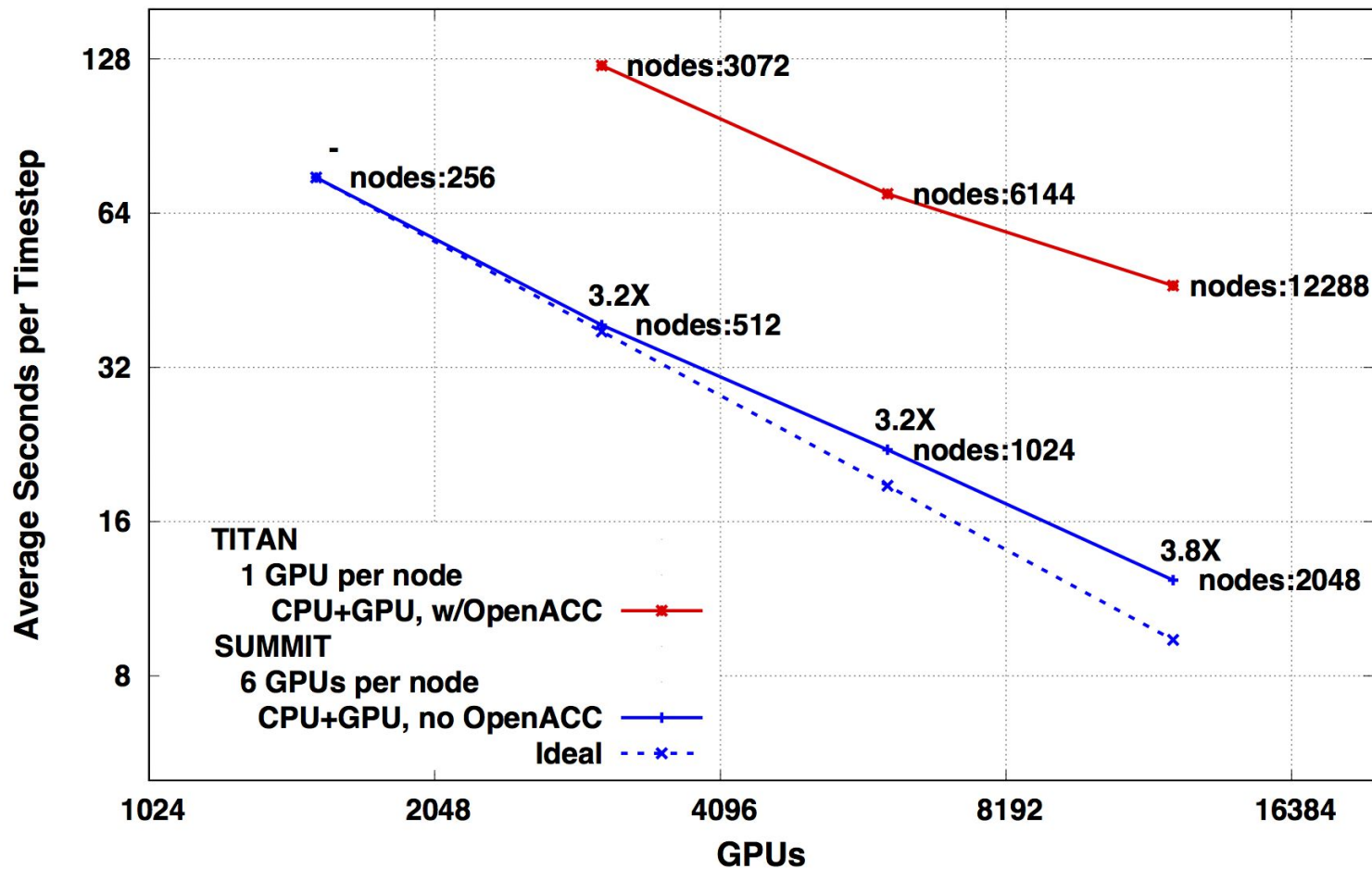
# Strong scaling of XGC on Summit

**XGC1 Performance: Strong scaling on DIII-D mesh  
(13K ions and 13K electrons per mesh cell, with collisions)**



# Comparison of Titan vs Summit

**XGC1 Performance: Strong scaling on DIII-D mesh  
(13K ions and 13K electrons per mesh cell, with collisions)**



# Porting challenges

- **Compiler**, system software, tools for profiling and debugging
- **Need stable system**
- Concern about race condition in OpenMP threads launching OpenACC kernels
- OpenACC has separate pool of device memory (optimization feature)
- Band solver on GPU can reduce data movement
- Load balancing between particle push, collision kernel (number of local mesh vertices), amount of GPU device memory



# Acknowledgements

- Support provided through the SciDAC program funded by US DOE Office of Advanced Scientific Computing Research and Office of Fusion Energy Sciences.
- Awards of computer time were provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program.
- This research used resources of OLCF, ALCF, and NERSC, which are U.S. DOE Office of Science User Facilities supported under contracts DE-AC05-00OR22725, DE-AC02-06CH11357, and DE-AC02-05CH11231, respectively.

# Questions?

Ed D'Azevedo  
[dazevedoef@ornl.gov](mailto:dazevedoef@ornl.gov)

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

