GPUS FOR SCIENCE DAY 2022

# QMCPACK AND GPU

QMCPACK

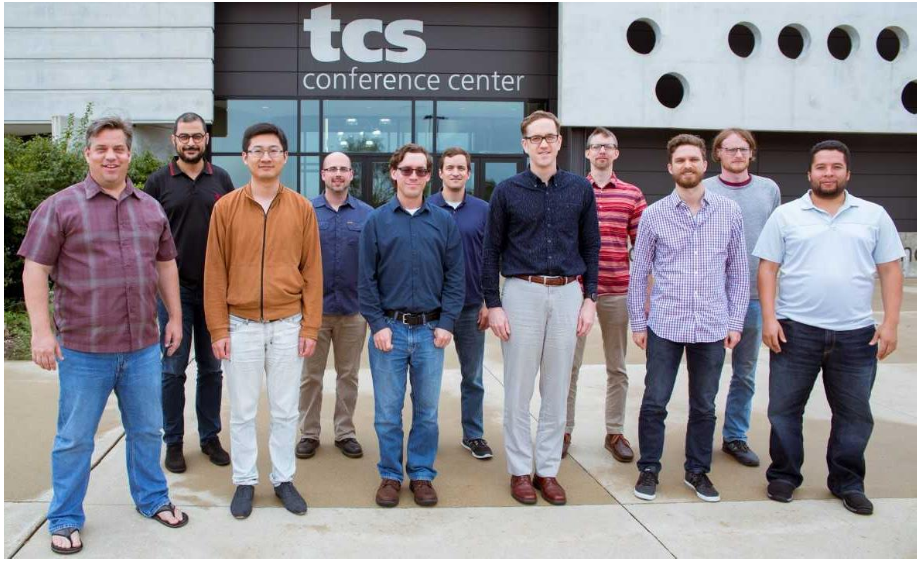
**YE LUO**
Computational Science Division,
Leadership Computing Facility,
Argonne National Laboratory

Oct. 25th 2022

# ACKNOWLEDGEMENT

## Exascale Computing Project : application development

- Lead PI: Paul Kent
- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

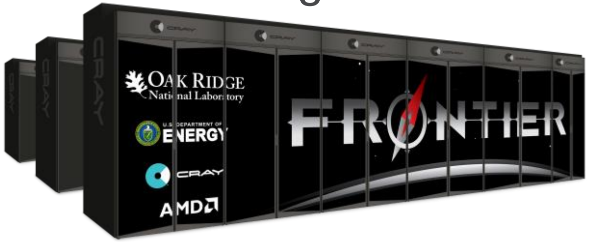


# PRE AND EXASCALE SYSTEMS IN 2022

## Need to make the code work on all machines

NERSC Perlmutter

Oak Ridge Frontier



Argonne Polaris

Argonne Aurora

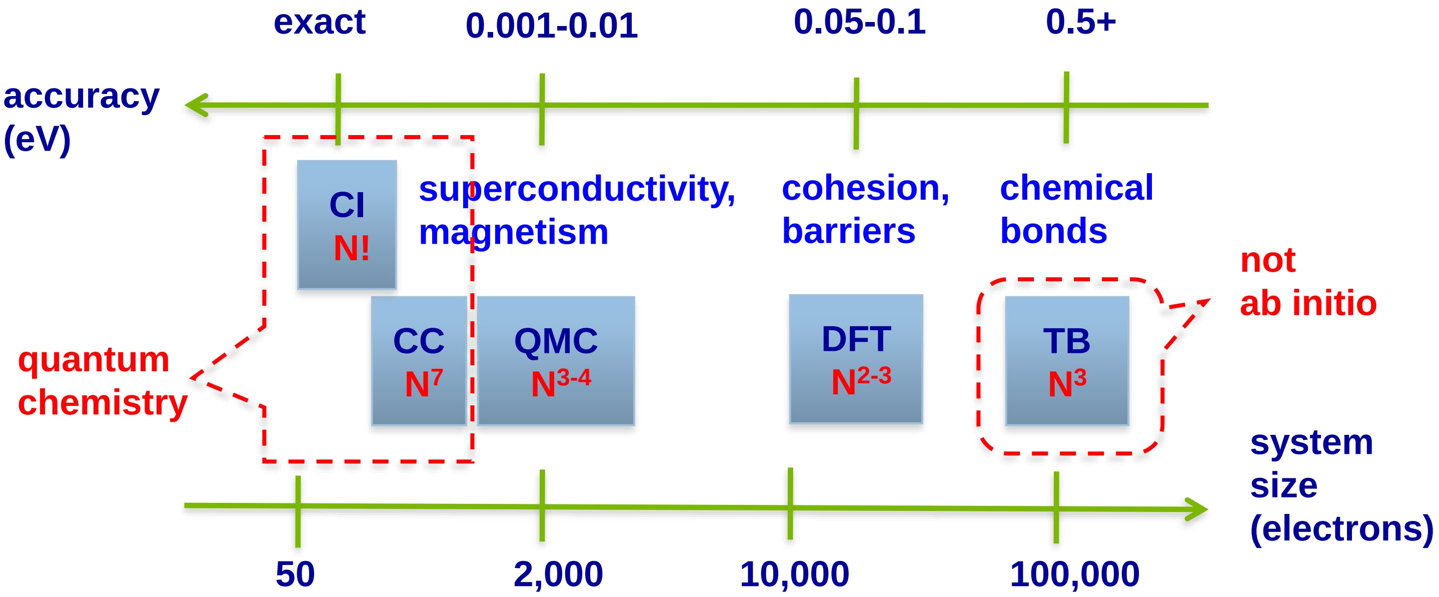
Make QMCPACK run well everywhere

# ELECTRONIC STRUCTURE METHODS

## QMC can be the new sweet spot

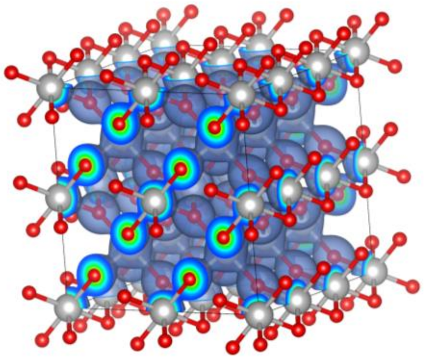
Time scale: picosecond = $10^{-12}$ seconds

Length scale: 10 nm = $10^{-8}$ meters



# PETASCALE TO EXASCALE CHALLENGE

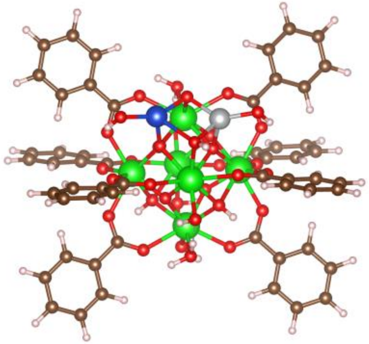
## How large problem can we solve?



TiO2 polymorphs

216 atoms with 1536 electrons, 10 meV/f.u.

YL et al. New J. Phys. 18 113049 (2016)



Metal organic framework

153 atoms with 594 electrons, 10 meV total energy.

A Benali, YL, et al. J. Phys. Chem. C, 122, 16683 (2018)

What is next?

1. Solve faster and more petascale problems
2. Solve much larger problems

1k atoms

10k electrons

# QMCPACK

- QMCPACK, is a modern high-performance open-source Quantum Monte Carlo (QMC) simulation code for electronic structure calculations of molecular, quasi-2D and solid-state systems.
- The code is C/C++ and adopts MPI+X (OpenMP/CUDA)
- Monte Carlo: massive Markov chains (walkers) evolving in parallel. 1st level concurrency. Good for MPI and coarse level threads.
- Quantum: The computation in each walker can be heavy when solving many body systems (electrons). 2nd level concurrency. Good for fine level threads and SIMD.
- Math libraries: BLAS/LAPACK, HDF5, FFTW

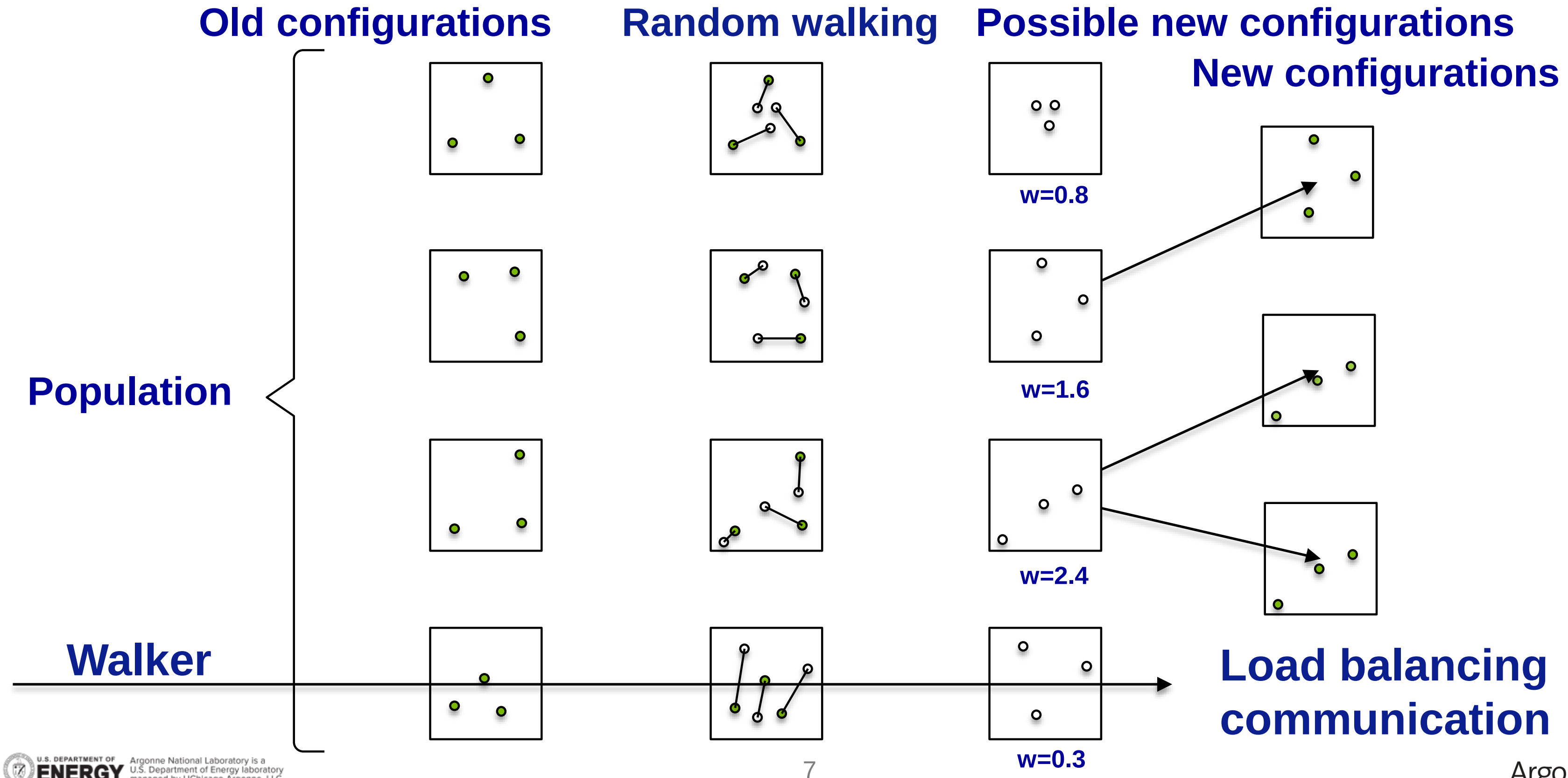
Correlated Defects via QMC: Mn dope

Quantum Monte Carlo is applied to phosphors for the first time.

NERSC

OAK RIDGE National Laboratory

# DIFFUSION MONTE CARLO SCHEMATICS

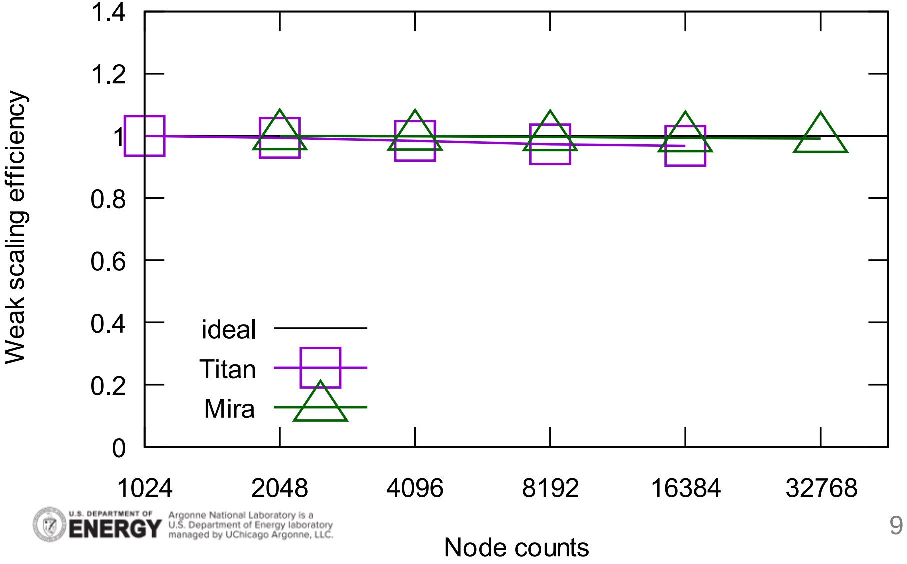


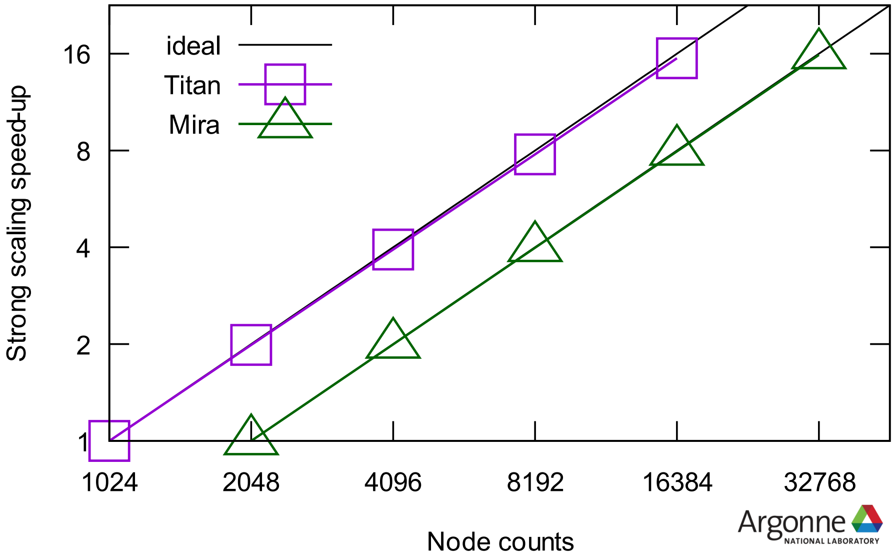
# PARALLELIZATION AND GPU PORTING

# WALKER BASED PARALLELISM

## Works extreme well on petascale supercomputers

- Weak scaling efficiency 99% on 2/3 Mira and 95% on almost full Titan.
- Weak scaling, fix work per node. Strong scaling, fix the total number of samples.
- Equilibration excluded.





# MAPPING CONCURRENCY TO PARALLELISM

## Monte Carlo can be a challenge for parallelism

Friendly
Unfriendly

- Walkers $N_w$ are not data parallel but task parallel
  - Workload per electron move depends on accept/reject. GPU
  - Workload per step moving all the electrons is roughly equal. CPU
- Electrons are data parallel
  - Naturally, $N_e$ vector computation utilizing SIMD and SIMT. CPU/GPU
  - Kernels are $O(N_e^{2\text{-}3})$ per sample. Large $N_e$ CPU. Small $N_e$ GPU.
- Need a tailored approach for performance portability beyond programming models.
- More details in Hipar22 publication “A High-Performance Design for Hierarchical Parallelism in the QMCPACK Monte Carlo code” https://arxiv.org/abs/2209.14487

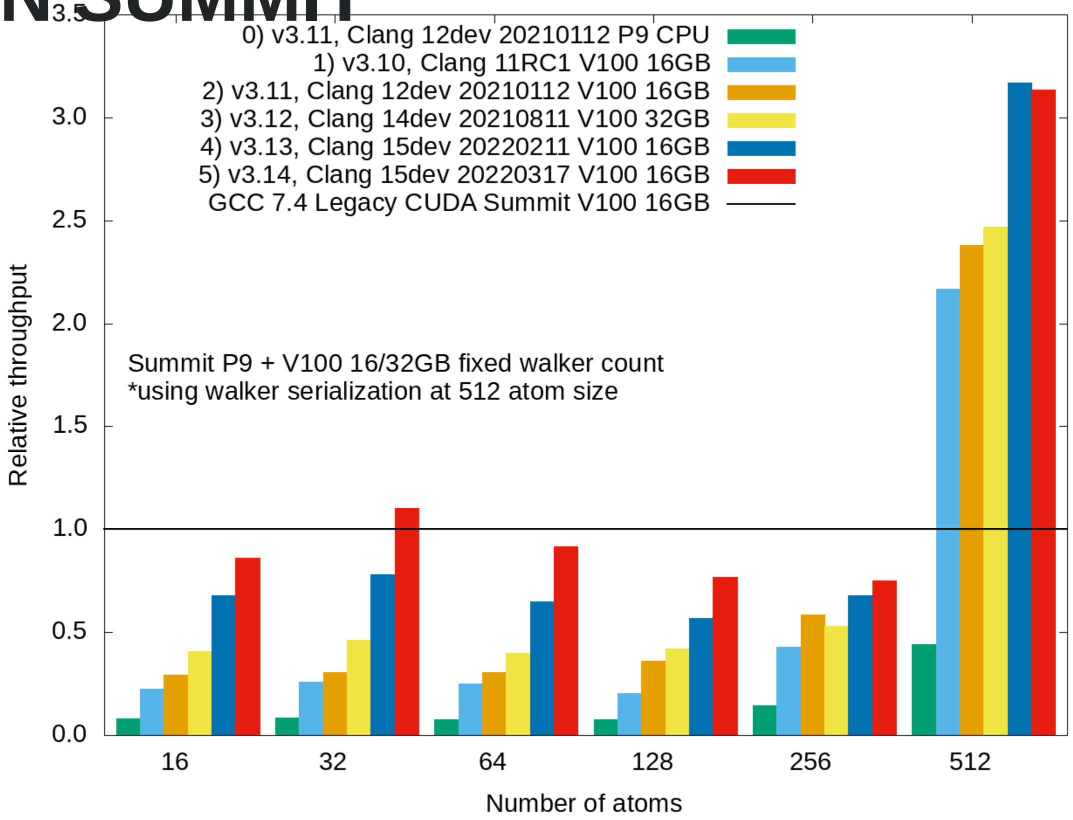
# OPENMP OFFLOAD GPU IMPLEMENTATION

## A bit more software technology to handle GPUs

- Use portable OpenMP target feature
  - Portable on NVIDIA, AMD, Intel GPUs. Fallback on CPU as well.
  - Multiple compilers. GNU, Clang, AOMP, NVHPC, OneAPI
- Multiple CPU threads to launch kernels to GPUs
  - Maximize GPU utilization. Overlapping compute and transfer by OpenMP.
- Specialized in CUDA/HIP to call NVIDIA/AMD/INTEL accelerated libraries.
  - cuBLAS/cuSolver, hipBLAS/hipSolver, MKL
- C++ templates to cover real/complex and full/mixed precision cases.
  - We use C++17 to keep our code concise.

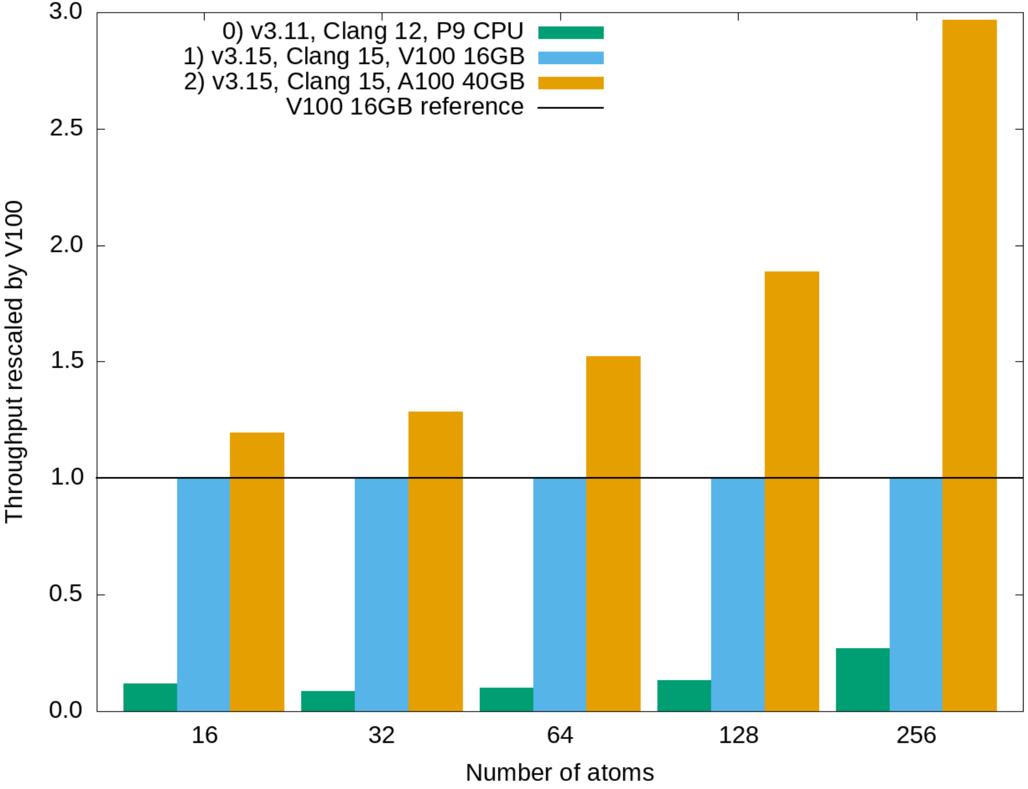
# PERFORMANCE ON SUMMIT

- Porting starts in 2019.
- Steady performance improvements 3.10 to 3.14(dev)
- Close or exceed CUDA performance
- There are still room to improve the performance.
- Passes tests, very usable in production



# V100 VS A100

- GPU acceleration is significant
- The larger HBM helps throughput. Very similar to ML.
- A100 is almost 3X when running 256 atom problem.
- 16 GB is the bottleneck on V100



# LESSONS LEARNED SO FAR

## What is needed for a performance portable code

- Understand how to make CPU and GPU work efficiently
- Analyze the compute pattern and map the existing concurrency to proper parallelism given by the hardware
- OpenMP + vendor libraries strategy works
  - ~100 CUDA kernels down to ~10 CUDA kernels + ~10 offload regions
  - Very maintainable code with decent performance

# MY OPINION ABOUT GPU PORTING

# DATA MOVEMENT IS THE KEY OF PORTING

- Data locality is the top priority
  - Interconnect is slower than memory. We write performant MPI code
  - Memory is slower than cache. We implement cache friendly algorithms.
  - CPU-GPU bus is slower than GPU memory. We need to first worry about data transfer.
- Avoid any programming model which
  - Ignores the performance difference from host and GPU memory spaces.
  - Doesn’t provide explicit data movement control
- GPU 101 teaches kernel programming. It is not the key.
- Managing data movement requires understanding the whole algorithm and implementation. This needs domain expert knowledge.

# MANAGE MORE LEVEL OF PARALLELISM

- At least two level of parallelism
  - CPU has core+SIMD
  - GPU has SM+SIMT lanes
  - In QMCPACK we have CPU threads + two level inside GPU kernel
- Reduction operation plays a key role in scientific computing
  - It is about locality and hierarchy

# CUDA LANGUAGE IS NOT THE BEST CHOICE

## For scientists

- Must be protected under macro. Macro means more test variants needed.
- Doesn’t run on hosts without GPUs.
  - Running and debugging on host are lot smoother and should catch most user errors
  - Host tooling are richer. Address and thread sanitizers, code coverage.
- Offload models are more appealing: OpenACC/OpenMP
- Restrict it to only serving library calls.
- Scientist needs to spend time on methods, algorithms. Leave the engineers to worry absolute performance.

# OPENACC AND OPENMP

## Competitors and siblings

- OpenMP has a richer feature set
- OpenACC is more developed for NVIDIA GPUs
- The needed developers’ knowledge is the same for GPU programming
- They are not necessarily interchangeable in syntax.
- Choose one based on your production environment and programming language.
- Keep in mind that compilers are never perfect. They have bugs but be kind to developers. Directive based programming model eases the burden of code developers and shifts that to compiler developers.
- CUDA doesn’t have a reduction!!!!

# SUMMARY

# SUMMARY

- After a careful redesign of QMCPACK, we have new performance portable implementation, and it is the second time of GPU porting.
- OpenMP + vendor library strategy works well on NVIDIA GPUs and we are expanding the list of supported GPUs from different vendors.
- GPU porting is a chance to force designing and implementing better code. Don’t miss the opportunity.

U.S. DEPARTMENT OF ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY