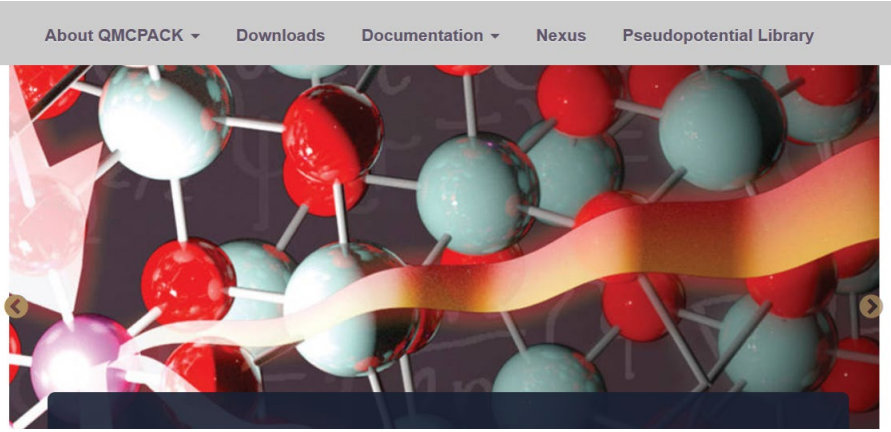
IXPUG ANNUAL CONFERENCE 2023

Argonne NATIONAL LABORATORY

PATH TO EXASCALE MATERIAL SIMULATION ON AURORA SUPERCOMPUTER



YE LUO
Computational Science Division,
Argonne National Laboratory

THOMAS APPLENCOURT
Leadership Computing Facility,
Argonne National Laboratory

JEONGNIM KIM
Intel Corp

Sep. 21th 2023, Santa Clara, CA

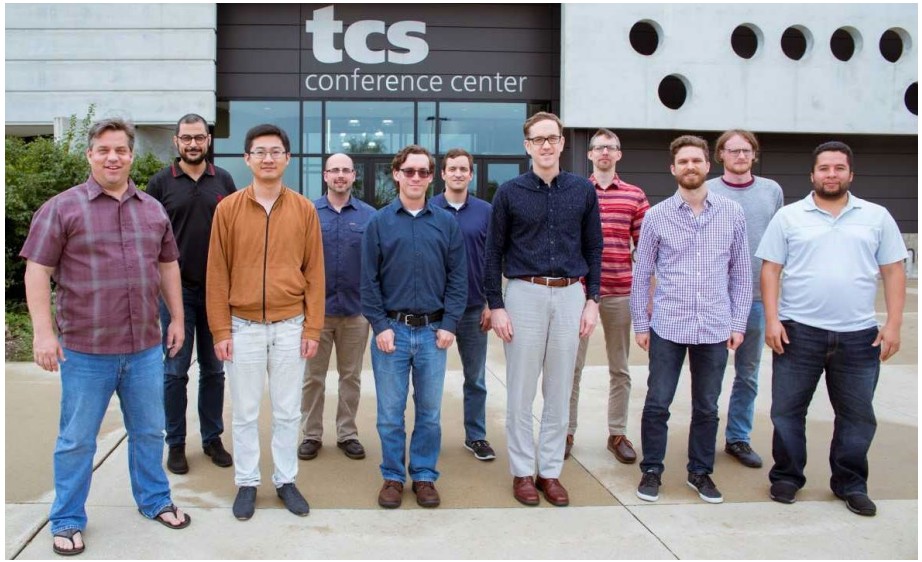
OUTLINE

- QMCPACK intro
- Redesign for performance portability
- GPU and OpenMP porting tips
- QMCPACK on INTEL GPUs

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.

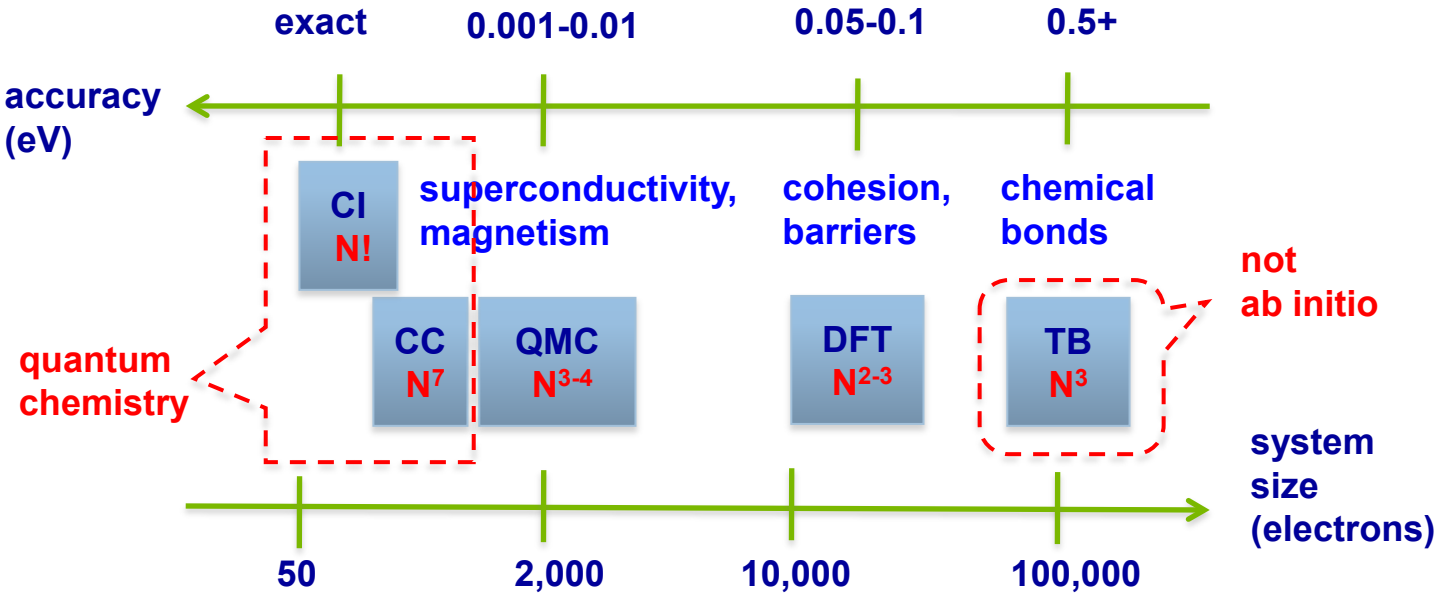


ELECTRONIC STRUCTURE METHODS

QMC can be the new sweet spot

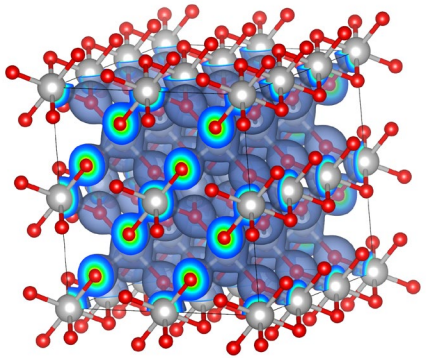
Time scale: picosecond = 10^{-12} seconds

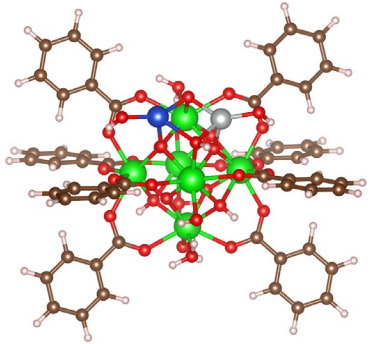
Length scale: 10 nm = 10^{-8} meters



PETASCALE TO EXASCALE CHALLENGE

How large problem can we solve?





Metal organic framework
153 atoms with 594 electrons, 10 meV total energy.
A Benali, YL, et al. J. Phys. Chem. C, 122, 16683 (2018)

TiO2 polymorphs
216 atoms with 1536 electrons, 10 meV/f.u.
YL et al. New J. Phys. 18 113049 (2016)

What is next?

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

1k atoms
10k electrons

QMCPACK

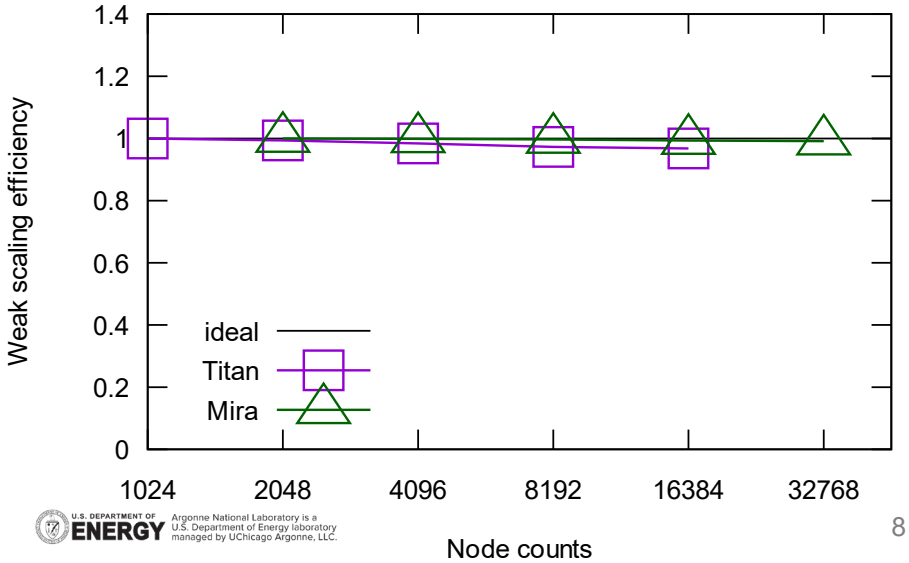
- QMCPACK, is a modern high-performance open-source Quantum Monte Carlo (QMC) simulation code for electronic structure calculations of molecular, quasi-2D and solid-state systems.
- The code is C/C++ and adopts MPI+X (OpenMP/CUDA)
- Monte Carlo: massive Markov chains (walkers) evolving in parallel. 1st 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

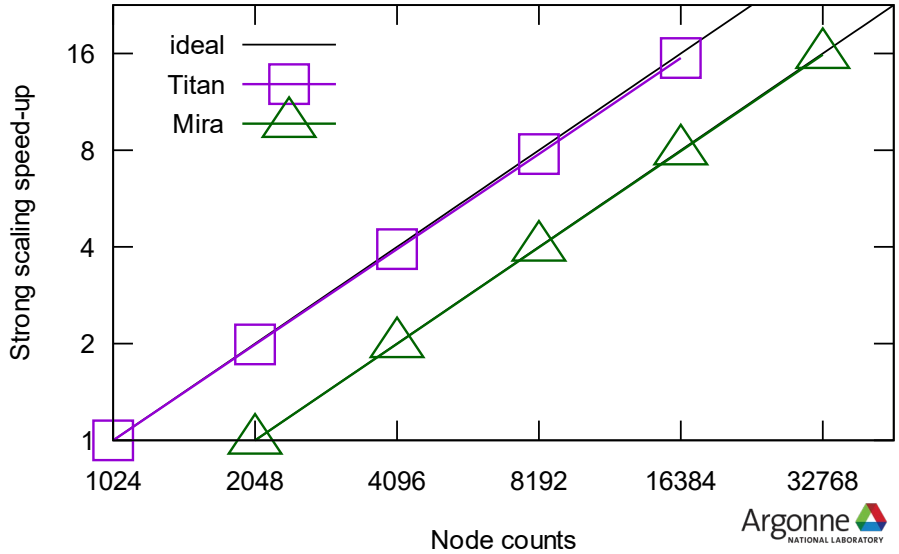
NERSC OAK RIDGE National Laboratory

WALKER BASED PARALLELISM

Works extreme well on petascale supercomputers

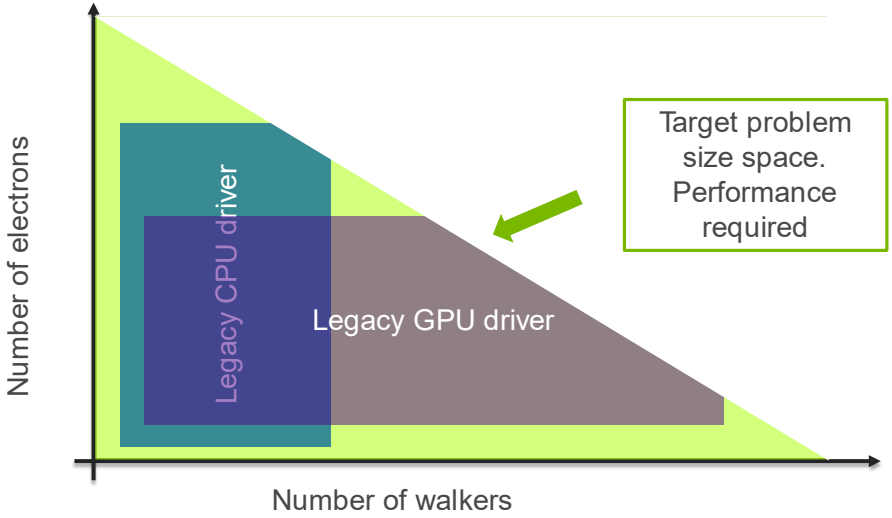
- Weak scaling efficiency 99% on 2/3 Mira and 95% on almost full Titan.
- Weak scaling, fix work per node. Strong scaling, fix the total number of samples.
- Equilibration excluded.

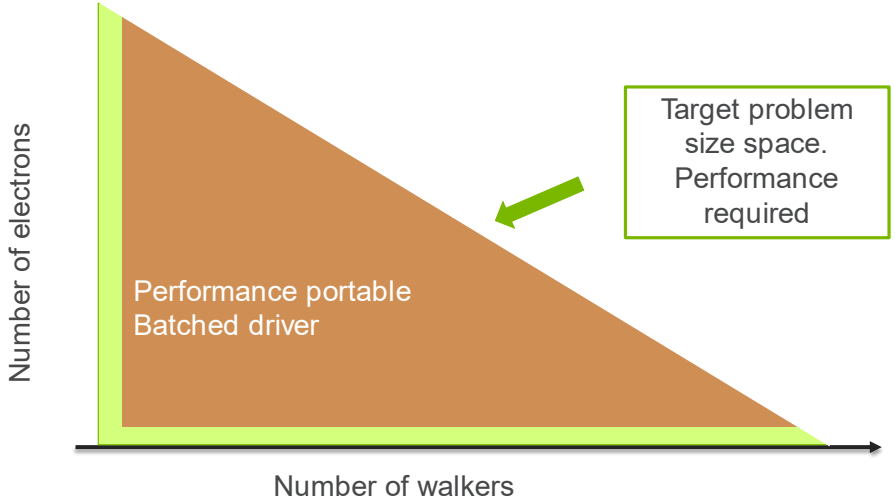




UNIFY BOTH IMPLEMENTATIONS

By design

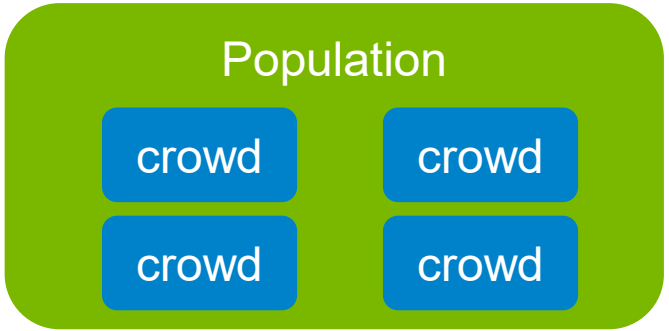




NEW DESIGN WITH CROWDS

Algorithm 4 Pseudocode for the batched DMC driver.

1: **for** MC generation $= 1 \cdots M$ **do** — seq.
2: #pragma omp parallel for
3: **for** crowd $= 1 \cdots C$ **do** — para.
4: **for** particle $k = 1 \cdots N$ **do** — seq.
5: Algorithm 1. Line 5,6,7,8,9 over all walkers with in this crowd — batched
6: **end for**{particle}
7: **local energy** $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$ over this crowd
8: reweight and branch walkers based on $E_L - E_T$
9: update E_T and load balance via MPI.
10: **end for**{crowd} CG
11: **end for**{MC generation}



- lock-step walkers within a crowd
- Independent crowds
- Decay to legacy implementations

doi: 10.1109/HiPar56574.2022.00008.

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 crowds (CPU threads) to launch kernels to GPUs
  - Maximize GPU utilization. Overlapping compute and transfer by OpenMP.
- Specialized in SYCL/CUDA/HIP to call INTEL/NVIDIA/AMD accelerated libraries.
  - MKL, cuBLAS/cuSolver, hipBLAS/rocSolver

MULTI-THREADED OFFLOAD

A few more tips

- Using pinned memory to keep CPU cores submitting work to GPUs.
  - Method 1. Pin host memory using vendor APIs like cudaHostRegister
  - Method 2. allocated pinned memory using vendor APIs like **sycl::aligned_alloc_device<T>**. github**#3973**
  - Method 3. Use OpenMP extension llvm/**omp_target_alloc_host** (supported by icx/icpx)

- Avoid allocating/deallocating GPU memory on the fly
  - Allocating/deallocating operations are very slow
  - Serialization prevents concurrent execution.

USING L0 COMMANDLISTIMMEDIATE

Low latency kernel submission

- Both OpenMP and SYCL are built on top of LevelZero
  - Command list (old) and “immediate” command list (new)
- OpenMP switch to L0 “immediate” command list by default
  - Used like a CUDA stream
  - Enqueue H2D/Kernel/D2H in a single shot and reduce time spent on L0 runtime.
- SYCL in-order queue
  - Use sycl::property::queue::in_order() when constructing the queue. github/**#4663**
  - Reduce effort for porting algorithms using CUDA streams.
  - No need of managing events by users. github/**#4738**

SYCL AND OPENMP INTEROPERABILITY

QMCPACK github #4382

- QMCPACK uses OpenMP to generate L0 device and context.

```
#pragma omp interop device(id) init(prefer_type("level_zero"), targetsync : interop)
auto hPlatform = omp_get_interop_ptr(interop, omp_ipr_platform, &err);
auto hContext = omp_get_interop_ptr(interop, omp_ipr_device_context, &err);
auto hDevice = omp_get_interop_ptr(interop, omp_ipr_device, &err);
```

- Build SYCL objects

```
sycl::ext::oneapi::level_zero::make_platform(reinterpret_cast<pi_native_handle>(hPlatform));
sycl::ext::oneapi::level_zero::make_device(sycl_platform,
reinterpret_cast<pi_native_handle>(hDevice));
default_device_queue =
std::make_unique<sycl::queue>(visible_devices[sycl_default_device_num].get_context(),
                              visible_devices[sycl_default_device_num].get_device(),
                              sycl::property::queue::in_order());
```

Keep a per device default queue for non-critical use

GPU MEMORY QUERY

QMCPACK Github #4692

- Not on default.
  - SYCL only code, user initializes sysman.
  - OpenMP code, Need environment variable ZES_ENABLE_SYSMAN=1
- get_info<sycl::ext::intel::info::device::free_memory>()
  - SYCL extension

Aurora

Computing Quantum Mechanical Properties faster.

SUMMARY

- QMCPACK was ported for Intel GPUs on Aurora with
  - OpenMP offload. Mostly validating compilers and runtime libraries.
  - Minimal SYCL code for optimal performance.
  - Using MKL libraries. Validating this correctness and performance.
- The overall performance portability strategy fits well on Intel software and hardware.
  - We achieved good performance which paves the way for the success of Aurora.
  - There will be further performance gain as we keep improving QMCPACK and software for intel GPUs.