QMCPACK Users Workshop 2023

12-14 December, Argonne National Laboratory

Towards QMCPACK v4.0

Paul Kent, kentpr@ornl.gov

https://github.com/QMCPACK/qmcpack_workshop_2023

Funding: U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division, as part of the Computational Materials Sciences Program and Center for Predictive Simulation of Functional Materials.

CPSFM Center for Predictive Simulation of Functional Materials

Sandia National Laboratories

Argonne NATIONAL LABORATORY

NC STATE UNIVERSITY

BROWN

OAK RIDGE National Laboratory

Outline

The Who/What/Why/When/How of the “batched drivers” / “performance portable” code / v4.0

Takeway

This change affects _everyone_ whether for CPU or GPU machines

Try the batched/performance portable drivers today

Build & use the QMCPACK development version (preferred, most features)

Set the driver_version in your qmcpack_input.xml:

```
<project id="name_for_calculations" series="0">
<parameter name="driver_version">batch</parameter>
</project>
…
```

Updates required for some parameters and observables e.g. “walkers”->”total_walkers” or “walkers_per_task” in DMC (reduces ambiguity, complexity)

…or use NEXUS!

Motivation

Keep QMC and QMCPACK running on modern computers, particularly GPU accelerated machines. Becoming ever more common due to power efficiency, high suitability for ML and AI workloads. Large CPU-only machines becoming rarer.

Perlmutter:
NVIDIA A100 GPUs
& AMD CPUs

Frontier: AMD GPUs and CPUs

Aurora: Intel Intel Xe GPUs & CPUs

Goals

Performance portable CPU and GPU support from a single codebase, >95% the same for all platforms. Not achieved before.

Productive: All features always available. Not achieved before.

Ability for excellent GPU performance, still excellent CPU performance.

Better defined inputs, error checking (aids automation).

Refactoring for easier development and testing.

More runtime instead of compile time options, e.g., single build for complex and real versions

Expected Features

Everything needed to do ~95% of recent publications

Core features on CPUs and partially or fully accelerated on NVIDIA, AMD & Intel GPUs:

- VMC, DMC, Trial wavefunction optimization.
- Spline basis sets.
- LCAO basis sets (early accelerated version).
- Multideterminant (early accelerated version).
- Limited selection of observables including electron density, density matrices.
- Wednesday's new features, e.g., Orbital optimization.

Ongoing support for the above!

Missing from v4.0 (without volunteers):

All particle moves

Reptation QMC

Backflow optimization

Less common observables (?)

Some model Hamiltonian features

…

=> Deferred to 4.1+ and prioritized based on requests and scientific impact.

Today's development version

Energies are good, tests pass, production science has been done.

Still to do before 4.0:

- Making batched drivers the default
- Removing/block the legacy code paths
- Updating more tests / Checking test coverage
- Cleaning documentations, tests, examples
- More real world-usage & taking user feedback into account

Aim to get v4.0 out "early" in 2024

CPU Runs

For modern/up-to-date inputs and standard VMC/DMC runs:

Put “<parameter name="driver_version">batch</parameter>”

(DMC) Change walkers to walkers_per_rank or total_walkers

Use MPI tasks and OpenMP threads as before

Should get ~identical performance to older drivers

[Not discussing “crowds” in this talk for simplicity]

Batching

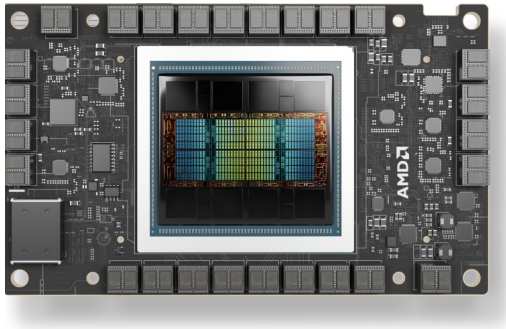
Exascale-generation GPUs from NVIDIA, AMD, and Intel all need >>10^4 similar operations in flight for optimum performance.

Updating a single walker of $10^{2\text{-}4}$ electrons will never be enough. Also need to hide GPU latencies, manage memory…

=> Write the code to work on batches (groups) of walkers simultaneously (Esler 2012).

=> A huge update to support all functionality.

=> Needs very large batches for small electron counts, and even for VMC, >>1000.



AMD MI300X
19456 stream processors

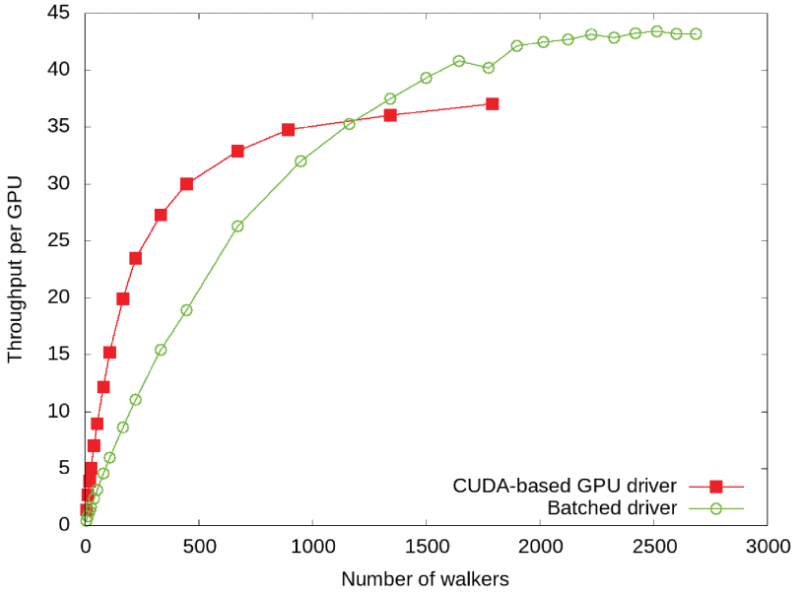
GPU Runs

For modern/up-to-date inputs:

Put `<parameter name="driver_version">batch</parameter>` within `<project>`

Set walkers_per_rank (batch size) to a large number. Perhaps 1000 for 200 electrons, 100 for 2000 electrons. Scan for best throughput e.g. largest that does not run out of GPU memory.

Start with 1 MPI task per GPU and 1 OpenMP thread. [Not discussing “crowds” in this talk for simplicity].



NiO 32 atom
V100 performance

Building the GPU version

See the manual and the config directory for examples.

NVIDIA: Working well with recent LLVM compiler, latest CUDA releases (12.3).

AMD, Intel: Vendors are still maturing their software. Frontier recipe in config directory. Talk with us for the latest recipes, current caveats.

Code will be optimized based on real-world feedback on all 3 vendor GPUs.
Current goal is reliable production science on all 3 vendor GPUs.

Documentation

Several sections in the QMCPACK manual:

https://qmcpack.readthedocs.io/en/develop/performance_portable.html

https://qmcpack.readthedocs.io/en/develop/methods.html#batched-drivers

Technical details:

"A High-Performance Design for Hierarchical Parallelism in the QMCPACK Monte Carlo code", Ye Luo, Peter Doak, Paul Kent. 2022 IEEE/ACM International Workshop on Hierarchical Parallelism for Exascale Computing (HiPar) (2023). https://doi.org/10.1109/HiPar56574.2022.00008

Takeway

This change affects _everyone_ whether for CPU or GPU machines

Try the batched/performance portable drivers today

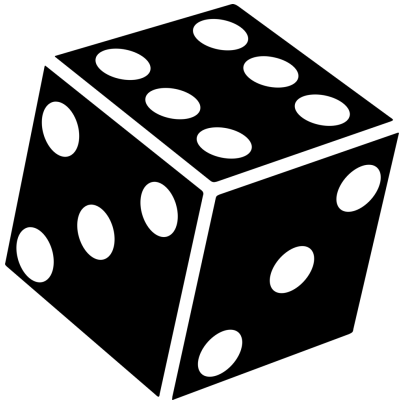
Build & use the QMCPACK development version (preferred, most features)

Set the driver_version in your qmcpack_input.xml:

```
<project id="name_for_calculations" series="0">
<parameter name="driver_version">batch</parameter>
</project>
…
```

Updates required for some parameters and observables e.g. “walkers”->”total_walkers” or “walkers_per_task” in DMC (reduces ambiguity, complexity)

…or use NEXUS!



“If you see something, say something”