QMCPACK USERS WORKSHOP 2019

Argonne NATIONAL LABORATORY

RUNNING QMCPACK EFFICIENTLY AT LARGE SCALE

QMCPACK

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

May 15th, 2019 Oak Ridge, TN, U.S.A.

OUTLINE

- New features
  - Size consistent T-moves
  - Hybrid orbital representation

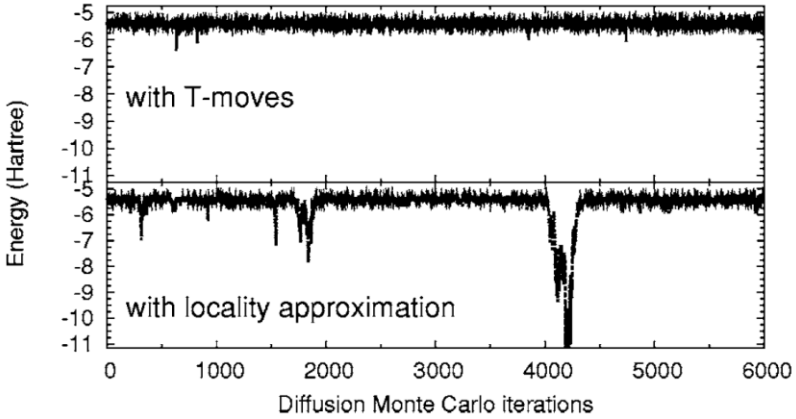
- Performance improvement
  - SoA transformation
  - Efficient walker data movement
  - Delayed update algorithm
  - Nested OpenMP threading
  - Thread affinity tool

NEW FEATURES

SIZE CONSISTENT T-MOVES

Needed for pseudopotential

- More stable DMC calculation
- Recover variational principle
- In DMC input section, nonlocalmoves=no, yes(v0), v1, v3
  - no, locality approximation
  - yes/v0 based on PRB 2006, not size consistent
  - v1, based on JCP 2010, size consistent. v2, not implemented
  - v3, an approximation v1, same accuracy with both light and heavy element.

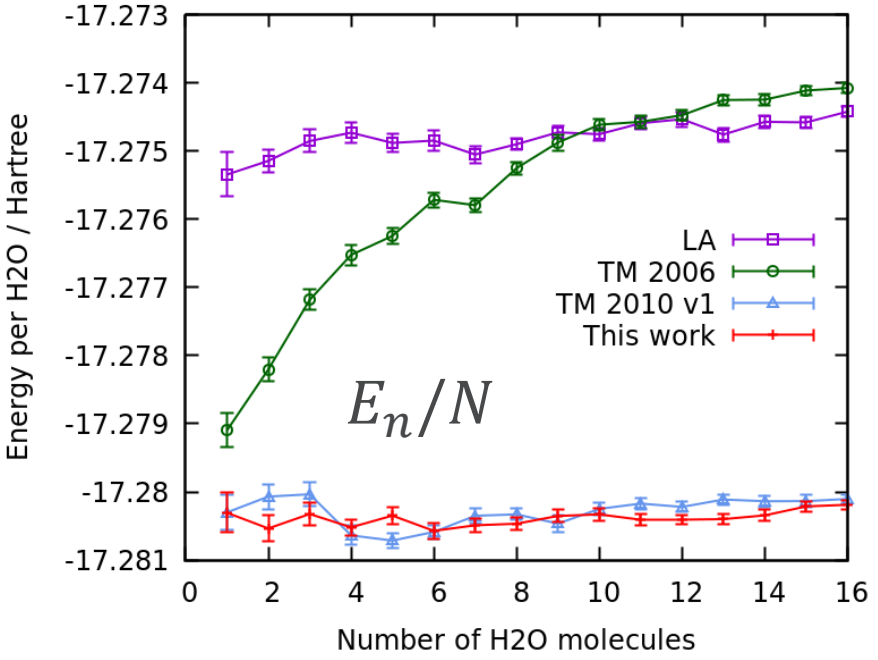


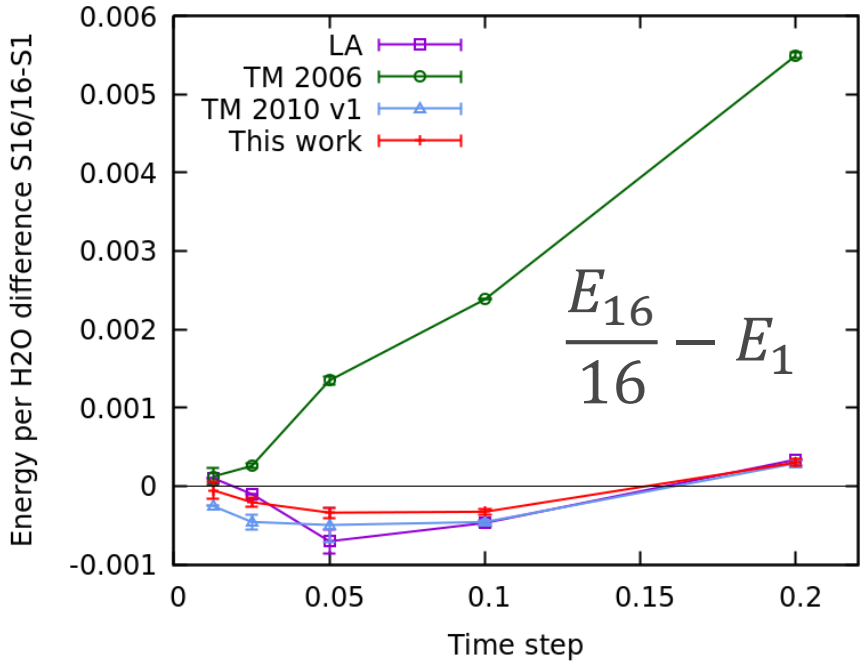
M. Casula, Phys.Rev.B 74, 161102(R) 2006
M. Casula et al, J.Chem.Phys. 132, 154113 (2010)

Computational cost no <= v0 < v3 << v1

SIZE CONSISTENT T-MOVES

Consistent energy per H2O molecule as the number increases

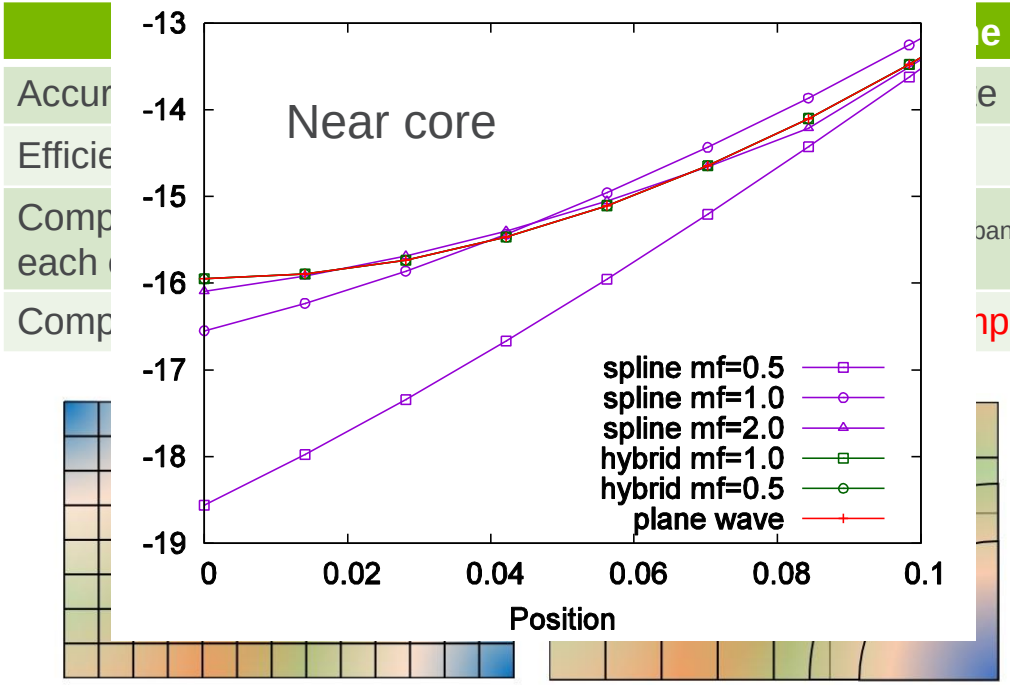




HYBRID ORBITAL REPRESENTATION

Retain high accuracy with much less memory

- Regular 3d cubic spline orbitals compute fast at the cost of large memory footprint.
- High bandwidth memory has limited capacity on modern architectures
- Atomic orbital in A, regular spline in C. Buffer region in B.
- The new way gives high accuracy with much less memory needed



HYBRID ORBITAL REPRESENTATION

Easy to use design

- AO conversion is parallelized inside QMCPACK
- A few input control needed
- Use nexus for parameter scan

```
<determinantset type="bspline" source="i"
   href="pwscf.h5"
   tilematrix="1 1 3 1 2 -1 -2 1 0"
   twistnum="-1" gpu="yes"
   meshfactor="0.8" twist="0 0 0"
   precision="single" hybridrep="yes"
   smoothing_scheme="Consistent"
   smoothing_function="LEKS2018">
   ...
</determinantset>
```

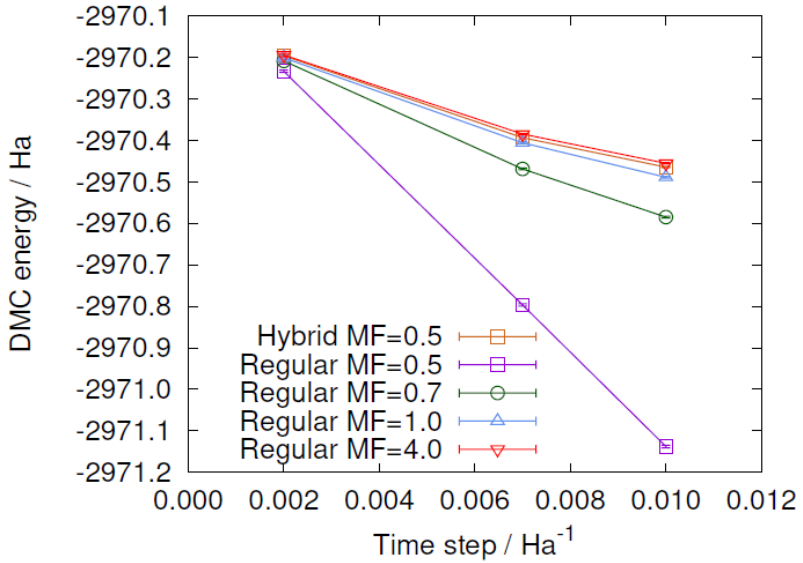
Per specie

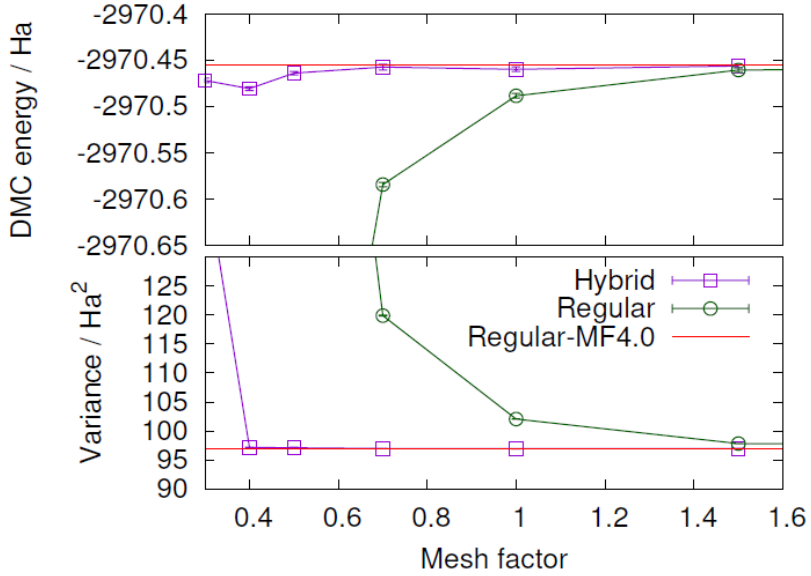
```
<group name="Ni">
   <parameter name="charge">         18 </parameter>
   <parameter name="valence">        18 </parameter>
   <parameter name="atomicnumber" > 28
</parameter>
   <parameter name="cutoff_radius" > 1.6
</parameter>
   <parameter name="inner_cutoff" >  1.3 </parameter>
   <parameter name="lmax" >          5 </parameter>
   <parameter name="spline_radius" > 1.8
</parameter>
   <parameter name="spline_npoints">  91
</parameter>
</group>
```

Red required, Blue optional

HYBRID ORBITAL REPRESENTATION

High accuracy NiO calculation with MF=0.5





8X memory saving

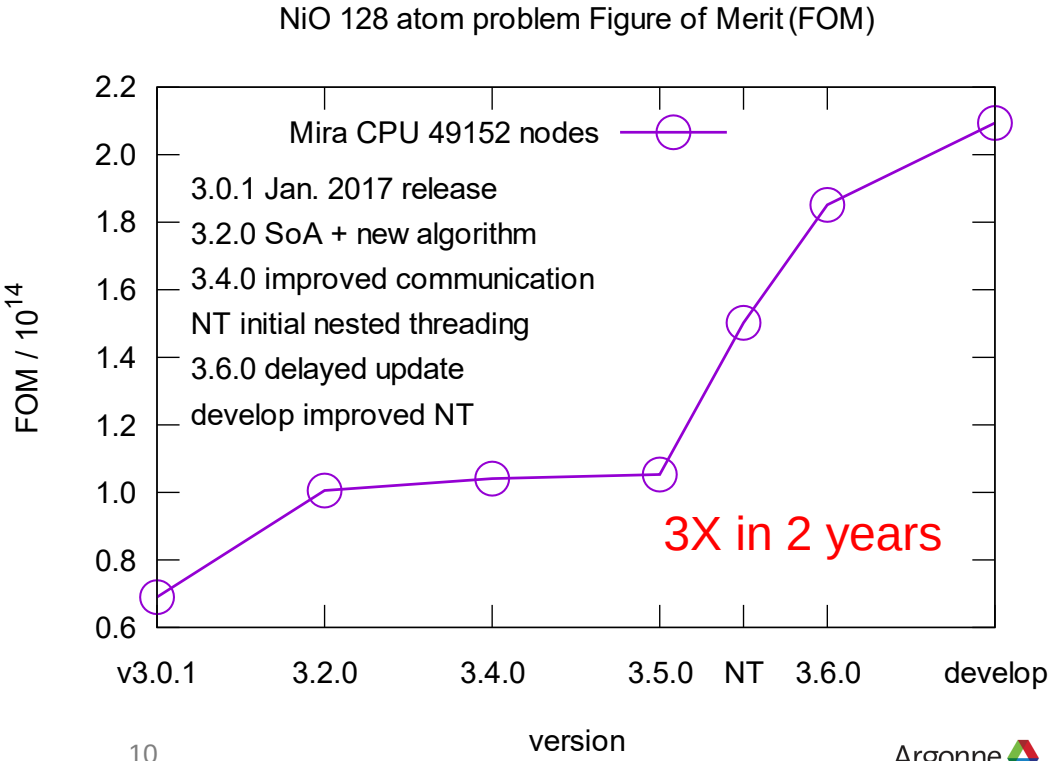
YL et al, J.Chem.Phys. 149, 034108 (2018)

PERFORMANCE IMPROVEMENTS

DRAMATIC IMPROVEMENT ON THE CPU

New parallelization and new algorithms

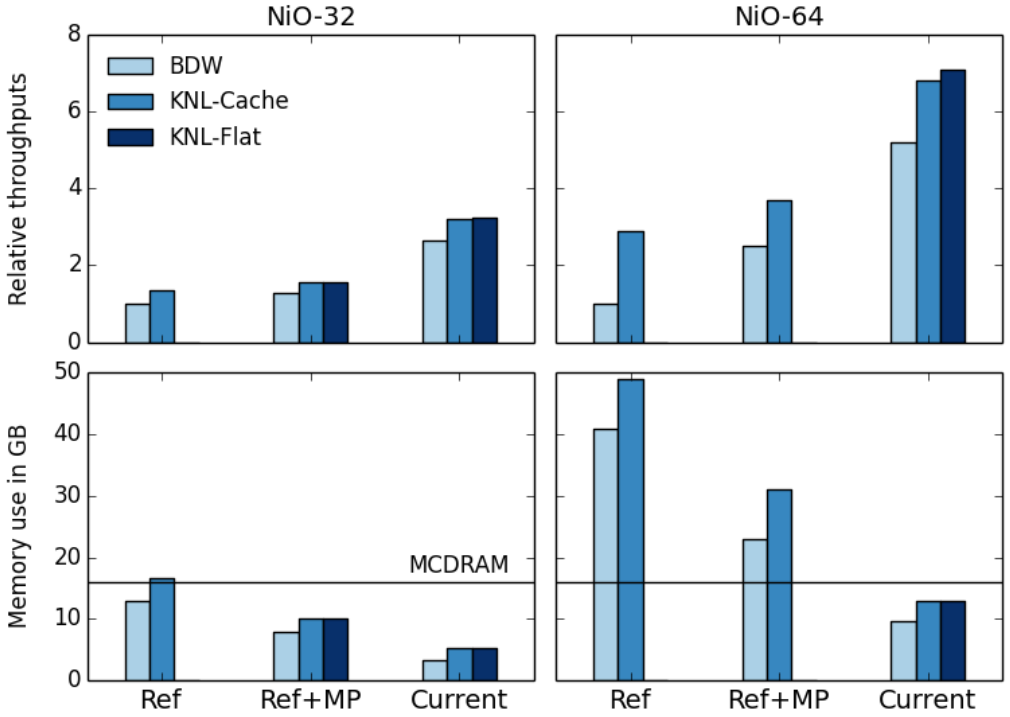
- Mixed precision support on CPU
- All the changes we did are portable to all the CPU architectures. Intel, AMD, IBM, ARM
- Xeon and Xeon Phi has much higher speed-up than 3x



Structure of Array (SoA) transformation

Compile time selection

- Introduced in v3.2 and turned on by default in v3.7
- Leverage CPU SIMD hardware using OpenMP 4.0 SIMD
- Jastrow e-e N^2 memory, Jastrow e-e-I N^3 memory down to N memory
- Jastrow computational cost reduces to a very small fraction



A. Mathuriya, YL et al. pp 38 SC17

EFFICIENT WALKER DATA MOVEMENT

Turn on by default since v3.4

- Keep a memory pool for each walker for the wavefunction data
- Wavefunction objects reference the pools without copying data

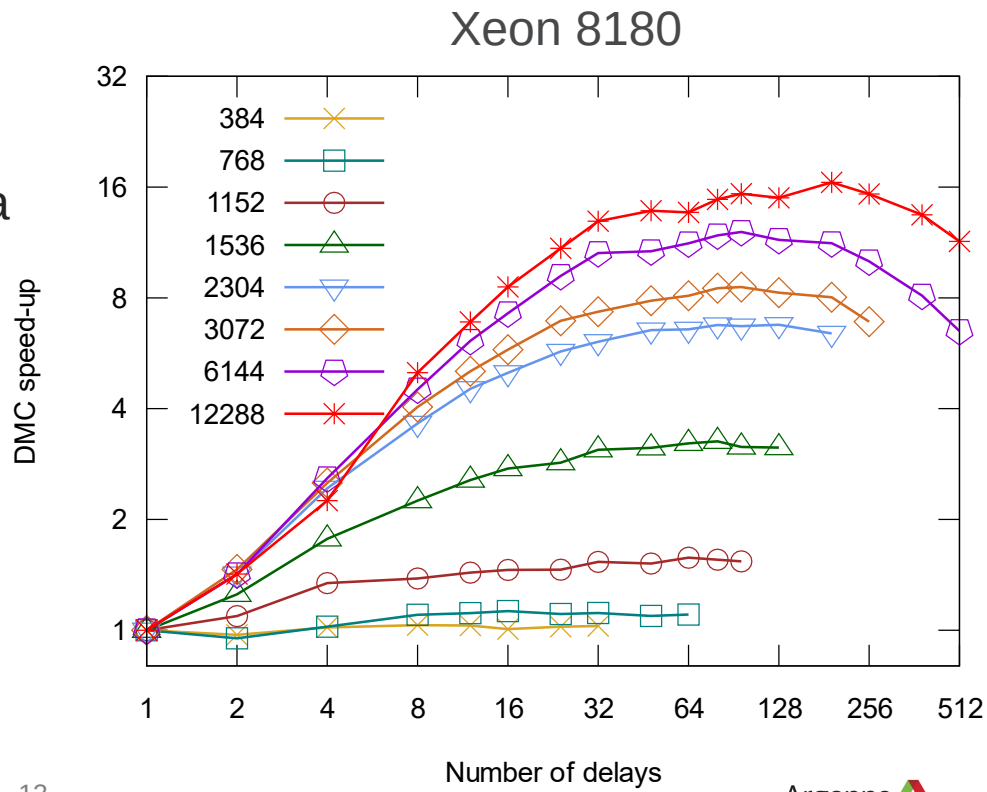
- Walker exchanging within a MPI rank copies memory directly
- Walker exchanging between MPI ranks uses non blocking send/recv pools without packing and unpacking

- 2x speed up on full machine runs on OLCF Titan.

DELAYED UPDATE ALGORITHM

Added since v3.6

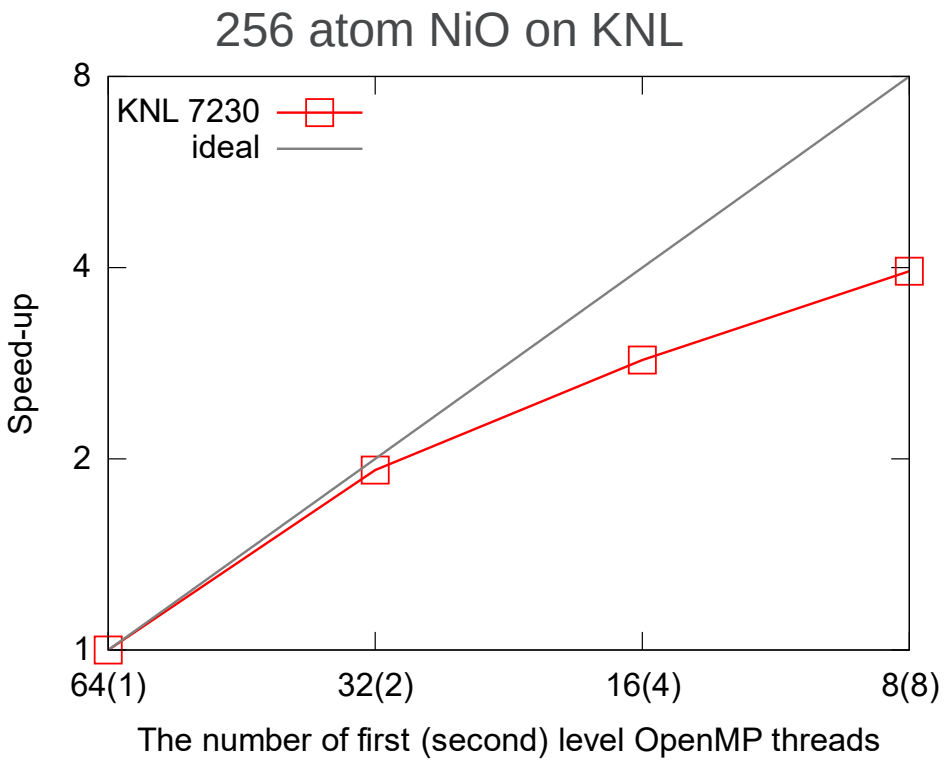
- Change Sherman-Morrison rank-1 update to delayed update using Sherman-Morrison-Woodbury formula
- Matters more on large problem sizes
- <slaterdeterminant delay_rank="32">
- Try starting from 32, the best choice depends on the system and QMC methods.



NESTED OPENMP THREADING

Go beyond 1 core per walker

- The computation of a walker is further accelerated by putting more than 1 core.
- Using two level OpenMP threading
- The speed-up of walkers evolving in one step, $1/t$



THREAD AFFINITY IS IMPORTANT

We provide a tool

- Use qmc-check-affinity to check if the threads are bind to the right cores
- Core ID order is vendor specific. Always check this on a new machine
- Nested OpenMP needs more nobs to control. It is similar to other MPI+OpenMP programs.
- Use one MPI rank per NUMA(socket) if memory allows.

OMP_NESTED=true OMP_PLACES=threads
OMP_PROC_BIND=spread,close OMP_NUM_THREADS=4,4
mpirun -np 1 ./bin/qmc-check-affinity
OpenMP enables 4 1st level threads, and 4 2nd level threads.

MPI rank 0 L1 tid 0 L2 tid 0 is placed on Core ID 0
MPI rank 0 L1 tid 0 L2 tid 1 is placed on Core ID 1
MPI rank 0 L1 tid 0 L2 tid 2 is placed on Core ID 2
MPI rank 0 L1 tid 0 L2 tid 3 is placed on Core ID 3
…
MPI rank 0 L1 tid 3 L2 tid 0 is placed on Core ID 12
MPI rank 0 L1 tid 3 L2 tid 1 is placed on Core ID 13
MPI rank 0 L1 tid 3 L2 tid 2 is placed on Core ID 14
MPI rank 0 L1 tid 3 L2 tid 3 is placed on Core ID 15

BATCHED NLPP EVALUATION

- Target batched nonlocal pseudo potential evaluation for hardware efficiency.
- Input example
  - <pairpot name="PseudoPot" type="pseudo" source="i" wavefunction="psi0" format="xml" algorithm="batched">
- Useful for hybrid orbital representation, nest threading.

MISC

- New QE releases with pw2qmcpack
  - https://github.com/ye-luo/q-e/releases

FUTURE DEVELOPMENT

- One code path with complete features for CPU and GPU.
  - New scheme has been prototyped.
  - Algorithm changes on QMCPACK facilitate this
  - Goal: efficient execution for small, medium and large problems.
- More observables.

Q&A