CAAR Porting Experience: QMCPACK

OLCF Summit Training Workshop

Andreas F. Tillack, Ying Wai Li, Paul R. C. Kent, Ed D’Azevedo, Tjerk P. Straatsma

December 6, 2018
QMCPACK on Summit

• QMCPACK: Accurate quantum mechanics based simulation of materials, including high temperature superconductors.

• QMCPACK runs correctly and with good initial performance on up to 1024 nodes (>20% Summit)
• A single Summit node is 50-times faster than a Titan node for this problem, indicating a ~3.7x increase in the complexity of materials (electron count) computable in the same walltime as Titan.

• Summit exceeds performance gains expected based on peak flops by a factor of 1.57x (Summit vs. Titan node)
Runtime trace

- Single electron update step (for set of walkers)
- Majority of work is typically in matrix inverse update (Blas-2, rank-1 update)
- Update from previous step overlaps with CPU portion of current step
- Wave function expressed using splines – uses majority of memory
New developments for better Summit utilization

- **Delayed updates** increase compute intensity on GPUs (Blas-2 → Blas-3)

- Wave function spline buffer is significant but static portion of QMCPACK memory. **Spline splitting** over multiple GPUs can increase available GPU memory (6 x 16GB = 96 GB)

- Spline splitting needs multiple MPI ranks on a node to access multiple GPUs: Cuda MPS, IPC pointers, and jsrun resource sets
Compiling can be tricky

summit> cmake -DCMAKE_C_COMPILER="mpicc" -DCMAKE_CXX_COMPILER="mpicxx" -DCMAKE_CXX_FLAGS="-std=c++11 -O3" -DBLAS_blas_LIBRARY="$OĽCF_MAGMA_ROOT/lib/libmagma.so" -DLAPACK_lapack_LIBRARY="$OĽCF_MAGMA_ROOT/lib/libmagma.so" -DBLAS_essl_LIBRARY="$OĽCF_ESSL_ROOT/lib64/libessl.so" -DQMC_CUDA=1 -DCUDA_ARCH="sm_70" -DBUILD_LMYENGINE_INTERFACE=0 ..

• Finding the set of libraries that work and give good performance was first challenge

• Some older libraries may need updated config.guess and config.sub (relying on autoconf ./configure) or other tweaks to compile on Summit

• In general, user support is fantastic and quick to help
And then there was jsrun ...

- Default: One GPU per MPI rank with 6 MPI ranks per node:
  
  ```bash
  jsrun --rs_per_host 6 --nrs ${NMPI} -c7 -g1 ./qmcpack <args>
  ```

- Split splines: All six GPUs visible per MPI rank, 6 MPI ranks per node/resource set:
  
  ```bash
  #BSUB -allocate_flags gpumps
  jsrun --tasks_per_rs 6 --nrs ${NNODES} -c42 -g6 -bpacked:7 -dpacked ./qmcpack <args>
  ```

- Binding was single most important consideration for getting performance
Visualized

One GPU per MPI rank with 6 MPI ranks per node

```
jsrun --rs_per_host 6 --nrs ${NMPI} -c7 -g1 ./qmcpack <args>
```

All six GPUs visible per MPI rank, 6 MPI ranks per node/resource set

```
jsrun --tasks_per_rs 6 --nrs ${NNODES} -c42 -g6 -bpacked:7 -dpacked ./qmcpack <args>
```
Summit overall experience

• The hardware is amazing

• The software has come together nicely over time, I am sure user feedback will make it even better

• jsrun is not mpirun
  – Default settings currently are optimized towards least resource usage (no GPU, everything goes onto core 0)
  – Must specify exact resource usage, placement and binding in order to get expected behavior and performance
Thank you for your attention!

Acknowledgments

- Frank Winkler, GWT-TUD GmbH (ScoreP & Vampir support)
- Ronny Brendel, formerly GWT-TUD GmbH (ScoreP & Vampir support), now Nvidia
- Jeff Larkin, Nvidia (development & coffee support)
- Steve Abbott, Nvidia (nvprof hair-pulling)
- Eric Lixiang Luo, IBM (SummitDev support)
- Bob Walkup, IBM (Minsky/SummitDev support)