Quantum Monte Carlo Calculations of Nucleon Systems and Cold atom gases

Scaling and Code Performance

Project participants:
PIs: Kevin E. Schmidt, Arizona State University;
Stefano Gandolfi, Los Alamos National Laboratory

March 28, 2019

1 Scaling

We present the scaling of our Auxiliary Field Diffusion Monte Carlo (AFDMC) code in several systems, but of course the most relevant information for this allocation is the scaling on XSEDE resources.

1.1 Stampede2

During the current allocation (TG-PHY160027), we computed the scaling of our code on Stampede2. Up to 4096 ??? cores, the largest number tested, the code scales strongly.

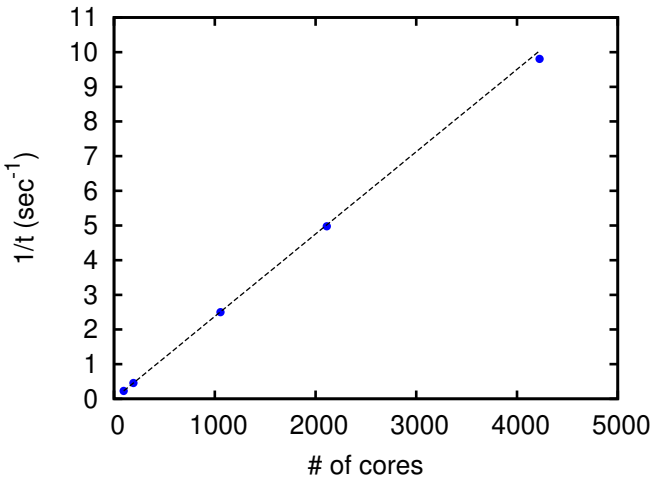

Figure 1: Scaling on Stampede2 using the time to propagate 10,000 configurations of an ^{16}O nucleus for 100 steps.

1.2 SuperMIC

We verified that our code scales strongly on SuperMIC, Fig. 2, up to 2560 cores (the largest number tested).

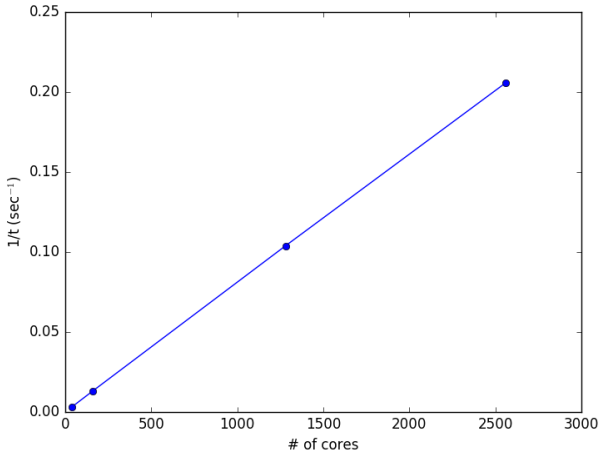

Figure 2: Scaling on SuperMIC using the time to propagate 10,000 configurations of an ^{16}O nucleus for 100 steps.

For the runs shown in Fig. 2, less than 16MB of RAM was utilized. During branching the number of walkers can change. A rebalancing of walkers over different CPUs is performed when the number of walkers fluctuates by more than 1% per CPU. The percentage of walkers that get moved was typically less than 5%.

1.3 Other systems

Although heavy load-balancing is involved, AFDMC shows very good strong scaling up to 96,000 cores on Edison at National Energy Research Scientific Computing Center (NERSC) [1], and up to (at least) 24,000 cores on Mustang[2]. We have also tested the performance on Mira at Argonne National Laboratory (ANL) [3], with good scaling up to 128,000 cores, Fig. 3. The code is very portable, and is being used extensively on Los Alamos National Laboratory (LANL) Institutional Computing resources Wolf, Pinto and Mustang, and in the past on Lobo, Conejo and Mapache [4]. In addition, it is being used at NERSC on Edison, Hopper and Carver, Fig. 4.

[1] Edison is a Cray XC30, with a peak performance of 2.57 petaflops/sec, 133,824 compute cores, 357 terabytes of memory, and 7.56 petabytes of disk. `https://www.nersc.gov/users/computational-systems/edison/`

[2] Mustang - 1,600 compute nodes, 38,400 cores, 353 TF system. 24-core AMD Opteron w/ InfiniBand. `http://www.lanl.gov/asc/tri-lab-resources.php`

[3] Mira, 10-petaflops IBM Blue Gene/Q system, consisting of 48 racks 786,432 processors, and 768 terabytes of memory, `https://www.alcf.anl.gov/mira`

[4] `http://www.lanl.gov/asc/tri-lab-resources.php`

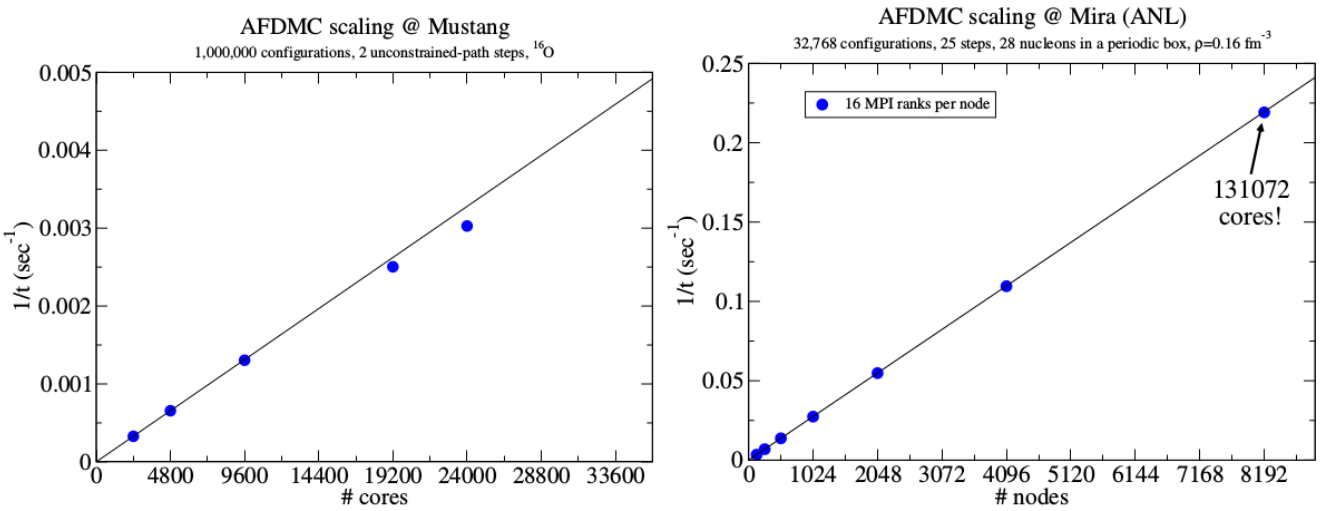

Figure 3: Efficiency of the AFDMC code. On Mustang we tested the unconstrained- path AFDMC for the ^{16}O (left panel), and the constrained-path version using fewer configurations on Mira for 28 nucleons in a box (right panel).

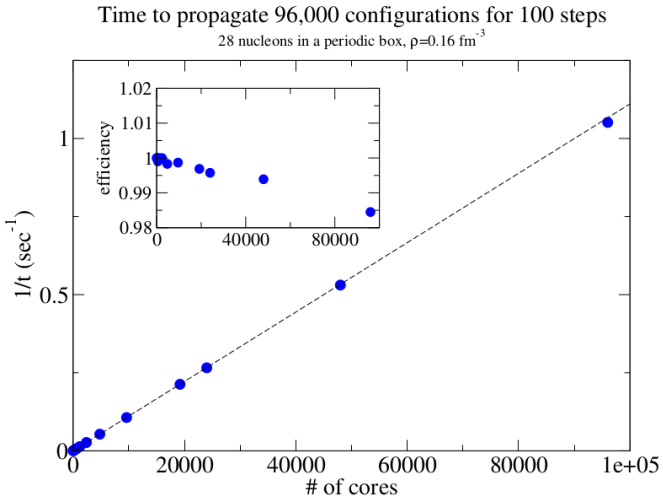

Figure 4: Scaling of the AFDMC code obtained on the Cray XC30 machine at NERSC.

2 High performance computing

The AFDMC code is ready to run on Stampede2, however we plan to address vectorization and data reuse, threading and checkpoint/restart capabilities in the future.

1. **Vectorization & data reuse:** The MPI-only version of AFDMC currently shows, on average, 20% floating point vector utilization based on Allinea MAP profile output, which is good, but we feel can be improved. During an Open Science Intel hackathon, using Intel vTune and compiler optimization reports, we were able to identify several hot spots and inhibitors to vectorization. Using well-placed pragmas, loop fissioning, and data restructuring, we were able to quickly see localized performance gains. We plan to continue using these

tools to identify opportunities for increased vectorization.

2. **Threading:** Currently, several configurations are distributed to each rank and, at each time step, we perform a dynamic load rebalancing since some configurations branch and others are removed. We plan to reduce the frequency (or need) of load rebalancing at each time step, by implementing an MPI+OpenMP programming model, so that each rank can receive a larger number of configurations and minimize the impact of the workload fluctuation within a node. The reduction in the time spent for load rebalancing and localization of data will significantly reduce our run time and improve our throughput.

3. **Checkpoint/restart and I/O:** AFDMC has checkpoint/restart capabilities using POSIX calls and an aggregator routine written in Fortran. At user-selected intervals, state information is aggregated and checkpoint/restart file(s) are written, with aggregators determined by the number of total configurations. A full configuration output file can be written at user defined intervals, however it is not written often due to the time required for I/O. The configuration output file can be used for additional post-processing and analysis. During a simulation, the observables can be calculated in two ways: they can be evaluated for each configuration on-the-fly, or can be saved and read in a separate post-processing calculation of the observables.

Computational Campaign: Observables including the momentum distribution are calculated every several hundred steps, along with statistical errors. Because of the short-range nature of the diffusion at each step, careful attention must be used to obtain statistically independent samples of the proton and neutron radii. To achieve sufficient statistics for the energy, AFDMC requires about 100,000 steps and at least 20,000 configurations. In the loop described above, it is necessary to calculate the complete wave function, Ψ, several times. The computational cost scales as N^3 to calculate Ψ, with N being the number of particles. The calculation of other observables (including the energy) adds another factor proportional to N^2 to the computational cost.