LSMS

Summary Version
1.1

Purpose of Benchmark
Single node performance with focus on dense linear algebra and parallel scaling efficiency to full system.

Characteristics of Benchmark
WL-LSMS is a code to perform first principles ground state calculations of solid state systems and statistical physics calculations with a focus on magnetic systems. This benchmark will combine these two aspects of the code characterize both single node performance and full system scalability.

The code, at the top level, is parallelized over Wang-Landau (WL) Monte-Carlo walkers. As the WL requires the update of the global density of states, the current implementation in WL-LSMS executes all WL walkers on a master node and sends configurations to individual LSMS instances that perform the computationally intensive first principles calculation. The LSMS first principles calculation parallelizes over individual atom or groups of atoms and its performance is largely determined by dense matrix operations, in particular complex matrix inversion, providing good predictions for linear algebra dominated performance.

Mechanics of Building Benchmark
The code uses both C++ and Fortran. The specific compiler options and libraries are specified in a file (`architecture.h`) include by the `Makefile`. (Examples are available in the `architecture/` directory.) In addition to the standard linear algebra libraries BLAS and LAPACK, WL-LSMS requires the serial HDF5 library.

Mechanics of Running Benchmark
Please also refer to the `README` file in the WL-LSMS directory.

1. Small problem: single node and/or single CPU
   Not required for scaling benchmark. As preparation and for testing and optimization of the WL-LSMS scaling benchmarks it is possible to execute a single LSMS instance without the WL part. (LSMS is limited to using #MPI Ranks * #OMP Threads <= #Atoms)

e.g. for 16 atoms use the input from LSMS/Test/Fe16:
`mpirun -np 1 lsms i_lsms` (if using openmp on the node)
`mpirun -np 16 lsms i_lsms` (if using MPI only)

The number of atoms can be changed in the i_lsms file by changing the integer parameters xRepeat, yRepeat, and zRepeat. The number of atoms is 2* xRepeat*yRepeat*zRepeat.

2. Medium problem: (<1K node) job
   Using 1024 Atoms/Walker, 32 MPI ranks/walker: (using openmp on node)
   ```
   mpirun -np 961 wl-lsms -i i_lsms -mode 1d -size_lsms
   1024 -num_lsms 30 -num_steps 600
   ```

3. Large Titan problem:
   580 walkers:
   ```
   mpirun -np 18561 wl-lsms -i i_lsms -mode 1d -
   size_lsms 1024 -num_lsms 580 -num_steps 11600
   ```

4. CORAL class problem:
   a. Scalable Science: At least 4X performance of full Sequoia/Titan job using weak scaling:
   ```
   mpirun -np 74240 wl-lsms -i i_lsms -mode 1d -
   size_lsms 1024 -num_lsms 2320 -num_steps 46400
   ```

Scaling metric:
Number of Steps / Walltime

Verification of Results

The validation number is the average band energy written at the end of a wl-lsms run:

```
...
Finished all scheduled calculations. Freeing resources.
Energy mean = 4325.68Ry
 WL-LSMS finished in 26618.4 seconds.
 Monte-Carlo steps / walltime = 0.133479/sec
```

Energy mean has to be 4.225 +/- 0.032 Ry per atom (i.e. for 1024 atoms 4326.400 +/- 32.768 Ry)