MILCmk

Summary Version
1.0

Purpose of Benchmark
These are some representative microkernels for the MIMD Lattice Computation (MILC) collaboration code. They test the compiler's ability to optimize code that uses complex arithmetic on small vectors and matrices with sizes that are not powers of 2. They also tend to be bandwidth limited for array sizes that are larger than the cache and can be a good test of streaming and prefetching.

Characteristics of Benchmark
The MILC code is designed to address fundamental questions in high-energy and nuclear physics such as the study of the mass spectrum of strongly interacting particles, the weak interactions of these particles, and the behavior of strongly interacting matter under extreme conditions. MILC simulates the fundamental theory of the strong nuclear force, quantum chromodynamics (QCD), formulated on a four-dimensional space-time lattice. The QCD gluon field is sampled using a Hybrid Monte Carlo algorithm, which requires the solution of a large sparse system at every step. The microkernels are some of the more heavily used routines taken from the QCD Linear Algebra (QLA) library developed under SciDAC. The routines perform some basic math operations among arrays of 3x3 complex matrices and length 3, 6 or 12 complex vectors.

The test programs run benchmarks of 7 kernels for various array sizes in single and double precision. In a real application the typical array sizes would correspond to about 16k elements per core, though the optimal value will depend greatly on the overall balance of the system (including network), and can still vary quite a bit depending on the problem being run. As a representative sample, the Mflops numbers should be reported for sizes of roughly 1k, 16k and 256k elements per core, or a similar range that is appropriate for the proposed architecture. As a further guide, on a current 10PF system, a full machine run would have about 12G elements in total. Weak scaling to a 320PF system would give 384G elements for the whole system. If that system had 64k nodes, then the typical size would be around 6M elements per node.

Mechanics of Building Benchmark
Simply edit the Makefile variables CC (C compiler), COPT (optimization flags) and OMP (flag to enable OpenMP) and make. This will produce the two executables qla_bench-qla-1.7.1-f3 and qla_bench-qla-1.7.1-d3 for single and double precision, respectively.

Mechanics of Running Benchmark

There are no command line parameters, so one can simply run each executable as a single process on a node, setting OMP_NUM_THREADS to the appropriate value, if necessary. For example:
OMP_NUM_THREADS=64 ./qla_bench-qla-1.7.1-f3

Verification of Results

Example outputs of a run on 1 node of a BG/Q are in
bench-qla-1.7.1-f3.output
bench-qla-1.7.1-d3.output

Within a run of the executable, it performs benchmarks for several array lengths. The output contains results for each of the form

```
len = 4096
len/thread = 64
QLA_V_vpeq_M_times_pV           :      430813 time= 0.28 mem=21394 mflops=10697
QLA_V_veq_Ma_times_V            :     49.5022 time= 0.35 mem=16798 mflops= 9239
QLA_V_vmeq_pV                   :    -56361.7 time= 0.46 mem=18245 mflops= 1520
QLA_D_vpeq_spproj_M_times_pD    :      644470 time= 0.43 mem=14248 mflops= 6649
QLA_M_veq_M_times_pM            :     181.402 time= 0.23 mem=20257 mflops=18569
QLA_r_veq_norm2_V               :      126976 time= 1.02 mem= 5861 mflops= 2930
QLA_c_veq_V_dot_V               :  1.6123e+10 time= 1.34 mem= 4470 mflops= 2235
```

len is the total array length for the process
len/thread is the array length per OepnMP thread

The next 7 lines contain the summary for each of the kernels tested, of the form
<kernel name> : <checksum> time=<time> mem=<mem> mflops=<mflops>

checksum is a simple sum of the result vector, and can be used as a basic test for correctness. It need not agree exactly, and can differ due to roundoff when optimizing if the order of operations is changed.

time is the total time for that benchmark in seconds. Each benchmark is repeated several times to ensure that the total time is large enough to get an accurate measurement (usually around 1 second). If the total time is too small, then the number 9.e9 that appears in qla_bench.c line 211:

```
double cf = 9.e9/n;
```

may need to be increased to get accurate measurements.

mem is the total memory bandwidth measured (including loads and stores) in GB/s.

mflops is the total flop rate measured in Mflop/s.