<u>NERSC-8 Capability Improvement Run Rules</u>

As described in section 3.5.5 of the technical requirements, NERSC will require the installed system to provide a nominal 8x capability improvement over NERSC's Hopper system (6,384 nodes, 153,214 cores). For NERSC, capability improvement will be judged by the results of three of the benchmarks supplied for the initial RFP: GTC, MILC, and miniDFT.

For each benchmark, NERSC will provide a benchmark problem that utilizes, nominally, two-thirds (2/3) of Hopper's current compute partition. The Offeror will then scale each benchmark problem, using the guidance provided in the README's for each benchmark and duplicated below, to utilize, nominally, two-thirds of the compute partition of the installed system. The capability metric for each benchmark will then be calculated as the product of run-time speedup and the increase in problem size. The capability improvement for each benchmark will then be arithmetically averaged to yield a capability improvement metric for the installed system.

As an example, one may scale each physical dimension (nx,ny,nz) of the MILC benchmark by a factor of 2, yielding an 8x increase in problem size. If running this problem on the installed system results in a run time speedup of 1.2, the overall capability improvement metric for this benchmark is 1.2*8 = 9.6.

Benchmark Guidance:

For each benchmark:

a. It is expected that the Offeror will use the MPI+X execution model and abide by the run rules for the same as described in the RFP run rules document.
b. Each benchmark's problem size can be increased as described. The number of MPI tasks and threads used to run the benchmark will be chosen by the vendor to provide the best performance. To reach the desired capability of the installed system it is expected a combination of increased problem size and increased concurrency will be required.
c. NERSC is the sole judge of the successful completion of each benchmark and the correctness of each benchmark's capability metric.
d. Any code modifications made must pass the verification tests as described in the RFP.

For convenience, we reproduce below the guidance provided in the README for each benchmark on how to increase the problem size for each of the provided capability benchmarks.

GTC:

Capability improvement runs are enabled by increasing three parameters in the input file. For the 19200 MPI rank large case these have the values

```
  micell=30000
  mecell=30000,
  npartdom=300,
```

To increase the size of the problem:

1) Increase npartdom. The total number of MPI ranks = 64*npartdom

2) Increase micell & mecell simultaneously. They should be equal to 100*mpartdom

For example, to increase the max MPI concurrency by a factor of 3 over the large problem nicell=mecell=90000 & npartdom=900. In this case the increase in problem size will be 3 and the capability increase will be 3 times the runtime change.

MILC:

For the capability improvement runs, you will need to scale up the large problem (which is, itself, a weak scaled version of the small problem). For the 24576 MPI rank large problem, the size of the four-dimensional space-time lattice is controlled by the following parameters in the input deck:

```
nx 64
ny 64
nz 128
nt 192
```

As an example, to weak scale an 8x8x8x8 (nx x ny x ny x nt) problem, one can begin by multiplying nt by 2, then nz, then ny, then nz so that all variables get sized accordingly in a round robin fashion. As mentioned, the large problem is a weak scaled version of the small problem (16x16x16x24). Decomposing the large problem and following the rule just mentioned, we can see that the last variable to be updated was nz. Thus, to continue scaling the large problem, your next option is to multiply ny by 2, then nx, and then nt, and then nz, and so forth.

Note that scaling the problem to a greater number of tasks may result in run-time stability issues where the code may report 'singularity error' during the second phase of the computation. It may be possible to eliminate this error by reducing the micro-canonical time step in the parameter list for the second phase (line 34 in the file benchmark_n8/large/n8_large.in). For the large benchmark, the current value of this parameter is 0.02. Testing by NERSC indicates reducing this parameter by a factor of 10 may eliminate this error without noticeably reducing the amount of work done by the code.

miniDFT:

Capability improvement measurements are enabled by increasing the number of k-points used in the large test case. The k-point grid is specified on the last two lines of large.in:

```
K_POINTS automatic
nk1 nk2 nk3 1 1 1
```

To increase the number of k-points, adjust the (integer) parameters nk1, nk2 and nk3, which determine the size of the k-point integration grid. The number of k points increases (roughly) linearly with the product nk1 * nk2 * nk3, though a significant fraction of these points are excluded due to symmetry. Grep for "number of k points" to determine the actual number of k-points used. The increase in capability for the capability improvement calculation is the increase in the number of k-points used for the large problem (1).

The rules for the capability improvement measurement are the same as the MPI+X case (D.1.b). The -npool command line argument should be set to the num ber of k-points.

Advice to vendors: The number of k-points is printed at the beginning of a MiniDFT run, but cannot be easily counted beforehand to set -npool. A reasonable solution is to initiate a trial-run with -npool=1, determine the number of k-points, cancel the trial-run, and restart with an appropriate value for npool.