KRIPKE

Summary Version

1.2.2-CORAL2 results and benchmark tarball are for git commit a12bce7 from the release/v1.2.0-CORAL2 branch of the Kripke repo at https://github.com/LLNL/Kripke/releases/download/v1.2.2-CORAL2/kripke-v1.2.2-CORAL2-a12bce7.tar.gz

Purpose of Benchmark

Test parallel performance of structured grid discrete ordinates particle transport.

Characteristics of Benchmark

Kripke is a simple, scalable, 3D Sn deterministic particle transport code. Its primary purpose is to research how data layout, programming paradigms and architectures effect the implementation and performance of Sn transport. A main goal of Kripke is investigating how different data-layouts affect instruction, thread and task level parallelism, and what the implications are on overall solver performance.
See a more detailed description of the code in the README.md file.

Mechanics of Building Benchmark

Step 1: Create a build space (assuming you are starting in the Kripke root directory)

```
mkdir build
```

Step 2: Run CMake in that build space

```
cd build
cmake ..

For BG/Q, we have a special cache init file that makes things easier:

cd build
cmake .. -C../host-configs/bgqos.cmake
```

Step 3: Now make Kripke:

```
make -j8
```

Step 5: Run Kripke's default problem:

```
./bin/kripke.exe
```

Mechanics of Running Benchmark

The base problem is a 1-node problem sized for a single BG/Q node, and consumes about

- 64 energy groups
- 128 directions (angles)
- 64x32x32 zones
- 4th order scattering
- 10 iterations
- fully-upwinded sweeps

Here are 2 examples that run kripke, the first uses 1 MPI task with 64 threads, the second uses 16 MPI tasks with 4 threads per task:

- OMP_NUM_THREADS=64 srun -N1 -n1 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 64,32,32 --procs 1,1,1
- OMP_NUM_THREADS=4 srun -N1 -n16 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 64,32,32 --procs 4,2,2

To scale the problem up, we weak-scale the number of zones and MPI ranks, keeping the problem size and decompostion as "cube" like as possible, so for a 64-node problem:

- OMP_NUM_THREADS=64 srun -N64 -n64 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 256,128,128 --procs 4,4,4
- OMP_NUM_THREADS=4 srun -N64 -n1024 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 256,128,128 --procs 16,8,8

For the "1/4 of Sequoia" problem we have:

- OMP_NUM_THREADS=64 srun -N24576 -n24576 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 2048,1024,768 --procs 32,32,24
- OMP_NUM_THREADS=4 srun -N24576 -n393216 ./bin/kripke.exe --groups 64 --gset 1 --quad 128 --dset 128 --legendre 4 --zones 2048,1024,768 --procs 128,64,48

For the target CORAL-2 problem the total number of zones should be increased to be 2048,2048,1536.

Modifications of the following parameters change the problem definition, and should be considered fixed:

- --groups
- --niter

- --sigt
- --sigs
- --pmethod
- --quad
- --legendre
- --zones

Modifications of the following parameters just change problem decomposition, and are fair-game for tuning:

- --gset
- --dset
- --zset
- --procs
- OMP_NUM_THREADS

Figure of Merit (FOM):

There are 4 values printed out at the end of each run: Throughput, Number of unknowns, grind time, and sweep efficiency.

The FOM we are interested in is throughput, which is the number of unknowns solved per second per iteration.

Benchmark Verification:

The benchmark delivers correct results if the 'particle count' and 'change' values printed out during the solve are consistent with the baseline version.

The convergence behavior of this problem will not change as the problem weak scales in the number of zones, however as the problem is refined it will converge to a *slightly* different solution. Changes to the "Final Particle Count" at a given problem size should be considered incorrect behavior.

Figure-of-Merit Data on BG/Q

Nodes	Ranks	Threads /Rank	Zones	Memory (GB)	Solve Time (sec)	Final Particle Count	Throughput (unknowns/ (sec/iter))
1	1	64	64x32x32	10	623	3.742347e+10	8.619823e+06
1	16	4	64x32x32	10	425	3.742347e+10	1.260770e+07
64	64	64	256x128x128	619	632	3.743771e+10	5.437729e+08
64	1024	4	256x128x128	619	434	3.743771e+10	7.912030e+08
4096	4096	64	1024x512x512	39380	666	3.744010e+10	3.302129e+10
4096	65536	4	1024x512x512	39380	468	3.744010e+10	4.700701e+10
24576	24576	64	2048x1024x768	236072	712	3.744099e+10	1.853260e+11
24576	393216	4	2048x1024x768	OOM			

Notes:

- **Solve Time** is reported in the "Timers" report as the "Solve" timer. This includes all runtime except for "Generate", which does the problem setup.
- **Throughput** is reported at the end of the run in the "Figures of Merit" report. This is the total number of unknowns (across all MPI tasks), divided by the "Solve" timer divided by the divided by the number of iterations (10).
- **Final Particle Count** is reported on the last iteration line ("iter 9:") in the "Steady State Solve" output.
- **OOM:** The run of 24576 nodes with 4 threads/rank was unsuccessful, as MPI required too much memory to complete.