HPCG

High Performance Conjugate Gradient benchmark

Table of Contents

1 Introduction

High Performance Conjugate Gradient benchmark version 3.0 was used which contains 7,600 lines of C++ code using MPI and OpenMP. "The High Performance Conjugate Gradient (HPCG) benchmark is a simple program that generates a synthetic sparse linear system that is mathematically similar to a finite element, finite volume or finite difference discretizations of a three-dimensional heat diffusion problem on a semi-regular grid. The problem is solved using domain decomposition with an additive Schwarz preconditioned conjugate gradient method where each subdomain is preconditioned using asymmetric Gauss-Seidel sweep." [1,2]

2 Building HPCG

The overall instructions to obtain and build HPCG are enumerated below.

1. The benchmark (version 3.0) can be downloaded from [icl.utk.edu] [3].
2. Update `Makefile' for local paths and compilers, as needed.
3. `make clean && make all'

[icl.utk.edu] http://www.icl.utk.edu/

3 Running HPCG
==============

There are two sets of command-line inputs that control the important aspects of the benchmark. The first is the size of the local memory mesh for each rank, expressed as `nx', `ny', and `nz'. For example, `nx=160', `ny=160', and `nz=160' expresses a local mesh that is 160x160x160 (i.e., 4,096,000) points per MPI rank. The global mesh is the sum of the local meshes over all MPI ranks.

The second is the arrangement of ranks in the processor mesh, expressed as `npx', `npy', and `npz'. The processor mesh determines the number of total ranks and how efficiently they communicate. For example, `npx=4', `npy=4', and `npz=2' represent a processor mesh that is 4x4x2 (i.e., 32) MPI ranks in size.

HPCG may be run with MPI, or OpenMP, or a combination of both.

3.1 Sample BASH Script Using SLURM
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Below is a sample BASH script utilizing SLURM.

```
,----
| #!/bin/bash
|
| # Select the number of MPI ranks (or tasks)
| # based on the number of nodes (SLURM_NNODES)
| let nodes_per_job=$SLURM_NNODES
| let tasks_per_node=32
| let tasks_per_job=$tasks_per_node*$nodes_per_job
| let threads_per_task=2
|
| # Run xhpcg, make sure PATH includes the directory
| # where xhpcg resides, and LD_LIBRARY_PATH includes
| # all directories containing the required libraries
| export OMP_NUM_THREADS=$threads_per_task
| srun \
|     --ntasks-per-node $tasks_per_node \
|     --ntasks $tasks_per_job \
|     xhpcg --nx=160 --ny=160 --nz=160
`----
```

3.2 Small Case (1 Node)
~~~~~~~~~~~~~~~~~~~~~~~~

An example run of HPCG on a single 2.3 GHz Intel Haswell node with 32 cores, Hyperthreading, and 128 GB of main memory, might use the following commands:

```
,----
| export OMP_NUM_THREADS=2
| srun \
|     --ntasks-per-node 32 \
|     --ntasks 32 \
|     xhpcg \
|     --npx=4 --npy=4 --npz=2 \
|     --nx=160 --ny=160 --nz=160
`----
```

This example uses 32 MPI ranks and 2 OpenMP threads per rank. To use MPI only, set `OMP_NUM_THREADS' to 1. To use only OpenMP, set `npx=1', `npy=1', and `npz=1'. This case uses approximately 3 GB of memory per MPI rank, i.e., 94 GB total. It yields a performance of 10.33 GFLOP/s, i.e., 0.32 GFLOP/s/core.

3.3 Medium Case (32 Nodes)
~~~~~~~~~~~~~~~~~~~~~~~~~~~

An example run of HPCG on a cluster of 32x 2.3 GHz Intel Haswell nodes with 32 cores, Hyperthreading, and 128 GB of main memory, might use the following commands:

```
,----
| export OMP_NUM_THREADS=2
| srun \
|     --ntasks-per-node 32 \
|     --ntasks 1024 \
|     xhpcg \
|     --npx=16 --npy=8 --npz=8 \
|     --nx=160 --ny=160 --nz=160
`----
```

This example uses 1,024 MPI ranks and 2 OpenMP threads per rank. This case uses approximately 3 GB of memory per MPI rank, i.e., 2,999 GB total. It yields a performance of 331 GFLOP/s, i.e., 0.32 GFLOP/s/core.

3.4 Large Case (4,352 Nodes)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

An example run of HPCG on a cluster of 4,352x 2.3 GHz Intel Haswell nodes with 32 cores, Hyperthreading, and 128 GB of main memory, might use the following commands:

```
,----
```

```
| export OMP_NUM_THREADS=2
| srun \
|     --ntasks-per-node 32 \
|     --ntasks 139264 \
|     xhpcg \
|     --npx=64 --npy=64 --npz=34 \
|     --nx=160 --ny=160 --nz=160
`----
```
This example uses 139,264 MPI ranks and 2 OpenMP threads per rank. This case uses approximately 3 GB of memory per MPI rank, i.e., 407,906 GB total. It yields a performance of 40,232.4 GFLOP/s, i.e., 0.29 GFLOP/s/core.

4 HPCG Figure of Merit
=====================

HPCG produces a text file named after the pattern `HPCG-Benchmark_version_date_time.yaml'. The total benchmark memory used for data, in gigabytes, is listed towards the beginning of the file as:
```
,----
| Memory Use Information::Total memory used for data (Gbytes)=memory
`----
```
The Figure of Merit (FOM) is represented as a total GFLOP/s rating and is listed towards the end of the file as:
```
,----
| Final Summary::HPCG result is VALID with a GFLOP/s rating of=rating
`----
```

5 HPCG Run Rules
================

The following enumerated list contains the overall run rules.
1. *Baseline Run* - HPCG must be run with minimal modification. In particular, the Offeror should not modify the code to call optimized math libraries. The Offeror may choose to make modifications such as aggressive compiler and OpenMP directive optimizations as long as the compiler and optimizations are generally available and fully supported on the proposed platform. The source code may be minimally altered to support changes to the directives only. Source code changes used for the benchmark FOM must be reported with the Offeror response. The benchmark must verify correct operation.
2. Baseline and Optimized runs of HPCG must be performed with the benchmark reporting total memory for data of at least 400,000 GB.

6 References
============

1. Jack Dongarra and Michael A. Heroux, "Toward a New Metric for Ranking High Performance Computing Systems," Sandia Report SAND2013-4744, June 2013, [http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf]
2. Michael A. Heroux, Jack Dongarra and Piotr Luszczek, "HPCG Technical Specification," Sandia Report SAND2013-8752, pg 7, October 2013, [https://software.sandia.gov/hpcg/doc/HPCG-Specification.pdf]
3. [http://icl.cs.utk.edu/projectsdev/hpcg/software/browse.html]
4. [http://www.hpcg-benchmark.org/downloads/hpcg-3.0.tar.gz]