Hackathon KAUST GPU 2020

Marcin Rogowski, Suha Kayum & Vincent Etienne

KAUST & Saudi Aramco

November 2020

Content

Content

Overview

hpcscan is a C++ code for benchmarking HPC kernels (mainly for solving PDEs with FDM)

- Simple code struture based on individual test cases
- Easy to add new test cases
- Main class is Grid: multi-dimension (1, 2 & 3D) Cartesian grid
- Hybrid MPI/OpenMP parallelism
- All configuration parameters on command line
- Support single and double precision computation
- Compilation with standard Makefile
- No external librairies
- Follows C++ Google style code

Overview

hpcscan embeds several test cases

Current version 1.0

- General operations on grids
- Memory operations
- MPI communication
- FD computation
- Basic wave propagator

Possible additions for future versions

- Operations on matrices full and sparse
- FFT
- IO
- Compression

Compilation and validation

Compiling hpcscan
go to `./build` and `make` (by default compilation with single precision float)
To compile with double precision float, `make precision=double`

Validating hpcscan
go to `./script` and `sh runValidationTests.sh`

Table: `runValidationTests.sh` [1]

Machine	Compiler	Single prec.	Double prec.
Mars	g++ 9.3.0	764 PASS / 0 FAIL / 0 ERR / 20 WARN	764 PASS / 0 FAIL / 0 ERR / 20 WARN
Shaheen	icpc 19.0.5.281	764 PASS / 0 FAIL / 0 ERR / 20 WARN	764 PASS / 0 FAIL / 0 ERR / 20 WARN

Numbers can differ due to availability of features depending on the platforms

[1] Updated Nov 25, 2020

Content

Test platform - Shaheen II (KAUST)

Machine Shaheen II / Cray XC40

- Computing nodes Intel Haswell 2.3 Ghz dual socket (16 cores / socket)
- RAM 128 GB with Peak memory BW 136.5 GB/s
- Peak performance Single Prec. 2.36 TFLOP/s / Double Prec. 1.18 TFLOP/s
- Interconnect Cray Aries with Dragonfly topology
  - 60 GB/s optical links between groups
  - 8.5 GB/s copper links between chassis
  - 3.5 GB/s backplane within a chassis
  - 5 GB/s PCIe from node to Aries router

Content

Test Case Grid - Description

- Fill grid (W = coef)
- L1 error between grid W and R
- Get min. grid W
- Get max. grid W
- Update pressure W = 2*U - W + C*L (used in propagator)
- Medium Grid size 4 GB (1000 x 1000 x 1000 points)

Test Case Grid - Results

- **Machine: shaheen**
- 1 node / 32 threads
- CPU Baseline kernel and GPU kernels

Table: Bandwidth GB/s [2]

Mode	Fill	L1 err.	Get max.	Get min.	Update Pres.
CPU	54	124	126	126	120
GPU1	54	124	126	126	120
GPU2	54	124	126	126	120
GPU3	54	124	126	126	120

Table: Bandwidth GPoints/s

Mode	Fill	L1 err.	Get max.	Get min.	Update Pres.
CPU	13.5	15.5	31.5	31.5	6.0
GPU1	13.5	15.5	31.5	31.5	6.0
GPU2	13.5	15.5	31.5	31.5	6.0
GPU3	13.5	15.5	31.5	31.5	6.0

Reproduce results with `./hackathonTestCases/testCase_Grid/runMediumGridShaheen.sh`
~~Elapsed few seconds.~~

[2] Updated Nov 28, 2020

Test Case Grid - Summary

Machine: Shaheen

- L1 Err., Get Min & Max: 125 GB/s close to peak BW (92 % Peak Mem. BW)
- Low perf for Fill: 54-58 GB/s (40-43 % Peak Mem. BW)
- Max Err. 72-91 GB/s (53-67 % Peak Mem. BW)
- Pressure update 6 GPoint/s (120 GB/s, 88 % Peak Mem. BW)

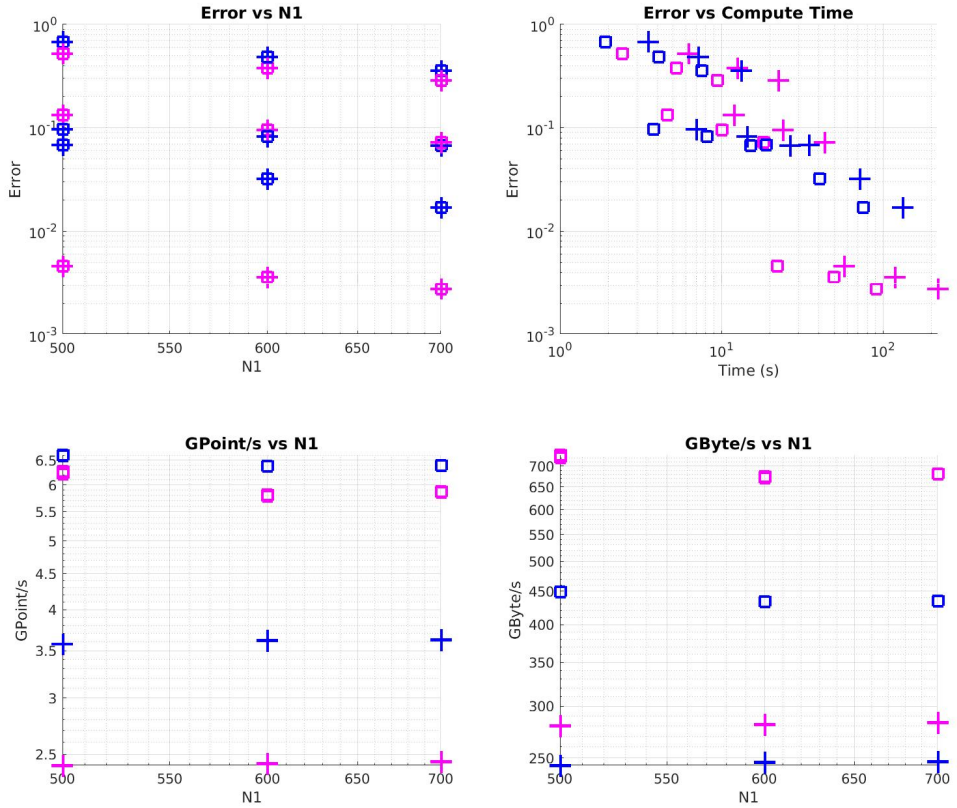
Content

Test Case Propa - Description

- Seismic wave propagator
- 2nd order acoustic wave equation
- Time-domain Finite-difference
- Various FD order in space
- 2nd FD order in time
- Various grid size and time steps
- Total 18 configurations
- Comparison against analytical solution (Eigen mode)

Test Case Propa - Results

- **Machine: shaheen**
- 1 node / 32 threads
- CPU Baseline kernel and GPU kernels [3]
- FD: Black O2, Blue O4, Pink O8, Red O12 / Square=CPU CacheBlk, Cross=GPU



Reproduce results with `./hackathonTestCases/testCase_Propa/runShaheen.sh`

~~Elapsed about an hour.~~

[3]

Content

Conclusions and next steps

TO DO

Content

Acknowledgements

- KAUST ECRC and KSL for access and support on Shaheen II & Ibex