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)
- Max. err. grid W
- L1 err. grid W
- Get min. grid W
- Get max. grid W
- Update pressure (used in propagator)
- Small Grid size 500 MB (500 x 500 x 500 points)
- Medium Grid size 4 GB (1000 x 1000 x 1000 points)

Test Case Grid - Results

- **Machine: shaheen**
- 1 node / 32 threads
- Baseline kernel

Table: Bandwidth GB/s [2]

Grid	Fill	Max. err.	L1 err.	Get max.	Get min.	Update Pres.
Small	58	72	122	125	125	119
Medium	54	91	124	127	127	120

Table: Bandwidth GPoints/s

Grid	Fill	Max. err.	L1 err.	Get max.	Get min.	Update Pres.
Small	14.4	9.0	15.2	31.3	31.2	6.0
Medium	13.4	11.4	15.5	31.8	31.8	6.0

Reproduce results with `./script/testCase_Grid/runSmallGridShaheen.sh` and `./script/testCase_Grid/runMediumGridShaheen.sh`
Elapsed time 5 and 7 sec.

[2] Updated Nov 26, 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 Comm - Description

Measure MPI communication bandwidth

MPI point to point communication

- Send with MPI_Send from proc X to proc 0 (Half-duplex BW)
- Send and receive with MPI_Sendrecv between proc X and proc 0 (Full-duplex BW)

MPI collective communication

- Exhange of halos used in FD kernel with MPI_Sendrecv
- Grid size 1000 x 1000 x 1000
- Domain decomposition with N1 x N2 x N3 subdomains

Test Case Comm - Results

- **Machine: Shaheen**
- 8 MPI processes (1 per computing node)
- Baseline kernel

Table: Bandwidth GB/s [3]

MPI#1	MPI#2	Send	Sendrecv	Halo exch.	Comm. size	Subdomains
0	1	8.5	15.3	-	47 MB	-
0	2	8.3	15.3	-	47 MB	-
0	3	8.6	15.3	-	47 MB	-
0	4	8.5	15.3	-	47 MB	-
0	5	8.2	15.3	-	47 MB	-
0	6	8.5	15.3	-	47 MB	-
0	7	8.6	15.3	-	47 MB	-
All	All	-	-	5.0	128 MB	1 4 2
All	All	-	-	5.1	128 MB	1 2 4
All	All	-	-	2.0	96 MB	2 2 2

Reproduce results with `./script/testCase_Comm/runTestShaheen.sh`
Elapsed time 9 seconds

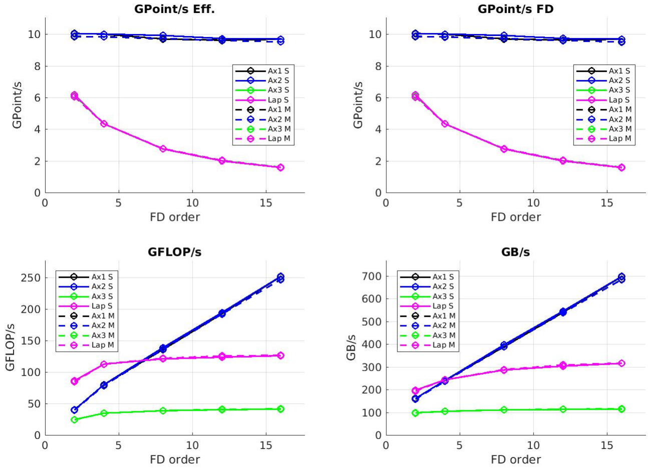
[3] Updated Sep 19, 2020

Content

Test Case FD_D2 - Description

- Computation of second order derivatives with finite-differnce stencil
- Directionnal derivatives
  - Axis 1 $W = \partial^2_{x1}(U)$
  - Axis 2 $W = \partial^2_{x2}(U)$
  - Axis 3 $W = \partial^2_{x3}(U)$
- Laplacian
  - For 2D grids $W = \Delta(U) = \partial^2_{x1}(U) + \partial^2_{x2}(U)$
  - For 3D grids $W = \Delta(U) = \partial^2_{x1}(U) + \partial^2_{x2}(U) + \partial^2_{x3}(U)$
- Stencil order 2, 4, 8, 12 & 16
- Grid size
  - Small 500 x 500 x 500
  - Medium 1000 x 1000 x 1000

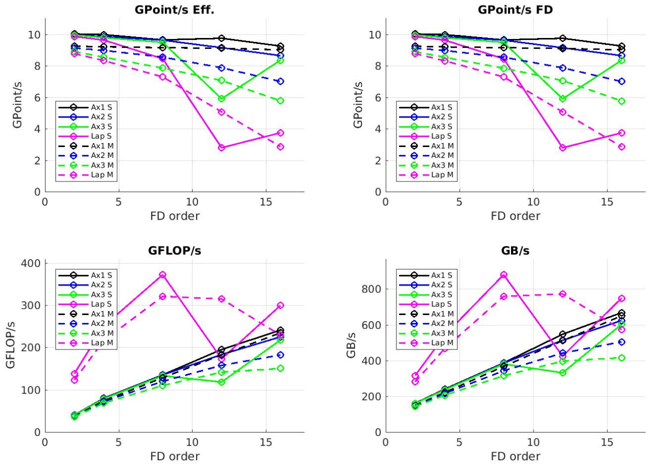
- **machine Shaheen** / 1 node with 32 threads / Baseline kernel [4]
- `./script/testCase_FD_D2/runSmallGridShaheen.sh` & `runMediumGridShaheen.sh`



[4] Updated Sep 26, 2020

Test Case FD_D2 - Results

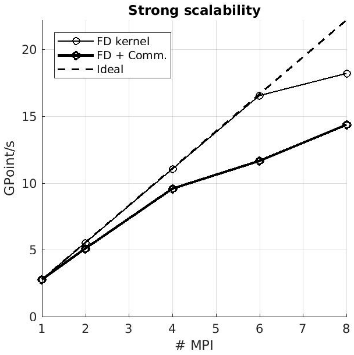
- **machine Shaheen** / 1 node with 32 threads / Cache blocking kernel [5]
- `./script/testCase_FD_D2/runSmallGridShaheen.sh` & `runMediumGridShaheen.sh`

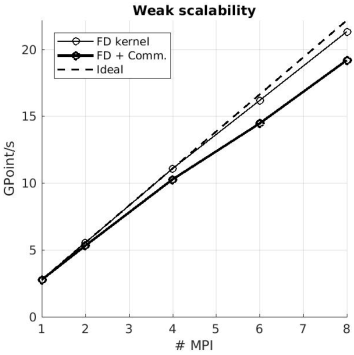


[5] Updated Sep 26, 2020

Test Case FD_D2 - Results

- **machine Shaheen**
- 1 to 8 nodes with 32 threads/node
- Baseline kernel [6]
- Strong scalabity: Grid 1000 x 1000 x 1000 (4 GB)
- Weak scalabity: Grids from 4 GB (1 proc) to 32 GB (8 proc)
- 3D Laplacian O8





[6] Updated Sep 26, 2020

Test Case FD_D2 - Summary

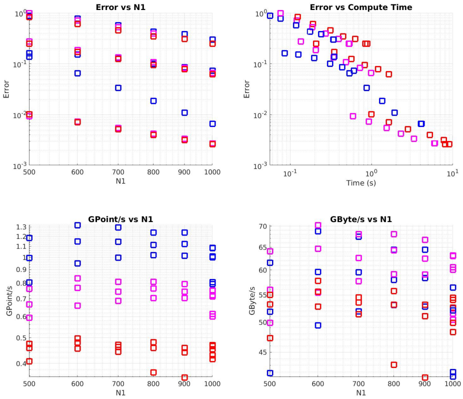
machine Shaheen

- Large benefit of cache blocking
- Significant effect of grid dimnsion and index (very bad performance for n3 without cache blocking)
- Min BW 50 GFLOP/s (∂^2_{x3} O2) $=$ 2 % peak BW [apparent Mem. BW 150 GB/s]
- Max BW 370 GFLOP/s (Δ O8) $=$ 16 % peak BW [apparent Mem. BW 900 GB/s]
- Apparent Mem. BW 150-900 GB/s (110-660 % Peak Mem. BW) $=$ shows data in-cache effect
- Typical stencils of interest for geophysical applications
  - Δ O4 BW $=$ 8-10 GPoint/s
  - Δ O8 BW $=$ 7-9 GPoint/s
  - Δ O12 BW $=$ 3-5 GPoint/s
- Parallel efficiency with 8 nodes 55 to 86 % (depends on workload on Shaheen)

Content

Test Case PropaAc2 - Results

- **machine Mars** / preliminary results [7]
- Eigen mode - 1D model
- FD: Black O2, Blue O4, Pink O8, Red O12 / Square=Baseline
- `./paramAnalysis/propaAccuracy/runMars.sh`



[7] Updated Nov 5, 2020

Content

Conclusions and next steps

TO DO

Content

Acknowledgements

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