

# Hackathon KAUST GPU 2020

Marcin Rogowski, Suha Kayum & Vincent Etienne

KAUST & Saudi Aramco

November 2020

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

- 1 **hpcscan**
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

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

- Simple code structure 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 libraries
- Follows C++ Google style code

## 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`<sup>1</sup>

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

---

<sup>1</sup>Updated Nov 25, 2020

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 **Test platforms**
  - **Shaheen II (KAUST)**
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

# 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





- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 **Test Case Grid**
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

# Test Case Grid - Description

- Fill grid ( $W = \text{coef}$ )
- 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 × 1000 × 1000 points)

# Test Case Grid - Results

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

Table: Bandwidth GB/s <sup>2</sup>

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.

<sup>2</sup>Updated Nov 28, 2020

## 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)

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm**
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

## 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

- Exchange of halos used in FD kernel with MPI\_Sendrecv
- Grid size  $1000 \times 1000 \times 1000$
- Domain decomposition with  $N1 \times N2 \times N3$  subdomains

# Test Case Comm - Results

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

Table: Bandwidth GB/s <sup>3</sup>

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 `sbatch ./script/testCase_Comm/runTestCasesShaheen.sbatch`  
Elapsed time 9 seconds

<sup>3</sup>Updated Sep 19, 2020

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2**
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

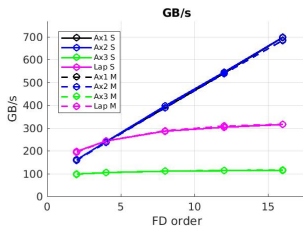
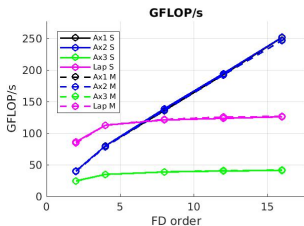
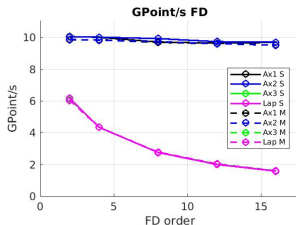
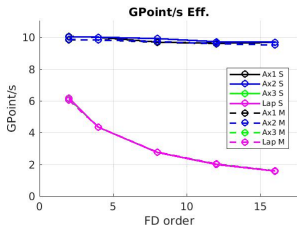


# Test Case FD\_D2 - Description

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

# Test Case FD\_D2 - Results

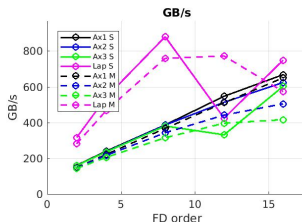
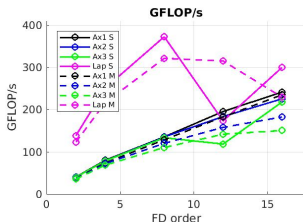
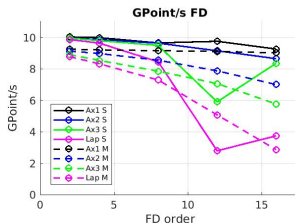
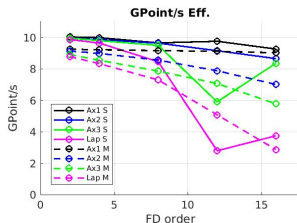
- machine Shaheen / 1 node with 32 threads / Baseline kernel <sup>4</sup>
- ./script/testCase\_FD\_D2/runSmallGridShaheen.sh & runMediumGridShaheen.sh



<sup>4</sup> Updated Sep 26, 2020

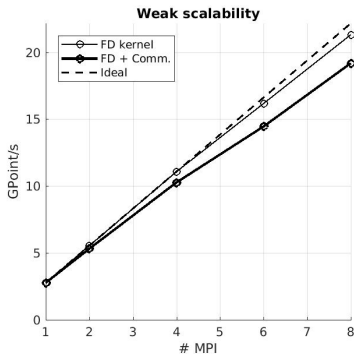
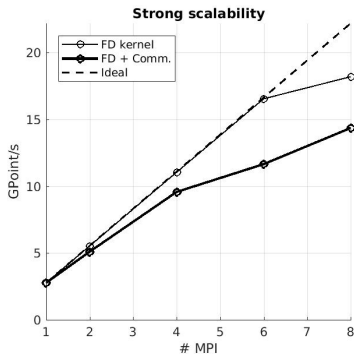
# Test Case FD\_D2 - Results

- machine Shaheen / 1 node with 32 threads / Cache blocking kernel <sup>5</sup>
- ./script/testCase\_FD\_D2/runSmallGridShaheen.sh & runMediumGridShaheen.sh



# Test Case FD\_D2 - Results

- machine Shaheen
- 1 to 8 nodes with 32 threads/node
- Baseline kernel <sup>6</sup>
- Strong scalability: Grid  $1000 \times 1000 \times 1000$  (4 GB)
- Weak scalability: Grids from 4 GB (1 proc) to 32 GB (8 proc)
- 3D Laplacian O8



<sup>6</sup> Updated Sep 26, 2020

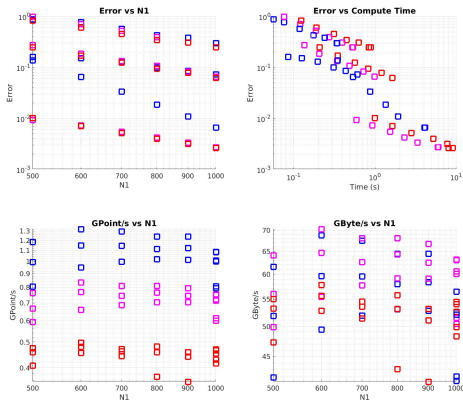
## machine Shaheen

- Large benefit of cache blocking
- Significant effect of grid dimension and index (very bad performance for n3 without cache blocking)
- Min BW 50 GFLOP/s ( $\partial_{x3}^2$  O2) = 2 % peak BW [apparent Mem. BW 150 GB/s]
- Max BW 370 GFLOP/s ( $\Delta$  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
  - $\Delta$  O4 BW = 8-10 GPoint/s
  - $\Delta$  O8 BW = 7-9 GPoint/s
  - $\Delta$  O12 BW = 3-5 GPoint/s
- Parallel efficiency with 8 nodes 55 to 86 % (depends on workload on Shaheen)

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2**
- 7 Conclusions and next steps
- 8 Acknowledgements

# Test Case PropaAc2 - Results

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



<sup>7</sup>Updated Nov 5, 2020

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps**
- 8 Acknowledgements



# Conclusions and next steps

TO DO

- 1 hpcscan
  - Overview
  - Compilation and validation
- 2 Test platforms
  - Shaheen II (KAUST)
- 3 Test Case Grid
- 4 Test Case Comm
- 5 Test Case FD\_D2
- 6 Test Case PropaAc2
- 7 Conclusions and next steps
- 8 Acknowledgements

# Acknowledgements

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