Performance Characterization of a Vector Architecture for Seismic Applications

V. Etienne, A. Momin, EXPEC ARC, Saudi Aramco, Dhahran, Saudi Arabia
L. Gatineau, NEC Deutschland GmbH
S. Momose, NEC Japan

5th EAGE workshop on High Performance Computing for Upstream
6-8 September 2021

Outline

Introduction
Overview of this work and expected outcomes

Seismic Exploration and High Performance Computing
Combining the best of both worlds

Performance Characterization of a Vector Engine
Comparative benchmarks with standard architectures

Conclusions

Introduction

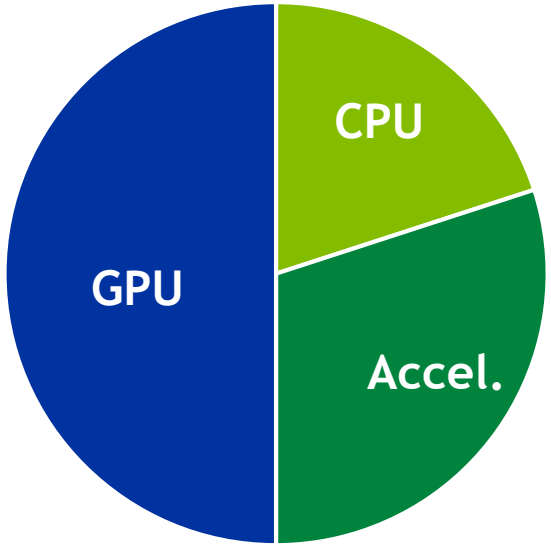
1

Introduction

HPC for the Oil and Gas Industry

source top500.org

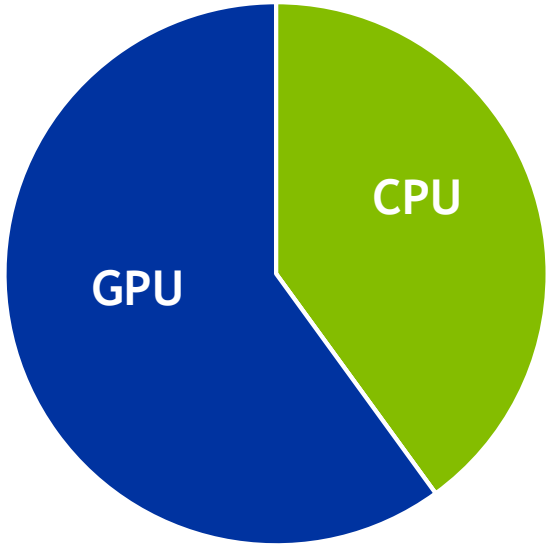
Top 10 - June 2018



3 years ago

- #1 machine GPU based (0.19 EFlop/s)
- 50% machines are GPU based
- **#13 ENI (GPU), #30 TOTAL (CPU), #35 PGS (CPU), 49 ENI (GPU)**

Top 10 - June 2021

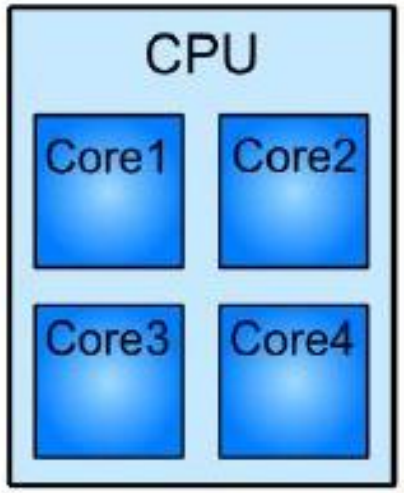


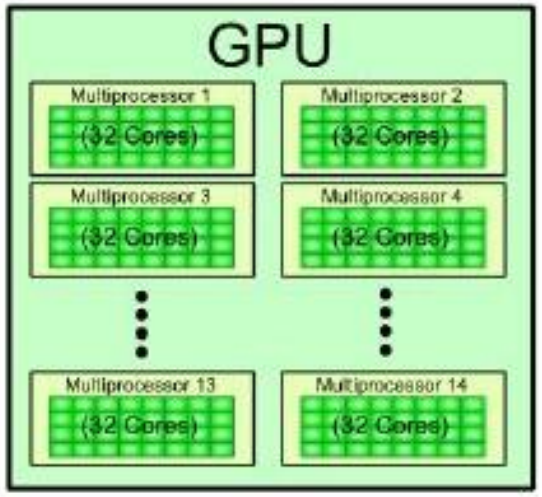
Today

- #1 machine CPU based (0.54 EFlop/s)
- 60% machines are GPU based
- **#9 ENI (GPU), #11 Saudi Aramco (GPU), #19 Saudi Aramco (CPU), #21 Total (GPU)**

Introduction

Standard HPC Architectures





Central Processing Unit

- Few tens of cores with extended capacity and performance
- Large main memory
- Large cache memory / core
- Low memory bandwidth
- Standard programming languages

Current trend: Increase memory bandwidth

Graphics Processing Unit

- Massive parallel device, with thousands of cores executing simple operations
- Limited main memory
- Limited cache memory / core
- High memory bandwidth
- Specific programming language

Current trend: Adopt standard languages

Introduction

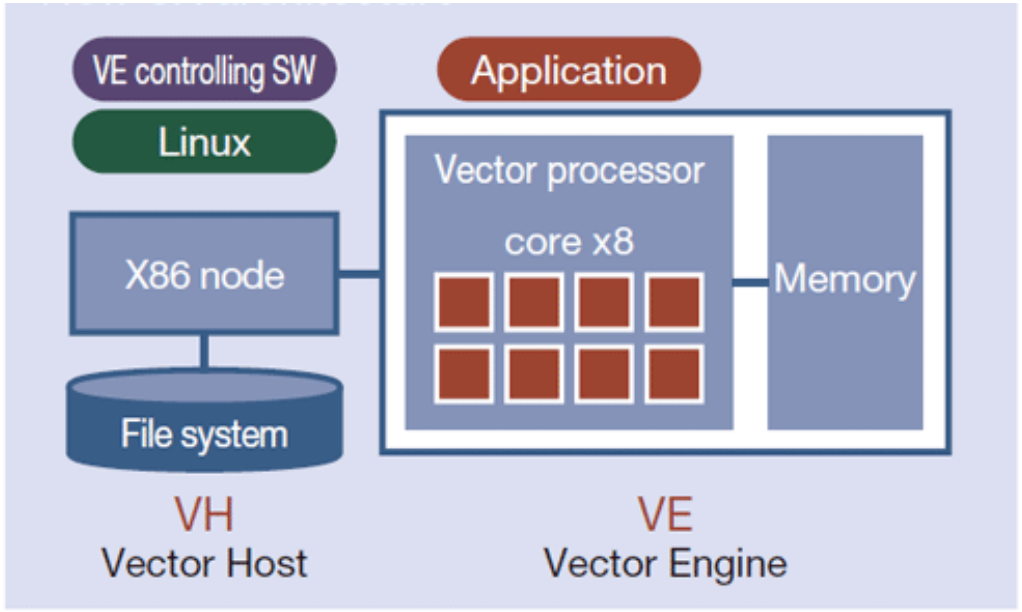
Vector Architecture

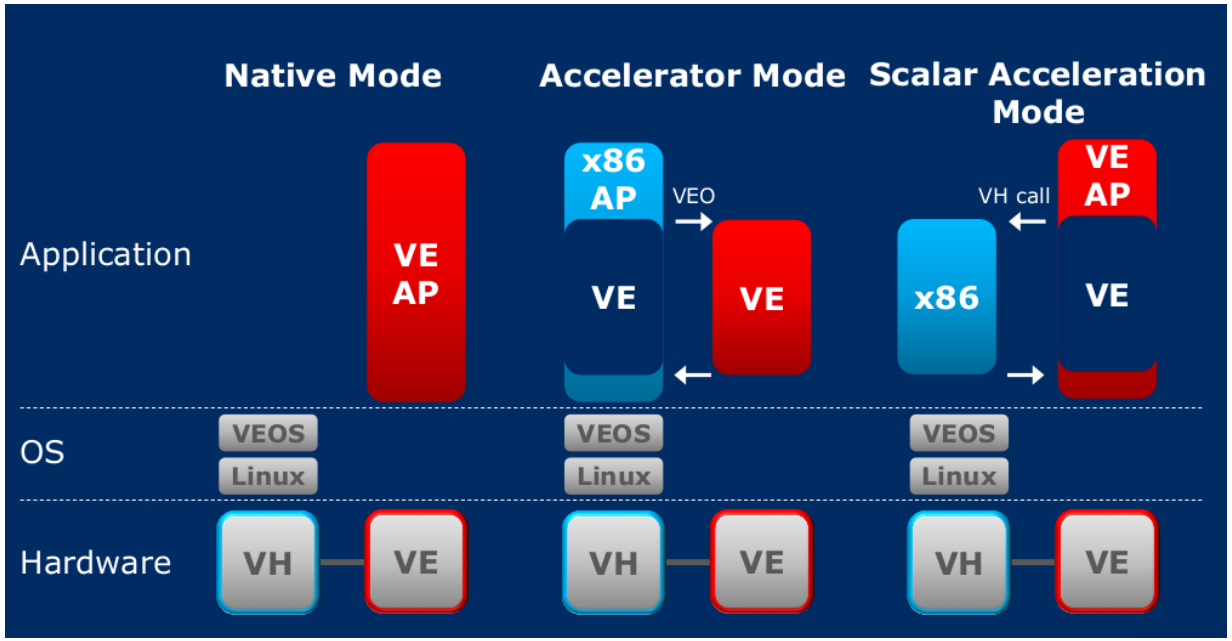
Vector Engine

- Few cores with extended capacity & performance
- Long SIMD vector length
- Limited main memory
- Large cache memory / core
- High memory bandwidth
- Standard programming languages

Combines CPU and GPU strengths

NEC SX-Aurora Vector Engine





3 modes supported
Native (this work)
Accelerator
Scalar accelerator

Introduction

Expected Outcomes of this Work

Our main objective: a quantitative comparison between CPU, GPU and VE

Considering the specificities of our applications, we evaluate:

Computation efficiency > Performance & run time

Energy efficiency > Electric consumption

Usability > Effort to implement & optimize algorithms

In an objective way

With robust protocols and metrics

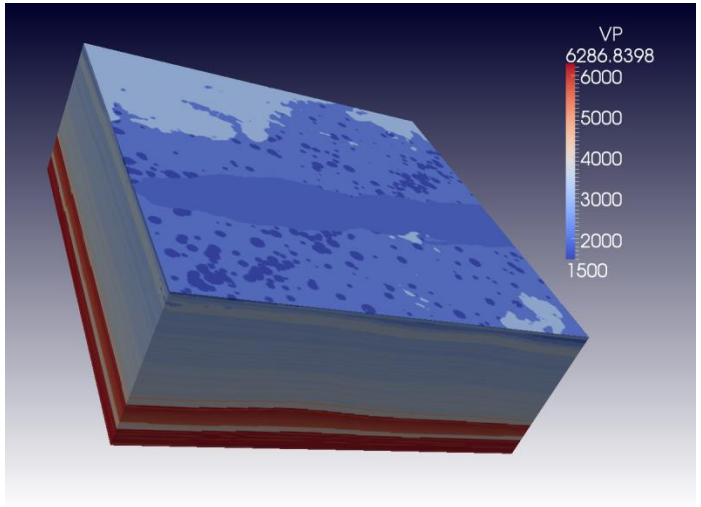
With an open source project available to the community

Seismic Exploration & HPC

2

Seismic and HPC

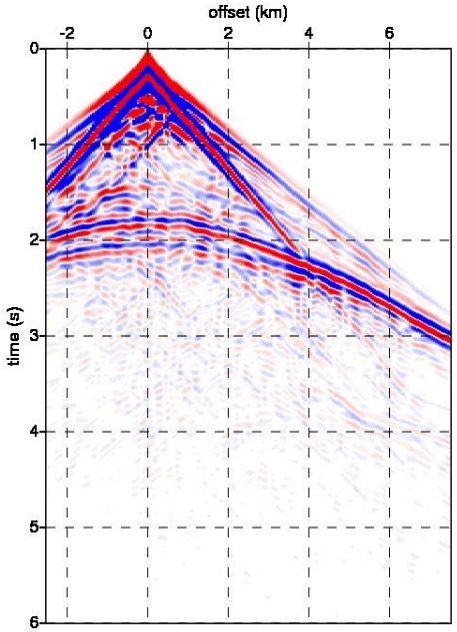
In O&G
2 main applications
Reservoir simulation
Seismic imaging & inversion



Model

Seismic modeling
Forward problem

Seismic inversion
Inverse problem



Data

Compute intensive applications that require HPC systems

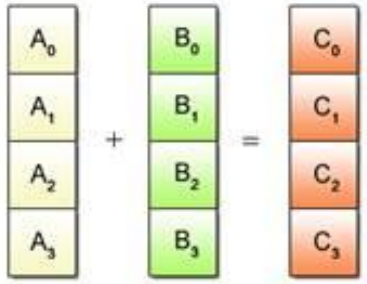
Ex. Elastic modeling in large scale survey about 100 EFlop

HPC

Seismic and HPC

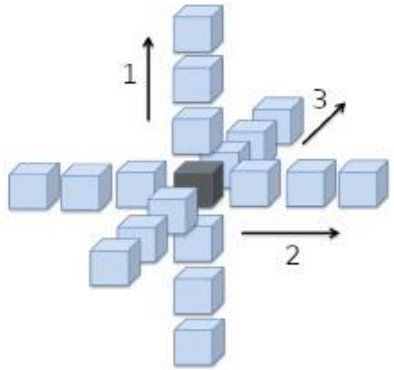
Memory Bandwidth

- Seismic applications performs lots of operations on structured **grids**
- Most of our computational kernels are **memory bounded**



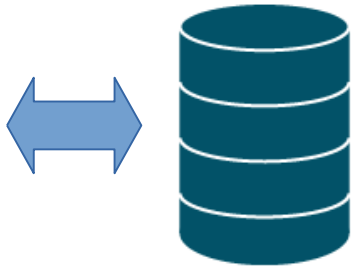
Cache size & efficiency

- Workhorse for seismic: **Finite-difference**
- Large amount of **(in-cache) data reuse**
- Proficiency of compiler vs programmer



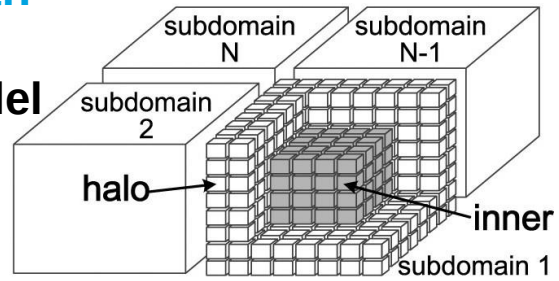
Storage bandwidth

- **Large amount of data** written to and read from disk
- Possibly associated with **compression** algorithm



Interconnect bandwidth

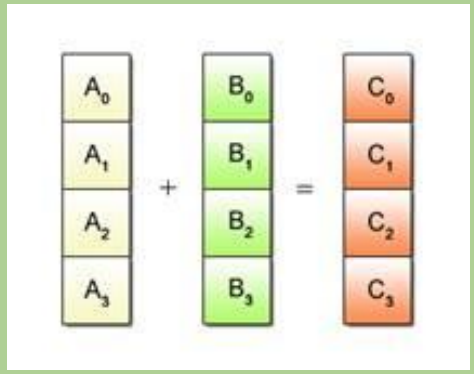
- Our applications are **embarrassingly parallel**
- Interconnect mostly used for **subdomain decomposition** (halo exchange)



Seismic and HPC

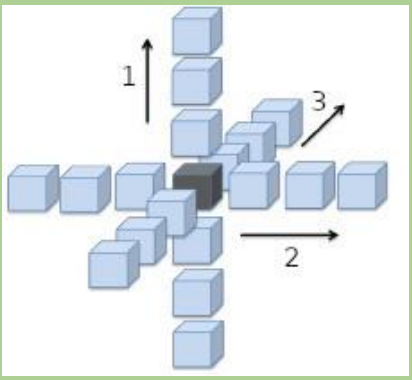
Memory Bandwidth

- Seismic applications performs lots of operations on structured **grids**
- Most of our computational kernels are **memory bounded**



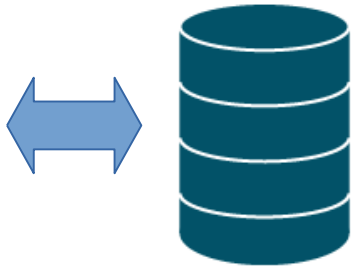
Cache size & efficiency

- Workhorse for seismic: **Finite-difference**
- Large amount of **(in-cache) data reuse**
- Proficiency of compiler vs programmer



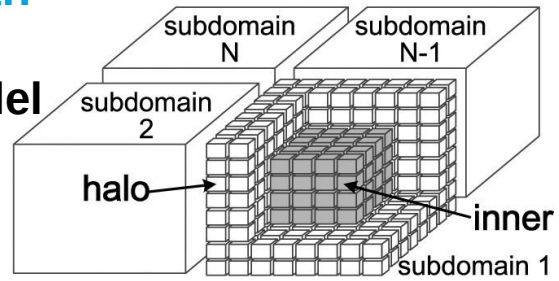
Storage bandwidth

- **Large amount of data** written to and read from disk
- Possibly associated with **compression** algorithm



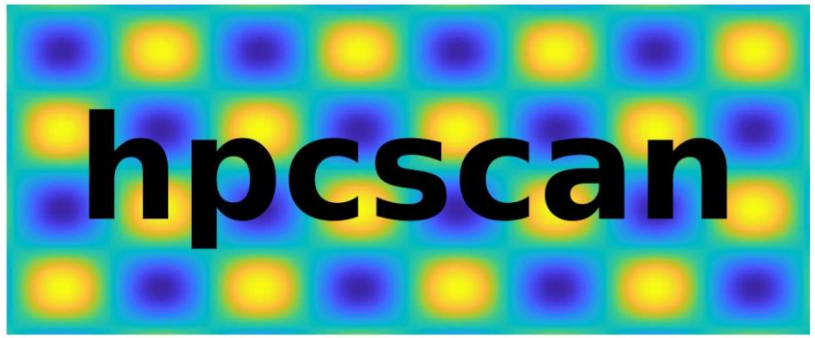
Interconnect bandwidth

- Our applications are **embarrassingly parallel**
- Interconnect mostly used for **subdomain decomposition** (halo exchange)



In this work we will focus on the architecture-related KPIs (memory/cache & flops)

Seismic and HPC



C++ code for benchmarking HPC kernels
with a a focus on Finite Difference

Multi-architectures (CPU, GPU & VE)
Built-in performance, accuracy and energy measurements

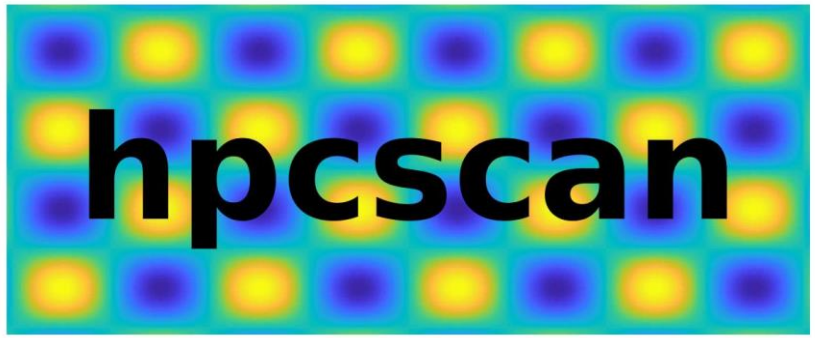
Available on GitHub
No confidentiality: ideal for external collaborations

Main concept: not focus on full applications (too complex)
Instead, perform a 'scanning' of most sensitive kernels

Simple to use and customize

```
mpirun -n 1 hpcscan -testMode Baseline -testCase Memory -dim 3 -n1 1000 -n2 1000 -n3 1000
```

Seismic and HPC



C++ code for benchmarking HPC kernels
with a a focus on Finite Difference

Multi-architectures (CPU, GPU & VE)
Built-in performance, accuracy and energy measurements

Available on GitHub
No confidentiality: ideal for external collaborations

Main concept: not focus on full applications (too complex)
Instead, perform a 'scanning' of most sensitive kernels

Simple to use and customize

```
mpirun -n 1 hpcscan -testMode Baseline -testCase Memory -dim 3 -n1 1000 -n2 1000 -n3 1000
```

- Baseline
- CacheBlk
- CUDA, CUDA_Opt
- DPC++
- HIP, HIP_Opt
- NEC, NEC_SCA

- Memory
- Grid
- FD_D2
- Comm
- Propa

Performance characterization

3

Performance

Hardware specifications
Representative products available on the market

Spec.	CPU	GPU	VE
# cores	2 x 38	5 120	8
Memory type	DDR4	HBM2	HBM2
Capacity GB	256	32	48
Peak GB/s	410	900	1351

Measured memory bandwidth

Case	CPU		GPU		VE	
	GB/s	% peak	GB/s	% peak	GB/s	% peak
Fill	200	49	725	81	1060	78
Copy	300	73	616	68	1085	80
Add	301	73	694	77	1083	80
Mul	300	73	695	77	1084	80
AddUpd	294	72	691	77	1083	80

Architectures reach 70-80 % of their peak bandwidth
Measured bandwidth is a key indicator to assess performance of kernels

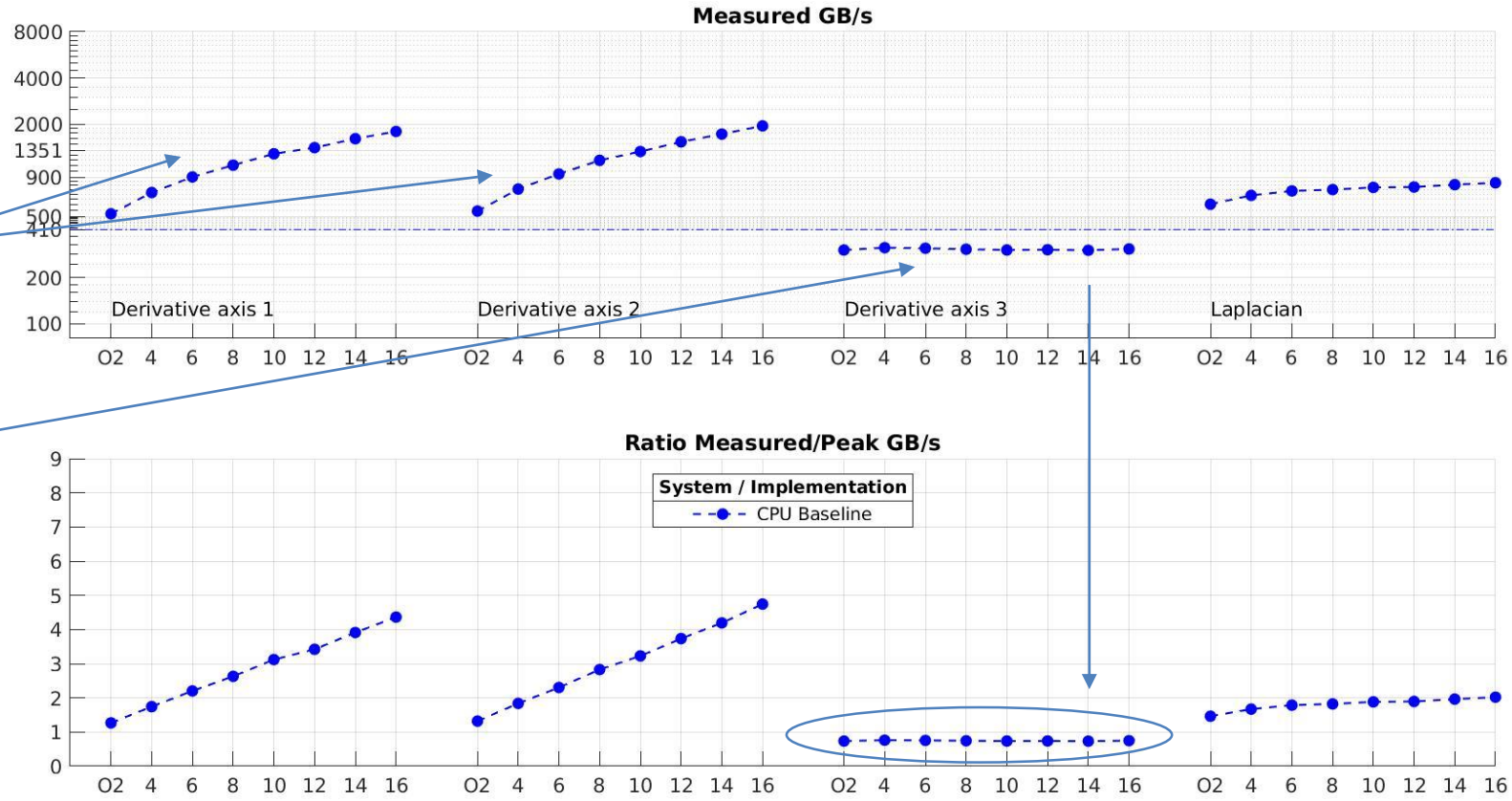
Performance

Starting point & Reference: CPU Baseline

Axis 1 & 2: High data reuse

Axis 3: Low data reuse
Large stride between points
►**Characteristic pathology**
No cache effect
Limited by Memory BW

Laplacian impacted by axis 3



Efficiency increases with FD order except Axis 3 and Laplacian
How to treat pathology on Axis 3?

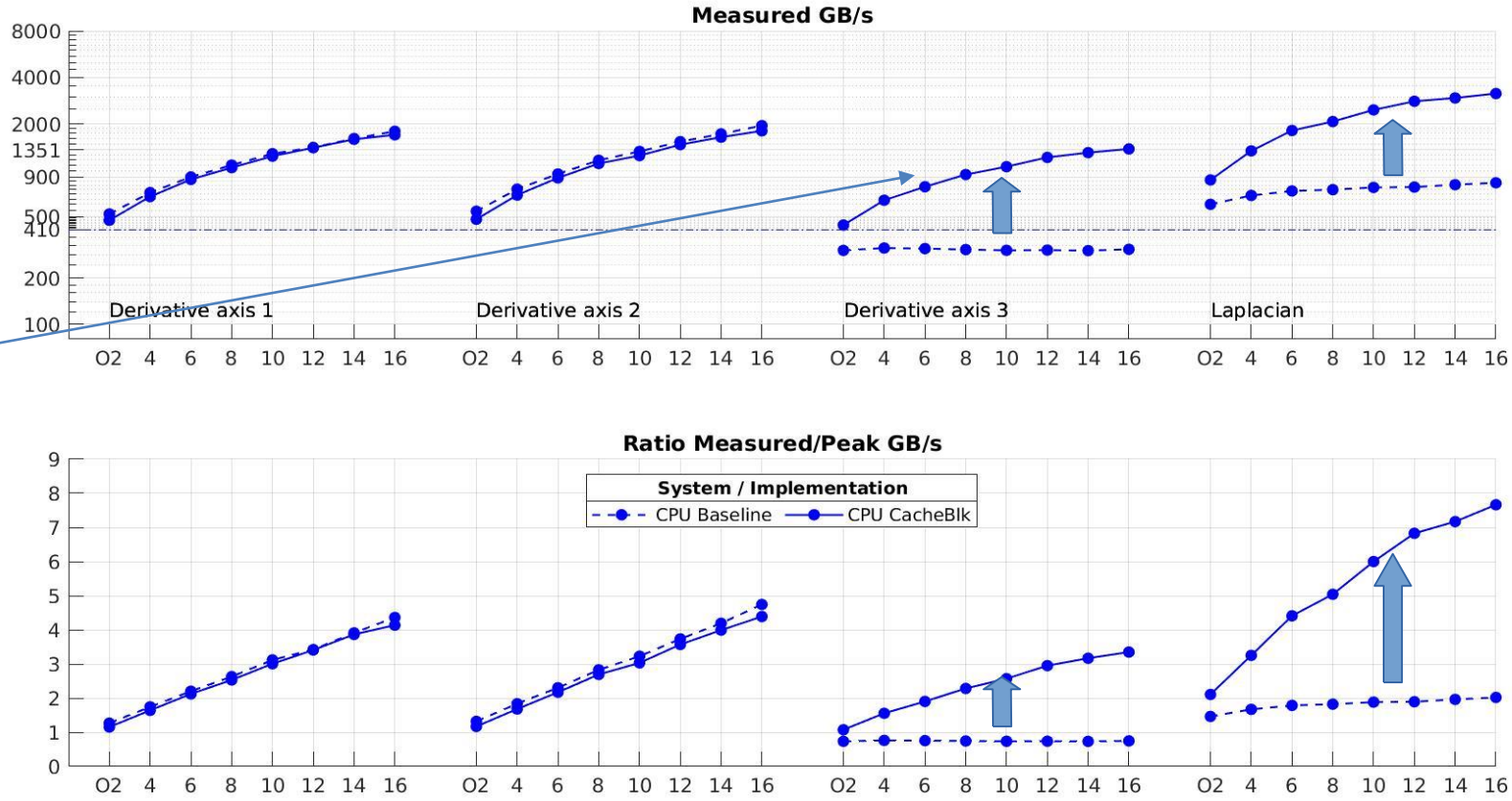
Performance

CPU Cache Blocking

Divide grid into small blocks
Force data to remain in cache

Axis 3 & Laplacian
►Considerable benefits

Implementation: easy
Tuning: finding optimal block size is not straight-forward



Efficiency increases with FD order for all orders
Important effect of cache blocking for the Laplacian

Performance

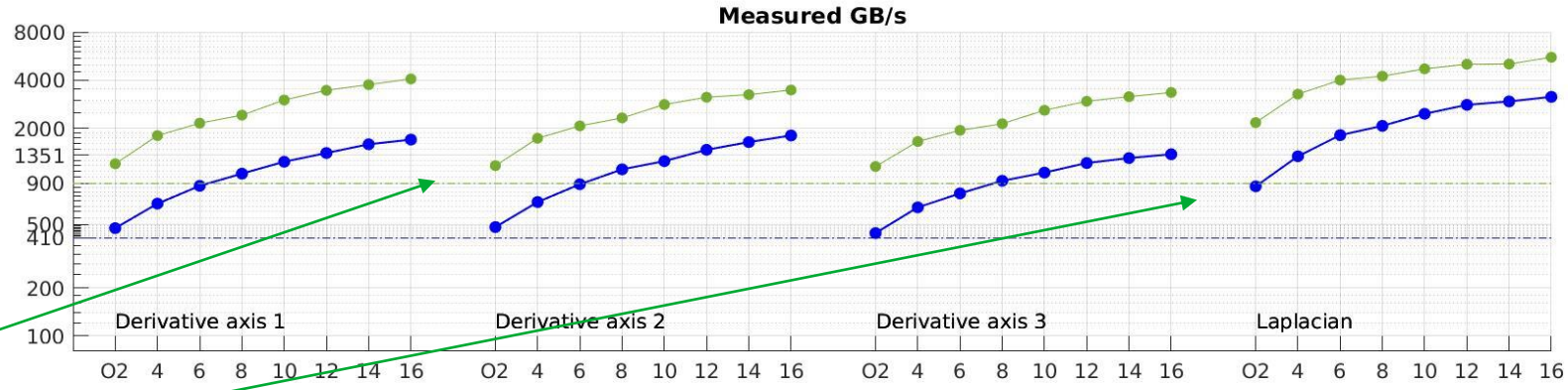
GPU Regular implementation
Constructor language
Grid divided into 3D thread blocks

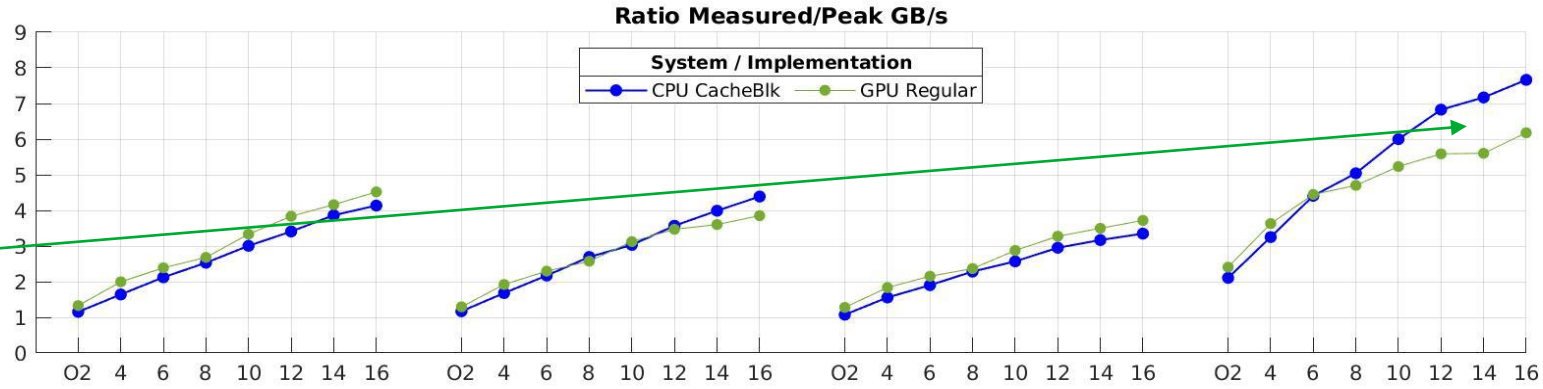
Axis 1 & 2: High data reuse

Axis 3 & Laplacian: Also good

Similar relative efficiency between CPU & GPU

Less efficient for large FD orders





GPU exhibits good performances for all operators
Can we get better?

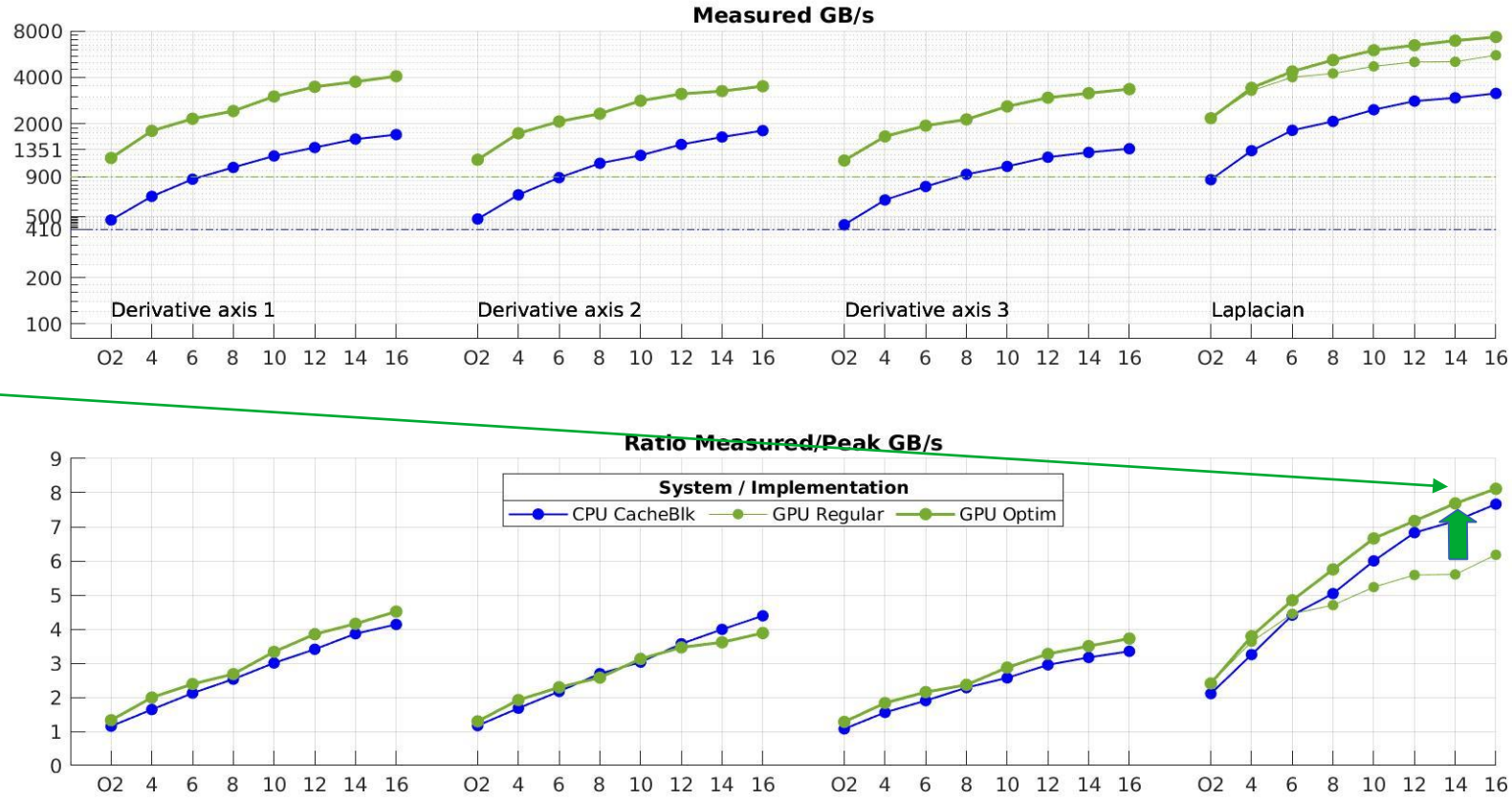
Performance

GPU Optim. implementation
Constructor language
2D ‘cache blocking’
data prefetching shared memory (slice N1 x N2)

Laplacian efficiency increased

Implementation: demanding
Tuning: finding optimal block size is not straight-forward

Note: implemented only for Laplacian



Similar relative efficiency between CPU & GPU
GPU naturally benefits from higher memory BW (x2.2)

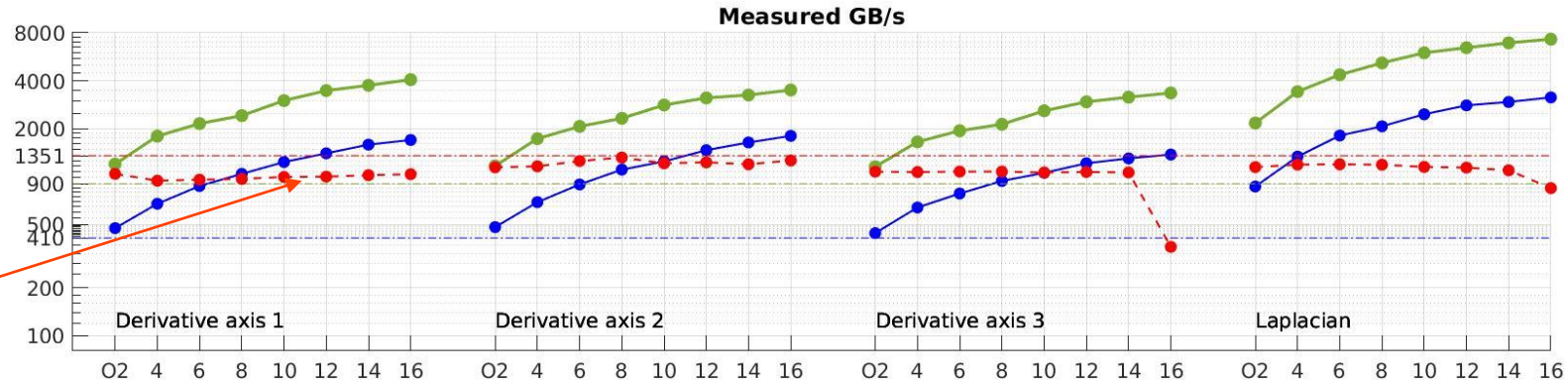
Test case FD_D2: FD Operators

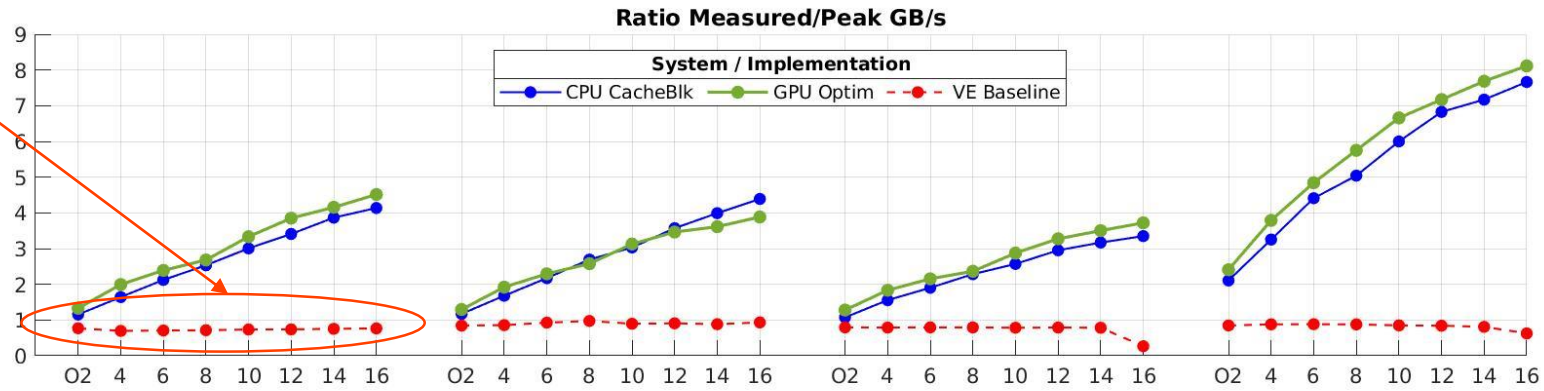
Performance

VE Baseline
hpcscan simply compiled as is with nc++

All operators: Low data reuse

Limited by Memory BW





With a simple compilation, FD operators run at the Memory BW of VE
There is no free lunch, let us do some effort!

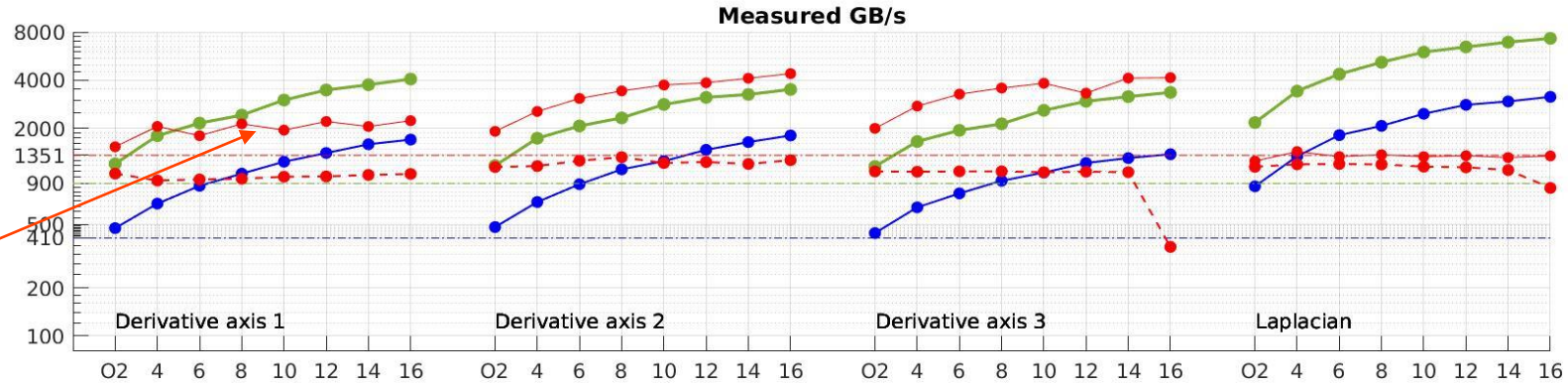
Performance

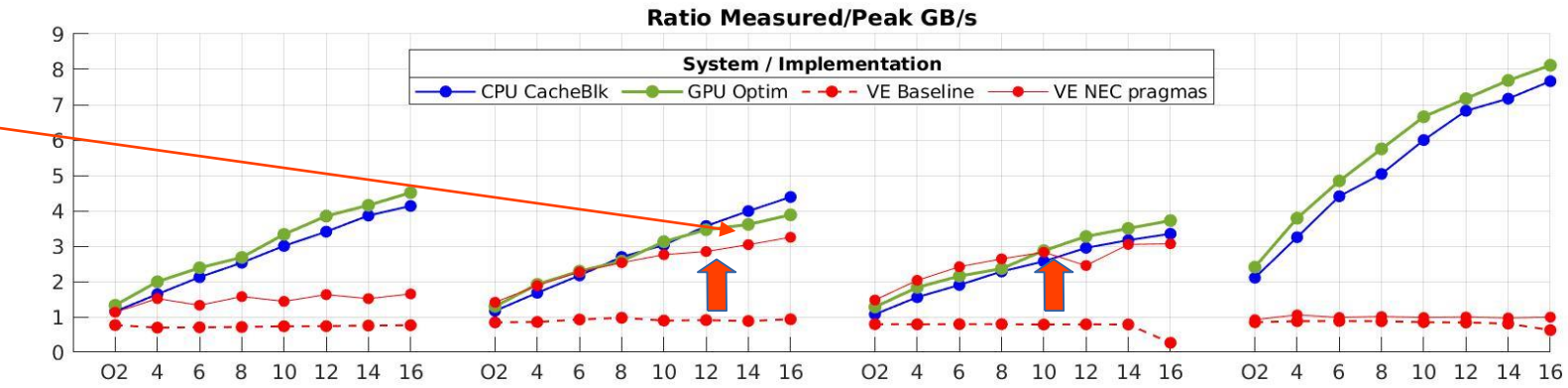
VE with compiler directives
NEC pragmas added to Baseline implementation

Axis 1: perf. Increase x2
`#pragma packed_stencil`
vector 512 SP flops (instead of 256)

Axis 2 & 3: high data reuse
`#pragma outerloop_unroll`
keep stencil data in registers

Slight improvements for the Laplacian





High performance for Axis 2 & 3
What about the Laplacian?

Performance

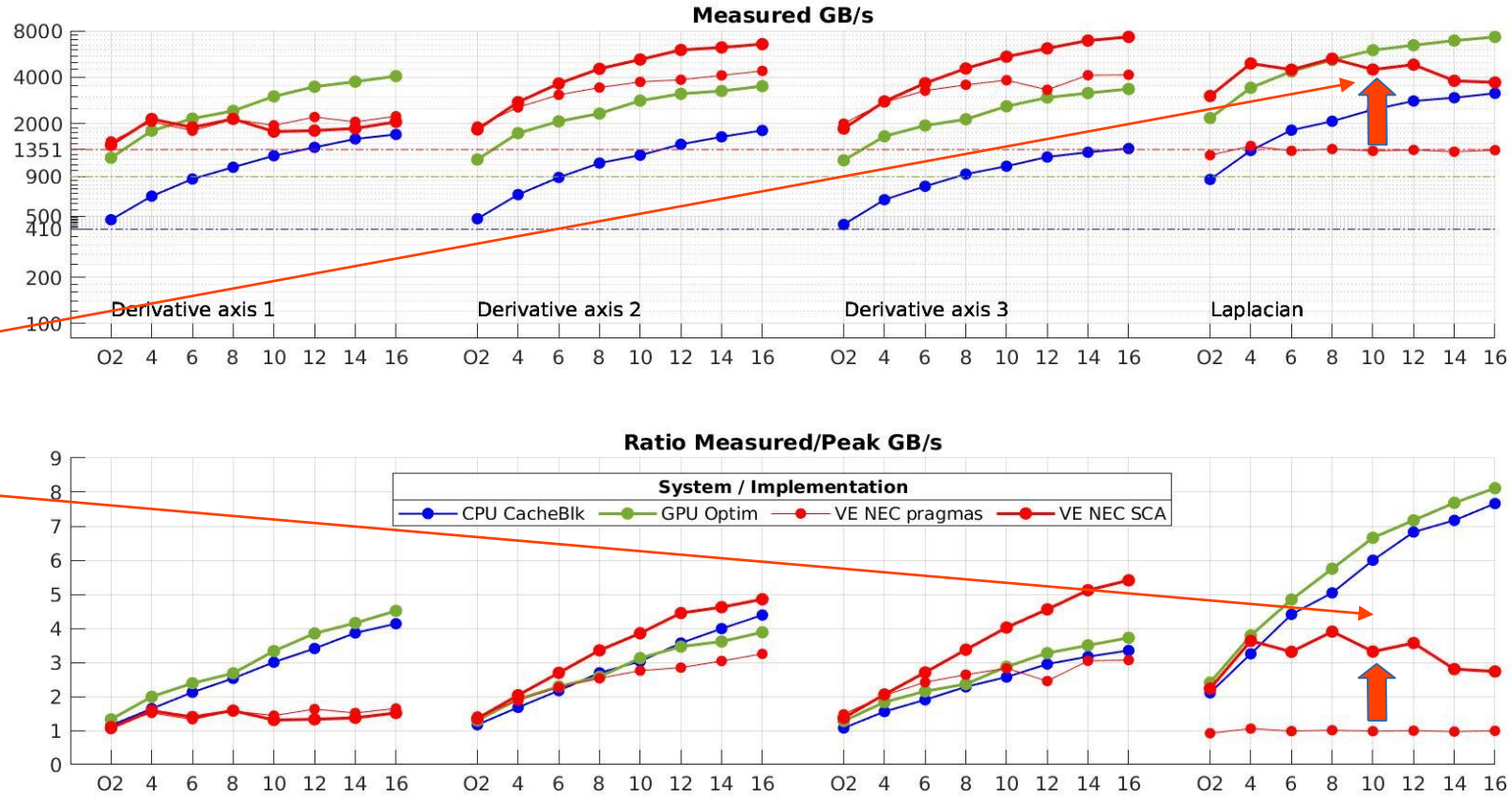
VE with NEC Math. Library Stencil Code Accelerator (SCA)
→ Talk of S. Momose

Large improvements for Axis 2 & 3, and Laplacian

Less efficient for high orders

Bottle-neck: Axis 1

Implementation: no effort, all done by the library



Limitation of SCA: Split computation for the acoustic propagator
2 steps: 1st Laplacian computation and 2nd pressure wavefield update

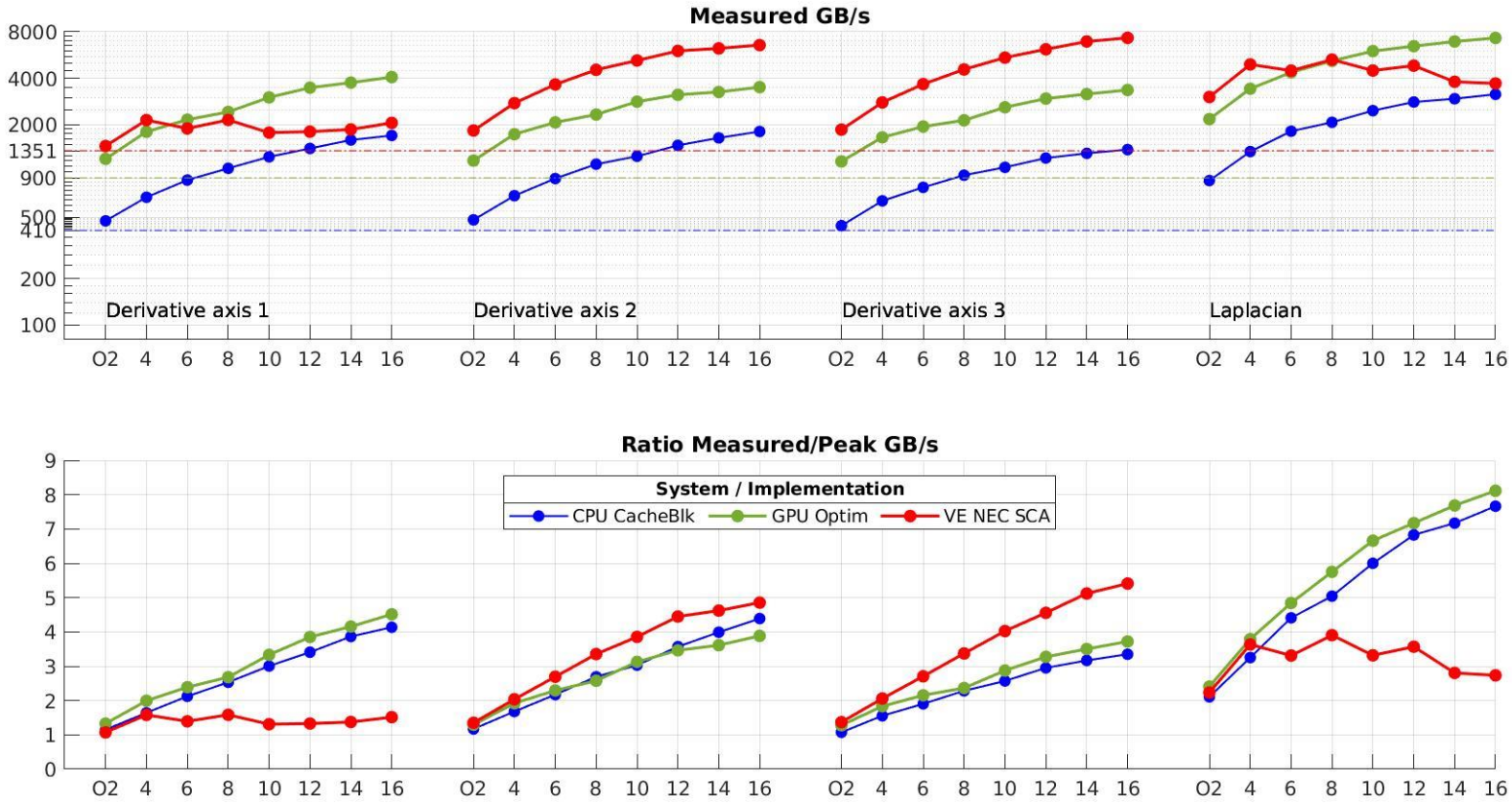
Performance

Summary of optimal implementation for each architecture

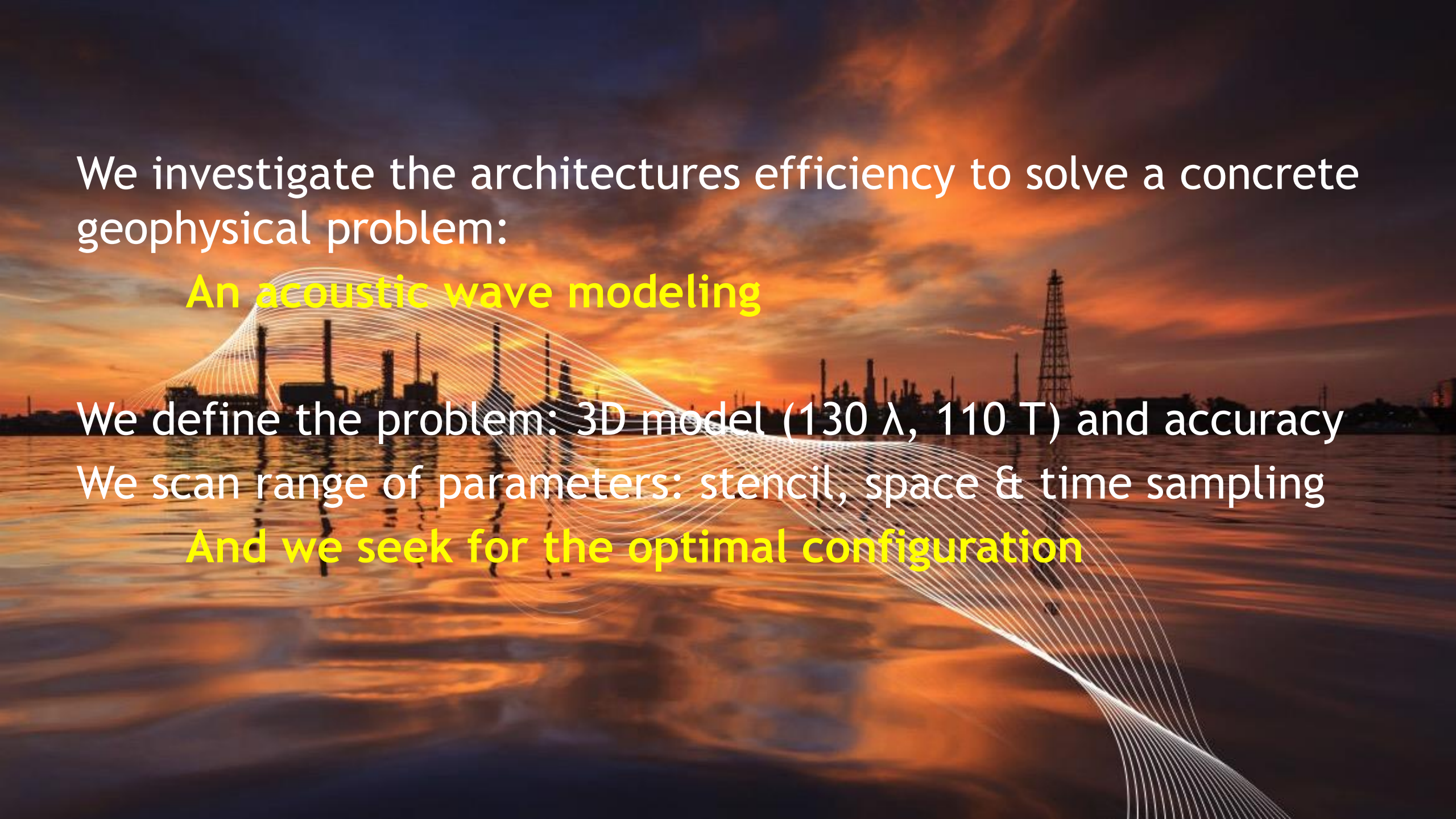
CPU: Cache blocking

GPU: Data prefetech in local (shared) memory

VE: NEC SCA library



Each architecture has its specific performance ‘signature’
How to select the optimum?



We investigate the architectures efficiency to solve a concrete geophysical problem:

An acoustic wave modeling

We define the problem: 3D model (130 λ, 110 T) and accuracy
We scan range of parameters: stencil, space & time sampling

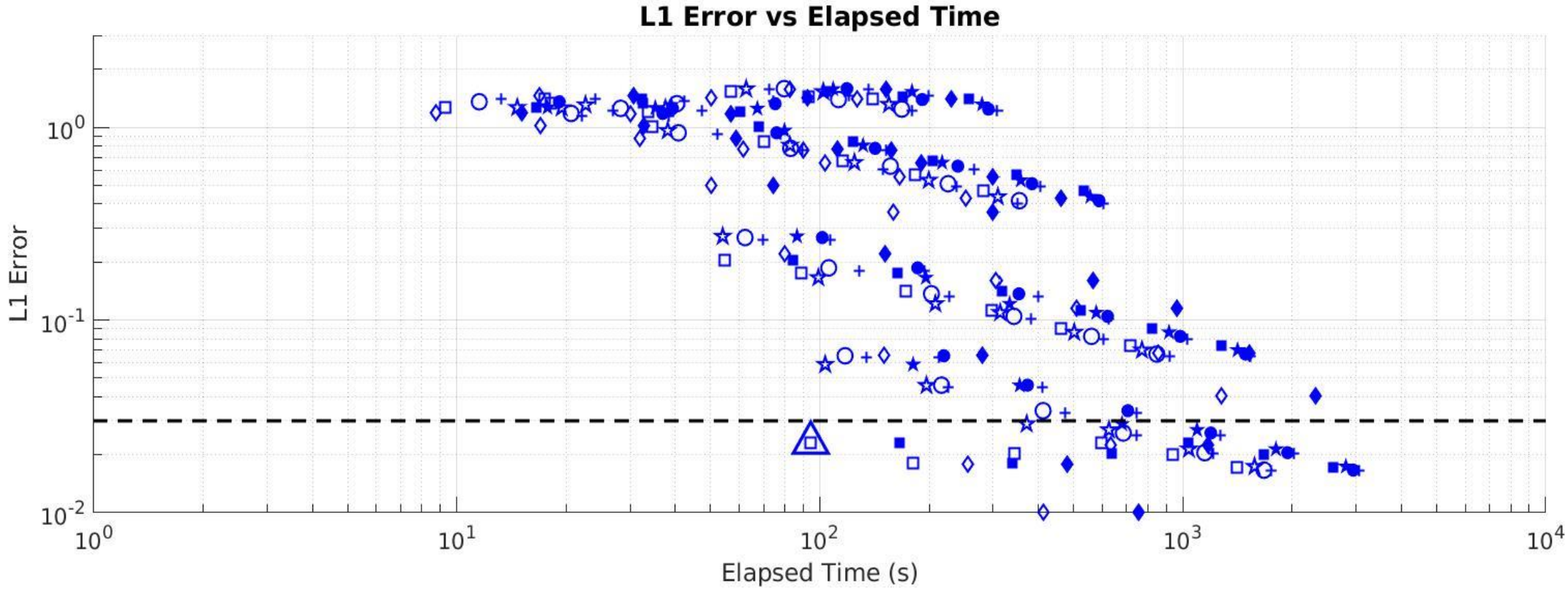
And we seek for the optimal configuration

Performance

Test case: Acoustic propagator

Parametric analysis – Run time

Max allowed error 3%
Case RTM production mode



	CPU
Optimal algo.	Stand. O6
Optimal time (s)	94.4
Speed-up	1

Grid sizes: from 500x500x500 to 1000x1000x1000 (inc 100)
FD order: from O4 to O12 (inc 2)
CFL ratios: 1, 0.5, 0.2 & 0.1
Propa. algo.: standard & split. computation

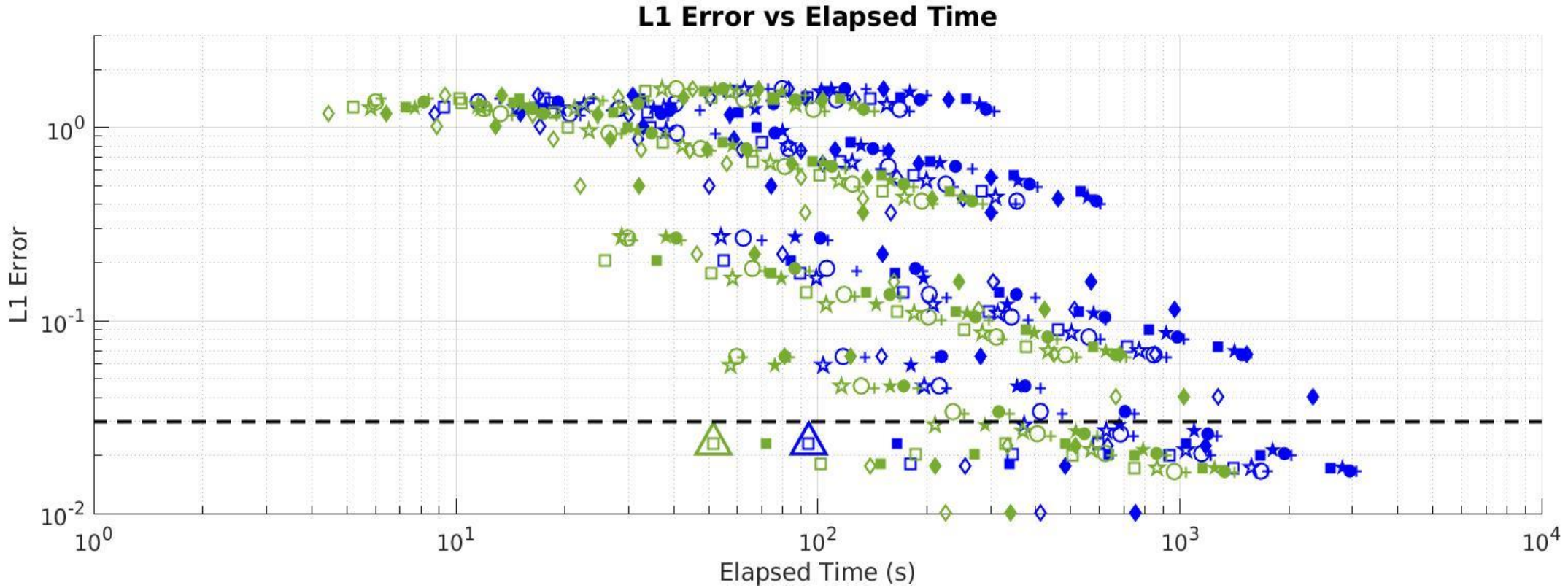
Total: 240 configurations

Performance

Test case: Acoustic propagator

Parametric analysis – Run time

Max allowed error 3%
Case RTM production mode



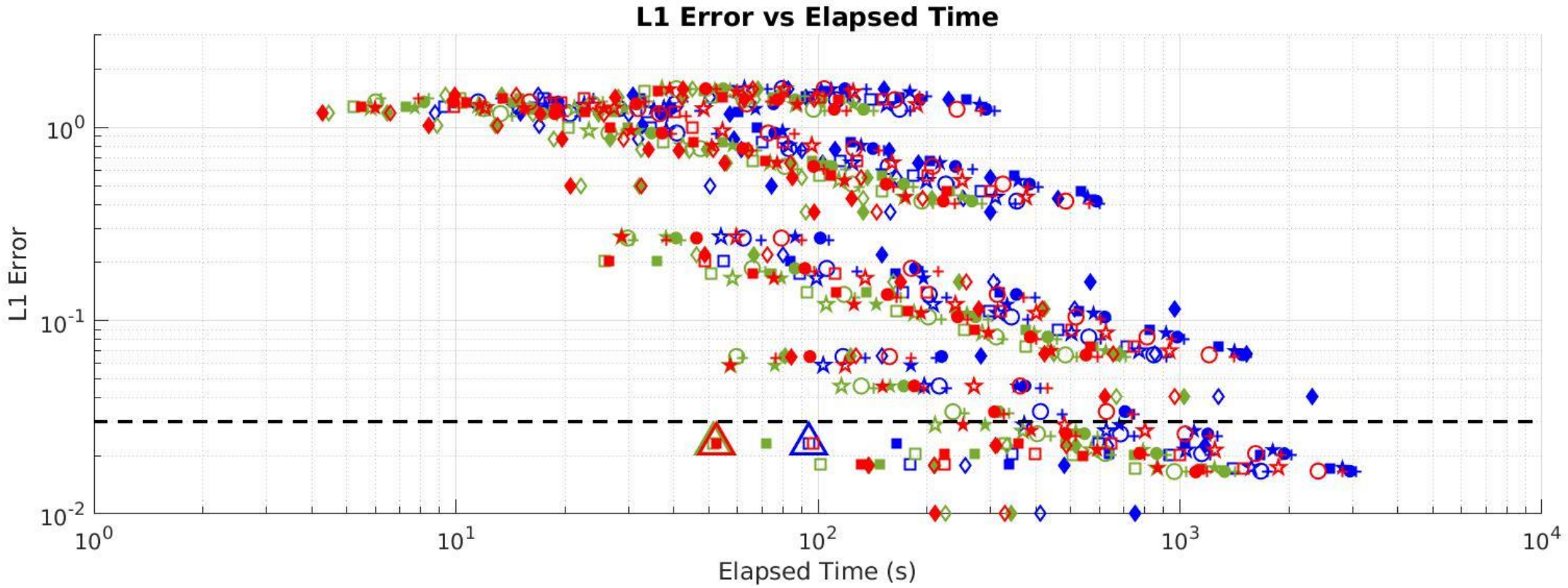
	CPU	GPU
Optimal algo.	Stand. O6	Stand. O6
Optimal time (s)	94.4	51.4
Speed-up	1	1.8

Performance

Test case: Acoustic propagator

Parametric analysis – Run time

Max allowed error 3%
Case RTM production mode



	CPU	GPU	VE
Optimal algo.	Stand. O6	Stand. O6	Split O6
Optimal time (s)	94.4	51.4	52.5
Speed-up	1	1.8	1.8

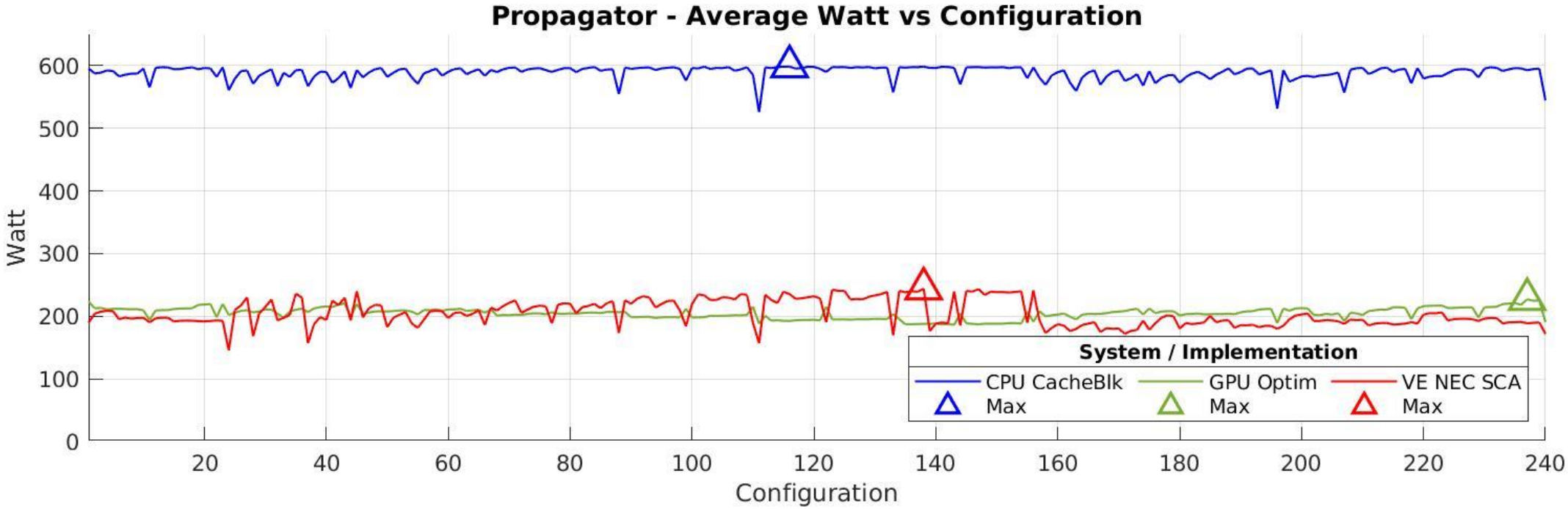
Optimal FD order 6
GPU & VE: 2x speedup vs CPU


A characterization including
an energy metrics is essential
Watt x runtime = W.h

Performance

Test case: Acoustic propagator

Parametric analysis – Energy



	CPU	GPU	VE
Peak Watt	600	300	300
Max measure	599	227	244
% peak	100	76	81

Algorithm	CPU	GPU	VE
Standard	585	208	190
Split Comp.	593	201	219

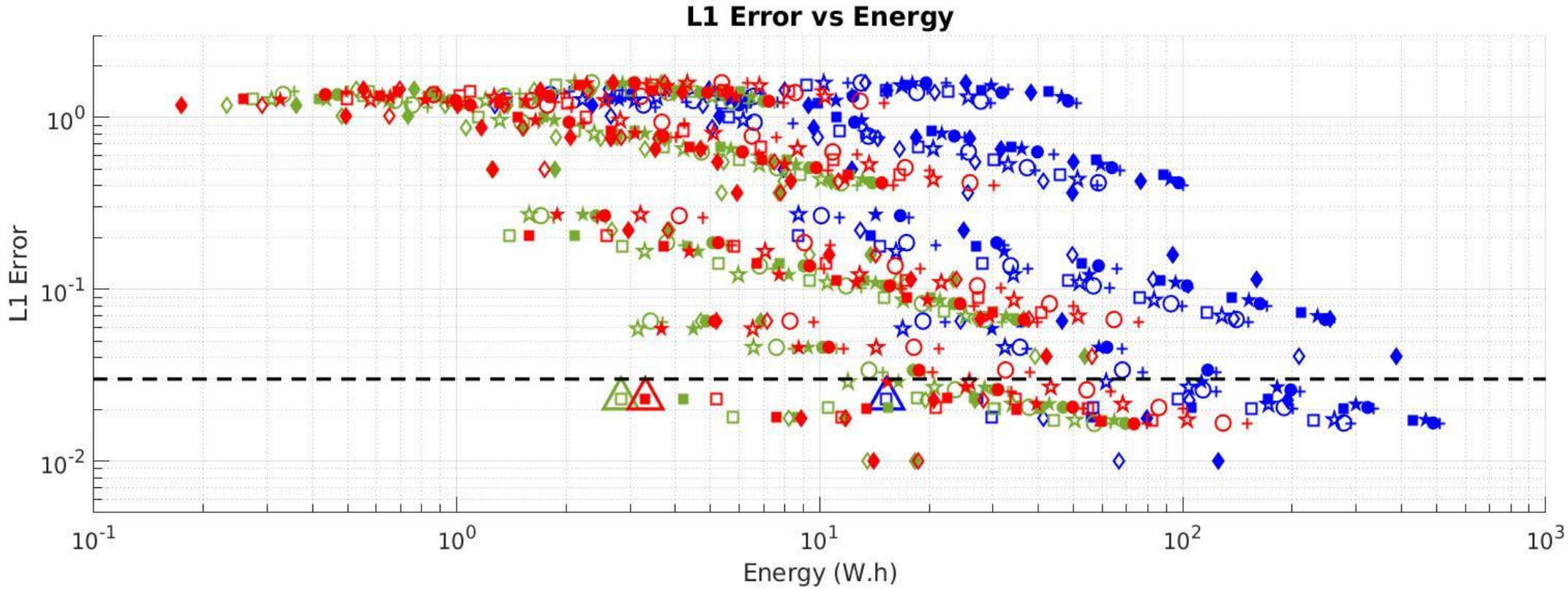
In average: GPU & VE: 2.5x less energy vs CPU

Performance

Test case: Acoustic propagator

Parametric analysis – Energy

Max allowed error 3%
i.e. RTM production mode



	CPU	GPU	VE
Optimal algo.	Stand. O6 N=500 0.1 CFL	Stand. O6 N=500 0.1 CFL	Split O6 N=500 0.1 CFL
Watt.Hour	15.24	2.83	3.31
Efficiency	1	5.4	4.6

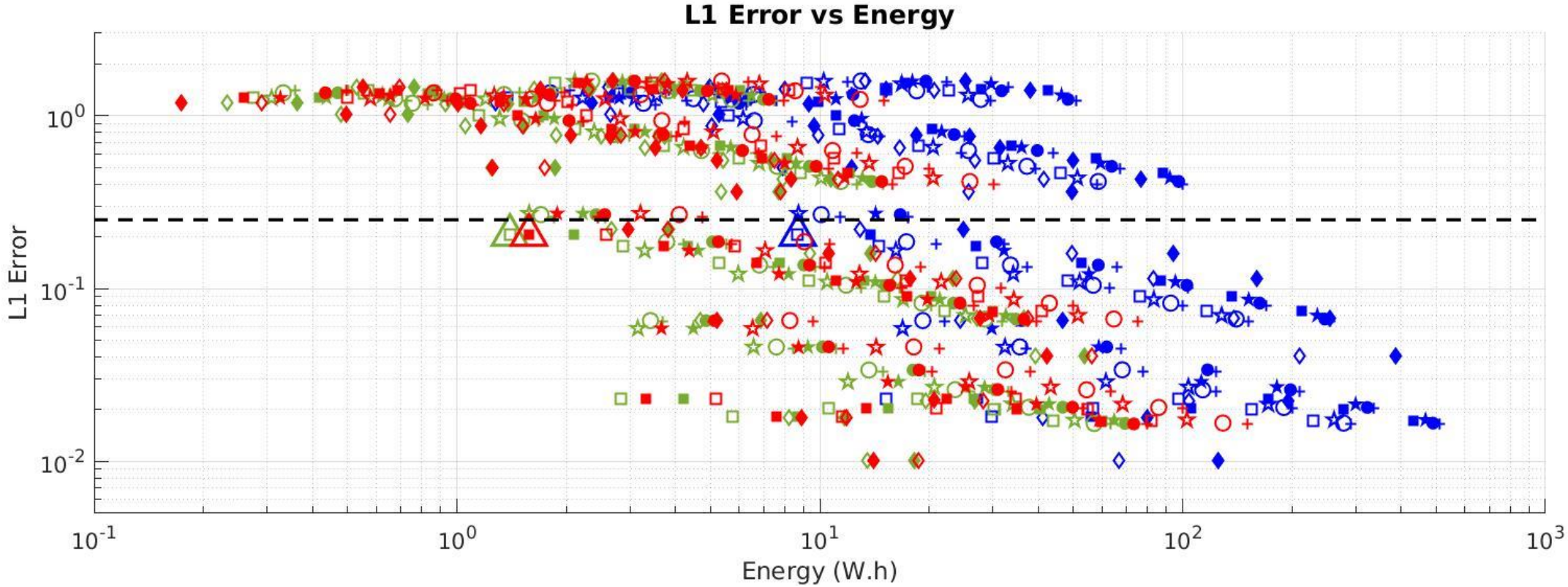
GPU & VE: Efficiency Combines Runtime + Energy

Performance

Test case: Acoustic propagator

Parametric analysis – Energy

Max allowed error 25%
i.e. RTM fast mode



	CPU	GPU	VE
Optimal algo.	Stand. O6 N=500 0.2 CFL	Stand. O6 N=500 0.2 CFL	Split O6 N=500 0.2 CFL
Watt.Hour	8.70	1.40	1.58
Efficiency	1	6.2	5.5

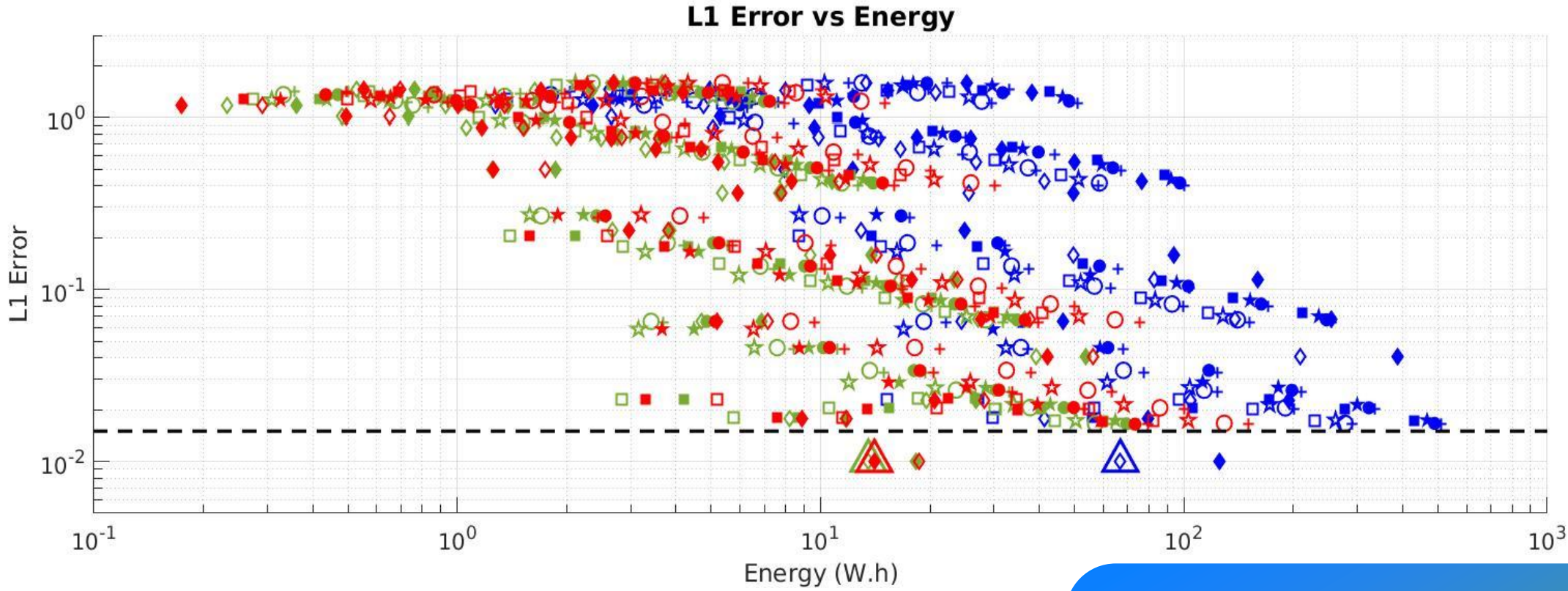
To relax accuracy
Changing Δt was most efficient

Performance

Test case: Acoustic propagator

Parametric analysis – Energy

Max allowed error 1.5%
i.e. modeling accurate mode



	CPU	GPU	VE
Optimal algo.	Stand. O4 N=900 0.2 CFL	Stand. O4 N=900 0.2 CFL	Split O4 N=900 0.2 CFL
Watt.Hour	66.8	13.5	14.1
Efficiency	1	5.0	4.7

To increase accuracy Smaller space sampling and lower order was most efficient However, ratio remains the same: GPU & VE 5x more efficient than CPU

Conclusions

4

Conclusions

We presented a **characterization of 3 HPC systems: CPU, GPU and VE** and showed

- the **importance of fine tuning for reliable comparison**
- the role of the **memory bandwidth and cache utilization** to achieve **high performance**

For an acoustic propagator (**compute intensive part of seismic imaging & inversion tools**)

Performance of VE is similar to GPU

- 2x runtime speed-up vs CPU
- **5x energy efficiency measured vs CPU**
- **Expected 4.4x energy efficiency at node level** (case of 8GPUs or 8VEs + 1 host (CPU) per node)

High performance easily obtained with NEC SCA lib. vs significant effort with GPU specific language

Future works

- Characterization of full application with all components (Interconnect + IO)
- More complex propagator (elastic vs acoustic)
- Higher order time integration scheme (high FD orders affected by 2nd order in time)
- Characterize more architectures (such as VE SX-Aurora 3, Q4 2022)

You are welcome to join this initiative!

Acknowledgements

Contributors to HPCscan

- Fabrice Dupros (Arm)
- Marcin Rogowski (KAUST)
- Philippe Thierry (Intel)
- Suha Kayum (Saudi Aramco)

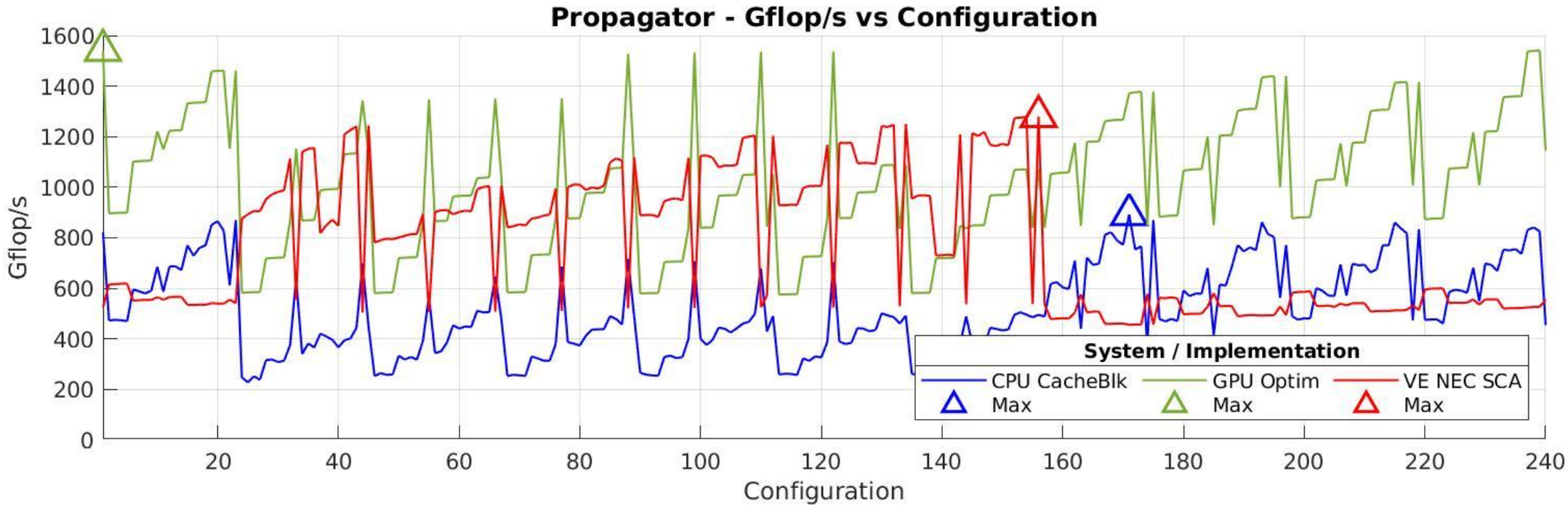
Discussions, Help and Support

- Adil Lashab, Dominic Charrier, Mazda Sabony (AMD)
- Asma Farjallah, Rached Abdelkhalek (NVIDIA)
- Bruno Lecointe (NEC)
- Yoshio Nakao (Fujitsu)

Backup Slides

Performance

Test case: Acoustic propagator
Parametric analysis – Run time

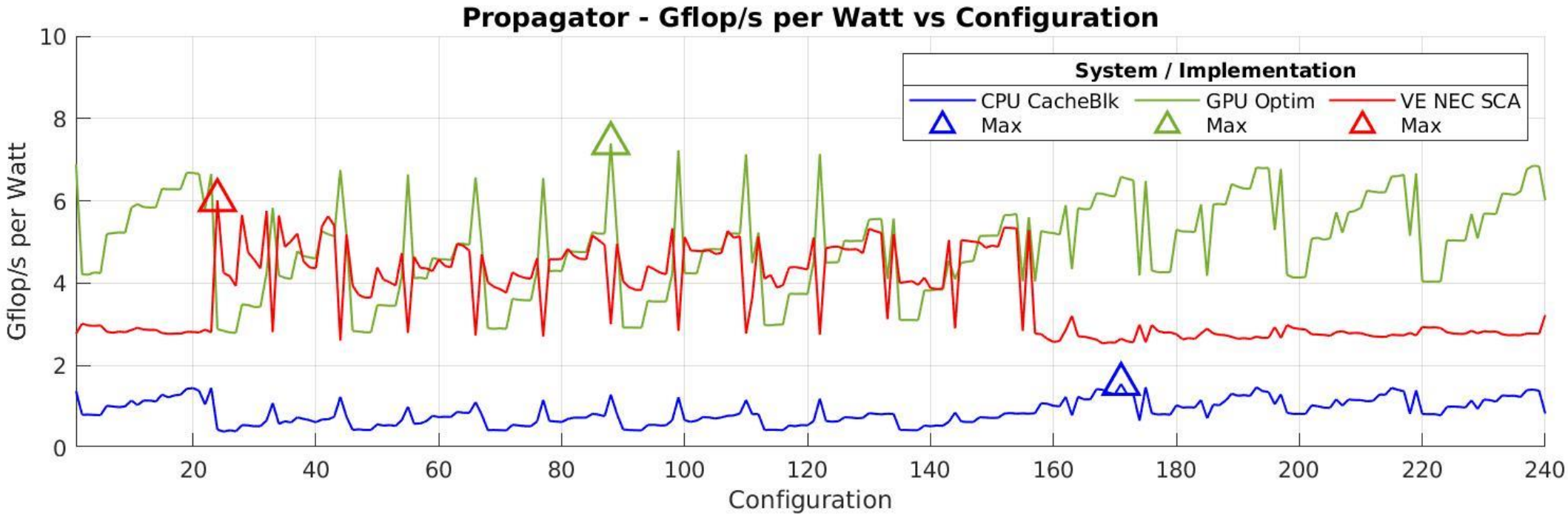


	CPU	GPU	VE
Peak Gflop/s	11 670	15 000	4 300
Max measure	892	1 542	1 280
% peak	7.6	10.3	29.8

Algorithm	CPU	GPU	VE
Standard	652	1179	531
Split Comp.	373	843	1 012

Performance

Test case: Acoustic propagator
Parametric analysis – Energy



	CPU	GPU	VE
Peak Gflop/s / Watt	19.5	50	14.3
Max measure	1.5	7.4	6.0
% peak	7.7	14.8	42.0

Algorithm	CPU	GPU	VE
Standard	1.1	5.6	2.8
Split Comp.	0.6	4.2	4.6