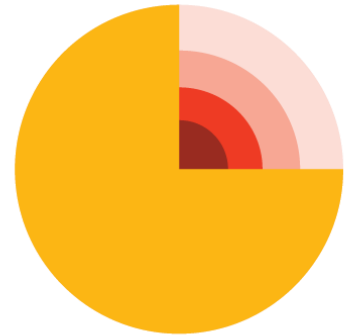


**Task 2.2:
Increasing the performance portability of
flagship codes for different accelerators**

g.amati@cineca.it

28/03/2023



ChEESE-2P



Tasks and task leaders

Task	Task name	Start	End	Lead
Task 2.1	Code audits (Jose Gracia, HLRS)	3 (27)	15 (36)	HLRS
Task 2.2	Increasing the performance portability of flagship codes for different accelerators (Giorgio Amati, Luca Ferraro, CINECA)	1	48	CIN
Task 2.3	Optimisation for exascale (Piero Lanucara, CINECA)	6	48	CIN
Task 2.4	Continuous Performance Efficiency Monitoring (Marta Garcia, BSC)	18	48	BSC

Effort (PMs)													
CIN	CSIC	BSC	INGV	HLRS	TUM	LMU	UMA	CNRS	IPGP	BULL	CSC	UM	SU
67	12	18	18	18	22	14	12	28	1	14	8	8	7



HW@EuroHPC (reprise)

EuroHPC-JU Supercomputer available in 2023

Name	Top500 Rank	GPU	CPU
Lumi	3	AMD MI250@128GB	AMD EPYC@64 core
Leonardo	4	Nvidia A100@64GB	Intel Sapphire Rapids
MareNostrum5	-	Nvidia H100/ Intel Rialto	Intel EMR/Nvidia Grace
Meluxina	52	Nvidia A100@40GB	-
Karolina	85	Nvidia A100@40GB	-
Discover	123	-	AMD 7H12@64 core
Vega	152	Nvidia A100@40GB	AMD 7H12@64 core
Decaulion	-	<i>Nvidia A100</i>	<i>AMD/Fujitsu A64FX</i>

Preparation strategy (Pillar 2). Task T2.2 (reprise)

Task T2.2: Increasing the performance portability of flagship codes for different accelerators.

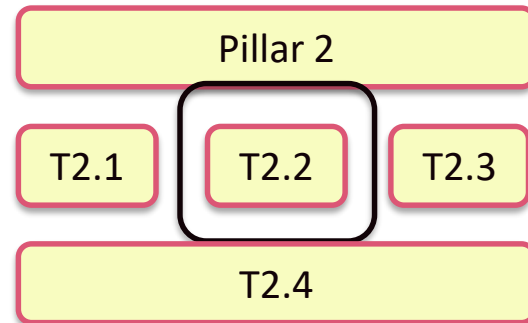
- ❑ **Lead: Giorgio Amati, Luca Ferraro, CINECA**
- ❑ **Partners: CSIC, INGV, TUM, LMU, UMA, CNRS, IPGP, CSC, UM, SU**

Task T2.2 will contribute to increase "performance portability" of the flagship codes in the "accelerator arena" (focusing on EuroHPC heterogeneous HW) using:

- ❑ **Methodology based on available literature (including metric for the target)**
- ❑ **Performance portable SW tools and libraries and parallel paradigm (including language standard based, OpenMP, OpenACC, Kokkos, etc etc)**
- ❑ **Development team acting on flagship codes/mini apps**

Task T2.2 will:

- **depend on the output of Task T2.1**
- **contribute to Deliverables D2.2 and D2.4**



Adequacy of flagship codes to
EuroHPC heterogeneous
hardware



Flagship code (reprise)

No.	Code	Initial Level	New code	Area	Lead
1	SeisSol	1	no	Computational seismology Tsunami modelling	LMU
2	SPECFEM3D	1	no	Computational seismology	CNRS
3	ExaHyPE	2	no	Computational seismology Tsunami modelling	TUM
4	Tandem	2	yes	Computational seismology	LMU
5	xSHELLS	1	no	Magnetohydrodynamics	CNRS
6	HySEA	2	no	Tsunami modelling	UMA
7	FALL3D	2	no	Volcanology	CSIC
8	OpenPDAC	2	yes	Volcanology	INGV
9	LaMEM	1	yes	Geodynamics	UM
10	pTatin3D	1	yes	Geodynamics	SU
11	ELMER/ICE	2	yes	Glacier hazards	CSC

Flagship code status (at March 2023)

Code	MPI	CUDA	HIP	SYCL/OpenCL	OpenACC	OMP offload	Library GPU
SeisSol							
SPECFEM3D							
ExaHyPE							
Tandem							
xSHELLS							
HySEA							
FALL3D							
OpenPDAC							
LaMEM							
pTatin3D							
ELMER/ICE							

Flagship code status (at M48)

Code	MPI	CUDA	HIP	SYCL/OpenCL	OpenACC	OMP offload	Library GPU
SeisSol					???		???
SPECFEM3D				???	???	???	???
ExaHyPE		???			???		???
Tandem		???	???	???	???	???	
xSHELLS			???	???	???	???	???
HySEA			???	???	???	???	???
FALL3D		???	???	???		???	???
OpenPDAC		???	???	???	???	???	???
LaMEM		???	???	???	???	???	
pTatin3D			???		???	???	
ELMER/ICE		???	???	???	???	???	

Flagship code HW request for Cheese2P

Code	Lumi	Vega	Karolina	Meluxina	Leonardo	MN5	MN5-NextG	Decaulion
SeisSol	Green	Red	Red	Red	Green	Yellow	Yellow	Red
SPECFEM3D	Green	Red	Red	Green	Green	Green	Yellow	Red
ExaHyPE	Green	Green	Red	Red	Green	Green	Green	Red
Tandem	Green	Green	Green	Green	Green	Yellow	Red	Red
xSHELLS	Green	Green	Green	Red	Red	Green	Yellow	Green
HySEA	Red	Green	Green	Green	Green	Green	Red	Red
FALL3D	Red	Red	Red	Red	Green	Green	Red	Red
OpenPDAC	Red	Red	Red	Red	Green	Red	Red	Red
LaMEM	Red	Red	Red	Green	Red	Red	Red	Red
pTatin3D	Green	Red	Green	Red	Green	Green	Yellow	Red
ELMER/ICE	Green	Red	Red	Red	Red	Red	Red	Green

Porting to GPU: possible solutions

- ❑ **Language intrinsic parallelism**
 - ❑ C/C++ standard parallelism
 - ❑ Fortran (do concurrent)
- ❑ **Directive-based parallelism**
 - ❑ OpenACC
 - ❑ OpenMP offload
- ❑ **Performance portability framework**
 - ❑ Kokkos
 - ❑ Raja, ...
- ❑ **SW Layer for heterogenous parallelism**
 - ❑ SYCL/OpenCL
 - ❑ CUDA
 - ❑ HIP, ...
- ❑ **GPU-based libraries**
 - ❑ AMGx
 - ❑ cuFFT
 - ❑ PETSc, ...



Performance portability approach

Approach #1

- ☐ Max performance using different accelerators
- ☐ Different extension for different device
 - ☐ **NVIDIA:** CUDA
 - ☐ **AMD:** HIP
 - ☐ **Intel:** SYCL/OneAPI
- ☐ Possible problem for external libraries (e.g., cuFFT)

Approach #2

- ☐ Acceptable performance using different accelerators
- ☐ Using a single extension for different device (e.g., OMP offload, sycl)
 - ☐ Suboptimal performance
- ☐ Different implementation to accomplish different HW

A more rigorous definition:

[Navigating Performance, Portability and Productivity](#)



Porting to GPU:

State of the art so far.....

 Full vendor support		 Vendor support, but not (yet) entirely comprehensive		 Indirect, but comprehensive support, by vendor		 Comprehensive support, but not by vendor		 Limited, probably indirect support – but at least some		 No direct support available, but of course one could ISO-C-		bind your way through it or directly link the libraries		C++ C++ (sometimes also C)		Fortran Fortran	
	CUDA		HIP		SYCL		OpenACC		OpenMP		Standard		Kokkos		ALPAKA		
	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	Python
NVIDIA	 1	 2	 3	 4	 5	 6	 7	 8	 9	 10	 11	 12	 13	 14	 15	 16	 17
AMD	 18	 19	 20	 4	 21	 6	 22	  23	 24	 24	 25	 25	 26	 14	 27	 16	 28
Intel	 29	 30	 31	 4	 32	 6	 33	 33	 34	 34	 35	 36	 37	 14	 38	 16	 39

From: <https://github.com/AndiH/gpu-lang-compat>

... sorry, no silver bullet!!!!!!

Task T2.2 planning: next steps

- ❑ A more correct "matrix" for each flagship code (with final target, available mini-app, efforts and so on)
- ❑ Organize a calendar of webinar (starting from April) on accelerators, porting techniques, measuring performances and so on, e.g.
 - ❑ **OpenMP offload**
 - ❑ **Kokkos (brief introduction)**
 - ❑ **OpenCL (brief introduction)**
 - ❑ **AMD webinar on GPU**
 - ❑ **Nvidia GTC material (slides)**

See the final presentation for the integration of T2.2 planning in the WP2 overall picture



Some Material from NVIDIA GTC 03/2023

[Cheese2P Generale Repository \(WP2/GTC 03 2023\)](#)

- S51043 - No More Porting GPU Computing with Standard C++ and Fortran
- S51054 - Accelerating HPC applications with ISO C++ on Grace Hopper.pdf
- S51119 - Optimizing Applications for Hopper Architecture
- S51120 - Programming Model and Applications for the Grace Hopper Superchip
- S51176 - Recent Developments in NVIDIA Math Libraries_1679417423686001guJJ
- S51205 - From the Macro to the Micro CUDA Developer Tools Find and Fix Problems at Any Scale
- S51210 - How to Write a CUDA Program
- S51225 - CUDA New Features and Beyond
- S51334 - Measure Right! Best Practices when Benchmarking CUDA Application
- S51755 - Asynchronous Acceleration in Standard C++
- S51772 - Debugging CUDA An Overview of CUDA Correctness Tools
- S51857 - Best Practices for Programming GPUs using Fortran, OpenACC, and CUDA
- S51882 - Become Faster in Writing Performant CUDA Kernels using the Source Page in Nsight Compute



Auxiliary slides

