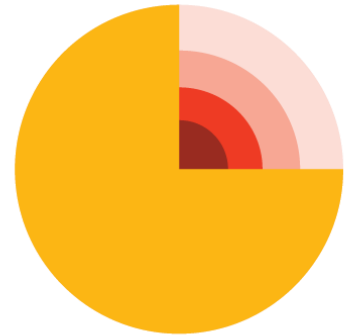


Task 2.2:

**Increasing the performance portability of
flagship codes for different accelerators**

27/04/2023



ChEESE-2P



Perfpr

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

Porting to GPU: State of the art

● 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		Python
	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	C++	Fortran	
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

Sorry, no silver bullet!!!!!!

Performance portability: recap

- A. **Intrinsic language constructs** (i.e., do concurrent Fortran or parallel stl C++)
- B. **Vendor specific paradigms** (i.e., CUDA/HIP/OneAPI)
- C. **OpenCL standard**
- D. **Directive-based standards** (i.e., OpenMP/OpenACX)
- E. **high-level portability framework** (i.e., Kokkos/Legion/OpenSYCL/alpaka/RAJA)
- F. **External GPU-based subroutine libraries** (i.e., PeTSc)

Main issues (to discuss)

A --> Not "complete" or even "partial" support by compilers

B --> it should give best performance. Translation between paradigm (e.g., CUDA --> HIP) could be not so simple (e.g., for Fortran)

C --> Code verbose, performance could be hard to achieve, only C.

D --> Not "complete" support and performance could be difficult to achieve

E --> Implies (serious) code rewriting and only some languages are supported (e.g., C/C++)

F --> Only few Algorithm were ported to GPU



How to achieve portability

Current Accelerator Framework	Suggested Portability Framework	Expected Effort
None	OpenMP	Medium (incremental)
CUDA	HIP	Low / Medium
OpenACC	OpenMP	Low
OpenMP/Kokkos/Occam	-	None
Intrinsic Parallelism	???	???
Library Accelerated Based	???	???

Depending on the starting Accelerator Framework (if any) a lower effort strategy is to choose the closer/best framework working for different vendor and supported by the language

- ❑ When dealing with intrinsic parallelism (such as C++ `std::par`) or with library accelerated code, the portability solution should be planned carefully



Planning Recipe

- ☐ Identify portability goal for the application
- ☐ Identify possible portability framework
 - ☐ developers should choose a solution to get familiar with
- ☐ Identify dwarfs or mini-app to work with
 - ☐ they will isolate specific algorithms from complexity of the full application
 - ☐ they should be easy to analyze/control/test
 - ☐ they should be easily reinjected into the full application
- ☐ Flag ship code developers will team up with our CINECA specialists to plan and work to the porting activities
- ☐ SCRUM like activity backlogs will be planned and revised informally every 2/3 week of work.



T2.2: **possible** workplan for Fall3D

WP2@t=0

- Release 1.0: MPI+OpenACC (Nvidia GPU)

For T2.1

- Code Audit: find info for tuning version (rel. 1.1)

For T2.2

- Check performance using AMD (OpenACC support using Cray compiler)
- Porting using openMP offload (Rel. 2.0)
 - Check performance using different devices (Nvidia, AMD, Intel)
- If necessary, move to another parallel paradigm ("Plan B" e.g., Hipfort, Rel. 3.0)



T2.2: **possible** Timeline for Fall3D

Activity	Month	HW
Check OpenACC performance on AMD	10	Lumi
Developing mini-App with OpenMP offload	12	Leonardo
Benchmarking mini-app	16	Leonardo, Lumi
Porting full code on OpenMP offload	26	Nvidia
Benchmarking full code	30	Leonardo
Plan B (if necessary)	36	?



Training

Planned Webinar in May

- Experiences using intrinsic language constructs for a Fortran CFD-based code
- Experiences using a OpenMP Offload for a Fortran CFD-based code

Possible Webinar

- OpenCL
- High-level portability framework
- Do you look for something in particular? Please let us know!



Auxiliary Slide

