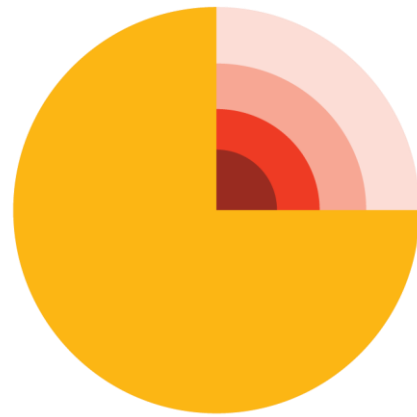




ChEESE

Deliverable

D3.1 Selected set of mini-apps and its requirements



Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Spain, Italy, Iceland, Germany, Norway, France, Finland and Croatia under grant agreement No 101093038.



ChEESE-2P Deliverable

D3.1 Selected set of mini-apps and Its requirements

Document Information

Contract Number	101093038
Project Website	https://cheese2.eu
Contractual Deadline	31/12/2023 (M12)
Activity	WP3
Dissemination Level	PUBLIC
Nature	R

Abstract

This document provides information related to ChEESE-2P WP3, including outlines of other projects involved in the co-design phase and the selected set of mini-applications that will be used, together with their software and platform requirements.

I. Copyright

Copyright Members of ChEESE collaboration, 2023/2026.

II. Delivery Slip

Role	Name	Partner/Activity
Author	Masini A.	CINECA
Contributor 1	Raffin E., Hamitou O.	Eviden
Contributor 2	Garcia M., Mantovani F.	BSC
Reviewer 1	Amati G.	CINECA
Reviewer 2	Pascual H.	CSIC

III. Version control

Version	Date	Description of change
V1	20/09/2023	Initial draft for internal working
V2	05/12/2023	Added contribution from EVIDEN
V3	11/12/2023	(GA): first comments Verified repository link
V4	19/12/2023	Added contribution from BSC
V5	20/12/2023	(HB): reviewed
V6	28/12/2023	(GA): reviewed

IV. List of Acronyms

Acronym	Description
ChEESE	Center of Excellence in Solid Earth
CPU	Central Processor Unit
EPI	European Processor Initiative
EUPEX	European Pilot for Exascale
EUPilot	European Pilot using Independent, Local and Open Technologies
EuroHPC JU	European High Performance Computing Joint Undertaking
GPP	General Purpose Processor
LPDDR	Low Power Double Data Rate
HPC	High Performance Computing
MSA	Modular Supercomputing Architecture
MPI	Message Passing Interface
OpenMP	Open Multiprocessor
SSE	Streaming SIMD Extension
SVE	Scalable Vector Extension
VPU	Vector Processor UNIT

Table of Contents

Table of Contents.....	4
1 - Introduction.....	5
2 – Technical background	6
2.1 – European Processor Initiative	6
2.2 – EUPEX	8
Current EUPEX hardware/software infrastructure	8
2.3 – EUPILOT.....	10
Current EUPILOT hardware/software infrastructure	10
2.4 – Connections with ChEESE-2P	11
Mini-apps for co-design (ChEESE-2P WP3).....	11
Mini-apps for performance portability (ChEESE-2P WP2).....	13
Mapping of mini-apps to co-design targets	13
3 – State of the art of mini-apps at M12.....	15
3.1 - SeisSol.....	15
3.2 - SPECFEM3D	16
3.3 - Tandem.....	16
3.5 - xSHELLS.....	18
3.5 - HySEA	18
3.6 - FALL3D	19
3.7 - pTatin3D	19
3.8 – Summary Table	21
Appendix A – Support Activities and Materials	22
A.1 – Webinar on Mini-app Extraction.....	22
A.2 – Guidelines for Mini-app Preparation.....	22
Appendix B – Mini-app software dependencies glossary	23
References	25

1 - Introduction

One of the main objectives of WP3 is to coordinate the effort to extract a set of mini-apps from ChEESE-2P flagship codes, making them available as test cases to be exploited for the co-design of new generation hardware platforms developed in the context of other European projects, such as the European Processor Initiative (EPI), the European Pilot Project (EUPILOT), and the European Pilot for Exascale (EUPEX).

At the start of T3.1 a subset of mini-apps already existed, mainly due to some ChEESE-2P flagship codes having participated in the first iteration of the ChEESE project. Together with these existing mini-apps, a set of new mini-apps is actively being prepared, with some of them having already reached readiness.

The objective of this document is to provide an account of hardware and software requirements of the above-mentioned mini-apps, in order to make them more readily available for the co-design process that will take place in the next tasks of ChEESE-2P WP3.

The document is articulated in two main sections:

- **Section 2** provides technical background on European projects involved in the co-design process, with some details about hardware and software stack of each pilot system, and their connection with ChEESE-2P.
- **Section 3** reports the state of the art of ChEESE-2P mini-apps at M12, together with information about how to access them, the software dependencies required to compile and run them, some examples models to be used as input, and the tools used to validate the obtained results.

Appendix A at the end of the document contains information on support material and activities devised to help ChEESE-2P partners in the preparation of their mini-app, while **Appendix B** contains a glossary with short descriptions of each software package that appears as a dependency for one or more mini-apps in section 3.

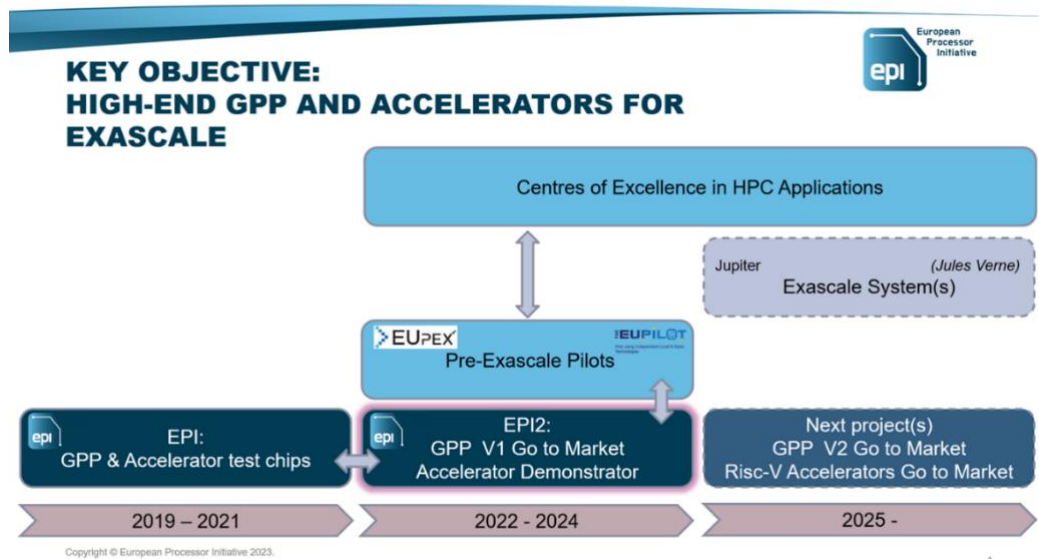
2 – Technical background

The following section contains info on the European projects involved in the co-design process. We provide a short description of each project, and where possible we also list some details about its hardware and software platform.

2.1 – European Processor Initiative

The European Processor Initiative (EPI) is project funded under the framework of the European High Performance Computing Joint Undertaking (EuroHPC JU), that specifically aims to conduct research to advance High Performance Computing (HPC) through the development of dedicated European hardware technologies.

Figure 1: Overview of EPI and its links with Centers of Excellence such as ChEESE-2P

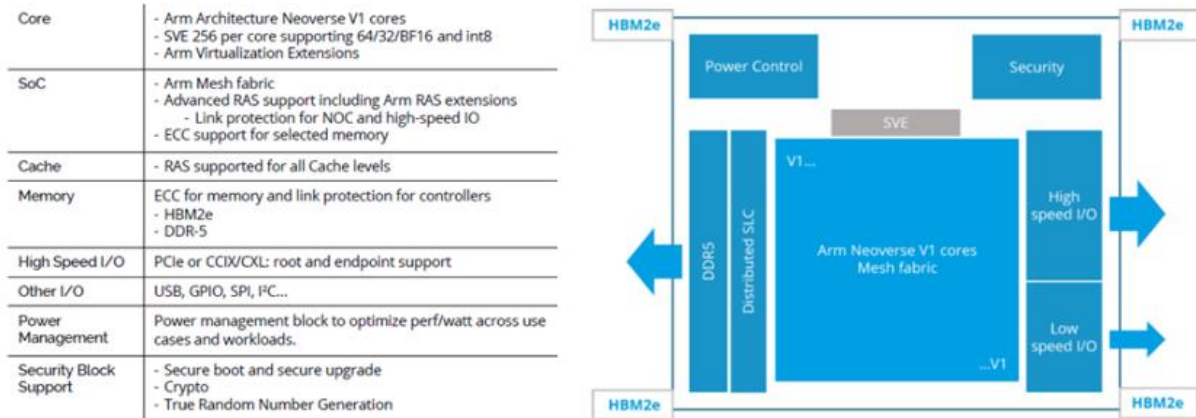


The effort of the project is divided in two main branches: one is led by SiPearl and focuses on the development of an ARM-based General-Purpose Processor (GPP) named RHEA, the other is jointly guided by a consortium of academic and industrial partners, collaborating for the development of a collection of RISC-V-based accelerators called EPAC.

The EuroHPC JU program has organized its various projects in a manner that facilitates the integration of the technological outcomes of EPI into two distinct pilot projects: the European Pilot for Exascale (EUPEX) and the European Pilot using Independent, Local and Open Technologies (EUPILOT). EUPEX is set to construct systems powered by the RHEA, while EUPILOT will incorporate technologies based on RISC-V stemming from EPAC. This strategic organization allows for practical implementation and testing of the advancements achieved by the European Processor Initiative in real-world pilot projects.

Through these pilot projects, which will provide access to prototype systems, EPI will also develop collaborations with HPC Centers of Excellence such as ChEESE-2P (see Figure 1).

Figure 2: Rhea Exascale Chip Characteristics



2.2 – EUPEX

As already mentioned, the European Pilot for Exascale (EUPEX) project is another important piece of the EuroHPC JU project. Its main goal is to gather and integrate European technologies from various domains, to create the first all-European platform for HPC: from system architecture and hardware, up to software and applications.

The EUPEX prototype is designed to be open, scalable, and flexible thanks to an innovative modular architecture, and it adopts cutting-edge hardware specifically targeted at high-performance and energy-efficient computing, such as the first-generation RHEA processor developed in the EPI project.

Together with the hardware, the EUPEX project also features the creation of a suitable software ecosystem for the pilot, designed by considering the modularity of the system and the needs of a set of key applications identified by the project.

Current EUPEX hardware/software infrastructure

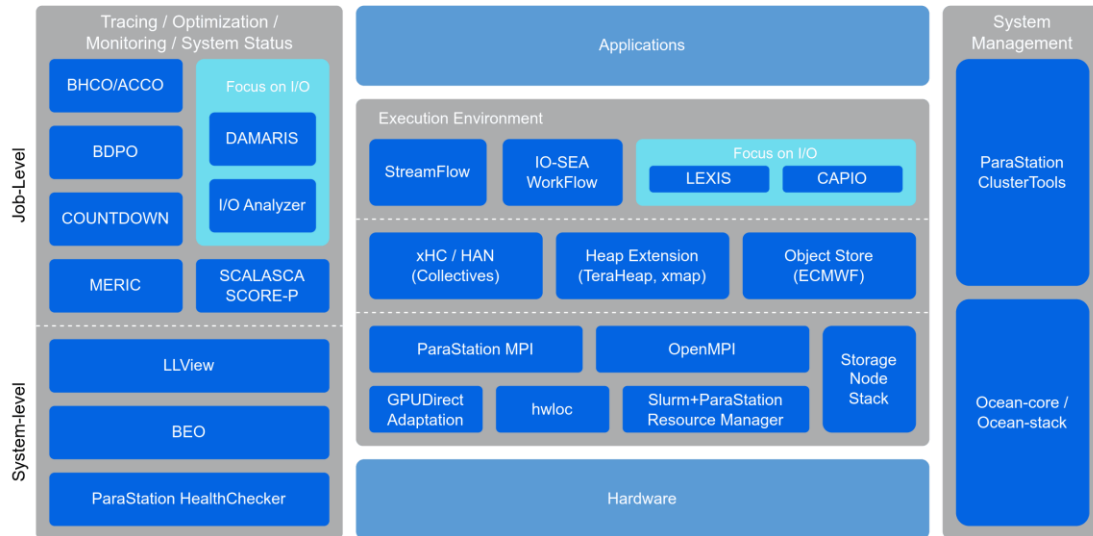
Processor	SiPearl RHEA Exascale Chip
Processor Architecture	ARM Neoverse V1
Core Count	Confidential
Socket Count	2
Accelerator	TBD GPU
Accelerator Architecture	TBD

Table 1: Overview of EUPEX Prototype's Hardware Components

The main component of the EUPEX prototype will be the Rhea chip developed by SiPearl (see Table 1). The number of cores of the processor is still not fixed. It is known however, that the cores will be based on an Arm Neoverse V1 design with

256bit Scalable Vector Extension (SVE) and Arm Mesh fabric, and that the processor will support HBM2e memory. Further technical details about the chip can be found in Figure 2.

Figure 3: Overview of EUPEX Prototype's Software Stack



Before the Rhea chip reaches general availability, co-design partners will have access to proxy systems/architectures sharing common characteristics with the prototype. The list of the available systems is the following:

- **Fujitsu A64FX:** this CPU includes SVE 512bit wide and HBM2. However, the internal pipeline and memory (hierarchy, size and generation) are different than the one planned for Rhea.
- **Amazon Graviton3:** this CPU implements Arm Neoverse V1 architecture identical to RHEA including SVE. From the micro-architecture point of view, this processor is the closest to Rhea.
- **Intel Sapphire Rapids HBM:** this processor is convenient for studying the impact of using HBM vs DDR5.
- **Ampere Altra:** this processor implements Arm Neoverse N1, thus no SVE but it is convenient to work with ARM.
- Potentially in the future GRACE with Arm Neoverse V2 (SVE2) and DDR5X and GRACE HOPPER and Graviton 4 which are supposed to use similar cores.

As previously stated, one of the objectives of EUPEX is to provide a software ecosystem for the pilot based on European technologies. This includes a management software stack, an execution environment, some tools for tracing, optimization, monitoring and system status, and a multi-tier storage architecture (see Figure 3). Its design will not only take into consideration the needs of the key applications identified in EUPEX, but also those of the system operators for the management of large-scale Modular Supercomputing Architecture (MSA) systems.

2.3 – EUPILOT

The European Pilot using Independent, Local and Open Technologies (EUPILOT) project aims to create a comprehensive demonstrator for accelerators designed for a pre-exascale system. Utilizing European and open-source technologies, the project plans three chip tape-outs. The first tape-out involves a test chip to validate the use of 12nm technology, migrating from a previous 22nm node. The second and third tape-outs will feature a vector accelerator with up to 16 cores and a machine learning/stencil accelerator with up to eight cores, developed concurrently.

These accelerator chips will be integrated with low-power double data rate (LPDDR) memory, forming chip modules for standard accelerator boards. The project adheres to the Open Compute Project Accelerator Module (OAM) standard to build these accelerator systems, paired with host servers. The host-accelerator combinations are deployed in liquid immersion tanks.

EUPILOT's success relies on open-source and European-developed software technologies, demonstrating Europe's capabilities for technological independence. The project's layered software stack spans applications, operating systems, libraries, runtime systems, resource managers, compilers, and toolchains. This stack showcases functionalities in two different accelerators (vector and machine learning/stencil) supporting traditional HPC applications, AI workloads, and novel HPC/AI applications.

The integration of European IP accelerators with the customized software stack aims to achieve future Exascale performance levels at an unprecedented scale. Standards like OpenMP and MPI (for HPC), open-software frameworks like TensorFlow and oneDNN (for AI), and European developments such as Tarantella, DaCe, TAMPI, or DLB contribute to programming productivity and usability.

Current EUPILOT hardware/software infrastructure

Accelerator #1	VEC (RISC-V scalar core coupled with 256 double precision vector processing unit)
Accelerator Architecture	RISC-V
Core Count	16
Accelerator #2	MLS (RISC-V scalar core focusing on network and stencil acceleration)
Accelerator Architecture	RISC-V
Core Count	32 (to be confirmed)

Table 2: Overview of EUPILOT Prototypes' Main Hardware Components

The current VEC prototype includes a RISC-V micro-tile, which is composed of an Avispado scalar core developed by Semidynamics, connected to a Vitruvius vector processing unit (VPU) with eight lanes (see Table 2). Each lane has a Floating-Point Unit (FPU) developed by the University of Zagreb. The micro-tile also has a Home Node and an L2 cache, which were respectively designed by Chalmers³ and FORTH⁴.

For software development the VEC micro-tile has been mapped into an FPGA. The micro-tile is connected to a few ancillary FPGA blocks in order to interface with on-board main memory (DDR4), the PCIe bus, and the Ethernet PHY. The most disruptive feature of this design is the presence of a vector processing unit that is capable of operating vectors of 256 double-precision elements (i.e., 16 kbits-wide vector registers).

The VEC-SDV@FPGA is running a single core at 50 MHz. The software stack offers a standard Linux distribution coupled with standard repositories for the distributions of all RISC-V packages currently available for RISC-V Linux systems.

2.4 – Connections with ChEESE-2P

The three projects mentioned in the previous sections share a collective goal of developing innovative platforms that will be fundamental for ensuring Europe's sovereignty in the computing domain. To facilitate this task and help partner institutions in the development of the prototypes, other projects founded in the framework of the EuroHPC JU are required to engage in a "co-design" process for hardware and software.

Co-design of a computing platform is a process that considers the entire system stack, from the underlying hardware to the applications. The latter have the crucial role of providing insights into compute and data movement patterns, that can later be used to optimize the hardware and software design of the prototype. In turn, the applications will ideally benefit from running natively on these new platforms, even before they are available for commercial use, thus removing the need to port the application later down the line.

The Center of Excellence for Solid Earth (ChEESE), now at its second iteration (ChEESE-2P), is one of the projects selected for the co-design of European HPC hardware prototypes. The related activities are delegated specifically to ChEESE-2P Work Package 3 (WP3), whose main goals are the selection and preparation of a set of applications based on the project's flagship HPC codes, and their use as co-design vehicles for EPI, EUPEX and EUPilot prototypes.

Mini-apps for co-design (ChEESE-2P WP3)

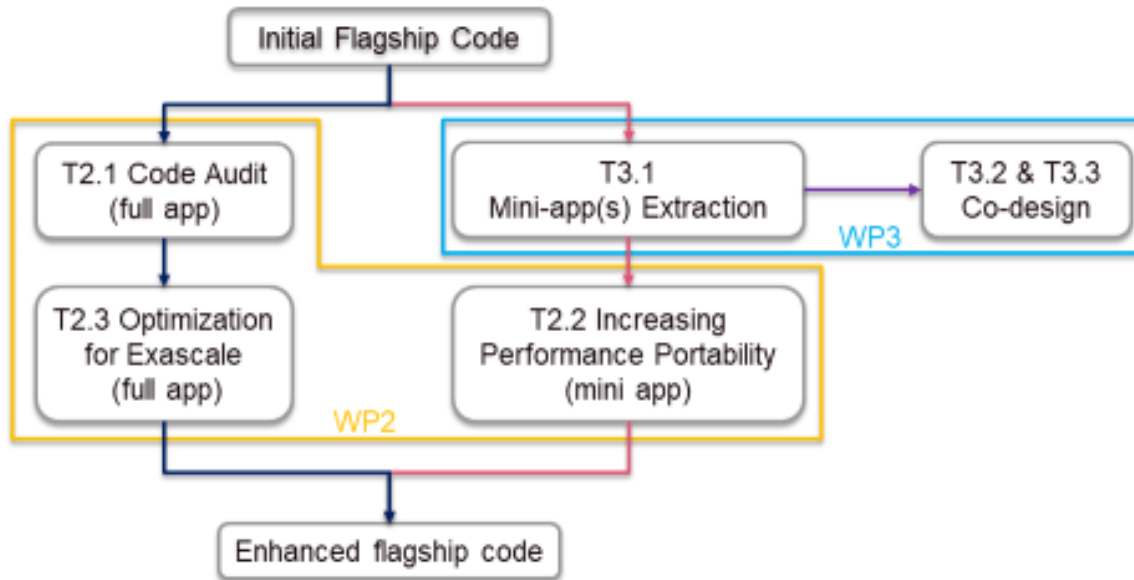
HPC applications are very complex pieces of software, often consisting of millions of lines of source code, and involving multiple programming languages, third-party library dependencies, and other complexities. For this reasons, co-design of HPC systems is carried out with the aid of application proxies, which usually take the shape of mini-application, i.e., scaled-down versions of target applications that

require reduced amounts of resources to run, while still retaining the fundamental behavior of the full codes.

Mini-apps are fundamental for the activities of ChEESE-2P WP3, and each flagship code participating in WP3 must provide one. For this reason, the first of the three tasks of WP3, namely T3.1, is entirely focused on helping developers in preparing mini-apps, which are later going to be used for co-design of the EUPEX prototype in T3.2, and of EPI and EUPilot prototypes in T3.3.

The main goal of the present document is to identify the needs of WP3 mini-apps, mostly in terms of third-party library dependencies and hardware compatibility, and present them in a unified reference, together with any further information on repositories, functionality, and validation methods for each mini-app available at the time of writing. This list of mini-apps is presented in Section 3 – State of the art of mini-apps at M12.

Figure 4: ChEESE-2P WP2 & WP3 Overview



Mini-apps for performance portability (ChEESE-2P WP2)

Beyond the usage in WP3, mini-apps are of fundamental importance also for other ChEESE-2P tasks included in WP2. A full representation of the expected workflow of WP2 and WP3 is provided in Figure 1. Figure 4, the WP2 task that is mainly concerned with mini-apps is T2.2, whose final goal is to increase the performance portability of flagship codes. This requires a cycle of experimentation and testing with different accelerated programming models, to find the one that best represents the balance between application performance and portability desired by the code developers.

Analogously to the case of co-design, due to their size and complexity, full codes are not suitable for use in this kind of iterative workflow. On the other hand, mini-apps allow quick prototyping and verification of various possible solutions, thus making them ideal vehicles for experimentation.

Mapping of mini-apps to co-design targets

To conclude this section, we provide some general considerations on the mapping of mini-apps to prototype systems. In general, for a mini app to run out-of-the-box on a new platform (such as the ones featured in the prototypes), there must be a certain level of compatibility between the architecture for which the mini-app was developed, and the architecture of the target platform.

This should not pose a problem for the EUPEX prototype, since the platform's architecture (Arm CPU + GPU) is the one that most current computing systems adopt. Thus, given a compiler capable of generating code for the Arm Neoverse

V1 instruction set with SSE, a runtime stack that supports the common parallel programming models (MPI, OpenMP), and dedicated fworking drivers for the GPU, all mini apps should be able to run on the EUPEX prototype.

The situation is a bit more complex for EPI and EUPilot prototypes. In these cases, the platform architecture is based on completely new accelerator types that rely on different parallelization strategies with respect to GPUs. This means that mini-apps versions that were adapted to run on GPUs will probably be unable to run efficiently (if at all) on these platforms. In these cases, co-design will possibly require an adaptation of the mini-apps, or the creation of some kind of translation layer to allow these GPU-based mini-apps to run natively on the new accelerators. In any case, ChEESE-2P mini-apps all provide multiple versions for CPU and GPU, thus co-design partners will be free to choose the one that best suits their needs.

3 – State of the art of mini-apps at M12

This section provides information on the status of T3.1 mini apps. For each mini-app we provide a short description of its differences with respect to the full code, the available versions, and the repository where it can be found (if available). We also provide an account of available tests or examples with associated validation methods, and of performance metrics expressed in the mini-app output, if any. A short presentation of software packages mentioned in the tables of each mini-app can be found in Appendix B – Mini-app software dependencies glossary.

The mini-apps presented in this section are the only ones available at the time of writing (M12). On the other hand, T3.1 is scheduled to end at M18, and new mini-apps could still be prepared if need arises. For this purpose, Appendix A – Support Activities and Materials - provides support materials and guidelines that should be helpful in the process of extracting new mini-apps.

3.1 - SeisSol

SeisSol is a software package for the simulation of seismic wave propagation and dynamic ruptures. The associated mini-app runs the computational kernels of the full code using random data to test their performance.

Mini-app	SeisSol-proxy
Repository	https://github.com/SeisSol/SeisSol
Versions	CPU, GPU (backends: SYCL + CUDA, HIP, SYCL)
Examples	N/A
Validation	N/A
Built-in Metric	Kernel execution times, FLOPS, bandwidth
Dependencies	Python, Numpy, Eigen, libxsmm, PSpaMM For GPU version: GemmForge, ChainForge (only for CUDA/HIP)
Notes	The mini app is available as compilation target inside the full code repo. Since it uses random data to run compute kernels, it does not require example inputs or validation.

3.2 - SPECFEM3D

SPECFEM3D is a software for the simulation of 3D seismic waves in any region of the Earth, handling a variety of physical conditions and models. The associated mini-app comes in three different flavors, respectively performing a simulation of simplified models representing elastic isotropic, elastic anisotropic and visco-elastic isotropic materials.

Mini-app	Mini-SPECFEM3D
Repository	https://gitlab.com/specfem_cheese_2p/mini_specfem/specfem_mini_app
Versions	CPU, GPU (backends: CUDA)
Examples	Three test cases with different sizes for each flavor
Validation	Comparison with reference solutions provided in the repo
Built-in Metric	Runtime
Dependences	MPI
Notes	-

3.3 - Tandem

Tandem is a software package for modelling sequences of earthquakes and aseismic slips (SEAS models). Its core is composed of solvers for Poisson and linear elasticity problems. The associated mini-app uses the solvers to compute solutions for some simple models representing these two categories of problems.

Mini-app	Tandem-static
Repository	https://github.com/TEAR-ERC/tandem
Versions	CPU, GPU (backends: CUDA/OpenCL - via PETSc)
Examples	Two example models (examples/elasticity, examples/poisson)
Validation	Embedded in application
Built-in Metric	Warmup time, Solver execution time
Dependences	Python, Numpy, MPI, Eigen, zlib, Lua, METIS, ParMETIS, PETSc

Notes

The mini app is available as compilation target inside the full code repo. Validation of examples models is possible since they have analytical solutions for comparison with numerical results.

3.5 - xSHELLS

XSHELLS is a high-performance code for simulating incompressible fluids in a spherical cavity. Due to having very few dependences, and a dedicated set of input test cases, the main code is also used as mini-app.

Mini-app	Mini-xSHELLS
Repository	https://gricad-gitlab.univ-grenoble-alpes.fr/schaeffn/xshells/-/tree/cheese-miniapp-fp32
Versions	CPU, GPU (backends: CUDA, HIP)
Examples	Multiple benchmark models, each with multiple sizes
Validation	Python script (test.py)
Built-in Metric	Average time per simulation step
Dependences	MPI, FFTW, Python (to run test script)
Notes	It is suggested to run the geodynamo benchmark for testing.

3.5 - HySEA

Tsunami-HySEA is a numerical model designed to simulate quake-generated tsunamis. The full code is not yet open-sourced, but a reduced version targeted to running on a single node is available in public online repositories. The mini-app for the code corresponds to this reduced single-node version.

Mini-app	Tsunami-HySEA (open source version)
Repository	https://github.com/edanya-uma/TsunamiHySEA https://github.com/edanya-uma/TsunamiHySEA-SYCL
Versions	CPU, GPU (backends: CUDA, SYCL)
Examples	One example model in folder Mediterranean
Validation	Diffing NetCDF results file (requires external tool)
Built-in Metric	Runtime
Dependences	NetCDF
Notes	There is no dedicated CPU version, but the mini app can run on CPU only through the SYCL backend.

3.6 - FALL3D

FALL3D is a high-performance code for modelling of atmospheric passive transport and deposition of particles, aerosols, and radionuclides. The corresponding mini-app uses the numerical solver from the full code to solve an example model.

Mini-app	Mini-FALL3D
Repository	https://gitlab.com/fall3d-suite/mini-fall3d https://gitlab.com/fall3d-suite/mini-fall3d-gpu
Versions	CPU, GPU (backends: OpenACC)
Examples	Two example problems for CPU, one for GPU
Validation	Embedded in application
Built-in Metric	Runtime, iterations per second
Dependencies	MPI, NetCDF
Notes	-

3.7 - pTatin3D

pTatin3D is a software framework providing functionality to study long-term geodynamic processes. At the time of writing the present document, the mini-app is still under construction. The table below only contains confirmed details.

Mini-app	Mini-pTatin3D (viscous_sinker)
Repository	https://bitbucket.org/dmay/ptatin-total-dev/
Versions	CPU, GPU (backends: CUDA/OpenCL/HIP - via PETSc)
Examples	src/models/viscous_sinker with multiple running options
Validation	Embedded
Built-in Metric	Runtime, FLOPS
Dependencies	MPI, PETSc
Notes	The mini-app is a dedicated model (viscous_sinker) that uses a subset of the functionality of the library. Validation is possible since an analytical solution exists for the model. Example on main branch may be out of date, for testing

use *anthony_jourdon/stokes_law* or the branch with most recent commits.

3.8 – Summary Table

Mini-app	CPU	GPU	GPU Backend	Built-in Metric
Seissol-proxy			SYCL + SYCL/CUDA/HIP	Kernel times, FLOPS, bandwidth
Mini-SPECFEM3D			CUDA	Runtime
Tandem-static			PETSc	Kernel times
Mini-xSHELLS			CUDA/HIP	Average time per step
Tsunami-HySEA			CUDA/SYCL	Runtime
Mini-FALL3D			OpenACC	Runtime, iterations per second
Mini-pTatin3D			PETSc	Runtime, FLOPS

Ready
 In Development
 Not Available

Appendix A – Support Activities and Materials

Here we list the resources that were provided to code developers to help them in the process of preparing their mini-app.

A.1 – Webinar on Mini-app Extraction

Title	Mini-app extraction and their usage in the optimization, porting and benchmarking process
Date	September 18 th , 2023
Organizer	Eviden
Attendance	ChEESE-2P partners (CSIC, BSC, CNRS, LMU, TUM, UMA, INGV, CINECA, Sorbonne Université)

2 hour-long online webinar focusing on showcasing useful techniques and best practices for the extraction of kernels and mini-apps starting from full-fledged codes. Materials provided during the presentation have been uploaded in the WP3 section of the [official SACO repository for ChEESE-2P](#), together with a recorded version of the webinar.

A.2 – Guidelines for Mini-app Preparation

A set of guidelines for the preparation of mini-apps was presented during WP3 meetings. The main points covered are briefly summarized in the following list:

- Know your code,
 - Understand underlying data structures/numerical method,
 - Create easy/browsable view of the code,
- Know your test case,
 - Consider the most relevant test case,
- Benchmarking method,
 - Frequency stepping,
 - Compact/scatter,
 - MPI profile,
 - Hyperthreading? (If needed),
- Application characterization,
 - Identify optimization levers,
- Mini-Apps: How to work on big codes?
- Mini apps: how to use them?
- Case study: SPECfem3d

Appendix B – Mini-app software dependencies glossary

The following table contains short descriptions of software packages mentioned as dependencies for the mini-apps in Section 2 of this document. The entries are ordered alphabetically.

ChainForge	Code generator for optimized GPU matrix multiplications, developed as part of the SeisSol project.
Easi	Library for simplified initialization of models for grid-based simulation software
Eigen	C++ template library for linear algebra, with matrix, vector, numerical solvers and related algorithms
FFTW	C library to compute discrete Fourier transforms in one or more dimensions, both for real and complex data
GemmForge	Code generator for optimized GPU matrix multiplications, developed as part of the SeisSol project.
HDF5	High-performance library for using the HDF5 data format, suitable for fast I/O processing and storage
Libxsmm	Specialized library for dense and sparse matrix operations and deep learning primitives such as small convolutions.
Lua	Efficient, lightweight, and embeddable scripting language.
METIS	Suite of serial programs for graph and finite-element meshes partitioning.
MPI	Placeholder for a suitable library implementing the Message Passing Interface programming model
NetCDF	Suite of software libraries and data-formats for creating, accessing, and sharing array-oriented scientific data.
Numpy	Fundamental package for scientific computing in Python, providing optimized array classes and algorithms
ParMETIS	Suite of MPI-based tools extending functionality provided by METIS to parallel machines.
PETSc	Software package providing parallel algorithms and tools for the solution of partial differential equation. Supports MPI and GPUs via CUDA, OpenCL, Kokkos and HIP (experimental)
PSpaMM	Code generator for optimized sparse matrix multiplication
Python	Simple, and flexible programming language, suitable for scripting and high-level programming. It is also extensible thanks to its rich environment of packages and modules.
Zlib	Library for lossless data compression.

References

- ChEESE-2P - Center of Excellence in Solid Earth*. (n.d.). The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/research-innovation/our-projects/cheese-2p-center-excellence-solid-earth_en
- CHEESE*. (n.d.-d). <https://cheese2.eu/>
- EuroHPC-JU. (2023, November 28). The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/index_en
- The European Processor Initiative (EPI)*. (n.d.). The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/research-innovation/our-projects/european-processor-initiative-epi_en
- European Processor Initiative*. (2023, April 27). *EPI - European Processor Initiative*. <https://www.european-processor-initiative.eu/project/epi/>
- EUPEX*. (n.d.). The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/research-innovation/our-projects/eupex_en
- Eupex*. (2023, November 28). *Home - EUPEX*. EUPEX. <https://eupex.eu/>
- The European PILOT*. (n.d.). The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/research-innovation/our-projects/european-pilot_en
- The EuPilot Project*. (2023, November 21). The EuPilot Project. <https://eupilot.eu/>
- Minervini, F., Palomar, O., Ünsal, O., Reggiani, E., Quiroga, J. V., Marimon, J., Rojas, C., Figueras, R., Ruiz, A., González, A., Mendoza, J., Vargas, I., Hernández, C. A., Cabré, J. J., Khoirunisya, L., Bouhali, M., Pavón, J., Moll, F., Olivieri, M., . . . Cristal, A. (2023). Vitruvius+: an Area-Efficient RISC-V decoupled vector coprocessor for high performance computing applications. *ACM Transactions on Architecture and Code Optimization*, 20(2), 1–25. <https://doi.org/10.1145/3575861>
- Kovač, M., Dragić, L., Malnar, B., Minervini, F., Palomar, O., Rojas, C., Olivieri, M., Knezović, J., & Kovač, M. (2023). FAUST: Design and implementation of a pipelined RISC-V vector floating-point unit. *Microprocessors and Microsystems*, 97, 104762. <https://doi.org/10.1016/j.micpro.2023.104762>