Master in High Performance Computing

Performance Analysis of Scientific Applications on Multiple RISC-V Platforms

Supervisors:
Xavier Teruel (BSC),
Ivan Girotto (ICTP)

Candidate:
Ludwing Osmar Asturias Alquijay

11th edition

2024–2025

Acknowledgements

I would like to express my sincere gratitude to the International Centre for Theoretical Physics (ICTP) for providing me with the opportunity to pursue the Master in High Performance Computing (MHPC) program and for fostering a stimulating academic environment that has greatly contributed to my personal and professional growth. Along this journey, I have also had the privilege of meeting outstanding peers, whose collaboration and friendship have made this experience even more rewarding.

I am deeply thankful to my supervisor, Xavier Teruel, for his invaluable guidance, support, and encouragement throughout the course of this thesis. I would also like to extend my gratitude to Regina Gachomba for her support and guidance with the BSC performance analysis tools, which were crucial for the successful completion of this thesis. Additionally, I would like to thank Xavier Martorell for his valuable feedback and insights that have helped improve the quality of this work.

Abstract

This thesis presents a comprehensive performance analysis of a scientific application, SeisSol, executed on multiple RISC-V platforms available in the HCA cluster at the Barcelona Supercomputing Center (BSC). The evaluation is conducted using the methodology developed by the Performance Optimization and Productivity (POP) Center of Excellence and performance analysis tools developed at BSC. The selected application is representative of high-performance computing (HPC) workloads and has been optimized for parallel execution. The performance analysis includes both scalability studies and tracing-based investigations in single-node and multi-node configurations. Detailed execution traces are captured using the Extrae tool, and the resulting traces are analyzed with Paraver and Dimemas to extract relevant performance metrics and identify potential sources of inefficiency. A comparative analysis is also performed to better understand the performance of the application on RISC-V architectures compared to traditional x86_64 architectures.

Contents

List of Figures

Acronyms

ACME Accelerated Compute and Memory Engines
AI Artificial Intelligence
API Application Programming Interface
BSC Barcelona Supercomputing Center
BZL Barcelona Zettascale Lab
CFD Computational Fluid Dynamics
ChEESE Centre of Excellence for Exascale in Solid Earth
CINECA Consorzio Interuniversitario del Nord-Est per il Calcolo Automatico
CNN Convolutional Neural Network
CoE Center of Excellence
CPU Central Processing Unit
DGEMM Double-precision General Matrix-Matrix Multiplication
DVFS Dynamic Voltage and Frequency Scaling
EPAC European Processor Accelerator
EPI European Processor Initiative
EUPEX European Platform for Exascale
EUPILOT European Pilot using Independent, Local and Open Technologies
EuroHPC JU European High Performance Computing Joint Undertaking
FLOPS Floating Point Operations Per Second
FoA Focus of Analysis
FPGA Field-Programmable Gate Array
GEMM General Matrix-Matrix Multiplication
GPP General Purpose Processor
GPU Graphics Processing Unit
HCA Heterogeneous Computing Architecture
HDR High Data Rate
HPC High Performance Computing
IPC Instructions Per Cycle
ISA Instruction Set Architecture
MEEP MareNostrum Experimental Exascale Platform
MHPC Master in High Performance Computing

ML Machine Learning
MPI Message Passing Interface
NUMA Non-Uniform Memory Access
OpenMP Open Multi Processing
POP Performance Optimisation and Productivity
PUML Parallel Unstructured Mesh Library
RISC-V Reduced Instruction Set Computer V
rvv RISC-V Vector Extension
SCEC Southern California Earthquake Center
SDV Software Development Vehicle
UPC Universitat Politècnica de Catalunya
USGS United States Geological Survey
VPU Vector Processing Unit

Chapter 1

Introduction

High Performance Computing (HPC) applications are typically complex software systems that leverage parallel computing techniques to solve large-scale problems efficiently. These applications often involve multiple layers of parallelism, including distributed memory parallelism using MPI, shared memory parallelism using OpenMP, and sometimes accelerator-based parallelism using technologies such as CUDA or OpenCL.

In the HPC landscape, the emergence of an open standard Instruction Set Architecture (ISA) like RISC-V presents new opportunities and challenges for application performance optimization. Several projects and initiatives are exploring the potential of RISC-V architectures for HPC, aiming to leverage their flexibility, extensibility, and open-source nature to develop high-performance computing solutions that can compete with traditional architectures. In the European context, there is a growing interest in establishing a sovereign HPC ecosystem based on RISC-V architectures. The European High Performance Computing Joint Undertaking (EuroHPC JU) has funded several projects to promote the development and adoption of RISC-V in HPC. Among them, the European Processor Initiative (EPI) project focuses on processor technologies based on RISC-V and ARM architectures, while other projects like the European Platform for Exascale (EUPEX), the European Pilot using Independent, Local and Open Technologies (EUPILOT), and the MareNostrum Experimental Exascale Platform (MEEP) aim to develop a complete hardware and software stack around these processors. Other commercial development boards, such as the HiFive Premier, the Milk-V Pioneer, and the Banana Pi BPI-F3, are already available and provide accessible platforms for testing and optimizing HPC applications on RISC-V architectures.

This study analyzes the performance of a selected HPC application, SeisSol, on multiple RISC-V platforms.

SeisSol is a software package that simulates seismic wave propagation and earthquake dynamics using high-order discontinuous Galerkin methods. SeisSol is optimized for large-scale supercomputing platforms and supports CPU parallelism via hybrid MPI+OpenMP and MPI+CUDA/HIP/SYCL for GPU acceleration. It exhibits excellent performance and scalability

on a suitable range of supercomputing platforms. Several studies have been conducted to analyze and optimize the performance of SeisSol both on traditional HPC architectures, including multi-core CPUs and GPUs [25], [34] and on RISC-V-based systems leveraging long vector architectures [2], [3].

For the performance analysis on each RISC-V platform, a suitable benchmark setup is selected from the SeisSol community datasets. Then, a scalability study is conducted to evaluate the application's performance. Following the scalability study, a more detailed analysis of the application's parallel performance is conducted using a methodology developed by the Performance Optimisation and Productivity (POP) Center of Excellence (CoE) in HPC.

The POP CoE in High Performance Computing (HPC) has developed a set of performance metrics that provide a structured and consistent framework for evaluating application performance. These metrics are designed to be applicable across different programming models, including MPI, OpenMP, and hybrid approaches, and they focus on key aspects such as scalability, efficiency, and resource utilization. While the POP workflow is designed to be independent of any specific tool and can, to a certain extent, be applied with different performance analysis environments, this project adopts the instrumentation and analysis suite from BSC, including Extrae, Paraver, and Dimemas.

1.1 Motivation

The emergence of RISC-V as an open and extensible ISA offers a promising foundation for building high-performance and energy-efficient computing systems. However, the ecosystem surrounding RISC-V, including compiler toolchains and runtime environments, is still under active development, and its capability to efficiently support large-scale scientific workloads remains an open question. To bridge this gap, it is essential to perform comprehensive performance analyses on representative HPC applications. SeisSol, a high-order seismic simulation code, serves as an ideal case study due to its proven high scalability and reliance on advanced parallel programming models such as MPI and OpenMP. Evaluating its performance on RISC-V architectures provides valuable insight into the maturity and scalability of the current RISC-V software-hardware stack for scientific computing.

1.2 Objectives

The main objective of this project is to perform a comprehensive performance analysis of the SeisSol application on multiple RISC-V architectures using the POP performance metrics and methodology. For this purpose, the following specific objectives are defined:

- Conduct a scalability study of SeisSol on different RISC-V platforms, evaluating its performance across various configurations and problem sizes.

- Instrument SeisSol using the Extrae tool to capture detailed execution traces, enabling in-depth performance analysis.
- Analyze the captured traces using Paraver and Dimemas to extract relevant performance metrics, identify bottlenecks, and understand the parallel behavior of the application.
- Apply the POP performance metrics framework to evaluate the efficiency and scalability of SeisSol on RISC-V architectures.
- Compare the performance of SeisSol on RISC-V architectures with its performance on traditional x86_64 architectures.

1.3 Planning

Project planning is organized in two groups of tasks: The first group focuses on the project management and technical tasks, while the second group is dedicated to writing and reporting tasks. Below is a breakdown of the tasks involved in each group:

- **Management Tasks:**
  - During the *Project Planning and Coordination* phase, the project scope and objectives are defined, the HPC application to be analyzed is selected, and the methodology and tools to be used are outlined. A timeline with milestones and deliverables is also established and access to the necessary hardware and software resources is requested. This phase also establishes the necessary collaborative tools, communication channels, and periodic meetings to facilitate effective teamwork and coordination throughout the project.
  - Once the plan is in place, the next step is *Background Research*, which is a prerequisite for the technical tasks. This phase includes reviewing relevant literature, studying the computational characteristics of the selected HPC application, understanding efficiency metrics and the role of RISC-V architectures in HPC, and becoming familiar with the chosen analysis tools.
  - Another important prerequisite is the *Related Work Review*, which examines previous studies and projects that have applied comparable performance analysis methodologies or explored the optimization of HPC applications on RISC-V architectures.
  - The final prerequisite is *Methodology Development*, which involves defining specific steps and procedures to be followed during the performance analysis in order to achieve the project's objectives. This includes selecting performance metrics and benchmarks, setting up the analysis environment and experimental design, and establishing data collection and analysis procedures.
- **Technical Tasks:**

  - An *Initial Performance Analysis* is conducted to establish a baseline for the selected HPC application. This involves building and running the application on the target RISC-V architecture, capturing execution traces, calculating performance metrics, and identifying initial bottlenecks and inefficiencies.
  - The *Mid-term Review and Discussion* is one of the most important phases because it determines the direction of the second part of the project. During this phase, the progress made in the initial performance analysis is reviewed, and preliminary results are presented and discussed in order to develop a plan for the subsequent technical tasks.
  - Depending on the results of the initial analysis and the conclusions of the midterm review, a *Second Performance Evaluation* is carried out to further explore the performance of the application or complement the initial analysis focusing on specific aspects identified previously. This process may involve re-running the application, capturing new performance data, calculating updated metrics, and comparing the results with those of the initial analysis.
  - The final technical task is the *Final Review and Wrap-up*, in which the findings from the performance evaluations are consolidated and summarized. This phase also involves preparing for the final reporting tasks by organizing the data, results, and insights gained throughout the project.
- **Reporting Tasks:**
  - As the project progresses, *Chapter Writing* is carried out to document the applied methodologies and observed results. Each chapter is written either after the corresponding technical task is completed or in parallel with project development. This includes writing the Background, Related Work, Methodology, and Performance Analysis chapters.
  - The last step is *Final Report Preparation*, which involves consolidating all chapters into a cohesive document, adding conclusions and recommendations for future work, and conducting a comprehensive review to ensure clarity, coherence, and technical accuracy. This phase also involves formatting the document according to the required guidelines and preparing it for final submission.

The timeline is illustrated in fig. 1.1.

1.4 Project Coordination

The coordination of the project relies on a combination of collaborative tools and regular communication strategies. The project uses GitHub or GitLab for version control and collaborative coding, Overleaf with Git integration for document preparation, and cloud starage options for file sharing. These tools facilitate efficient collaboration between the project participants,

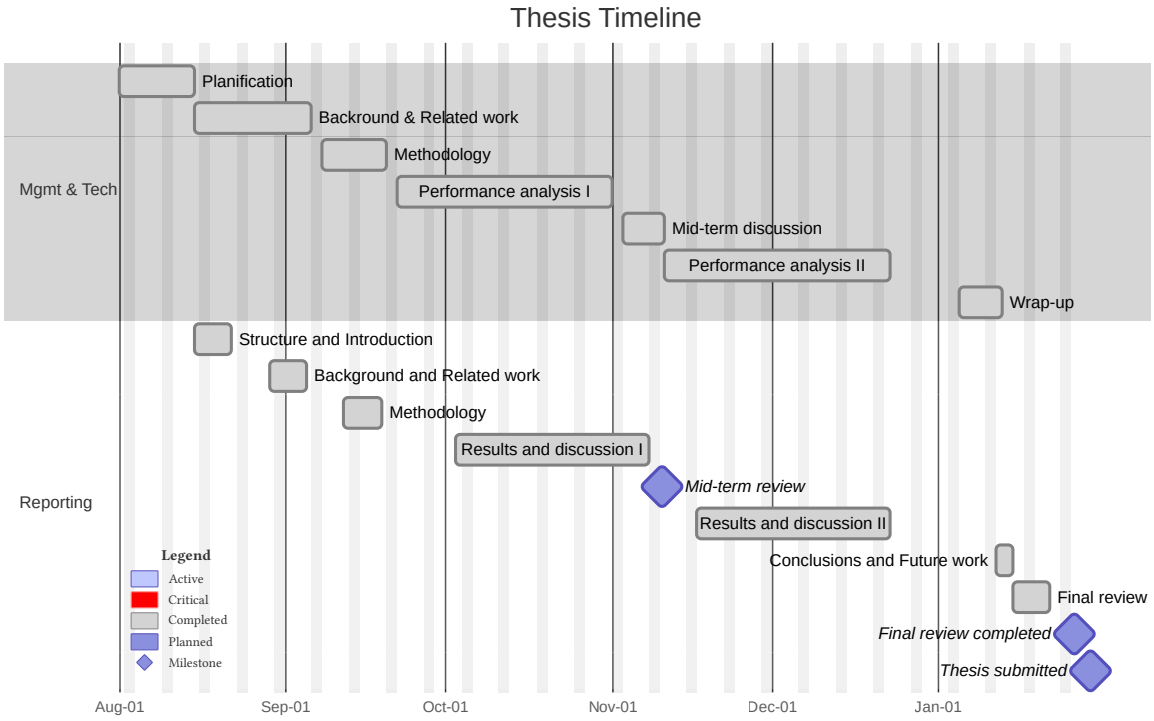


Figure 1.1: Gantt chart illustrating the project timeline, including management, technical, and reporting tasks.

enabling seamless contribution integration and easy access to project resources. To ensure alignment and consistent progress, weekly stand-up meetings are held to review and discuss ongoing tasks, and plan short-term activities. Additionally, monthly review meetings assess the overall project status, evaluate milestone progress, and adjust planning if necessary. Meeting notes and follow-up action are systematically recorded and tracked to ensure accountability and continuity in the project workflow.

1.5 Mid-term Review and Discussion

The mid-term review represented a key milestone in the project, providing an opportunity to evaluate progress, present preliminary results, and refine the direction of the remaining work. By this stage, the initial performance analysis of SeisSol on multiple RISC-V platforms had been successfully completed using the established methodology. The results were presented and discussed with the project supervisors, with particular attention given to the observed performance characteristics and the challenges encountered, including limited availability of performance counters on RISC-V platforms and unexpected execution bottlenecks. Feedback from the review emphasized the need for a reference benchmark based on a conventional architecture to better contextualize the RISC-V results. Consequently, it was decided to extend the performance evaluation to the Leonardo supercomputer, which is based on an x86_64

architecture. The second phase of the project therefore focuses on analyzing SeisSol on Leonardo using a comparable methodology, enabling a direct and meaningful comparison with the RISC-V results.

1.6 Document Structure

This document is structured as follows: First, the introduction describes the project's motivation, objectives, planning, and coordination. Then, the background necessary for developing the project is presented in chapter 2. This section covers programming models, performance tools, and performance analysis. Chapter 3 discusses related work, including the POP CoE, similar studies in the Master in High Performance Computing (MHPC) program, and the European RISC-V ecosystem. The methodology, including benchmark setup and performance methodology is detailed in chapter 4. Chapter 5 presents the performance analysis results carried out on the RISC-V Heterogeneous Computing Architecture (HCA) cluster at BSC, while chapter 6 discusses the performance analysis on the x86_64 Leonardo supercomputer and compares the results with those obtained on the RISC-V platform. Finally, chapter 7 discusses the results, summarizes the performance on both architectures, and outlines future work.

Chapter 2

Background

This chapter provides the necessary background information for understanding the performance analysis of SeisSol on RISC-V architectures. It covers the following key areas:

- an overview of the programming models used in SeisSol, specifically MPI and OpenMP;
- a description of the performance analysis tools developed by BSC, including Extrae, Paraver, and Dimemas;
- an explanation of the POP performance analysis methodology and metrics; and
- a brief introduction to the SeisSol application and its computational characteristics.

2.1 Programming Models

SeisSol employs a hybrid MPI+OpenMP parallel programming model to leverage both distributed and shared memory architectures. This section provides a brief overview of these two widely used programming models in HPC.

2.1.1 Message Passing Interface (MPI)

The Message Passing Interface (MPI) [46] is a portable message-passing standard designed to function on parallel computing architectures, primarily designed for distributed-memory systems where each process has its own local memory and data must be explicitly shared through message passing. It provides a set of communication protocols and functions that enable processes to exchange data and synchronize their actions, making it a fundamental tool for developing scalable and portable parallel applications in HPC environments. The standard is implemented by various libraries, such as MPICH and OpenMPI, which provide the necessary functions and protocols for message passing, and can be used with languages like C, C++, Fortran, and Python.

MPI supports several communication methods. Point-to-point communication involves sending a message from one specific process to another. These operations can be blocking (where the sender waits for the receiver to acknowledge receipt) or non-blocking (where the sender continues immediately after initiating the send). Collective communication involves

all processes in a group, such as broadcasting data from one process to all others. One-sided communication allows a process to directly read from or write to the memory of another process without requiring active participation from the target process [44].

2.1.2 Open Multi Processing (OpenMP)

OpenMP [48] is a directive-based Application Programming Interface (API) for developing parallel programs on shared-memory architectures. It supports shared-memory parallel programming in C, C++, and Fortran on many platforms.

OpenMP provides a set of compiler directives, library routines, and environment variables that enable developers to specify parallel regions in their code. Compiler directives use commands starting with `#pragma` in C/C++, or one of the `!$omp`, `c$omp`, or `*$omp` sentinels in Fortran to indicate parallel regions, work-sharing constructs, synchronization points, and data environment specifications. It is worth noting that compilers play a critical and indispensable role when creating a parallel application with OpenMP. They interpret the directives and generate the appropriate parallel code, which is then executed on the target architecture. If the compiler does not support OpenMP, the directives will be ignored, and the code will run sequentially. Therefore, it is essential to ensure that the compiler used for building the application supports OpenMP directives.

The runtime library is a collection of routines that manage the execution environment for OpenMP programs, enabling features like thread management, synchronization, timing, and device information access. And OpenMP-specific environment variables are available to control the execution of parallel code.

Since OpenMP version 4.0 [47], the standard has introduced support for heterogeneous systems, which consist of a host architecture and one or more external accelerator devices. The host architecture is where the program begins its execution, while the target accelerators (such as GPUs) are external devices attached to the host, capable of executing portions of the computation. As a key feature, the OpenMP offload model enables performance portability across different HPC clusters by abstracting the user from device-specific architectures.

2.2 BSC Performance Tools

Performance analysis tools are designed to assist developers in evaluating, tuning, and optimizing their codes. The BSC has developed open-source performance analysis tools that provide a detailed analysis of parallel applications. This analysis allows users to understand the behavior of parallel applications and identify performance-critical issues. These tools offer insight into both the application and the underlying system [6], [11]. This section provides an overview of the main tools used in this study.

2.2.1 Paraver

The core tool is Paraver [10], a trace-based performance analyser offering great flexibility for exploring and extracting information. Paraver provides two main visualizations: Timelines, which graphically display the evolution of the application over time, and tables (profiles and histograms), which provide statistical summaries. Together, these complementary views make it easier to identify parallel and computational inefficiencies such as load imbalance, serialization that limits scalability, the impact of cache and memory on performance, and regions with generally low efficiency.

The Paraver trace file is a set of three text files containing application activity (`.prv`), labels associated with numerical values (`.pcf`), and resource usage (`.row`). The trace file contains a textual header followed by a list of timestamped records that can be grouped into three categories: Events, states, and links (i.e. communications in the Paraver terminology). Events represent instantaneous occurrences at a specific timestamp. Examples include entry and exit points of user functions, MPI routines, OpenMP parallel regions, and CUDA kernels. Each event record stores two integer values that identify the activity. States represent time intervals during which a thread remains in a specific status. They indicate, for instance, whether the application was waiting at an MPI or OpenMP barrier, waiting to receive a message, or performing a memory transfer. Links represent relationships between two objects in the trace. Common examples include communication between processes (MPI applications), task migration between threads (OmpSs applications), or memory transfers (CUDA/OpenCL applications) observed during execution.

In addition to visualization, Paraver also allows to manipulate trace files: filtering, cutting, combining, and other operations. It also supports the computation of derived metrics and the extraction of specific data subsets for further analysis. Some examples of the types of information that can be extracted from Paraver traces are: useful duration, MPI calls and their duration, message sizes, OpenMP outlined functions, and hardware counters such as Instructions Per Cycle (IPC), frequency, and cache misses.

2.2.2 Extrae

Extrae [8], a dynamic instrumentation package, is the tool devoted to generate Paraver trace files for a post-mortem analysis. It supports multiple parallel programming models, including MPI, OpenMP, OmpSs, and CUDA. Extrae instruments the code to record runtime events, performance counters, and source code references. It can automatically instrument applications or allow users to manually instrument their code through its dedicated API. Extrae does not only offer the possibility to instrument the application code, but also offers to use sampling mechanisms to gather performance data. Adding sampling capabilities into Extrae allows providing performance information of regions of code which has not been instrumented.

2.2.3 Dimemas

Traces can also be produced by the Dimemas simulator [7], a message-passing simulation tool that models application execution on an abstract machine defined by network and CPU characteristics. Dimemas receives a Paraver trace file as input. Using the information in the file, Dimemas simulates the execution of the application on a user-defined target platform. This produces a new Paraver trace file, enabling direct comparison between simulated and real executions. This helps evaluate the potential benefits of a faster network or improvements from better workload balancing.

Since the trace file format is open, third-party translators can convert traces from other performance tools into the Paraver format, expanding compatibility and integration possibilities.

2.3 Performance Optimization and Productivity

The POP CoE has developed a comprehensive methodology for performance analysis and optimization of parallel applications [52]. This methodology is designed to systematically identify performance bottlenecks and inefficiencies, providing a structured approach to improving application performance. Furthermore, the structured methodology facilitates the comparison of performance across different versions of the same application or between different applications, promoting best practices in parallel programming and optimization.

This section explains the key steps of the POP performance analysis methodology, which are also applied in this study.

2.3.1 Focus of Analysis

Performance analysis tools such as Paraver enable the visualization of execution data in a time-related manner, for example through timelines that represent the chronological sequence of application phases. These views facilitate the identification of execution characteristics, including the distribution of time among functions, the activity of processes and threads, and the presence of computational hot-spots.

On the basis of this overview, one or more Focus of Analysiss (FoAs) can be defined. A FoA denotes the portion of the execution on which the remainder of the analysis will focus. This procedure allows the study to concentrate on the most relevant and representative sections of the application while limiting the volume of data to be analyzed. The FoAs may correspond to hot-spots that dominate runtime, selected loop iterations, or processes and threads exhibiting anomalous behavior. Once these regions have been identified, more detailed investigations can be undertaken in a targeted and systematic manner.

2.3.2 Scalability Analysis

Once the regions of interest have been identified, the next step is to investigate the scalability of the application. Scalability describes how the execution time evolves as the number of processing elements is varied and is typically studied in two forms. Strong scaling examines how performance changes when increasing the number of cores for a fixed total problem size, thereby highlighting potential bottlenecks in parallelization and communication. Weak scaling, in contrast, investigates how the runtime evolves when both the problem size and the number of cores are increased proportionally, thus reflecting the efficiency of handling larger workloads on larger systems. Assessing both aspects provides a more complete picture of the application's ability to utilize additional computational resources effectively [52].

2.3.2.1 Speedup and Efficiency

The scalability can be shown in various ways, such as runtime, speedup or efficiency.

Runtime is the time it takes to execute the application.

Speedup $S(p) = {}^{T_1}/_{T_p}$ shows how much faster a parallel program runs compared to a reference version (the baseline is usually a serial run), where T_1 is the execution time on one computing unit, and T_p is the time on p computing units. If the program cannot be executed serially also a higher number of processes and threads can be used. Speedup plots make it is easy to differentiate the scaling behavior of different regions.

Efficiency $E(p) = {}^{S(p)}/_{p}$ indicates how effectively parallel resources are being used. Efficiency values range from 0 to 1, where 1 indicates perfect efficiency.

These can help identify regions with the greatest potential for parallelization improvements. However, such speedup and efficiency plots lose information about the actual runtime. It does not necessarily make sense to focus on a region with poor scaling behavior if it only accounts for a small percentage of the total runtime, as predicted by Amdahl's law, which states that the overall speedup of a parallel program is limited by the fraction of serial execution within it.

It is important to note that the scalability of an application is not solely determined by the number of processes or threads used, but also by the underlying system. This information is crucial for interpreting the results correctly. Such details include the version of the application, the dataset, the compiler, the libraries, the hardware architecture, and the system software.

2.3.3 POP Standard Metrics

The POP Standard Metrics [52], [53] provide a structured approach to evaluating how effectively a parallel application utilizes available resources. Metric values typically range between 0 and 1, where higher values indicate a more efficient use of computational resources with respect to the aspect under consideration. They are organized in a hierarchical manner, beginning with Global Efficiency, which gives a high level overview of overall performance. From there, the hierarchy enables the examination of program behavior at a more refined level, thereby

identifying specific performance bottlenecks or sources of inefficiency. The hierarchy of the POP efficiency metrics is shown in fig. 2.1.

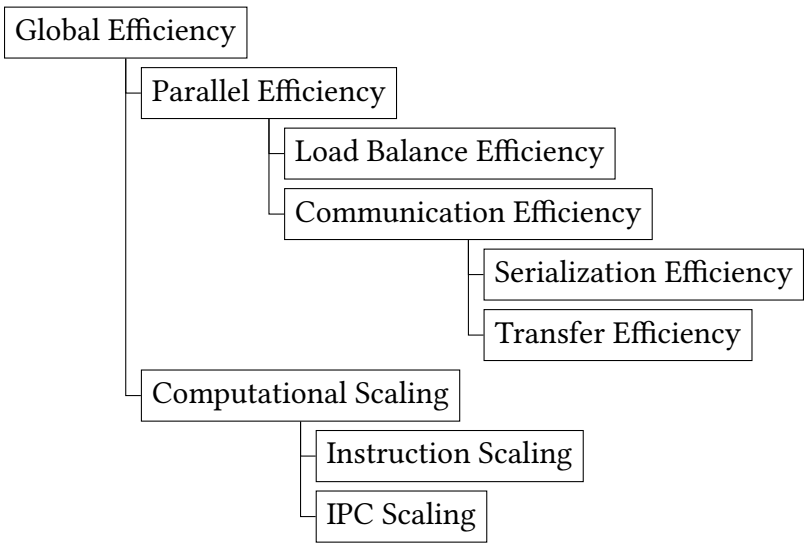


Figure 2.1: Hierarchy of the POP efficiency metrics. It is a multiplicative hierarchy, where each metric is a product of its sub-metrics.

- **Global Efficiency** is the top-level metric that provides an overall measure of how efficiently the application utilizes parallel computational resources. It is defined as the product of the Parallel Efficiency and Computational Efficiency metrics:

$$\mathrm{GE} = \mathrm{ParE} \times \mathrm{CompE}$$

  - **Parallel Efficiency** captures the impact of parallelization on performance. It reflects how evenly work is distributed and how much overhead arises from communication or synchronization.

$$\mathrm{ParE} = \mathrm{LBE} \times \mathrm{CommE}$$

    * **Load Balance Efficiency** indicates how evenly computational work is distributed across all processes or threads. Imbalanced workloads, which are typically observed at synchronization points, can result in idle time and reduced performance. It is defined as:

$$\mathrm{LBE} = \frac{\overline{T_{\mathrm{comp}}}}{T_{\mathrm{comp,max}}}$$

      where $\overline{T_{\mathrm{comp}}}$ is the average computation time across all processing units, and $T_{\mathrm{comp,max}}$ is the maximum computation time among all processing units.

* **Communication Efficiency** accounts for overheads due to communication among parallel entities. It can be computed as:

$$\text{CommE} = \max_i \left\{ \frac{T_{\text{comp},i}}{T_{\text{comp},i} + T_{\text{comm},i}} \right\}$$

where $T_{\text{comp},i}$ is the computation time of process i, and $T_{\text{comm},i}$ is the communication time of process i. It can be further decomposed into two sub-metrics:

$$\text{CommE} = \text{SerE} \times \text{TranE}$$

  · **Serialization Efficiency** can be computed similarly to the Communication Efficiency:

$$\text{SerE} = \max_i \left\{ \frac{T_{\text{comp},i}}{T_{\text{comp},i} + T_{\text{wait},i}} \right\}$$

where $T_{\text{wait},i}$ is the time process i waits for other processes to reach a communication point. Even on an ideal network with no communication overhead, T_{wait} can be non-zero due to synchronization points in the code, such as barriers, blocking sends, or collective operations. Such ideal network can be simulated with Dimemas.

  · **Transfer Efficiency** accounts for the time spent transferring actual data between processes, which can be significant in applications with large data transfers.

$$\text{TranE} = \max_i \left\{ \frac{T_{\text{comp},i}}{T_{\text{comp},i} + T_{\text{tran},i}} \right\}$$

where $T_{\text{tran},i}$ is the time process i spends transferring data over the network.

– **Computational Scaling** represents the speedup achieved by the parallel application when considering only the computational time, without accounting for communication or synchronization overheads. In practice, computational efficiency is evaluated by taking the execution time of a reference run with a certain number of processes as the baseline, and then comparing the speedup obtained when the application is executed with a larger number of processes relative to this baseline. Metrics based on processor hardware counter data can be used to investigate possible causes of poor computation scaling.

  * **Instruction Scaling** measures changes in the number of executed instructions relative to a baseline.
  * **IPC Scaling** is a measure of how the number of IPC scales with the number of processes. IPC may decrease when more processes are used because hardware resources such as caches, memory bandwidth, or network interconnects are

shared among a larger number of processes. In this case, individual cores may stall more frequently while waiting for data, leading to a lower IPC. Conversely, IPC can increase in some cases, for example under strong scaling, where smaller partitions of work per process may improve cache locality and raise the cache hit rate, allowing instructions to be executed more efficiently.

* **Frequency Scaling** quantifies the variation between the base frequency of a processor and the effective frequency achieved during execution. It reflects how the hardware dynamically adjusts clock frequency in response to power, thermal, or instruction mix constraints, such as when executing vector instructions.

These metrics provide a comprehensive framework for evaluating the performance of parallel applications and set the stage for further optimization efforts.

2.3.4 POP Hybrid Metrics

Hybrid parallelism is a common programming paradigm in HPC that combines multiple parallel programming models, such as MPI for distributed memory and OpenMP for shared memory, within a single application. This approach allows developers to leverage the strengths of both models to optimize performance on modern HPC systems, which often consist of clusters of multi-core nodes. The POP Hybrid Metrics [29], [55] extend the standard metrics to account for the complexities introduced by hybrid parallelism.

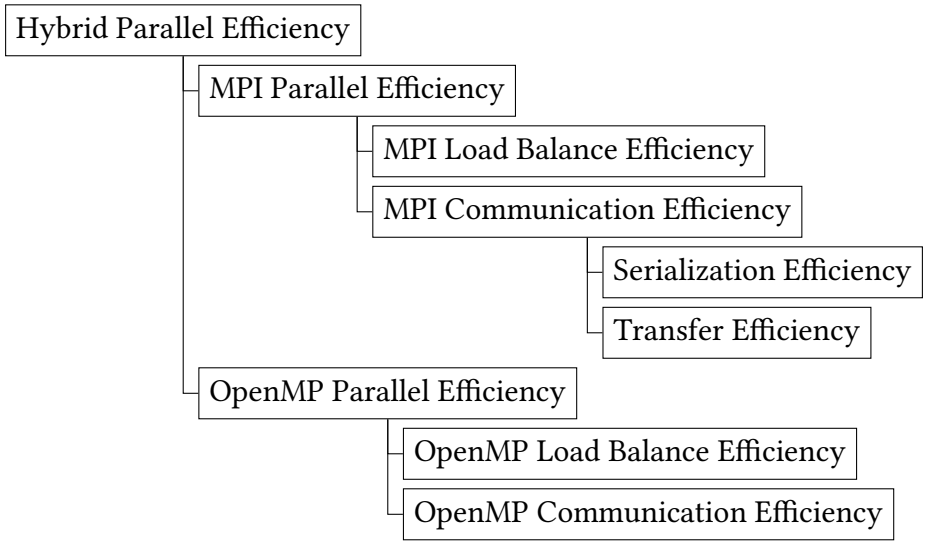


Figure 2.2: Hierarchy of the POP efficiency metrics for hybrid applications. It is a multiplicative hierarchy, where each metric is a product of its sub-metrics.

The hierarchy of the POP hybrid efficiency metrics is shown in fig. 2.2. The top-level metric, Hybrid Parallel Efficiency, measures the overall efficiency of the hybrid application

and can be decomposed into global load balance and communication efficiencies for both MPI and OpenMP components.

Although the purpose of the POP methodology is to evaluate the performance of parallel applications and provide developers with optimization insights, it can also be used to evaluate the performance of the same application on different machines.

2.4 SeisSol

SeisSol [27] is an open-source high-performance computational seismology software to simulate complex earthquake scenarios developed by the Centre of Excellence for Exascale in Solid Earth (ChEESE) CoE as one of its flagship codes. SeisSol is written in C++ and optimized for large-scale supercomputing platforms [16], [25], [34], [69], supporting CPU parallelization via hybrid MPI+OpenMP and MPI+CUDA/HIP/SYCL for GPU architectures. It is based on the Discontinuous Galerkin (DG) method, the Arbitrary high-order DERivative (ADER) time integration scheme, and the Local Time Stepping (LTS) technique [15], [26], [39]. The codes operates on unstructured tetrahedral meshes, and follows a design based on custom Domain Specific Language (DSL) through the open source library YATeTo [68] that generates highly optimized code for the small General Matrix-Matrix Multiplication (GEMM) routines that dominate the computational cost of the solver. The design of SeisSol is modular, meaning that different physical models, numerical methods, and optimization techniques can be activated or deactivated at compile time or runtime via configuration files. The basic code structure of SeisSol is organized into several main directories, as summarized in table 2.1.

Table 2.1: Main directories in the SeisSol source code repository.

Directory	**Description**
src/	Main source code of SeisSol
codegen/	Code generation tools and scripts
preprocessing/	Input data preparation tools and scripts
postprocessing/	Validation and visualization tools

The `src/` directory contains the core implementation of SeisSol, including modules for mesh handling, numerical solvers, time integration, and parallelization. For compiling SeisSol, the following dependencies are required during the build process:

- A C++17 compliant compiler, such as GCC, Clang, or Intel OneAPI.
- CMake as the build system.
- NumPy and setuptools Python packages.
- MPI
- HDF5

- easi

Additionally, for optimal performance, it is recommended to install a code generator for GEMM operations, a mesh partitioning library, and the NetCDF and ASAGI libraries for input/output operations. SeisSol supports the following CPU code generators:

- LIBXSMM: A library for small matrix multiplications, optimized for x86_64 architectures with SSE, AVX, AVX2, AVX-512, and AMX instruction sets. Its JIT variant also supports ARM and RISC-V architectures.
- PSpaMM: A code generator for small matrix multiplications, designed to support x86_64, ARM64, and RISC-V architectures.
- Eigen: A C++ template library for linear algebra.
- MKL: Intel's Math Kernel Library (MKL).
- BLAS libraries: Other BLAS libraries such as OpenBLAS or BLIS can also be used, although they may not be as optimized for small matrix multiplications as LIBXSMM or PSpaMM.

It is suggested to install both LIBXSMM and PSpaMM for CPU code generation (or, as an alternative, only PSpaMM). Both of these two code generators are used for different matrix multiplications; LIBXSMM by default for dense-dense matrix multiplications, and PSpaMM for dense-sparse and sparse-dense matrix multiplications; which are used for most operations in SeisSol. Without PSpaMM installed, those operations are converted to dense-dense matrix multiplications for LIBXSMM or Eigen instead.

For a good load balance on large clusters, SeisSol utilizes a mesh partitioning library during the startup of the simulation. The supported mesh partitioning libraries are: ParMETIS, SCOTCH, and ParHIP.

As mentioned before, SeisSol can be built using CMake. Several configuration options are available to customize the build process, some of which are listed below:

- `CMAKE_BUILD_TYPE`: Release (default), RelWithDebInfo, or Debug.
- `EQUATIONS`: elastic (default), viscoelastic2, anisotropic, or poroelastic.
- `ORDER`: the expected convergence order.
- `PRECISION`: single or double.
- `HOST_ARCH`: target CPU architecture (e.g., x86_64, arm64, riscv64).
- `NUMA_AWARE_PINNING`: Enable or disable NUMA-aware thread pinning (default: ON).
- `GEMM_TOOLS_LIST`: list of code generators to be used.

Additionally, SeisSol provides support for Score-P profiling and tracing. In this study, however, Extrae is used to instrument the code and generate Paraver traces. These traces are used to visualize execution behavior and obtain detailed performance data for post-mortem analysis.

After compiling SeisSol, two executables are generated: the main application (`SeisSon_bin`) and the mini benchmark (`SeisSol_proxy`).

SeisSol provides several environment variables to control its execution and performance. Some of the most relevant ones are listed below:

- `SEISSOL_COMMTHREAD`: By default, any SeisSol run with more than one MPI rank will use a communication thread to advance the MPI progress engine.
- `SEISSOL_MINISEISSOL`: When running with multiple ranks, SeisSol will estimate the node-level performance, enabling better load balancing. For that, it runs the so-called Mini SeisSol benchmark stimating the performance of all nodes relative to each other.
- `SEISSOL_MPI_PERSISTENT`: SeisSol employs a static communication pattern, meaning that the same MPI transfer requests are issued across time steps. Consequently, persistent MPI communication is used to reduce communication latency. Persistent communication can be disabled by setting `SEISSOL_MPI_PERSISTENT=0`, in which case SeisSol falls back to standard `MPI_Isend` and `MPI_Irecv` calls.

The problem size and simulation parameters are specified through input files. The mesh file describes the computational domain and is provided in PUML format. The number of cells, the polynomial order, and the number of time steps determine the overall problem size. SeisSol supports both global and local time-stepping schemes. Local time-stepping is generally recommended for simulations because it can lead to a faster solution time. When using local time-stepping, SeisSol only updates elements as needed. To do this, SeisSol computes a time step size for each element independently. The elements are then grouped into so-called time-clusters, which are updated together. Other input files include configuration files for simulation parameters, material properties, and source definitions. SeisSol outputs simulation results in HDF5 format, which can be post-processed and visualized using various tools.

Chapter 3

Related Work

This chapter reviews related work in the areas of performance analysis of HPC applications using the POP methodology and tools, as well as studies involving RISC-V architectures and HPC applications. It covers the following sections:

- Similar studies of HPC scientific applications;
- An overview of CoEs in HPC, with a focus on the POP and ChEESE CoEs; and
- Development platforms for RISC-V hardware/software co-design.

Several studies have investigated the performance analysis and optimization of HPC scientific applications using the POP methodology and tools. Clascà et al. [21] analyzed a multiscale agent-based cellular simulation on a General Purpose Processor (GPP) block of MareNostrum4. Garcia-Gasulla et al. [29] studied the performance of three CFD codes on the same system, while Banchelli et al. [4] evaluated four parallel scientific applications on an Arm-based HPC cluster; in a separate study, the performance of a CFD application on MareNostrum4 was investigated [1].

Recent work has also explored co-design studies involving RISC-V architectures and HPC applications. Mantovani et al. [42] introduced Software Development Vehicles (SDVs) for co-designing HPC systems, comprising a RISC-V based cluster, an LLVM-based compiler toolchain developed at BSC, a software emulator for the RISC-V Vector Extension (rvv), an FPGA-based prototype, and performance analysis tools. Gupta et al. [32] implemented a vectorized and optimized version of the Winograd algorithm, commonly used in Convolutional Neural Network (CNN) models, on RISC-V processors with vector extensions, and performed a co-design study to evaluate performance sensitivity to rvv hardware parameters using the gem5 simulator.

Other studies have focused on evaluating the performance of HPC applications, including SeisSol, on RISC-V architectures to assess the potential of long vector processing units. Banchelli et al. analyzed SeisSol using different GEMM libraries and developed a batched DGEMM library in C to exploit RISC-V vector registers for improved efficiency while maintaining portability. Their work was conducted using both a software emulator and the EPAC hardware prototype [2], [3], [5].

Similar studies in previous MHPC program editions have investigated the performance and energy efficiency of HPC applications on emerging architectures, such as Arm-based systems, RISC-V processors, and FPGA-based platforms. Gachomba [28] carried out an examination of the potential of RISC-V architectures for HPC workloads on the HCA cluster at BSC, covering both intra-node and inter-node scalability aspects using a set of representative HPC benchmarks. Pacheco [49] evaluated the performance of object detection Machine Learning (ML) models on Intel and AMD CPUs, Nvidia GPUs, and suggested other architectures such as RISC-V and Arm-based processors as future work. Tuteja [67] performed a comprehensive analysis of the power consumption and performance characteristics of high energy physics applications on a server equipped with an Intel Platinum 8362 CPU running at 2.8 GHz with 64 cores and an Nvidia L4 GPU under various configurations, delving into different CPU vectorization strategies, such as no vectorization, SSE4, and AVX2. Barnaba [13], as a contribution to the RED-SEA project [14], explored in a co-design approach the performance of network interconnects for scientific HPC applications on the Dibona cluster, an Arm-based High Performance Computing system developed within the European project Mont-Blanc 3 [41]. Gorlani [31] evaluated the feasibility of using FPGA-based computer system in HPC developed within the ExaNeSt project [40]. Baricevic [12] conducted a performance and energy efficiency analysis of a series of HPC workloads on a system from the same family as the Eurotech Eurora cluster [17], which was hosted at CINECA.

3.1 Centers of Excellence in HPC

The Centers of Excellence (CoEs) are specialised projects under the EuroHPC JU framework dedicated to advancing HPC across key scientific and industrial domains in Europe. The CoEs aim to develop and optimise HPC applications, ensuring they meet the highest performance and efficiency standards. They also engage in co-design activities with other EuroHPC JU projects to align application requirements with emerging hardware and software technologies [35].

3.1.1 The POP CoE

The development of performance analysis tools in Barcelona started in 1991 within a research group at the Universitat Politècnica de Catalunya (UPC). After BSC's formation the Performance Tools Group became one of the main research areas within UPC's Computer Science Department. From the very beginning BSC promoted designs and methodologies with flexibility, simplicity and the ability to interact with qualitative and quantitative information at the core of its tools. These features allow performance analysis experts to use the same tools whether using novel homogeneous and heterogeneous multi-core architectures or using more traditional highly scalable cluster systems. It was BSC's tools, expertise, and the knowledge

acquired over the years on performance analysis for HPC systems that motivated BSC to initiate and coordinate the POP service oriented project [54].

The POP Center of Excellence (CoE) gathers leading experts in performance analysis and programming models to offer services to the academic and industrial communities to help them better understand the behaviour of their applications, suggest the most productive directions for optimizing the performance of the codes, and help implementing those transformations in the most productive way.

Since its first phase in 2015, the Center of Excellence on Performance Optimisation and Productivity assists a broad community of HPC application developers and users in both science and industry domains helping them to understand the performance-related issues of their applications and thus improve their efficiency and productivity. This fundamental purpose is reached by externally and objectively auditing the performance of the codes of all interested users, by providing not only qualitative but also quantitative analysis through the use of POP tools and methodology.

The POP services mainly focus on performance assessments with the goal to evaluate code performance and scaling, identifying the main sources of inefficiencies, and providing some insights and recommendations for performance improvements. Second level services may follow after conclusion of an initial performance assessment, such as proof of concept, correctness check, energy efficiency study, and advisory studies.

The current phase, POP3, is articulated in three main pillars: services, users, and co-design. While POP3 will continue targeting all scales and types of users, it focuses on much larger scales, assessing the execution on the EuroHPC hosting sites of some of the European flagship HPC applications for other CoEs [37], [56], [70].

3.1.2 The ChEESE CoE

The Centre of Excellence for Exascale in Solid Earth (ChEESE) intends to anchor itself as a central hub for HPC software within the solid earth community to evaluate and forecast geohazards. In the project's first phase, ChEESE's results had direct applications, such as operational forecast products. In its second phase, ChEESE is refining exascale computing to lessen the impact of natural disaster risks, and to evaluate and forecast geohazards, such as earthquakes, tsunamis, and volcanic eruptions. To achieve this, ChEESE is developing 10 community flagship codes, SeisSol being one of them, to address 12 domain-specific exascale computational challenges. Codes will be optimized in terms of performance on different types of accelerators, scalability, deployment, containerization, and portability. Codes and workflows will combine to form a new generation of 9 pilot demonstrators underpinned by concepts like multi-scale, multi-source, and multi-physics, that will materialise in 15 simulation cases representing capability and capacity use cases of particular relevance in terms of science, social relevance, or urgency. [19], [36].

HPC applications are complex software consisting of millions of lines of code and multiple programming languages, as well as third-party library dependencies and other complexities. For this reason, HPC system co-design is carried out with the aid of application proxies, which are scaled-down versions of target applications that require fewer resources to run while retaining the fundamental behavior of the full codes [43]. ChEESE activities also include co-design activities, which mainly aim to select and prepare a set of applications based on the project's flagship HPC codes. These mini-apps allow for quick prototyping and verification of various solutions. This makes them ideal vehicles for experimentation within the EPI, EUPEX, and EUPILOT projects [43]. The mini-app associated with SeisSol, called `SeisSol-proxy`, runs the computational kernels of the full code using random data to test their performance. It includes built-in performance metrics and is available as compilation target inside the full SeisSol repository [64].

3.2 EU RISC-V Ecosystem

In the European context, the EU has established an ecosystem dedicated to the design and manufacture of its own chips. For this purpose, they elected RISC-V as it is an open-source ISA. The EPI is the foundational element of this initiative, and several projects funded by the EuroHPC JU complement it. These projects share the objective of advancing HPC platform design through a hardware–software co-design approach. Co-design encompasses the entire computing stack, leveraging application insights on computation and data movement to guide hardware and software optimization. This process enables early adaptation of applications to emerging architectures, improving performance and reducing future porting efforts once the platforms reach production maturity.

3.2.1 European Processor Initiative

The EPI project, funded under the EuroHPC JU framework, is a European initiative that aims to design and implement a roadmap for a new family of low-power European processors for extreme scale computing, high-performance Big-Data and a range of emerging applications. The EPI project is structured into two main streams:

- **GPP stream:** The objective of this stream is to move forward with the development of an Arm-based GPP named RHEA.
- **Accelerator stream:** This stream focuses on designing and developing a heterogeneous accelerator named EPAC, which is based on RISC-V ISA and integrates RISC-V vector extensions, specialized blocks for deep learning and stencil computations, and variable precision arithmetic units.

Other projects under the EuroHPC JU program include the EUPEX, the EUPILOT, and the MEEP, all of which are closely related to the EPI project. These projects focus on different aspects of the HPC ecosystem, which aim to develop, integrate, test and co-design a complete hardware and software stack around the EPI processors, including system software, compilers, libraries, and applications. Additionally, the EPI project collaborates with other CoEs in HPC, such as ChEESE and POP, to ensure that the developed processors meet the needs of the European scientific and industrial communities.

3.2.2 European Platform for Exascale

EUPEX aims to develop and demonstrate the first European platform for HPC, gathering and integrating European technologies from system architecture, processor, system software and development tools all the way to applications [18].

The EUPEX system is a modular and production-grade platform designed to demonstrate the viability of the Modular Supercomputing Architecture. It integrates diverse hardware modules, including emerging architectures. The system is large enough to serve as a proof of concept for modular architectures based on European technologies, particularly the European Processor Initiative (EPI), and to showcase the Exascale readiness of co-designed applications.

In addition to the hardware platform, EUPEX also focuses on developing a comprehensive software stack addresing four main areas: management and administration, execution environment, performance tools, and storage architecture. The software stack is designed to be modular and adaptable, allowing it to support a wide range of applications and workloads.

In the context of applications and co-design, EUPEX aims to identify application requirements, optimize them for the Arm SVE architectures, adapt and benchmark them on the modular pilot system, and derive recommendations to guide application development for future Exascale platforms.

3.2.3 European Pilot using Independent, Local and Open Technologies

The European Pilot using Independent, Local and Open Technologies (EUPILOT) project has been developed to demonstrate an autonomous set of European accelerators that incorporates technologies from EPAC and other European Intellectual Properties (IPs). Built on the RISC-V ISA, the project integrates open accelerator designs with representative HPC and high performance data analytics (HPDA) applications. The EUPILOT project targets the development of three chip tapeouts, including a 12 nm test chip and two accelerators: a 16-core vector accelerator and an eight-core machine learning and stencil accelerator. A complete software stack including applications, runtimes, libraries, compilers, and tools, will enable these architectures to support both traditional HPC and Artificial Intelligence (AI) workloads. By integrating hardware accelerators with optimized programming models and frameworks such as OpenMP,

MPI, and others, EUPILOT aims to enhance programmability and performance scalability toward future Exascale systems [30].

3.2.4 MareNostrum Experimental Exascale Platform

The MareNostrum Experimental Exascale Platform (MEEP) is a flexible FPGA-based emulation platform that explores hardware and software co-designs for Exascale Supercomputers and other hardware targets based on European-developed pre-silicon IPs.

The full stack platform layers include HPC applications, a software toolchain, emulation platforms, and RISC-V-based accelerator prototypes, known as Accelerated Compute and Memory Engines (ACME). The novelty of the accelerator is that it is self-hosting, meaning the entire application resides in the accelerator's memory and executes entirely on the accelerator.

The ACME accelerator consists of two main components: the computational engine and the memory engine. Both are responsible for improving the accelerator's performance in different contexts because they are designed for specific tasks. The computational engine operates with data, including scalar and vector elements. The memory engine, on the other hand, is responsible for memory transactions and accesses to reduce energy usage and save memory.

MEEP's objectives include leveraging and extending projects like EPI and the POP CoEs, improving the RISC-V software ecosystem with an improved and extended software tool chain and suite of HPC applications, and delivering a series of Open-Source IPs for academics, traditional and emerging HPC applications [65].

3.2.5 Barcelona Zettascale Laboratory

The Barcelona Zettascale Lab (BZL) project, led by BSC, focuses on developing high performance chip prototypes based on the RISC-V architecture for integration into future supercomputing systems. It also aims to build a compatible software ecosystem that ensures optimal performance and interoperability with HPC workloads. Through collaboration with strategic partners and initiatives such as the European Processor Initiative (EPI), the project contributes to advancing next generation, energy efficient HPC technologies. BZL's chip designs will integrate multiple RISC-V processors, a Vector Processing Unit (VPU), and a memory hierarchy tailored to the needs of the HPC domain. The project also emphasizes the development of a robust software ecosystem designed to support and optimize the performance of HPC applications on RISC-V architectures. This includes compilers, libraries, and performance analysis tools [57].

Chapter 4

Methodology

In this study, the performance of SeisSol is evaluated on multiple RISC-V architectures available in the HCA cluster at BSC. This chapter presents the methodology and experimental setup used for the performance analysis. It covers the following key aspects:

- the benchmark setup, which defines the execution environment and problem parameters;
- the scalability analysis, conducted under different parallel configurations;
- the tracing-based performance investigation; and
- the evaluation of results using the POP performance metrics framework.

4.1 Benchmark Setup

SeisSol is part of the Southern California Earthquake Center (SCEC)/United States Geological Survey (USGS) Spontaneous Rupture Code Verification Project [33], an international collaboration to verify and validate computer codes used to simulate earthquakes as spontaneous dynamic ruptures. Several SCEC benchmarks are available [63] for testing and validating the code. In this study, the focus is placed on the computational performance of SeisSol rather than its physical accuracy. Therefore, a benchmark that is computationally intensive and representative was selected for performance analysis on the target RISC-V platform.

4.1.1 Build and Input Configuration

A suitable SCEC benchmark was chosen for the analysis. The input parameters for this benchmark, such as mesh size, simulation time, and other relevant physical and numerical settings, were defined to ensure a representative workload. The configuration was designed so that the sequential execution time ranged between 20 and 30 minutes on the target platform, providing a balanced runtime for detailed performance investigation.

In addition to the benchmark input parameters, the build configuration of SeisSol was also specified.

4.2 Scalability Analysis

For the scalability analysis, intra-node and inter-node strong scaling experiments were conducted on each hardware platform.

In the intra-node tests, three configurations were evaluated:

- **MPI-only:** SeisSol was executed with varying numbers of MPI ranks within a single node. Each rank was pinned to a specific core to minimize context switching and ensure consistent performance.
- **OpenMP-only:** SeisSol was executed using different numbers of OpenMP threads within a single node, with each thread pinned to a dedicated core.
- **Hybrid MPI+OpenMP:** All available computational resources in a single node were utilized. SeisSol was executed using a combination of MPI processes and OpenMP threads, varying their proportions to explore different configurations and identify the optimal balance between the two parallel programming models.

For the inter-node experiments, a hybrid MPI+OpenMP configuration was employed to evaluate the scalability of SeisSol across multiple nodes.

In all cases, the Floating Point Operations Per Second (FLOPS) reported by SeisSol's output were used as the primary performance metric to assess computational efficiency.

4.3 Tracing Analysis

For the tracing analysis, the following steps were performed:

- **Tracing:** SeisSol was executed with Extrae instrumentation using a relevant input and core configuration to generate detailed event traces.
- **Selection of the Focus of Analysis (FoA):** The resulting trace files were analyzed using Paraver to identify de FoA for further investigation.
- **POP Metrics Collection:** Based on the identified FoA, POP performance metrics were collected and evaluated to assess parallel efficiency and scalability.
- **Granularity Analysis:** On selected boards, the granularity of the FoA was analyzed to understand the balance between computation and communication.

Chapter 5

Performance Analysis I

Chapter 5 presents the performance analysis of SeisSol, a high-performance computational seismology software, on multiple RISC-V architectures available under the HCA infrastructure at BSC following the methodology described in chapter 4. SeisSol is an ideal case study for evaluating RISC-V systems due to its excellent performance and scalability, demonstrated on a suitable range of supercomputing platforms [25], [34], and winning several HPC awards [59], [60].

This chapter starts by describing the hardware platform, software environment, and benchmark setup used for the performance analysis. Next, section 5.1 presents an examination of SeisSol's execution structure to identify the Focus of Analysis (FoA) for further investigation. Following that, section 5.2 and section 5.3 present the intra-node and inter-node performance analyses, respectively, where scalability and tracing-based investigations are conducted. Finally, section 5.4 summarizes the key findings and insights from the performance analysis.

The HCA [9] infrastructure at BSC is a small data center composed of multiple platforms based on RISC-V and x86_64 architectures. For this thesis, the performance analysis focuses on three RISC-V partitions with four nodes each: Milk-V Pioneer, Banana Pi BPI-F3, and SiFive Premier. These platforms are selected due to their distinct architectural features, which provide a comprehensive overview of SeisSol's performance across different RISC-V implementations. The internal network that connects the nodes within each partition uses 1 Gigabit Ethernet.

The Milk-V Pioneer [45] is a development board based on the SOPHON SG2042 chip, featuring a 64-core RISC-V CPU with rvv 0.7 support using short vector registers (128 bit). It operates at frequencies up to 2 GHz and is equipped with 128 GB of DDR4 memory across 4 Non-Uniform Memory Access (NUMA) nodes. The latency cost between NUMA nodes is detailed in table 5.1.

Table 5.1: Milk-V Pioneer NUMA node distances.

Node	0	1	2	3
0	10	15	25	30
1	15	10	30	25
2	25	30	10	15
3	30	25	15	10

The values in the table represent relative memory latency between NUMA nodes. A distance of 10 indicates local memory access, while higher values indicate increased latency for remote memory access. For example, accessing memory from Node 0 to Node 1 has a latency cost of 15, indicating 1.5× more latency compared to local access.

The Banana Pi BPI-F3 [51] is a development board powered by SpacemiT K1 octa-core RISC-V chip with rvv 1.0 support (128 bit vector registers). It features 16 GB LPDDR4 memory and operates at frequencies up to 1.6 GHz.

The SiFive Premier P550 [62] is a high-performance development board featuring quad-core SiFive P550 RISC-V CPU (no rvv support) running at 1.4 GHz. It is equipped with 32 GB of LPDDR5 memory.

The Operating System running across all RISC-V platforms is Ubuntu LTS. Milk-V Pioneer runs Ubuntu 22.04.4 LTS (GNU/Linux 6.1.80 riscv64), Banana Pi BPI-F3 runs Armbian Noble built on Ubuntu 24.04 LTS (GNU/Linux 6.6.63 riscv64), and SiFive Premier runs Ubuntu 24.04.2 LTS (GNU/Linux 6.6.77 riscv64).

The software environment for building and running SeisSol is consistent across all RISC-V platforms. The compiler toolchain is based on LLVM, specifically, a custom version of the Clang compiler (v22.0.0) developed at BSC as part of the EPI project which provides support for RISC-V. MPI v4.1.6 from OpenMPI is used for distributed memory parallelism, and OpenMP v5.1 is employed for shared memory parallelism. Eigen is used as GEMM generator for CPU to create optimized code for small matrix-matrix multiplications. SeisSol utilizes a mesh partitioning library during the startup of the simulation for a good load balance. Here, the mesh partitioning library is ParMETIS v4.0.3. CMake v3.28.1 is utilized along with Ninja v1.10.1 as the build system. The following CMake options are set for the build configuration:

- `EQUATIONS=elastic`
- `ORDER=4`
- `PRECISION=double`
- `HOST_ARCH=rvv128`
- `NUMA_AWARE_PINNING=ON`
- `GEMM_TOOLS_LIST=eigen`
- `ASAGI=ON`

The execution structure of SeisSol is analyzed with Paraver to identify the different phases of the application and select the FoA for further investigation.

The first performance property to analyze is the application's scalability, which is determined by comparing different program runs with increasing computational resources. If the performance increase is less than linear with respect to the increase in resources, it is an initial sign of potential inefficiencies in the application or underlying system. In scalability analyses, the performance metric used for relative speedup and efficiency is the hardware FLOP/s, in double precision, reported by SeisSol's output. This metric measures the number of floating-point operations per seconds executed by the hardware in compute kernels. It depends on the underlying hardware, scenario, and equation system to be solved [61]. It is important to note that this metric does not account for data movement operations, which can also impact overall performance.

To better understand the performance behavior of SeisSol, a tracing-based analysis is conducted. To trace the execution of SeisSol, Extrae v4.3.3 based on the interposition mechanism is used. This interposition is done by the runtime loader by substituting the original symbols found in the application binary or shared libraries through `LD_PRELOAD` by those provided by the instrumentation package. Hardware performance counters are collected using PAPI v7.0.1.

The TPV5 SCEC dynamic simulation example is selected as the benchmark for performance analysis. To ensure that the serial execution time falls within the range of 20 to 30 minutes on the target platform, the mesh geometry is scaled down to 10 % of its original size for intra-node experiments. The original TPV5 mesh is used for inter-node experiments. The simulation time is set to 1.0 s in the parameter file for both intra-node and inter-node tests. Writing output files is disabled to avoid I/O overhead affecting the performance measurements. For this analysis, SeisSol is run without a dedicated communication thread, the Mini SeisSol benchmark enabled for better load balancing, and persistent MPI operations enabled.

5.1 Execution Structure

The execution of SeisSol is divided into initialization, simulation, and finalization phases. During the initialization phase, the mesh and model are set up, and various components of the simulation are initialized. The simulation phase, corresponding to the `Simulator::simulate` code region located in `src/Solver/Simulator.cpp`, is the main computational phase of the application, where the time-stepping loop is executed. The finalization phase involves writing output data and cleaning up resources.

Figure 5.1 shows different Paraver timeline visualizations of a SeisSol's execution on one node of Milk-V Pioneer using a hybrid 4×16 configuration (only 4 OpenMP threads per MPI rank are shown for clarity). Similar execution structures are observed across other RISC-V platforms and parallel configurations. Figure 5.1a illustrates the simulation phase (`Simulator::simulate` code region), which is the main computational phase of the application. Only the original

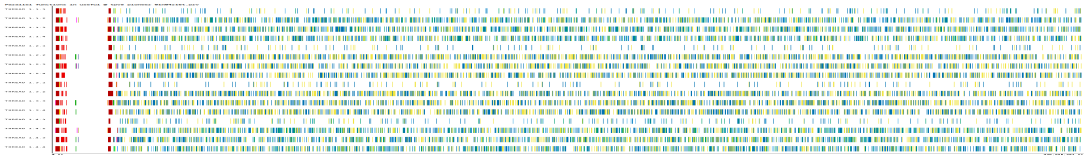
threads that fork multiple parallel OpenMP regions during execution are shown with a color code. The initialization and finalization phases are easily identifiable, even though they are not shown in color. They occur before and after the simulation phase, respectively. Figure 5.1b illustrates the useful duration of the execution, which represents the time spent performing actual computations, excluding communication and synchronization overheads. In this timeline, it is possible to observe the alternating pattern of computation and communication phases. The MPI calls are shown in Figure 5.1c, while the OpenMP parallel functions executed during the useful duration are depicted in Figure 5.1d.

At the start of the execution, there is a brief period of parallel computation with both MPI calls and OpenMP parallel regions when the mini SeisSol benchmark is executed to determine the node weights for load balancing. Then, several MPI calls are made without spawning any OpenMP regions. This corresponds to the time when the mesh is read and partitioned, the model is initialized, the initial conditions are set, and the simulation parameters are configured. After this, the simulation enters the main time-stepping loop, where both MPI and OpenMP parallelism are used extensively.

(a) Simulation phase (`Simulator::simulate` code region)

(b) Useful Duration

(c) MPI Calls

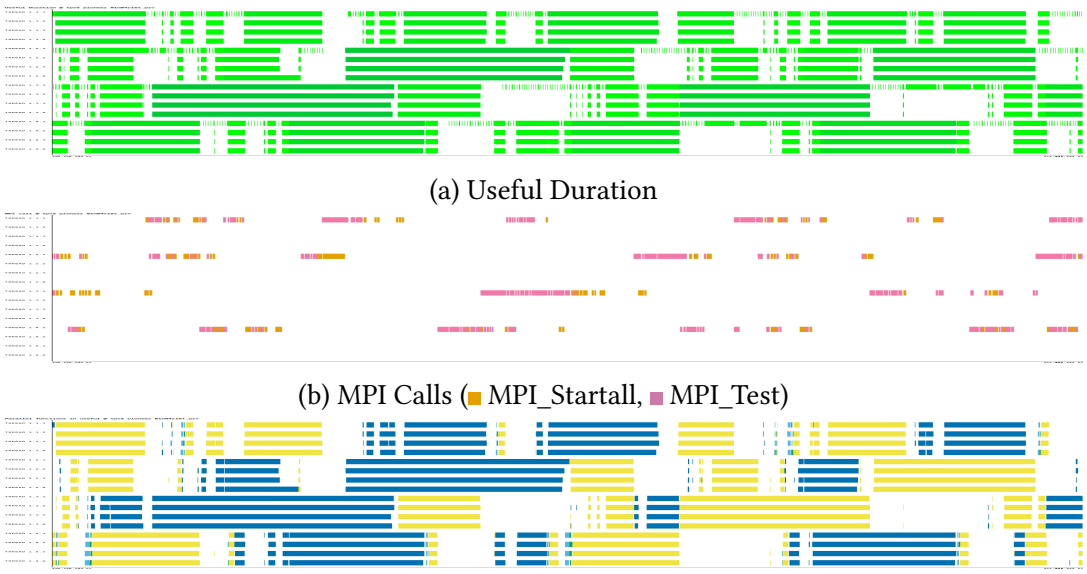


(d) OpenMP Parallel Functions in Useful Duration

Figure 5.1: Paraver timeline visualization of SeisSol execution structure on Milk-V Pioneer using hybrid MPI and OpenMP configuration (4 MPI ranks and 4 OpenMP threads per rank are shown).

During the simulation phase, several key computational kernels are executed for advancing the solution in time. Figure 5.2 provides a zoomed-in view of the execution structure, high-

lighting the detailed behavior of the useful duration, MPI calls, and OpenMP parallel functions during the simulation phase.



(a) Useful Duration

(b) MPI Calls (■ MPI_Startall, ■ MPI_Test)

(c) OpenMP Parallel Functions in Useful Duration (■ compute local integration, ■ compute neighboring integration, ■ compute dynamic rupture)

Figure 5.2: Zoomed Paraver timeline visualization of SeisSol execution structure on Milk-V Pioneer using hybrid MPI and OpenMP configuration (4 MPI ranks and 4 OpenMP threads per rank are shown).

For this particular execution, as can be seen in fig. 5.2a, most of the time is spent on useful computation. Further information on this and other executions with different parallel configurations can be found in table 5.2.

Table 5.2: Average time percentage spent in useful computation and MPI calls across different parallel configurations during the simulation phase on Milk-V Pioneer.

Configuration	Useful Computation (%)	MPI_Test (%)	MPI_Startall (%)
01r64t	100.00	0.00	0.00
02r32t	85.33	7.35	7.32
04r16t	76.28	16.38	7.34
08r08t	73.41	20.83	5.76
16r04t	72.85	22.46	4.69
32r02t	70.64	24.95	4.42
64r01t	63.80	31.23	4.96

As mentioned earlier, persistent MPI operations are enabled in this execution. This is reflected in fig. 5.2b, where `MPI_Startall` and `MPI_Test` are used for communication instead of `MPI_Isend`/`MPI_Irecv` and `MPI_Wait`. Another notable observation in the communication pattern is the absence of synchronization barriers between MPI ranks. This is because persistent MPI allows each rank to progress independently without waiting for others to reach the same

point in execution. It may be interesting in future work to compare the execution structure when persistent MPI is disabled to observe any differences in communication patterns and their impact on performance.

Figure 5.2c shows the OpenMP parallel functions executed during the useful duration. In this execution, the majority of the computation time is spent in the `compute local integration` and `compute neighboring integration` kernels, with a smaller portion allocated to `compute dynamic rupture` (see table 5.3 for more details).

Table 5.3: Average time percentage spent in OpenMP parallel functions during the useful duration across different parallel configurations on one node of Milk-V Pioneer.

Configuration	Outside OMP region (%)	c.local_int. (%)	c.neighboring_int. (%)
01r64t	7.45	47.83	43.30
02r32t	27.03	37.91	33.65
04r16t	32.94	36.42	29.70
08r08t	42.67	31.93	24.59
16r04t	45.70	31.15	22.38
32r02t	57.14	24.90	17.35
64r01t	62.76	21.77	14.95

It is interesting to note that the reported statistics are consistent: increased time outside parallel regions correlates with a higher proportion of MPI calls, suggesting that this overhead is mainly due to MPI execution.

5.1.1 Focus of Analysis

The identified simulation phase (`Simulator::simulate` code region) is selected as the FoA for further performance analysis in the subsequent sections for several reasons. First, it encompasses the core computational workload of SeisSol, where the majority of the execution time is spent. Second, it involves significant use of both MPI and OpenMP parallelism, making it an ideal candidate for investigating the performance characteristics and scalability of the application on RISC-V platforms. And third, selecting a few representative time-step iterations within the simulation phase is not feasible due to the asynchronous nature of the execution across MPI ranks when using persistent MPI.

5.2 Intra Node Analysis

The intra-node performance analysis evaluates the scalability and performance of SeisSol on a single node of each RISC-V platform under different parallel configurations: MPI-only, OpenMP-only, and hybrid MPI+OpenMP. The analysis starts with the scalability evaluation, followed by a tracing-based investigation of the FoA identified in section 5.1. The mesh size

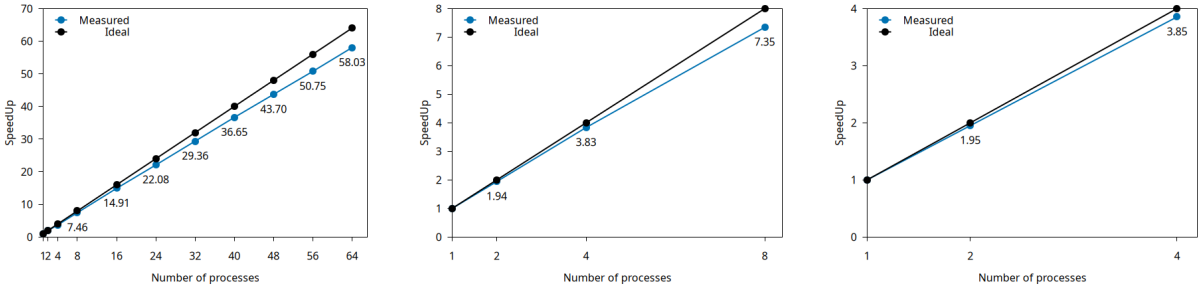
used for intra-node experiments is composed of 183 125 cells and 32 341 vertices. It is a scaled-down version of the original TPV5 mesh to ensure that the serial execution time falls within the desired range on the target platforms.

5.2.1 Scalability Analysis

The first analysis focuses on the scalability of SeisSol under different parallel configurations on each RISC-V platform. The performance metric used for this analysis is the hardware FLOP/s reported by SeisSol's output. This metric measures the number of floating-point operations per second executed by the hardware in compute kernels.

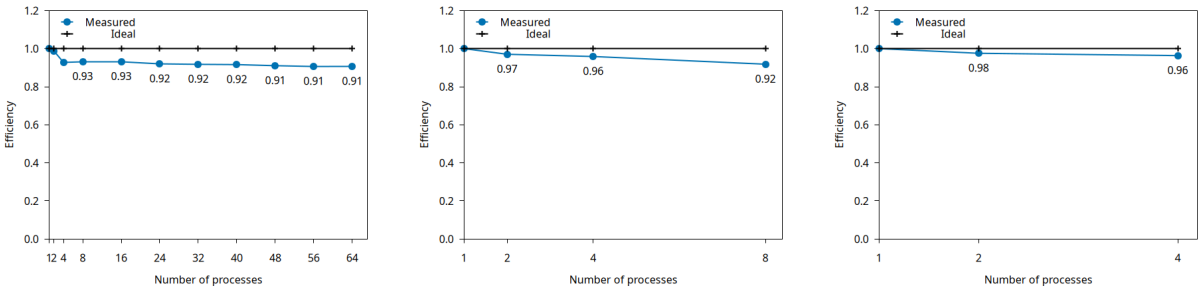
5.2.1.1 MPI-only Analysis

The scalability results of the MPI-only configuration are presented in fig. 5.3 and fig. 5.4. Each measurement represents the average performance over five runs with a relative standard deviation below 2 %. Milk-V Pioneer demonstrates excellent and stable scaling across the entire range of tested ranks, with minor fluctuations. The efficiency remains above 90 % up to 64 ranks, and there is no visible saturation point before the last configuration. The mesh is large enough that MPI domain decomposition keeps work per rank balanced, and communication overhead is minimal compared to computation. Banana Pi BPI-F3 shows excellent near-linear scaling across the entire range of tested ranks, indicating stable communication and balanced workload distribution. From 1 to 8 ranks, the speedup is 7.3×, which is highly efficient. The trend is similar for SiFive Premier, which also exhibits near-ideal scaling up to 4 ranks.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.3: MPI speedup curves for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.4: MPI efficiency curves for SeisSol on HCA RISC-V platforms.

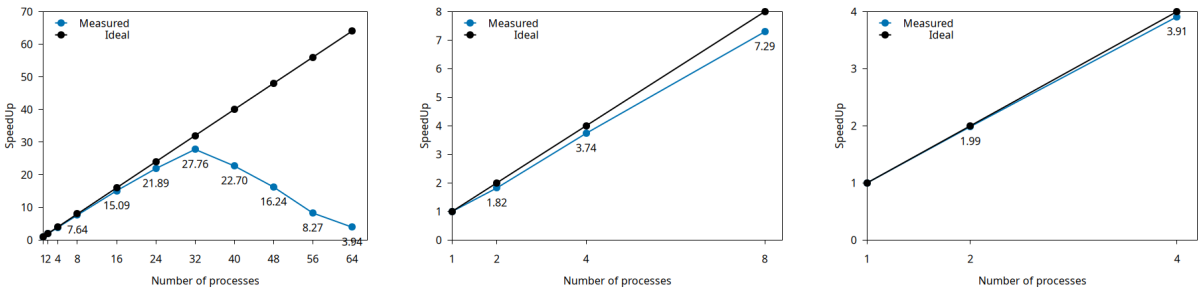
5.2.1.2 OpenMP-only Analysis

The scalability results of the OpenMP-only configuration are presented in fig. 5.5 and fig. 5.6.

Milk-V Pioneer exhibits excellent scaling up to 32 threads, with performance nearly doubling each time until that point. However, as the number of threads exceeds 32, the shared-memory parallelism saturates, and there is a substantial decline in performance, accompanied by a rapid decrease in throughput as the thread count increases further. So much so that at 64 threads, the performance is comparable to that of 4 threads. The saturation point at 32 threads is likely due to the architecture of Pioneer, which consists of 4 NUMA nodes with 16 cores each. When the thread count exceeds 32, threads start to compete for resources across NUMA nodes, leading to increased memory access latency and contention, which degrades performance. A detailed tracing analysis is provided in the next section to further investigate this behavior.

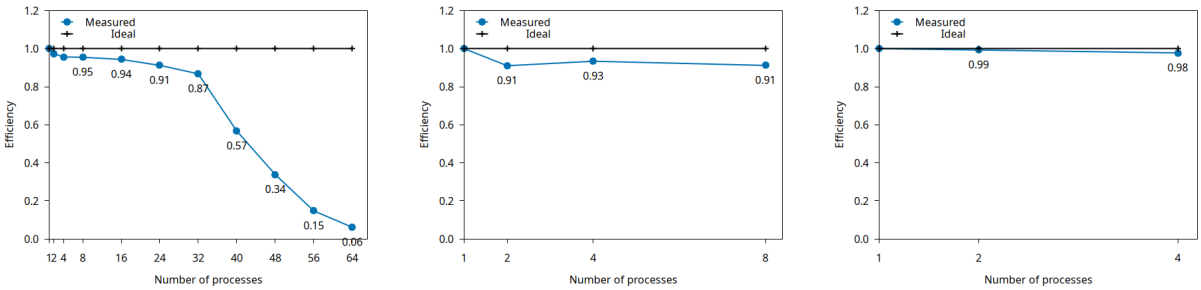
The OpenMP-only scaling for Banana Pi BPI-F3 is almost identical to its MPI-only counterpart, showing excellent scaling across all thread counts.

And SiFive Premier also shows similar behavior to its MPI-only results, with almost perfect scaling up to 4 threads.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.5: OMP speedup curves for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.6: OMP efficiency curves for SeisSol on HCA RISC-V platforms.

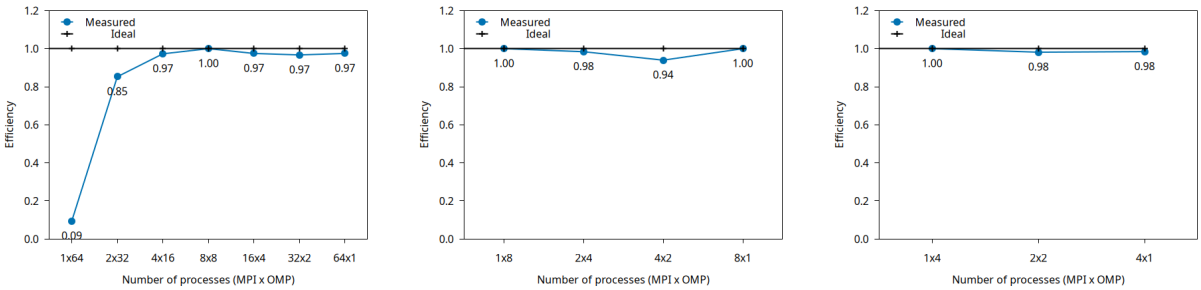
5.2.1.3 Hybrid MPI+OpenMP Analysis

The hybrid MPI+OpenMP performance analysis aims to explore the combined use of both parallel programming models to fully utilize the computational resources of each platform. The results are presented in fig. 5.7. The configurations range from pure OpenMP to pure MPI, with various intermediate hybrid configurations.

For Milk-V Pioneer, the first two hybrid configurations, 1×64 and 2×32, show similar performance degradation as observed in the OpenMP-only configuration, indicating that using too many threads per rank leads to inefficiencies probably due to NUMA effects and resource contention. The performance of the 1×64 configuration is suboptimal, similar to the OpenMP-only result with 64 threads. The 2×32 configuration shows a significant improvement, reaching approximately 85 % of the peak performance observed in the 4×16 configuration. From the 4×16 hybrid configuration to fully 64×1 MPI decomposed configuration, the performance remains relatively stable with no significant differences. The optimal configuration appears to be between 4×16 and 8×8. These configurations reduce the extra overhead from OpenMP and make the most of the available cores in each NUMA region.

Banana Pi BPI-F3 configurations range from 1 rank with 8 threads to 8 ranks with 1 thread. The performance remains roughly the same across all configurations, and no clear gain or loss is observed when switching between pure and hybrid modes.

Similarly, SiFive Premier configurations range from 1 rank with 4 threads to 4 ranks with 1 thread. The performance is relatively stable across all configurations, with no significant differences between pure and hybrid modes.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.7: Hybrid MPI and OpenMP relative efficiency for SeisSol on HCA RISC-V platforms.

5.2.2 Tracing Analysis

To complement the scalability analysis, a tracing-based performance investigation is conducted using the POP performance metrics framework. The focus of the analysis is on the `Simulator::simulate` code region of SeisSol, identified as the FoA in the execution structure. The efficiency tables for SeisSol on Milk-V Pioneer, Banana Pi BPI-F3, and SiFive Premier platforms are presented in figs. 5.8, 5.11 and 5.15 for MPI-only, OpenMP-only, and hybrid MPI+OpenMP configurations, respectively.

5.2.2.1 MPI-only Tracing Analysis

For the MPI-only configuration, Milk-V Pioneer shows good global efficiency above 80 % up to 8 threads. After that, the global efficiency collapses rapidly. At 16 ranks, the global efficiency is 66 %, then it drops to 53 % at 32 ranks, and finally to 34 % at 64 ranks. It is worth noting that the entire observed efficiency loss in Pioneer comes from Extrae's impact on per-rank computation, rather than MPI scaling or hardware limitations. To support this, in fig. 5.9, the efficiency curves derived from tracing analysis (with Extrae) closely match those obtained from the scalability analysis (without Extrae) at lower rank counts, and only start to diverge at higher rank counts where Extrae's overhead becomes more pronounced. Continuing with the tracing analysis, the parallel efficiency remains relatively high, above 90 %, up to 16 ranks, but then drops to 83 % at 32 ranks and further down to 65 % at 64 ranks. The load balance efficiency degrades steadily from 99 % at 8 ranks to 74 % at 64 ranks. And the communication efficiency remains high throughout, above 88 % even at 64 ranks. The computation efficiency stays above 89 % up to 8 ranks, but then drops to 73 % at 16 ranks, 64 % at 32 ranks, and finally to 53 % at 64 ranks. This is very similar to the global efficiency collapse. Decomposing the computation efficiency further, the IPC scalability falls steadily from 90 % at 8 ranks to 64 % at 64 ranks; the instruction scalability decreases from 98 % at 8 ranks to 81 % at 64 ranks; and the frecuency scalability remains high, above 98 %, across all rank counts.

Banana Pi BPI-F3 demonstrates excellent global efficiency above 89 % across all tested ranks. The parallel efficiency is almost perfect, above 98 %, up to 8 ranks. The same applies to load balance and communication efficiencies, which remain above 99 % throughout. The

computation efficiency is also very high, above 90 %, across all ranks; the IPC scalability is above 94 %, though the average IPC is below 1.0; the instruction scalability is above 96 %; and the frequency scalability is above 99 %. For this board, the MPI scaling is nearly ideal, and the overhead introduced by Extrae is minimal.

SiFive Premier also exhibits strong global efficiency above 93 % up to 2 ranks, and remains above 83 % up to 4 ranks. The parallel efficiency is high, above 96 %, up to 2 ranks, and drops to 88 % at 4 ranks. This light efficiency drop is mainly due to communication efficiency, which decreases from 96 % at 2 ranks to 90 % at 4 ranks. The load balance efficiency remains above 97 % across all ranks. The computation efficiency is excellent, above 94 % across all ranks; the IPC scalability is above 96 % and the average IPC is the highest among the three platforms, above 2.2; the instruction scalability is above 98 %; and the frequency scalability is above 99 %. Premier shows good MPI scaling with minimal overhead from Extrae.

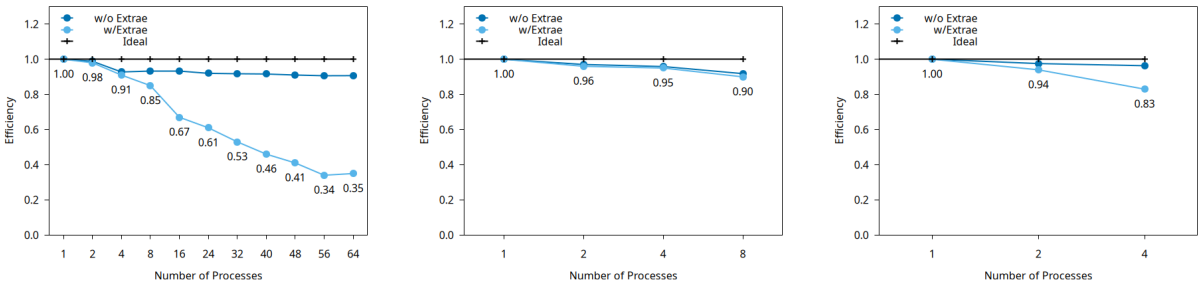
	1	2	4	8	16	24	32	40	48	56	64
Global efficiency	100.00	98.08	91.43	84.89	66.56	60.56	53.49	46.46	40.93	33.88	34.67
-- Parallel efficiency	100.00	99.29	97.97	95.24	90.62	85.99	83.57	77.75	75.03	74.65	65.48
-- Load balance	100.00	100.00	99.53	99.30	94.71	89.48	89.86	84.98	82.14	82.03	74.11
-- Communication efficiency	100.00	99.29	98.43	95.91	95.68	96.09	92.99	91.50	91.34	91.00	88.36
-- Computation scalability	100.00	98.78	93.32	89.13	73.44	70.42	64.01	59.75	54.56	45.38	52.95
-- IPC scalability	100.00	98.83	93.83	90.90	80.44	78.07	73.33	70.48	66.77	60.80	65.82
-- Instruction scalability	100.00	99.93	99.43	98.05	92.01	91.09	88.21	85.63	82.52	75.29	81.26
-- Frequency scalability	100.00	100.03	100.03	100.01	99.23	99.03	98.95	99.01	99.02	99.13	99.00
Average IPC	1.69	1.67	1.58	1.53	1.36	1.32	1.24	1.19	1.13	1.02	1.11
Average frequency (GHz)	2.00	2.00	2.00	2.00	1.98	1.98	1.98	1.98	1.98	1.98	1.98

	1	2	4	8
Global efficiency	100.00	96.50	94.58	89.55
-- Parallel efficiency	100.00	99.78	99.44	98.89
-- Load balance	100.00	99.96	99.93	99.58
-- Communication efficiency	100.00	99.81	99.51	99.30
-- Computation scalability	100.00	96.72	95.11	90.56
-- IPC scalability	100.00	98.50	97.26	94.12
-- Instruction scalability	100.00	98.27	97.74	96.39
-- Frequency scalability	100.00	99.92	100.05	99.82
Average IPC	0.83	0.81	0.80	0.78
Average frequency (GHz)	1.60	1.60	1.60	1.60

	1	2	4
Global efficiency	100.00	93.63	83.45
-- Parallel efficiency	100.00	96.58	88.65
-- Load balance	100.00	99.94	97.65
-- Communication efficiency	100.00	96.64	90.79
-- Computation scalability	100.00	96.95	94.13
-- IPC scalability	100.00	97.85	96.12
-- Instruction scalability	100.00	99.00	98.20
-- Frequency scalability	100.00	100.08	99.72
Average IPC	2.26	2.21	2.17
Average frequency (GHz)	1.40	1.40	1.40

(a) Pioneer (b) BPI-F3 (c) Premier

Figure 5.8: MPI-only efficiency tables for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.9: Comparison of efficiency curves for SeisSol on HCA RISC-V platforms in MPI-only configuration, with and without Extrae instrumentation.

The granularity analysis for Milk-V Pioneer is shown in fig. 5.10. The granularity, or grain size, of a task is a measure of the amount of computation which is performed by that task [38]. Using Paraver, the average instantaneous granularity of useful duration is calculated for each MPI rank across different rank counts. In general, for MPI-only configurations, the granularity in not well defined on a specific duration, but it is possible to observe that task duration decreases as the number of MPI ranks increases.

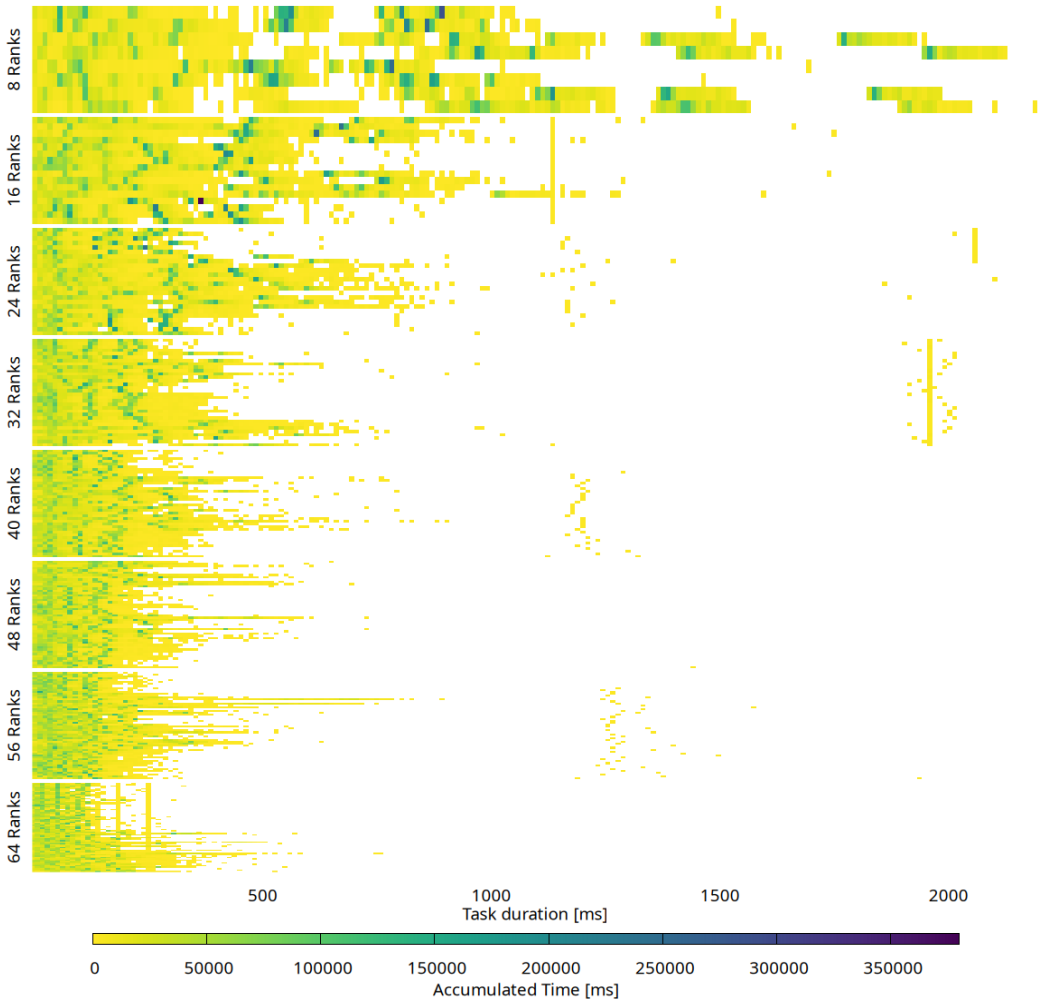


Figure 5.10: MPI-only average instantaneous granularity of useful duration for SeisSol on Milk-V Pioneer.

5.2.2.2 OpenMP-only Tracing Analysis

For the OpenMP-only configuration, Milk-V Pioneer, up to 32 threads, shows good global efficiency above 88 %. Beyond 32 threads, the global efficiency degrades considerably, dropping to 51 % at 48 threads and further down to 23 % at 64 threads. Differently from the MPI-only case, the efficiency loss here is mainly due to OpenMP scaling and hardware limitations rather than Extrae's overhead on computation. In fact, the efficiency curves from tracing analysis (with Extrae) are similar to those from scalability analysis (without Extrae) across all thread counts, as shown in fig. 5.12. Continuing with the tracing analysis, despite the global efficiency drop, the parallel efficiency remains relatively high, above 96 % up to 32 threads, and still between 85 % to 95 % beyond 40 threads. The load balance efficiency is excellent, above 95 % across all thread counts. Also, the communication efficiency remains high, above 84 % throughout. This indicates that the threads are well balanced and syncronization overhead is not dominating the performance loss. However, the computation efficiency experiences a significant decline, dropping from 90 % at 32 threads to 53 % at 48 threads, and further down to

26 % at 64 threads. This is the main contributor to the global efficiency collapse. Decomposing the computation efficiency further, the IPC scalability falls sharply from 90 % at 32 threads to 56 % at 48 threads, and further down to 29 % at 64 threads, indicating a severe pipeline stall; the instruction scalability remains relatively high, above 90 % even at 64 threads, suggesting that the slowdown is due to instructions taking longer to execute rather than a reduction in the number of instructions. This contributes to the hypothesis presented before regarding resource contention and NUMA effects. However, the lack of a bigger set of hardware performance counters limits a deeper analysis in this regard. Finally, the frequency scalability remains high, above 97 % across all thread counts.

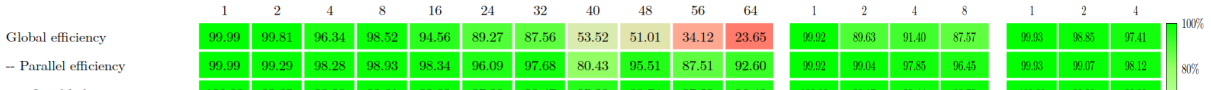
Banana Pi BPI-F3 maintains excellent global efficiency above 87 % across all tested threads. The parallel efficiency is nearly perfect, above 96 %, up to 8 threads. The same applies to load balance and communication efficiencies, which remain above 97 % throughout. The computation efficiency is also very high, above 90 %, across all threads; the IPC scalability is above 93 % and the average IPC is below 1.0; the instruction scalability is above 97 %; and the frequency scalability is above 99 %.

SiFive Premier shows strong global efficiency above 97 % across all tested threads. And in general, all other efficiency metrics remain above 98 %, indicating excellent OpenMP scaling with minimal overhead from Extrae.

	1	2	4	8	16	24	32	40	48	56	64
Global efficiency	99.99	99.81	96.34	98.52	94.56	89.27	87.56	53.52	51.01	34.12	23.65
-- Parallel efficiency	99.99	99.29	98.28	98.93	98.34	96.09	97.68	80.43	95.51	87.51	92.60
-- Load balance	100.00	99.85	98.98	99.61	99.09	97.38	98.47	95.30	98.74	97.29	96.48
-- Communication efficiency	99.99	99.45	99.29	99.31	99.24	98.67	99.20	84.40	96.72	89.95	95.97
-- Computation scalability	100.00	100.52	98.03	99.59	96.16	92.91	89.64	66.54	53.41	38.99	25.54
-- IPC scalability	100.00	100.33	97.71	99.92	97.62	96.29	91.71	81.95	56.50	46.31	28.92
-- Instruction scalability	100.00	100.16	100.30	99.63	98.45	96.57	97.68	84.00	94.97	86.23	90.16
-- Frequency scalability	100.00	100.02	100.03	100.04	100.06	99.91	100.06	96.67	99.54	97.64	97.93
Average IPC	1.65	1.66	1.61	1.65	1.61	1.59	1.52	1.35	0.93	0.77	0.48
Average frequency (GHz)	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.93	1.99	1.95	1.96

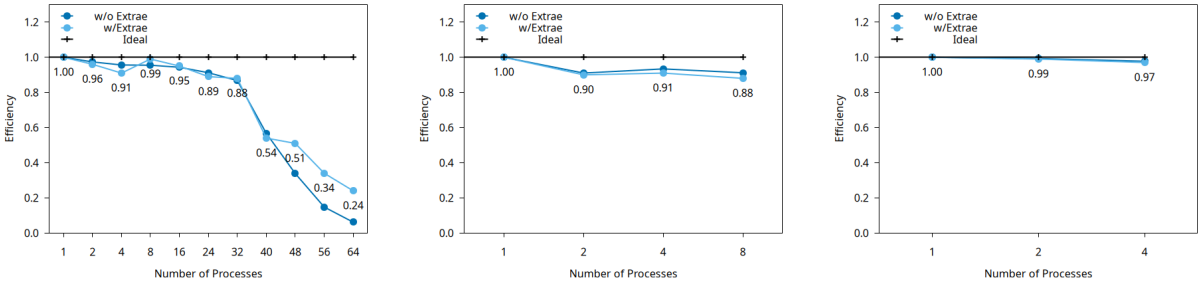
	1	2	4	8
Global efficiency	99.92	89.63	91.40	87.57
-- Parallel efficiency	99.92	99.04	97.85	96.45
-- Load balance	100.00	99.17	98.44	98.72
-- Communication efficiency	99.92	99.87	99.40	97.70
-- Computation scalability	100.00	90.50	93.41	90.79
-- IPC scalability	100.00	90.53	93.28	93.49
-- Instruction scalability	100.00	100.07	100.15	97.34
-- Frequency scalability	100.00	99.89	99.98	99.76
Average IPC	0.83	0.76	0.78	0.78
Average frequency (GHz)	1.60	1.60	1.60	1.60

	1	2	4
Global efficiency	99.93	98.85	97.41
-- Parallel efficiency	99.93	99.07	98.12
-- Load balance	100.00	99.29	99.09
-- Communication efficiency	99.93	99.78	99.03
-- Computation scalability	100.00	99.78	99.27
-- IPC scalability	100.00	99.76	99.42
-- Instruction scalability	100.00	100.00	100.00
-- Frequency scalability	100.00	100.02	99.86
Average IPC	2.26	2.26	2.25
Average frequency (GHz)	1.40	1.40	1.40



(a) Pioneer (b) BPI-F3 (c) Premier

Figure 5.11: OpenMP-only efficiency tables for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.12: Comparison of efficiency curves for SeisSol on HCA RISC-V platforms in OpenMP-only configuration, with and without Extrae instrumentation.

The granularity analysis for Milk-V Pioneer is shown in fig. 5.13. In general, for OpenMP-only configurations, the granularity is better defined on specific durations compared to MPI-only cases. Useful duration histograms reveal four distinct stripes, indicating the computational durations that appear most frequently during execution. As the thread count increases, these stripes shift towards shorter durations, indicating that tasks are completing faster with more threads. The accumulated time per thread also tends to decrease as the stripe color intensity lightens with increasing thread counts. Across all thread counts, the second stripe consistently appears as the most prominent, suggesting that this duration is the most common for task completion in SeisSol's `Simulator::simulate` region. Starting from 8 threads, the task duration of the second stripe is around 450 ms. As the thread count increases to 16, 24, 32, 40, 48, 56, and 64, the second stripe shifts to approximately 230, 150, 115, 90, 80, 65 and 60 ms, respectively.

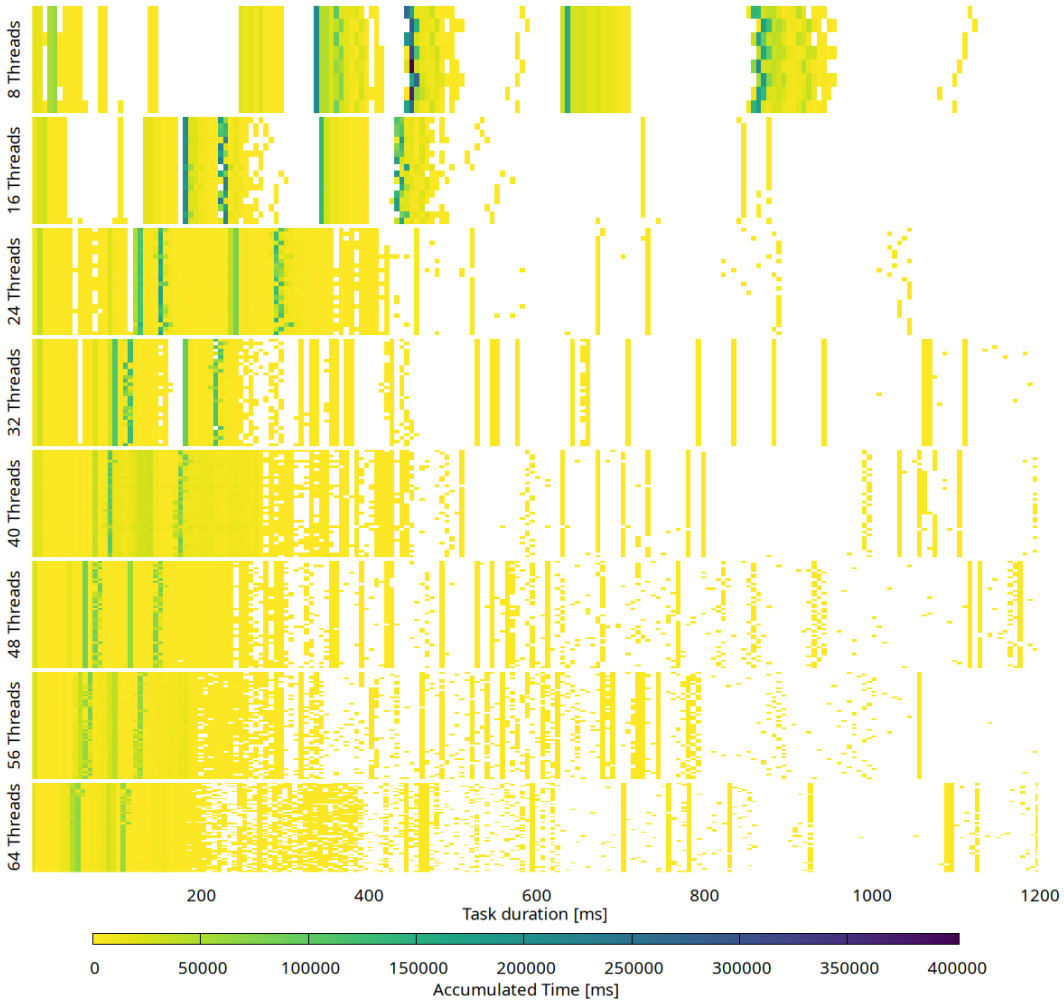


Figure 5.13: OpenMP granularity analysis for SeisSol on Milk-V Pioneer.

5.2.2.3 Hybrid MPI+OpenMP Tracing Analysis

Hybrid configurations transitioned from pure OpenMP to pure MPI by varying the number of MPI ranks and OpenMP threads per rank. Efficiency tables for SeisSol on Milk-V Pioneer,

Banana Pi BPI-F3, and SiFive Premier platforms are presented in fig. 5.14 for global efficiency and fig. 5.15 for hybrid parallel efficiency. These efficiency metrics are computed relative to the most efficient configuration observed for each platform. The global efficiency metric provides an overall measure of how effectively the application utilizes the available computational resources in a hybrid parallel environment. In addition to the global efficiency, the hybrid parallel efficiency metric offers insights into the specific performance characteristics of the hybrid parallelization approach.

For Milk-V Pioneer, the optimal configuration is found at 4 ranks with 16 threads, which is used as the baseline for calculating the computational scalability for other configurations. In general, MPI parallel efficiency decreases as the number of ranks increases, whereas OpenMP parallel efficiency remains relatively stable across non-pure configurations. This behavior is consistent with the statistics presented in section 5.1, which show that MPI communication time grows with the number of ranks, thereby reducing overall MPI efficiency.

As in previous tracing analyses, the overhead introduced by Extrae becomes more pronounced as the number of ranks increases, leading to a steady decline in global efficiency (see fig. 5.16). Again, this apparent degradation is primarily due to the impact of Extrae on per-rank computation rather than inherent limitations in the scalability of the application or hardware. Focusing on individual configurations, the configuration with 1 rank and 64 threads tested first shows the lowest global efficiency. Although the parallel efficiency is relatively high at 92 %, the computation efficiency is quite low at 34 %, primarily due to poor IPC scalability at 36 %. The second configuration, with 2 ranks and 32 threads, exhibits a significant improvement in global efficiency: the MPI and OpenMP parallel efficiencies are both relatively good at 85 % and the computation scalability improves to 86 %. As previously mentioned, the third configuration with 4 ranks and 16 threads achieves the highest global efficiency. In this configuration, performance loss is primarily caused by a decrease in MPI communication and OpenMP load balancing efficiencies. From the fourth configuration with 8 ranks and 8 threads to the last configuration with 64 ranks and 1 thread, communication efficiency remains relatively stable at around 85 %, while load balance efficiency varies slightly between 64 % and 72 %; computation efficiency gradually decreases, primarily due to declining IPC and instruction scalability, while frequency scalability remains at its highest level.

Banana Pi BPI-F3 shows excellent global efficiency above 90 % across all hybrid configurations, with the highest efficiency achieved in pure mode configurations. The parallel efficiency remains above 94 % for all configurations, with almost perfect MPI parallel efficiency. The computation efficiency is also almost perfect; only the third configuration with 4 ranks and 2 threads shows a slight drop to 94 %. For this board, the presence of Extrae is negligible, as the comparison in fig. 5.16 shows.

For SiFive Premier, the optimal configuration is found in pure OpenMP mode with 4 threads per rank. The global efficiency decreases from 92 % at 1 rank and 4 threads to 83 % at 4 ranks and 1 thread. The load balance efficiency remains above 98 % across all configurations. The

communication efficiency decreases slightly from 98 % at 1 rank and 4 threads to 88 % at 4 ranks and 1 thread, primarily due to MPI overhead. The computation scalability is excellent, above 95 % across all configurations; the IPC scalability is above 97 % and the average IPC is above 2.1; the instruction scalability is above 98 %; and the frequency scalability is above 99 %. The impact of Extrae becomes more noticeable as the number of ranks increases, as shown in fig. 5.16.

	1x64	2x32	4x16	8x8	16x4	32x2	64x1
Global efficiency	31.62	63.13	67.49	58.68	56.18	43.60	37.10
-- Parallel efficiency	92.56	73.29	67.49	59.07	58.27	55.97	63.16
-- Load balance	95.71	86.80	86.96	71.92	66.73	65.34	72.73
-- Communication efficiency	96.71	84.44	77.61	82.12	87.32	85.66	86.84
-- Computation scalability	34.17	86.14	100.00	99.35	96.40	77.89	58.74
-- IPC scalability	36.33	86.02	100.00	99.33	96.63	84.31	70.60
-- Instruction scalability	95.66	100.08	100.00	100.21	100.38	93.49	84.81
-- Frequency scalability	98.33	100.06	100.00	99.81	99.39	98.83	98.10
Average IPC	0.58	1.38	1.61	1.59	1.55	1.35	1.13
Average frequency (GHz)	1.96	2.00	2.00	1.99	1.99	1.97	1.96

	1x8	2x4	4x2	8x1
Global efficiency	96.35	94.84	90.67	96.18
-- Parallel efficiency	96.35	94.59	95.79	98.33
-- Load balance	98.42	97.45	97.76	99.35
-- Communication efficiency	97.90	97.07	97.98	98.97
-- Computation scalability	100.00	100.26	94.65	97.82
-- IPC scalability	100.00	97.95	93.61	99.16
-- Instruction scalability	100.00	102.32	101.04	98.50
-- Frequency scalability	100.00	100.05	100.07	100.14
Average IPC	0.77	0.75	0.72	0.76
Average frequency (GHz)	1.60	1.60	1.60	1.60

	1x4	2x2	4x1
Global efficiency	98.08	89.28	83.82
-- Parallel efficiency	98.08	90.78	87.58
-- Load balance	99.15	98.13	98.67
-- Communication efficiency	98.92	92.51	88.77
-- Computation scalability	100.00	98.35	95.70
-- IPC scalability	100.00	98.86	97.30
-- Instruction scalability	100.00	99.48	98.48
-- Frequency scalability	100.00	100.00	99.88
Average IPC	2.25	2.22	2.19
Average frequency (GHz)	1.40	1.40	1.40

100% 80% 60% 40% 20% 0%

(a) Pioneer (b) BPI-F3 (c) Premier

Figure 5.14: Global efficiency tables for SeisSol on HCA RISC-V platforms in hybrid MPI and OpenMP configuration.

	1x64	2x32	4x16	8x8	16x4	32x2	64x1
-- Hybrid Parallel efficiency	92.56	73.29	67.49	59.07	58.27	55.97	63.16
-- MPI Parallel efficiency	99.99	85.31	76.24	73.36	72.80	70.57	63.74
-- MPI Load balance	100.00	99.46	96.98	88.94	82.18	78.89	72.87
-- MPI Communication efficiency	99.99	85.78	78.62	82.49	88.59	89.46	87.47
-- OpenMP Parallel efficiency	92.56	85.91	88.52	80.51	80.05	79.30	99.09
-- OpenMP Load Balance	95.71	87.27	89.67	80.87	81.21	82.82	99.81
-- OpenMP Communication efficiency	96.71	98.44	98.72	99.55	98.57	95.75	99.28

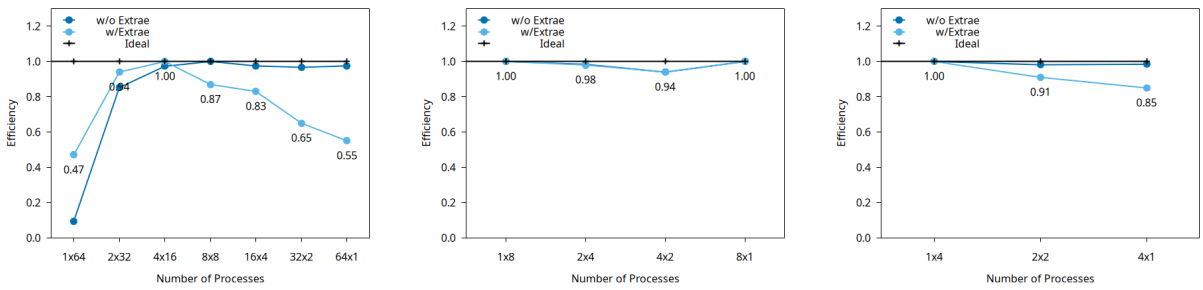
	1x8	2x4	4x2	8x1
-- Hybrid Parallel efficiency	96.35	94.59	95.79	98.33
-- MPI Parallel efficiency	100.00	99.29	99.14	98.91
-- MPI Load balance	100.00	99.94	99.83	99.62
-- MPI Communication efficiency	100.00	99.34	99.31	99.29
-- OpenMP Parallel efficiency	96.35	95.27	96.62	99.41
-- OpenMP Load Balance	98.42	97.51	97.93	99.73
-- OpenMP Communication efficiency	97.90	97.71	98.67	99.68

	1x4	2x2	4x1
-- Hybrid Parallel efficiency	98.08	90.78	87.58
-- MPI Parallel efficiency	99.99	92.85	87.84
-- MPI Load balance	100.00	99.22	98.74
-- MPI Communication efficiency	99.99	93.57	88.96
-- OpenMP Parallel efficiency	98.09	97.77	99.71
-- OpenMP Load Balance	99.15	98.90	99.92
-- OpenMP Communication efficiency	98.93	98.86	99.79

100% 80% 60% 40% 20% 0%

(a) Pioneer (b) BPI-F3 (c) Premier

Figure 5.15: Hybrid parallel efficiency tables for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.16: Comparison of efficiency curves for SeisSol on HCA RISC-V platforms in hybrid MPI and OpenMP configuration, with and without Extrae instrumentation.

The granularity analysis for Milk-V Pioneer is shown in fig. 5.17. In general, for hybrid MPI+OpenMP configurations, the granularity transitions from being well defined on specific durations in OpenMP-only cases to being less defined in MPI-only cases. In each hybrid configuration, the useful duration histograms reveal multiple stripes grouped by the common granularity of threads within each MPI rank. The first two hybrid configurations show four vertical stripes, with the second stripe being the most prominent around 57 ms, similar to the OpenMP-only case. The next three hybrid configurations exhibit two main vertical stripes ranging from 60 ms to 180 ms, with more variability across ranks. The last two hybrid configurations display a more diffused pattern, resembling the MPI-only case.

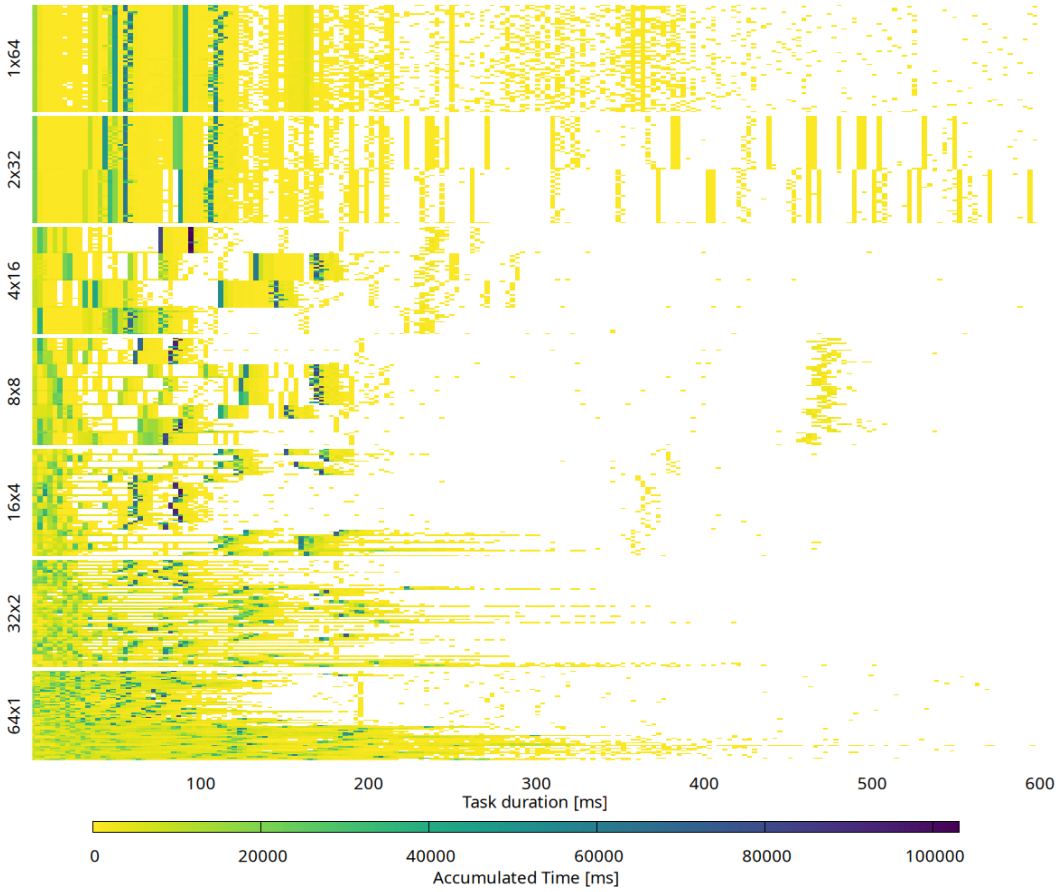


Figure 5.17: Hybrid MPI+OpenMP granularity analysis for SeisSol on Milk-V Pioneer.

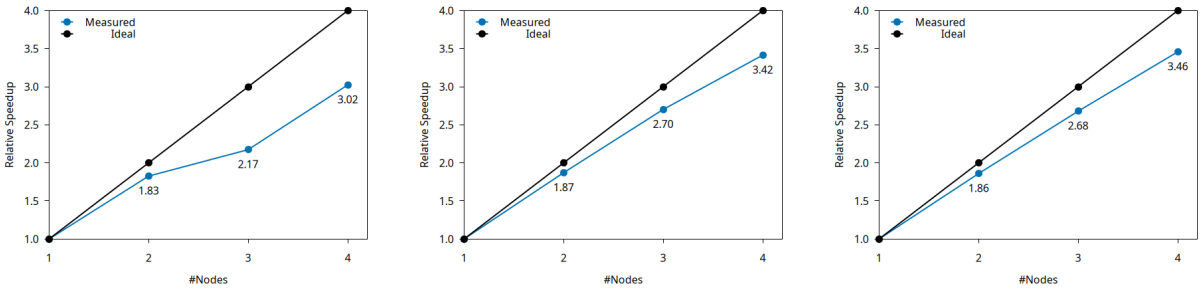
5.3 Inter Node Analysis

Inter-node performance analysis evaluates the scalability of SeisSol across the network fabric connecting multiple HCA RISC-V nodes with an optimal hybrid MPI+OpenMP configuration. For Milk-V Pioneer nodes, the full mesh is used, which contains 1 695 313 cells and 278 428 vertices. For Banana Pi BPI-F3 and SiFive Premier nodes, the same TPV5 SCEC benchmark test case is used as in the intra-node analysis. The simulation time is set to 1.0 s for all platforms. The build configuration for SeisSol remained consistent with the intra-node analysis.

5.3.1 Scalability Analysis

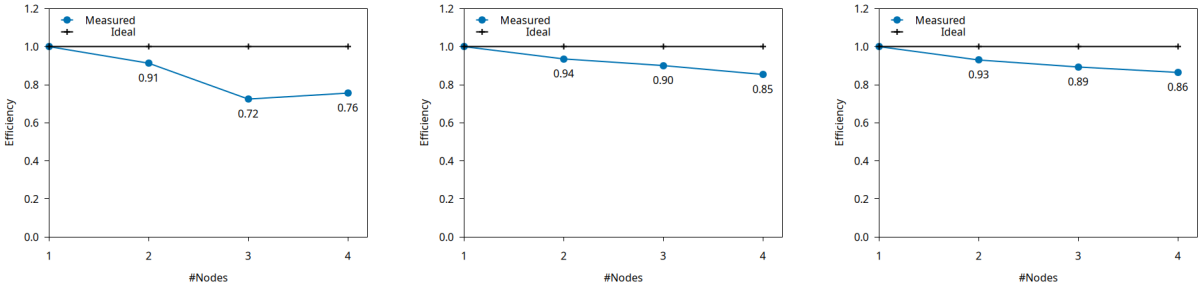
For the inter-node performance analysis, strong scaling experiments were conducted on Milk-V Pioneer, Banana Pi BPI-F3, and SiFive Premier using a hybrid MPI+OpenMP configuration, up to 4 nodes. The performance metric used for evaluation is the hardware FLOP/s reported by SeisSol's output. The scalability results of the inter-node configuration are presented in fig. 5.18 and fig. 5.19.

Milk-V Pioneer, using 4 ranks per node and 16 threads per rank, shows good but not ideal inter-node scaling. From 1 to 2 nodes, the speedup is 1.8× with an efficiency of 91 %. From 1 to 4 nodes, the speedup reaches 3.0× with an efficiency of 75 %. The slowdown in scaling may be due to limitations of the network fabric and increased communication overhead as the number of nodes increases. Banana Pi BPI-F3, using 1 ranks per node and 8 threads per rank, exhibits near-ideal inter-node scaling. The efficiency at 4 nodes is 85 %, which suggests that the problem size is sufficiently large to amortize communication overheads, and MPI message sizes are small and communication is simple. SiFive Premier, using 1 rank per node and 4 threads per rank, also demonstrates good inter-node scaling. With 4 nodes it reaches 86 % of ideal scaling. The communication overhead grows gradually but does not dominate within the tested node range.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.18: Inter-node speedup curves for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.19: Inter-node efficiency curves for SeisSol on HCA RISC-V platforms.

5.3.2 Tracing Analysis

To complement the inter-node scalability analysis, a tracing-based performance investigation is conducted using the POP performance metrics framework. The focus of the analysis is on the `Simulator::simulate` code region, identified as the FoA in the execution structure. The inter-node efficiency tables for SeisSol on Milk-V Pioneer, Banana Pi BPI-F3, and SiFive Premier platforms are presented in fig. 5.20. And the comparison of efficiency curves with and without Extrae instrumentation is shown in fig. 5.21.

Milk-V Pioneer, similar to the scalability results, has reasonably good global efficiency up to 2 nodes, then drops significantly at 3 and 4 nodes. Communication efficiency remains high, above 86 %, across all node counts, indicating that communication overhead is well managed. However, load balance efficiency decreases from 96 % with one node to 69 % with four nodes, suggesting uneven workload distribution as the number of nodes increases. This imbalance explains the large drop in parallel efficiency. Computation efficiency remains high across all node counts, with instructions, IPC, and frequency efficiencies above 97 %, indicating well-optimized computational kernels that effectively utilize the hardware capabilities. The overhead introduced by Extrae is present but does not significantly distort the overall efficiency trends.

Banana Pi BPI-F3 maintains a high global efficiency across all node counts, with only a slight decrease from 92 % with one node to 75 % with four nodes. Parallel efficiency degrades slowly from 92 % to 75 %. The load balance remains strong until the last node, when it drops to 78 %, indicating an imbalance at higher node counts. Communication efficiency remains excellent, staying above 92 % across all nodes. The computation efficiency is also very high, with all metrics above 95 %, demonstrating effective use of the hardware. The impact of Extrae is minimal, as shown in fig. 5.21, indicating that the instrumentation does not significantly affect the performance measurements.

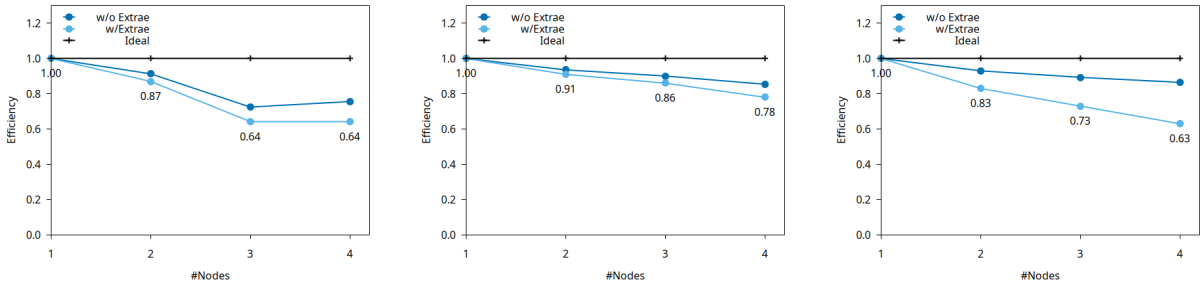
For SiFive Premier, even when adding just a second node, it does not scale smoothly, with global efficiency dropping from 98 % at 1 node to 81 % at 2 nodes, and further down to 71 % at 3 nodes. The load balance remains above 92 % across all node counts. However, communication efficiency drops significantly, from 98 % with 1 node to 79 % with 3 nodes, indicating that communication overhead becomes a bottleneck as more nodes are added. The computation efficiency remains solid, with all metrics above 97 %, demonstrating that the computational aspects of the application are well optimized. For this board, the presence of Extrae has a noticeable impact on the measured efficiencies as the node count increases, as illustrated in fig. 5.21.

	64(4x16)	128(8x16)	192(12x16)	256(16x16)	8(1x8)	16(2x8)	24(3x8)	32(4x8)	4(1x4)	8(2x4)	12(3x4)
Global efficiency	90.79	78.76	57.94	58.38	96.44	87.78	83.41	75.32	97.83	80.90	71.60
-- Parallel efficiency	90.79	79.33	60.69	60.64	96.44	89.05	85.73	78.86	97.83	82.44	73.53
-- Load balance	95.78	87.87	68.51	69.99	98.26	94.83	90.65	84.80	98.96	94.71	92.21
-- Communication efficiency	94.79	90.27	88.59	86.64	98.15	93.90	94.57	92.99	98.85	87.04	79.75
-- Computation scalability	100.00	99.29	95.48	96.28	100.00	98.57	97.30	95.52	100.00	98.14	97.37
-- IPC scalability	100.00	100.24	98.32	98.67	100.00	99.07	98.72	97.97	100.00	98.57	98.11
-- Instruction scalability	100.00	99.13	97.31	97.70	100.00	99.34	98.31	97.22	100.00	99.39	98.96
-- Frequency scalability	100.00	99.93	99.79	99.87	100.00	100.16	100.26	100.28	100.00	100.18	100.28
Average IPC	1.60	1.60	1.57	1.58	0.78	0.77	0.77	0.76	2.25	2.22	2.21
Average frequency (GHz)	2.00	2.00	2.00	2.00	1.60	1.60	1.60	1.60	1.40	1.40	1.40

100%
80%
60%
40%
20%
0%

(a) Pioneer (b) BPI-F3 (c) Premier

Figure 5.20: Inter-node efficiency tables for SeisSol on HCA RISC-V platforms.



(a) Milk-V Pioneer (b) Banana Pi BPI-F3 (c) SiFive Premier

Figure 5.21: Comparison of efficiency curves for SeisSol on HCA RISC-V platforms in inter-node configuration, with and without Extrae instrumentation.

5.4 Analysis Summary

This section summarizes the key findings from the intra-node and inter-node performance analyses of SeisSol on HCA RISC-V platforms.

In single-node scenarios, Milk-V Pioneer, Banana Pi BPI-F3, and SiFive Premier platforms demonstrate that SeisSol can effectively utilize their computational resources. Nevertheless, the performance is influenced by several factors, including the overhead introduced by Extrae instrumentation, NUMA effects in OpenMP-only configurations, and the choice of parallelization strategy. In the context of MPI-only configurations, SeisSol demonstrates excellent scalability up to all available cores on each platform. However, the instrumentation overhead resulting from Extrae compromises the integrity of the performance metrics, particularly at high rank counts. This overhead primarily impacts the computation efficiency metrics, such as IPC and instruction scalability, due to the additional instructions introduced by Extrae. In OpenMP-only configurations, performance can be affected by memory access patterns, especially on systems like Milk-V Pioneer with non-uniform memory access architectures. At high thread counts, contention and latency can lead to performance degradation. Hybrid MPI+OpenMP configurations offer a balanced approach to parallelization, but the choice of ranks and threads per rank is crucial for optimal performance. Granularity analysis reveals

that task durations tend to shorten with increasing core counts. The grain size is more defined in OpenMP-only configurations compared to MPI-only and hybrid cases. One possible cause is the persistent MPI communication patterns that allow ranks to progress independently, leading to more variability in task durations. In contrast, in OpenMP-only mode, threads share memory and can synchronize more efficiently, resulting in more consistent task durations.

In multi-node scenarios, communication overhead significantly impacts scalability. On platforms such as Milk-V Pioneer and SiFive Premier, communication efficiency decreases as more nodes are added. This indicates that limitations in the network fabric and increased message passing overhead can impact performance. Conversely, Banana Pi BPI-F3 maintains high communication efficiency across multiple nodes; however, load balancing issues arise at higher node counts. Despite these challenges, the computational kernels of SeisSol remain well-optimized across all platforms, consistently achieving high computation efficiency metrics.

While SeisSol demonstrates good scalability on HCA RISC-V platforms, certain limitations inherent to these architectures may contribute to performance overheads observed in the analyses. The lack of a larger set of hardware performance counters limits a more detailed investigation into the specific causes of performance bottlenecks. Some mathematical libraries used by SeisSol may not be fully optimized for RISC-V architectures, leading to suboptimal performance in certain computational kernels. Additionally, the network fabrics connecting these nodes may not be optimized for high-performance computing workloads, resulting in increased communication latency and reduced bandwidth.

To extend the performance evaluation of SeisSol beyond RISC-V platforms, during the midterm discussion of this thesis, the direction was set to include an analysis on a well established high-performance computing system. Since SeisSol is developed primarily for exascale and petascale computing systems, including CPU- and GPU-based architectures, evaluating its performance on such a platform provides a benchmark for comparison with the RISC-V results obtained in this chapter. The next chapter presents a performance analysis of SeisSol on the Leonardo supercomputer, which is based on an x86_64 architecture. The analysis focuses on similar intra-node and inter-node performance metrics to facilitate a direct comparison with the RISC-V results.

Chapter 6

Performance Analysis II

To complement the performance analysis of SeisSol on RISC-V architectures presented in chapter 5, this chapter focuses on evaluating the performance of SeisSol on a supercomputer for HPC applications based on the x86_64 architecture. The analysis compares how SeisSol performs on traditional HPC hardware, emphasizing the differences and similarities between RISC-V and x86_64 architectures.

This chapter begins with an introduction to the hardware and software environments. sections 6.1 and 6.2 present intra-node and inter-node performance analyses, respectively. These analyses include scalability and tracing-based investigations conducted under various parallel configurations. Finally, the analysis summary contrasts x86_64 architecture results with those from the RISC-V architecture.

The Booster module of the Leonardo Supercomputer, which is hosted by the CINECA [20] interuniversity consortium in Italy, serves as the testbed for this analysis. The Booster module consists in 3456 nodes configured with a 32-core Intel Ice Lake CPU [22], and four NVidia A100 Tensor Core GPU chips [24]. The internal network for inter-node communication relies on 200 Gbps Mellanox's Infiniband High Data Rate (HDR) technologies [23] and is organized in a Dragonfly+ topology [20], [66]. The analysis presented in this chapter focuses exclusively on the CPU part of the Booster nodes, limiting the inter-node experiments to 4 nodes. The memory configuration of each node consists of two NUMA domains, each with 16 cores and 256 GB of DDR4 memory, resulting in a total of 512 GB of memory per node. The latency cost between NUMA nodes is detailed in table 6.1.

Table 6.1: NUMA distances in a Booster node of the Leonardo Supercomputer.

Node	**0**	**1**
0	10	11
1	11	10

The sofware stack for building and executing SeisSol on Leonardo is based on the GNU Compiler Collection (GCC) version 12.2.0 and MPI 4.1.6. using the following cmake options:

- EQUATIONS=elastic
- ORDER=4
- PRECISION=double
- HOST_ARCH=skx
- NUMA_AWARE_PINNING=ON
- GEMM_TOOLS_LIST=eigen
- ASAGI=ON

The operating system is Red Hat Enterprise Linux 8.7 (Ootpa), and the workload manager is SLURM 23.11.

The Instrumentation framework to generate execution traces is Extrae version 4.3.3 [8], with PAPI support to collect hardware performance counters of the microprocessor during the execution. The analysis of the traces is performed using Paraver along with the POP performance metrics framework keeping the same FoA as in chapter 5. For the scalability analysis in both intra-node and inter-node experiments, the performance metric used for the speedup and efficiency calculations is the number of FLOP/s reported by SeisSol's output. The test case and input parameters were consistent with those used in the previous chapter for RISC-V architectures, ensuring a fair comparison of performance across different hardware platforms.

6.1 Intra Node Analysis

The intra-node performance analysis evaluates the scalability and performance of SeisSol on a single node of the Booster partition of the Leonardo Supercomputer with different parallel configurations. The analysis is divided into two main parts. Similar to chapter 5, the first part focuses on scalability analysis, while the second part delves into tracing-based performance investigation. The mesh is kept the same as in section 5.2 to ensure consistency in the workload across different hardware platforms.

6.1.1 Scalability Analysis

For the intra-node scalability analysis, three parallel configurations are evaluated: MPI-only, OpenMP-only, and hybrid MPI+OpenMP. The performance metric used for speedup and efficiency calculations is the number of FLOP/s reported by SeisSol's output.

6.1.1.1 MPI-only Analysis

In MPI-only configurations, SeisSol is executed with varying numbers of MPI ranks from 1 to 32, with each rank pinned to a specific core. The results of relative speedup and efficiency scalability are presented in fig. 6.1. Each measurement is averaged over five runs, and the relative standard deviation is below 1 %. The base configuration for calculating speedup and

efficiency is the single-rank execution, achieving 14.1 GFLOP/s. The peak performance is 336 GFLOP/s achieved with 32 MPI ranks. In this configuration, SeisSol demonstrates good scalability above 90 % up to 16 MPI ranks, after which the efficiency begins to decline more noticeably as the number of ranks approaches the maximum of 32. One possible reason for this behavior is the communication overhead that becomes more pronounced with a higher number of MPI ranks sharing the same physical resources.

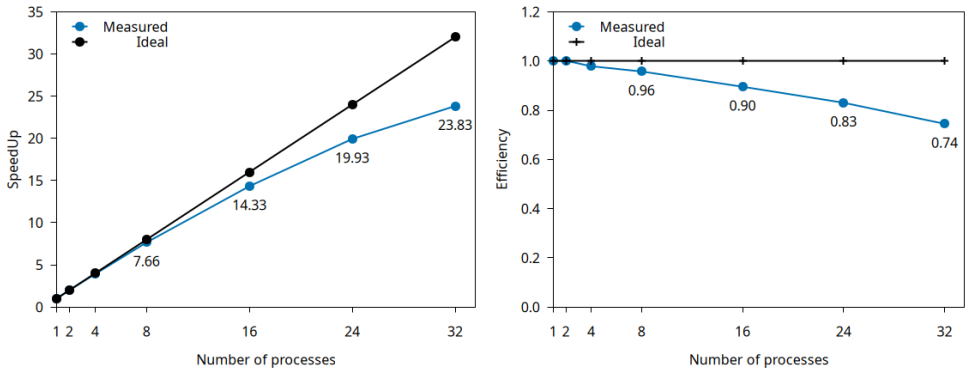


Figure 6.1: Speedup and efficiency plots for SeisSol on Leonardo Supercomputer using MPI-only configuration.

The observed scalability behavior may also be influenced by Intel's Dynamic Voltage and Frequency Scaling (DVFS) mechanisms. Leonardo's Intel Ice Lake processors implement both Intel Speed Shift Technology and Intel Turbo Boost Technology, which dynamically adjust core frequencies in response to workload demands and thermal conditions. Speed Shift allows the processor to dynamically adjust its performance states based on workload demands, improving power efficiency [58], while Turbo Boost enables the processor to increase its clock speed beyond the base frequency when thermal and power conditions permit, offering a performance boost for demanding workloads [50]. These dynamic frequency adjustments can lead to variations in performance as the number of active cores changes, potentially impacting the scalability.

6.1.1.2 OpenMP-only Analysis

In OpenMP-only configurations, SeisSol is executed using different numbers of OpenMP threads from 1 to 32, placing the worker threads in contiguous cores to optimize memory access patterns. The results of speedup and efficiency scalability are presented in fig. 6.2. The base configuration for calculating speedup and efficiency is the single-thread execution, achieving 14.1 GFLOP/s. The peak performance is 335 GFLOP/s achieved with 32 MPI threads. Similar to the MPI-only case, the OpenMP-only configuration shows good scalability up to 16 threads. However, efficiency decreases as the number of threads increases beyond 16. This is consistent with the observations made in section 5.2 regarding RISC-V architectures, where

OpenMP performance degraded due to suboptimal memory access patterns. In this case, though, the relatively low latency cost between NUMA domains suggests that the performance degradation may be more related to computational factors, such as IPC and frequency scaling, than to memory access issues.

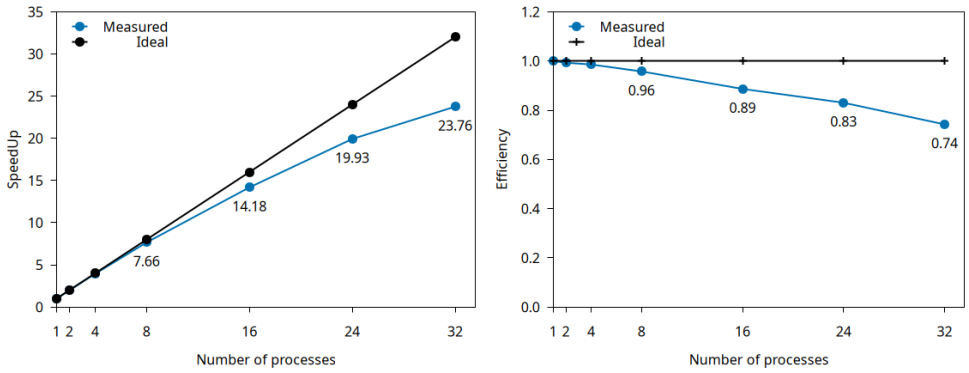


Figure 6.2: Speedup and efficiency plots for SeisSol on Leonardo Supercomputer using OMP-only configuration.

6.1.1.3 Hybrid MPI+OpenMP Analysis

In the hybrid MPI+OpenMP configuration, SeisSol is executed using a combination of MPI ranks and OpenMP threads, varying their proportions to explore different configurations while utilizing all available cores in the node. The relative efficiency scalability results are presented in fig. 6.3. All hybrid configurations reach a peak performance between 334 to 336 GFLOP/s, closely matching the peak performances observed in the MPI-only and OpenMP-only configurations. In this way, the relative efficiency remains above 99 % across all tested configurations, demonstrating that SeisSol can effectively leverage both parallel programming models in a hybrid setup without significant performance degradation.

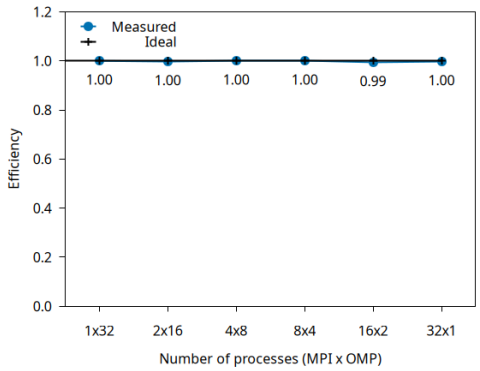


Figure 6.3: Efficiency plot for SeisSol on Leonardo Supercomputer using hybrid MPI and OMP configuration.

6.1.2 Tracing Analysis

The second part of the intra-node performance analysis extends the scalability study with a detailed tracing-based performance investigation. The FoA selected for this analysis is the same as in chapter 5, focusing on the simulation phase of SeisSol.

6.1.2.1 MPI-only Tracing Analysis

The POP efficiency metrics for the MPI-only configuration are presented in fig. 6.4. The global efficiency reflects the scalability trend observed in the direct measurements, maintaining very high values above 94 % up to 8 ranks, decreasing to 70 % at 32 ranks. Regarding parallel efficiency, the application exhibits remarkably high load balance and communication efficiencies, both remaining above 99 % across all configurations. The computation scalability remains excellent up to 8 ranks and then decreases steadily due to the microprocessor's dynamic frequency scaling mechanisms at higher core counts; this confirms the observations made in the scalability analysis. The average IPC is very good, above 2.5 instructions per cycle for all configurations. An important observation is that the overhead introduced by Extrae instrumentation is negligible. This is evidenced by the close alignment between the scalability outcomes derived from the tracing analysis and those obtained from direct scalability measurements (refer to fig. 6.5). This indicates that the tracing methodology provides an accurate representation of the application's performance without significantly impacting its execution.

	1	2	4	8	16	24	32
Global efficiency	100.00	99.39	96.84	94.91	87.68	80.94	70.92
-- Parallel efficiency	100.00	99.96	99.90	99.81	99.67	99.56	99.45
-- Load balance	100.00	99.99	99.96	99.97	99.87	99.80	99.71
-- Communication efficiency	100.00	99.97	99.94	99.83	99.80	99.76	99.74
-- Computation scalability	100.00	99.43	96.93	95.09	87.96	81.30	71.32
-- IPC scalability	100.00	99.69	98.64	97.09	93.28	95.01	95.98
-- Instruction scalability	100.00	99.72	98.22	97.90	97.18	95.10	95.44
-- Frequency scalability	100.00	100.02	100.05	100.05	97.03	89.97	77.86
Average IPC	2.67	2.66	2.63	2.59	2.49	2.54	2.56
Average frequency (GHz)	3.30	3.30	3.30	3.30	3.20	2.97	2.57

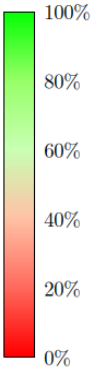


Figure 6.4: MPI-only efficiency table for SeisSol on Leonardo Supercomputer.

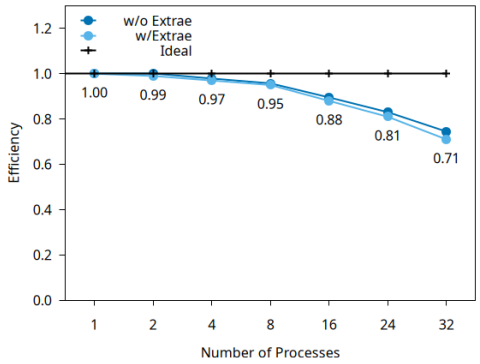


Figure 6.5: Comparison of MPI-only relative efficiency for SeisSol on Leonardo Supercomputer from scalability and tracing analysis.

The task granularity during useful computation phases for different MPI rank configurations is illustrated in fig. 6.6. The grain size distributions demonstrate that, as the number of MPI ranks increases, task granularity tends to decrease. However, the most frequent grain sizes are not well-defined and span a wide range, indicating variability in task sizes for different ranks. A similar trend is observed in the RISC-V architectures analyzed in chapter 5, suggesting that this behavior is inherent to the application's parallelization strategy rather than being specific to a particular hardware architecture.

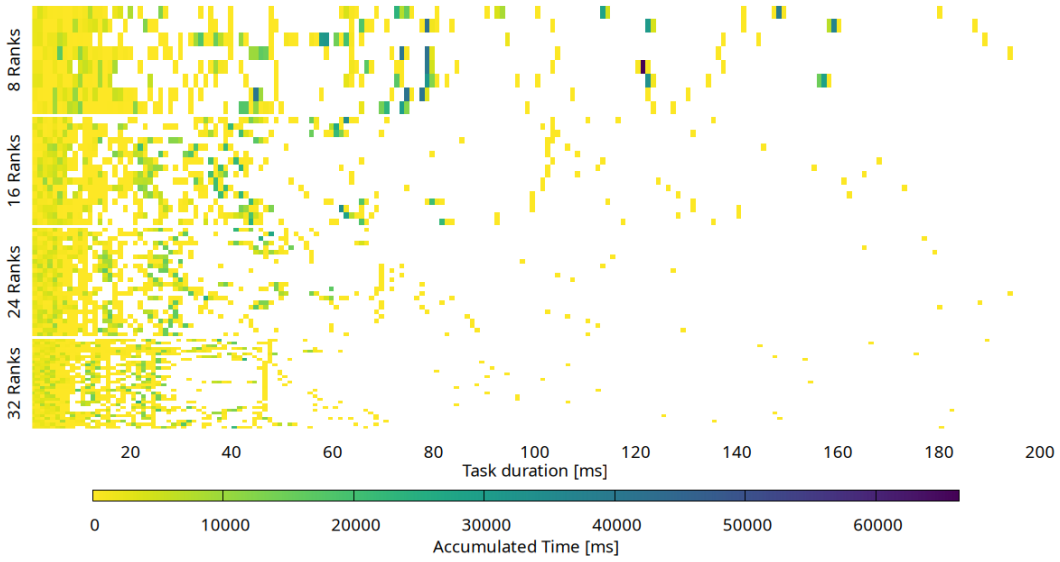


Figure 6.6: MPI-only granularity analysis for SeisSol on Leonardo Supercomputer.

6.1.2.2 OpenMP-only Tracing Analysis

The efficiency metrics for the OpenMP-only configuration are presented in fig. 6.7. The global efficiency trend mirrors that observed in the direct measurements, starting at 99 % with 1 thread and gradually decreasing to 73 % at 32 threads. The parallel efficiency is almost as scalable as in the MPI-only case, with only minor differences at higher thread counts. For example, with 32 threads, the parallel efficiency is 96 %, compared to 99 % in the MPI-only

configuration. Overall, the application maintains a very good load balance and very low communication overhead. The IPC and instruction scaling remain strong. As the number of threads increases, dynamic frequency scaling becomes the main factor affecting the linearity of computation scalability. The frequency scaling metric decreases from 97 % at 16 threads to 80 % at 32 threads. As fig. 6.8 shows, the scalability results from the tracing analysis closely align with those obtained from direct scalability measurements, indicating that the impact of Extrae instrumentation is minimal.

	1	2	4	8	16	24	32
Global efficiency	99.99	98.96	97.93	95.23	88.08	82.13	73.45
-- Parallel efficiency	99.99	99.44	99.02	98.25	97.66	96.71	96.08
-- Load balance	100.00	99.97	99.56	98.70	98.81	97.85	98.79
-- Communication efficiency	99.99	99.46	99.46	99.54	98.84	98.84	97.26
-- Computation scalability	100.00	99.52	98.89	96.93	90.19	84.92	76.45
-- IPC scalability	100.00	99.42	98.83	96.89	92.94	94.08	94.74
-- Instruction scalability	100.00	100.00	100.00	100.00	99.99	99.98	99.97
-- Frequency scalability	100.00	100.10	100.07	100.04	97.05	90.27	80.71
Average IPC	2.67	2.65	2.64	2.59	2.48	2.51	2.53
Average frequency (GHz)	3.30	3.30	3.30	3.30	3.20	2.98	2.66

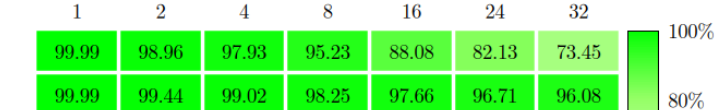


Figure 6.7: OpenMP-only efficiency table for SeisSol on Leonardo Supercomputer.

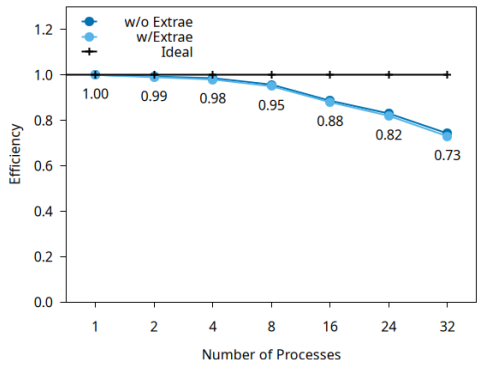


Figure 6.8: Comparison of OMP-only relative efficiency for SeisSol on Leonardo Supercomputer from scalability and tracing analysis.

The task granularity during useful computation phases for different OpenMP thread configurations is illustrated in fig. 6.9. The OpenMP granularity plots show more defined stripes compared to the MPI-only case, indicating that certain task sizes are more prevalent. To be specific, four vertical stripes can be distinguished in all thread configurations, which is in fact similar to the granularity patterns observed in the RISC-V architectures analyzed in chapter 5. As the number of threads increases, the most frequent grain sizes tend to decrease, reflecting the finer granularity of tasks as more threads share the workload. From 8 to 16 threads, the duration of the most frequent grain sizes is reduces by roughly half, indicating a substantial

decrease in task size as the thread count doubles. From 16 to 32 threads, these grain sizes continue to shrink, although not by exactly half, suggesting that the workload is still being divided among the available threads, but with diminishing returns due to thread management overheads, such as scheduling, context switching, and synchronization costs.

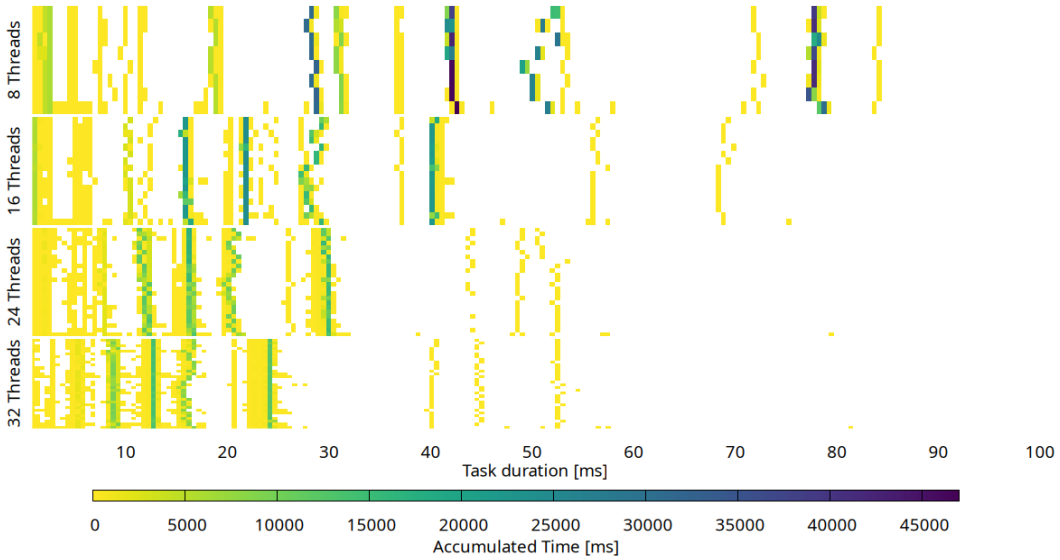


Figure 6.9: OpenMP-only granularity analysis for SeisSol on Leonardo Supercomputer.

6.1.2.3 Hybrid MPI+OpenMP Tracing Analysis

The hybrid MPI+OpenMP efficiency metrics are presented in two separate tables: the global efficiency metrics in fig. 6.10 and the hybrid parallel efficiency metrics in fig. 6.11 where the parallel efficiency is broken down into MPI and OpenMP components. Each hybrid configuration utilizes all 32 cores of the node, varying the number of MPI ranks and OpenMP threads accordingly. Generally, the global efficiency is very good across all configurations, but it tends to decrease slowly as the number of MPI ranks increases. The highest global efficiency of 96 % is observed in the configuration with 1 MPI rank and 32 OpenMP threads, while the lowest efficiency of 89 % occurs in the configuration with 32 MPI ranks and 1 OpenMP thread. The parallel efficiency remains high in all configurations, with both MPI and OpenMP components contributing positively to the overall efficiency. The MPI parallel efficiency is particularly strong in all configurations, with values above 99 %, while the OpenMP parallel efficiency is slightly lower but still very good, ranging from 92 % to 96 %. Computation scalability is excellent across all configurations, decreasing slightly only in the last two configurations with higher MPI ranks due to a slight drop in instruction and frequency scaling. IPC scaling remains very good in all configurations, above 2.5 instructions per cycle. The overhead introduced by Extrae instrumentation remains negligible in hybrid configurations as well, only causing minimal deviations in the last two configurations with higher MPI ranks (see fig. 6.12).

	1x32	2x16	4x8	8x4	16x2	32x1
Global efficiency	96.05	92.80	92.66	91.72	88.32	89.68
-- Parallel efficiency	96.05	95.03	93.39	91.99	91.97	95.63
-- Load balance	98.71	97.08	96.48	94.57	96.79	99.65
-- Communication efficiency	97.30	97.89	96.80	97.28	95.03	95.96
-- Computation scalability	100.00	97.65	99.22	99.70	96.03	93.78
-- IPC scalability	100.00	100.68	100.86	100.83	100.47	101.61
-- Instruction scalability	100.00	99.87	99.55	98.82	98.04	95.60
-- Frequency scalability	100.00	97.11	98.82	100.06	97.49	96.53
Average IPC	2.52	2.53	2.54	2.54	2.53	2.56
Average frequency (GHz)	2.72	2.64	2.68	2.72	2.65	2.62

100% 80% 60% 40% 20% 0%

Figure 6.10: Global efficiency table for SeisSol on Leonardo Supercomputer in hybrid MPI and OpenMP configuration.

	1x32	2x16	4x8	8x4	16x2	32x1
-- Hybrid Parallel efficiency	96.05	95.03	93.39	91.99	91.97	95.63
-- MPI Parallel efficiency	100.00	99.60	99.52	99.52	99.51	99.47
-- MPI Load balance	100.00	99.99	99.92	99.87	99.78	99.72
-- MPI Communication efficiency	100.00	99.61	99.61	99.65	99.73	99.75
-- OpenMP Parallel efficiency	96.05	95.41	93.84	92.44	92.42	96.14
-- OpenMP Load Balance	98.71	97.08	96.56	94.69	97.00	99.93
-- OpenMP Communication efficiency	97.31	98.28	97.18	97.62	95.28	96.20

100% 80% 60% 40% 20% 0%

Figure 6.11: Hybrid parallel efficiency table for SeisSol on Leonardo Supercomputer.

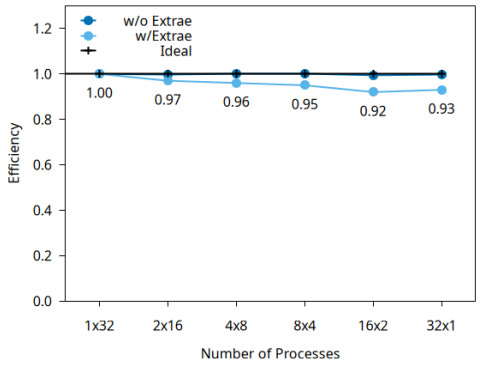


Figure 6.12: Comparison of hybrid MPI+OpenMP relative efficiency for SeisSol on Leonardo Supercomputer from scalability and tracing analysis.

The task granularity during useful computation phases for different hybrid MPI and OpenMP configurations is illustrated in fig. 6.13. For the first two configurations, 1×32 and 2×16

(MPI ranks × OpenMP threads), the granularity patterns are similar to those observed in the OpenMP-only case, with well-defined stripes indicating prevalent task sizes. As the number of MPI ranks increases and the number of OpenMP threads decreases, the granularity patterns begin to resemble those seen in the MPI-only case, with less defined stripes and a wider range of task sizes. Short task sizes become more frequent, particularly in configurations with higher MPI ranks, possibly as a result of redundant instructions being executed concurrently by multiple MPI ranks.

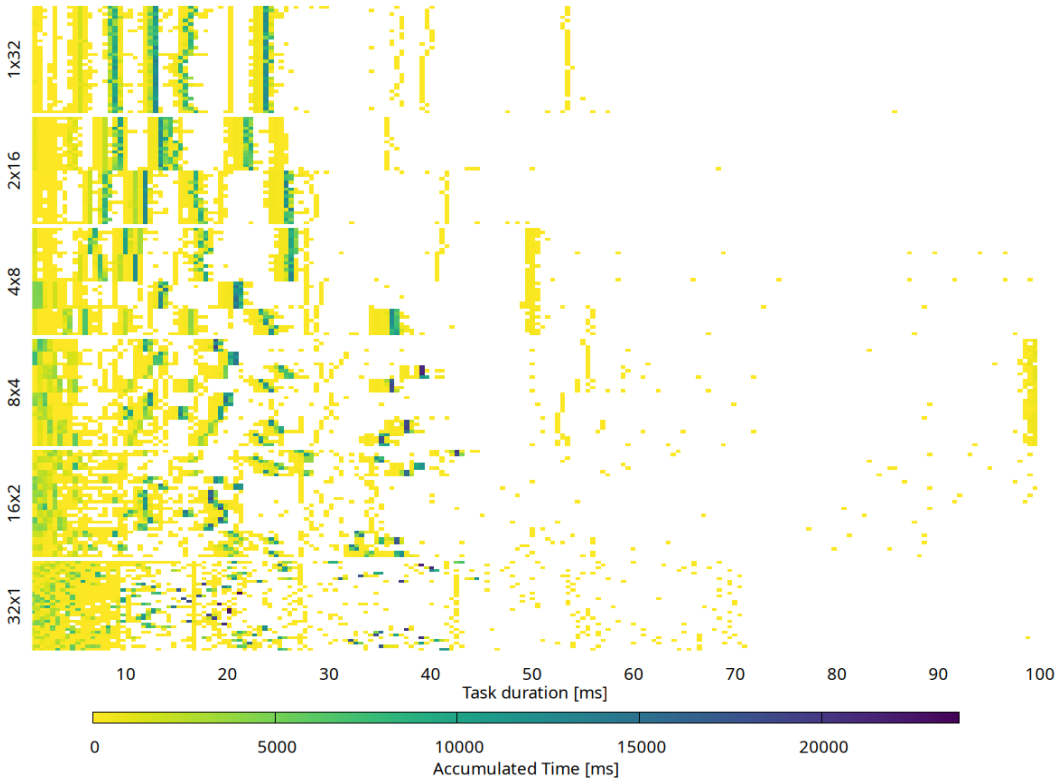


Figure 6.13: Hybrid MPI and OpenMP granularity analysis for SeisSol on Leonardo Supercomputer.

6.2 Inter Node Analysis

The inter-node performance analysis evaluates the scalability and performance of SeisSol from 1 to 4 compute nodes of the Booster partition of the Leonardo Supercomputer. Each compute node with a single MPI rank and 32 OpenMP threads. The analysis is divided in a scalability analysis and a tracing-based performance investigation. The full mesh is used for all inter-node experiments, similar to the Pioneer nodes in section 5.3.

6.2.1 Scalability Analysis

The results of relative speedup and efficiency scalability are presented in fig. 6.14. The base configuration for calculating speedup and efficiency is the single-node execution, achieving

320 GFLOP/s. At 4 compute nodes, the performance reaches 1.27 TFLOP/s. Each measurement is averaged over five runs, and the relative standard deviation is below 1 %. SeisSol demonstrates perfect scalability across all tested configurations, maintaining an efficiency above 99 % up to 4 nodes.

Although the inter-node scalability analysis in this thesis is limited to four nodes, other studies indicate that SeisSol can effectively utilize multiple nodes in a supercomputing environment. For instance, a performance assessment report conducted on the MareNostrum 4 supercomputer at BSC revealed that SeisSol exhibited excellent inter-node scalability when a dedicated communication thread was used, with relative efficiencies (compared to the smallest configuration with four compute nodes) above 80 % up to 128 nodes [71].

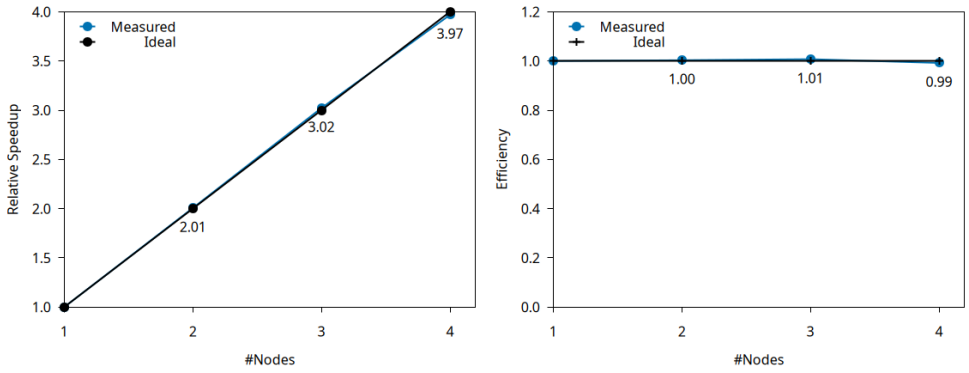


Figure 6.14: Speedup and efficiency plots for SeisSol on Leonardo Supercomputer using inter-node configuration.

6.2.2 Tracing Analysis

To provide a detailed performance investigation of SeisSol in inter-node configurations, a tracing-based analysis is conducted using Extrae and Paraver. The FoA selected for this analysis is consistent with the previous sections, focusing on the simulation phase of SeisSol.

The global efficiency metrics for the inter-node configuration are presented in fig. 6.15. The global efficiency remains very high across all configurations, between 93 % and 97 %. The parallel efficiencies mirror the global efficiency trend, with both load balance and communication efficiencies remaining above 95 % in all configurations. Computation scalability is excellent, with all metrics (computation, instruction, IPC, and frequency scaling) maintaining values above 98 %. The average IPC is very good, around 2.5 instructions per cycle for all configurations. It is observed that all nodes appear to function at the base frequency of 2.6 GHz during the simulation phase. The overhead introduced by Extrae instrumentation is negligible, as evidenced by the close alignment between the scalability outcomes derived from the tracing analysis and those obtained from direct scalability measurements (refer to fig. 6.16).

	32(1x32)	64(2x32)	96(3x32)	128(4x32)
Global efficiency	97.53	95.89	93.67	94.38
-- Parallel efficiency	97.53	96.78	93.84	93.34
-- Load balance	98.60	97.83	95.06	95.08
-- Communication efficiency	98.91	98.93	98.72	98.16
-- Computation scalability	100.00	99.08	99.81	101.12
-- IPC scalability	100.00	100.51	100.33	100.33
-- Instruction scalability	100.00	99.95	99.81	99.79
-- Frequency scalability	100.00	98.62	99.68	101.00
Average IPC	2.48	2.50	2.49	2.49
Average frequency (GHz)	2.58	2.54	2.57	2.61

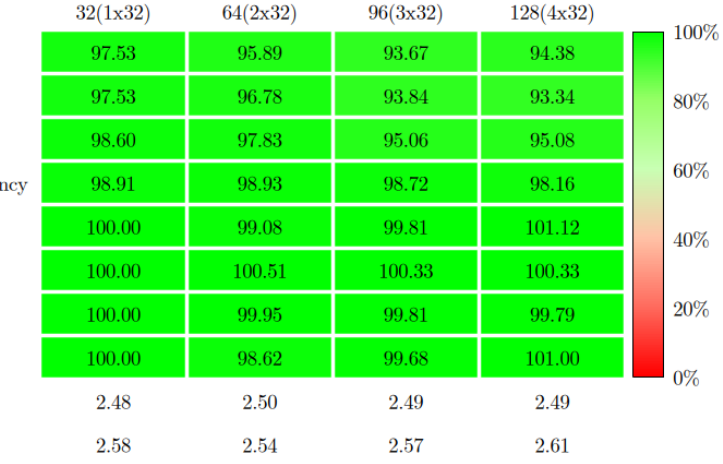


Figure 6.15: Efficiency table for SeisSol on Leonardo Supercomputer using inter-node configuration.

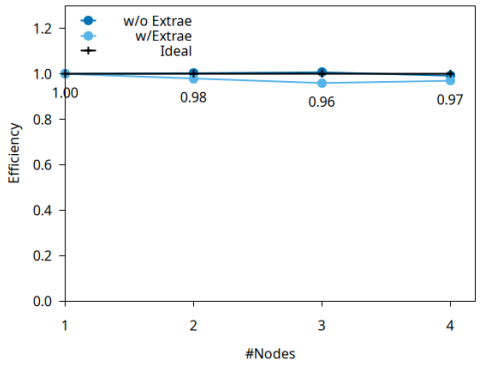


Figure 6.16: Comparison of inter-node relative efficiency for SeisSol on Leonardo Supercomputer from scalability and tracing analysis.

6.3 Multiplatform Extrae overhead analysis

To get a better understanding of Extrae's impact, fig. 6.17 presents the breakdown of Extrae overhead by component for SeisSol Proxy on HCA platforms and Leonardo Supercomputer with different cell sizes and 10 iterations. Only Extrae routines that introduce measurable overhead are included in the plots. In this case, the observed instrumentation overhead is dominated by the Extrae routines that perform call-stack inspection, while the rest of the tracing functionality has a negligible impact. Basic tracing operations such as timestamp acquisition and direct performance counter reads introduce minimal overhead. These operations mainly involve lightweight buffer writes and, in the case of PAPI, straightforward hardware counter accesses, which scale well and do not significantly perturb execution. In contrast, a clear overhead increase is observed for `extrae_user_function`, `extrae_get_caller1`, `extrae_get_caller6`, and `extrae_trace_callers`. All of these routines rely on traversing the processor call stack to recover caller information. Additionally, this overhead becomes

more pronounced when using smaller mesh sizes, or more generally when the computational domain is decomposed into fewer cells per subdomain, particularly for Milk-V Pioneer. In the regime of 10 000 cells, the relative cost of instrumentation rises because the amount of useful work performed between successive instrumentation points decreases. This behavior is consistent with observations in MPI configurations on RISC-V architectures in chapter 5. as the number of MPI ranks increases, each rank operates on a smaller subdomain, resulting in fewer cells per rank. The resulting finer-grained workload typically triggers more frequent function calls, including the call-stack–based Extrae routines. Consequently, the fixed cost of stack traversal is amortized over less computation, making the instrumentation overhead increasingly visible at higher rank counts.

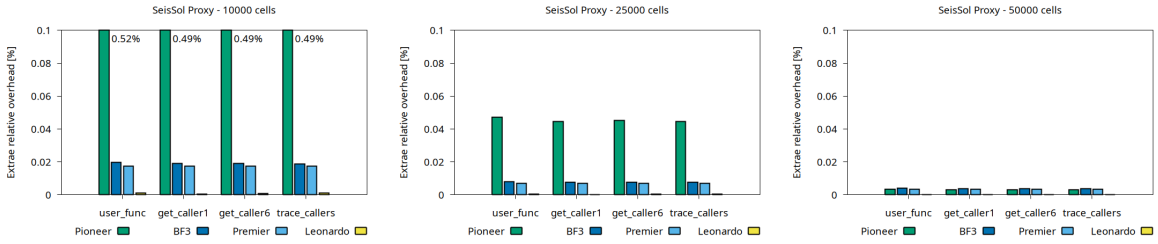


Figure 6.17: Relative overhead of Extrae components for SeisSol Proxy on HCA platforms and Leonardo Supercomputer with different cell sizes.

6.4 Analysis Summary

SeisSol demonstrates high performance and scalability on the CPU nodes of the Leonardo Booster partition, achieving optimal performance in both intra-node and inter-node configurations. This section summarizes the performance analysis of SeisSol on the Leonardo Supercomputer and compares the results with those obtained on RISC-V architectures in chapter 5.

In single-node configurations, DVFS mechanisms of the Intel Ice Lake processors play a significant role in performance scaling, particularly as the number of active cores increases. Both MPI-only and OpenMP-only configurations exhibit linear scalability up to 16 ranks or threads. Beyond 16 ranks or threads, efficiency declines due to frequency scaling, which limits the achievable performance but still allows for good overall scalability. This behavior contrasts with the RISC-V architectures analyzed in chapter 5. In MPI-only configurations on RISC-V, scalability remained high across all core counts, with minimal impact from frequency scaling. In OpenMP-only configurations on RISC-V Milk-V Pioneer, scalability was good up to 32 threads but degraded significantly beyond that point, where memory access patterns had a more pronounced impact on performance at higher core counts. Hybrid MPI+OpenMP configurations maintain high efficiency across all tested setups. In Leonardo, the best performance is achieved with 1 MPI rank and 32 OpenMP threads, but other configurations also perform

very well. In RISC-V architectures, hybrid configurations also show good performance, but the optimal balance between MPI ranks and OpenMP threads may vary depending on the specific architecture and memory access patterns. Tracing-based performance analysis reveals that both architectures[1] maintain high load balance and low communication overhead in all configurations. Instruction scalability is excellent in both architectures. IPC scaling is very good in Leonardo, remaining above 2.5 instructions per cycle across all configurations. In RISC-V architectures, IPC scaling is also good, but it tends to decrease more noticeably at higher core counts, particularly in OpenMP-only configurations due to memory access inefficiencies.

Granularity analysis are similar in both architectures. In MPI-only configurations, task granularity decreases as the number of MPI ranks increases, with a wide range of task sizes observed. In OpenMP-only configurations, more defined task size patterns emerge, with prevalent grain sizes that decrease as the number of threads increases. In hybrid configurations, granularity patterns transition from OpenMP-like to MPI-like as the number of MPI ranks increases and the number of OpenMP threads decreases.

In multi-node configurations, SeisSol exhibits near-perfect scalability on Leonardo, maintaining efficiency above 99 % up to 4 nodes. On HCA platforms, SeisSol demonstrates good inter-node scalability but load balance and communication overhead become more significant factors as the number of nodes increases, leading to a gradual decline in efficiency.

Extrae instrumentation overhead is negligible in Leonardo across all configurations. In RISC-V architectures, Extrae overhead is also minimal in most configurations, except for high MPI rank counts on Milk-V Pioneer.

[1]With the exception of MilK-V Pioneer with high MPI rank counts, whose POP performance metrics were affected by Extrae overhead.

Chapter 7

Conclusions

This thesis presents a comprehensive performance analysis of the SeisSol application on various RISC-V architectures. The evaluation was conducted using the POP performance methodology and BSC tools. SeisSol was utilized as a means to investigate the capabilities and performance characteristics of RISC-V platforms. This decision was influenced by the demonstrated scalability and high-performance capabilities of SeisSol across a diverse range of HPC systems. The performance analysis encompassed both scalability studies and tracing-based investigations in single-node and multi-node configurations. To capture detailed execution traces, the Extrae tool was employed, and the resulting traces were analyzed using Paraver and Dimemas to extract relevant performance metrics and identify potential sources of inefficiency. The POP performance metrics framework was applied to evaluate the parallel efficiency and computation scalability of SeisSol on RISC-V architectures. A comparative analysis was also performed to better understand the performance of SeisSol on RISC-V architectures. This analysis was conducted in order to provide a basis for comparison with its performance on traditional x86_64 architectures.

In chapter 5, the performance of SeisSol was evaluated on three different RISC-V boards available in the HCA cluster at BSC. The scalability analysis revealed that RISC-V architectures are capable of delivering competitive performance for SeisSol. However, certain limitations were identified. In intra-node configurations, the performance was found to be constrained by a significant drop in instructions per cycle as the number of cores increased. This limitation can be attributed to various factors, including longer memory access times, cache contention, stall cycles, or bandwidth bottlenecks. These factors reduce the number of instructions that can be executed per cycle. Nevertheless, the set of PAPI counters available on HCA RISC-V platforms is limited, which complicates conducting a more in-depth analysis of these performance issues. In inter-node configurations, the performance was constrained by parallel overheads rather than computation. The analysis revealed that some boards exhibited high communication overheads, and others suffered from load imbalance as the number of nodes increased.

In chapter 6, the performance of SeisSol was evaluated on the Leonardo supercomputer, which is based on x86_64 architecture. The scalability analysis demonstrated that SeisSol

achieves high parallel efficiency on Leonardo, both in intra-node and inter-node configurations. The tracing-based performance investigation revealed that the application maintains a high level of computation scalability, with minimal parallel overheads. An interesting observation from the intra-node scalability analysis was the effect of DVFS features on performance. When scaling the number of cores, the frequency scaling mechanisms led to non-linear performance improvements. This behavior highlights the importance of considering Dynamic Frequency Scaling when analyzing performance on modern processors.

A comparison of SeisSol on RISC-V and x86_64 architectures revealed both similarities and clear performance differences. While both platforms were able to execute the application and achieve reasonable scalability, the x86_64 system consistently delivered higher parallel efficiency and better computational scalability. The evaluated RISC-V platforms exhibited limitations in instruction throughput and higher parallel overheads, which were largely negligible on x86_64. These results indicate that, although RISC-V shows potential for HPC workloads, further architectural and software optimizations are required to reach the performance levels of established systems. It should also be emphasized that the RISC-V platforms considered here are recent and primarily research-oriented, whereas the x86_64 system represents a mature, production-grade HPC architecture refined over decades. Therefore, part of the observed performance gap reflects differences in ecosystem maturity and system-level optimization rather than ISA characteristics alone.

Overall, this thesis provides insight into the performance characteristics of scientific workloads on RISC-V architectures and identifies several directions for further investigation. Future work should extend the analysis of SeisSol to additional communication strategies, such as employing a dedicated communication thread and integrating I/O phases, refining the study of tracing overhead on Milk-V Pioneer by limiting MPI caller depth, and conducting a systematic comparison between fully scalar and vectorized code to allow the evaluation of vectorization intensity and efficiency relative to the architectural vector width. Complementing this analysis with architectural simulators such as RAVE developed at BSC could provide deeper insight into microarchitectural behavior and optimization opportunities.

Additionally, evaluating a broader set of HPC applications on RISC-V architectures and comparing their behavior with x86_64 systems would provide a more comprehensive assessment of the suitability of RISC-V for HPC workloads.

Furthermore, extending the analysis of SeisSol to emerging RISC-V platforms with wider vector units and more mature microarchitectural features, such as the EPAC1.5 and EPAC2.0 processors developed within the European Processor Initiative, would help clarify the impact of vector support, memory hierarchy improvements, and architectural refinements. As both hardware and system software continue to mature, such evaluations will be essential to better understand the long-term potential of RISC-V in production HPC environments.

References

[1] F. Banchelli, "Methods and measurements for evaluating hpc systems," M.S. thesis, Universitat Politècnica de Catalunya, 2020.

[2] F. Banchelli, M. Garcia-Gasulla, and F. Mantovani, "Batched dgemms for scientific codes running on long vector architectures," in *International Conference on Parallel Processing and Applied Mathematics*, Springer, 2024, pp. 17–31.

[3] F. Banchelli, D. Jurado, M. Garcia-Gasulla, and F. Mantovani, "Exploring RISC-V long vector capabilities: A case study in Earth Sciences," *Future Generation Computer Systems*, vol. 174, p. 107 932, 2026.

[4] F. Banchelli et al., "Performance study of HPC applications on an Arm-based cluster using a generic efficiency model," pp. 167–174, 2020.

[5] F. Banchelli Gracia, "Evaluation and methods to increase efficiency of hpc systems with different maturity levels," Ph.D. dissertation, Universitat Politècnica de Catalunya, 2025.

[6] Barcelona Supercomputing Center, *BSC Performance Tools*, 2025.

[7] Barcelona Supercomputing Center, *Dimemas*, 2025.

[8] Barcelona Supercomputing Center, *Extrae*, version 4.3.1, 2025.

[9] Barcelona Supercomputing Center. "Hca cluster," Accessed: Sep. 11, 2025. [Online]. Available: `https://repo.hca.bsc.es/gitlab/epi-public/risc-v-software-development-vehicles/-/wikis/HCA-Nodes-and-Queues`.

[10] Barcelona Supercomputing Center, *Paraver*, 2025.

[11] Barcelona Supercomputing Center. "Barcelona Supercomputing Center - Centro Nacional de Supercomputación," Accessed: Aug. 11, 2025. [Online]. Available: `https://www.bsc.es/`.

[12] M. Baricevic, "Application-level energy profiling: The co.s.in.t. case study," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2014.

[13] M. Barnaba, "Exploring innovative approaches in industrial and academic hpc applications," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2020.

[14] A. Biagioni et al., "Red-sea: Network solution for exascale architectures," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 712–719.

[15] A. Breuer, A. Heinecke, and M. Bader, "Petascale local time stepping for the ader-dg finite element method," in *2016 IEEE international parallel and distributed processing symposium (IPDPS)*, IEEE, 2016, pp. 854–863.

[16] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel, and C. Pelties, in *International supercomputing conference*, Springer, 2014, pp. 1–18.

[17] C. Cavazzoni, "Eurora: A european architecture toward exascale," in *Proceedings of the Future HPC Systems: The Challenges of Power-Constrained Performance*, ser. FutureHPC '12, Venezia, Italy: Association for Computing Machinery, 2012.

[18] D. Cesarini et al., *D3.1 Application Analysis Report*, version 1.0, 2022.

[19] CHEESE-2P. "Centre of Excellence (CoE) for Exascale in Solid Earth," Accessed: Sep. 11, 2025. [Online]. Available: https://cheese2.eu/project/.

[20] CINECA. "Leonardo supercomputer," Accessed: Sep. 11, 2025. [Online]. Available: https://leonardo-supercomputer.cineca.eu/hpc-system/.

[21] M. Clascà, M. Garcia-Gasulla, A. Montagud, J. Carbonell Caballero, and A. Valencia, "Lessons learned from a performance analysis and optimization of a multiscale cellular simulation," Davos, Switzerland: Association for Computing Machinery, 2023.

[22] I. Corporation. "Intel xeon platinum 8358 processor," Accessed: Sep. 11, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/212282/intel-xeon-platinum-8358-processor-48m-cache-2-60-ghz/specifications.html.

[23] N. Corporation. "Introducing 200g hdr infiniband solutions," Accessed: Sep. 11, 2025. [Online]. Available: https://network.nvidia.com/files/doc-2020/wp-introducing-200g-hdr-infiniband-solutions.pdf.

[24] N. Corporation. "Nvidia a100 tensor core gpu," Accessed: Sep. 11, 2025. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf.

[25] R. Dorozhinskii and M. Bader, "Seissol on distributed multi-gpu systems: Cuda code generation for the modal discontinuous galerkin method," in *The International Conference on High Performance Computing in Asia-Pacific Region*, ser. HPCAsia '21, Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 69–82.

[26] M. Dumbser and M. Käser, "An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes—ii. the three-dimensional isotropic case," *Geophysical Journal International*, vol. 167, no. 1, pp. 319–336, 2006.

[27] A.-A. Gabriel et al., *Seissol*, version v1.3.2, Jun. 2025.

[28] R. Gachomba, "Comparative Performance Analysis of RISC-V Architectures," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2024.

[29] M. Garcia-Gasulla et al., "A Generic Performance Analysis Technique Applied to Different CFD Methods for HPC," *International Journal of Computational Fluid Dynamics*, vol. 34, no. 7-8, pp. 508–528, 2020.

[30] M. Garcia-Gasulla et al., "D10.5 Software and system integration specification and requirements," The EUPILOT Project, Deliverable D10.5 EUPILOT-2022-D10.5, version 2.0, 2022.

[31] P. Gorlani, "Fpga in hpc: High level synthesys of opencl kernels for molecular dynamics," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2016.

[32] S. R. Gupta, N. Papadopoulou, and M. Pericàs, "Challenges and Opportunities in the Co-design of Convolutions and RISC-V Vector Processors," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1550–1556.

[33] R. A. Harris et al., "A suite of exercises for verifying dynamic earthquake rupture codes," *Seismological Research Letters*, vol. 89, no. 3, pp. 1146–1162, 2018.

[34] A. Heinecke et al., "Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2014, pp. 3–14.

[35] HPC-Portal. "Centres of excellence," Accessed: Aug. 11, 2025. [Online]. Available: `https://hpc-portal.eu/coes`.

[36] HPC-Portal. "Coe cheese-2p," Accessed: Sep. 11, 2025. [Online]. Available: `https://hpc-portal.eu/coes/coe-cheese-2p`.

[37] HPC-Portal. "Coe pop3," Accessed: Aug. 11, 2025. [Online]. Available: `https://hpc-portal.eu/coes/coe-pop3`.

[38] K. Hwang and N. Jotwani, *Advanced computer architecture: parallelism, scalability, programmability*. McGraw-Hill New York, 1993, vol. 199.

[39] M. Käser and M. Dumbser, "An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes—i. the two-dimensional isotropic case with external source terms," *Geophysical Journal International*, vol. 166, no. 2, pp. 855–877, 2006.

[40] M. Katevenis et al., "Next generation of exascale-class systems: Exanest project and the status of its interconnect and storage development," *Microprocessors and Microsystems*, vol. 61, pp. 58–71, 2018.

[41] F. Mantovani et al., "Performance and energy consumption of hpc workloads on a cluster based on arm thunderx2 cpu," *Future Generation Computer Systems*, vol. 112, pp. 800–818, 2020.

[42] F. Mantovani et al., "Software Development Vehicles to enable extended and early co-design: a RISC-V and HPC case of study," in *International Conference on High Performance Computing*, Springer, 2023, pp. 526–537.

[43] A. Masini et al., "Cheese-2p deliverable: D3. 1 selected set of mini-apps and its requirements," 2024.

[44] Message Passing Interface Forum, *MPI: A message-passing interface standard version 5.0*, 2025.

[45] Milkv. "MilkV Pioneer RISC-V Development Platform," Accessed: Sep. 11, 2025. [Online]. Available: `https://milkv.io/docs/pioneer/overview/`.

[46] MPI Forum. "Message passing interface (mpi) standard," Accessed: Sep. 11, 2025. [Online]. Available: `https://www.mpi-forum.org/docs/`.

[47] OpenMP Architecture Review Board, *OpenMP application program interface version 4.0*, 2013.

[48] OpenMP Architecture Review Board. "OpenMP - the open standard for parallel programming," Accessed: Sep. 11, 2025. [Online]. Available: `https://www.openmp.org/`.

[49] F. Pacheco, "Performance evaluation of object detection in different architectures," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2022.

[50] R. Pattan and H. Khan, "Intel turbo boost technology," Intel Corporation, 2023.

[51] B. Pi. "Banana pi bpi-f3," Accessed: Sep. 11, 2025. [Online]. Available: https://docs.banana-pi.org/en/BPI-F3/BananaPi_BPI-F3/.

[52] POP-CoE. "How to create a pop performance audit," Accessed: Aug. 11, 2025. [Online]. Available: https://pop-coe.eu/sites/default/files/pop_files/whitepaperperformanceaudits.pdf.

[53] POP-CoE. "Pop standard metrics," Accessed: Aug. 11, 2025. [Online]. Available: https://pop-coe.eu/node/69.

[54] POP-CoE. "POP partner profile: The Barcelona Supercomputer Center Performance Tools Group," Accessed: Aug. 11, 2025. [Online]. Available: https://pop-coe.eu/blog/pop-partner-profile-the-barcelona-supercomputer-center-performance-tools-group.

[55] POP-CoE. "Pop hybrid metrics," Accessed: Aug. 11, 2025. [Online]. Available: https://pop-coe.eu/further-information/learning-material/pop-standard-hybrid-metrics-for-parallel-performance-analysis.

[56] POP-CoE. "Performance Optimisation and Productivity: A centre of excellence in HPC," Accessed: Aug. 11, 2025. [Online]. Available: https://pop-coe.eu/services.

[57] B. Project. "BZL - barcelona zettascale laboratory," Accessed: Sep. 11, 2025. [Online]. Available: https://www.bzl.es/en.

[58] M. Schweikhardt and A. Slota, "Empowering mixed-criticality industrial real-time computing with intel's dvfs evolution," Intel Corporation, 2024.

[59] SeisSol. "Seissol highlights," Accessed: Sep. 11, 2025. [Online]. Available: https://seissol.org/about/highlights/.

[60] SeisSol. "SeisSol History," Accessed: Sep. 11, 2025. [Online]. Available: https://seissol.readthedocs.io/en/latest/introduction.html#history.

[61] SeisSol. "Seissol performance measurement," Accessed: Sep. 11, 2025. [Online]. Available: https://seissol.readthedocs.io/en/latest/performance-measurement.html.

[62] SiFive. "Hifive premier p550," Accessed: Sep. 11, 2025. [Online]. Available: https://www.sifive.com/document-file/hifive-premier-p550-datasheet.

[63] S. D. Team. "Seissol examples repository," Accessed: Sep. 11, 2025. [Online]. Available: https://github.com/SeisSol/Examples.

[64] S. D. Team. "Seissol github repository," Accessed: Sep. 11, 2025. [Online]. Available: https://github.com/SeisSol/SeisSol.

[65] X. Teruel et al., "Applications Ported (full software-stack)," The MareNostrum Experimental Exascale Platform (MEEP) Project, Deliverable D5.3 MEEP-2022-D5.3, 2022.

[66] M. Turisini, G. Amati, and M. Cestari, "Leonardo: A pan-european pre-exascale supercomputer for hpc and ai applications," *arXiv preprint arXiv:2307.16885*, 2023.

[67] K. Tuteja, "Power efficiency of high energy physics applications on cpu and gpu," M.S. thesis, MHPC - Scuola Internazionale Superiore di Studi Avanzati (SISSA), 2022.

[68] C. Uphoff and M. Bader, "Yet another tensor toolbox for discontinuous galerkin methods and other applications," *ACM Transactions on Mathematical Software (TOMS)*, vol. 46, no. 4, pp. 1–40, 2020.

[69] C. Uphoff et al., "Extreme scale multi-physics simulations of the tsunamigenic 2004 sumatra megathrust earthquake," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2017, pp. 1–16.

[70] R. Vavrik, T. Panoc, M. Garcia-Gasulla, B. J. N. Wylie, and B. Mohr, "POP3: Advancing HPC Performance and Productivity," New York, NY, USA: Association for Computing Machinery, 2025, pp. 157–162.

[71] B. Wylie, "Seissol (commthread) performance assessment report," Jülich Supercomputing Centre, 2025.