Master in High Performance Computing

Performance and Energy Efficiency Analysis of RISC-V Architecture for Direct N-Body Simulations

Supervisor:
Mario Spera

Candidate:
Jenny Lynn Almerol

10^{th} edition
2023–2024

Abstract

With the increase in computational demands in high-energy physics, astrophysics, and gravitational wave studies, finding innovative computational solutions that balance performance and energy efficiency has become a critical issue. In this thesis, we investigate the applicability of RISC-V architectures for direct N-body simulations, a critical computational challenge in astrophysics.

We first benchmark the performance of a direct N-Body code on a dual-socket RISC-V Sophon SG2042 processor. The force evaluation kernel, implemented in mixed precision, is parallelized with MPI and OpenMP and is optimized with vector intrinsics (RVV). Comparative experiments are conducted on an AArch64 platform (NVIDIA Grace), with optimizations exploiting NEON vectorization, and on an x86 architecture (AMD EPYC 9554) exploiting AVX-512 vectorization optimizations, enabling a comprehensive cross-platform analysis. At their respective optimal configurations, RISC-V operates at approximately 4–5% of x86 throughput and ∼10% of AArch64 throughput, resulting in a 15.67× slowdown relative to x86 and 8.86× relative to AArch64. Despite the SG2042's lower thermal design power (120 W), the longer execution time means it consumes 4.79× more total energy than x86 and 2.87× more than AArch64, yielding an Energy-Delay Product 25.4× worse than AArch64 and 75.0× worse than x86.

We then port the N-Body force kernel to the RISC-V-based Tenstorrent Wormhole n300 accelerator using the TT-Metalium programming interface, to the best of our knowledge the first astrophysical application to leverage this class of hardware. We evaluate single-device performance and energy efficiency against a highly optimized AMD EPYC 9124 CPU baseline parallelized with OpenMP and optimized with AVX-512 intrinsics, demonstrating a 2.23× speedup and 1.80× energy savings. We further investigate three strategies for scaling the code across multiple Wormhole cards and chips using MPI, analyzing their scalability and energy-delay characteristics.

The results demonstrate significant differences in computational and energy efficiency across platforms, emphasizing the role of architectural and software-level optimizations in achieving energy-efficient scientific computing. Our findings provide valuable insights into the potential of RISC-V systems for scalable, low-energy HPC applications in astrophysics and beyond.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Mario Spera (SISSA), and to Daniele Gregori, and Elisabetta Boella (E4 Computer Engineering), for their invaluable guidance, support, and patience throughout this research. Their expertise and insights were instrumental in shaping this work.

I am deeply grateful to E4 Computer Engineering (Scandiano, Italy) for providing access to the computational resources, including the Tenstorrent Wormhole `n300` accelerators and the RISC-V server, that made this research possible. Special thanks to F. Proverbio, A. D'Apice, and F. Magugliani at E4 for their technical support and valuable comments on the publication manuscripts. I also thank E. Duffy, R. Friedman, R. Ganisetti, and F. LeClair at Tenstorrent for their guidance on the TT-Metalium programming interface.

I am also grateful to the Master in High Performance Computing (MHPC) program at SISSA and ICTP for the stimulating academic environment and for the scholarship provided by ICTP that supported my studies. These opportunities have been transformative.

This research is supported by the Italian Research Center on High Performance Computing, Big Data and Quantum Computing (ICSC), project funded by the European Union – NextGenerationEU – and the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, within the activities of Spoke 3 (Astrophysics and Cosmos Observations).

To my colleagues and fellow students, thank you for the fruitful discussions and camaraderie that made this journey enjoyable. Finally, I extend my deepest appreciation to my family for their unwavering support and encouragement throughout my studies.

Publications

The following publications arose from the work presented in this thesis:

- Jenny Lynn Almerol, Elisabetta Boella, Mario Spera, and Daniele Gregori. Accelerating gravitational N-body simulations using the RISC-V-based Tenstorrent Wormhole. In *SC25-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025. arXiv:2509.19294.

- Jenny Lynn Almerol, Elisabetta Boella, Mario Spera, and Daniele Gregori. Assessing performance and porting strategies for gravitational N-body simulations on the RISC-V-based Tenstorrent Wormhole. *Astronomy & Computing*, 2025. Manuscript submitted for publication, December 2026.

Contents

Chapter 1

Introduction

1.1 Background

The trajectory of modern scientific discovery is increasingly constrained by the availability, scalability, and sustainability of computational resources. Across a wide range of disciplines, including bioinformatics, climate modeling, large-scale machine learning, and fundamental physics, there is a persistent demand for ever-increasing processing throughput. However, this exponential growth in computational capability has been accompanied by an unsustainable increase in energy consumption, with significant implications for the operational cost and carbon footprint of high-performance computing (HPC) systems.

In response to these challenges, new architectures offering improved energy efficiency have emerged. Platforms such as RISC-V and AArch64 present viable alternatives to traditional x86 architectures, providing advantages in terms of architectural simplicity, scalability, and reduced instruction sets complexity [1, 2, 3]. In particular, RISC-V ISA has attracted considerable attention in recent years owing to its open-source and royalty-free nature, designed to be modular and extensible, enabling the integration of customized instructions and extensions tailored to specific application domains. This interest has been further fueled by the rapid growth of artificial intelligence (AI) and machine learning (ML), whose computational demands strongly overlap with those of traditional scientific workloads [4, 5].

A major obstacle to the adoption of emerging architectures in scientific computing is the scale and maturity of existing software ecosystems. Large scientific applications typically comprise millions of lines of code, and their porting to new architectures is justified only if substantial performance gains or energy efficiency can be demonstrated. With the emergence of the first generations of RISC-V–based processors targeting HPC systems and RISC-V-based accelerators, the scalability of scientific software development with respect to both power efficiency and energy consumption has become a critical issue.

RISC-V architectures are of particular interest for fundamental research, especially in next-generation astrophysical experiments that demand extreme computational and energy resources. One significant application is the simulation of gravitational dynamics and the evolution of dense stellar systems, such as

star clusters, which play a key role in the formation of compact objects and gravitational-wave sources. Accurate direct N-body simulations of these systems are computationally demanding, exhibiting a computational complexity that scales as $O(N^2)$. Direct N-Body simulations of larger systems would require excessive computational time and energy on existing GPU-based clusters. Therefore, innovative hardware–software co-design approaches are required in order to enable simulations of larger systems and to efficiently explore increasingly complex parameter spaces.

1.2 Problem Statement

Despite the theoretical promise of the RISC-V ISA as an energy-efficient and customizable alternative to high-performance computing (HPC), its practical efficacy for large-scale scientific workloads, such as direct N-body simulations, remains largely insufficiently assessed. Currently, x86 and AArch64 architectures dominate the HPC field due to their mature software ecosystems, yet their increasing energy demands represent a significant barrier for the next generation of astrophysical simulations.

The primary problem is that the *energy-to-solution* advantages of RISC-V have not yet been demonstrated for the $\mathrm{O(N^2)}$ complexity inherent in direct N-Body simulations. Specifically, the following open questions motivate this work:

- **Software Maturity and Portability:** It is unclear whether scientific kernels can be efficiently ported to emerging hardware such as the Sophon SG2042 or the Tenstorrent Wormhole without substantial performance degradation, given the relative immaturity of RISC-V toolchains and programming interfaces.
- **Performance Gap:** There is a critical need to quantify the performance gap between RISC-V and established x86/AArch64 platforms for $O(N^2)$ workloads and to identify the architectural and software-level constraints that drive it, particularly memory bandwidth limitations and vectorization support.
- **Energy Efficiency:** Despite the lower thermal design power of RISC-V processors, it is unknown whether reduced power translates into lower energy-to-solution when execution times are substantially longer. Similarly, it is unclear whether RISC-V-based accelerators can deliver meaningful energy savings over optimized CPU implementations.
- **Scalability:** The scalability of RISC-V platforms, both CPU and accelerator, for scientific workloads across multiple cores, chips, and cards remains unexplored.

Without a rigorous cross-architectural evaluation, it remains unknown whether RISC-V can provide the computational throughput and energy efficiency required for the high-resolution simulations demanded by next-generation gravitational wave observatories such as the Einstein Telescope [6].

1.3 Objectives

The central goal of this thesis is to perform a cross-architectural evaluation of RISC-V hardware for scientific computing, spanning both RISC-V CPUs and RISC-V-based accelerators. To achieve this, we define the following objectives:

1. **Code Portability and Optimization:** To port a C++/MPI+OpenMP direct N-body simulation kernel to the RISC-V ISA, implementing architecture-specific optimizations for the Sophon SG2042 (exploiting the RVV vector extension), AArch64 platforms (exploiting ARM NEON intrinsics), and x86 platforms (exploiting AVX-512 intrinsics) and to map the force evaluation workload to the Tenstorrent Wormhole's Tensix cores using the TT-Metalium programming interface.

2. **Performance Characterization:** To benchmark execution throughput in terms of time-to-solution, and MPI and OpenMP configurations on multiple hardware platforms.

3. **Comparative Benchmarking:** To conduct a comparison against traditional x86 (AMD EPYC 9554) and AArch64 (NVIDIA Grace) platforms to quantify the current performance gap, and to compare the Wormhole accelerator against a highly optimized AMD EPYC 9124 CPU baseline.

4. **Energy Efficiency Analysis:** To quantify energy-to-solution by measuring real-time power consumption of CPUs and accelerators during simulations, identifying the most sustainable hardware configuration for long-term astrophysical deployments.

5. **Multi-device Scalability:** To evaluate three strategies for scaling the N-body code across multiple Wormhole cards and chips using MPI, and to identify the configuration that provides the best trade-off between execution time and energy efficiency.

1.4 Relevance

This study is highly relevant to both the computer science and astrophysics communities for several reasons. First, it provides some of the first independent benchmarks for the Tenstorrent Wormhole in a non-AI, purely scientific context, testing the versatility of spatial RISC-V accelerators. To the best of our knowledge, this work presents the first astrophysical N-body application ported to Tenstorrent hardware, and the first scientific code capable of running across multiple Tenstorrent cards and chips simultaneously. Second, it contributes to the "Sovereign Hardware" movement by evaluating royalty-free ISAs that can reduce the dependency of research institutions on proprietary silicon.

Third, the work provides a comprehensive cross-architectural perspective spanning RISC-V CPUs (Sophon SG2042), AArch64 CPUs (NVIDIA Grace), x86

CPUs (AMD EPYC), and RISC-V accelerators (Tenstorrent Wormhole), yielding actionable guidance for HPC practitioners evaluating emerging hardware for scientific workloads.

Finally, this work explores a pathway toward sustainable computational astrophysics. If RISC-V architectures can achieve competitive energy-to-solution ratios, they could significantly lower the operational barriers for simulating dense stellar systems, such as Globular Clusters and Galactic Nuclei, which are fundamental to understanding gravitational wave sources in the era of LISA and the Einstein Telescope.

Chapter 2

Background and related work

2.1 The Direct N-Body Simulation

Direct gravitational N-body simulations numerically solve the equations of motion for a system of N particles under the influence of their mutual gravitational forces, expressed as:

$$\mathbf{F}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i), \tag{2.1}$$

where m_i and m_j are the particle masses, $\mathbf{r}_i$ and $\mathbf{r}_j$ their position vectors, $r_{ij} = |\mathbf{r}_j - \mathbf{r}_i|$ is the inter-particle distance, and G the gravitational constant. It is the most straighforward as it employs a direct summation of the force contributions of all the other particles. It is also the most accurate, as it does not rely on approximations [7], which are essential for accurately studying the evolution of dense stellar systems, such as star clusters. These systems are considered the primary environments for the formation of compact object binaries, such as black hole binaries, whose coalescence is detectable by gravitational wave interferometers (LIGO-Virgo-KAGRA [8], and the future Einstein Telescope[6]). The development of *accurate*, *efficient*, and *scalable* direct N-body codes is crucial for the astrophysical interpretation of gravitational waves, particularly in light of future experiments that will detect millions of sources.

In this work, we employ a high-order time integration scheme, the sixth-order Hermite integrator [9]. The Hermite scheme comprises three iterative stages: prediction, evaluation, and correction [10]. In the prediction step, the positions, velocities, and accelerations of all particles are estimated from their previously known values. In the evaluation step, accelerations and their first time derivatives (jerks) are computed using these predicted quantities. Finally, in the correction step, the predicted positions and velocities are refined using the newly evaluated accelerations and jerks, achieving sixth-order time integration accuracy. A small softening parameter, must be included in the distance computation to avoid numerical singularities when particles are close.

2.2 The Sophon SG2042

The Sophon SG2042 CPU is a publicly available server-class RISC-V processor designed for high-performance computing workloads. According to publicly available specifications, it is a 64-core processor operating at up to 2 GHz, organized into 16 clusters of four RISC-V cores each [11]. Each core provides private L1 instruction and data caches of 64 KB, while each cluster shares a 1 MB L2 cache. The processor further integrates a shared last-level (L3) system cache with a total capacity of 64 MB. Memory access is managed by four DDR4-3200 memory controllers, which interface the entire 64-core processor to main memory [11, 12]. This relatively modest number of memory channels, four controllers serving 64 cores, places a hard ceiling on aggregate memory bandwidth and constitutes a key architectural constraint for memory-intensive parallel workloads.

2.2.1 RISC-V ISA Extensions

Each core on the experimental platform implements the RV64I base instruction set with several standard extensions, indicated by the ISA string `RV64IMAFDCV`. These extensions are defined as follows:

- **I** – Base integer instructions (loads, stores, arithmetic, control flow).
- **M** – Integer multiply and divide instructions.
- **A** – Atomic instructions for multi-threaded synchronization.
- **F** – Single-precision floating-point instructions.
- **D** – Double-precision floating-point instructions.
- **C** – Compressed instructions, reducing code size and improving instruction cache efficiency.
- **V** – Vector extension (RVV), which provides hardware support for vector operations on multiple data elements in parallel.

The presence of these extensions determines the available instruction-level operations for both scalar and vectorized code, and is crucial for reproducing the performance behavior of the system.

2.2.2 Vector Extension

The RISC-V Vector Extension (RVV) exposes vector registers with a fixed width called **VLEN**. On the Sophon SG2042, VLEN is 128 bits, meaning that each vector register can hold up to 128 bits of data. The RVV instruction set also allows grouping multiple vector registers together using the **LMUL** parameter, which in this work is set to 8. Combined, VLEN and LMUL determine the maximum number of elements that can be processed simultaneously in a single vector instruction, often referred to as the *vector length* for a given data type.

For example, with LMUL=8, a single RVV instruction can process up to 32 FP32 elements or 16 FP64 elements at once. This directly affects the throughput of vectorized loops, as more data elements can be operated on in parallel per instruction. Understanding the VLEN and LMUL configuration is crucial for tuning data layouts, memory alignment, and loop unrolling to achieve maximum utilization of the vector hardware.

In practice, the vector length interacts with the number of hardware threads (harts) and cache hierarchy to determine overall performance. Efficient vectorization requires balancing vector usage to avoid register spilling while keeping the pipelines fully utilized. In this work, all vectorized kernels are compiled and tuned to match the hardware VLEN and LMUL settings of the SG2042, ensuring reproducible and high-performance execution.

2.3 The Tenstorrent Wormhole™ Accelerator

The Tenstorrent Wormhole n300 accelerator is a RISC-V-based processor designed primarily for Artificial Intelligence (AI) and Machine Learning (ML) workloads. It targets large-scale, dataflow-oriented computation driven by the recent rapid growth of foundation models and distributed training. The n300 card integrates two identical Wormhole™ ASICs, each operating at up to 160 W [13].

Each Wormhole ASIC is composed of an array of programmable tiles interconnected by a high-bandwidth Network-on-Chip (NoC). The primary compute elements are Tensix cores, which combine matrix and vector compute units with local SRAM and multiple lightweight Baby RISC-V control processors. This tightly coupled compute–memory organization enables efficient fine-grained data movement and minimizes off-chip memory traffic.

Each Wormhole ASIC connects to 12 GB of external GDDR6 memory via a 192-bit memory bus [13]. One of the two ASICs connects directly to the host system via a PCIe 4.0 ×16 interface. The second ASIC does not have a direct PCIe connection; instead, it communicates with the first ASIC through a high-speed on-board Ethernet-based interconnect. As a result, host-to-device communication with the second ASIC is routed internally through the first ASIC.

The Wormhole ASIC can be described as a two-dimensional grid of heterogeneous tiles, as summarized in Table 2.3.1 and illustrated in Figure 2.3.1. Each tile connects to its neighboring tiles via the NoC, which provides four directional links (north, south, east, and west), enabling packet-based communication between tiles across the two-dimensional mesh. These links are bidirectional and support independent traffic flows, allowing concurrent data movement and synchronization between compute, memory, and I/O tiles.

The E tiles serve as Ethernet interface tiles, each supporting up to 100 Gb/s of bandwidth per direction. The n300 card additionally features two QSFP-DD ports capable of bidirectional data transfer at up to 200 Gbps for external high-speed networking [13, 14]. The Ethernet tiles are used both for inter-card scaling and for the internal Ethernet-based link between the two Wormhole ASICs on the n300 accelerator.

The NoC provides a scalable communication substrate across the Wormhole

ASIC. Through NoC transactions, any tile may initiate read or write operations to memory or registers located on remote tiles, supporting distributed execution and pipeline parallelism. In contrast to conventional cache-coherent systems, communication is explicitly orchestrated by software, granting programmers fine control over data placement and movement.

The D tiles act as bridges between the NoC and off-chip GDDR6 memory, while the T tiles correspond to Tensix compute cores. Figure 2.3.1 illustrates the spatial arrangement of these tile types within the Wormhole ASIC.

Table 2.3.1: Tile types and composition of the accelerator fabric.

Tile	**Count**	**Contents (per tile)**
A	1	1× Argonaut RISC core; system control and management (connected to PCIe)
D	6×3	Bridge to GDDR6 memory; each tile group interfaces with 2× 1 GB GDDR6 devices
E	16	1× Baby RISC-V CPU (E variant); 256 KB SRAM; optional bridge to 100 Gb Ethernet
P	1	PCIe 4.0 ×16 host interface
T	64	5× Baby RISC-V CPUs (B/T/T/T/NC variants); 1.5 MB SRAM; 1× matrix unit (2048 multipliers, 5 b × 7 b); 1× vector/SIMD unit (32 lanes, 32 b each)

2.3.1 The Tensix Core

The Tensix core constitutes the primary computational building block of the Wormhole accelerator. It is explicitly designed to sustain high throughput for matrix- and vector-oriented workloads typical of modern deep learning models, while maintaining fine-grained programmability and efficient data movement. Each Tensix core integrates compute, control, memory, and communication resources within a single tile.

A Tensix core comprises five embedded Baby RISC-V processors, a tensor-focused floating-point unit (FPU) optimized for low-precision matrix arithmetic, a wide SIMD engine referred to as the scalar floating-point unit (SFPU) for general-purpose vector operations, 1.5 MB of on-tile SRAM acting as L1 memory, and two NoC router interfaces that connect the core to the chip-wide Networks-on-Chip (NoCs) [16, 14].

The Baby RISC-V processors embedded within each Tensix core are lightweight 32-bit, in-order, single-issue cores optimized for power and area efficiency. These processors operate at approximately 1 GHz and are primarily responsible for coordinating data movement and issuing instructions to the Tensix coprocessor units, rather than performing heavy arithmetic themselves [14].

As illustrated in Figure 2.3.2, the five Baby RISC-V cores are functionally partitioned into two data movement cores (RISC-V NC and RISC-V B) and

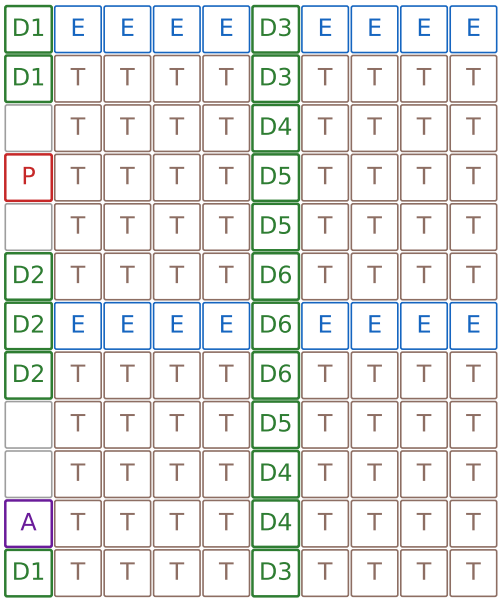

Figure 2.3.1: **Logical tile map of the Tenstorrent Wormhole ASIC n150.** The chip is organized as a 10 × 12 grid of heterogeneous functional tiles interconnected via a 2D torus Network-on-Chip (NoC). Each tile communicates with its four neighbors through dedicated 32-byte-wide bidirectional channels forming two logical networks (NoC #0 and NoC #1).

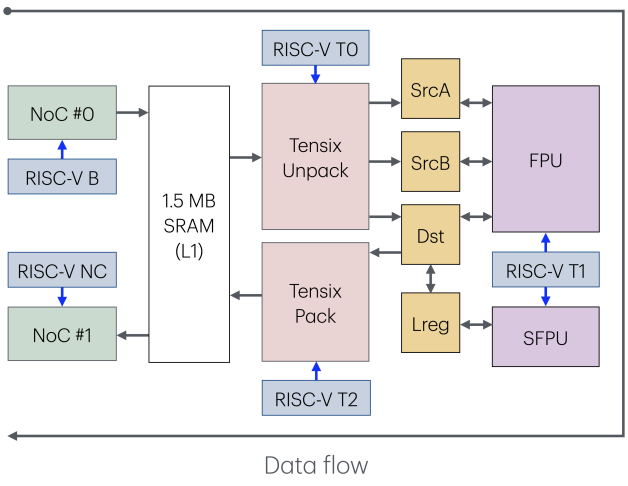

Figure 2.3.2: A simplified schematic of a Tensix core within the Tenstorrent Wormhole AI accelerator. Blue arrows represent instruction dispatch, while black arrows denote data movement. Adapted from Chang (2025)[15].

three compute coordination cores (RISC-V T0, T1, and T2). The data movement cores orchestrate transfers between the Tensix core, the NoC, and off-chip DRAM, enabling asynchronous prefetching and write-back of data. The compute coordination cores manage the execution of arithmetic and logical operations by dispatching instructions to the tensor FPU, SFPU, and associated datapath components.

Within the compute pipeline, the three compute coordination cores assume specialized roles. RISC-V T0 (UNPACK) controls the unpacker stage, issuing instructions that load data from SRAM into two source registers, `srcA` and `srcB`. Each source register has a capacity of 4 KiB, sufficient to hold up to 1024 single-precision floating-point values. RISC-V T1 (MATH) governs the main arithmetic datapath, dispatching operations to the tensor FPU, SFPU, and control logic (ThCon) to process the contents of `srcA` and `srcB`. Finally, RISC-V T2 (PACK) manages the packer stage, directing results from the 32 KiB destination register `dst`, organized into 16 segments, back into the local SRAM [17].

This hardware-level decomposition is directly exposed in the Tenstorrent software stack. In the TT-Metalium SDK, data movement cores execute dedicated data movement kernels, while compute kernels are structured around the UNPACK, MATH, and PACK phases, mirroring the underlying Tensix execution pipeline [18]. This close alignment between hardware and software enables explicit control over dataflow and contributes to the high efficiency of Tensix cores for large-scale AI workloads.

From a performance perspective, the explicit decomposition of execution into UNPACK, MATH, and PACK stages is central to the Tensix core's ability to achieve high sustained throughput while tolerating memory and communication latency. By decoupling data movement, computation, and result writeback across separate coordination cores, the Tensix architecture enables software-managed pipelining, where data transfers and arithmetic operations can overlap in time. This staged execution model effectively transforms each Tensix core into a deeply pipelined dataflow engine, allowing compute units such as the tensor FPU and SFPU to remain busy even in the presence of NoC or DRAM access latency.

2.3.2 The TT-Metalium Programming Model

TT-Metalium is Tenstorrent's low-level programming interface that exposes explicit control over the Wormhole hardware, enabling the construction of highly parallel and fine-grained execution pipelines. Rather than abstracting hardware details behind opaque runtime mechanisms, TT-Metalium provides direct access to device resources such as Tensix cores, NoC communication, and on-chip memory, allowing software to closely follow the underlying execution and dataflow model of the accelerator [19, 20].

A typical TT-Metalium application follows a structured workflow. Execution begins with device initialization and configuration, during which communication channels between the host and the accelerator are established. Compute kernels are then compiled and loaded onto the device, with execution parameters specified explicitly. Memory buffers are allocated in device memory, followed by

asynchronous data transfers between the host and the device to prepare input data. Finally, kernels are scheduled for execution through a command queue, which manages kernel dispatch, synchronization, and ordering across the hardware fabric [19, 20].

Applications are commonly decomposed into three distinct kernel types—read, compute, and write—corresponding to successive stages of a dataflow pipeline. Read kernels execute primarily on data movement cores and are responsible for fetching data from off-chip DRAM or remote tiles. Compute kernels execute on Tensix compute cores and perform arithmetic operations using the tensor FPU and SFPU units. Write kernels handle the transfer of results back to memory or onward to subsequent pipeline stages. These kernels communicate via software-managed circular buffers (CBs), enabling asynchronous producer–consumer interactions and allowing computation and communication to overlap in time.

Data movement across the memory hierarchy is explicitly orchestrated by the programmer. Input and output buffers are allocated in off-chip DRAM, and data is transferred asynchronously from the host to the device using non-blocking command queue operations such as `EnqueueWriteBuffer`. These transfers can proceed concurrently with device-side execution, allowing host–device communication to be overlapped with computation. Read kernels subsequently consume input data directly from DRAM and stream it into the on-chip execution pipeline.

To enable efficient coordination between pipeline stages on the device, additional buffers are allocated in on-chip SRAM in the form of circular buffers. These SRAM-resident buffers serve as staging areas between the read, compute, and write kernels, decoupling off-chip memory accesses from on-chip computation. Circular buffers support fine-grained producer–consumer synchronization and play a key role in amortizing DRAM and NoC latency, thereby sustaining high utilization of the Tensix compute units.

Upon completion of computation, write kernels store results back into DRAM-resident buffers. The host can retrieve these results using non-blocking device-to-host transfers, such as `EnqueueReadBuffer`, allowing result collection to overlap with subsequent kernel execution. This explicit separation between DRAM-resident buffers and on-chip circular buffers reflects TT-Metalium's software-managed memory hierarchy and provides deterministic control over data placement and movement throughout the execution pipeline.

TT-Metalium further optimizes data movement through support for tilized tensor layouts, in which tensors are partitioned into contiguous 32×32 tiles. This tilization aligns naturally with the Tensix execution units and enables efficient, high-bandwidth transfers across DRAM, the NoC, and Ethernet links, reducing memory access overheads and improving pipeline efficiency [5, 21, 20].

Synchronization between pipeline stages is enforced using TT-Metalium's circular buffer control primitives. Consumer-side synchronization is managed using `cb_wait_front` and `cb_pop_front`, which ensure that a kernel blocks until sufficient data is available and consumes buffer entries in program order. On the producer side, `cb_reserve_back` prevents a kernel from writing into a buffer until adequate space is available, thereby enforcing back-pressure and preventing overwriting of unconsumed data. After producing new data, the kernel in-

vokes `cb_push_back` to commit the entry and advance the buffer state. Together, these primitives provide deterministic producer–consumer coordination, preserve data dependencies, and prevent race conditions in highly parallel execution pipelines [5, 21, 20].

It is worth noting that the TT-Metalium programming model has evolved across software releases. Earlier versions, such as v0.60.1—used in the single-device campaign [22]—employ a tile-centric programming model in which kernels are explicitly bound to individual Tensix cores and communication paths are manually orchestrated by the programmer. While this approach offers fine-grained control over execution and data movement, it requires detailed reasoning about tile placement, NoC communication, and synchronization.

Release v0.62.2, adopted for the multi-device campaign [23], introduced a mesh-based programming abstraction that elevates the model from individual tiles to logical meshes of Tensix cores. In this approach, kernels are expressed over multi-tile meshes, and the runtime assists in mapping computation and communication patterns onto the underlying hardware fabric. This abstraction simplifies the expression of large-scale parallelism while preserving the explicit dataflow execution semantics and circular-buffer-based synchronization model of earlier versions.

2.4 State of the Art

The use of RISC-V in scientific high-performance computing is an emerging research area that has gained momentum in recent years, driven both by the maturation of RISC-V silicon and by growing interest from the HPC community.

On the CPU side, the Monte Cimone cluster [24] was among the first RISC-V HPC deployments in Europe, demonstrating the feasibility of running scientific workloads on RISC-V hardware. Its successor, Monte Cimone v2, extended this effort with improved processor designs and software stack maturity [25]. Studies comparing RISC-V and AArch64 system-on-chip designs for HPC workloads have found that, while RISC-V platforms currently lag behind AArch64 in raw throughput due to less mature toolchains and microarchitectural features, they offer a compelling open and customizable foundation for future optimization [1]. Mahale et al. [26] investigated optimizations for sparse and long vector operations on RISC-V vector processing units, highlighting both the promise and current limitations of the RVV extension for irregular scientific workloads.

On the accelerator side, Tenstorrent hardware has recently attracted attention from the HPC community. Brown and Barton [4] demonstrated the acceleration of stencil computations on the Tenstorrent Grayskull RISC-V accelerator, an earlier generation device, achieving significant speedups over CPU baselines for structured grid operations. Brown et al. [5] subsequently explored Fast Fourier Transform implementations on the Tenstorrent Wormhole, finding that the device's high-bandwidth memory and tile-based dataflow model are well-suited to highly regular, bandwidth-intensive computations.

To the best of our knowledge, no prior work has evaluated RISC-V CPUs or accelerators for direct gravitational N-body simulations before the contributions

presented in this thesis. This work therefore fills a critical gap by providing both a systematic cross-platform evaluation on RISC-V CPUs and the first porting of an astrophysical N-body code to a RISC-V-based spatial accelerator.

Chapter 3

Methods

This chapter describes the N-body simulation code, the hardware platforms used for benchmarking, the porting strategy to the Tenstorrent Wormhole accelerator, and the procedures adopted for performance and energy measurements.

3.1 The N-Body Simulation Code

The direct N-Body simulation code used in this work is implemented in C++ and parallelized using a combination of MPI and OpenMP. The code follows the sixth-order Hermite integration scheme described in Section 2.1, comprising three iterative stages per time step: prediction, evaluation, and correction. The force evaluation step, which computes pairwise gravitational accelerations and jerks for all particles, constitutes the dominant computational bottleneck, scaling as $\mathcal{O}(N^2)$.

All CPU implementations of the force evaluation kernel employ architecture-specific intrinsics to maximize vectorization performance: RVV intrinsics on the Sophon SG2042, ARM NEON intrinsics on the NVIDIA Grace, and AVX-512 intrinsics on x86 platforms. To evaluate the impact of memory access patterns on vectorization efficiency, the code was implemented using three distinct data layouts: Arrays (ARR), where each scalar quantity (r_x, r_y, r_z, v_x, v_y, or v_z) is stored in a separate, independent array; Structure of Arrays (SOA), which groups these arrays into a single logical structure to facilitate unit-stride SIMD loads; and Array of Structures (AOS), where all physical quantities for a single particle are encapsulated within a struct and stored contiguously.

In all experimental configurations, the product of MPI processes and OpenMP threads was mapped to the available physical core count of the respective architecture. The use of `OMP_PLACES=cores` and `OMP_PROC_BIND=close` ensures hardware-enforced thread locality by pinning OpenMP threads to specific physical execution units. Consequently, any execution configuration exceeding the physical core count resulted in an oversubscribed state. In this regime, the operating system must perform frequent context switching to manage the excess threads on a single core.

The code supports two numerical precision modes. In the full double-precision (FP64) mode, all computations are performed in 64-bit floating point. In the

mixed-precision mode, used for all cross-platform benchmarking experiments reported in this thesis, the force evaluation kernel employs single precision (FP32) for intermediate quantities such as displacements, distances, accelerations, and jerks, while the remaining steps (prediction and correction) retain FP64 precision. The correctness of the mixed-precision results is verified by confirming that computed accelerations and jerks deviate by no more than 0.05% and 0.2%, respectively, relative to a full FP64 reference implementation. These tolerances represent worst-case deviations observed across all tested particle configurations (up to $N = 102{,}400$ particles) and over all simulated timesteps. Previous studies have shown that mixed-precision schemes offer significant improvements in performance and energy efficiency while maintaining sufficient numerical accuracy for this class of simulations [27, 28].

A small softening parameter $\epsilon = 1.0 \times 10^{-7}$ is included in the inter-particle distance computation to prevent numerical singularities when particles approach each other closely.

3.2 Experimental Platforms

3.2.1 RISC-V: Sophon SG2042

The RISC-V benchmarks were conducted on a dual-socket server equipped with Sophon SG2042 processors, running Fedora Linux 38. Each socket hosts one 64-core SG2042 processor, yielding 128 hardware threads (harts) in total. The system is configured as an eight-node NUMA platform. Each hart implements the `RV64IMAFDCV` ISA and supports the Sv39 virtual memory scheme.

Applications were compiled using a RISC-V GNU cross-compilation toolchain rather than a native compiler, because the native toolchain available on the platform did not provide sufficient support for generating code exploiting the RISC-V Vector (RVV) extension. The relevant compilation flags were `-std=c++17 -march=rv64imafdc_v0p7_zfh_zvamo0p7_zvlsseg0p7_xtheadc -mabi=lp64d -O3 -D_RVV32v1`, with explicit tuning for the VLEN=128, LMUL=8 vector configuration of the SG2042. For OpenMP thread binding, the environment variables `OMP_PROC_BIND=true` and `OMP_PLACES=cores` were set. The `-bind-to numa` option was employed to exploit NUMA locality. The force evaluation was distributed across MPI processes and OpenMP threads; the thread count and process count were varied to study strong scaling behavior and sensitivity to parallelization strategy.

3.2.2 AArch64: NVIDIA Grace

Comparative experiments on the AArch64 architecture were conducted on an NVIDIA Grace CPU, a 72-core processor operating at up to 3.5 GHz. The system runs a Linux kernel (version 6.2.0-1008-nvidia-64k) on Ubuntu with an AArch64 GNU/Linux environment. Each core features L1 instruction and data caches of 64 KB each (128 KB combined per core), a private L2 cache of 1 MB per core (72 MB in aggregate), and a shared L3 cache of 114 MB. All 72 cores reside within

a single NUMA domain, which simplifies NUMA-aware binding relative to the multi-socket SG2042 configuration.

The N-Body code was compiled using the NVIDIA HPC SDK (`nvidia/nvhpc-hpcx 23.11`) with flags `-O3 -fopenmp -march=native`. Thread binding was controlled via `OMP_PLACES=cores` and `OMP_PROC_BIND=close`. In addition to the baseline OpenMP+MPI implementation, we implemented an architecture-specific version of the force evaluation kernel using ARM NEON intrinsics [3], enabling explicit single-instruction multiple-data (SIMD) vectorization. For FP32, NEON provides 128-bit vector registers capable of processing four single-precision values simultaneously. NEON was chosen over SVE to ensure portability and simplicity, as it provides predictable 128-bit SIMD behavior that is sufficient for this workload.

3.2.3 x86: AMD EPYC 9554 and AMD EPYC 9124

The x86 reference platform used for the cross-platform comparison (RISC-V vs. AArch64 vs. x86) was equipped with AMD EPYC 9554 processors arranged in a dual-socket configuration, with each socket forming a separate NUMA domain. For the Tenstorrent Wormhole benchmarks, the host system was equipped with dual-socket AMD EPYC 9124 processors (two NUMA domains), offering 64 hardware threads (2 sockets $\times$ 16 cores $\times$ 2 threads per core) and a maximum clock frequency of 3.71 GHz, with 1.5 TB of DDR5 RAM, running Ubuntu 24.04.2 LTS with Linux kernel 6.8.0.

The baseline x86 CPU implementation for the cross-platform benchmarks is compiled using the NVIDIA HPC SDK (`nvidia/nvhpc 23.9`) with flags `-O3 -fopenmp -march=native -mavx512f`, thread binding via `OMP_PROC_BIND=true` and `OMP_PLACES=cores`, and optimized with AVX-512 intrinsics for the force evaluation kernel. The reference CPU implementation for the Wormhole experiments exploits AVX-512 intrinsics and was compiled with GCC 13.3.0 (C++20 standard, flags `-O3 -fopenmp -march=native -mavx512f`), with distributed parallelism via Open MPI 4.0.6. The Wormhole-accelerated version used the same compiler and standard but with Open MPI 5.0.7-ULFM—required for the fault-tolerant MPI layer used by `tt-run`—and the vectorization target `-march=x86-64-v3`, consistent with the ISA features of the AMD EPYC 9124 host.

3.2.4 Tenstorrent Wormhole n300

The Tenstorrent Wormhole `n300` accelerator is described in detail in Section 2.3. Four `n300` cards are connected to the host via PCIe Gen 4, though experiments were initially conducted using a single card, and subsequently scaled up to four cards. Each `n300` card hosts two Wormhole ASICs (L-chip, connected via PCIe, and R-chip, connected to the L-chip via on-board Ethernet).

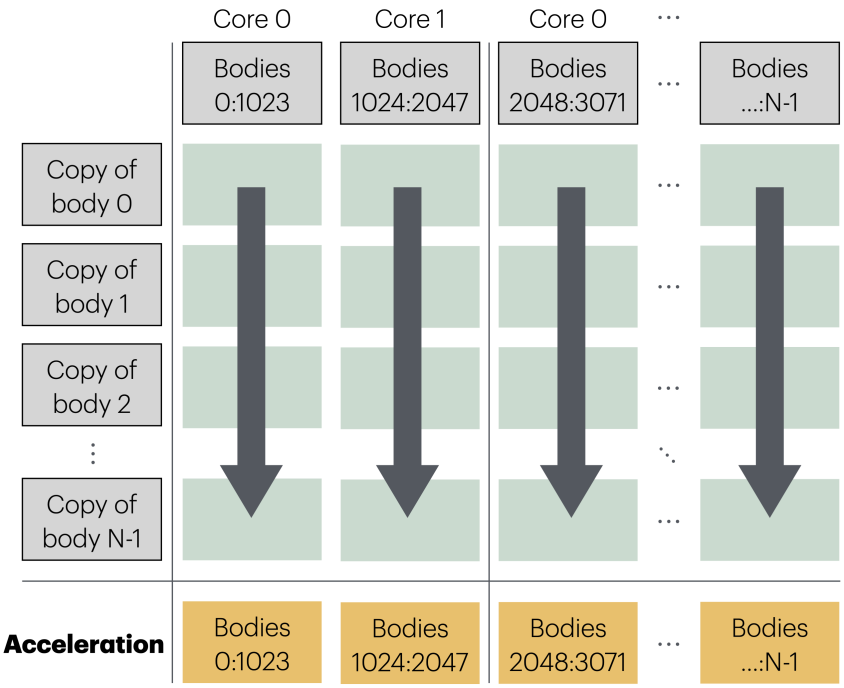

Figure 3.3.1: **Parallel execution of the N-Body force evaluation kernel on the Tenstorrent Wormhole.** The kernel is executed in parallel across multiple Tensix cores, each handling a subset of particle interactions.

3.3 Porting the N-Body Code to Tenstorrent Wormhole

The force evaluation kernel, the computational core of the N-body simulation, was ported to the Tenstorrent Wormhole architecture using the TT-Metalium SDK. The single-device campaign [22] used TT-Metalium v0.60.1, while the multi-device campaign [23] leveraged v0.62.2, which introduced the `MeshDevice` abstraction required for the Mesh-Based configuration. Porting follows the dataflow execution model of TT-Metalium (described in Section 2.3.2), with three custom kernels: `read`, `compute`, and `write`.

3.3.1 Data Layout and Tiling

Particle data are organized into tiles of 32×32 elements, which constitute the fundamental unit of computation and data movement in the Tensix processor. Since the Wormhole natively supports up to FP32, a mixed-precision strategy is adopted: the force evaluation kernel on the accelerator operates in FP32, while the remaining steps on the CPU (prediction and correction) retain FP64. The force evaluation kernel is parallelized by distributing tiles across the Tensix cores using a Single Program, Multiple Data (SPMD) model, as shown in Figure 3.3.1. The standard direct N-body algorithm consists of two nested loops: an outer loop over the particles for which forces are computed, and an inner loop over all

particles contributing to those forces. In the implementation, only the outer loop is parallelized on the accelerator. Particles in this loop are distributed across multiple Tensix cores, with each core processing a batch of 1024 particles. Since computing the net gravitational force on a given particle requires access to the positions and velocities of all other particles in the system, not only those locally assigned to a given core, the complete set of particle data must be made available to every core. To accommodate this within the tile-based data-access model of the Tensix architecture, particle data are replicated and organized into N tiles of 1024 elements each. Each tile contains 1024 identical copies of a single scalar quantity (r_x, r_y, r_z, v_x, v_y, or v_z) for one particle. This data layout ensures that each Tensix core has efficient, tile-aligned access to the full set of particle information required for force evaluation, while operating on its own assigned batch of 1024 particles.

3.3.2 Kernel Implementation

The **read kernel** runs on data movement cores and implements an asynchronous double loop: the outer loop loads the source particle data (positions and velocities of particle i) tile by tile into circular buffers (CBs), while the inner loop loads the replicated target particle tiles (positions, velocities, and masses of all particles).

The **compute kernel** runs on Tensix compute cores and consumes data from the CBs in the same order as produced by the read kernel. It computes element-wise pairwise accelerations and jerks using SFPU operations, including `sub_binary_tile()`, `square_tile()`, and `rsqrt_tile()`. Custom ternary SFPU functions were also implemented to compute the squared inter-particle distance and the multiply-add operations required for acceleration and jerk accumulation in a single fused step. Since the Tensix `dst` register can accommodate only 8 FP32 tiles, frequently reused intermediate values (displacement vector components Δx, Δy, Δz) are staged in on-chip SRAM via CBs to avoid register spills.

The **write kernel** transfers the accumulated accelerations and jerks from the compute stage back to off-chip DRAM.

The pipelined execution of these kernels is illustrated in Figure 3.3.2, where the use of circular buffers allows the read, compute, and write stages to execute concurrently across different iterations of the dataflow pipeline, effectively overlapping communication and computation to maximize throughput.

3.3.3 Parallelization Strategy and Multi-Device Scaling

Intra-chip parallelism is achieved through a Single Program, Multiple Data (SPMD) model: each Tensix core executes the same kernel but operates on a distinct subset of source particles, with workload distribution managed by the `split_work_to_cores` function of TT-Metalium.

To scale beyond a single chip, we implemented and evaluated three configurations, as shown in Figure 3.3.3:

1. **Multi-Host Single-Chip (MHOST):** Only the L-chip (PCIe-connected)

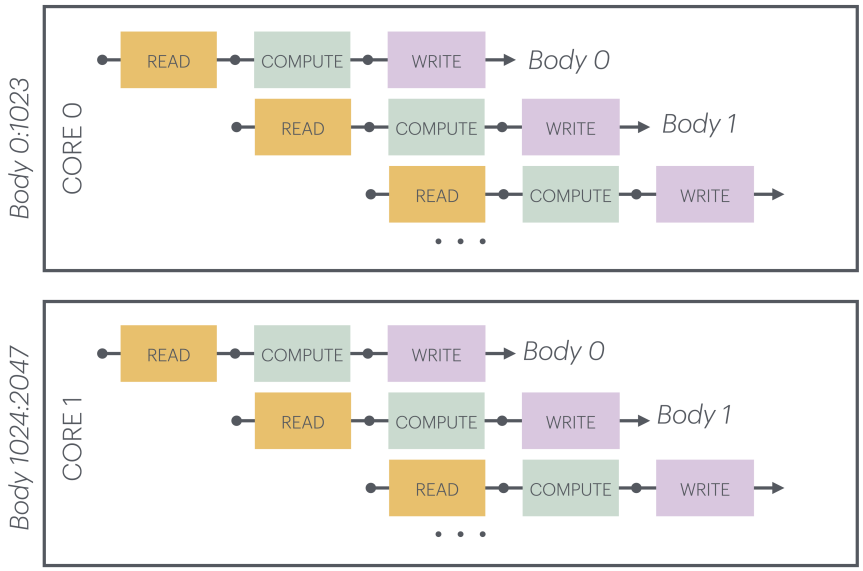

Figure 3.3.2: **Pipelined execution of the kernels on the Tenstorrent Wormhole.** Data movement and computation are overlapped, allowing the read, compute, and write kernels to execute concurrently. The full implementation of the N-Body code is publicly available at `https://github.com/jlalmerol/$N$-Body-Code`.

of each `n300` card is used. Each MPI process controls one chip. Inter-device communication uses MPI over the host network.

2. **Multi-Host Multi-Chip (MCHIP):** Both chips (L-chip and R-chip) of each card are used within a single MPI process. Separate program instances are defined for each device, with explicit buffer allocation and kernel enqueueing per chip. The R-chip communicates with the L-chip over the on-board Ethernet link.

3. **Mesh-Based Configuration (MESH)**: The TT-Metalium `MeshDevice` abstraction (introduced in v0.62.2) is used to manage a $(1, 2)$ mesh topology per MPI process, spanning the two chips on each card. Command queues automatically distribute operations to all devices in the mesh. Domain-decomposed data are managed through sharded buffers; globally shared particle data are managed through replicated buffers.

Multi-host execution was orchestrated using `tt-run`, the TT-Metalium distributed process launcher, which provides a YAML-based MPI-compatible interface. OpenMP was used to parallelize the prediction and correction steps on each host CPU.

3.3.4 Validation

The correctness of the Wormhole implementation was verified by comparing the computed accelerations and jerks against a double-precision brute-force CPU ref-

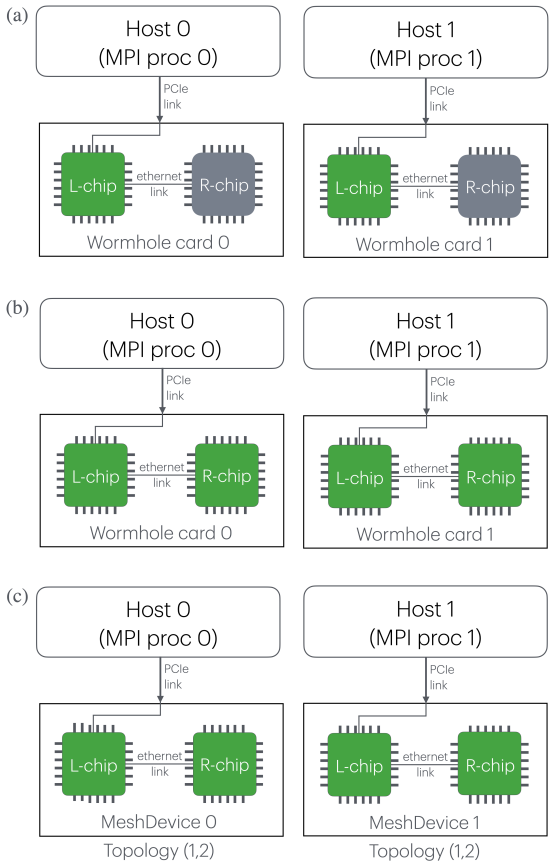

Figure 3.3.3: **Multi-device scaling configurations for the Tenstorrent Wormhole accelerator.** (Top) Multi-Host Single-Chip (MHOST) configuration, where only the L-chip of each `n300` card is used. (Middle) Multi-Host Multi-Chip (MCHIP) configuration, where both chips of each card are used within a single MPI process. (Bottom) Mesh-Based Configuration (MESH), leveraging the TT-Metalium `MeshDevice` abstraction to manage a mesh topology spanning both chips on each card.

erence implementation (the "golden reference"). The discrepancy in each acceleration component was required to be within 0.05% of a typical force magnitude relative to the double-precision result, and within 0.2% for each jerk component. These tolerances represent worst-case values observed for the representative particle configuration of $N = 102{,}400$ particles and were consistently satisfied across all simulated timesteps and parallelization configurations.

3.4 Performance and Energy Measurement Methodology

3.4.1 Time-to-Solution

Execution time (time-to-solution) is measured using `MPI_Wtime()` calls placed at the beginning and end of the simulation loop. Benchmark runs are executed in batches, with sleep periods inserted before and after each run to allow the system to return to idle conditions. These sleep intervals are excluded from the reported execution time. For statistical robustness, each benchmark configuration is repeated over multiple independent runs (typically 20–50), and average values with standard deviations are reported.

3.4.2 Energy-to-Solution

This section describes the energy measurement methodologies employed across the two experimental campaigns. Two distinct approaches are used depending on the platform under evaluation: PDU-based whole-node AC measurements for the cross-platform CPU comparison (Section 4.1.4), and a combination of Intel RAPL and the Tenstorrent `tt-smi` interface for the Wormhole versus CPU comparison (Sections 4.2.1–4.2.4). Because these approaches differ in their system boundaries—PDU measurements capture the full node power envelope, whereas RAPL and `tt-smi` operate at the package and device level respectively—they are not directly comparable across campaigns. Each campaign is nonetheless internally consistent, and any systematic bias affects all platforms within a campaign equally, leaving the relative efficiency conclusions unaffected.

Energy consumption for the RISC-V, AArch64, and x86 CPU platforms

Energy consumption is measured using a Power Distribution Unit (PDU). For each identified PDU outlet, active power (W) and apparent power (VA) were polled using the Simple Network Management Protocol (SNMP) in conjunction with the vendor-supplied Eaton ePDU Management Information Base (MIB), which exposes per-outlet active and apparent power measurements. Instantaneous readings from all outlets supplying the target node were summed to yield the total power draw at each sample point. Samples were collected concurrently with the N-Body simulation at a nominal interval of approximately 0.5 seconds, with minor timing jitter attributable to variable network and SNMP response

latency. Occasional gaps of up to twice the nominal interval were observed, consistent with sporadic dropped samples arising from polling timeouts. Each sample was timestamped at nanosecond resolution using the Unix epoch and recorded to a CSV file for post-processing.

This methodology captures whole-node AC power consumption, encompassing all system components including processor(s), memory, storage, and network interfaces. It is worth noting that the configurations examined in this study differ in processor count. This distinction is relevant when interpreting absolute power figures across platforms.

Energy consumption for the x86 CPU platform and Tenstorrent Wormhole

Energy consumption is sampled in user space at approximately 1 Hz over the full duration of each job (including sleep periods). For the CPU platforms, energy is measured using Intel's Running Average Power Limit (RAPL) interface via `perf stat -a -e`, which on AMD systems exposes the energy of CPU packages and cores. For the Wormhole cards, power is collected via the Tenstorrent system management interface `tt-smi`.

The energy-to-solution for each run is computed as the discrete integral of power over the active simulation time (excluding sleep periods). For multi-card experiments, the total energy is the sum of contributions from all active Wormhole chips and the dual-socket CPU. The energy-delay product—the product of energy-to-solution and time-to-solution—is also computed as a joint figure of merit that captures the efficiency-performance trade-off [29].

Chapter 4

Results and Discussion

This chapter presents a comprehensive performance evaluation of the RISC-V SG2042 platform against established x86 (AMD EPYC 9554) and AArch64 (NVIDIA Grace) architectures, followed by a performance analysis of the Tenstorrent Wormhole accelerator against an x86 CPU (AMD EPYC 9124) baseline, all conducted by running an N-Body simulation code with $N = 102400$ bodies. The results show significant architectural limitations in current RISC-V implementations while demonstrating the potential of emerging accelerator technologies. Key findings include a pronounced scaling degradation beyond 16 threads for RISC-V (SG2042) and effective but limited multi-device scaling for Wormhole accelerators.

The results presented in this chapter have been disseminated in two publications. The single-device Wormhole campaign (Section 4.2.1) is reported in [22]; the multi-device scaling analysis (Section 4.2.2) is reported in [23].

4.1 Cross-Platform Performance Comparison

Figure 4.1.1 (left) shows total N-Body simulation time for RISC-V, AArch64, and x86 across 1–128 CPU cores. x86 consistently achieves the lowest execution time, followed by AArch64, while RISC-V exhibits substantially higher wall times. All platforms improve with increasing core count (up to 16 cores for RISC-V and up to 64 for AArch64 and x86), after which scaling degrades.

To establish architectural baselines, RISC-V performance was normalized against x86 and AArch64 platforms for single-node multi-threaded execution. Figure 4.1.1(right) presents the relative performance expressed as

$$P_{rel} = 100 \times \frac{T_{baseline}}{T_{riscv}}$$

for the total simulation time as a function of CPU count. At CPU counts (1–16), RISC-V platform maintains a stable but significantly lower performance profile compared to the reference architectures with RISC-V operating at approximately $4 - 5\%$ of the x86 performance and approximately 10% of the AArch64 performance level. The horizontal scaling curves indicate comparable scaling rates across architectures despite RISC-V's lower absolute throughput.

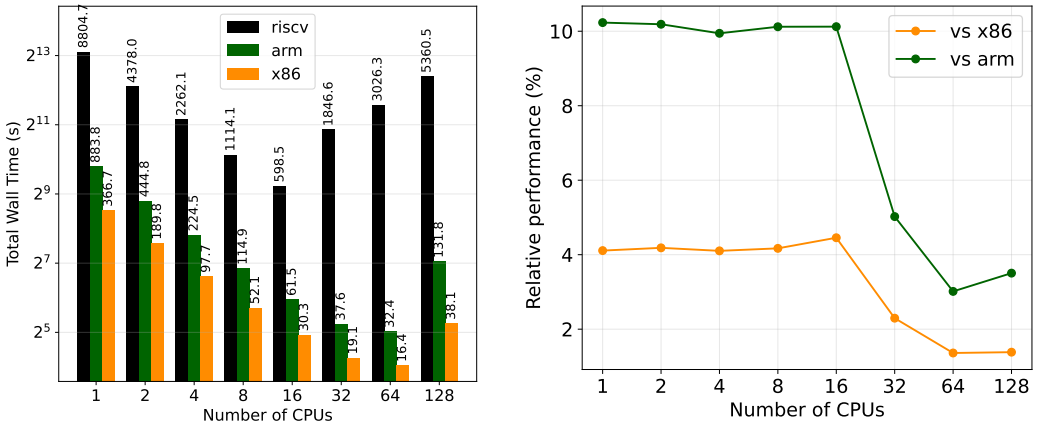

Figure 4.1.1: **Cross-platform performance comparison for intranode execution of N-Body Simulation.** (Left) Total simulation time on RISC-V, AArch64 and x86. (Right) RISC-V total simulation time performance relative to x86 and AArch64 baselines

A sharp drop in performance beyond 16 threads can be observed for as low as 0.5% relative to x86 and as low as 1% relative to AArch64. Up to 16 threads, the architecture utilizes its available resources with consistent efficiency relative to the baselines. Beyond it, additional threads do not yield proportional gains, likely due to substantial overheads that the x86 and AArch64 systems manage more effectively.

The steeper decline in total simulation time suggests that system-level factors, such as memory bandwidth saturation, NUMA effects, or synchronization overhead, become the dominant constraints beyond 16 threads. The RISC-V SG2042 exhibits a sharp performance degradation at exactly 16 threads, indicating a hardware locality boundary rather than the gradual resource saturation typical of mature architectures. These observations motivate a more detailed analysis of the internal scaling behavior of the RISC-V platform.

4.1.1 Strong Scaling Characteristics

Figure 4.1.2 presents strong scaling behavior for total simulation time across the three platforms. Across all platforms, near-ideal scaling is observed up to 8–16 processes. In this regime, parallel efficiency remains above 90%, indicating that communication overhead and synchronization costs are negligible compared to computation. However, as the core count increases further on the x86 and NVIDIA Grace (AArch64) platforms, a gradual decrease in parallel efficiency is observed. This behavior is consistent with strong-scaling limitations arising from a reduced per-rank workload; as the problem size per core diminishes, the ratio of communication and synchronization overhead to actual computation increases, eventually reaching a point of diminishing returns.

Beyond 16 processes, the SG2042 RISC-V system exhibits fundamentally different behavior with a pronounced efficiency collapse, dropping from 95% to under

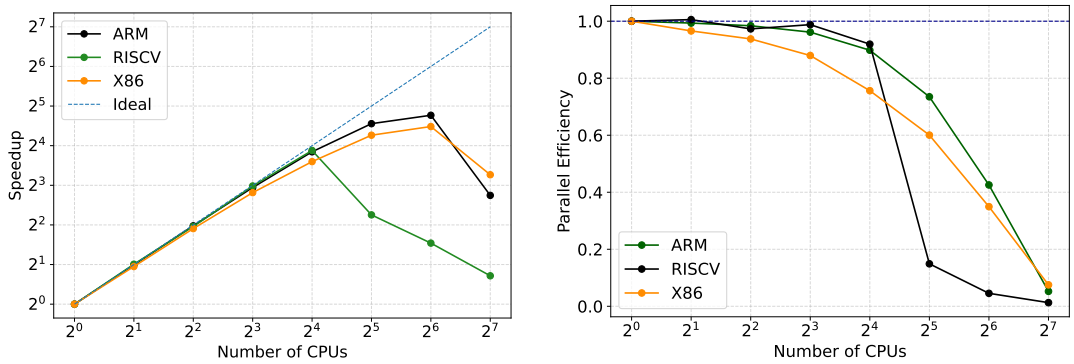

Figure 4.1.2: **Strong scaling analysis.** (Left) Speedup relative to single-process baseline with ideal scaling reference. (Right) Parallel efficiency demonstrating architectural scaling limits.

20% at 32 processes. Efficiency falls to very low levels, indicating that additional parallelism not only fails to improve performance but significantly degrades it. This degradation suggests structural hardware limitations. As strong scaling reduces the computational work per process, the application becomes increasingly sensitive to memory access latency and bandwidth. The abrupt performance degradation likely corresponds to crossing a cache hierarchy or NUMA domain boundary within the SG2042 design, compounded by a critical memory bandwidth bottleneck: the entire 64-core SG2042 processor is served by only four DDR4-3200 memory controllers [11, 12]. With 16 or fewer active cores, memory traffic can be handled by the four channels without contention; beyond this point, however, the aggregate bandwidth demand from additional cores saturates the available memory bus, causing the sharp degradation observed. This interpretation is consistent with independent analyses of the SG2042 for AI inference workloads, which identified memory bandwidth as the primary bottleneck under high core-count parallelism [12]. Less optimized RISC-V toolchains and MPI libraries may further amplify these hardware limitations.

4.1.2 Force Evaluation Kernel Analysis

To further investigate the performance characteristics of the RISC-V architecture, the strong scaling and parallel efficiency of the force calculation kernel were analyzed and compared against the x86 and AArch64 reference (as shown in Figure 4.1.3). Up to 16 threads, the RISC-V platform scales linearly, demonstrating that its internal computational units are effectively utilized for the force evaluation task. This aligns with the previous observation that the relative performance between RISC-V and the baselines remains stable in the 1 to 16 thread range, as all systems operate close to their theoretical ideal scaling.

Beyond 16 threads, the RISC-V speedup plateaus immediately, showing virtually no improvement as concurrency increases. This scaling behavior directly explains the sharp decline observed in the relative performance comparison. The plateau in RISC-V speedup strongly suggests a hardware-defined boundary, con-

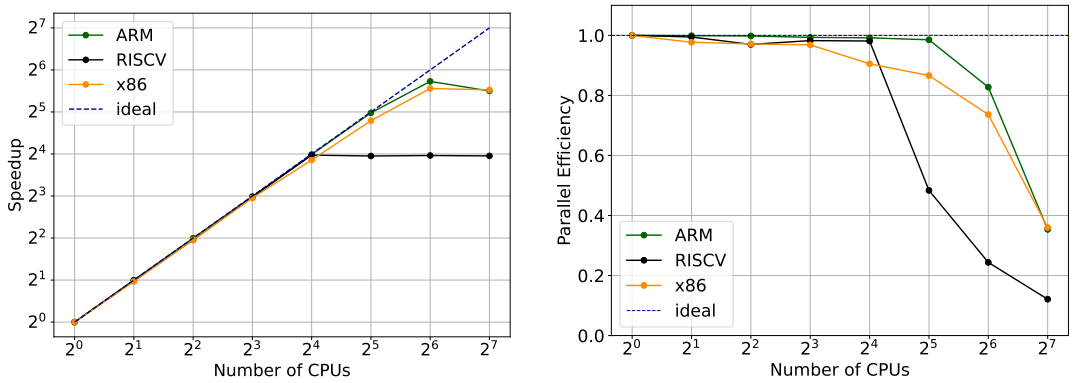

Figure 4.1.3: **Force evaluation kernel scaling.** (Left) Speedup with ideal linear scaling reference. (Right) Parallel efficiency showing compute-specific scaling characteristics across architectures.

sistent with saturation of the SG2042's four DDR4-3200 memory controllers [11, 12]. Although the force evaluation kernel is compute-intensive, its $O(N^2)$ pairwise loop requires each thread to stream particle data continuously from memory; when the number of active threads exceeds the bandwidth capacity of the four memory channels, threads stall waiting for data, negating the benefit of additional parallelism. For the force evaluation kernel, the fact that efficiency drops so sharply while the baselines remain performant further indicates that the RISC-V system suffers from significant contention in its memory subsystem when traversing across core clusters—an architectural constraint that does not afflict the higher-bandwidth memory subsystems of the AMD EPYC and NVIDIA Grace platforms. This suggests that for high-concurrency workloads on the SG2042, the current RISC-V implementation is limited by its memory bandwidth rather than the raw execution throughput of the individual cores.

4.1.3 Data Layout and Parallelization Strategy Impact

Figure 4.1.4 summarizes the total simulation time for the ARR, SOA, and AOS data layouts across different hybrid OpenMP+MPI configurations within a single node on the three architectures.

Across all architectures, increasing the number of OpenMP threads within a single MPI rank consistently provides the most effective performance improvement. For one MPI rank per node, runtime decreases steadily as the thread count increases up to the physical core limit. In contrast, dividing the node into multiple MPI ranks (2 or 4 ranks per node) generally results in higher runtimes, particularly at larger thread counts. This trend indicates that, for this workload, shared-memory parallelism is more efficient than hybrid decomposition within a single node.

For x86, the optimal configuration is 64 OpenMP threads with 1 MPI rank, where ARR and SOA layouts deliver near-identical best performance (∼16–17 s), while AOS incurs a modest overhead. The AArch64 platform follows the same pattern, with 64 OpenMP threads and 1 MPI rank being optimal and minimal

Table 4.1.1: Cross-architecture performance and energy comparison for the N-body simulation (ARR layout, 64 parallel workers). Estimated power figures reflect estimated system-level power consumption under load; Thermal design power (TDP) values are provided for reference. EDP is the product of energy-to-solution and time-to-solution; lower is better.

Metric	**x86**	**AArch64**	**RISC-V**
Best config (OMP × MPI)	64×1	64×1	16×4
Wall clock time (s)	16.39 ± 0.06	29.00 ± 0.44	257.00 ± 3.95
Speedup vs. RISC-V	15.67×	8.86×	1.00×
TDP (W)	360	200	120
Estimated power (W)	558.81 ± 2.41	527.25 ± 2.28	170.59 ± 0.09
Energy-to-solution (kJ)	9.16	15.29	43.84
Normalized energy (vs. x86)	1.00×	1.67×	4.79×
EDP (kJ s)	150.20	443.30	11,265.80
Normalized EDP (vs. x86)	1.00×	2.95×	75.00×

sensitivity to data layout, reflecting the hardware prefetcher's ability to tolerate different access patterns.

In contrast, the SG2042 RISC-V system demonstrates greater sensitivity to both data layout and hybrid configuration. Performance degrades more noticeably at higher levels of concurrency, especially when multiple MPI ranks are used. The ARR layout consistently yields the best RISC-V performance among the three layouts, particularly at the optimal 16-thread configuration; the additional memory indirection of SOA provides no benefit at this scale, and AOS suffers the most from the limited bandwidth of the four DDR4-3200 memory controllers. These results confirm that data layout choices that improve spatial locality have an outsized effect on platforms where bandwidth is the primary limiting resource.

4.1.4 Best-Configuration Cross-Platform Comparison

To provide a consolidated summary of the SG2042's capabilities relative to mature architectures, Table 4.1.1 compares all three platforms at their individually optimal configurations for the ARR data layout with 64 parallel workers, reporting not only time-to-solution but also energy consumption and the Energy-Delay Product (EDP). Energy-to-solution is calculated using estimated system-level power consumption figures.

Several observations emerge. First, regarding raw performance, x86 is the fastest platform at 15.67× over RISC-V, with AArch64 at 8.86× faster. Notably, RISC-V requires a hybrid MPI+OpenMP decomposition (16×4) to reach its optimum, whereas both AArch64 and x86 achieve their best times with a flat shared-memory configuration (MPI=1, OMP=64), reflecting the 16-thread scaling ceiling of the SG2042 discussed in Section 4.1.

Second, despite having the lowest estimated power of the three platforms (170.59 W), the SG2042 consumes 4.79× more total energy than x86 (43.84 kJ vs. 9.16 kJ), because its much longer execution time far outweighs the power

saving. This confirms that *low power draw does not translate into low energy-to-solution when the time penalty is as large as observed here*: for the $O(N^2)$ force evaluation kernel, the SG2042's four DDR4-3200 memory controllers are saturated well before all cores are productive, prolonging the run beyond any power advantage.

Third, the EDP reveals the most striking disparity. RISC-V's EDP of 11,265.80 kJ s is 75.00× worse than x86 and 25.4× worse than AArch64, reflecting the compounding penalty of both higher energy and longer runtime. x86 achieves the best EDP despite its higher estimated power consumption, because its execution time is short enough to dominate the product. AArch64's EDP (2.95× x86) is moderate, but the substantial gap with x86 underscores the cost of its longer execution time under realistic power estimates.

4.1.5 RISC-V Performance Assessment Summary

Table 4.1.1 consolidates the key findings of this section by comparing all three platforms at their individually optimal configurations. The SG2042 evaluation reveals four critical limitations:

1. **Absolute performance gap.** At their respective optima, x86 is 15.67× faster than RISC-V and AArch64 is 8.86× faster. The RISC-V optimum itself requires a hybrid MPI+OpenMP decomposition (16×4) that incurs additional communication overhead, whereas both reference platforms achieve their best times with a flat shared-memory configuration (64×1).

2. **Scaling bottleneck.** Parallel efficiency collapses from above 90% to under 20% beyond 16 threads, attributable to saturation of the processor's four DDR4-3200 memory controllers and NUMA domain crossing [11, 12]. This ceiling forces the use of MPI decomposition to reach peak throughput, adding latency that further widens the gap with mature architectures.

3. **Energy-to-solution penalty.** Despite the lowest estimated power of the three platforms (170.59 W vs. 527.25 W for AArch64 and 558.8 W for x86), the SG2042 consumes 4.79× more total energy than x86 (43.84 kJ vs. 9.16 kJ) and 2.87× more than AArch64 (15.29 kJ). Low power draw does not translate into low energy-to-solution when the execution time penalty is this large. By contrast, AArch64 consumes 1.67× more energy than x86, a penalty driven by its longer execution time despite a moderate power advantage, demonstrating that competitive performance remains the dominant factor in achieving energy-efficient HPC.

4. **EDP disparity.** The Energy-Delay Product of 11,265.80 kJ s is 75.00× that of x86 and 25.40× that of AArch64 (150.20 and 443.30 kJ s respectively). As a combined measure of both energy consumption and runtime, the EDP captures how a moderate power advantage is offset by a substantial time-to-solution penalty, resulting in an overall efficiency gap relative to both reference architectures under the tested conditions.

Taken together, these results show that while the SG2042's low thermal envelope may be attractive for power-constrained or embedded deployments, the current generation of RISC-V server hardware faces practical challenges in compute-intensive scientific workloads — namely, limited memory bandwidth, lower operating frequencies, and a maturing vectorization toolchain. Importantly, these characteristics are specific to the SG2042's microarchitectural implementation and memory subsystem, and are not inherent to the RISC-V ISA itself. Successor designs addressing both the memory subsystem and the software stack for RVV exploitation are already underway, and are expected to narrow this gap considerably.

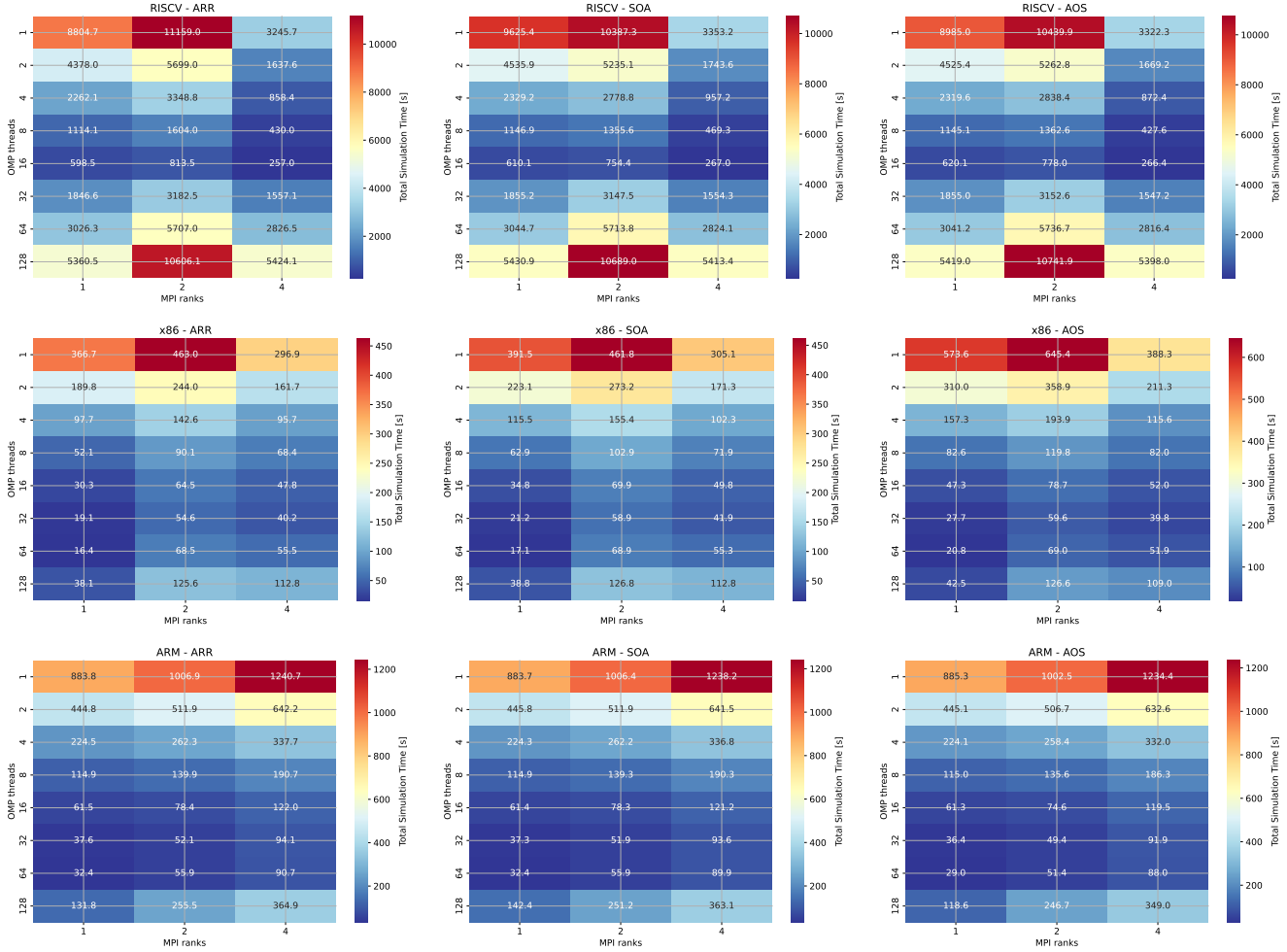

Figure 4.1.4: **Hybrid OpenMP+MPI performance heatmaps.** Runtime comparison across data layouts (AOS, ARR, SOA) for RISC-V (top row), x86 (middle row), and AArch64 (bottom row). Darker blue colors indicate better performance. X-axis shows OpenMP thread count, Y-axis shows MPI ranks per node.

4.2 Tenstorrent Wormhole Accelerator Performance

This section presents measured performance and energy results for the Tenstorrent Wormhole accelerator obtained in two experimental campaigns. The first campaign evaluated single-device performance [22]. The second extended the study to multi-device scaling using three parallelization strategies [23]. Two execution modes are compared throughout: Multi-Host Single-Chip (MHOST), in which only the PCIe-connected L-chip per card is used, and Multi-Host Multi-Chip (MCHIP), which employs both the L-chip and the Ethernet-connected R-chip per card within a single MPI process. A MESH configuration using the TT-Metalium `MeshDevice` abstraction was also evaluated.

4.2.1 Single-Device Performance and Energy Efficiency

The first experimental campaign characterized a single Wormhole `n300` card on a representative simulation of 102 400 particles over ten time cycles [22]. The accelerated code used a single OpenMP thread and one MPI task, while the CPU reference employed 32 OpenMP threads exploiting all physical cores and AVX-512 intrinsics. Results are summarized in Table 4.2.2.

Table 4.2.2: Single-device Wormhole performance versus CPU baseline (102 400 particles, 10 time cycles). Values are mean ± standard deviation over 26 accelerated and 49 reference runs [22].

Configuration	**Time-to-solution [s]**	**Energy-to-solution [kJ]**
Wormhole n300 (1 chip)	301.40 ± 0.24	71.56 ± 0.13
CPU (32 OMP + AVX-512)	672.90 ± 7.83	128.89 ± 1.52
Speedup / savings	**2.23×**	**1.80×**

The Wormhole card completes the simulation in 301.40 ± 0.24 s on average, compared to 672.90 ± 7.83 s for the CPU implementation, yielding a speedup of 2.23×. The substantially lower standard deviation of the accelerated runs (± 0.24 s versus ± 7.83 s) reflects the more deterministic execution environment of the accelerator compared to a general-purpose CPU subject to OS scheduling and resource contention.

On the energy side, the accelerated configuration consumes an average of 71.56 ± 0.13 kJ, compared to 128.89 ± 1.52 kJ for the CPU-only baseline—a reduction of 1.80×. This saving is achieved despite the Wormhole's peak power draw being somewhat higher during computation (≈260 W combined with the host CPU) compared to the CPU-only baseline (≈210 W), because the shorter execution time more than compensates for the elevated instantaneous power. Idle power of the four `n300` cards before and after simulation ranges between 10 and 11 W per card, rising to 26–33 W on the active device during force evaluation. These results confirm that the N-body force computation is well-matched to the Wormhole's tile-based dataflow architecture and SFPU vector engine.

4.2.2 Multi-Device Execution Time Across Parallelization Strategies

The second campaign [23] evaluated three strategies for scaling the code across up to four Wormhole n300 cards on a larger simulation of 409 600 particles over three time steps, hosted on a dual-socket AMD EPYC 9124 system. All simulations were launched using `tt-run` with each MPI task bound to one device and approximately 20 repetitions performed per configuration. Results, including the Energy-Delay Product (EDP), are reported in Table 4.2.3.

Table 4.2.3: Time-to-solution and Energy-Delay Product (EDP) for the three multi-device parallelization strategies and the CPU baseline (409 600 particles, 3 time steps; averages over ≈20 runs [23]). The CPU baseline uses 32 OpenMP threads with AVX-512 intrinsics. EDP is the product of energy-to-solution and time-to-solution; lower is better.

Approach	**Cards**	**# chips**	**Time [s]**	**EDP [kJ s]**
MHOST	1	1	1459.46 ± 0.47	563.10 ± 6.47
MHOST	2	2	1318.54 ± 2.72	479.43 ± 5.41
MCHIP	1	2	1511.67 ± 0.95	636.84 ± 0.82
MESH	1	2	9614.54 ± 1.14	27822.11 ± 52.17
CPU	–	–	2875.39 ± 5.30	1927.70 ± 7.30

The single-chip MHOST result (1459.46±0.47 s) corroborates the ≈2× speedup over CPU observed in the first campaign. Among the three multi-device strategies, MHOST consistently delivers the best performance. When two chips on different cards are used (two-card MHOST), execution time drops to 1318.54±2.72 s.

The MCHIP configuration, which employs both chips on the same card through a single MPI process, results in a slightly longer execution time (1511.67 s) than single-chip MHOST—a counterintuitive outcome indicating that the synchronization and data-movement overhead imposed by the on-board Ethernet link between the two ASICs exceeds the additional computational capacity provided by the second chip. In contrast, when chips from two separate cards are used via the MHOST approach, communication is routed through the host's PCIe fabric rather than the lower-bandwidth Ethernet, yielding better overall throughput. This asymmetry underscores that *the physical interconnect topology matters more than the raw number of chips employed.*

The MESH configuration suffers the most degradation, with a time-to-solution of 9614.54±1.14 s—approximately 6.6× slower than single-chip MHOST—and an EDP of 27822.11±52.17 kJ s (≈49× worse than two-card MHOST). This penalty is attributed to the high communication overhead introduced by the MeshDevice abstraction when coordinating workload distribution across chips connected via Ethernet rather than PCIe. The current implementation uses device-level sharding, which enables multi-chip execution but does not fully exploit intra-device parallelism [23]. Importantly, this degradation is primarily a consequence of the immaturity of the MeshDevice abstraction in TT-Metalium v0.62.2 rather than

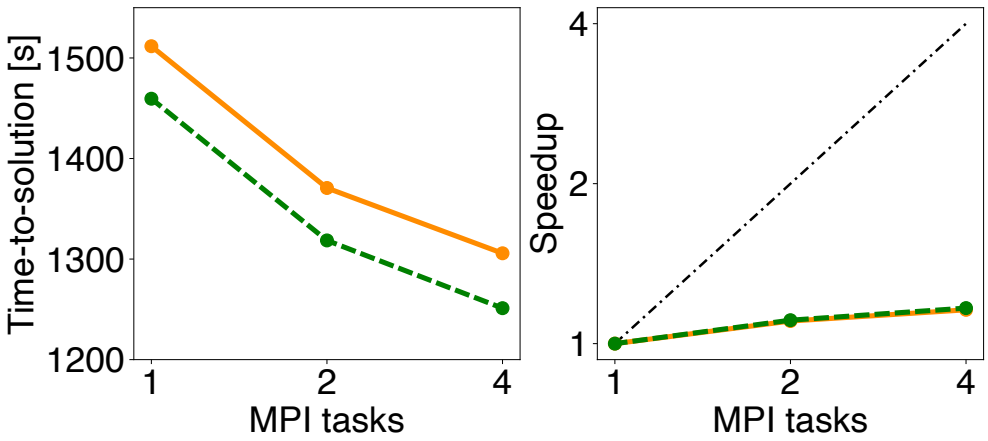

Figure 4.2.5: **Time-to-Solution and strong scaling of multi-device Wormhole configurations.** (Left) Time-to-solution for MHOST (green dashed line) and MCHIP (orange solid line). (Right) Strong scaling parallel speedup; dashed-dotted line marks ideal scaling.

a fundamental architectural limitation of the Wormhole hardware itself; future software releases that better exploit intra-device parallelism within a mesh are expected to reduce this overhead substantially.

As a contextual data point, preliminary results on an NVIDIA H100 GPU (AMD Genoa host, same particle count and time steps) show a 2.6× speedup over the single-chip MHOST Wormhole configuration. This performance gap can reasonably be attributed to the relative maturity of the Wormhole software stack and its FP32-only SFPU compared to the highly optimized CUDA ecosystem, and is expected to narrow as the TT-Metalium platform matures and successor hardware such as the Tenstorrent Blackhole p150 becomes available.

4.2.3 Strong Scaling and Parallel Efficiency

Strong scaling tests were conducted for the MHOST and MCHIP configurations across 1, 2, and 4 MPI tasks. The results are illustrated in Figure 4.2.5.

The left panel of Figure 4.2.5 shows time-to-solution as a function of MPI task count. Both MHOST and MCHIP exhibit a monotonically decreasing trend with increasing device count, confirming that additional chips do contribute to reducing wall-clock time. However, MHOST consistently outperforms MCHIP by approximately 1.04× for the same number of MPI tasks, reflecting the lower communication latency of PCIe-linked chips on separate cards relative to Ethernet-linked chips on the same card.

The right panel presents the corresponding parallel speedup, with the ideal linear reference shown as a dashed line. Both strategies fall well short of ideal scaling. For MHOST, a speedup of 1.10× (parallel efficiency: 55%) is achieved at two MPI tasks, rising to 1.16× (efficiency: 29%) at four MPI tasks. MCHIP shows a comparable trend [23]. Parallel efficiency is defined as $E(p) = S(p)/p$,

where $S(p)$ is the speedup at p MPI tasks.

The rapid efficiency decline is consistent with Amdahl's Law: as device count grows, the fixed overhead of MPI synchronization, PCIe data transfer, and host-side orchestration (kernel launch, global reductions) constitutes an increasing fraction of total runtime. For the problem size tested (409 600 particles, 3 time steps), the per-device computation is insufficient to offset these overheads at higher device counts. The inferior performance of MCHIP relative to MHOST further underscores that intra-card Ethernet communication imposes non-negligible penalties at this workload scale.

4.2.4 Energy-to-Solution and Energy-Delay Trade-Off

Figure 4.2.6 presents the energy-to-solution (left panel) and peak system power (right panel) as a function of MPI task count for both MHOST and MCHIP configurations.

Unlike time-to-solution—which decreases monotonically with device count—the energy-to-solution exhibits a non-monotonic trend for both configurations, reaching a minimum at two MPI tasks with an energy reduction of approximately 10% compared to the single-task configuration. This non-monotonic behavior arises because, while additional devices reduce execution time (lowering the duration over which power is drawn), they also increase the instantaneous system power, and the two effects do not cancel uniformly as device count grows. At four MPI tasks, the marginal reduction in execution time is insufficient to offset the added power draw of the extra cards, causing energy to rise again.

The EDP mirrors this ranking and is also minimized at two MPI tasks: MHOST with two cards achieves the lowest EDP of 479.43 ± 5.41 kJ s, compared to 563.10 ± 6.47 kJ s for single-chip MHOST and 636.84 ± 0.82 kJ s for MCHIP with one card. The CPU baseline EDP of 1927.70 ± 7.30 kJ s is more than $4\times$ higher than this best Wormhole configuration, confirming that even with limited scaling, the Wormhole offers a substantive energy-efficiency advantage over a highly optimized CPU implementation [29].

The right panel of Figure 4.2.6 shows that employing two chips per card in MCHIP increases peak system power by up to $\approx 8.7\%$ relative to single-chip MHOST configurations. This additional power draw, combined with the inferior time-to-solution of MCHIP, further reinforces the preference for the MHOST strategy when energy efficiency is a design priority.

4.2.5 Synthesis and Root Causes

The results across both experimental campaigns establish a consistent picture. A single Wormhole chip delivers substantial and reliable benefits: $2.23\times$ speedup and $1.80\times$ energy savings over a 32-thread, AVX-512-optimized CPU baseline [22]. Multi-device scaling is limited by three interacting bottlenecks [23]: (i) communication overhead grows as per-device computation shrinks under strong scaling; (ii) host-side orchestration cost—kernel launch, MPI synchronization, global reductions—becomes the dominant runtime term; (iii) card topology matters more

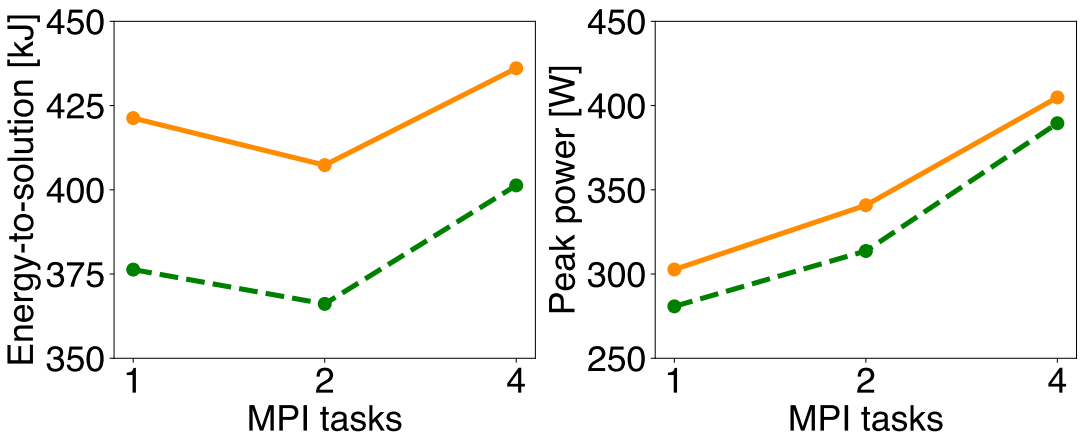

Figure 4.2.6: **Energy-to-Solution and Peak System Power vs. MPI Task Count**. Energy-to-solution (left) and peak power (right) as functions of the number of MPI tasks for simulations performed using the Multi- Host Single-Chip (green dashed line) and Multi-Host Multi-Chip (orange solid line) configurations.

than raw chip count, with PCIe-linked chips on separate cards (MHOST) outperforming Ethernet-linked chips on the same card (MCHIP). Addressing these bottlenecks through larger problem sizes, asynchronous host–device overlap, and direct chip-to-chip communication is identified as the primary avenue for future improvement.

4.3 Summary of Results

Three platforms were evaluated for direct N-body simulations: the RISC-V SG2042 CPU benchmarked against x86 and AArch64 references, and the Tenstorrent Wormhole `n300` accelerator tested in single- and multi-device configurations. The key findings are organized around three themes.

RISC-V Performance and Energy Gap. At their respective optimal configurations, x86 (AMD EPYC) is 15.67× faster than the RISC-V SG2042 and AArch64 (NVIDIA Grace) is 8.86× faster. At low thread counts (1–16), RISC-V achieves only 4–5% of x86 throughput and ≈ 10% of AArch64 throughput. Despite having the lowest estimated power draw of the three platforms (170.59 W vs. 527.25 W for AArch64 and 558.8 W for x86), the SG2042 consumes 4.79× more total energy than x86 (43.84 kJ vs. 9.16 kJ), because the pronounced time penalty far outweighs the power saving. The Energy-Delay Product (EDP) exposes the starkest disparity: RISC-V's EDP of 11,265.80 kJ s is 75.0× worse than x86 and 25.4× worse than AArch64, confirming that the SG2042 is currently not competitive for energy-efficient HPC under the tested conditions.

Hardware Topology and Data Layout Sensitivity. Parallel efficiency on RISC-V collapses from above 90% to under 20% beyond 16 threads, driven by saturation of the SG2042's four DDR4-3200 memory controllers and NUMA domain crossing. This forces the use of a hybrid 16 × 4 OpenMP+MPI decomposition to reach peak throughput—a strategy unnecessary on x86 and AArch64, both of

which achieve their optima with a flat 64×1 shared-memory configuration. Data layout also plays an outsized role: the ARR layout consistently outperforms SOA and AOS on RISC-V by maximizing spatial locality under constrained memory bandwidth.

Accelerator Promise with Interconnect-Limited Scaling. A single Tenstorrent Wormhole `n300` chip delivers a 2.23× speedup and a 1.80× energy saving over a 32-thread AVX-512 CPU baseline (102 400 particles, 10 time steps), confirming that the N-body force computation is well-matched to the Wormhole's dataflow execution model. Multi-device scaling is, however, interconnect-limited. Communication topology proves more decisive than raw chip count: two chips on separate cards connected via PCIe (MHOST) outperform two chips on the same card connected by on-board Ethernet (MCHIP), with MCHIP's second chip actually *degrading* performance relative to single-chip MHOST. Parallel efficiency at four MPI tasks drops to $\approx 29\%$ (MHOST) and $\approx 24\%$ (MCHIP). The MESH strategy performs worst, running 6.6× slower than single-chip MHOST with an EDP $\approx 49\times$ higher, owing to the immaturity of the `MeshDevice` abstraction in TT-Metalium v0.62.2. The energy-optimal operating point is two-card MHOST (EDP 479.43 kJ s), more than 4× better than the CPU baseline (1927.70 kJ s). A preliminary comparison against an NVIDIA H100 GPU shows a 2.6× performance gap in favor of the H100, providing a concrete target for future Tenstorrent hardware generations.

The performance and energy deficits of the SG2042 are traceable to hardware constraints specific to this first generation of RISC-V server silicon—insufficient memory bandwidth, lower clock frequency, and an immature RVV toolchain—rather than to fundamental limitations of the RISC-V ISA itself. These are generational constraints with identified architectural remedies, and the results here quantify precisely how large the gap is and where it originates.

The Wormhole accelerator tells a different story. At the single-device level it already outperforms a highly optimized CPU baseline in both time- and energy-to-solution, demonstrating that RISC-V-based accelerators can be competitive when the workload maps well to the hardware. Multi-device scaling is currently limited by software stack immaturity rather than by the Tensix cores themselves, and the energy-optimal configuration—two-card MHOST—is identified from measured EDP minima rather than assumed.

Physical interconnect topology is found to matter as much as chip count in the Wormhole experiments. The PCIe-versus-Ethernet asymmetry between MHOST and MCHIP shows that routing data through the host fabric outperforms on-board Ethernet even when the latter connects chips on the same card. The choice of scaling strategy must therefore account for the underlying interconnect, not just the number of devices.

4.4 Conclusions

This thesis establishes the first quantitative performance and energy efficiency characterization of RISC-V hardware for direct N-body simulations, spanning both a general-purpose server CPU (RISC-V SG2042, benchmarked against AMD

EPYC x86 and NVIDIA Grace AArch64) and a RISC-V-based dataflow accelerator (Tenstorrent Wormhole `n300`).

RISC-V CPU vs. x86 and AArch64

The performance and energy gaps measured for the SG2042 are traceable to three hardware-level constraints: insufficient memory bandwidth for high-concurrency workloads, a lower operating clock frequency, and the relative immaturity of the RVV toolchain. Crucially, these constraints are not intrinsic to the RISC-V ISA; rather, they reflect the early-stage nature of this first generation of RISC-V server silicon. The starkest measure of this gap is the EDP: at 11,265.80 kJ s, the SG2042 is 75.0× worse than x86 and 25.4× worse than AArch64 under the tested conditions. The observed bottleneck is architectural and generational in character: the SG2042's four DDR4-3200 memory controllers saturate beyond 16 threads, and the SG2044—its direct successor—doubles this to eight controllers, directly targeting the primary constraint identified here. The results therefore quantify what must improve and by how much before RISC-V CPUs become competitive for energy-efficient HPC workloads of this class.

Tenstorrent Wormhole Accelerator

To the best of our knowledge, this work represents the first demonstrated porting of an astrophysical N-body code to the Tenstorrent Wormhole platform, establishing a reproducible baseline for future comparisons as the TT-Metalium software stack matures. The single-device results are notably encouraging: the regular, data-parallel structure of the all-pairs force loop maps naturally onto the Wormhole's tile-based dataflow execution model and SFPU vector engine, yielding a 2.23× speedup and 1.80× energy saving over a highly optimized CPU baseline. This demonstrates that RISC-V-based accelerator designs can deliver genuine energy-efficiency advantages when the architecture is well-matched to the workload — a result that stands in instructive contrast to the CPU findings and underlines the heterogeneous and evolving nature of the RISC-V hardware ecosystem.

The multi-device results reveal two distinct bottlenecks. First, host-side orchestration overhead and immature multi-chip abstractions in TT-Metalium v0.62.2 limit scaling efficiency for both MHOST and MCHIP, and are responsible for the severe degradation of the MESH configuration (6.6× slower than single-chip MHOST). Second, and more fundamentally, physical interconnect topology determines multi-device throughput more than raw chip count: routing data through the host PCIe fabric (MHOST) outperforms on-board Ethernet between chips on the same card (MCHIP), with the MCHIP second chip actually degrading performance relative to single-chip MHOST. The energy-optimal configuration—two-card MHOST at EDP 479.43 kJ s, more than 4× better than the CPU baseline—is determined by these interconnect constraints, not by compute capacity. The 2.6× performance gap relative to an NVIDIA H100 GPU is therefore most appropriately interpreted as a software stack maturity gap and a

target for the next hardware generation, rather than as an inherent limitation of the RISC-V accelerator architecture.

Future Work

The following directions are identified for future investigation:

- **RISC-V CPU next generation:** Evaluation of the Sophon SG2044 is the natural direct follow-up to this work. The SG2044 doubles the number of DDR memory controllers from 4 to 8, directly targeting the memory bandwidth bottleneck identified here. This architectural improvement is expected to substantially increase available memory bandwidth, alleviate the scaling saturation observed beyond 16 cores, and narrow the energy-to-solution gap relative to AArch64 and x86 platforms.
- **Future hardware evaluation:** Extending the benchmark suite to the Tenstorrent Blackhole p150 is a natural next step, given its Tensix core heritage combined with higher on-chip bandwidth and improved inter-chip connectivity. A direct CUDA/SYCL comparison on current NVIDIA and AMD GPUs using the same N-body kernel would further position the RISC-V accelerator roadmap within the broader HPC landscape.

The results presented in this thesis provide a quantitative, platform-by-platform characterization of the RISC-V hardware ecosystem for compute-intensive astrophysical workloads. The findings reveal a heterogeneous and evolving landscape. The SG2042 CPU is currently constrained by memory bandwidth limitations and toolchain immaturity that are specific to this hardware generation, have an identifiable architectural origin, and are addressable in successor designs. The Wormhole accelerator, by contrast, already achieves competitive energy efficiency at the single-device level, with multi-device scaling contingent on continued development of the TT-Metalium software stack and improvements to the hardware interconnect fabric. Taken together, these results show that achieving good performance and energy efficiency on emerging RISC-V hardware requires careful matching of parallelization strategy and data layout to the specific memory and interconnect constraints of each platform—as demonstrated here by the NUMA-aware decomposition on the SG2042 and the topology-driven choice of MHOST over MCHIP on the Wormhole. The experimental results reported here establish a reproducible quantitative baseline for N-body workloads on RISC-V platforms, against which the performance and energy efficiency of future hardware generations can be systematically evaluated.

Bibliography

[1] Daniel Suarez, Francisco Almeida, and Vicente Blanco. Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs. *The Journal of Supercomputing*, pages 1–19, 2024.

[2] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanovic. The RISC-V instruction set manual, volume I: User-level ISA, version 2.0. Technical Report UCB/EECS-2014-54, EECS Department, University of California, Berkeley, 2014.

[3] Arm Limited. *ARM® Architecture Reference Manual for A-profile Architecture*. Cambridge, UK, 2024. Issue 01, Document ID: 102105_L.a.

[4] Nick Brown and Richard Barton. Accelerating stencils on the Tenstorrent Grayskull RISC-V accelerator. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1690–1700. IEEE, 2024.

[5] Nick Brown, J. Davies, and F. LeClair. Exploring fast Fourier transforms on the Tenstorrent Wormhole. *arXiv preprint arXiv:2506.15437*, 2025.

[6] Einstein Telescope Collaboration. Einstein telescope. `https://www.einstein-telescope.it/en/einstein-telescope-en/`, 2025. Accessed: 2025-01-17.

[7] Rainer Spurzem. Direct n-body simulations. *Journal of Computational and Applied Mathematics*, 109(1–2):407–432, 1999.

[8] Benjamin P. Abbott et al. Prospects for observing and localizing gravitational-wave transients with advanced LIGO, advanced Virgo and KAGRA. *Living Reviews in Relativity*, 23:1–69, 2020.

[9] Keigo Nitadori and Junichiro Makino. Sixth- and eighth-order Hermite integrator for N-body simulations. *New Astronomy*, 13(7):498–507, 2008.

[10] Mario Spera. *High Precision, High Performance Simulations of Astrophysical Stellar Systems*. PhD thesis, Università degli Studi di Roma "La Sapienza", 2014.

[11] MilkV. Introduction of Sophon SG2042. `https://milkv.io/docs/pioneer/getting-started/processorintroductionof-sophon-sg2042`, 2024.

[12] Giulio Malenza, Adriano Marques Garcia, Robert Birke, Luca Benini, and Marco Aldinucci. Analysis of model parallelism for AI applications on a 64-core RV64 server CPU. *International Journal of Parallel Programming*, 2025.

[13] Tenstorrent. Wormhole™ n150d/n150s/n300d/n300s Tensix processor: Specifications and requirements. `https://docs.tenstorrent.com/aibs/wormhole/specifications.html`, 2025. Tenstorrent Hardware Documentation, v1.0. Accessed: February 2026.

[14] Peter Cawley. Tenstorrent Wormhole series part 5: Taking apart T tiles. `https://www.corsix.org/content/tt-wh-part5`, 2024. Blog post, corsix.org. Posted September 22, 2024. Accessed: February 2026.

[15] M. Chang. Programming Tenstorrent processors. `https://clehaxze.tw/gemlog/2025/04-21-programming-tensotrrent-processors.gmi`, 2025. Accessed: 2025-08-08.

[16] Tenstorrent. Tenstorrent ISA documentation: Low-level architectural documentation for Wormhole B0 and Blackhole A0. `https://github.com/tenstorrent/tt-isa-documentation`, 2025. GitHub repository. Accessed: February 2026.

[17] DeepWiki. tenstorrent/tt-isa-documentation: Tensix tile architecture. `https://deepwiki.com/tenstorrent/tt-isa-documentation/3-tensix-tile-architecture`, 2025. DeepWiki indexed documentation of the Tenstorrent ISA Documentation repository. Accessed: February 2026.

[18] Tenstorrent. TT architecture and Metalium guide. `https://github.com/tenstorrent/tt-metal/blob/main/METALIUM_GUIDE.md`, 2025. TT-Metalium SDK documentation, tenstorrent/tt-metal GitHub repository. Accessed: February 2026.

[19] DeepWiki. Tensix coprocessor — tenstorrent/tt-isa-documentation. Online, 2025. Accessed: Jul. 11, 2025.

[20] Tenstorrent. tt-isa-documentation: Tenstorrent ISA documentation. GitHub repository, 2025. Accessed: Jul. 10, 2025.

[21] C. Glagovich. FlashAttention on Tenstorrent's Wormhole architecture. `https://github.com/tenstorrent/tt-metal/blob/main/tech_reports/FlashAttention/FlashAttention.md`, 2025. Accessed: 2025-07-18.

[22] Jenny Lynn Almerol, Elisabetta Boella, Mario Spera, and Daniele Gregori. Accelerating gravitational N-body simulations using the RISC-V-based Tenstorrent Wormhole. In *SC25-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025. arXiv:2509.19294.

[23] Jenny Lynn Almerol, Elisabetta Boella, Mario Spera, and Daniele Gregori. Assessing performance and porting strategies for gravitational N-body simulations on the RISC-V-based Tenstorrent Wormhole. *Astronomy & Computing*, 2025. Submitted December 2025, under review.

[24] Andrea Bartolini, F. Ficarelli, E. Parisi, et al. Monte Cimone: Paving the road for the first generation of RISC-V high-performance computers. In *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, pages 1–6, 2022.

[25] E. Venieri, S. Manoni, G. Ceccolini, et al. Monte Cimone v2: Down the road of RISC-V high-performance computers. *arXiv preprint arXiv:2503.18543*, 2025.

[26] G. Mahale, T. Limbasiya, M.A. Aleem, et al. Optimizations for very long and sparse vector operations on a RISC-V VPU: A work-in-progress. In A. Bienz et al., editors, *High Performance Computing*, pages 472–485, Cham, 2023. Springer Nature Switzerland.

[27] M. Rexroth, C. Schäfer, G. Fourestey, and J.P. Kneib. High performance computing for gravitational lens modeling: Single vs double precision on GPUs and CPUs. *Astronomy and Computing*, 30:100340, 2020.

[28] C. Schäfer, G. Fourestey, and J.P. Kneib. LensTool-HPC: A high performance computing based mass modelling tool for cluster-scale gravitational lenses. *Astronomy and Computing*, 30:100360, 2020.

[29] G. Amati, M. Turisini, A. Monterubbiano, et al. Experience on clock rate adjustment for energy-efficient GPU-accelerated real-world codes. In *ISC 2025 Proceedings*, Lecture Notes in Computer Science. Springer, 2025.