The Journal of Supercomputing (2025) 81:1484
https://doi.org/10.1007/s11227-025-07972-7

Distributed fast and accurate simulation platform for advanced ARM- and RISC-V-based HPC systems

Nikolaos Tampouratzis[1,3] · Ioannis Papaefstathiou[2,3] · Gabriel Gomez-Lopez[4] · Miguel Sánchez De la Rosa[4] · Jesus Escudero-Sahuquillo[4] · Pedro Javier Garcia[4]

Received: 23 September 2024 / Accepted: 9 October 2025

Abstract

The new design paradigms of HPC systems utilizing newly developed CPUs (i.e., ARM and RISC-V) trigger an urgent need for simulators that can handle in an integrated manner both the processing and the network components of a system-under-design (SuD). The presented simulation framework is the first known open-source approach that efficiently integrates well-established simulation subsystems in a single framework with a single notion of time. The framework operates in a fully distributed manner, transparent to the user enabling accurate simulations of both ARM- and RISC-V-based HPC systems interconnected with different networks (e.g., InfiniBand and BXI). Our framework can work in two different modes; in the main one the processing systems and the interconnection networks are simulated at a cycle-accurate manner. The second option can be used if the designer wants to focus on the design space exploration of the architectures/topologies/technologies of the interconnection network, in which case the data collected when running the simulator on the main node, are utilized so as to model and simulate extremely fast different intercommunication infrastructures. The presented framework has been evaluated when simulating widely used benchmarks (HPCG & LAMMPS) on both ARM & RISC-V CPUs; the results demonstrate that our approach has up to 6% performance error in the reported SuD aspects, while the simulation times increase at a very slow pace, when the size of the HPC SuD gets bigger, due to the fully distributed nature of the proposed toolset.

Keywords HPC simulator · Distributed systems simulator · Integrated simulator ARM · RISC-V

Extended author information available on the last page of the article

Published online: 21 October 2025

1 Introduction

Nowadays, there is a rapid advancement in the capabilities of Highly Parallel and Distributed computing systems, commonly referred to as HPC systems. These systems encompass a wide range of processing units from well-known X86/ARM to modern RISC-V architectures interconnected through multiple networks. A significant challenge encountered by designers of heterogeneous systems is the lack of simulation tools capable of providing comprehensive insights beyond basic functional testing, especially for advanced low-power (ARM- & RISC-V-based) HPC systems. Such insights include the actual performance of the nodes, accurate overall system timing, power/energy estimations, and network deployment issues. However, in an era of very complex HPC systems, simulating independently only parts, components, or attributes of a system-under-design is a cumbersome, inaccurate, and inefficient approach.

In this paper, we present a simulation framework called COSSIM-HPC which extends the COSSIM open-source framework [1–3] and aims to address all the aforementioned constraints. The COSSIM framework effectively incorporates a collection of subtools that simulate the computing devices of the processing nodes, as well as the intercommunication network(s). It provides cycle-accurate results by simulating the actual application and system software executed on each node together with the actual networks employed. Specifically, we first extend COSSIM to support modern RISC-V architectures and create an HPC runtime environment enabling the designer to execute the whole Message Passing Interface (MPI) application simulating both the heterogeneous processing part (ARM & RISC-V) and the network part, in a fully distributed manner.

At the same time, the very detailed simulated models always come at a cost of large computing and memory requirements and high simulation times. Thus, in certain cases (e.g., in the first design phases), quick and maybe not very accurate simulations are needed. As a result, simulation frameworks sometimes integrate different tools that allow the fast simulation of a large system or a specific network technology (e.g., BXI or InfiniBand) at the cost of reduced accuracy of some parts of the system-under-design (SuD). For instance, network simulators abstract the modeling of the computing nodes. This is the case of the OMNeT++-based SAURON simulator [4], which can simulate high-performance interconnection networks with thousands of server nodes, which are modeled just as injectors of traffic to the network. SAURON also permits the simulation of different link speeds, network topologies, routing algorithms, switch architectures, flow-control policies, and strategies for congestion management, quality of service (QoS), and power management. Hence, it can assess network designers who are developing new solutions that improve network performance. The proposed integration allows flexibility to use network simulators other than SAURON, which will need to link the VEF traces framework.[1] A critical factor in these scale-out simulators is their ability to reproduce realistic

[1] Note that it is possible to link the VEF traces framework with other network simulator tools whenever that simulator includes the `VEF-Tracelib` library.

workloads in the interconnection network; very often, the simulation frameworks only model ad hoc or synthetic traffic patterns, which do not reproduce the behavior of communication operations observed in real systems.

For this reason, COSSIM-HPC efficiently integrates the VEF traces framework [5], a set of tools that model the communication traffic of HPC applications. In this manner, MPI-based applications can be run in COSSIM-HPC, and the communication operations can be recorded into VEF traces and then used as inputs for the network models of any network simulator compatible with the VEF traces framework. In our case, the data recorded by the VEF toolkit is used as input to the SAURON network simulation, which can simulate seamlessly and very fast HPC systems consisting of hundreds of HPC nodes.

The contribution of this paper can be summarized in the following points:

- The first known open-source[2] integrated simulation framework, which can simulate complete heterogeneous HPC systems supporting ARM and **RISC-V** architectures.
- Full support of PCI devices and Multi-Gigabit Network Interface Cards (NICs) in simulated multi-node RISC-V systems (the developed code is merged into the development code of the official gem5 repository).
- An innovative flow to enable the designers to accurately simulate several important aspects of HPC systems (i.e., CPU and Network Environment) when executing real MPI applications within a single simulation framework.
- An integrated Framework [5] that allows the analysis of communication traffic of MPI-based applications and generate traffic traces that can be used to feed other network simulator tools.
- A complete toolset allowing extremely fast simulations of HPC systems consisting of hundreds of nodes and interconnected with InfiniBand and/or BXI network technologies.
- A thorough evaluation of the end system based on widely used HPC benchmarks (HPCG & LAMMPS) executed on both ARM & RISC-V architectures.

2 Related work

In this section, we provide a brief overview of the most widely used simulators and frameworks that have been proposed in the HPC domain.

SIM-MPI [6] and LogGOPSim [7] are trace-driven HPC simulators modeling the traces and predicting the overall performance of the target system. The SIM-MPI input consists of a sequence of traces of MPI operations (including their computation times) simulated through the LogGOP model. LogGOPSim predicts the communication time by simulating the send and receive traces using the LogGOP model. PS-SIM [8], SimHPC [9] are HPC simulators that utilize execution-driven methodologies. These simulators execute the MPI program at the specified scale on a real

[2] https://github.com/ntampouratzis/COSSIM-HPC.

cluster with fewer processors compared to the target system. They use local hosts and the mathematical interconnection network model LogP to simulate the target system. To enhance the accuracy of their microarchitecture modeling and acquire precise computation time traces, those four simulators usually collect the computation traces on local nodes of the same architecture as the target system.

A parallel simulation framework, the Structural Simulation Toolkit (SST) [10], was developed to explore innovations in highly concurrent systems. The components operate as independent processes on separate physical processor cores, allowing for parallel simulation and communication with other components via MPI communication. Hsieh et al. [11] introduced a scalable execution-driven simulation framework for the HPC system by integrating gem5 into the SST, allowing for cycle-accurate simulations of the processing units. In addition, dist-gem5 [12] provides a distributed GEM5 version that supports only ARM-based processors and a simplistic network model. On the other hand, other simulators (e.g., [13, 14]) focus on the simulation of real networks using the OMNeT++/OMNEST simulation frameworks making assumptions about the applications by using models of computation. Our approach goes beyond those approaches by modeling successfully both the processing and network aspects of HPC applications while fully supporting RISC-V architectures and peripherals.

There are also several simulators [15–17] which can simulate multi-core X86, ARM, and RISC-V machines with different NoC configurations supporting different tradeoffs between accuracy and simulation speed. However, the notion of relaxed synchronization, in those simulators, is different from the one used in this work; in our approach, relaxed synchronization is applied in the intercommunication between the processing nodes (through external networks), while the above simulators employ relaxed synchronization within the multi-core CPUs only and cannot simulate interconnected parallel and/or distributed systems of systems.

Furthermore, authors in [18] propose an approach to simulate HPC systems by simulating only one node using a traditional execution-driven full-system simulator and a message emulation environment. However, their approach has the following drawbacks: (1) they generate huge traces that cannot easily be handled, (2) they lack synchronization and data exchange mechanisms, and (3) they require pre-execution of the target HPC application in an existing HPC system, while they target only ARM-based systems.

There is also a wide variety of alternative simulators focusing on simulating the interconnection networks of data centers and HPC systems. Specifically, there are simulators supporting high levels of detail, which cannot, though, simulate large-scale simulations, such as SST (already mentioned) or NS-3 [19], while other simulators, such as INRFlow [20] or GridSim [21] simulate the network at the flow level instead of at the packet level. As a result, they provide high simulation speeds at the cost of lower accuracy (e.g., since they are not modeling concrete congestion situations). Another fundamental aspect of these simulators is that they can generate traffic loads similar to real systems. In this vein, FOGsim [22] can simulate scenarios using both synthetic traffic and real traffic based on traces generated by parallel applications. The main limitation of this simulator is that it is only designed for DragonFly-type networks; thus, it cannot simulate different network topologies.

Table 1 Comparative analysis of the most widely used HPC simulation tools

Simulation tool	Simulator type	Processing support	Network support
SIM-MPI	Trace-driven	✗	✓
LogGOPSim	Trace-driven	✗	✓
SimHPC	Execution-driven	X86	✓
dist-gem5	Cycle-accurate	X86, ARM	✓(only simple switches)
SST + gem5	Cycle-accurate	X86, ARM	✗
Sniper	Event-driven	ARM	✗
Coyote	Cycle-accurate	X86	✗
NS-3	Event-driven	✗	✓
INRFlow	Event-driven	✗	✓
GridSim	Event-driven	✗	✓
FOGsim	Event-driven	✗	✓
SAURON	Trace-driven	✗	✓
COSSIM-HPC	Cycle-accurate	ARM, RISC-V	✓

The SAURON simulator [4] is orthogonal to those simulators, maintaining a high level of detail, running simulations at the packet level, keeping track of statistics at the flow level, and modeling different switch architectures accurately. SAURON can simulate networks with more than 100,000 nodes, supporting several switch architectures (input- and output-buffered), flow-control techniques (credit-based and stop&go), a wide range of topologies (direct, indirect, hierarchical, etc.), routing algorithms (oblivious, deterministic and adaptive), congestion control, QoS, and power management techniques. Indeed, thanks to the multiple functionalities offered by SAURON, it is possible to model different network technologies, such as InfiniBand [23] or BXI [24]. Moreover, as previously stated, SAURON is compatible with the VEF Traces framework. This framework allows us to characterize the MPI calls executed by the applications run in HPC systems and record this information into traffic traces. The VEF traces store only the information necessary to replicate the traffic that will later be injected into the network. Alternative options to VEF traces, such as [25–28] profile additional aspects related to parallel applications execution, such as the communication among CPU, caches, main memory, or I/0, or the data to be included into the packet payloads. Consequently, the generated trace files are difficult to handle, which makes these tools not suitable for generating traces only based on the communication operations that MPI-based applications perform in large-scale interconnection networks.

To summarize, to the best of the authors' knowledge, no open-source integrated solution exists that can handle the simulation of actual HPC systems, including their complete software stack and network dynamics. Specifically, our approach efficiently integrates well-established simulation subsystems, in a single framework, that works in a transparent to the user way, in a fully distributed manner, allowing accurate modeling and simulation of both ARM- and RISC-V-based HPC systems interconnected with different network technologies.

To further emphasize our novelty, Table 1 provides a comparative analysis of widely used HPC simulation tools. As summarized in Table 1, prior tools are typically either (i) trace-driven or network-only simulators that lack full-system CPU/OS fidelity, or (ii) execution/cycle-accurate simulators that are ISA-limited and/or monolithic, thus constraining multi-node studies. In contrast, COSSIM-HPC uniquely couples cycle-accurate full-system simulation of both ARM and RISC-V nodes (running unmodified OS/MPI stacks) with real network models, while executing each node in parallel across hosts via HLA/CERTI. This integration positions our framework to address research topics that were previously difficult to study in a single environment, including: cross-ISA analysis of MPI/collective behavior and CPU–network co-design. Finally, by allowing distributed execution of many cycle-accurate nodes, COSSIM-HPC enables early-stage evaluation of not-yet-available CPUs and interconnects under realistic software stacks.

3 Simulation framework overview

Figure 1 shows a general overview of the simulation subtools that we have efficiently integrated under a single framework. First of all, the COSSIM simulator [1] efficiently utilizes well-established simulation subsystems allowing accurate development of both ARM- and RISC-V-based HPC systems. Gem5 [29] is used for the simulation of the processing components of each node in the system, while OMNeT++ [30] is employed to simulate the networking infrastructure between those nodes. To unify the entire framework and establish a common notion of time, COSSIM employs the IEEE1516 HLA architecture [31] through the open-source CERTI package [32] as described in detail in Sect. 4.

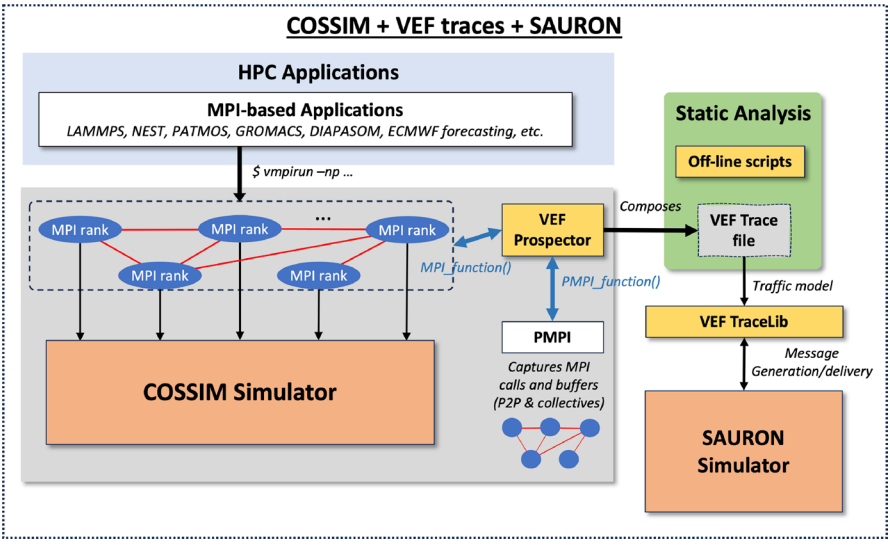


Fig. 1 Top-level view of the COSSIM-HPC framework

The VEF traces framework [5, 33] comprises several tools that allow developers to model network workloads of MPI applications. One of these tools is the VEF-Prospector, which captures the application MPI calls and stores them in trace files using a special format (i.e., VEF trace format). These traces can be used to reproduce the application's behavior in network simulators whenever they support VEF traces through the `TraceLib` library. This library can be integrated with virtually all network simulators, and it is in charge of reading the VEF trace and generating the corresponding messages, which are inserted in the simulated NICs so that the simulator injects them into the network. This library is also responsible for collecting the packets received by the end-node NICs and finishing the communication operations. Moreover, `TraceLib` permits running several applications (i.e., traces) simultaneously in the same simulation. It also allows configuring the task mapping of jobs to nodes and choosing a different implementation of a given MPI collective communication algorithm. Most importantly, the VEF traces framework allows network simulators to reproduce the application behavior in a completely agnostic manner. That is, neither the end-node architecture nor the timestamps of the system in which the VEF traces are collected are stored in the traces. Instead, the files contain only data describing the communication operations (e.g., source, destination, prior message dependency, etc.).

Finally, the SAURON simulator is based on the OMNeT++ framework (currently it utilizes OMNeT++ v6.0). During its development, several interconnection architectures have been implemented. Initially, it was designed to support only Input-Queued (IQ) switches [34], such as those used in InfiniBand, and fixed-size packets. Later, to support more intercommunication architectures, variable packet sizes, Priority Flow Control (PFC) [35], and Combined Input–Output-queued (CIOQ) switch architecture [36] were also implemented. So now, a number of network technologies, such as InfiniBand and BXI, can be seamlessly simulated. Finally, Static Queuing Schemes (SQS) and Congestion Control (CC) mechanisms have also been fully supported. Regarding the latter, FECN/BECN [37] (as defined in InfiniBand specifications) and DCQCN [38], both CC mechanisms based on injection throttling, have been modeled, as well as AccuRATE [39]. The supported topology options have also been increased, from the standard Fat-Trees, tori, or mesh, to state-of-the-art proposals such as the Megafly/DragonFly+. Traffic generation is performed through synthetic traffic or VEF traces using the VEF-TraceLib library.

As described in detail in the next sections, the extended COSSIM and the VEF/SAURON subsystems have all been tightly connected, allowing for efficient simulation of different network configurations using SAURON, under traffic patterns collected by the VEF traces toolset, which has been tightly integrated with the COSSIM-HPC simulator. The end result is a complete toolset allowing the seamless yet very accurate simulation of HPC systems with CPUs and networks that have not been implemented yet (only modeled).

4 COSSIM-HPC

In this section, we introduce the extended version of COSSIM, focusing on the enhancements/modifications developed to efficiently support RISC-V architectures, HPC runtime environments, and the VEF framework.

Figure 2 illustrates the COSSIM-HPC simulator with all its components, interfaces, and modifications. Multiple instances of a node simulator module (i.e., a gem5-based module) are required for the efficient simulation of the numerous processing nodes of an HPC system. The network that binds together the different nodes is simulated by the network simulation module (i.e., an OMNeT++-based module). The older version of COSSIM was supporting only ARM multi-core processors, while in this work we extend the COSSIM functionality to support multi-core RISC-V processors running a full OS (Ubuntu 22.04) and PCI peripherals. There is also an augmented version of COSSIM which can also simulate custom co-processors implemented with a High Level Synthesis Flow (HLS) such as the one described in [40, 41].

In addition, VEF-Prospector is successfully integrated into our simulator which captures the application MPI calls and stores them in trace files using a special format (i.e., VEF). Hence, the MPI traffic generated in COSSIM-HPC can be used to feed other network simulators and to extend the network protocol modeling capabilities.

Figure 2 illustrates the primary inputs that a user has to provide to the overall system and the primary outputs of the simulation process. Specifically, the System Configuration (number of CPUs, memory subsystem, peripherals, etc.) has to be defined either using a configuration file or the supplied graphical interface. Furthermore,

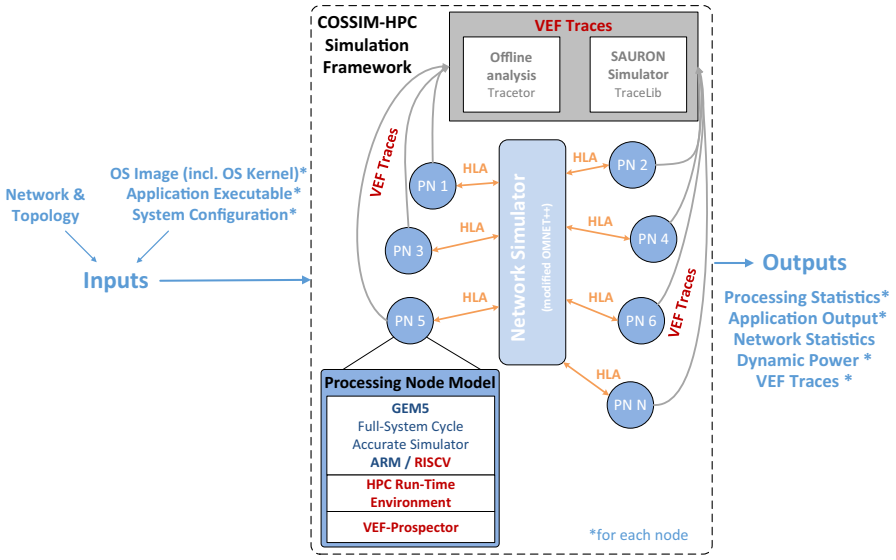


Fig. 2 COSSIM-HPC framework; newly developed modules are denoted by the red color

image files of the OSes that will be executed on the nodes of the simulated platform have to be provided by the user. An image file includes the application executable and all the libraries that are required. In addition, the first-stage bootloader and Linux Kernel must be specified. Finally, through a network and topology description, the user can define a network topology (e.g., distance between nodes, topology, mobility between nodes) and describe the interconnection channels and the selected network protocols.

The Simulation framework output comprises the following: (a) Processing & Network statistics containing all the measured metrics of the simulation process (e.g., clock ticks, real-time execution, cache misses, number of packets sent, number of packets dropped, delay of received packets), (b) the output of the actual application that is executed, and (c) VEF Traces from each processing node as well as for the network(s) utilized.

4.1 Extending RISC-V model of gem5

The main goal of our work was to first extend the baseline gem5 RISC-V system to support PCI interconnections so as to be able to simulate any PCI-compatible device. The hardware configuration developed, as an extension of the RISC-V gem5 platform, is depicted in Fig. 3. We utilize the bus subsystem, CPU, HiFive Platform,

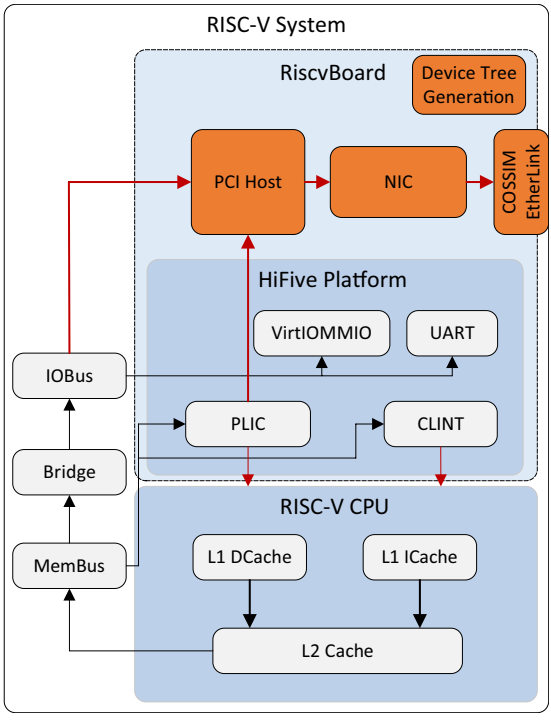


Fig. 3 PCI integration in RISC-V model

and RISCVBoard modules from the conventional gem5 and we added the modules which are denoted by the orange boxes.

In more detail, in gem5, system configurations are organized into container classes called platforms. A Platform class is a hierarchical extension of a parent class that provides a standardized set of utility functions and peripherals that can be used to tailor the configuration to a particular board or system. PC is the common Platform class in X86, whereas RealView is the common Platform class in ARM. For RISC-V, the HiFive platform has recently been introduced, within gem5, which aligns with SiFive's HiFive series of boards. In our work, and to be able to evaluate the accuracy of our approach as demonstrated in the results section, the memory map conventions, and peripheral addresses are fully in line with the specifications outlined in the gem5 RISCVMatchedBoard which is based on SiFive's Freedom U740 multi-core processor. However, our work can easily be extended and used by other SoCs as long as the memory map and peripheral address tables are altered accordingly.

HiFive platform contains a Core Local Interrupt Controller (CLINT), Platform-Level Interrupt Controller (PLIC), UART, and VirtIOMMIO. All models are connected to either a memory bus (MemBus) or IO bus (IOBus), and these buses are interfaced through both a system bridge and IO bridge. The CLINT module manages software and timer interrupts by utilizing a memory-mapped input/output (MMIO) interface. UART and VirtIOMMIO are fundamental components for a fully functional simulated system. The UART interface offers an interactive command line terminal, whereas the VirtIOMMIO provides a copy-on-write root filesystem where operating system binaries and workload scripts are stored. The PLIC controller effectively manages the reception of concurrent interrupt signals originating from peripheral devices. The goal of the PLIC is to receive all external interrupt signals, order them by priority, and then delegate them to an available hart for handling.

To support multiple interconnected RISC-V nodes, we added a PCI host model which is attached to the generic RISC-V system. Specifically, we implemented a GenericRiscvPciHost object that inherits the methods of the base gem5 GenericPciHost object. The main purpose of the custom PCI host is to map the correct source interrupt number from the PCI host to the RISC-V Platform-Level Interrupt Controller (PLIC). In our implementation, the base interrupt source number is added to the PCI interrupt number; thus, the base source number for the PCI host is 0x10, and the source numbers for the PCI interrupts are 0x11, 0x12, 0x13 and 0x14 (additional interrupts can be easily added for other PCI devices).

The Host-PCI bridge is responsible for the conversion of CPU requests into PCI bus transactions. Additionally, it facilitates the conversion of requests originating from the PCI bus hierarchy into memory requests. Both PCI endpoints and bridges expose three address spaces to device drivers and enumeration software. The *Configuration space* is used by PCI devices to map their configuration registers, while PCI devices expose two additional address spaces namely the *memory space* and the *I/O space*. Device drivers utilize both memory and I/O spaces as means of establishing communication with a specific device. *Memory space* refers to address locations that can be accessed through memory-mapped I/O (MMIO) using regular memory

load and store instructions. On the other hand, *I/O space* refers to the address locations that can be accessed through port-mapped I/O using special instructions.

Due to the absence of distinction, within gem5, between requests pertaining to configuration, I/O, and memory space, it becomes necessary to allocate distinct configuration, I/O, and memory ranges for the simulated PCI devices. In our implementation, we used the address ranges from the widely used ARM Vexpress gem5 V1 platform; 16MB (address range 0x2f000000–0x2fffffff), 256MB (address range 0x30000000–0x3fffffff), and 1024MB (address range 0x40000000–0x7fffffff) are assigned for the I/O space (pci pio), PCI configuration space (pci conf), and Memory space (pci mem), respectively. DRAM is mapped from 2GB to 512GB while all PCI memory regions are uncacheable. Another challenge that needed to be addressed was the right generation of the Device Tree so that the operating system's kernel can use and manage the newly added PCI component.

To provide a more easily used RISC-V simulation environment we ported (OpenSBI) [42] bootloader and a Linux kernel binary making the bootloader and the kernel binaries completely independent. OpenSBI is a reference implementation of RISC-V SBI (Supervisor Binary Interface), and it can act as a first-stage bootloader setting up the environment before jumping to the start of the main runtime/OS. In our configuration, after the first-stage booting is done by OpenSBI (which is located at 0x80000000), it jumps to the instruction at the address 0x80200000 which is located in the Linux kernel. Finally, the device tree is located at 0x87e00000 as presented in Table 2 which summarizes the full-system memory map. It should be noted that all PCI memory regions are assigned as uncacheable; this is necessary for the proper functioning of all PCI peripherals as it ensures that memory requests to these devices will not be cached from RISC-V Physical Memory Attribute (PMA) Checker.

The only network interface card that has been implemented, and verified in the publicly accessible repositories of gem5 is an Ethernet adapter based on the Intel 8254x which is a PCI gem5 network device using the e1000 Linux driver. To extend our base simulation framework, we modified gem5 to support the Intel 8254x

Table 2 Memory map of gem5 RISC-V full system

Device	Address range
PLIC	0x0c000000:0x0fffffff
UART	0x10000000:0x1000000f
CLINT	0x02000000:0x0200bfff
PCI	0x2f000000:0x7fffffff
pci pio	0x2f000000:0x2fffffff
pci conf	0x30000000:0x3fffffff
pci mem	0x40000000:0x7fffffff
RAM	0x80000000:0xffffffff
OpenSBI Bootloader	0x80000000:0x801fffff
Linux Kernel	0x80200000:0x86ffffff
Device Tree	0x87e00000:0x8fffffff

network card, with arbitrary link speeds, thus allowing RISC-V CPUs' PIO ports for PCI devices to be connected to the master ports of the IOBus, while the DMA ports can be connected to the slave ports of the IOBus master.

4.2 HPC runtime environment

Our extended system can simulate HPC parallel applications/programs in combination with any OS including the required libraries (e.g., MPI). To facilitate the use of our simulator by any interested HPC stakeholder, Ubuntu 22.04 LTS gem5 compatible images, utilizing Linux Kernels v6.5.5 and bootloaders, have been created and configured for both ARM and RISC-V architectures and are available in open-source.

To fully simulate a parallel program/application on a network of computers via MPI, an MPI Cluster is set up. In this respect, the rsh server has also been successfully imported into all gem5 nodes to allow the launching of commands on remote nodes. In addition, MPICH v4.0 has been imported and successfully evaluated using both simple and collective communication MPI routines, while a *hosts* file (/etc/hosts) is configured by each gem5 node OS to map hostnames to IP addresses. Finally, a *host_file* is created to declare the number of MPI ranks that will be executed in each COSSIM node. All those configurations are also available in open source together with the corresponding instructions.

Our framework is also capable of efficiently communicating with other network simulators (i.e., SAURON) through VEF traces so as to allow for the exploitation of more complex and extreme scale network topologies. Specifically, the VEF-prospector tool was successfully ported in both ARM & RISC-V simulated systems producing accurate VEF traces.

Code Segment 1 presents the MPI runtime environment for HPCG execution on 3 gem5 nodes using 16 MPI ranks per node, which is identical to a real system as described in detail in the next section. It should be noted that this script has to be executed only in the first gem5 node of the MPI cluster distributing the MPI tasks to the other nodes automatically through the rsh server.

The user can seamlessly modify the COSSIM-HPC configuration file using the developed Graphical User Interface (GUI), so as to accurately model the computing node of the target HPC system, including the processor ISA and microarchitecture, memory system, OS, kernel, parallel environment (MPI library), and the targeted network. The powerful, yet simple, new GUI provides easy simulation setup, execution, and visualization of the reported measurements.

Code Segment 1 MPI runtime environment for 3 gem5 nodes

```
1: hostname node0 #Define the hostname for node0 (localhost)
2: rsh 192.168.0.3 hostname node1 #Define the hostname for node1
3: rsh 192.168.0.4 hostname node2 #Define the hostname for node2
4: #Define the VEF-prospector path
5: export PATH=$PATH:/opt/vef_prospector/bin/
```

Code Segment 1 MPI runtime environment for 3 gem5 nodes

```
6: export LD_LIBRARY_PATH=/opt/vef_prospector/lib/
7: echo "node0:16 node1:16 node2:16" > host_file
8: m5 resetstats #reset the gem5 statistics before mpi execution
9: #Run the HPCG benchmark on 3 nodes (16 MPI ranks/node) using VEF Traces tool
10: vmpirun -launcher rsh -n 48 -f host_file ./hpcg
11: m5 dumpstats #dump the gem5 statistics
```

5 Validation and performance analysis

This section presents the experimental results that validate the accuracy and scalability of COSSIM-HPC using the widely used HPCG and LAMMPS HPC benchmarks. In all experiments, we execute our simulations on one or more servers with AMD Ryzen 9 7950X running at 4.50 GHz with 128 GB of RAM.

5.1 COSSIM-HPC accuracy

To verify the correct behavior and accuracy of the COSSIM-HPC runtime environment the widely used HPCG v3.1 benchmark is ported successfully in simulated ARM- and RISC-V-based parallel systems and the reported metrics are compared with the actual metrics measures on real systems; ARM-based Dibona-TX2 cluster [43] and RISC-V-based HiFive Unmatched board [44] with SiFive Freedom U740 SoC. Note that the full-system mode of gem5 has been used with the exact same MPI libraries and OS for both sets of experiments.

First of all, we configure COSSIM-HPC to simulate from 2 up to 64 ARMv8 cores @2GHz & DDR4 memory (identical to the Dibona Cluster). In Fig. 4, we can see the HPCG performance comparison (in GFLOPS) as reported by the COSSIM-HPC simulator and measured on the Dibona Cluster using from 2 to 64 ARMv8

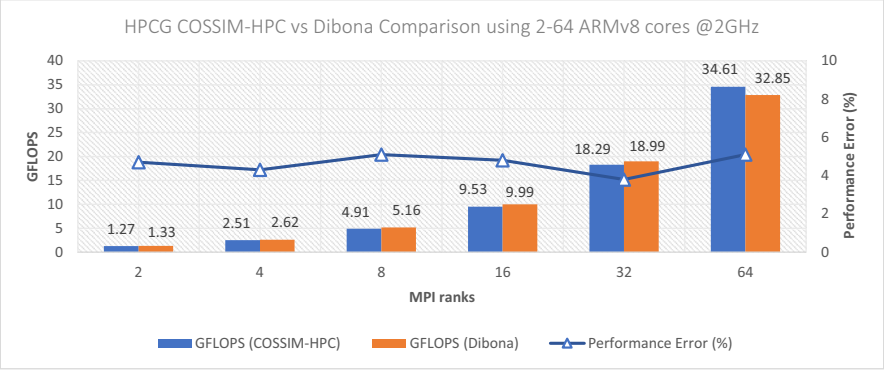


Fig. 4 HPCG COSSIM versus Dibona comparison using 2–64 ARMv8 cores

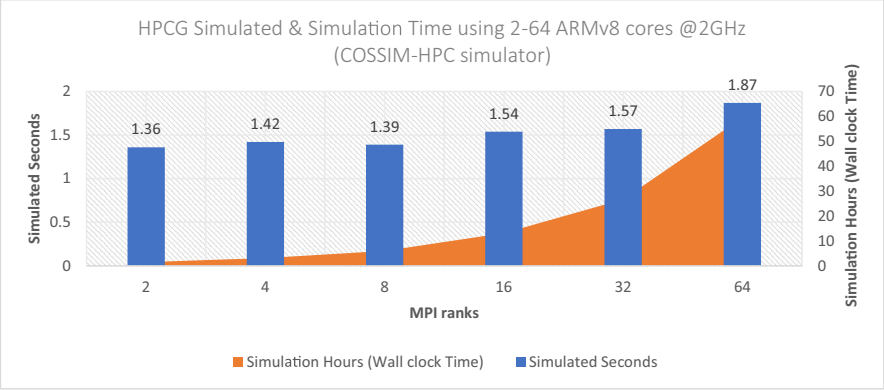


Fig. 5 HPCG simulated and simulation time using 2–64 ARMv8 cores

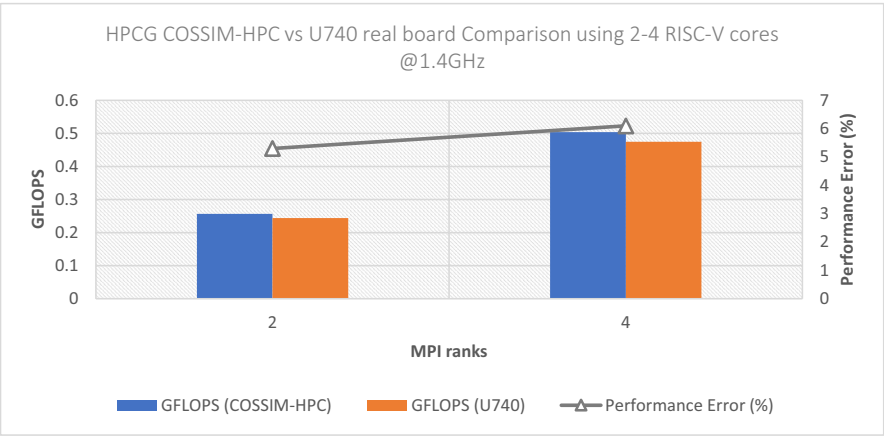


Fig. 6 HPCG COSSIM versus HiFive board comparison using 2–4 SiFive U740 cores

cores. On the left *y* axis, we show the GFLOPS, while on the right *y* axis, we show that the performance error between the simulated and the real system is always below 5%; the performance error is calculated as the ratio of the real system's performance (in GFLOPS) to that of the COSSIM-HPC simulator. It should be noted that the functional error is zero in all experiments (i.e., there are no differences in the final results between the simulated and the real system).

In Fig. 5, we can see the HPCG Simulated Time and Simulation Time (Wall Clock Time) using 2–64 ARMv8 cores @2 GHz (in the COSSIM-HPC simulator). As demonstrated, while the simulated seconds remain approximately constant, the simulation hours increase proportionally with the doubling of MPI ranks due to the corresponding increase in total simulated instructions. In more detail, to maintain an HPCG execution of 1 s, more MPI messages and CPU instructions are generated as the MPI ranks are doubled.

In addition, we configure COSSIM-HPC to simulate from 2 up to 4 RISC-V cores per gem5 instance with the same architectural characteristics as SiFive's U740 boards. Specifically, we configure the CPU to run at 1.4 GHz with 16 GB DDR4 memory, 32 kB L1 instruction and data caches, and 2MB L2 cache size. In Fig. 6, we can see the performance comparison (in GFLOPS) as reported by the COSSIM-HPC simulator and measured on the SiFive U740 system. In the left y-axis, we can see the GFLOPS, while in the right y-axis, we can see the performance error between the simulated and the real system; the performance error remains below 7%, while the functional error is zero (i.e., there are no differences in the final results between the simulated and the real system). In both cases, HPCG is used with compiler optimizations applied.

5.2 COSSIM-HPC scalability

Apart from the widely used HPCG benchmark, we port the LAMMPS molecular dynamics HPC benchmark on the COSSIM-HPC simulator on both ARM & RISC-V architectures to measure and verify COSSIM-HPC's scalability. We configure each gem5 node to simulate from 2 up to 8 ARM & RISC-V cores with the same architectural characteristics, so as to compare the performance reported and the VEF traces generated. Specifically, we configure each gem5 node processor on both architectures @1.4 GHz with 16 GB DDR4 memory, 32 kB L1 instruction and data caches, and 2MB L2 cache size (similar to SiFive U740 SoC). In our study, we create a star network topology in OMNeT++, characterized by a central gem5 node (Node0) that is connected to all other gem5 nodes through an Ethernet switch, thereby ensuring efficient and centralized management of the whole simulation environment from Node0. Finally, in all LAMMPS experiments, 10 steps with 32,000 atoms have been used.

Figures 7 and 8 present the performance reported when simulating the LAMMPS benchmark on RISC-V and ARM CPUs from 1 up to 16 nodes. In the single-node setup (Fig. 7), the ARM processor outperforms the RISC-V processor by

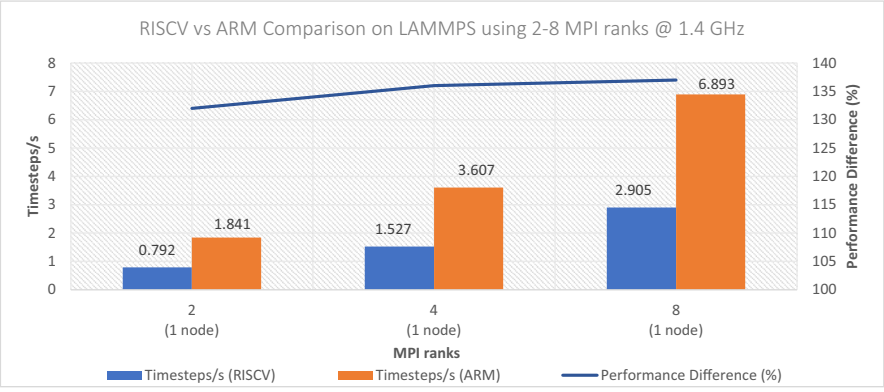


Fig. 7 RISC-V versus ARM performance on LAMMPS using 2-8 MPI ranks for 32K atoms-10 steps

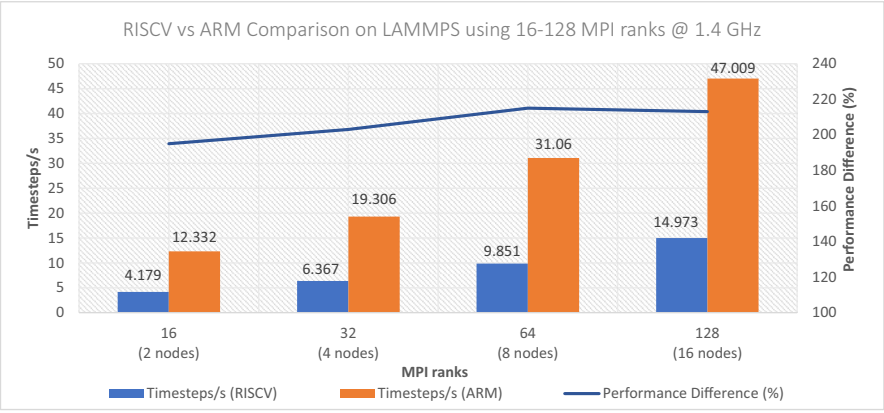


Fig. 8 RISC-V versus ARM performance on LAMMPS using 16-128 MPI ranks for 32K atoms-10 steps

approximately 140% across all ranks, while no functional errors are observed. Moving to multiple distributed COSSIM-HPC nodes (Fig. 8), the absolute performance gap for the ARM architecture increases to around 215%. As can be observed, this performance difference arises primarily in the transition from single-node to multi-node execution. Once multiple nodes are employed, the relative performance difference between ARM and RISC-V remains almost constant as the number of MPI ranks increases (from 16 up to 128 ranks), which is clearly reflected in the nearly horizontal performance-difference line. To further illustrate this effect, Fig. 9 presents the isolated speedup lines for ARM and RISC-V, showing that both architectures scale in a very similar manner. Therefore, the main bottleneck arises from the CPU side, specifically during the initialization and processing of remote connections.

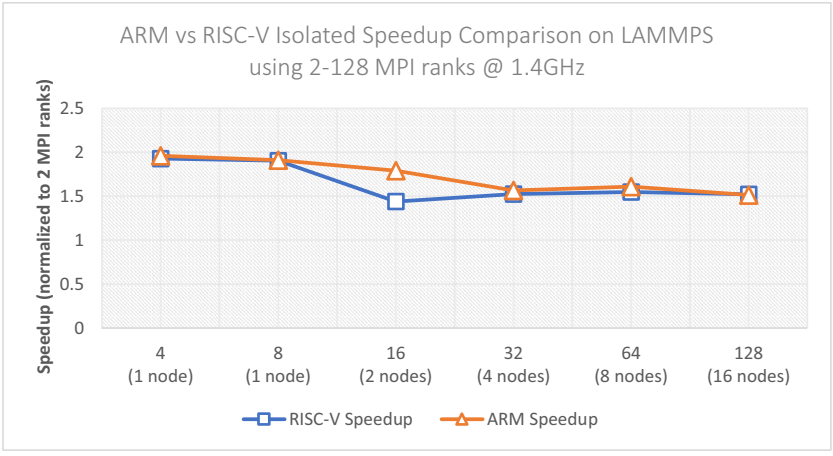


Fig. 9 ARM versus RISC-V isolated speedup comparison on LAMMPS using 16–128 MPI ranks @ 1.4 GHz

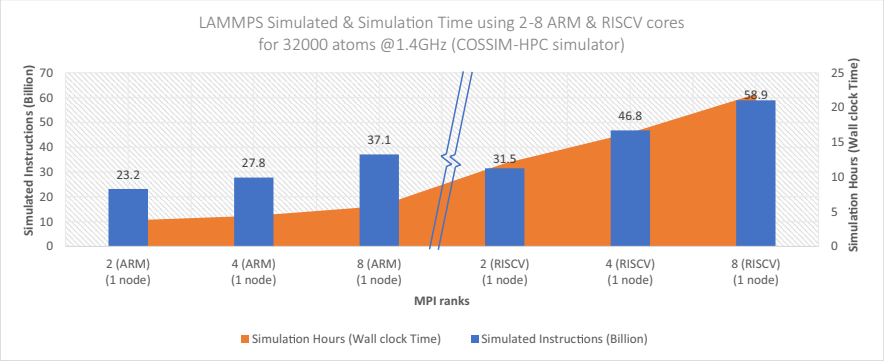


Fig. 10 LAMMPS simulated instructions versus simulation time using 2–8 MPI ranks for 32K atoms—10 steps (1 gem5 node)

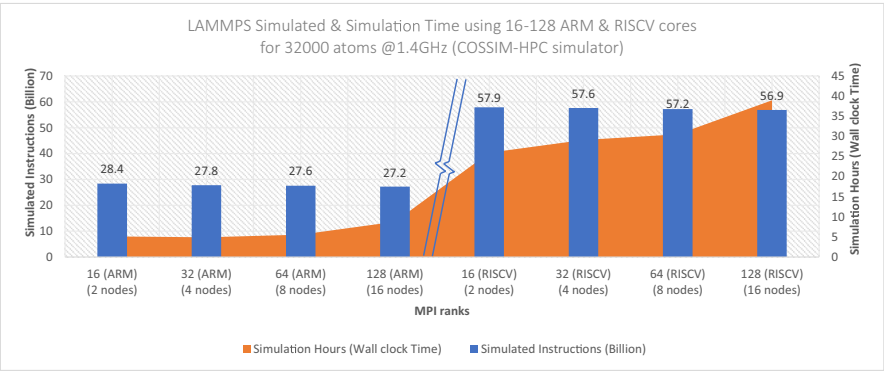
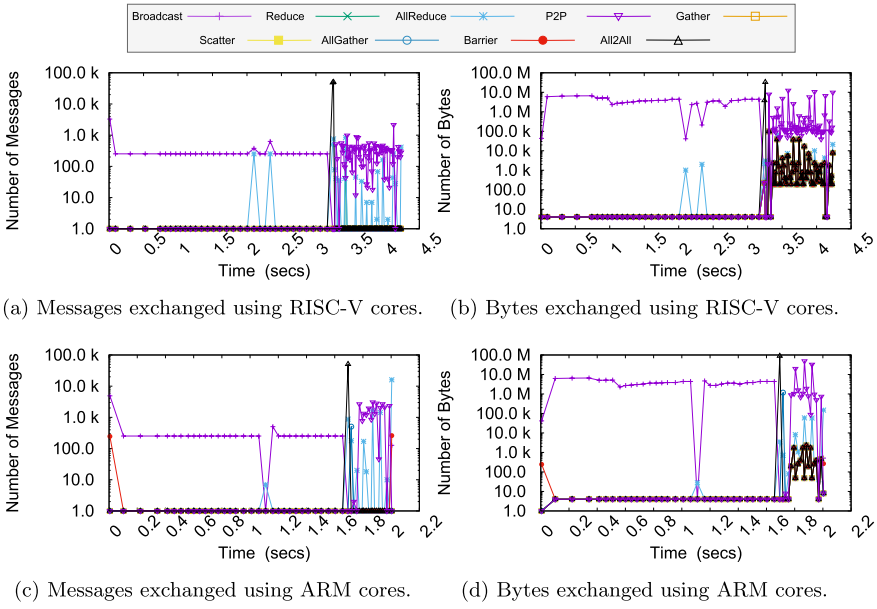


Fig. 11 LAMMPS simulated instructions versus simulation time using 16–128 MPI ranks for 32K atoms—10 steps (2 up to 16 gem5 nodes)

Figure 10 presents the LAMMPS Simulated Instructions versus Simulation Time (Wall Clock Time) using 2-8 ARM & RISC-V cores @1.4GHz in COSSIM-HPC simulator for 1 gem5 node. We can observe that the number of simulated instructions increases when doubling the MPI ranks as all MPI ranks are executed in the one gem5 instantiation; for the same reason the simulation time (wall-clock time) increases with the MPI ranks (since all MPI ranks are executed in one single-threaded gem5).

Moving to multiple gem5 nodes (Fig. 11), the simulated instructions on the central gem5 node (Node0) are almost constant when doubling the MPI ranks because the amount of total work is distributed among the MPI ranks on other gem5 nodes as the problem size is constant (32,000 atoms).[3] Finally, as Fig. 11 demonstrates,

[3] The simulated instructions on other gem5 nodes have the same behavior.



(a) Messages exchanged using RISC-V cores. (b) Bytes exchanged using RISC-V cores.

(c) Messages exchanged using ARM cores. (d) Bytes exchanged using ARM cores.

Fig. 12 LAMMPS simulation over time (seconds) in a 128-node network using end nodes with RISC-V or ARM cores

due to the fully distributed nature of COSSIM-HPC, the simulation time (wall-clock time) overhead is negligible when moving from 2 to 8 gem5 nodes since all of them are executed in parallel in different physical cores, while, in the 16 node experiment, there is a slight increase in simulation time (under 20%). Those numbers clearly demonstrate the scalability of our approach.

The observed differences of ARM over RISC-V architectures in Figs. 10 and 11 reflect inherent differences in ISA characteristics as faithfully modeled by the gem5 simulator rather than any influence from the COSSIM-HPC framework itself. It is important to emphasize that COSSIM-HPC does not modify, increase or affect the performance characteristics of gem5. The performance differences can be attributed to several key factors such as: (1) ARM's instruction set efficiency, where more complex instructions can perform multiple operations per cycle, reducing overall instruction count; (2) superior microarchitectural of ARM, including memory hierarchy interactions, cache behavior, and prefetching mechanisms. However, the performance difference (Figs. 4, 6) is consistent with gem5's accurate representation of real-world ISA behavior, as validated by our experiments showing less than 7% error when compared against actual ARM and RISC-V hardware platforms.

It should be noted that each experiment was executed three times, yielding identical performance and accuracy results across all runs. This perfect reproducibility is inherent to gem5's deterministic, cycle-accurate simulation engine, ensuring zero variance between repeated experiments under the same configuration. Consequently, statistical measures such as standard deviation or confidence intervals are not applicable, as no variation in the recorded metrics was observed.

To analyze the communication patterns and overheads further, we collected some traces using the VEF traces framework. First, the VEF-prospector tool was successfully ported to the COSSIM-HPC simulator in both ARM & RISC-V systems. Next, we used the `offline-vef-analysis` tool to produce the following plots.[4] Specifically, we obtained VEF traces for LAMMPS runs in COSSIM-HPC using 128 MPI ranks. We configured COSSIM-HPC with 16 gem5 nodes using RISC-V or ARM architectures. Figure 12 shows the number of messages and bytes that MPI operations generate over time (in seconds), according to the information recorded in the obtained VEF traces. We can see that ARM cores run the applications twice as fast as the RISC-V cores.

It can be observed in this figure that, when running an application on RISC-V-based systems, the messages are more spaced out in time, while the number of bytes exchanged is identical since the application and the OS executed are the same in all cases. The time discrepancy is, obviously, because the ARM processor has better performance than the RISC-V (see Fig. 8), or in other words, it executes more instructions per simulated second. As a result, more MPI messages and CPU Instructions are produced per second in the case of ARM architecture, even though we use the exact hardware specifications (frequency, DRAM, cache sizes).

Moreover, the communication operations of this application are dominated by Broadcast and peer-to-peer (P2P) messages, in terms of the number of messages and bytes exchanged. In addition, "All2All" and "AllReduce" operations are executed by the end of the execution. It is worth mentioning that there is a peak in the number of exchanged messages and bytes around the 3.2 s of execution for the RISC-V cores and 1.6 s for the ARM cores. This contention situation will increase the message latency in the network (or flow completion time), as it will be observed when we use these VEF traces to feed the network simulator in the next section.

5.3 Network simulations

This section analyzes the results of the experiments performed using the VEF traces generated by COSSIM-HPC, which have been used as inputs to SAURON. First, we describe the experiments' configuration and then we analyze the simulation results.

5.3.1 Experiments configuration

We have run our large-scale simulations with large and small MPI traces. For large traces (128-Task LAMMPS), we use a 128-node Real Life Fat-Tree (RLFT) topology, while for the smaller traces (64-Task HPCG), we use a 64-node Fat-Tree topology. We have run each trace with two different interconnection network architectures, which we will refer to as 'Conf #1' and 'Conf #2'. In more detail, 'Conf #2' is based on common features found in the latest generations of Ethernet-based network technologies, such as the next-generation of the BXI technology, which has

[4] Further information on the VEF Traces framework and offline analysis can be obtained from this tutorial: https://redsea-project.eu/the-vef-traces-framework/.

Table 3 Configuration for the large-scale experiments

Network parameters	Conf #1	Conf#2
Link BW (Gbps)	100	400
Link length	3 m	3 m
Link latency	5 ns/m	5 ns/m
MTU (bytes)	4096	9600
Variable packet Size	No	Yes
Flow Ctrl	Credits	PFC
Sw. Arch	IQ	CIOQ
Buff. size (KiB)	128	48
Num. VLs	1	1

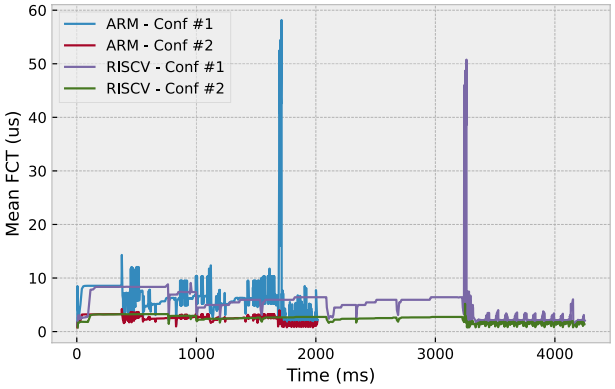


Fig. 13 Mean FCT over time for two 128 MPI rank LAMMPS VEF traces (ARM- and RISC-V-based) when using network configurations #1 and #2

been investigated in the RED-SEA project [45, 46]. These features include the use of combined input and output queuing (CIOQ) [36] switch architectures, PFC flow control [35], or the use of 400Gbps links. Moreover, variable-sized packets are another important feature of next-generation BXI technologies, which permit generating packets at network interfaces smaller in size than the network MTU (i.e., JUMBO frames of 9600 Bytes for 'Conf #2') whenever applications generate messages that fit in that size. By contrast, 'Conf #1' presents inferior characteristics relative to 'Conf #2', e.g., 100Gbps data rate, and a fixed MTU packet size, which will negatively impact certain types of traffic, as discussed later. Furthermore, 'Conf #1' employs an input-queued (IQ) [34] switch architecture and credit-based flow control [47]. Table 3 summarizes the details of these architectures. There are several examples of network technologies that mimic these features, such as the current generation of BXI networks [48] or the InfiniBand EDR generation.

Although SAURON calculates a large set of metrics regarding the interconnection network performance, in this study we focus mainly on the Flow Completion Time (FCT), which covers the time from message generation to complete arrival at

the destination, since this is, probably, the most critical factor in interconnection networks for HPC systems. Based on the FCT values, we have built a histogram demonstrating their cumulative distribution function (CDF). Moreover, as we want to observe the evolution of the network performance metrics over the simulated time, we have measured their maximum, minimum, and mean values periodically. Note that SAURON collects metrics specific to switches and NICs, such as the number of discarded packets, the number of bytes sent, usage of the port, etc., which are all very important for accurately simulating the generated traffic routed over the simulated network.

5.3.2 Simulation results

First, we examine the reported results for the LAMMPS traces. Figure 13 shows the average FCT over time for network configurations #1 and #2 when using the LAMMPS VEF traces collected through HPC-COSSIM. Figure 14 shows the CDF for the same scenario.

As shown in those figures, network configuration #2 (i.e., Conf #2) reduces the FCT for high percentiles (tail latency) on both ARM and RISC-V compared to network configuration #1 (i.e., Conf #1). Although the execution time is shorter for the ARM architecture, the FCT values are higher at some points in the simulated time, since when using the more efficient ARM processors, the interval time between generated messages is lower.

Specifically, the higher packet generation rate in the ARM case compared to the RISC-V case increases network contention and, consequently, FCT values. This is illustrated in Figs. 13 and 14, in which the mean FCT reaches 60µs and the 99th-percentile FCT (tail latency) approaches 140µs, respectively. Note that the contention observed at 1.6 s for ARM cores and 3.2 s for RISC-V cores (see Fig. 13) is due to the "All2All" communication, as described in the comments of Fig. 12. Note also that this contention does not appear when we use network configuration #2, as the provided link speed is enough to prevent contention.

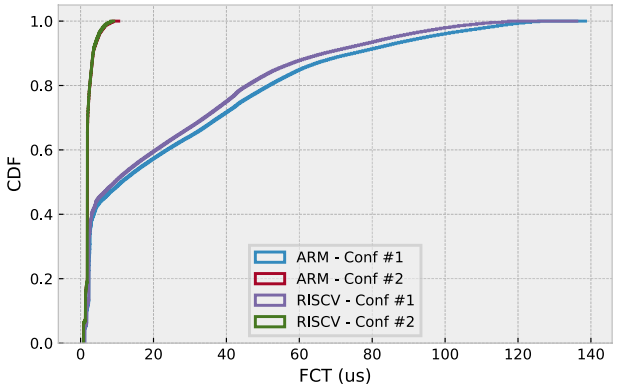


Fig. 14 CDF for the FCT values using the LAMMPS traces (ARM and RISC- V)

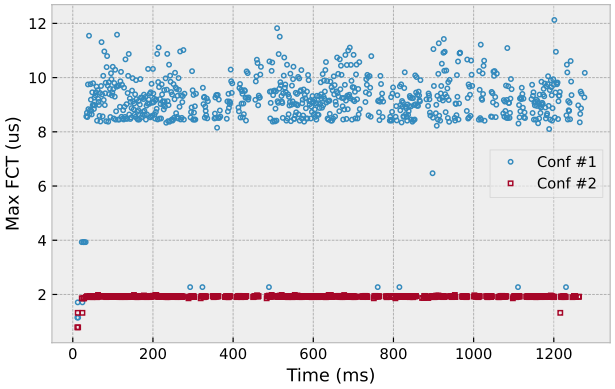


Fig. 15 HPCG trace—maximum FCT over time

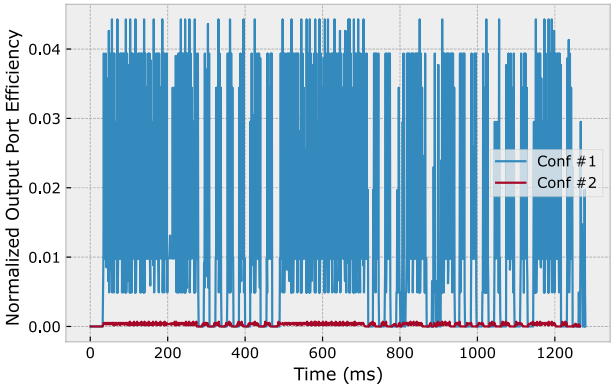


Fig. 16 HPCG trace—normalized output port efficiency

Figure 15 shows the Maximum FCT values for a 64-MPI-rank HPCG-based VEF trace, collected through COSSIM-HPC. These results reveal that, once more, the FCT obtained for Conf #1 is worse than that of Conf #2. Furthermore, note that there are larger oscillations for this metric (between 8 and 12 µs) when using Conf #1 than when using Conf #2, where the latency remains stable at 2 µs during the whole trace execution.

Figure 16 shows the Normalized output port utilization efficiency for both network configurations over time in the HPCG case. To calculate this metric, we have measured the amount of bytes sent during each interval and normalized them against the port's theoretical maximum. As shown in that figure, the normalized output port efficiency is significantly higher for Conf #1 than for Conf #2, with maximum utilization below 0.1% on average for Conf #2, while Conf #1 obtains values between 1% and 4%, with a high variability. Note that the HPCG trace generates minimal load on the network and utilizes small messages, typically smaller than the network

MTU employed in each configuration (4096 Bytes for Conf #1 and 9600 Bytes for Conf #2). Also, note that Conf #1 employs fixed-size packets and thus full MTUs (4096B) are transmitted even when the MPI-generated messages are smaller. This results in switch output ports being utilized for a longer duration for Conf #1, which increases network power consumption and execution time.

Finally, although port utilization, on average, is low in both network configurations (i.e., below 5%), there is a high variability (between 1% and 4%, see Fig. 16) for Conf #1, which correlates with the observed variability for Maximum FCT values in Fig. 15.

All those results clearly demonstrate that the integration of COSSIM-HPC, VEF Traces and SAURON allows the HPC designer to perform a design space exploration of the different processing units and the numerous network topologies/technologies/ configurations of large-scale HPC systems (SAURON can easily simulate thousands of HPC nodes) seamlessly and accurately.

6 Conclusion

This paper presents an open-source integrated simulation framework that can simulate complete heterogeneous HPC systems consisting of ARM and **RISC-V** architectures and different network technologies (e.g., InfiniBand, BXI). Specifically, we extend the COSSIM simulator to support PCI peripherals for RISC-V cores and arbitrary link speeds in the network cards while we have also created a full reference HPC runtime environment (including OS, libraries, etc) for simulating multi-node ARM and RISC-V systems. We also extended the SAURON simulator to model several interconnection network architectures commonly used in HPC systems, as well as network technologies leveraging these architectures, such as InfiniBand or BXI. Moreover we have integrated the VEF traces framework, into COSSIM which allows us to analyze communication patterns and generate traffic traces of complete MPI applications executed on yet-to-be implemented CPUs. These traces have been used as inputs to the SAURON interconnection network simulator thus allowing accurate and seamless modeling and simulation of different HPC network topologies/ architectures/technologies.

Our novel approach enables, probably for the first time, the HPC designers to simulate the complete aspects of HPC systems (i.e., Processing and Network Environment) when executing real HPC applications within a single simulation framework. As demonstrated in a number of different experiments with widely used HPC benchmarks, the presented framework produces very accurate results while the simulation times grow insignificantly when the size of the simulated HPC system increases.

Acknowledgements This work was supported in part by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "Framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union—NextGenerationEU" under grant agreement No 74967 (REDESIGN project) and in part by Vitamin-V Horizon Europe project under grant agreement No 101093062 for the authors Nikolaos Tampouratzis and Ioannis Papaefstathiou. Furthermore, this work was supported by RED-SEA, a project that receives funding from the European High-Performance Computing Joint Undertaking (JU)

under grant agreement No 955776 and the Ministry of Science, Innovation and Universities of Spain under grant PCI2021-121976 for authors Gabriel Gomez-Lopez, Miguel S anchez De la Rosa, Jesus Escudero-Sahuquillo and Pedro J. Garcia.

Author contributions N.T. and I.P. wrote the main manuscript text, while G.G. , M.S. , J.E. and P.G. prepared Sects. 3, 5 as well as contributed to Sect. 2. All authors have reviewed and agreed to the published version of the manuscript.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Tampouratzis N, Papaefstathiou I, Nikitakis A, Brokalakis A, Andrianakis S, Dollas A, Marcon M, Plebani E (2020) A novel, highly integrated simulator for parallel and distributed systems. ACM Trans Arch Code Optim 17(1):1. https://doi.org/10.1145/3378934
2. Nikolaos, Tampouratzis P, Mousouliotis I, Papaefstathiou (2023) A Novel Integrated Simulation Framework for Cyber-Physical Systems Modelling. IEEE Trans Parallel Distrib Syst 34(10) 2684-2698. https://doi.org/10.1109/TPDS.2023.3300081
3. Brokalakis, A, Tampouratzis, N, Nikitakis, A, Andrianakis, S, Papaefstathiou, I, Dollas, A (2017) An open-source extendable, highly-accurate and security aware cps simulator. In: 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 81–88. https://doi.org/10.1109/DCOSS.2017.15
4. Yebenes P, Escudero-Sahuquillo J, Garcia PJ, Quiles FJ (2013) Towards modeling interconnection networks of exascale systems with omnet++. In: 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp 203–207. https://doi.org/10.1109/PDP.2013.36
5. Andujar FJ, Rosa M, Escudero-Sahuquillo J, Sánchez J (2022) Extending the vef traces framework to model data center network workloads. J Supercomput 79:1. https://doi.org/10.1007/s11227-022-04692-0
6. Zhai J, Chen W, Zheng W, Li K (2016) Performance prediction for large-scale parallel applications using representative replay. IEEE Trans Comput 65(7):2184–2198. https://doi.org/10.1109/TC.2015.2479630
7. Hoefler T, Schneider T, Lumsdaine A (2010) Loggopsim: simulating large-scale applications in the loggops model. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. HPDC '10, Association for Computing Machinery, New York, NY, USA, pp 597–604. https://doi.org/10.1145/1851476.1851564
8. Barnwal RP, Ghosh N, Ghosh SK, Das SK (2018) Ps-sim: a framework for scalable simulation of participatory sensing data. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), pp 195–202. https://doi.org/10.1109/SMARTCOMP.2018.00091
9. Liu Y, Zhi Y-Z, Zhang X, Li H, Jiao L, Zhang P, Su Y-M, Ni Z-H, Qian D-P (2014) Simhpc: an execution-driven simulator for high-performance computers. Chin J Comput 36:738–746. https://doi.org/10.3724/SP.J.1016.2013.00738
10. Rodrigues AF, Hemmert KS, Barrett BW, Kersey C, Oldfield R, Weston M, Risen R, Cook J, Rosenfeld P, Cooper-Balis E, Jacob B (2011) The structural simulation toolkit. ACM SIGMETRICS Perform Eval Rev 38(4):37–42. https://doi.org/10.1145/1964218.1964225
11. Hsieh M, Pedretti K, Meng J, Coskun A, Levenhagen M, Rodrigues A (2012) SST + gem5 = a scalable simulation infrastructure for high performance computing, pp 196–201. https://doi.org/10.4108/icst.simutools.2012.247745
12. Mohammad A, Darbaz U, Dozsa G, Diestelhorst S, Kim D, Kim NS (2017) dist-gem5: distributed simulation of computer clusters. In: 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp 153–162. https://doi.org/10.1109/ISPASS.2017.7975287

13. Minkenberg C, Denzel W, Rodriguez G, Birke R (2012) In: Bangsow S (ed), End-to-End modeling and simulation of high- performance computing systems, Springer, Berlin, pp 201–240. https://doi.org/10.1007/978-3-642-28777-0_11
14. Hurst W, Ramaswamy S, Lenin R, Hoffman D (2010) Development of generalized HPC simulator. In: Janowski T, Mohanty H (eds), Distributed computing and internet technology, Springer, Berlin, pp 176–179. https://doi.org/10.1007/978-3-642-11659-9_18
15. Carlson TE, Heirman W, Eeckhout L (2011) Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In: SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, pp 1–12.https://doi.org/10.1145/2063384.2063454
16. Zaourar L, Benazouz M, Mouhagir A, Jebali F (2021) Sassolas: multilevel simulation-based co-design of next generation HPC microprocessors. In: 2021 International workshop on performance modeling, benchmarking and simulation of high performance computer systems (PMBS), pp 18–29. https://doi.org/10.1109/PMBS54543.2021.00008
17. Perez B, Fell A, Davis JD (2021) Coyote: an open source simulation tool to enable risc- v in hpc. In: 2021 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp 130–135. https://doi.org/10.23919/DATE51398.2021.9474080
18. Lin F, Liu Y, Wang X, Gai X (2023) Leveraging simulation of high performance computing systems with node simulation using architecture simulator. CCF Trans High Perform Comput 5:1. https://doi.org/10.1007/s42514-023-00173-9
19. Riley GF, Henderson TR (2010) In: Wehrle K, Güneş M, Gross J (eds), The ns-3 network simulator, Springer, Berlin, pp 15–34. https://doi.org/10.1007/978-3-642-12331-3_2
20. Navaridas J, Pascual JA, Erickson A, Stewart IA, Luján M (2019) INRFlow: an interconnection networks research flow-level simulation framework. J Parallel Distrib Comput 130:140–152. https://doi.org/10.1016/j.jpdc.2019.03.013
21. Quinson M (2009) Simgrid: a generic framework for large-scale distributed experiments. In: 2009 IEEE Ninth International Conference on Peer-to-Peer Computing, pp 95–96. https://doi.org/10.1109/P2P.2009.5284500
22. FOGsim by fuentesp. https://fuentesp.github.io/fogsim/. Accessed on 27 May 2024
23. InfiniBand Trade Association.: InfiniBandTM Architecture Specification Volume 1 - Release 1.3 (2015). https://www.infinibandta.org/ibta-specification/
24. Derradji S, Palfer-Sollier T, Panziera J-P, Poudes A, Atos FW (2015) The BXI interconnect architecture. In: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects (HOTI), pp 18–25. https://doi.org/10.1109/HOTI.2015.15
25. Structural Simulation Toolkit (SST) DUMPI trace library. Accessed 26 Aug 2025. https://github.com/sstsimulator/sst-dumpi
26. Knüpfer A, Brunst H, Doleschal J, Jurenz M, Lieber M, Mickler H, Müller MS, Nagel WE (2008) The Yampir performance analysis tool-set. In: Resch M, Keller R, Himmler V, Krammer B, Schulz A (eds) Tools for high performance computing, Springer, Berlin, pp 139–155. https://doi.org/10.1007/978-3-540-68564-7_9
27. Knüpfer A, Rössel C, Mey Da, Biersdorff S, Diethelm K, Eschweiler D, Geimer M, Gerndt M, Lorenz D, Malony A, Nagel WE, Oleynik Y, Philippen P, Saviankou P, Schmidl D, Shende S, Tschüter R, Wagner M, Wesarg B, Wolf F (2012) Score-P: a joint performance measurement run-time infrastructure for periscope,Scalasca, TAU, and Vampir. In: Brunst H, Müller MS, Nagel WE, Resch MM (eds) Tools for high performance computing 2011, Springer, Berlin, pp 79–91
28. Wylie BJN, Giménez J, Feld C, Geimer M, Llort G, Mendez S, Mercadal E, Visser A, Garcéa-Gasulla M (2025) 15+ Years of joint parallel application performance analysis/tools training with Scalasca/Score-P and Paraver/Extrae toolsets. Future Gener Comput Syst 162:107472–13. https://doi.org/10.1016/j.future.2024.07.050
29. The GEM5 Simulator. Accessed 26 Aug 2025. http://gem5.org/
30. OMNeT++ Discrete Event Simulator. Accessed 26 Aug 2025. https://omnetpp.org/
31. IEEE 1516-010-Standard for modeling and simulation high level architecture-framework and rules. Accessed 26 Aug 2025. https://standards.ieee.org/findstds/standard/1516-2010.html
32. CERTI Project. Accessed 26 Aug 2025. http://savannah.nongnu.org/projects/certi
33. Andújar FJ (2015) VEF traces: a framework for modelling MPI traffic in interconnection network simulators. In: The 1st IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era, Chicago, IL, USA, pp 841–848. https://doi.org/10.1109/CLUSTER.2015.141

34. Karol M, Hluchyj M, Morgan S (1987) Input versus output queueing on a space-division packet switch. IEEE Trans Commun 35(12):1347–1356. https://doi.org/10.1109/TCOM.1987.1096719
35. 802.1Qbb: IEEE standard for local and metropolitan area networks—virtual bridged local area networks—amendment: priority-based flow control (2011). https://1.ieee802.org/dcb/802-1qbb/ Accessed 2018-09-23
36. Awan A, Venkatesan R (2004) Design and implementation of enhanced crossbar CIOQ switch architecture. In: Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513), vol 2, pp 1045–10482. https://doi.org/10.1109/CCECE.2004.1345297
37. Gran EG, Eimot M, Reinemo S-A, Skeie T, Lysne O, Huse LP, Shainer G (2010) First experiences with congestion control in infiniband hardware. In: 2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp 1–12. https://doi.org/10.1109/IPDPS.2010.5470419
38. Zhu Y, Eran H, Firestone D, Guo C, Lipshteyn M, Liron Y, Padhye J, Raindel S, Yahia MH, Zhang M (2015) Congestion control for large-scale RDMA deployments. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. SIGCOMM '15, Association for Computing Machinery, New York, NY, USA, pp 523–536. https://doi.org/10.1145/2785956.2787484
39. Giannopoulos D, Chrysos N, Mageiropoulos E, Vardas G, Tzanakis L, Katevenis M (2018) Accurate congestion control for RDMA transfers. In: 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), pp 1–8. https://doi.org/10.1109/NOCS.2018.8512155
40. Tampouratzis N, Papaefstathiou I (2018) A novel, simulator for heterogeneous cloud systems that incorporate custom hardware accelerators. IEEE Trans Multiscale Comput Syst 4(4):565–576. https://doi.org/10.1109/TMSCS.2018.2879601
41. Georgopoulos K, Chrysos G, Malakonakis P, Nikitakis A, Tampouratzis N, Dollas A, Pnevmatikatos D, Papaefstathiou Y (2016) An evaluation of VIVADO HLS for efficient system design. In: 2016 International Symposium ELMAR, pp 195–199. https://doi.org/10.1109/ELMAR.2016.7731785
42. RISC-V Open Source Supervisor Binary Interface (OpenSBI) (2025) GitHub. Accessed 26 Aug 2025. https://github.com/riscv-software-src/opensbi
43. Banchelli F, Garcia M, Josep M, Mantovani F et al (2019) MB3 D6.9—performance analysis of applications and mini-applications and benchmarking on the project test platforms. Technical report. Accessed 5 Sep 2024
44. HiFive Unmatched Board. Accessed 26 Aug 2025. https://www.sifive.com/boards/hifive-unmatched
45. Gomez ME, Sahuquillo J, Biagioni A, Chrysos N et al (2024) Red-sea project: towards a new-generation European interconnect. Microprocess Microsyst 105102:1. https://doi.org/10.1016/j.micpro.2024.105102
46. Biagioni A (2022) RED-SEA: network solution for exascale architectures. In: 25th Euromicro Conference on Digital System Design, DSD 2022, Maspalomas, Spain, August 31–September 2, 2022, IEEE, pp 712–719. https://doi.org/10.1109/DSD57027.2022.00100
47. Kung NT, Morris R (1995) Credit-based flow control for atm networks. IEEE Netw 9(2):40–48. https://doi.org/10.1109/65.372658
48. BXIv2: high performance interconnect for extreme HPC workloads. https://redsea-project.eu/peek-bxi/. Accessed 26 Aug 2025

Authors and Affiliations

Nikolaos Tampouratzis[1,3] · Ioannis Papaefstathiou[2,3] · Gabriel Gomez-Lopez[4] · Miguel Sánchez De la Rosa[4] · Jesus Escudero-Sahuquillo[4] · Pedro Javier Garcia[4]

✉ Nikolaos Tampouratzis
ntampouratzis@ihu.gr

Ioannis Papaefstathiou
ygp@ece.auth.gr

Gabriel Gomez-Lopez
gabriel.gomez@uclm.es

Miguel Sánchez De la Rosa
miguel.sanchez@uclm.es

Jesus Escudero-Sahuquillo
jesus.escudero@uclm.es

Pedro Javier Garcia
pedroJavier.garcia@uclm.es

[1] Department of Industrial Engineering and Management, International Hellenic University, 57400 Sindos, Greece

[2] School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

[3] Exascale Performance Systems - EXAPSYS Plc, 70013 Heraklion, Greece

[4] Department of Computing Systems, Universidad de Castilla-La Mancha, 13071 Ciudad Real, Spain