Advances in Adapting Memory-Bound CFD Computations to RISC-V Multicore Architecture

Tomasz Olas[1] Lukasz Szustak[1] Roman Wyrzykowski[1]
Mateusz Olas [1] Marco Lapegna[2]

Institute of Computer and Information Sciences, Czestochowa University of Technology
{olas, lszustak, roman}@icis.pcz.pl

Faculty of Mechanical and Industrial Engineering, Warsaw University of Technology, Poland
marco.lapegna@unina.it

ICCS 2025

Agenda

- Motivation and goal of our research
- RISC-V-based platforms
- Overview of MPDATA application
- Parallelization of MPDATA on RISC-V Platforms
- Vectorization using Various RVV Extensions
- Performance Evaluation of MPDATA on RISC-V
- Conclusions

RISC-V architectures

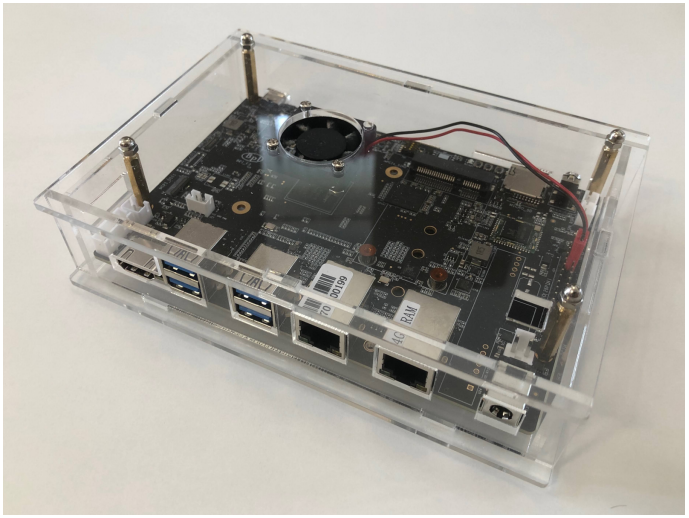
- RISC-V is an open standard Instruction Set Architecture (ISA) that enables the royalty-free development of CPUs and a common software ecosystem
- While RISC-V has already become very popular in some fields, like embedded and edge computing, it has yet to gain traction in general-purpose computing, including HPC, and AI/M
- In particular, recent advances in RISC-V make it a more realistic proposition for HPC workloads than ever before
- At the same time, the performance of publicly available RISC-V CPUs is still behind even mobile x86 and ARM CPUs, but progress in this area is proceeding at a significant pace
- Since the RISC-V software stack includes all the necessary tools for application development, it is of considerable interest to study porting real-life applications to computing platforms based on the RISC-V architecture
- Knowledge gained in this way will allow application programmers to identify bottlenecks in existing approaches to mapping and optimization codes for HPC architectures, considering the characteristics of available computing platforms and the software stack supporting them

Motivation and goal of our research

- In this paper, we tackle the challenge of adapting HPC applications to RISC-V architectures for real-world problems with memory-bound codes, for which memory performance is the main factor affecting computation time
- The application we study as a use case implements the Multidimensional Positive Definite Advection Transport Algorithm (MPDATA)
- We will perform tests on two available commercial platforms:
  - **Banana Pi BPI-F3 Low-Power Platform**
  - **Milk-V Pionier Platform with 64-core Sophon SG2042 Processor**

Banana Pi BPI-F3

- Banana Pi BPI-F3 is a industrial grade RISC-V development board with 4GB 32-bit LPDDR4 SDRAM with 2666Mbps operation (max 16GB)
- It is equipped with RISC-V SpacemiT® X60™ dual-cluster octa-core processors (2 * 4 cores):
  - Each core has 32KB L1-I Cache
  - Each core has 32KB L1-D Cache
  - Each cluster has 512KB L2 Cache
  - Vector Extension: RVV1.0 256-bit
  - Eight-stage dual-issue in-order pipeline

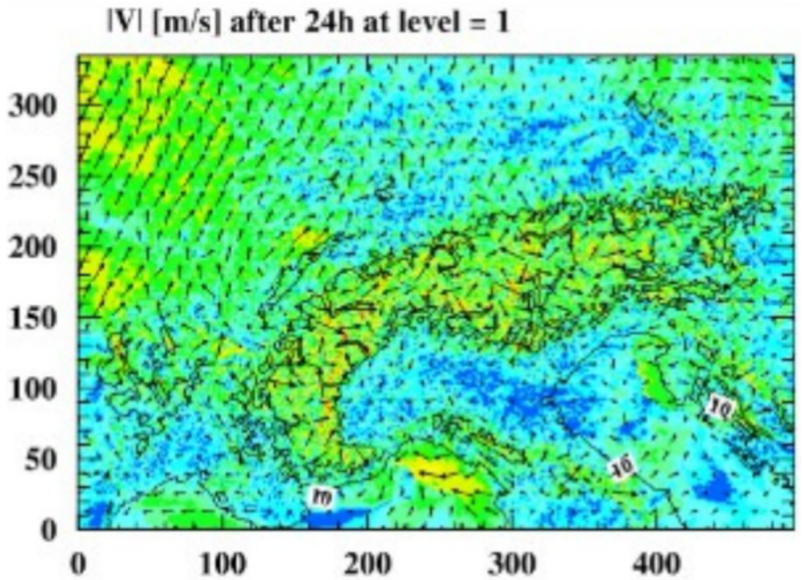


Milk-V Pionier with 64-core Sophon SG2042 Processor

- Milk-V Pioneer is a developer motherboard based on the 64-core Sophon SG2042 RISC-V CPU
- It runs at 2GHz and is organized in 16 clusters of four XuanTie C920 cores
- Clusters are connected through the network-on-chip (NoC) with a 2D mesh topology
  - L1 Instruction Cache - 64KB
  - L1 Data Cache - 64KB
  - L2 Cache - 1MB (shared across a cluster of four core)
  - L3 Cache - 16GB
  - 12 stages out-of-order multiple-issue superscalar pipeline
  - Vector Extension: RVV 0.7.1 128-bit
  - RAM: 128GB of DDR4-3200 (four DDR4-3200 memory controllers)

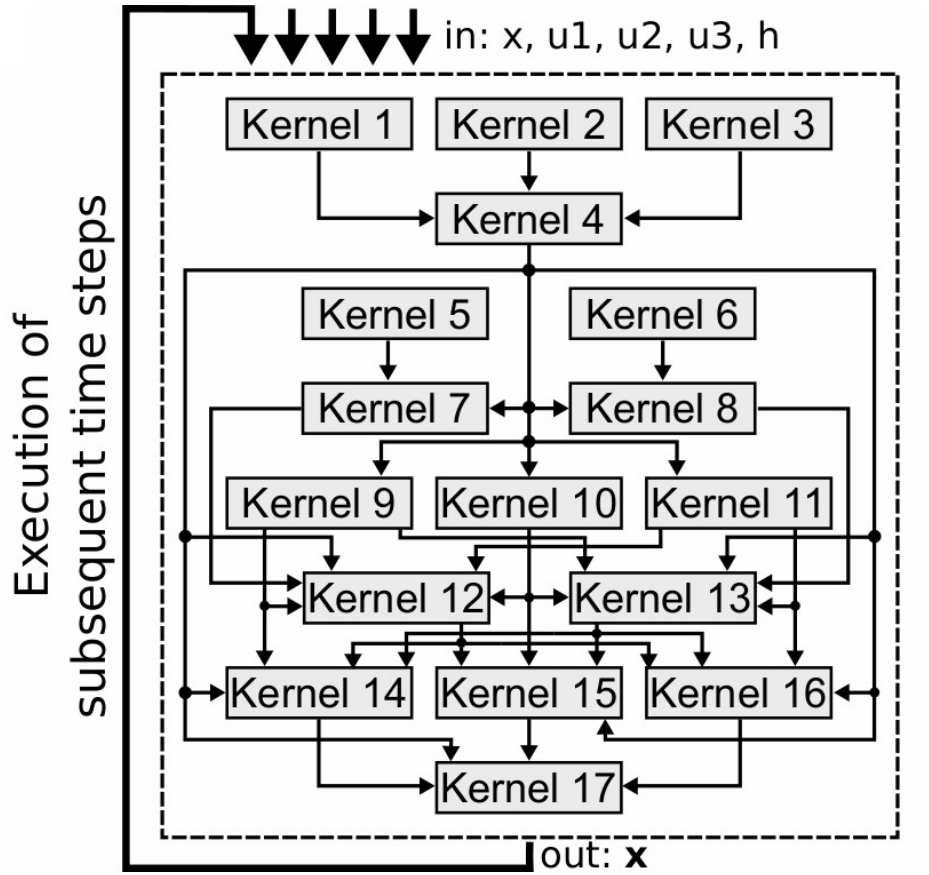
MPDATA

- MPDATA (Multidimensional Positive Definite Advection Transport Algorithm) is one of the main parts of EULAG
- EULAG is an established computational model developed by the group headed by Piotr K. Smolarkiewicz for simulating thermo-fluid flows across a wide range of scales and physical scenarios
- One of the most interesting applications of the EULAG model is numerical weather prediction (NWP)
- MPDATA is a real-life CFD application
- MPDATA belongs to the class of the forward-in-time algorithms which assume iterative execution of multiple time steps
- We focus on simulations using 3D grid



MPDATA: Data dependencies

- The whole MPDATA computations in each time step are decomposed into a set of 17 heterogeneous stencils
- The stages depend on each other
- A single MPDATA time step requires 5 input and 1 output matrices



A naive, non-optimized MPDATA

```
#pragma omp for
  for( i=0; i<n; ++i )
    for( ... )
      #pragma vector
      for( ... )
        f1(i,j,k) = ...

#pragma omp for
  for( i=0; i<n; ++i )
    for( ... )
      #pragma vector
      for( ... )
        f2(i,j,k) = ...
/*....*/
#pragma omp for
  for( i=0; i<n; ++i )
    for( ... )
      #pragma vector
      for( ... )
        x(i,j,k) = ...
```

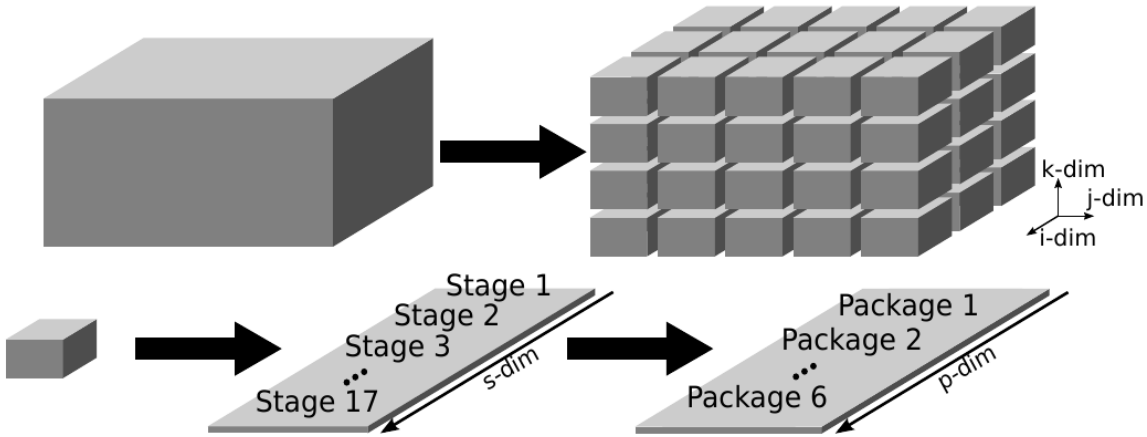
- The basic version of MPDATA is parallelized using **OpenMP**
- All MPDATA kernels are executed sequentially, one by one
- Each kernel processed in parallel
  - by using **#pragma omp for** directive across 1^{st} dimension,
  - and then incorporates automatic vectorization along 3^{rd} dimension using **#pragma vector** directive

Methodology for mapping MPDATA algorithm onto multicore platforms

- However, the operational intensity for naive version of MPDATA is not high enough to utilize computing resources of modern processors efficiently
- **MPDATA is a memory-bound algorithm**
- In order to alleviate such performance constraints we proposed recently parallelization methodology for SMP/ccNUMA architectures
- To ensure the performance portability across various computing platforms, the following parametric optimization techniques were proposed targeting x86 architecture:
  - **(3+1)D decomposition of MPDATA**
  - Islands-of-cores strategy
  - **Data-flow strategy of synchronization**
  - **Vectorization**

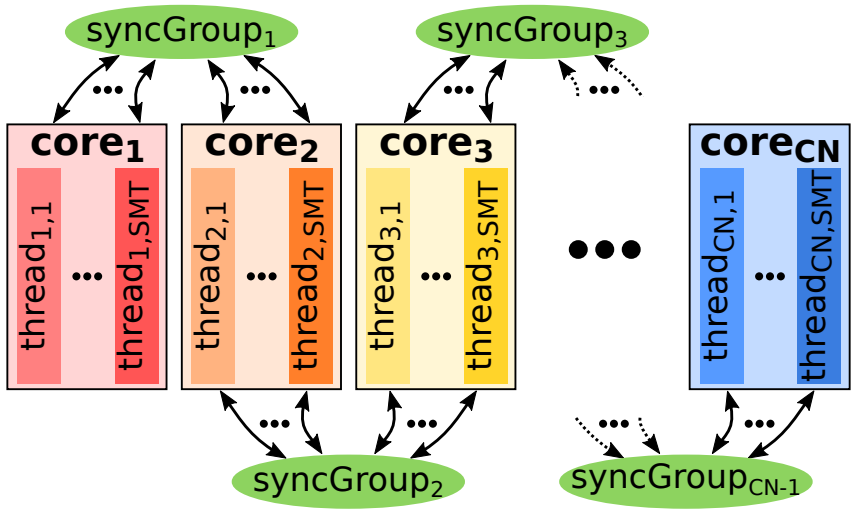
(3+1)D Decomposition of MPDATA

- The prime goal of is to take advantage of cache reusing by transferring the data traffic between kernels from the main memory to the cache hierarchy
- For this aim, a combination of loop tiling and loop fusion optimization techniques is used, that allows reducing the main memory traffic at the cost of additional computations



Data-flow strategy of synchronization

- The main idea is to synchronize only interdependent threads instead of using the barrier approach that – in contrast to the developed approach – synchronize all threads
- This strategy reduces the cost of synchronization
- Implementing this strategy needs needs to reveal the scheme of inter-thread data traffic during execution of MPDATA kernels



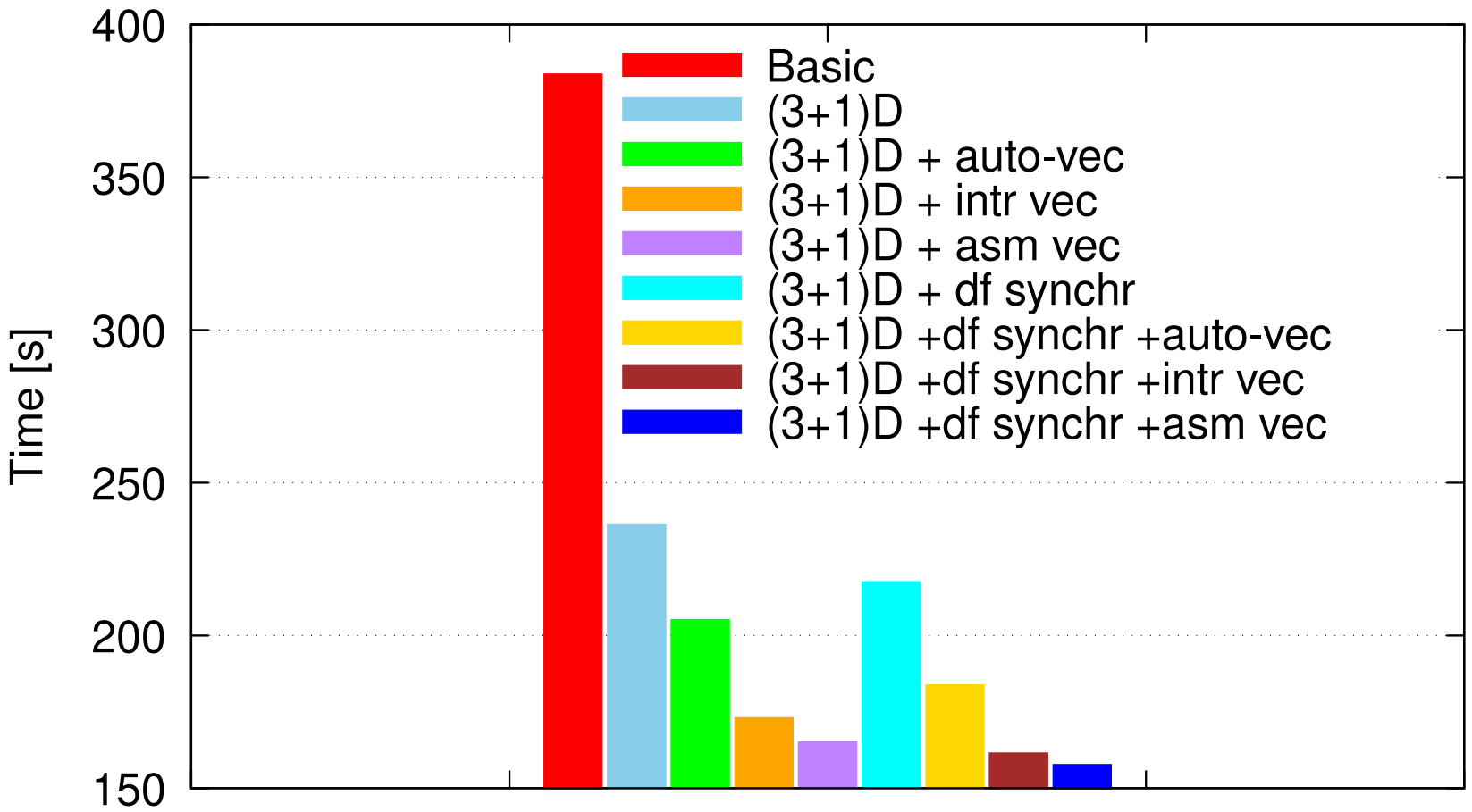
Vectorization using Various RVV Extensions

- Using vector (or SIMD) unit has yielded notable performance improvements for Intel and AMD processors
- In RISC-V processors, the vector (or "V") extension of ISA is dedicated to supporting vectorization. The studied platforms implement different versions of this extension:
  - **SG2042 with RVV 0.7.1 Extension** - Due to difficulties in finding a compiler that supports vector intrinsics on this CPU, we manually vectorize code using assembly. Assembly fragments targeting the xtheadvector extension are compiled with gcc 9.2.0, while the rest of the C++ code and final linking are handled by gcc 13.2.0
  - **SpacemiT K1 with RVV 1.0 Extension** - With modern Clang and gcc compilers supporting RVV 1.0, automatic vectorization by the compiler becomes feasible. MPDATA kernels are adapted similarly to x86, using Clang directives like `#pragma clang loop vectorize(enable)` and intrinsics to guide and enhance vectorization

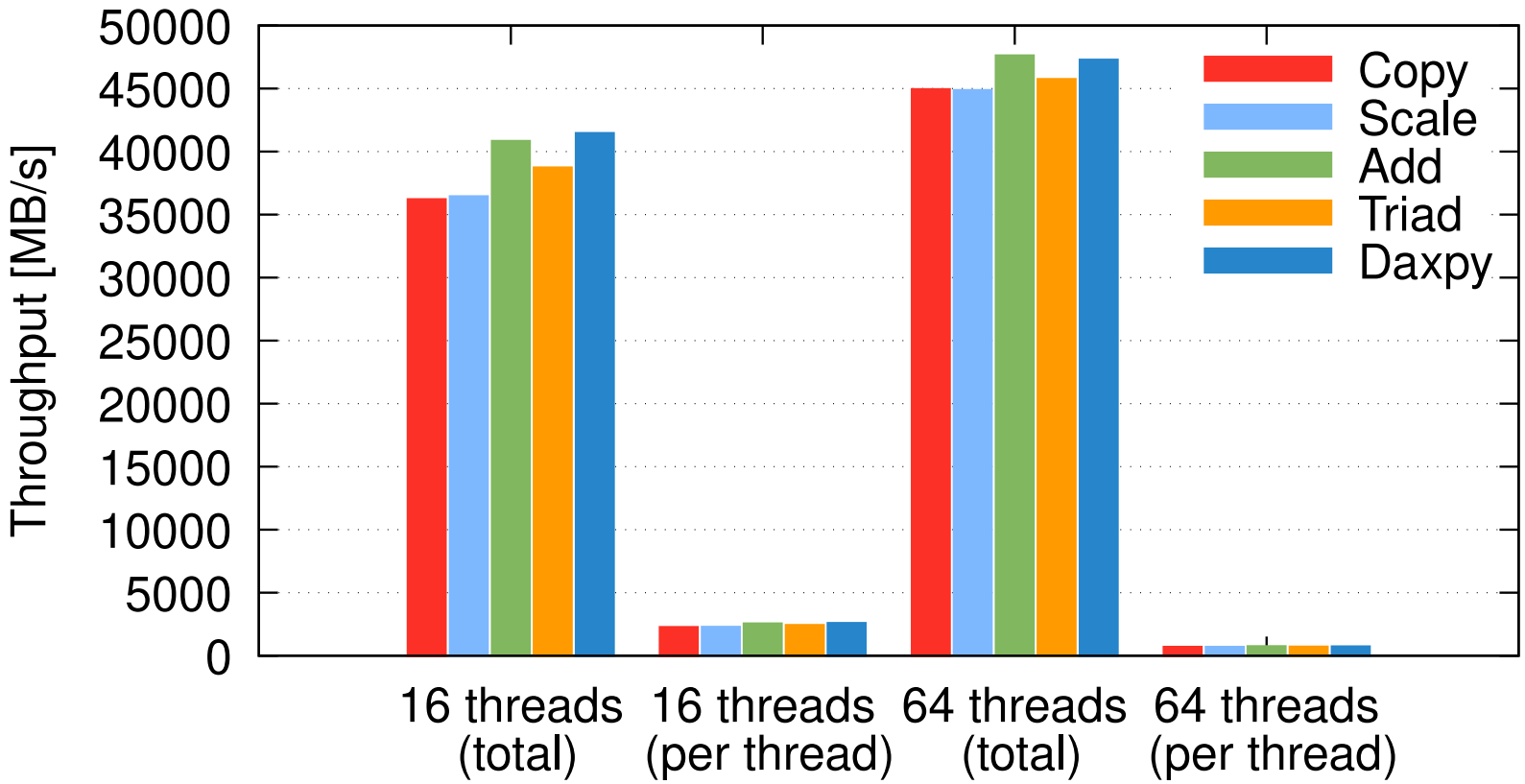
Metodology of experiments

- Evaluating the impact of optimization steps on MPDATA execution time on Banana Pi BPI-F3
- For the SG2042:
  1. Determining the memory throughput using the STREAM benchmark
  2. Determining the overall performance for executing various algorithms using the NPB benchmark
  3. Investigating the performance of different versions of MPDATA
  4. Performance analysis based on the Roofline model
  5. Using single precision for improving performance
  6. Preliminary assessment of energy consumption and energy efficiency for the analyzed platforms

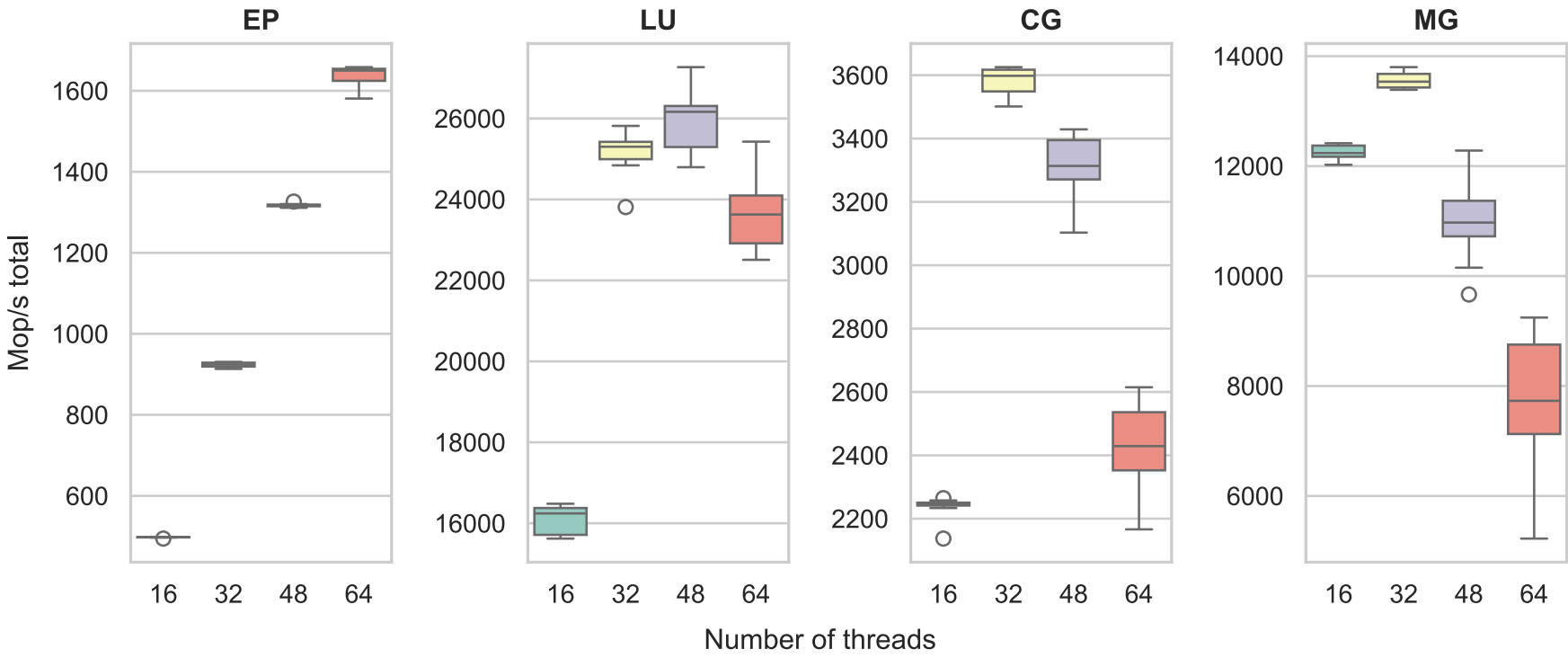
Execution time for running different versions of MPDATA code on Banana Pi BPI-F3



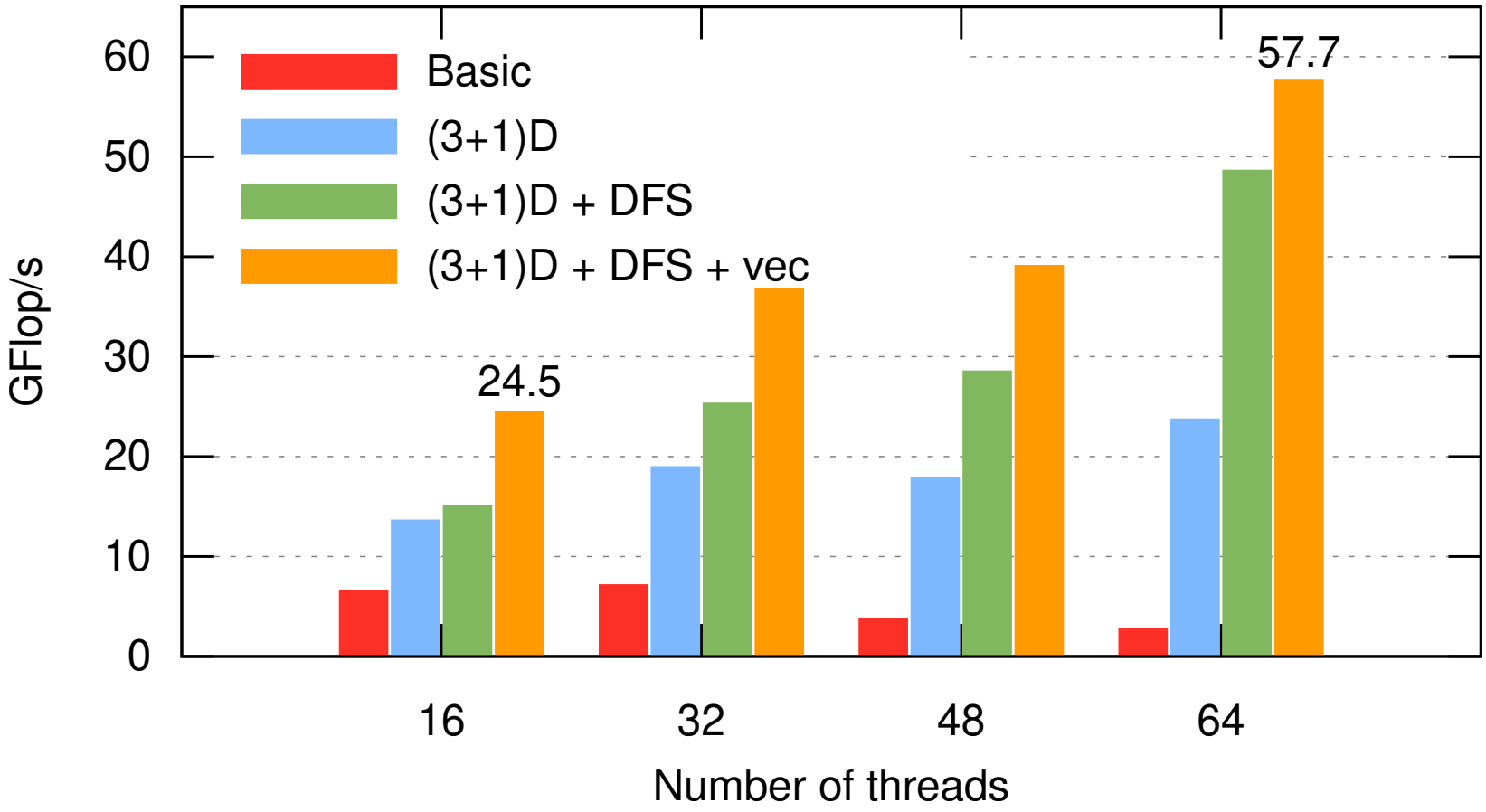
Determining memory throughputs using the STREAM benchmark



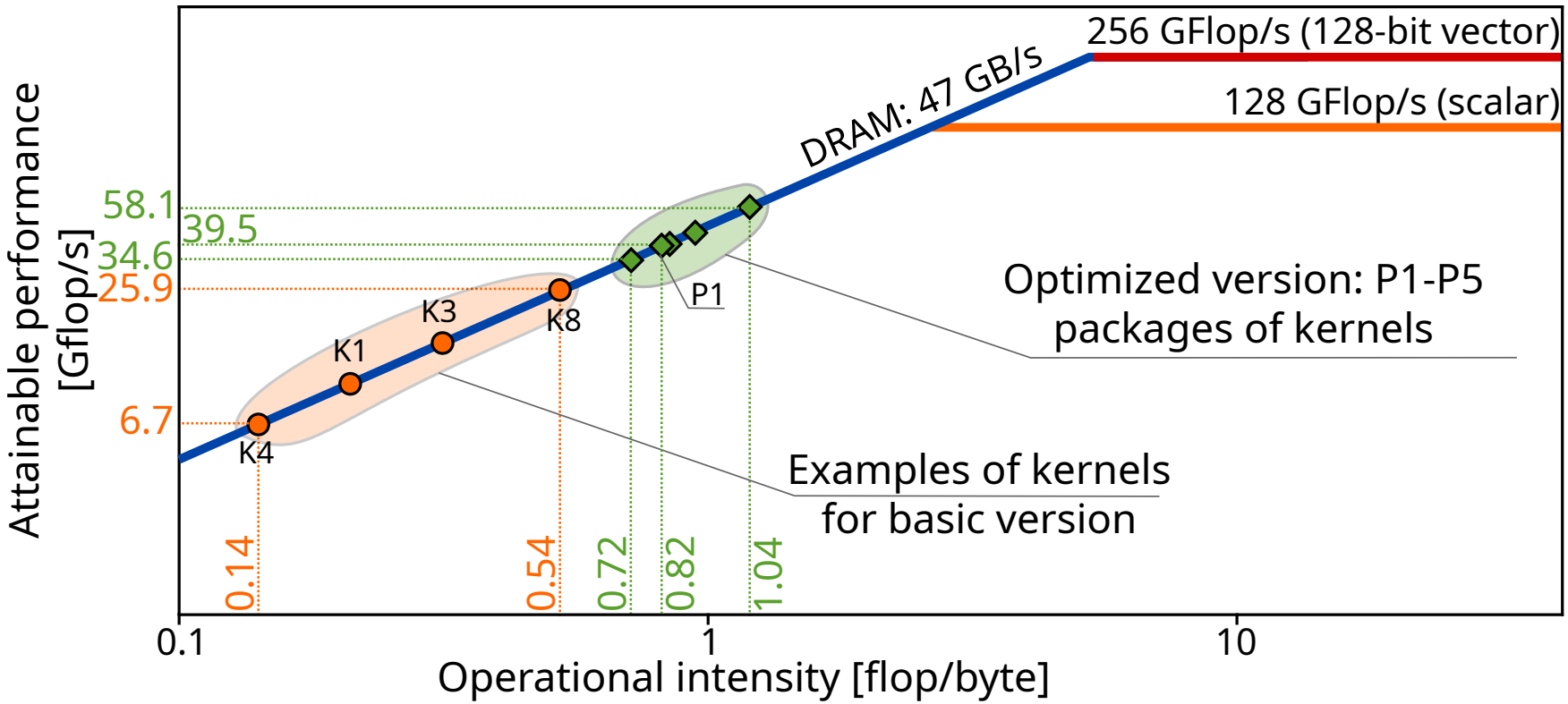
Scalability of selected NPB codes on SG2042



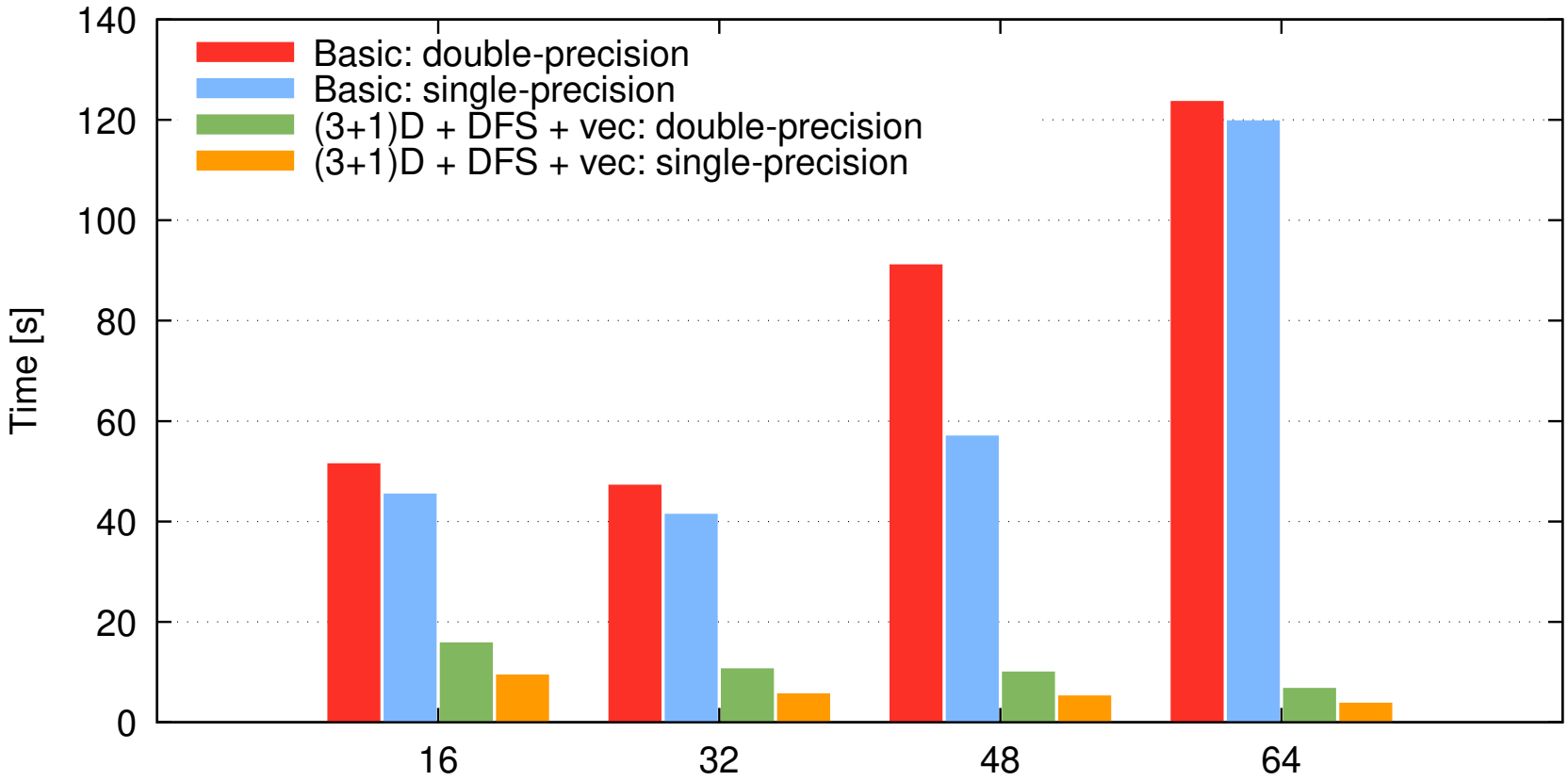
Performance of different versions of MPDATA on SG2042



The Roofline model for the double-precision MPDATA on Milk-V

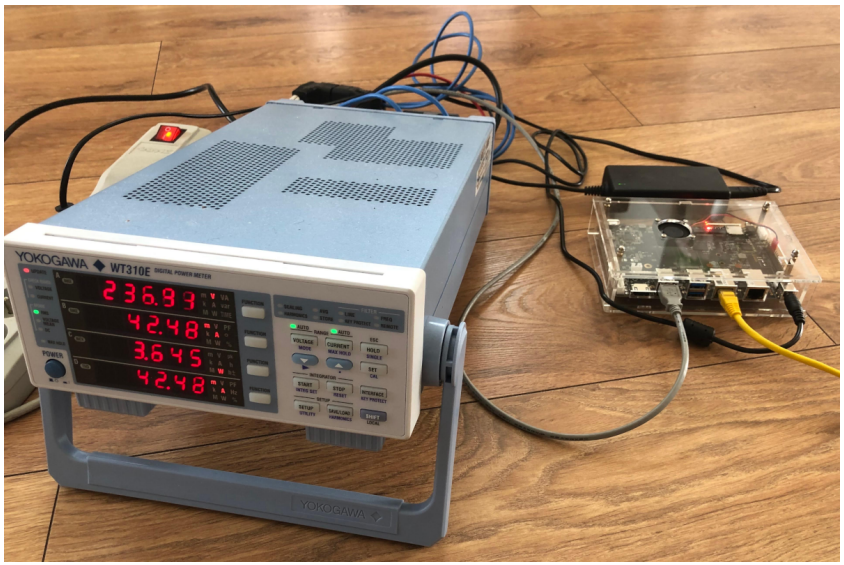


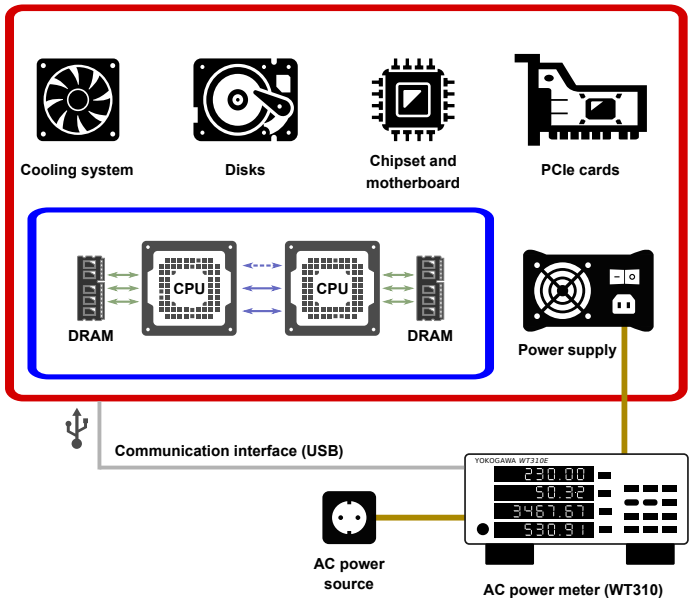
Performance of single-precision codes on SG2042 versus double-precision



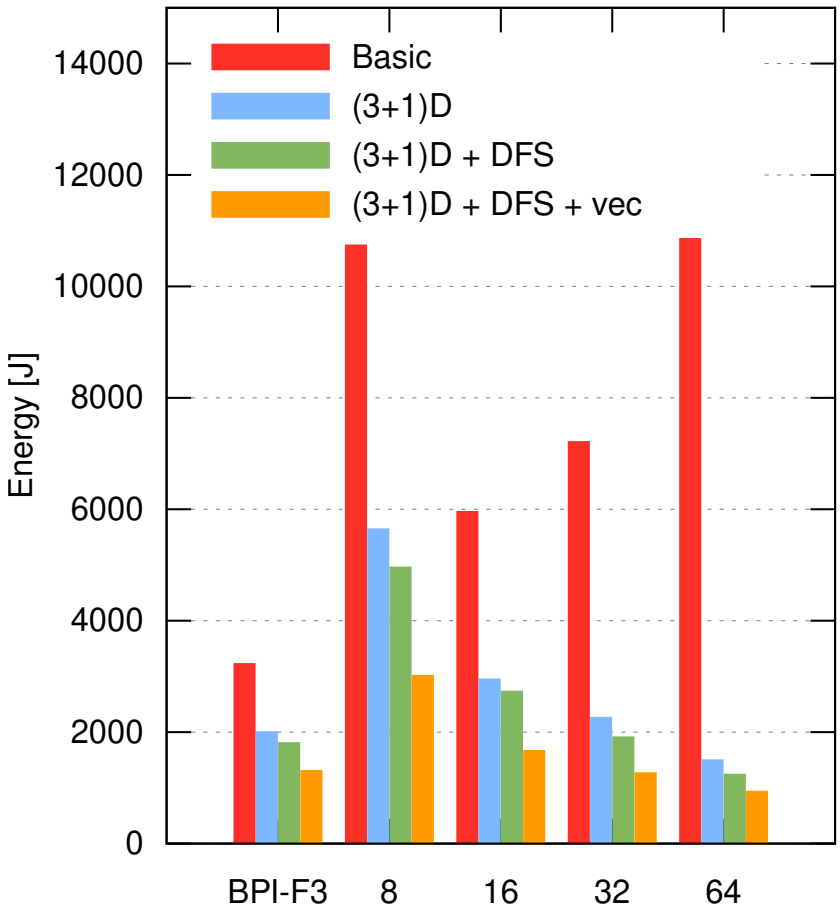
Methodology of power and energy measurements

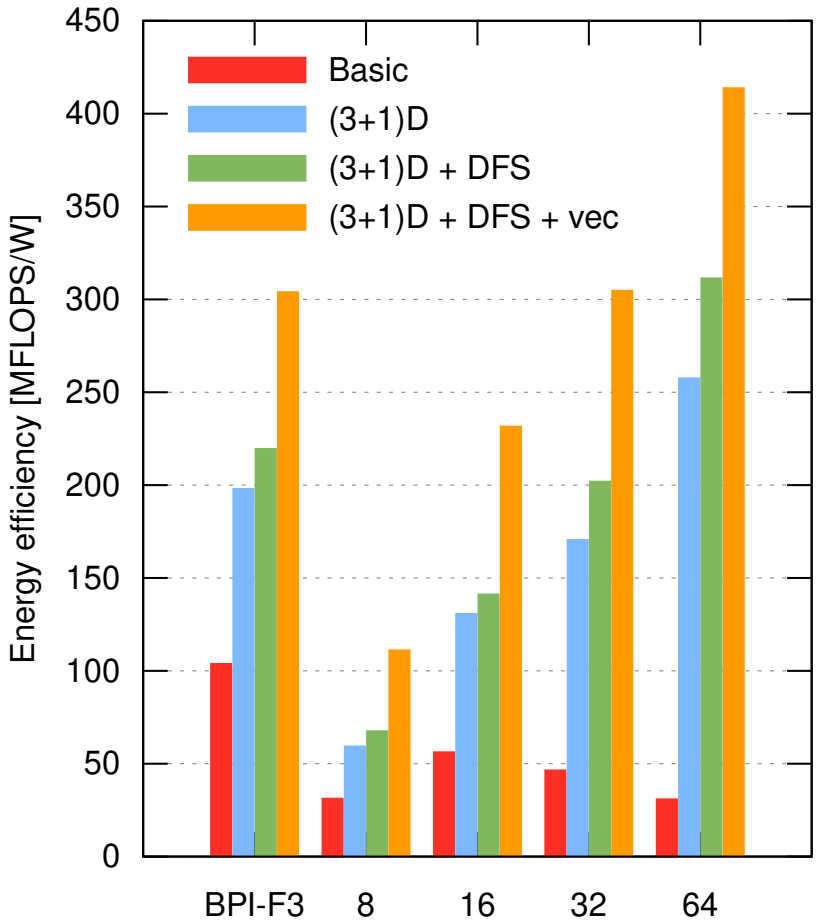
- The Yokogawa WT310 digital power meter is used to measure the power and energy of the platform
- It is installed between the input power sockets of the server and the wall AC outlets and measures power and energy consumed in real time of the whole platform



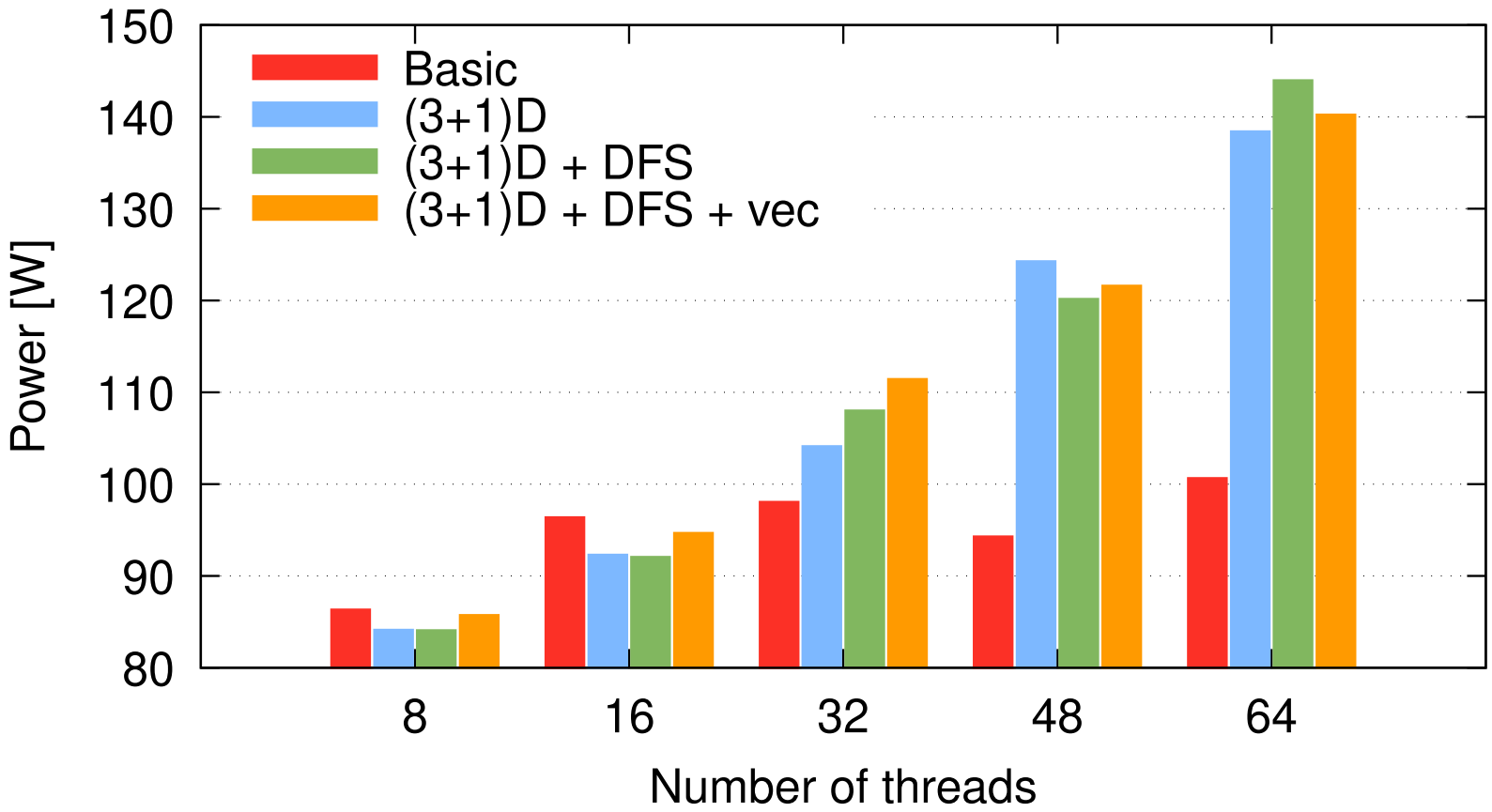


Comparison of energy consumption and energy efficency for MPDATA versions, various computing platforms





Average power consumption in watts for double-precision versions of MPDATA on the Milk-V platform



Summary (I)

- We demonstrate that our optimization methodology developed previously for Intel and AMD x86 architectures can efficiently address performance trade-offs and bottlenecks of two resource-constrained, multicore RISC-V platforms while executing the memory-bound MPDATA code
- To efficiently utilize the vector hardware of the considered CPUs, besides using the auto-vectorization for the SpacemiT K1 CPU, we develop a manual vectorization of MPDATA codes for both versions of the RVV extension and different numerical precisions
- The experimental evaluation of MPDATA codes on the Sophgon SG2042 CPU shows that, unlike most tests from the NPB test suite and the basic MPDATA code, our optimized code is scalable up to all 64 cores of this CPU

Summary (II)

- In double precision, the code optimizations allow us to speed up computation more than 7 times compared to the original code. For single precision, this speedup is even higher, exceeding 10 times
- The experimental evaluation of energy consumption demonstrates convincingly the energy savings achieved by the code optimizations performed for both platforms. In particular, on the Milk-V Pionier platform, energy consumption is reduced radically - by more than 11 times
- The evaluation of energy efficiency shows that while for eight cores, the Banana BPI-F3 low-power platform beats the Milk-V Pionier platform by more than two times, already by using 32 cores, Milk-V catches up with BPI-F3, and by exploiting 64 cores, Milk-V beats the opponent by 36%