

Taking on RISC in HEP for Energy-Efficient Computing

October 19 - 25, 2024

**CHEP
2024**



Outline

- RISC-V architecture
 - introduction and motivations (what, why, where)
 - RISC-V and High-Performance Computing (**HPC**)
 - available hardware solutions (**Pioneer Milk-V** desktop)
- Software availability
 - Linux OS and general tools availability
 - HEP & WLCG software: **ROOT**, **Geant4**, **CVMFS**, **XRootD**
 - LHC experiments software frameworks: **CMS**
- Benchmarks and Power Usage
 - selected benchmarks: **ROOT bench**, **DB12**, **Geant4** (2x)
 - comparison of different benchmarks
 - performance and energy considerations
- Outlook
 - porting software (**CMSSW**)
 - waiting for new hardware (**RISC-V** servers)
 - RISC-V “benchmarking suite”



What is RISC-V

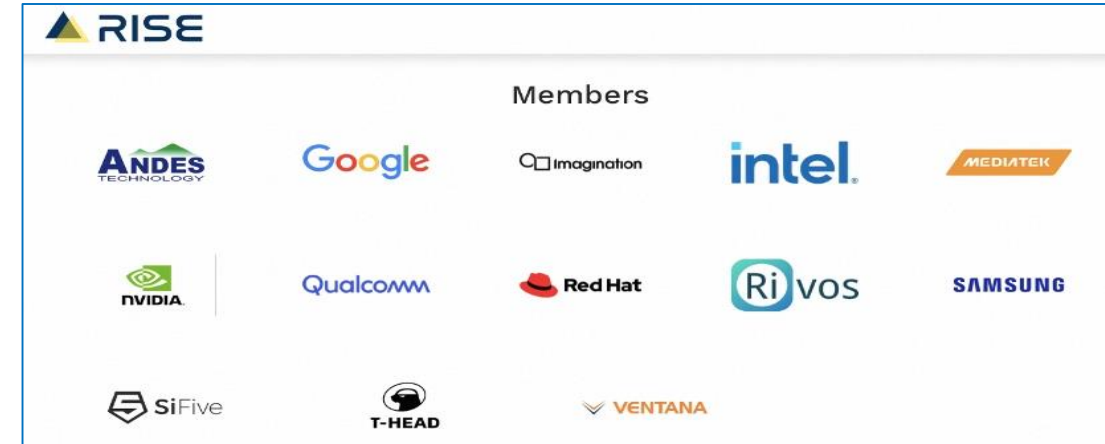


RISC-V is an open standard Instruction Set Architecture (**ISA**) based on established **Reduced Instruction Set Computing** principles, and offered under royalty-free open-source license. Support for RISC-V was added to the Linux 5.17 kernel in 2022, and in 2023 its 64-bit variant called **riscv64**.

- ISA released by UC-Berkeley in 2010
 - Mostly, documents describing the interactions between software and hardware,
 - Managed by the **RISC-V Foundation**: <http://riscv.org/>
- Peculiarity: “*The RISC-V ISA is free and open with a permissive license for use by anyone in all types of implementations. Designers are free to develop proprietary or open source implementations for commercial or other exploitations as they see fit. RISC-V International encourages all implementations that are compliant to the specifications.*”
Very different from **aarch64** or (even more) **x86_64** !
- Like open software, but for hardware: everyone* can design / print / use a RISC-V chip.



* who happens to have an **ASML** micro-lithography machine and a few silicon wafers to spare ...



Moreover, **ISA** is extension friendly: “easy” to design accelerators, specific functions.

See, for instance “**RISC-V acceleration**” talk: <https://indico.cern.ch/e/1329694>

Indeed initial EU design is trying to implement RISC-V as a GPU accelerator.

Why RISC-V

RISC-V is an open-source and royalty-free architecture, easily expandable and with low power usage. And ... there is a fast growing ecosystem around it, holding potential for fast innovation: the **European Processor Initiative (EPI)** will eventually build on RISC-V.

Is **RISC-V** today expected to be a viable platform for WLCG computing? **NO!**

For the time being, **RISC-V** is at the same level as the initial **ARM** test-chips (2013), therefore:

- a similar if not faster ramp up is expected,
- we need a setup to test new boards as soon as they appear (codes, tools, setup).

Moreover:

- As per the name, it is a reduced instruction architecture (like **ARM**): designed for simplicity, performance and low power consumption,
- Initially used (mostly) for teaching chip designs in Electronic Engineering classes,
- Now supported by largest vendors and becoming a real solution from embedded systems, from phones to **HPC**,
- Interest from regions with no in-house technology for chips (e.g., Europe).



 ODROID Wiki

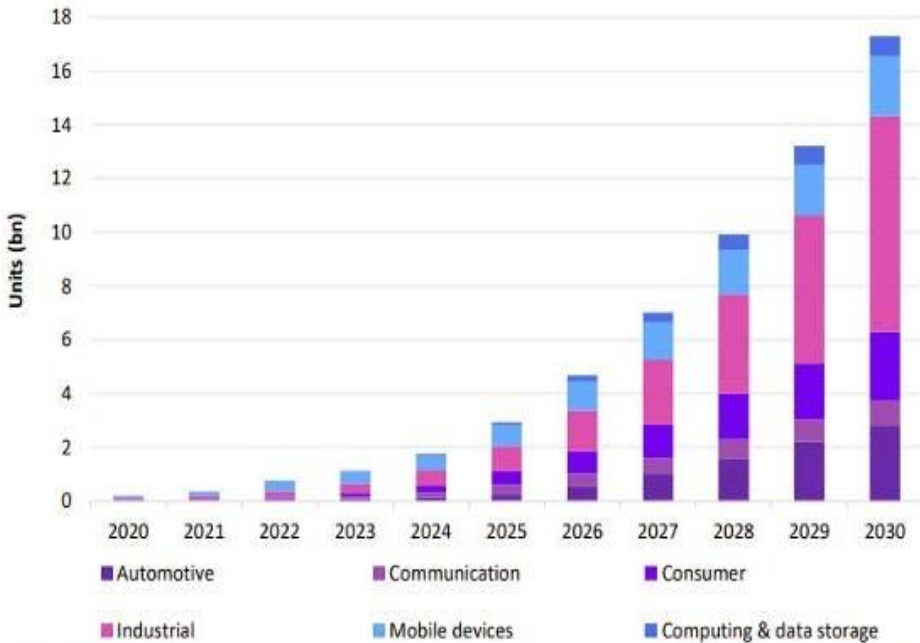
Odroid used in 2013 for the 1st ARM explorations: <https://arxiv.org/pdf/1311.1001>

Where is RISC-V

RISC-V chips are most probably already in your car, your phone, your modem ...

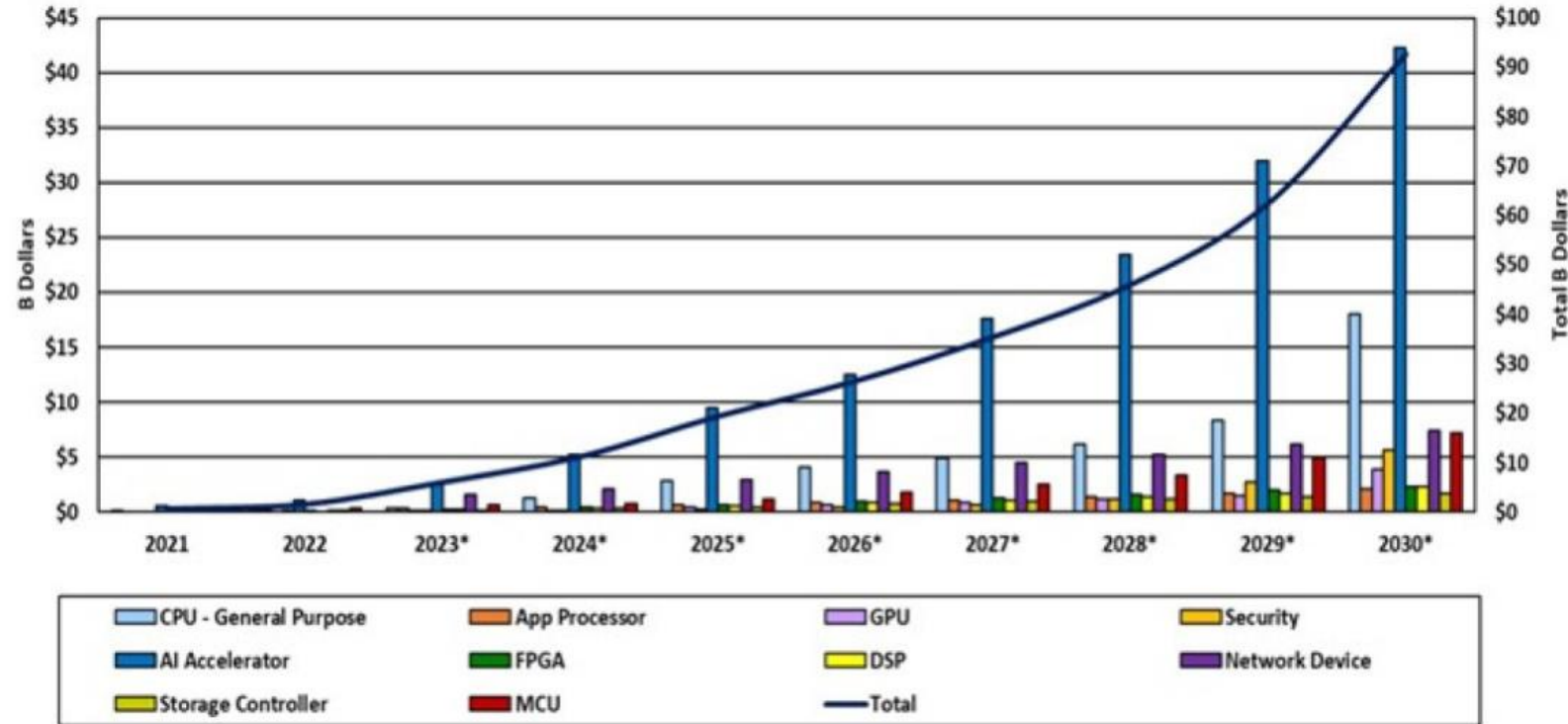
- for specific tasks: we are already at 3 Billion CHiPs per year produced (mostly automotive and industry),
- expected exponential growth for RISC-V as **AI Accelerators** and generic **CPUs** (see projections).

RISC-V processor volume by application



Source: Omdia

© 2024 Omdia



RISC-V in volume:

<https://www.sifive.com/blog/risc-v-for-the-datacenter-introducing-the-p870-d>

RISC-V in \$\$:

<https://www.newswire.com/news/the-shd-group-has-released-a-complimentary-version-of-the-2024-risc-v-22213417>

RISC-V and EU roadmap for HPC

European Processor Initiative (EPI):

<https://www.european-processor-initiative.eu/>

- In the “near future” aims to build an exascale level **European HPC** based on sovereign solutions, including chip design and realization,
- This is not yet around the corner: planning to start with a more standard (**ARM**) CPU and use **RISC-V** as GPU accelerator,
- Eventually, they will develop a full **RISC-V** architecture CPU.



Mission



European independence in High Performance Computing Processor Technologies

EU Exascale machine based on EU processor

Based on solid, long-term economic model

Go beyond HPC market

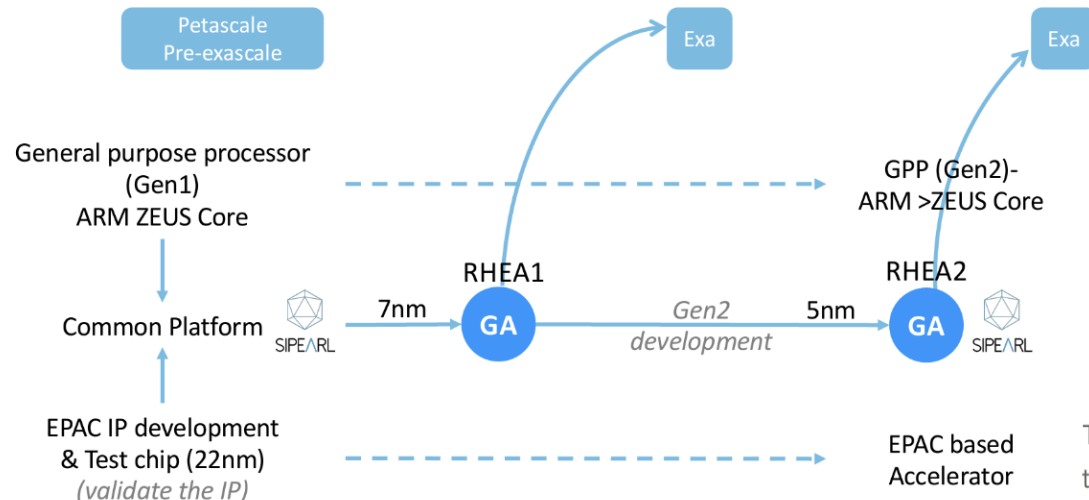
Address the needs of European industry

End-to-end security

Roadmap:

arm
Short term
objective
TRL >=6

mid term
objective
RISC-V
TRL <4



The Accelerator stream will develop and demonstrate fully European processor IPs based on the RISC-V Instruction Set Architecture, providing power efficient and high throughput accelerator tiles within the GPP chip. Using RISC-V allows leveraging open source resources at hardware architecture level and software level, as well as ensuring independence from non-European patented computing technologies.

But in practice, what do we have today?

- Up to 6 months ago, “Raspberry PI” like devices: **Visionfive 2**
 - 4 GB RAM, 4 cores, Fedora 37, Ubuntu 20.04,
 - Useful for initial tests, bootstrap of some HEP codes (Geant4, ROOT, some benchmarks),
 - 12 hours to (fail to) compile gcc.
- More recently, fully integrated desktop PC: **Milk-V Pioneer***
 - 128 GB RAM, 64 cores, Fedora 38, Ubuntu 24.04, Debian,
 - Out-of-the box compatibility with HEP codes (Geant4, ROOT, more benchmarks),
 - Node Performances almost comparable to a standard x86 mid range desktop.
- Upcoming: **MIPS P8700**, up to 512 cores !
 - detailed specs yet to be defined.




Visionfive 2
~100 € on Amazon



Milk-V Pioneer
~2.5k € from China

Cores ramp per chip: 4 → 64 → 512 in just 1 year!

* Now available at CERN, Glasgow, Pisa, Bologna, and CNAF (as far as we know).



Powered by SOPHON SG2042

64 Core RISC-V CPU	
Main Frequency	2GHz
L1 Cache	I:64KB and D:64KB
L2 Cache	1MB/Cluster
L3 Cache	64MB System Cache
Typical power consumption	120W
DDR	4 channel 3200Hz ECC RDIMM/UDIMM/SODIMM
PCI-E	2 x 16x Gen4 with CCIX support

Testing HEP software

Progress in porting **HEP** software to **RISC-V**:

- We managed to compile and install **HEP** software and **Grid** middleware by building from source:

ROOT: <https://github.com/root-project/root> (master Git)

CVMFS: <https://github.com/cvmfs/cvmfs.git> (master Git)

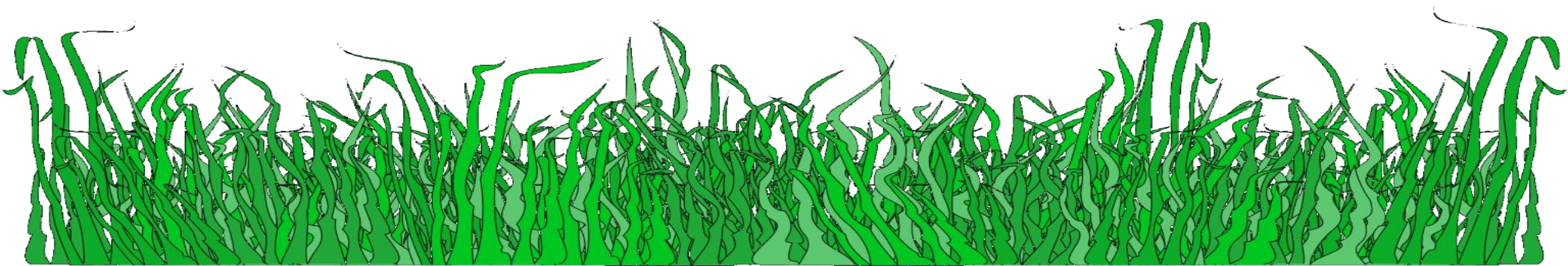
XRootD: <https://github.com/xrootd/xrootd> (master Git)

Geant4: <https://gitlab.cern.ch/geant4/geant4> (master Git)

- **CMS** made some progress in porting the **CMSSW** framework to RISC-V:

most code can be ported, major issues are: missing CUDA & TensorFlow (see next slides ...)

Obviously, kudos to the developers which either made clean portable code or already ported their code to the new architecture! Without them, we won't be talking about this today.



Comparable Machines

These are the machines we used for comparing performances: 2 servers, 1 desktop, and the RISC-V

AMD48c/AMD96ht: Single AMD EPYC 7003 48-Core Processor (GIGABYTE)

CPU: x86 AMD EPYC 7643, 48C/96HT @ 2.3GHz (TDP 225W)

RAM: 256GB (16 x 16GB) DDR4 3200MHz → 2.7 GB/core

HDD: 3.84TB SSD SATA



x86

Intel4ht: Single Socket Intel i5 4-Core CPU (DELL)

CPU: x86 Intel Core i5-7500 CPU, 4C @ 3.4GHz@ (TDP 65W)

RAM: 24GB (1 x 16GB + 1 x 8GB) DDR4 2400 MHz → 6 GB/core

HDD: 500GB SSD SATA



x86

ARM80c: Single socket Ampere Altra Q80-30 80-Core Processor (GIGABYTE)

CPU: ARM Ampere Q80-30, 80C @ 3GHz (TDP 210W)

RAM: 256GB (16 x 16GB) DDR4 3200MHz → 3.2 GB/core

HDD: 3.84TB SSD SATA



ARM

RISCV64c: Single socket RISC-V 64-Core Processor (Milk-V Pioneer)

CPU: SOPHON SG2042 (64 Core C920, RVV 0.71) riscv64 @ 2GHz (TDP 120W)

RAM: 128GB (4 x 32GB) DDR4 3600 MT/s → 2 GB/core

HDD: 1TB PCIe 3.0 NVMe



RISC-V

Selected Benchmarks

We ran **ROOTbench** (a few script to test ROOT functionalities), two **Geant4** simulations (**AnaEx01**: a basic event generation example; and **ParFullCMS**: a realistic CMS simulation with full geometry), and the **DB12** benchmark developed by **LHCb** (single-core and whole-node).

ROOT bench: <https://github.com/root-project/rootbench>

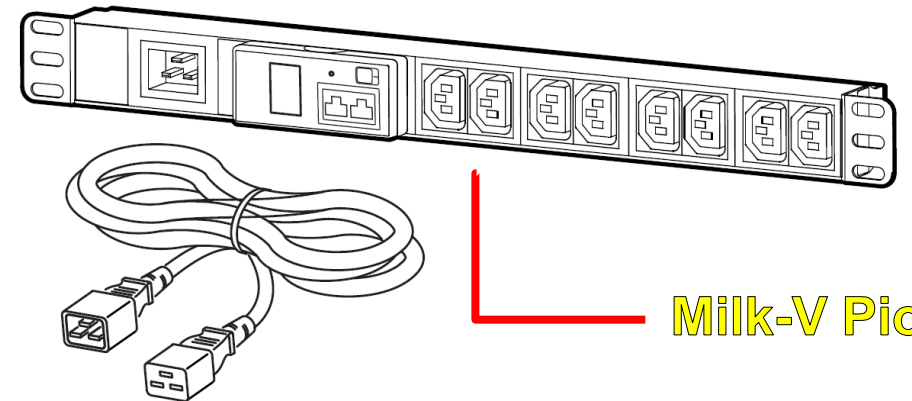
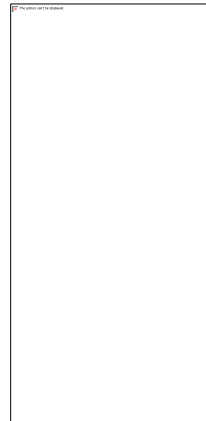
Geant4: <https://gitlab.cern.ch/geant4/geant4>

ParFulCMS: <https://github.com/cms-externals/parfullcms>

DB12: <https://github.com/DIRACGrid/DB12>

Indeed, **HEP-Score** cannot run on RISC-V, as no experimental workload has been ported yet.

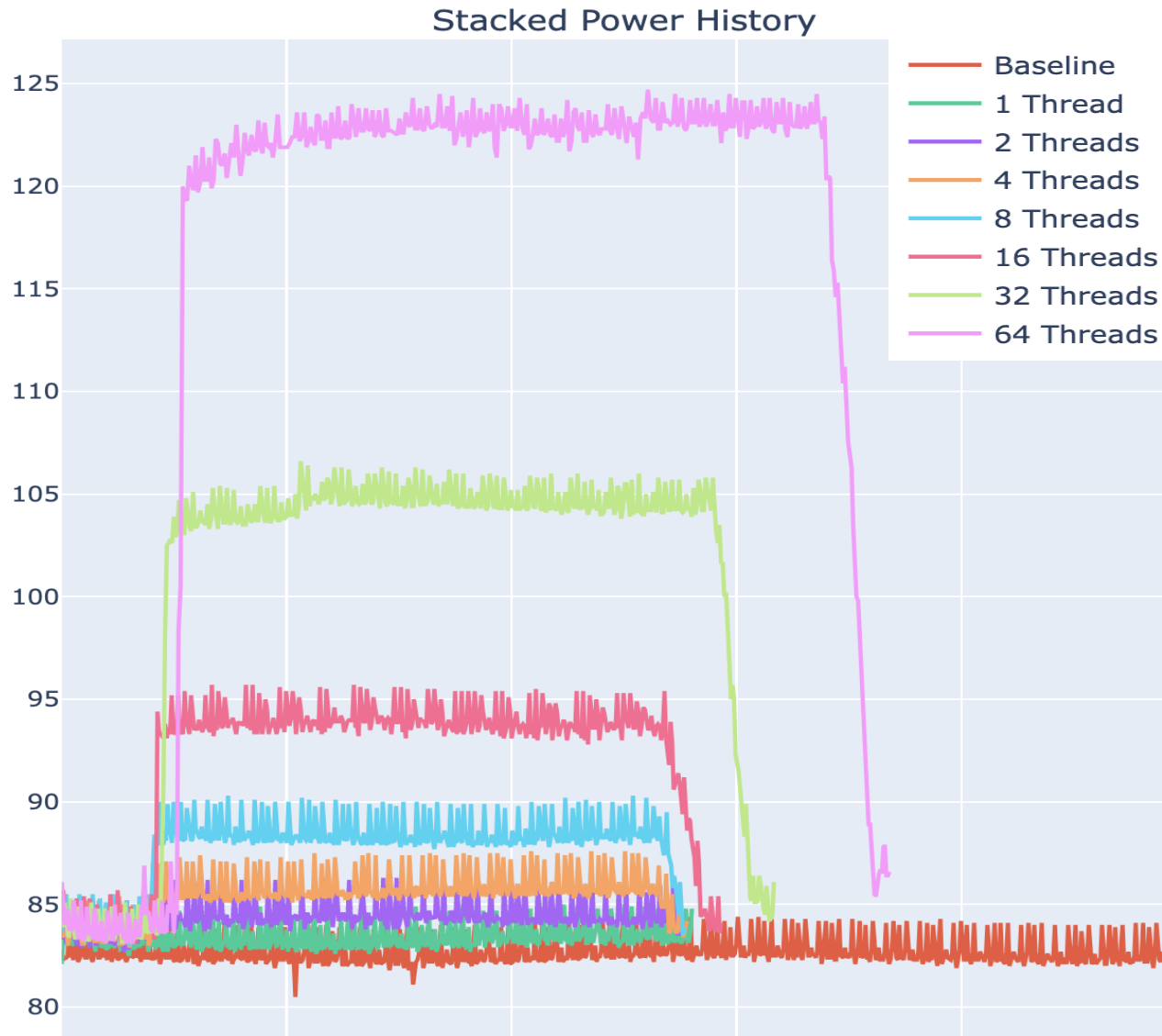
The power usage is measured via **IPMI tools** on the servers, with a **metered plug** on the x86 desktop, and via a **metered PDU** on the Milk-V Pioneer.



So, there might be some uncertainty on the reported values.

Power Usage

To get a better feeling for the power usage of the **Milk-V Pioneer**, we have measured the idle baseline while running **Geant4** jobs with increasing number of threads.



We notice that the **Milk-V Pioneer** has a large power usage in idle mode, comparable to the idle power of a single socket servers, but definitely too high if compared to the mid-range **x86** desktop!

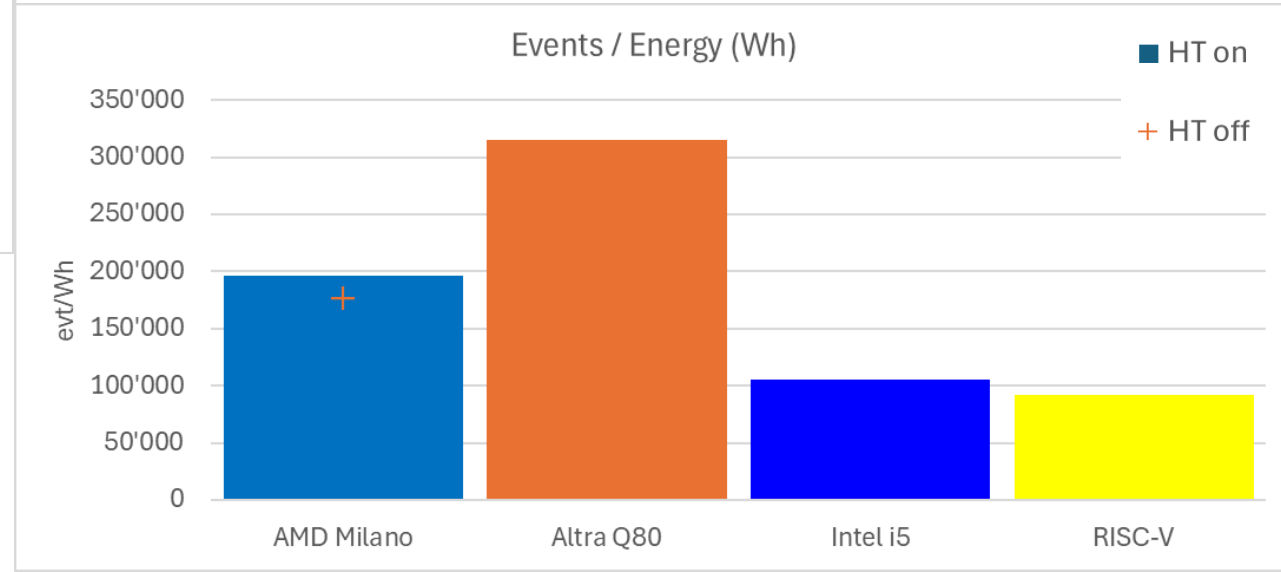
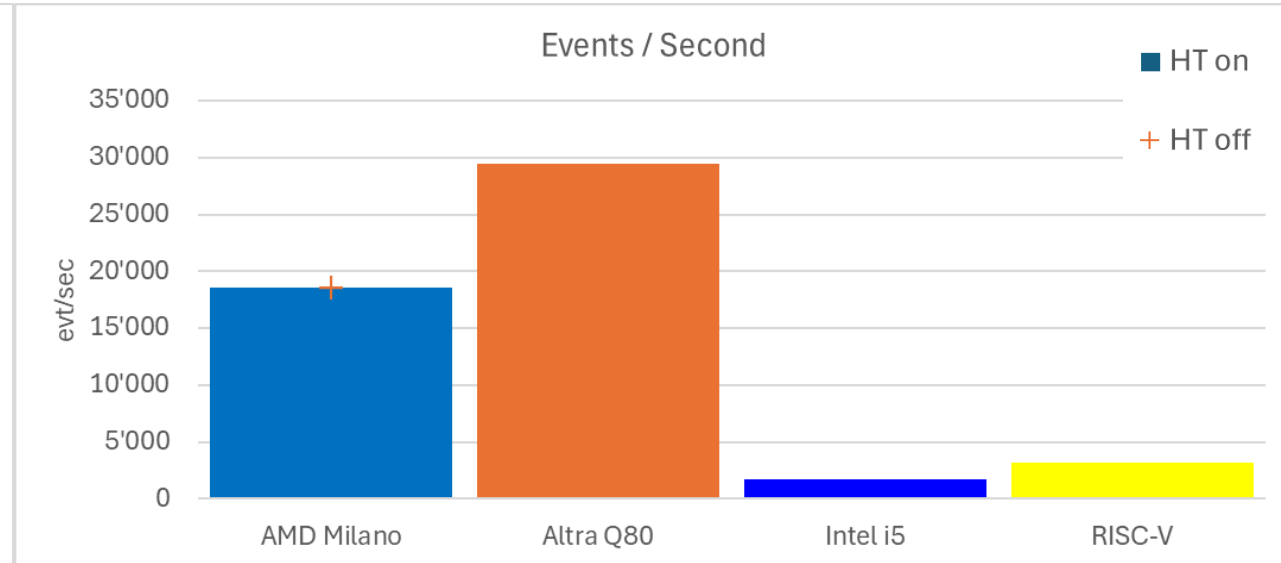
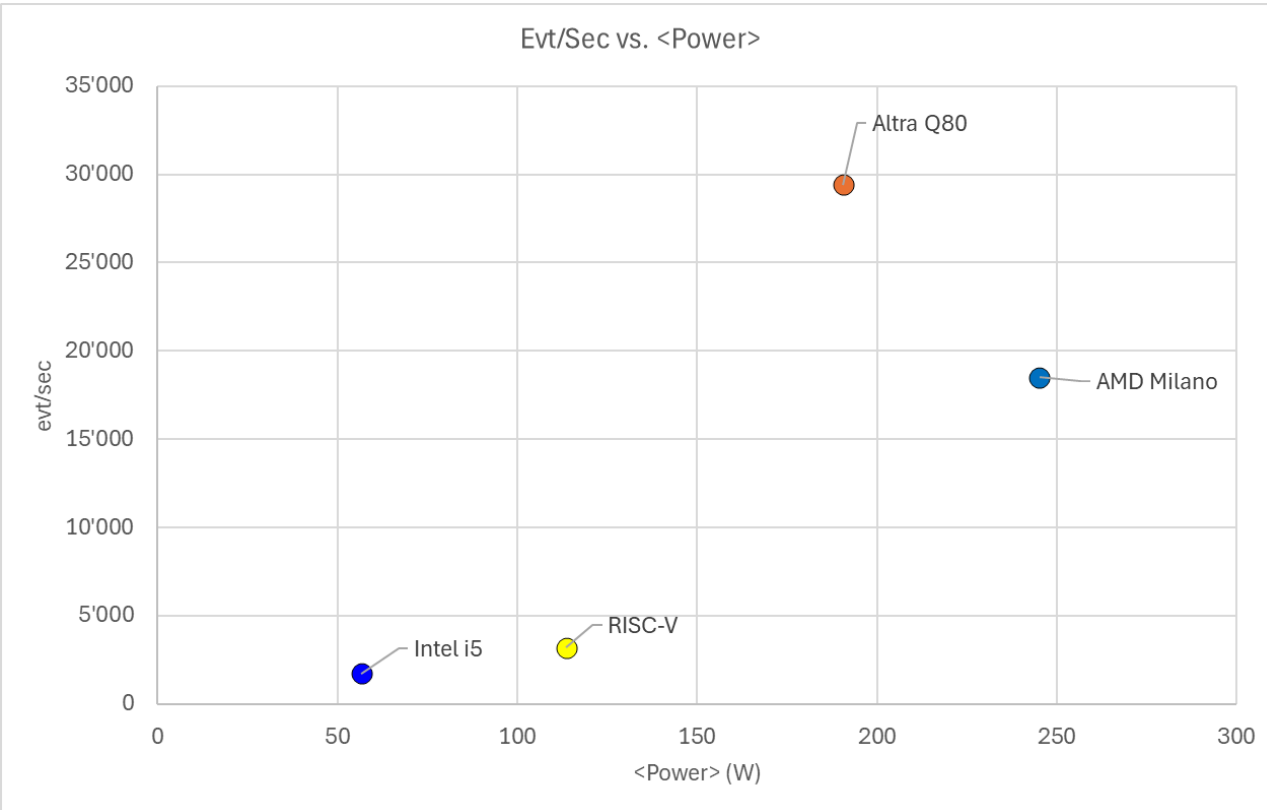
idle power

<i>AMDMilan</i>	~	110W
<i>Altra Q80</i>	~	100W
<i>Intel i5</i>	~	13W
<i>RISC-V</i>	~	80W



Geant4 (AnaEx01)

We ran a standard **Geant4** simulation example (**AnaEx01**) on the 4 types of hardware listed above for 1M events. Power is calculated as a simple average, as the load shape is almost flat (see previous slide).

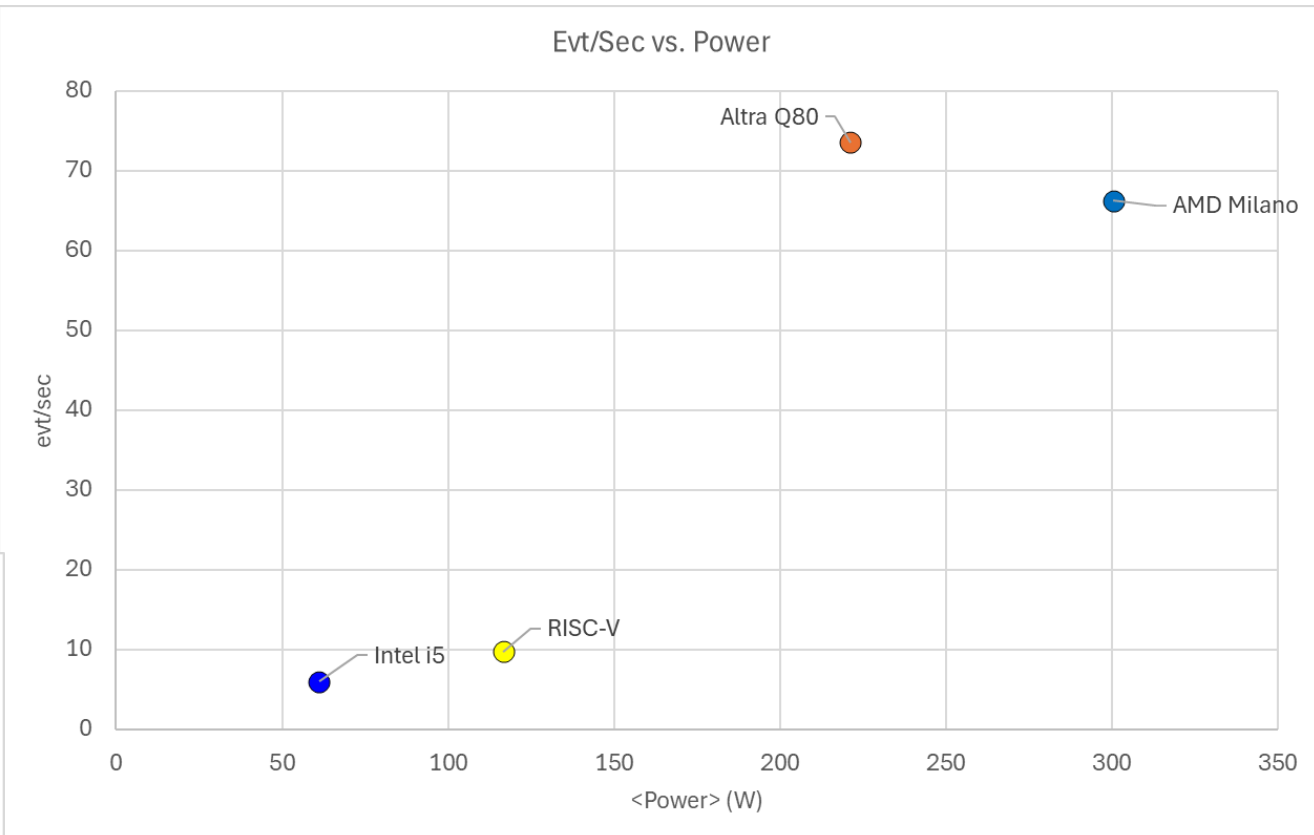
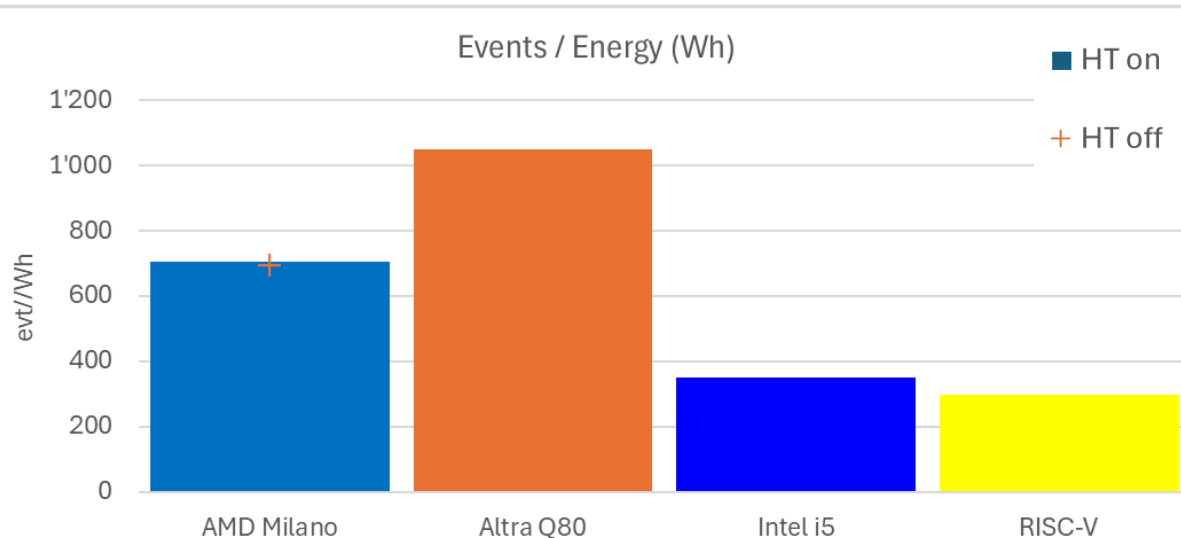
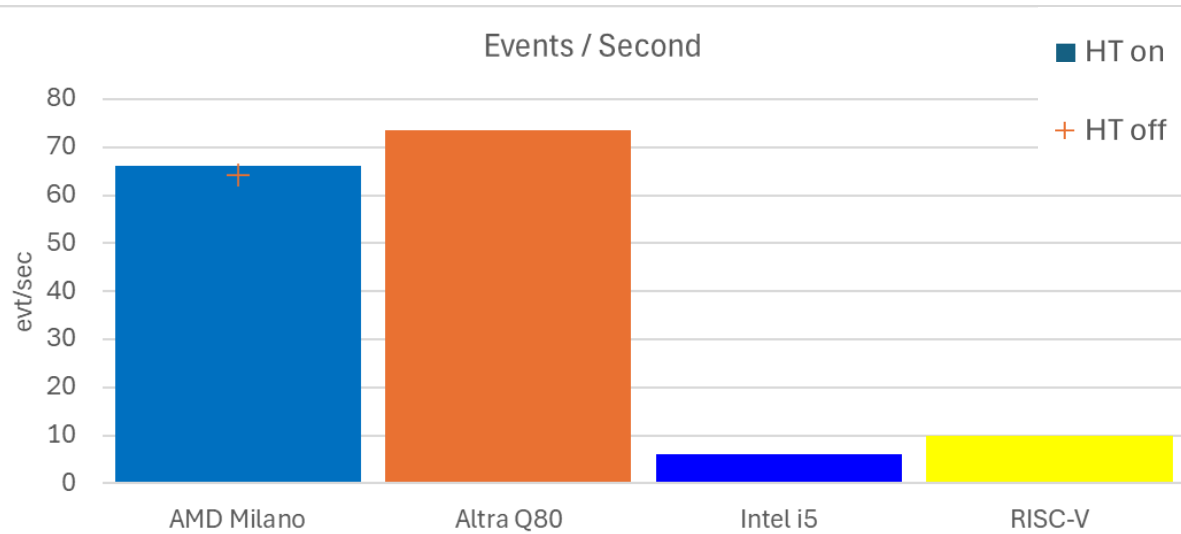


Note that:

$$(\text{Evt/Sec})/\text{Watt} = \text{Evt/Joule} \approx \text{Evt/Wh}$$

Geant4 (ParFullCMS)

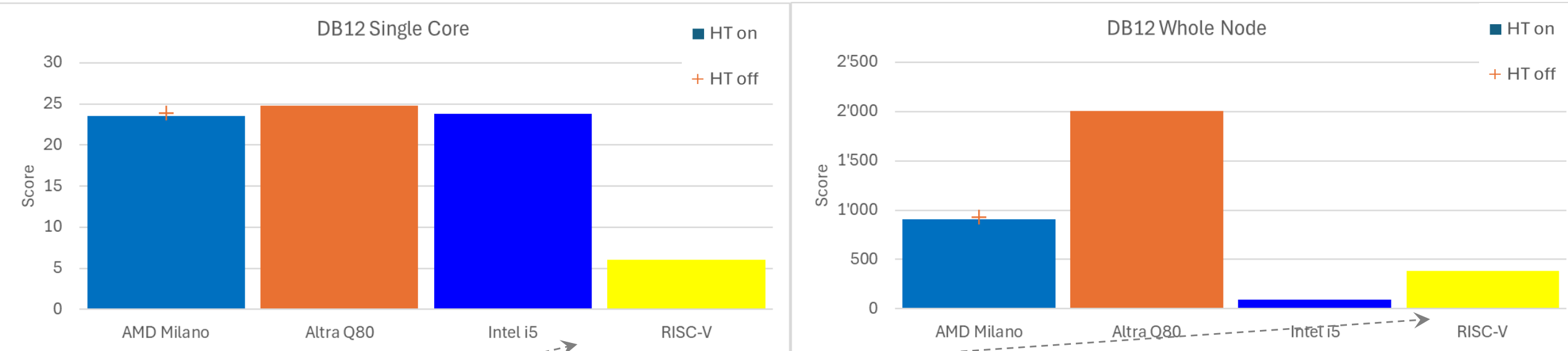
And we ran a **Geant4** simulation using the full **CMS** geometry (**ParFullCMS**) on these same hardware for 10k events (less than the previous example because the full CMS detector simulation takes a lot longer).



Results are pretty similar to the previous Geant4 simulation: the 64-core **RISC-V** CPU almost matches the mid-range 4-core **x86** desktop from 2017.

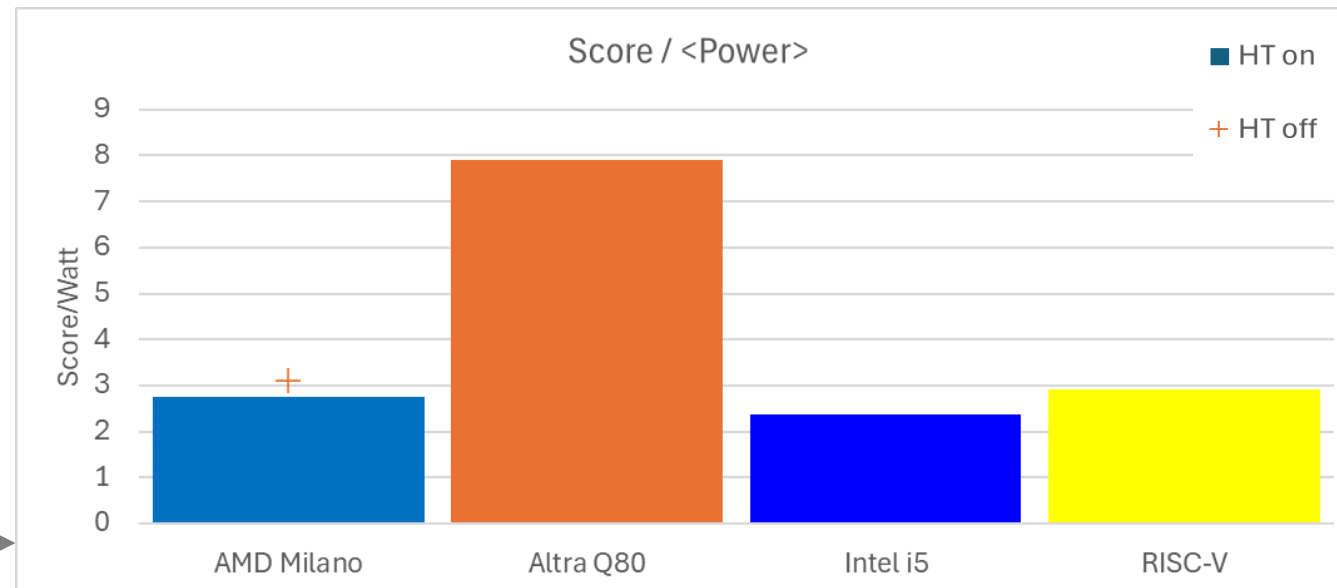
DB12

Then we ran the **DB12** benchmark, in both single-core and whole-node mode:



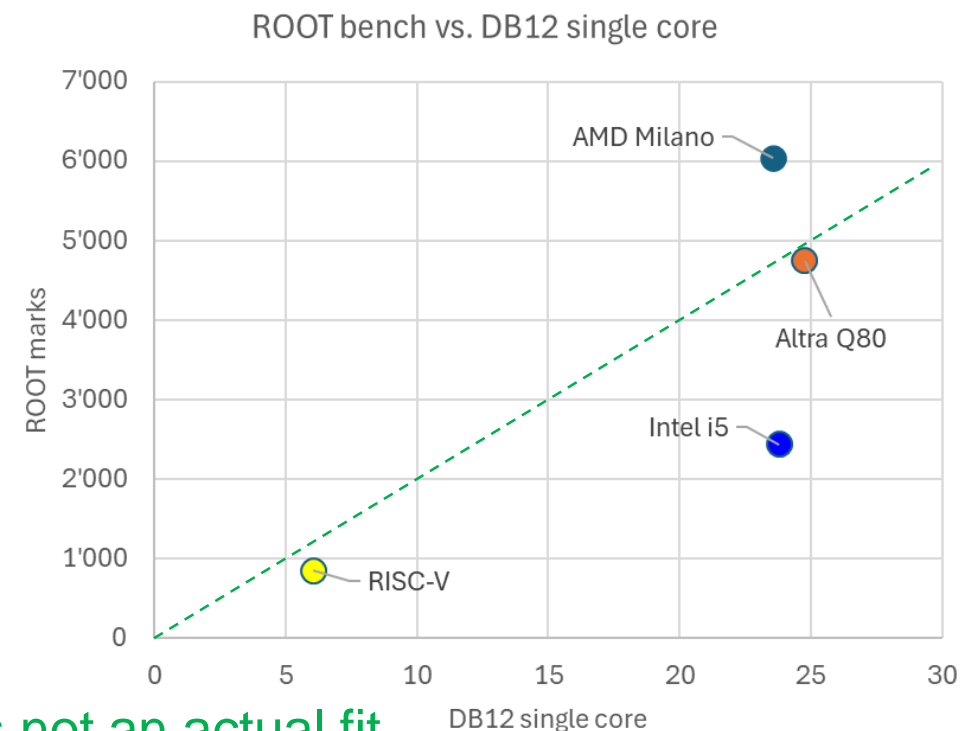
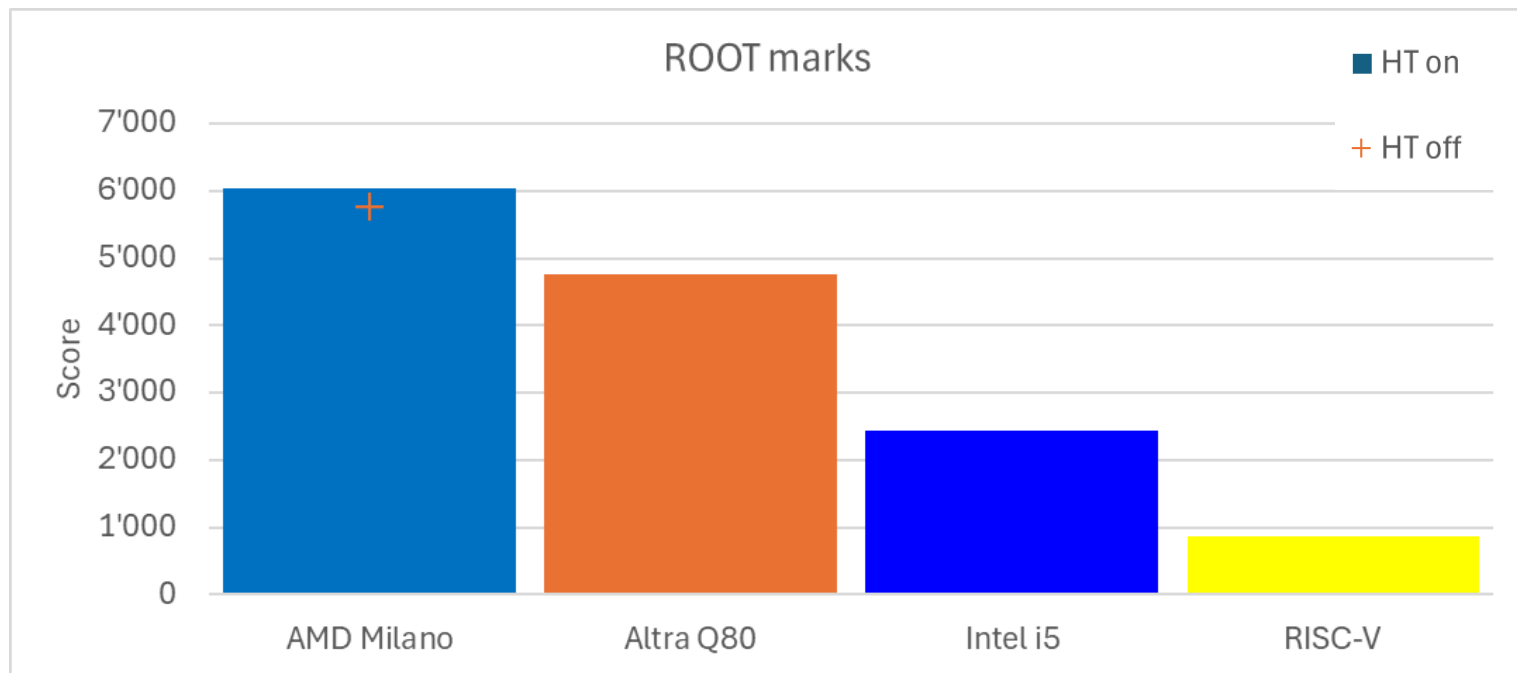
A single core **RISC-V** core is pretty bad, but the whole CPU surpasses the mid-range **x86** desktop ... because it has 16x the number of cores!

In terms of performance per Watt, in this specific test the 64-core **RISC-V** CPU also surpasses the mid-range 4-core **x86** desktop (from 2017).



ROOT benchmark

We have also tried the standard **ROOT bench**: a single threaded benchmarks which, indeed, shows poor performances on RISC-V ...



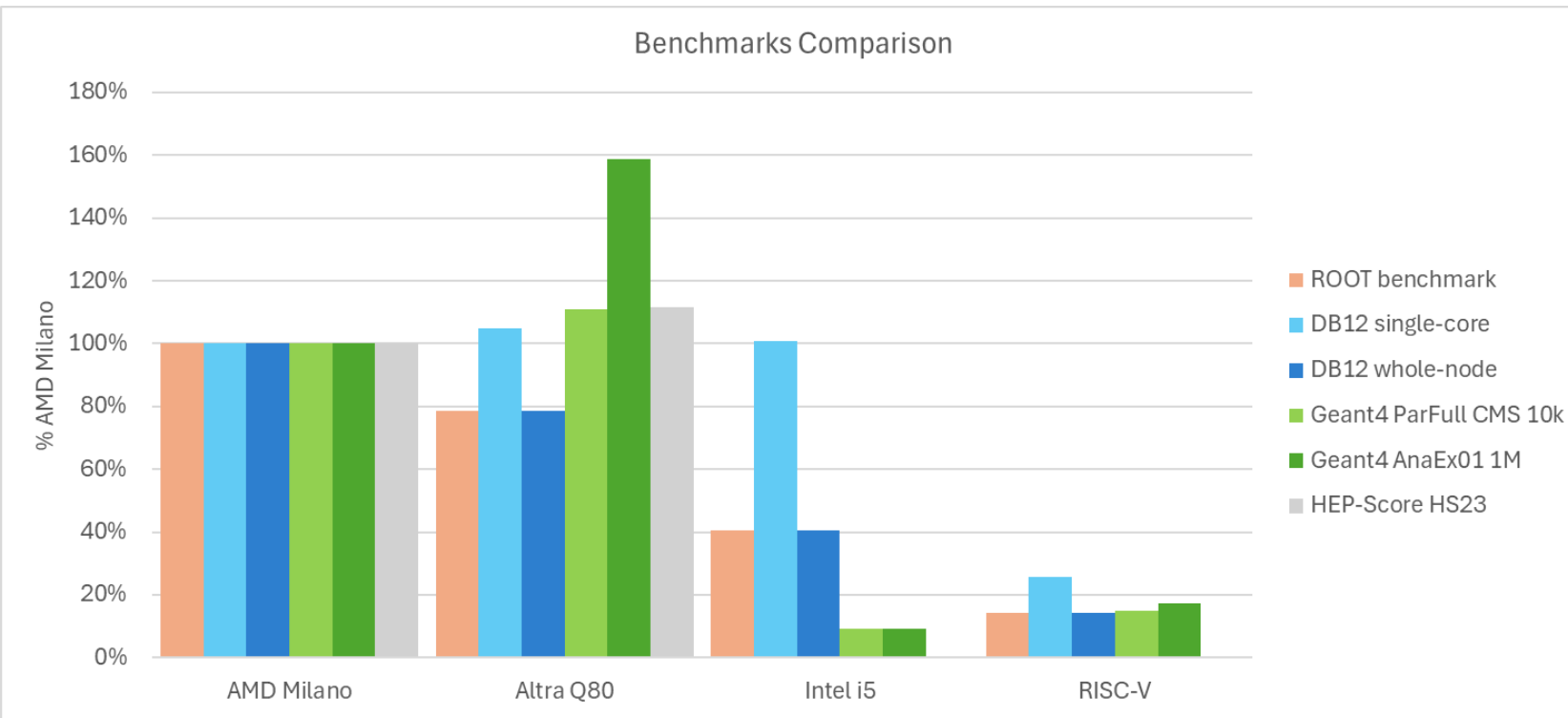
Note: the green line is not an actual fit.

Despite the known limitation of such approach, we can try to compare the **DB12** single-core benchmark with the **ROOT bench**. If you really squint your eyes, you may see some correlation ...



Benchmarks Comparison

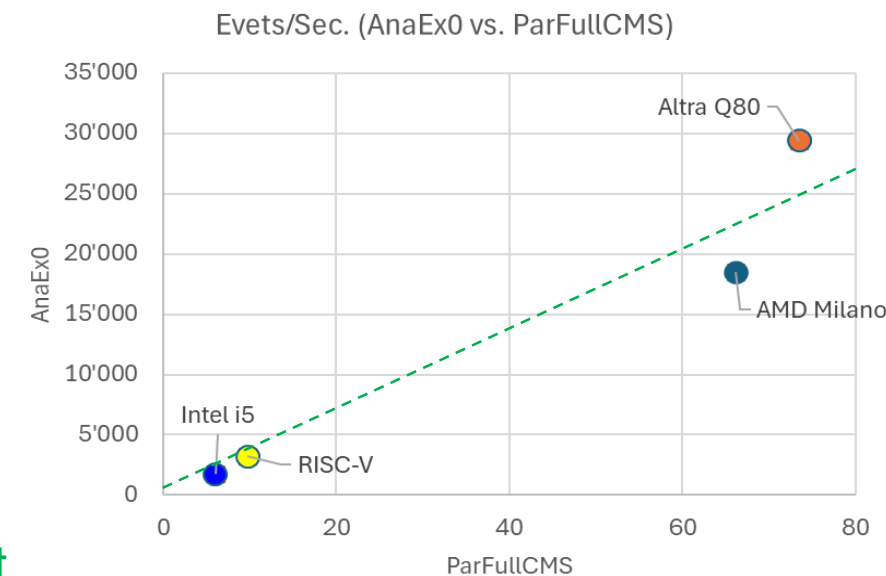
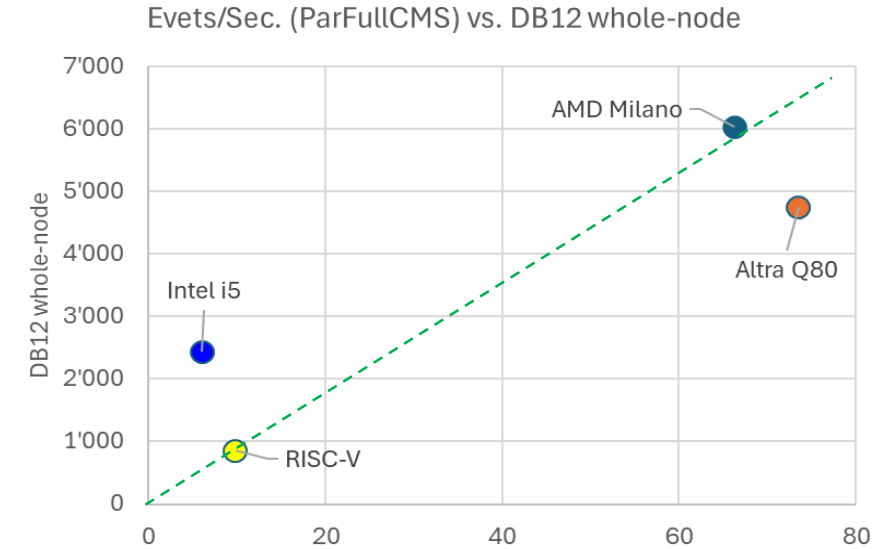
The histogram below compares the results of the benchmarks so far, normalized to the AMD server. On the two servers (**AMD & ARM**), the HEP-Score results are also shown.



As mentioned, the first two benchmark (**ROOT** & **DB12 single**) are single-threaded, the others are multi-threaded (**DB12 whole**, both **Geant4** simulations, and the **HEP-Score**).

The scatter plots on the right show how compatible are their results.

Note: the green line is not an actual fit.



The CMSSW Software Stack

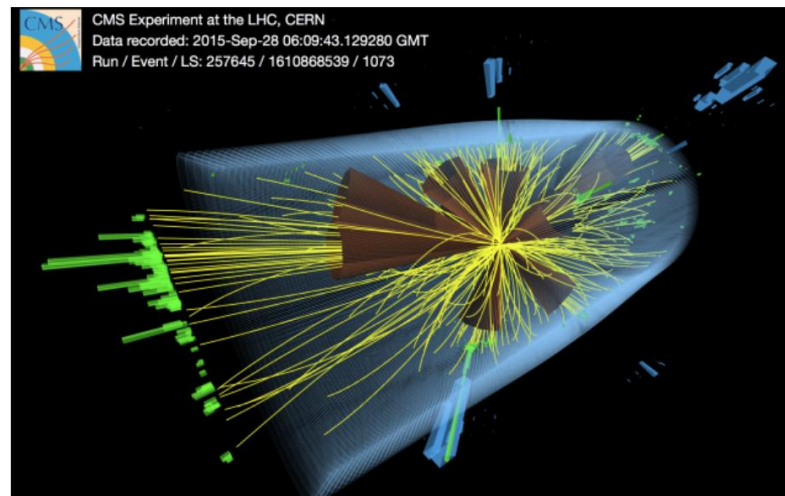
- The software stack for the CMS is available on github, with an Apache 2.0 license
- The core code ([CMSSW](#)) consists of ~ 6 MSLOCs (Python + C++ mostly), and has been development since 2004
 - 1340 “packages” (in ~100 “subsystems”)
- On top of that, external dependencies are packaged by CMS, in order to have a stable computing environment and a low reliance of the host system software
 - ~670 external packages, including gcc, ROOT, Geant4, Tensorflow, PyTorch, ...

- Executes on x86_64, (ppc64le), aarch64; supports CUDA directly and more via Alpaka
- Builds multiple times per day → rpm packages
- Actual deployment in most cases via CVMFS
- All details:

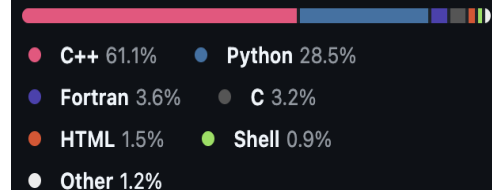
<https://cms-sw.github.io/>


https://cds.cern.ch/record/2725597/files/10.1088_1742-6596_1525_1_012037.pdf

Welcome to CMS and CMSSW



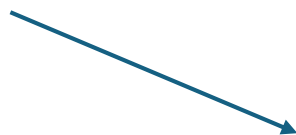
Languages



 Apache-2.0 license

And porting it to RISC-V

- Used Milk-V Pioneer boxes, with Fedora 38
 - Inside this, containers are used to work in a Fedora 39 environment
- Start from externals, when a good fraction is ok you can compile the core code
- Externals
 - 660 compile with gcc13
 - The important missing ones:
 - CUDA, Tensorflow



Google??

<https://www.hpcwire.com/2022/09/23/nvidia-shuts-out-risc-v-software-support-for-gpus/>

- Particularly CUDA and Tensorflow are problematic: many packages depend on them
- Core code:
 - 65 out of 1340 packages do NOT build
 - 2.8k out of 3.5k executable build (vast majority fails linking to CUDA and TF)
 - > 1k unit tests are successful (out of 1.3k)
 - Data reading works, any real processing not possible atm due to the above
- All progresses are in CVMFS:
 - [/cvmfs/cms-ib.cern.ch/sw/riscv64](https://cvmfs.cern.ch/sw/riscv64/)

Conclusions & Outlook

- ❖ Our test-box RISC-V does not excel in performances, however:
 - on most benchmarks (Geant4, DB12 whole) it is on par with a mid-range x86 desktop, both in terms of performance and performance per Watt,
 - we expect the situation to improve at a fast pace.
- ❖ Following up on **RISC-V** updates and integration:
 - we are actively participating in the testing and benchmarking cycles,
 - we have prepared a test suite to evaluate future RISC-V machines,
 - we are looking into new hardware solutions (RISC-V servers?).
- ❖ Keep exploring new hardware:
 - we are waiting for the **512 cores** RISC-V CPU & more ... our “test suite” is ready!



Aknowledgments

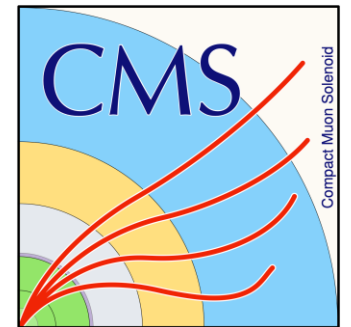
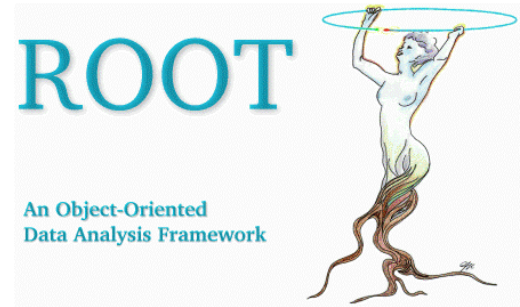
For the hardware:

- University of Glasgow
- ScotGrid* WLCG Tier2
- Bologna University
- CNAF WLCG Tier1
- INFN Pisa
- The Italian National Center for Big Data, HPC and Quantum Computing

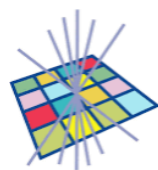


For the code:

- The **ROOT**, **Geant4**, **XrootD**, **CVMFS** teams
- The **CMS** and **LHCb** collaborations



* **ScotGrid Glasgow:** Emanuele Simili, Gordon Stewart, Samuel Skipsey, Albert Borbely, David Britton



end



References

Coming soon ...

Abstract

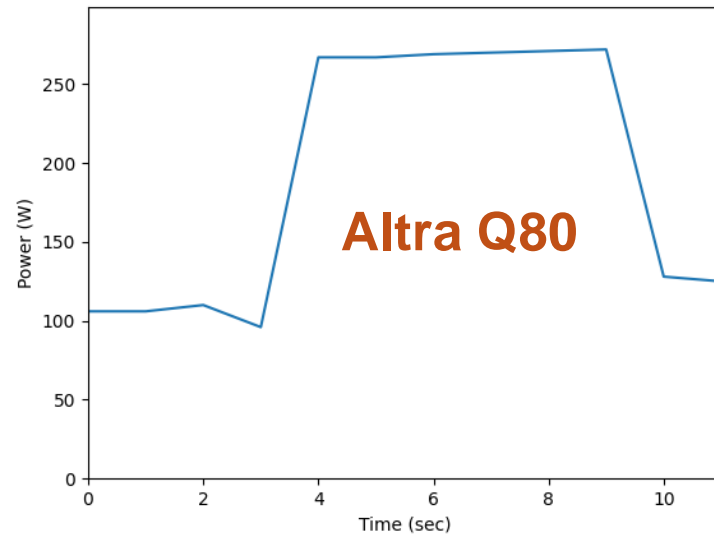
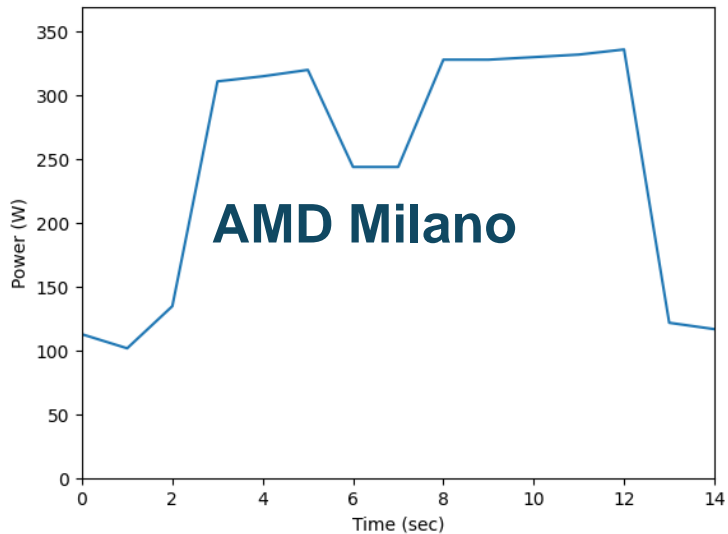
In pursuit of energy-efficient solutions for computing in High Energy Physics (HEP) we have extended our investigations of non-x86 architectures beyond the ARM platforms that we have previously studied. In this work, we have taken a first look at the RISC-V architecture for HEP workloads, leveraging advancements in both hardware and software maturity.

We introduce the Pioneer Milk-V, a 64-core RISC-V machine running Fedora Linux, as our new testbed, available at ScotGrid Glasgow (UK) and INFN Bologna (Italy). Despite this early stage of RISC-V adoption in HEP, significant progress has been made in software compatibility. Standard frameworks such as ROOT, Geant4, CVMFS, and XRootD can be successfully built and run on the RISC-V platform, showcasing the evolving ecosystem. Additionally, efforts are underway to port CMSSW, promising further integration of HEP experiment software. In this first study, we assess performance and power efficiency, and we leverage various benchmarking tools to compare the RISC-V system with existing ARM and x86 architectures. Although it is not yet possible to run the HEP Score suite, we have conducted ROOT tests and benchmarks, along with DB12 and HS06 benchmarks, demonstrating promising performance-per-watt on the RISC-V platform.

These early results suggest that RISC-V architecture holds potential for advancing energy-efficient computing in HEP, offering decent performance and significantly better power efficiency, while contributing to an increasingly heterogeneous computing landscape.

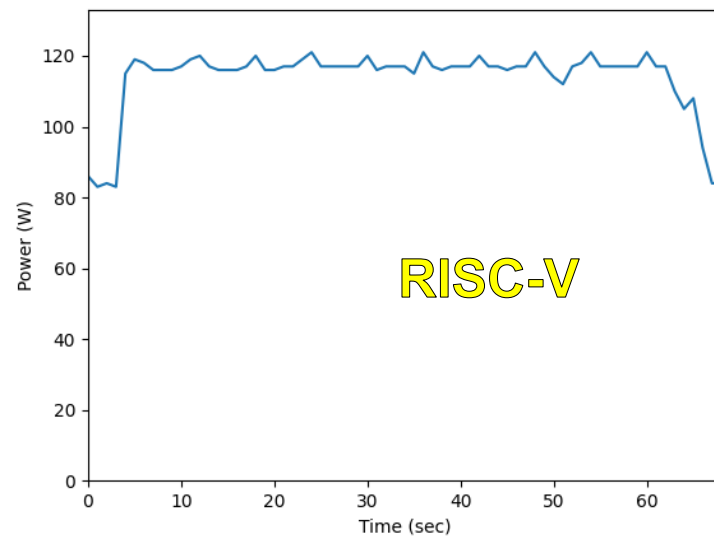
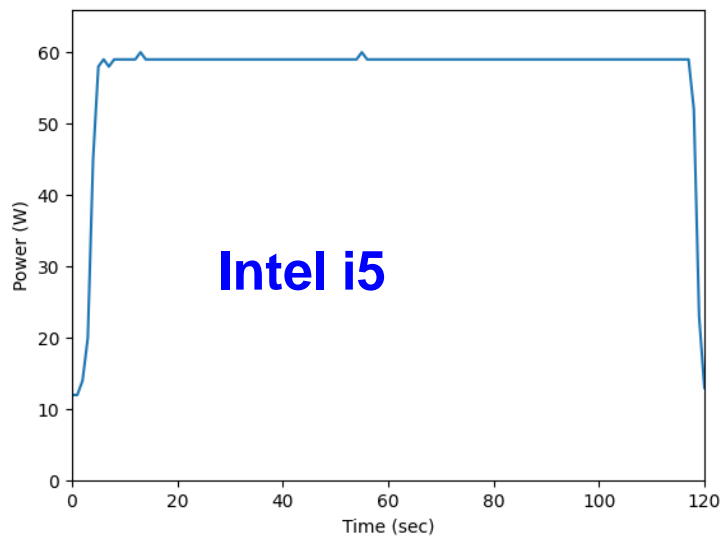
G4 AnaEx01 Power Profiles

Power vs. time for the **Geant4** simulation example (**AnaEx01**) on the 4 types of hardware:



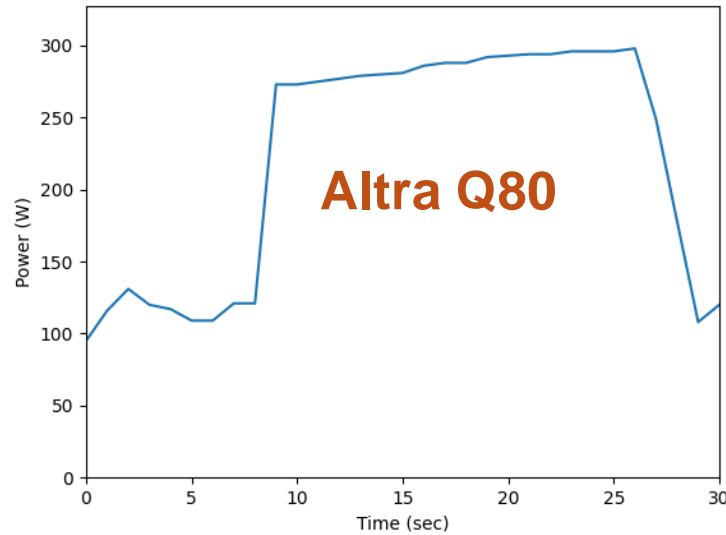
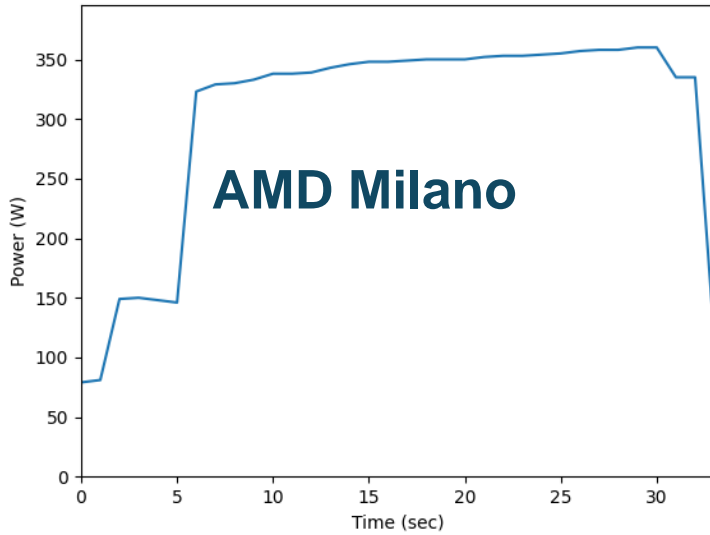
Note: 1 Million AnaEx01 events are generated in a matter of seconds on server-level hardware !

For a better power measurement we may want to run for longer ...

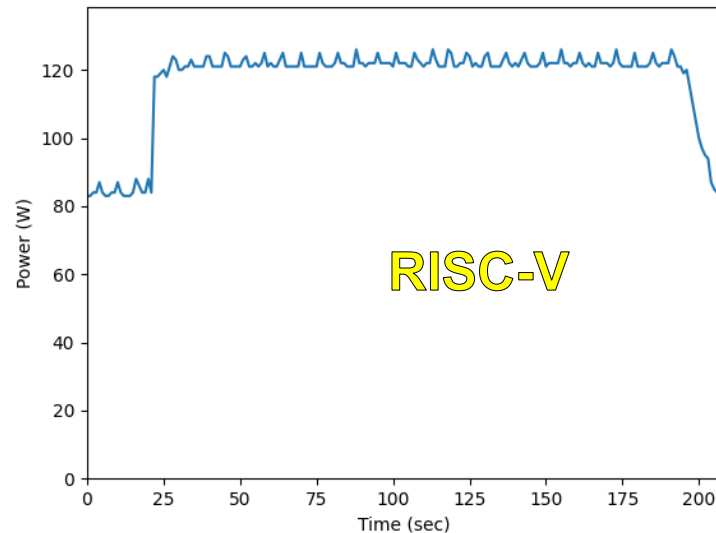
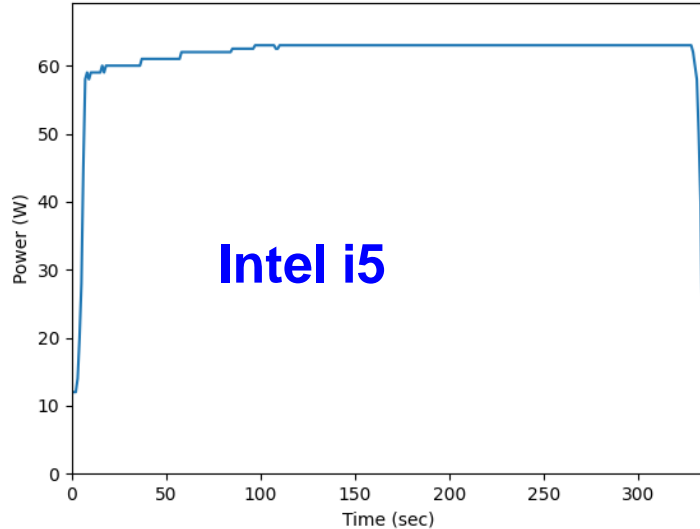


G4 ParFullCMS Power Profiles

Power vs. time for the **Geant4** simulation example (**ParFullCMS**) on the 4 types of hardware:



Note: 10 Thousand ParFullCMS events are generated in half minute on server-level hardware !

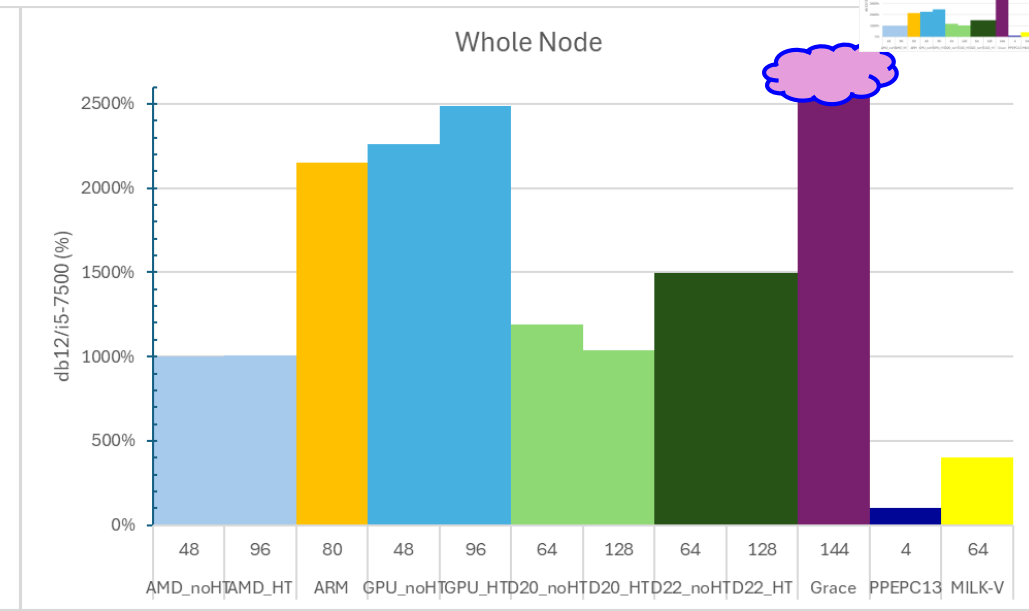
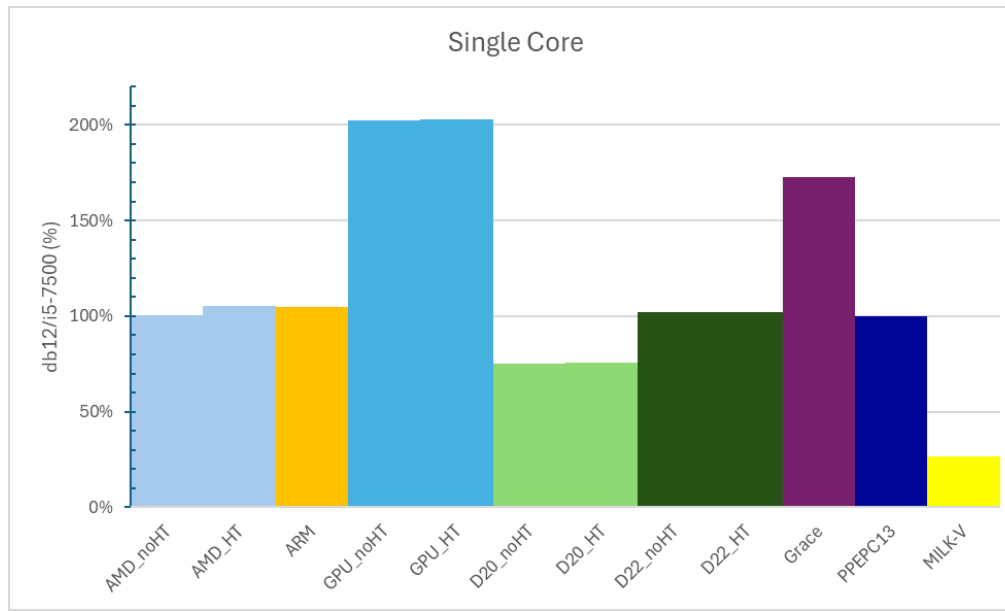


Results (preliminary)

We could run **ROOT tests**, **ROOT benchmarks**, and **DB12** ... results are a bit tricky to compare, because each benchmark has its “peculiarities” and not all machines can run all benchmarks.

machine	nproc	CPU	single	whole	single_core		whole_node		HEPscore		ROOT benchmark	
			db12_single	db12_wholenode	/i5-7500	/AMD_noHT	/i5-7500	/AMD_noHT	/AMD_noHT	/AMD_noHT		
AMD_noHT	48	AMD EPYC 7643 48-Core Processor	23.85752688	938.0106703	100%	100%	1003%	100%	1,169.49	100%	5,853.11	100%
AMD_HT	96	//	24.96483826	939.6929573	105%	105%	1005%	100%	1,341.37	115%	5,703.80	97%
ARM	80	Ampere Altra Q80-30 Processor	24.90039841	2010.311923	105%	104%	2151%	214%	1,467.27	125%	4,430.31	76%
GPU_noHT	48	2 * AMD EPYC 7443 24-Core Processor	48.07692308	2115.670718	202%	202%	2263%	226%		0%		0%
GPU_HT	96	//	48.16955684	2323.572578	203%	202%	2486%	248%		0%		0%
D20_noHT	64	2 * AMD EPYC 7452 32-Core Processor	17.91120081	1111.082808	75%	75%	1189%	118%	1,479.85	127%		0%
D20_HT	128	//	18.03861789	970.5602849	76%	76%	1038%	103%	1,826.35	156%		0%
D22_noHT	64	2 * AMD EPYC 7513 32-Core Processor	24.31506849	1401.588433	102%	102%	1499%	149%	1,682.17	144%		0%
D22_HT	128	//	24.28180575	1396.361662	102%	102%	1494%	149%	1,928.05	165%		0%
Grace	144	2 * NVidia Grace 72-Core	40.98360656	5903.329871	173%	172%	6315%	629%	4,205.43	360%		0%
PPEPC13	4	Intel Core i5-7500 CPU @ 3.4GHz	23.75690608	93.47930739	100%	100%	100%	10%		0%	3,885.49	66%
MILK-V	64	Milk-V riscv64	6.311537491	375.5319286	27%	26%	402%	40%		0%	786.24	13%

So far, **DB12** is the only benchmark that can run on all hardware !



2014 – A cute, open ISA



Recommendations
and Roadmap
for European Sovereignty
in Open Source
Hardware, Software,
and RISC-V Technologies

Report from the
Open Source Hardware & Software Working Group

August 2022



2023 – Disruptive Force

2023 RISC-V International more than 26% membership growth year-over-year, with over 3,180 members across 70 countries. More than 10 billion RISC-V cores in the market, 10K+ engineers working on RISC-V



Members

