Watt Counts: ARM Compute & Energy Accounting in the WLCG

Emanuele Simili, Gordon Stewart, Samuel Skipsey Albert Borbely & David Britton (**ScotGrid Glasgow**)

WLCG Environmental Sustainability Workshop @ CERN - 12 December 2024

Outline

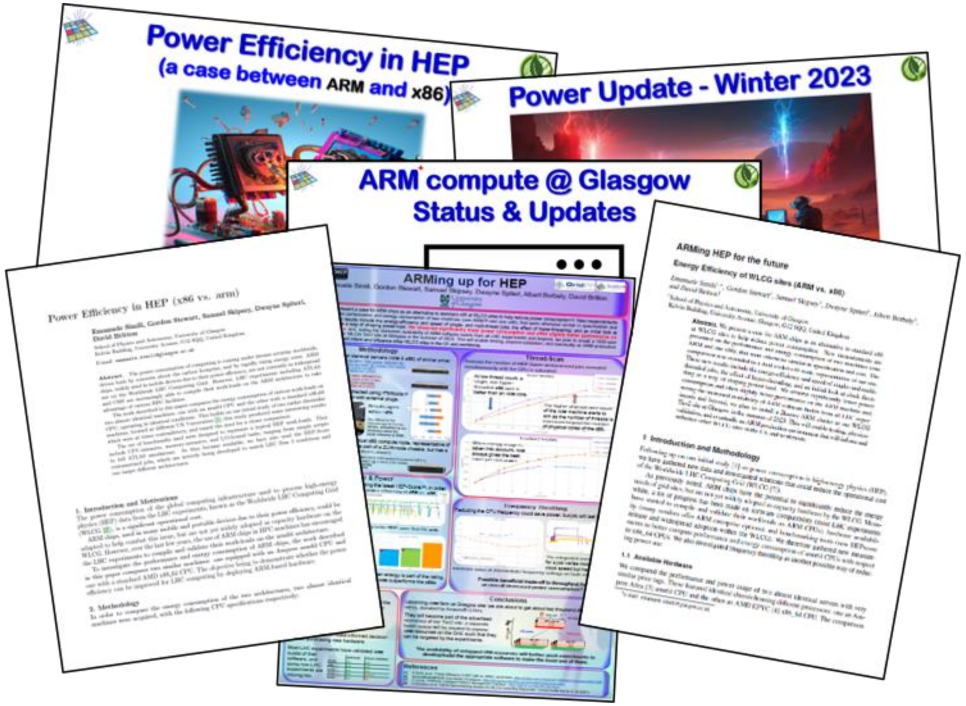
- Performance per Watt (**x86** vs **ARM**)
  - motivations & methodology
  - updated **HS23/Watt** and **Frequency Scan** results
  - data processing improvements (thanks **HEP-Score working group**)

- Heterogeneous Tier2 Cluster @ **ScotGrid Glasgow**
  - configuration and dual queue management
  - ARM energy savings

- Toward a WLCG global CO_2 accounting …
  - what is currently measured (**ATLAS PanDA**)
  - what we can measure more precisely (kWh)
  - accounting strategy and Proof of Concept

Motivations & Methodology

In 2021 we started investigating alternative architectures for Grid computing, starting with **ARM** chips …



Lot has happened since then:

- most LHC experiments ported their software to ARM,
- physics validations had been performed,
- heterogeneous computing cluster set-up (x86 + ARM),
- HEP-Score collaboration and improved methodology,
- dissemination of results, …

Methodology:

As benchmark, we rely on the HEP-Score & the HEP-Benchmarking Suite:

HEP-Suite: https://gitlab.cern.ch/hep-benchmarks/hep-benchmark-suite
HEP-Score: https://gitlab.cern.ch/hep-benchmarks/hep-score

As for collecting metrics, so far I have been using my own script to exports CPU, RAM, Frequency and Power usage (via **IPMI tools:** https://github.com/ipmitool/ipmitool) into a CSV file … but now the HEP-Suite itself provides such metric exporters as plug-ins.

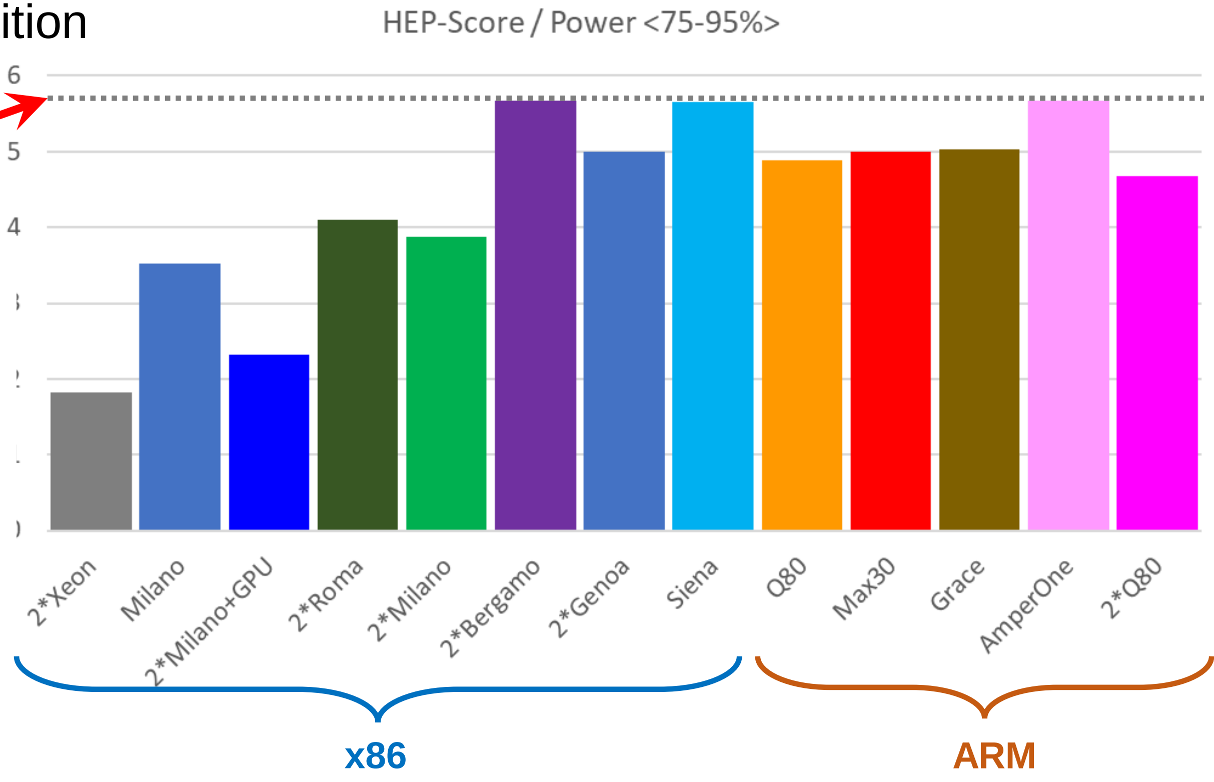
Results are then processed to generate plots, integrate the energy usage, and do some statistical calculations.
The final output (**HS23/Watt**) enable us to compare various machine based on performance per unit power.

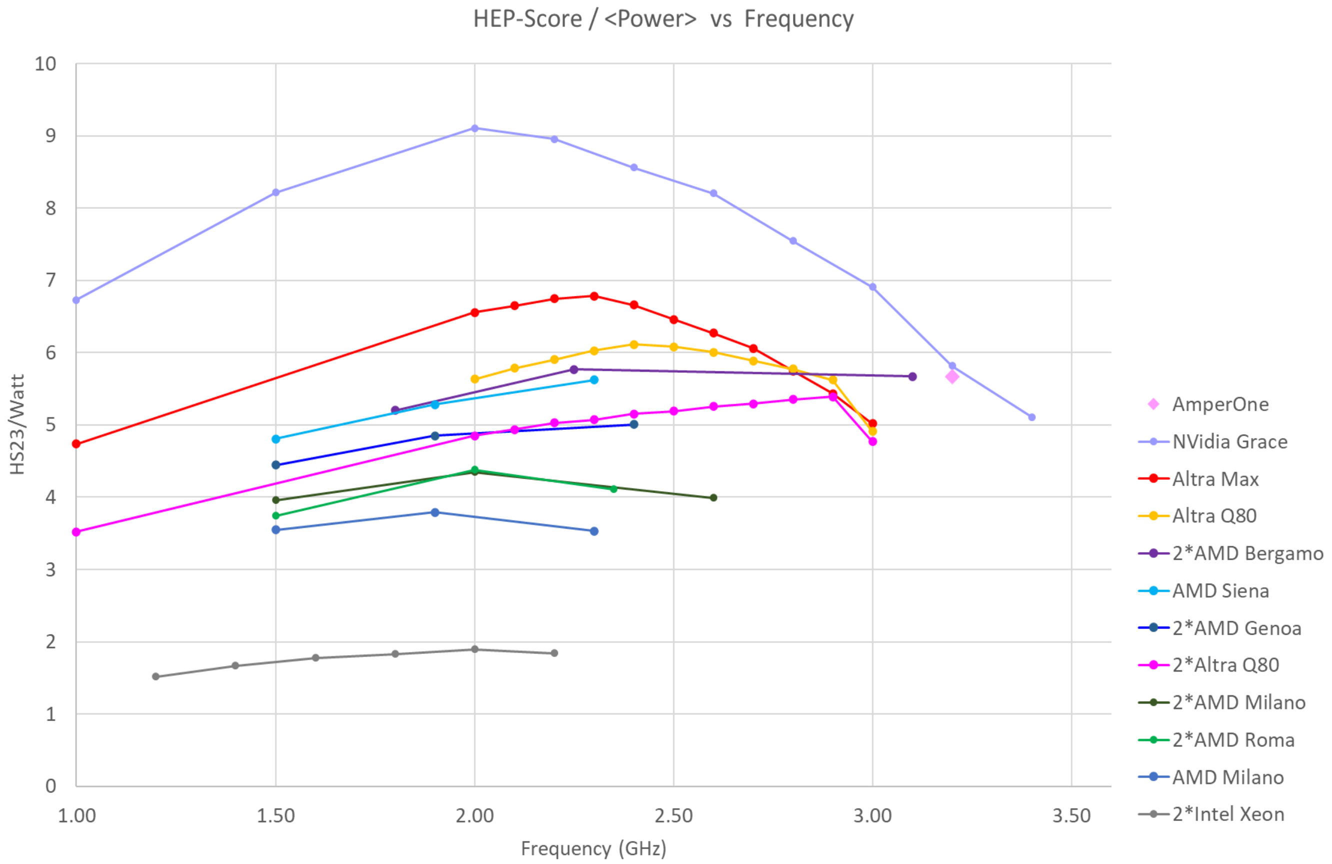
Performance/Watt (ARM vs. x86)

To account for the shape of HEP-Score workloads, we calculate the **Figure of Merit** (**FoM**) as: **HS23/Power<75-95%>**
And, finally, we can see how various machines compare against each other.

The trend of power optimization has sparked a healthy competition between hardware manufacturers …

<u>New generation **x86** now match **ARM**s performance/watt!</u>





HEP-Score/Watt vs. **CPU Frequency** gives a more complete picture, and shows optimal performance per watt at mid frequency range. Also, **ARM** CPUs allow for a finer tuning of the clock speed to obtain a better **HS23/Watt** (for a slightly longer run-time).

Data Processing (past)

This was the flow, from data collection to processing and visualization …

jdump.sh
(**root**'s script)

/tmp/ipmidump.json


```
{
  "date_yyyymmdd": "$DATE" ,
  "time_hhmmss": "$TIME" ,
  "cpu_usage_percent": "$CPUSE" ,
  "memory_usage_gb": "$MEMUSE" ,
  "cpu_frequency_ghz": "$FRE
  "ipmi_power_watt": "$I
}
```


volatile

jget.sh
(**user**'s script)

ipmi_runtime.csv

date_yyyymmdd	time_hhm	cpu_usage	memory_	cpu_frequ	ipmi_pow
08/06/2023	23:37:44	1.1	5.51	1	88
08/06/2023	23:37:49	0.3	5.52	1	88
08/06/2023	23:37:54	0.3	5.51	1	89
08/06/2023	23:37:59	0.8	5.51	1	89
08/06/2023	23:38:04	1.1	5.53	1	88
08/06/2023	23:38:09	0.3	5.53	1	88
08/06/2023	23:38:14	0.9	5.53	1	89
08/06/2023	23:38:19	0.5	5.53	1	89
08/06/2023	23:38:24	1	5.54	1	[illegible]
08/06/2023	23:38:29	0.7	5.54	1	[illegible]
08/06/2023	23:38:34	0.8	5.55	[illegible]	[illegible]
08/06/2023	23:38:39	0.7	5.53	[illegible]	[illegible]
08/06/2023	23:38:44	1.3	[illegible]	[illegible]	[illegible]
08/06/2023	23:38:49	1.4	[illegible]	[illegible]	89
08/06/2023	23:38:54	[illegible]	[illegible]	[illegible]	92

persistent

run_HEPscore.sh
(**user**'s script)

bmkrun_report.json


```
    "cms-reco-run3-ma-bmk": {
        "reco": 11.7333,
        "reco_ref": 4.814
    },
    "lhcb-sim-run3-ma-bmk": {
        "sim": 4825.6219,
        "sim_ref": 1950
    },
    "belle2-gen-sim-reco-ma-bmk": {
        "gen-sim-reco": 36.84867737547333,
        "gen-sim-reco_ref": 15.4
    },
    "alice-digi-reco-core-run3-ma-bmk": {
        "digi-reco": 1.9819,
        "digi-reco_ref": 0.762
    }
},
"score": 2528.2055,
"status": "success"
```

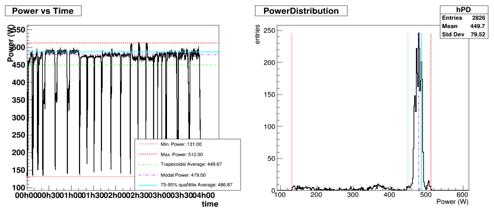

persistent

ipmi2root.C

ROOT

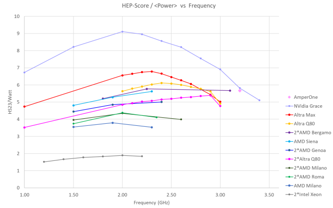
An Object-Oriented Data Analysis Framework

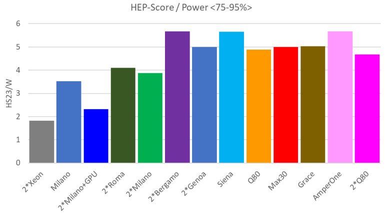
```
Machine , Time (s) , Energy(kW*h) , CPU min (%) , CPU max (%) , Freq min (GHz) , Fr

AltraMax , 15349 , 1.06719 , 0.2 , 100.0 , 2.5 , 2.5 , 8.1 , 255.4 , 97 , 306 , 10
AltraQ80 , 16806 , 1.99072 , 0.4 , 100.0 , 1.0 , 3.0 , 5.9 , 344.4 , 176 , 576 ,
Grace , 9285 , 1.51394 , 0.3 , 100.0 , 0.1 , 3.4 , 5.3 , 334.2 , 157 , 862 , 587 ,
Bergamo , 21541 , 5.46918 , 0.1 , 100.0 , 1.8 , 1.8 , 14.8 , 927.6 , 163 , 1118 , 2
Siena , 13954 , 1.22110 , 0.1 , 100.0 , 2.3 , 3.1 , 2.8 , 211.5 , 139 , 411 , 147.0
Milano , 13578 , 1.72491 , 0.1 , 100.0 , 1.6 , 3.7 , 4.9 , 215.4 , 59 , 511 , 108.
IntelXeon , 23771 , 1.24745 , 0.1 , 100.0 , 2.2 , 2.2 , 2.4 , 69.2 , 69 , 235 , 18
...
```



Excel

Nickname	Machine	CPU	Arch	HT	Threads	Governor	Max Freq. (GHz)
2*Xeon	2xIntel20ht	2 * Intel XEON 10-Core CPU E5-2630 v4	2*x86_64	on	40	conservative	2.2
Milano	AMD96ht	AMD EPYC 7643 48-Core Processor	x86_64	on	96	conservative	2.3
2*Milano+GPU	2*AMD48ht_gpu	2 * AMD EPYC 7443 24-Core Processor + 2* NVIDIA A100 PCIe 80GB	2*x86_64	on	96	conservative	4.0
2*Roma	2xAMD64ht	2 * AMD EPYC 7452 32-Core Processor	2*x86_64	on	128	conservative	3.3
2*Milano	2xAMD64ht_m	2 * AMD EPYC 7513 32-Core Processor	2*x86_64	on	128	conservative	2.6
2*Bergamo	2xAMD256ht	2 * AMD EPYC 9754 128-Core Processor	2*x86_64	on	512	conservative	3.1
2*Genoa	2xAMD192ht_cor	2 * AMD EPYC 9654 96-Core Processor	2*x86_64	on	384	conservative	3.7
Siena	AMD128ht	AMD EPYC 8534P 64-Core Processor	x86_64	on	128	conservative	3.1
Q80	ARM80c	Ampere Altra Q80-30	aarch64	//	80	conservative	3.0
Max28	ARM128c_2.8	Ampere Altra Max M128-28	aarch64	//	128	conservative	2.8
Max30	ARM128c	Ampere Altra Max M128-30	aarch64	//	128	conservative	3.0
Grace	NVidia144c	NVidia Grace 144-Core 480GB DDR5	2*aarch64	//	144	conservative	3.4
2*Q80	2xARM80c	2 * Ampere Altra Q80-30	2*aarch64	//	160	conservative	3.0





Data Processing (present)

In view of CHEP 2024 and to populate our web interface, I simplified the flow with a Python script …

jdump.sh
(**root**'s script)

/tmp/ipmidump.json


```
{
  "date_yyyymmdd": "$DATE" ,
  "time_hhmmss": "$TIME" ,
  "cpu_usage_percent": "$CPUSE" ,
  "memory_usage_gb": "$MEMUSE" ,
  "cpu_frequency_ghz": "$FRE…
  "ipmi_power_watt": "$I…
}
```


volatile

jget.sh
(**user**'s script)

ipmi_runtime.csv

date_yyyymmdd	time_hhm	cpu_usage	memory_u	cpu_frequ	ipmi_pow
08/06/2023	23:37:44	1.1	5.51	1	88
08/06/2023	23:37:49	0.3	5.52	1	88
08/06/2023	23:37:54	0.3	5.51	1	89
08/06/2023	23:37:59	0.8	5.51	1	89
08/06/2023	23:38:04	1.1	5.53	1	88
08/06/2023	23:38:09	0.3	5.53	1	88
08/06/2023	23:38:14	0.9	5.53	1	89
08/06/2023	23:38:19	0.5	5.53	1	89
08/06/2023	23:38:24	1	5.54	1	[illegible]
08/06/2023	23:38:29	0.7	5.54	1	[illegible]
08/06/2023	23:38:34	0.8	5.55	[illegible]	[illegible]
08/06/2023	23:38:39	0.7	5.53	[illegible]	[illegible]
08/06/2023	23:38:44	1.3	[illegible]	[illegible]	[illegible]
08/06/2023	23:38:49	1.4	[illegible]	[illegible]	89
08/06/2023	23:38:54	[illegible]	[illegible]	[illegible]	92

persistent

run_HEPscore.sh
(**user**'s script)

bmkrun_report.json


```
"cms-reco-run3-ma-bmk": {
    "reco": 11.7333,
    "reco_ref": 4.814
},
"lhcb-sim-run3-ma-bmk": {
    "sim": 4825.6219,
    "sim_ref": 1950
},
"belle2-gen-sim-reco-ma-bmk": {
    "gen-sim-reco": 36.84867737547333,
    "gen-sim-reco_ref": 15.4
},
"alice-digi-reco-core-run3-ma-bmk": {
    "digi-reco": 1.9819,
    "digi-reco_ref": 0.762
}
},
"score": 2528.2055,
"status": "success"
```

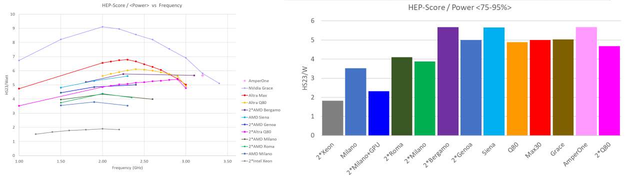

persistent

pow_report.json

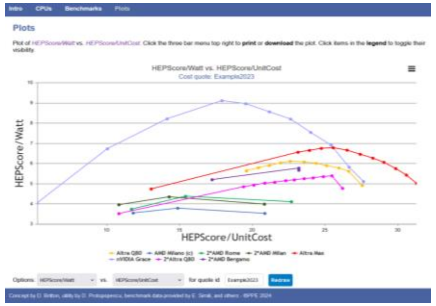
j2r.py

Excel

Nickname	Machine	CPU	Arch	HT	Threads	Governor	Max Freq. (GHz)
2*Xeon	2xIntel20ht	2 * Intel XEON 10-Core CPU E5-2630 v4	2*x86_64	on	40	conservative	2.2
Milano	AMD96ht	AMD EPYC 7643 48-Core Processor	x86_64	on	96	conservative	2.3
2*Milano+GPU	2*AMD48ht_gpu	2 * AMD EPYC 7443 24-Core Processor + 2* NVIDIA A100 PCIe 80GB	2*x86_64	on	96	conservative	4.0
2*Roma	2xAMD64ht	2 * AMD EPYC 7452 32-Core Processor	2*x86_64	on	128	conservative	3.3
2*Milano	2xAMD64ht_m	2 * AMD EPYC 7513 32-Core Processor	2*x86_64	on	128	conservative	2.6
2*Bergamo	2xAMD256ht	2 * AMD EPYC 9754 128-Core Processor	2*x86_64	on	512	conservative	3.1
2*Genoa	2xAMD192ht_cor	2 * AMD EPYC 9654 96-Core Processor	2*x86_64	on	384	conservative	3.7
Siena	AMD128ht	AMD EPYC 8534P 64-Core Processor	x86_64	on	128	conservative	3.1
Q80	ARM80c	Ampere Altra Q80-30	aarch64	//	80	conservative	3.0
Max28	ARM128c_2.8	Ampere Altra Max M128-28	aarch64	//	128	conservative	2.8
Max30	ARM128c	Ampere Altra Max M128-30	aarch64	//	128	conservative	3.0
Grace	NVidia144c	NVidia Grace 144-Core 480GB DDR5	2*aarch64	//	144	conservative	3.4
2*Q80	2xARM80c	2 * Ampere Altra Q80-30	2*aarch64	//	160	conservative	3.0



Web-Import



SmartPro Web:
https://www.ppe.gla.ac.uk/smartpro/

Data Processing (future)

With the new HEP-Score plugins, the flow will simplify even further …

`jdump.sh`
(**root**’s script)

`/tmp/ipmidump.json`


```
{
  "date_yyyymmdd": "$DATE" ,
  "time_hhmmss": "$TIME" ,
  "cpu_usage_percent": "$CPUSE" ,
  "memory_usage_gb": "$MEMUSE" ,
  "cpu_frequency_ghz": "$FRE
  "ipmi_power_watt": "$
}
```


volatile

`run_HEPscore.sh`
(**user**’s script)

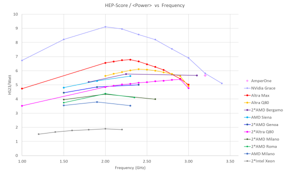
`bmkrun_report.json`

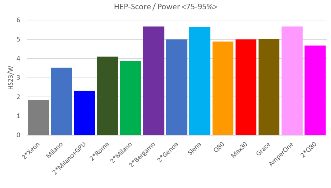

```
    "cms-reco-run3-ma-bmk": {
        "reco": 11.7333,
        "reco_ref": 4.814
    },
    "lhcb-sim-run3-ma-bmk": {
        "sim": 4825.6219,
        "sim_ref": 1950
    },
    "belle2-gen-sim-reco-ma-bmk": {
        "gen-sim-reco": 36.84867737547333,
        "gen-sim-reco_ref": 15.4
    },
    "alice-digi-reco-core-run3-ma-bmk": {
        "digi-reco": 1.9819,
        "digi-reco_ref": 0.762
    }
},
"score": 2528.2055,
"status": "success"
```


persistent

In particular, the HEP-Score plugins produce a power report within the standard JSON output (`bmkrun_report.json`) …

Nickname	Machine	CPU	Arch	HT	Threads	Governor	Max Freq. (GHz)
*2*Xeon*	2xIntel20ht	2 * Intel XEON 10-Core CPU E5-2630 v4	2*x86_64	on	40	conservative	2.2
Milano	AMD96ht	AMD EPYC 7643 48-Core Processor	x86_64	on	96	conservative	2.3
*2*Milano+GPU*	2*AMD48ht_gpu	2 * AMD EPYC 7443 24-Core Processor + 2* NVIDIA A100 PCIe 80GB	2*x86_64	on	96	conservative	4.0
*2*Roma*	2xAMD64ht	2 * AMD EPYC 7452 32-Core Processor	2*x86_64	on	128	conservative	3.3
*2*Milano*	2xAMD64ht_m	2 * AMD EPYC 7513 32-Core Processor	2*x86_64	on	128	conservative	2.6
*2*Bergamo*	2xAMD256ht	2 * AMD EPYC 9754 128-Core Processor	2*x86_64	on	512	conservative	3.1
*2*Genoa*	2xAMD192ht_cor	2 * AMD EPYC 9654 96-Core Processor	2*x86_64	on	384	conservative	3.7
Siena	AMD128ht	AMD EPYC 8534P 64-Core Processor	x86_64	on	128	conservative	3.1
Q80	ARM80c	Ampere Altra Q80-30	aarch64	//	80	conservative	3.0
Max28	ARM128c_2.8	Ampere Altra Max M128-28	aarch64	//	128	conservative	2.8
Max30	ARM128c	Ampere Altra Max M128-30	aarch64	//	128	conservative	3.0
Grace	NVidia144c	NVidia Grace 144-Core 480GB DDR5	2*aarch64	//	144	conservative	3.4
*2*Q80*	2xARM80c	2 * Ampere Altra Q80-30	2*aarch64	//	160	conservative	3.0





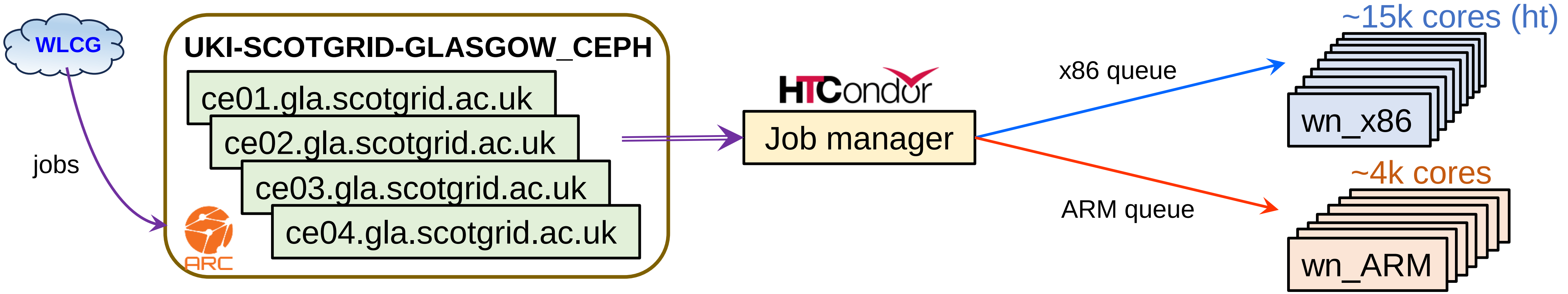
`untitled_analyser.py`

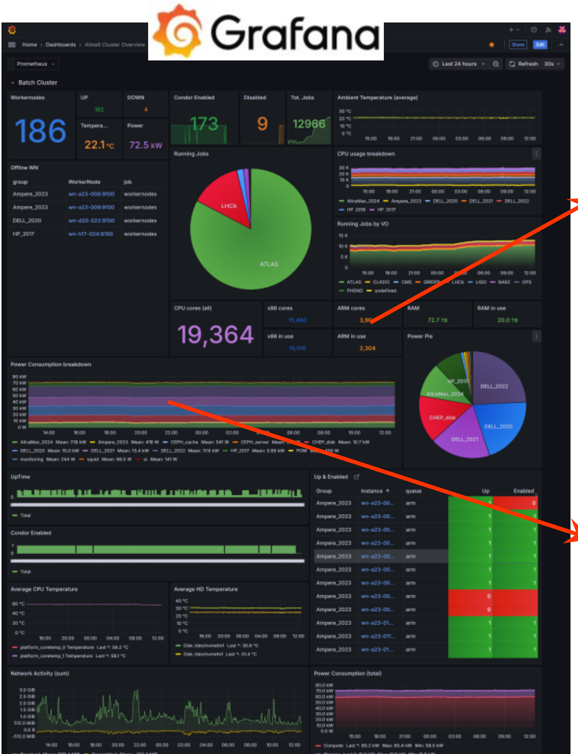
The only thing left to do is divide the score by a sensitive **F.o.M.** (<75-95%> ≈ q75 ≈ q85 ≈ K-Mean)
See study by *Kacper Kamil Kozik*:
https://indico.cern.ch/event/1433496/

And … add the power metric (and the **HS23/Watt**) to the **HEP-Score DB** !

Heterogeneous Compute Cluster @ ScotGrid

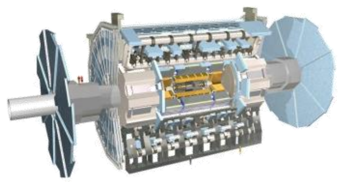
The **ScotGrid Glasgow** Tier2 cluster provides **ARM** resources to the WLCG via standard job submission endpoints.

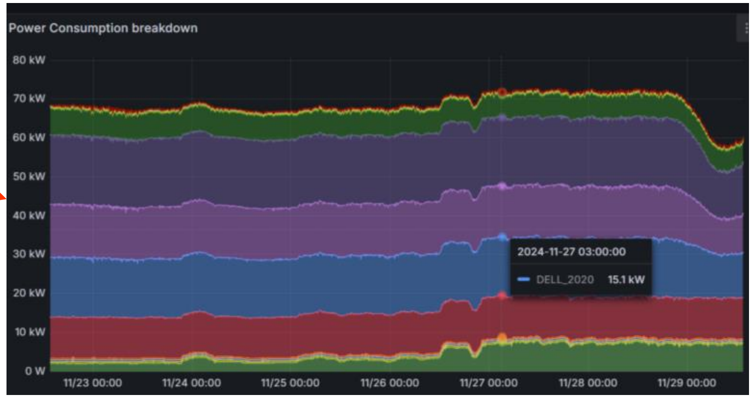




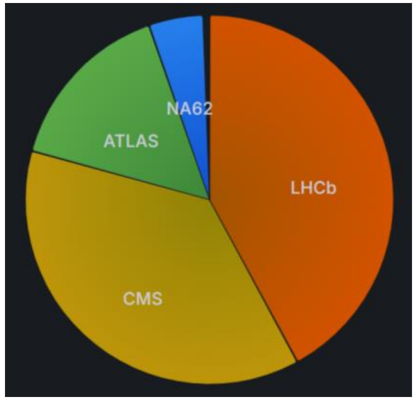
Our monitoring system provides real-time view of the cluster via a **Grafana** dashboard.

At present, **ARM** resources are mainly used by the **ATLAS collaboration**, after they have successfully completed a physics validation campaign on our cluster.





We monitor power usage at machine level, and by combining it with job runtime information, we aim to set up a **CO_2 accounting system** for Virtual Organizations (VOs) within the WLCG ...



ARM energy savings

Calculating energy savings at our site is a little tricky, because ARM resources are provided as an extra (overpledged): we are offering ARM computing resources to whoever can use them (especially, ATLAS).

But, if we compare the power usage of our latest ARMs machines (**Ampere AltraMax**) with the power that would have been used to run the same jobs on our older x86 infrastructure (**AMD EPYC Milan**), and scale by the number of jobs each VO has been running:

$$\text{Power Saved} = \text{n\_cores} * (\text{Power/core}_{x86} - \text{Power/core}_{ARM}) \approx \text{Power}_{ARM} * \Delta_{HS23/W}\,(\text{AltraMax} - \text{AMD Milan})$$

For example: ATLAS has been running a stable enough chunk of jobs on ARM last month, with an average usage 2k cores utilization …

VO **ATLAS**:
ARM: 3760.25 kWh
x86: 26086.83 kWh

→ $\text{Savings}_{\textbf{ATLAS}} = 3760\ \text{kWh} * 0.2 = 752\ \text{kWh}$

preliminary

VO **LHCb**:
ARM: 3720.07 kWh
x86: 25797.80 kWh

→ $\text{Saving}_{\textbf{LHCb}} = 3720\ \text{kWh} * 0.2 = 744\ \text{kWh}$

Note: all this was developed during the last week, using the existing monitoring infrastructure and a couple of hand-crafted scripts to export and process Prometheus data. Numbers may not be accurate!

Toward a site-level CO_2 accounting?

Fernando Barreiro Megino has implemented a rough CO_2 calculator in **ATLAS PanDA**, based on the following formula: (https://panda-wms.readthedocs.io/en/latest/advanced/carbon_footprint.html)

$$n\_cores \times core\_power\_consumption \times \int_{starttime}^{endtime} emission\_intensity(t)dt$$

How power hungry the CC/HW is (currently hardcoded to 10W)

How CO2 intense the local power is

Easy number

Easy to estimate (site specific)

Very hard!

This number comes straight out of any monitoring system (local Prometheus, ATLAS PanDA, …)

ATLAS hardcoded this as **10W**, but we can do better by looking at actual hardware at our site …

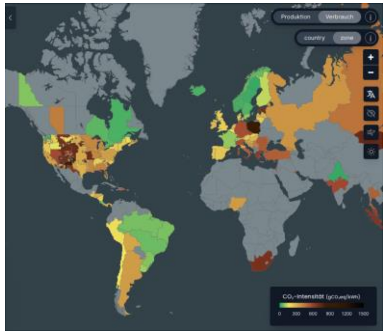


Image from https://app.electricitymaps.com/map

Estimating the **CO_2/kWh** is a complex issue, which depends on specific country, geographic area, electricity demand, time of the year, time of the day, … any power grid has a variable fraction of renewable vs. non-renewable power, and being able to track this in real time is complicated. But, we don't have to do it ourself !

Toward a site-level Power accounting

So … to avoid headaches and stick to hard data, we start from a straight power accounting.
This can be done quite precisely at any WLCG site. Here our strategy:

1) Gather Metrics from local Prometheus:
   - Resource utilization by VO: `(count(node_condor_cpu{job="workernodes"}) by (vo))`
   - Cluster-wide power consumption: `(node_power_watts{job="workernodes"})`

2) Perform some Calculations:
   - Total Power = Active power + Idle power
   - VO-specific power = %_usage × Total Power

   Optional: adjust for machine types and specific job efficiencies (for now we only split between **ARM** & **x86**).

3) Generate Reports:
   - Produce tables or charts showing energy usage per VO.

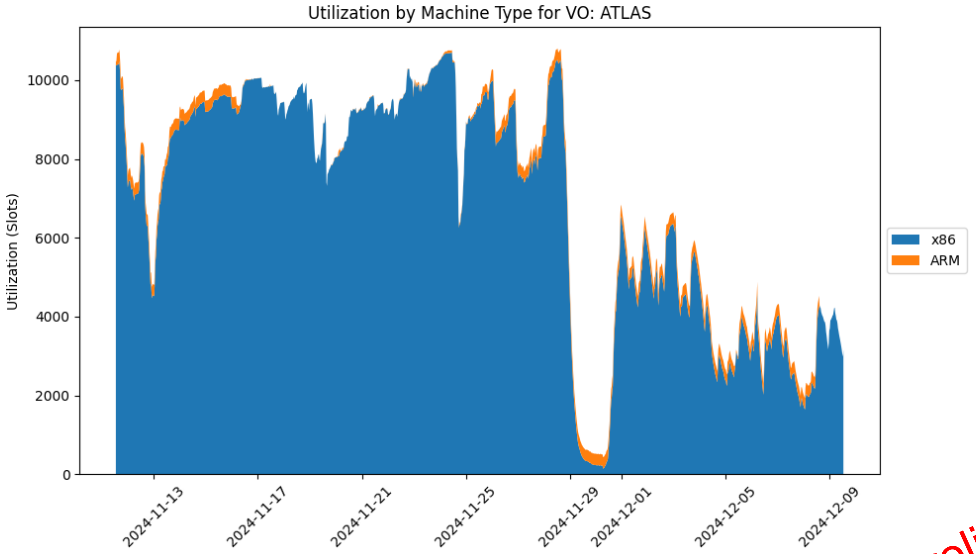
---

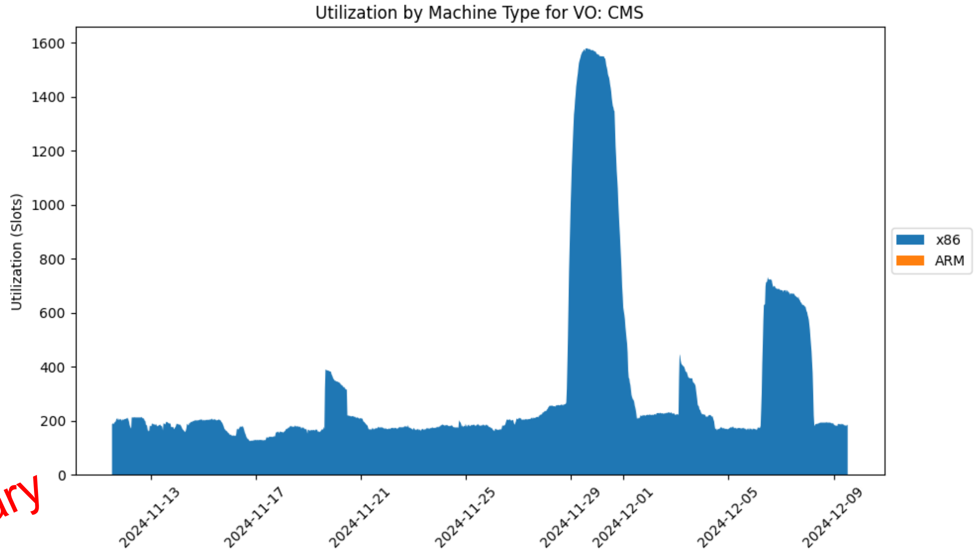
4) From Power to CO_2 usage can be as simple as multiplying by the country average CO2/kWh, or as complicated as integrating the convolution of Power Usage * instantaneous local CO_2/kWh during the sampled period:

$$\int_{starttime}^{endtime} power\_consumption \times emission\_intensity(t)dt$$

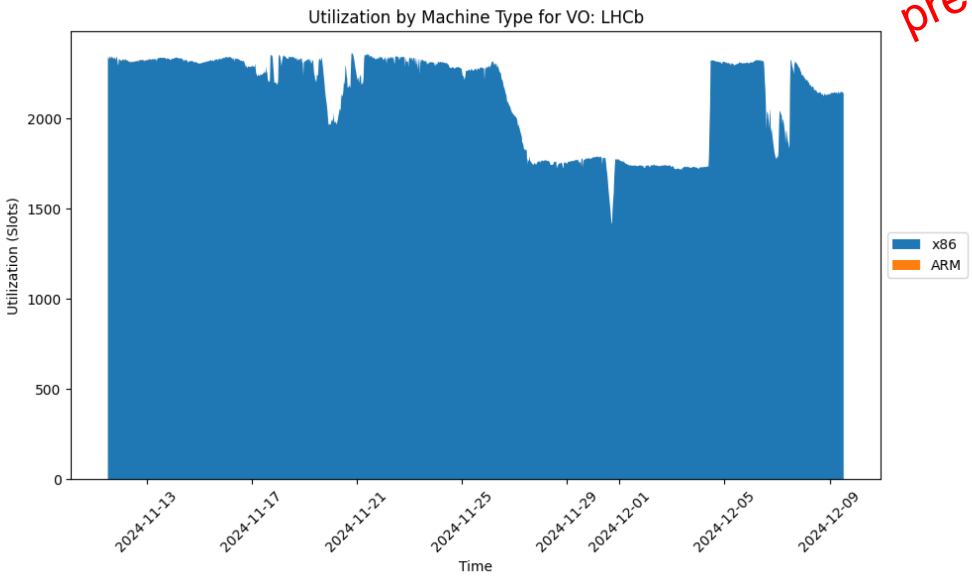
Proof of Concept

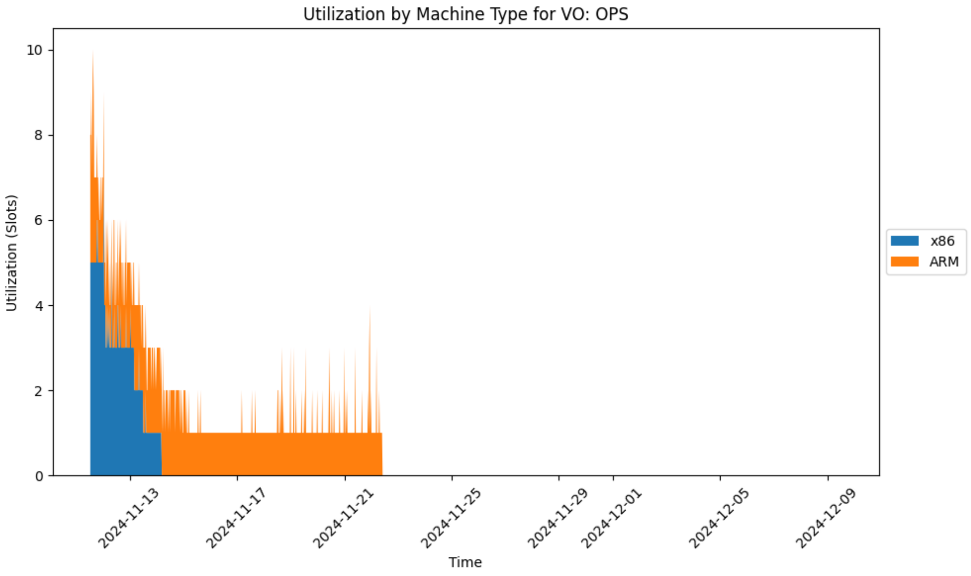
So far, I made a **Python** script that exports metrics from the **Prometheus API** and does some calculations.
The first step is extracting Job-Slots occupancy by VO (and Architecture). See examples in 1 month period:





preliminary

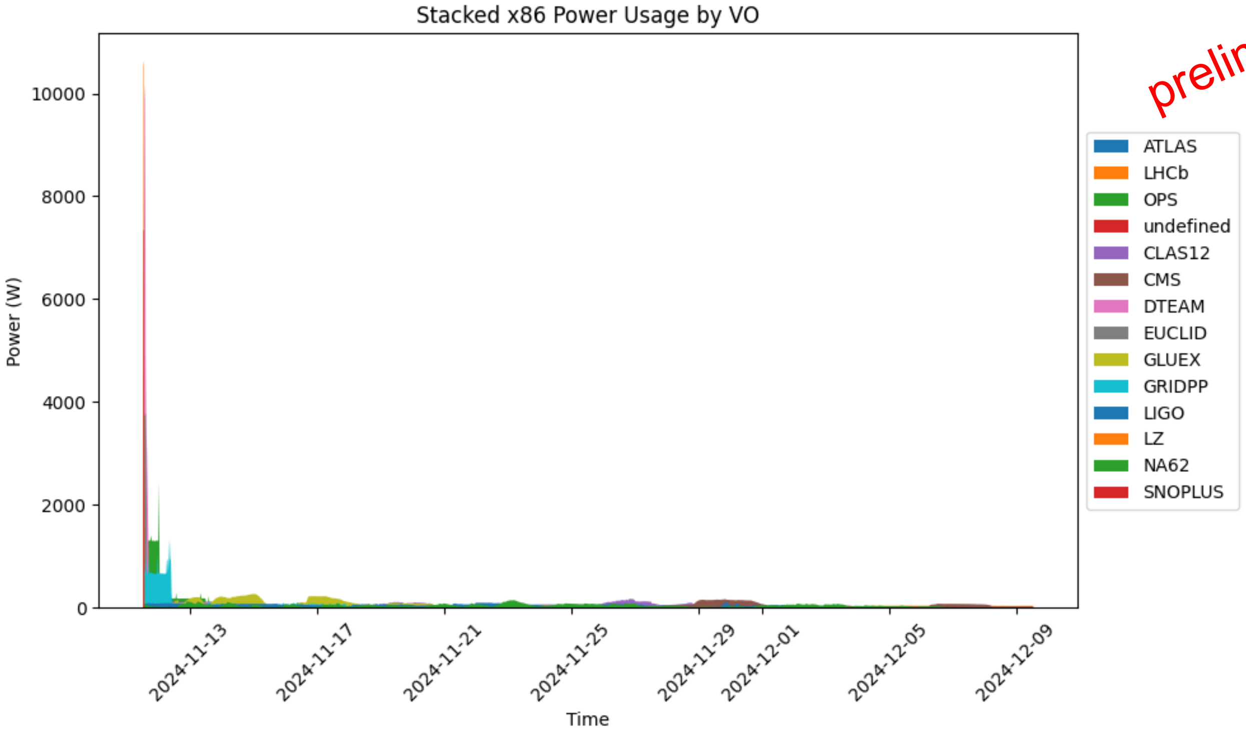




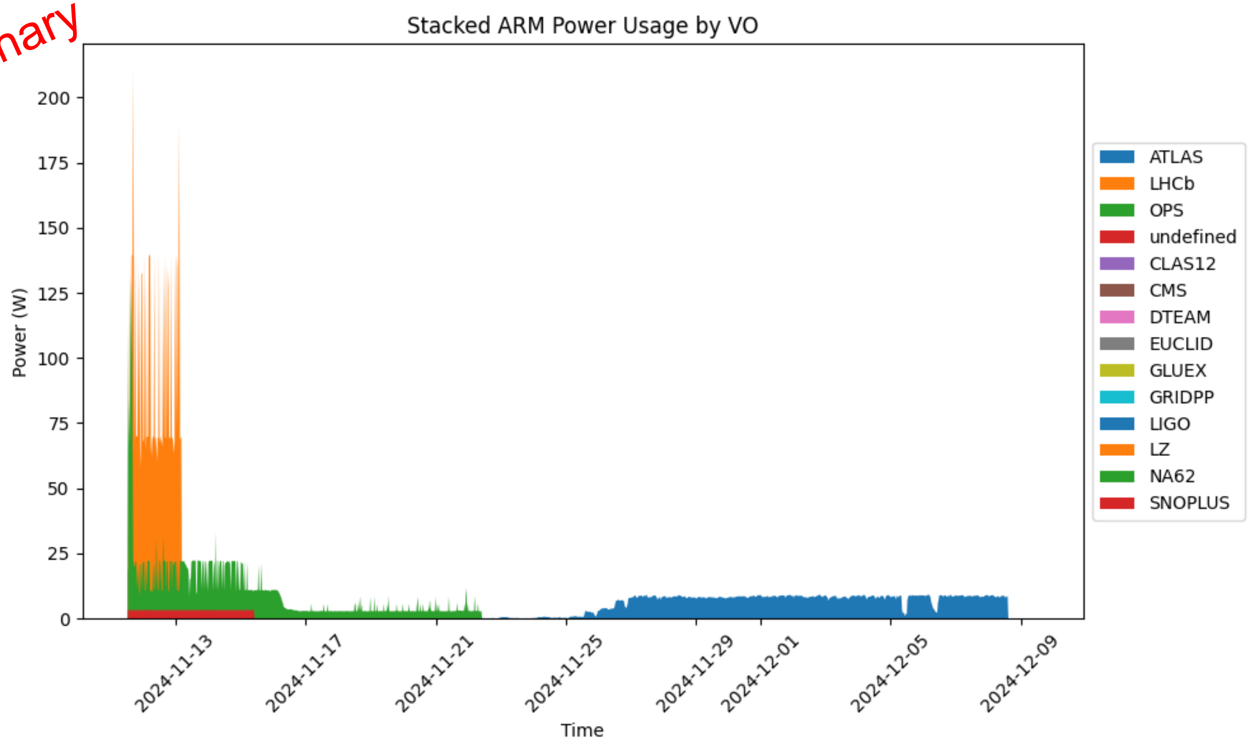
Proof of Concept (2)

With these numbers, we can then scale the power usage and assign it proportionally to VOs.

The 2 plots below show the proportional power usage by VO over 1 month period:



preliminary



And these are integrated power usages (kWh) for a few sampled VOs over the same period:

VO ATLAS:
ARM: 3760.25 kWh
x86: 26086.83 kWh
VO LHCb:
ARM: 3720.07 kWh
x86: 25797.80 kWh

VO OPS:
ARM: 2721.78 kWh
x86: 25950.92 kWh
VO CLAS12:
ARM: 0.00 kWh
x86: 25628.82 kWh

VO CMS:
ARM: 0.00 kWh
x86: 24582.94 kWh
VO DTEAM:
ARM: 0.00 kWh
x86: 25812.16 kWh

VO EUCLID:
ARM: 0.00 kWh
x86: 10602.08 kWh
VO GLUEX:
ARM: 0.00 kWh
x86: 26231.67 kWh

preliminary

Issue and possible solutions

The current strategy just calculates the proportional Power Distribution: use the % resource utilization by VO to allocate total power usage. For example: VO “ScotGrid” uses 30% of the cluster resources, so ScotGrid gets billed for 30% of the total power (active + idle). This is an executive choice, but I am ready to hear your suggestions and criticism … for instance:

1. **Idle Power Allocation**

Option A: Divide idle power among all VOs (including inactive ones)
- Pros: Simple and reflects the cost of having infrastructure ready for all VOs.
- Cons: Inactive VOs may complain about being "billed" for resources they didn’t use.

Option B: Divide idle power among active VOs only
- Pros: Aligns cost with actual resource usage.
- Cons: Few active VOs at any time may bear a disproportionate share of the idle power cost, which could seem unfair.

Suggestion: Option A is better for fairness as all VOs benefit from having the cluster online.
Rationale: if you rent a car and keep it parked in the drive-way, you may save on gas but you still pay for the rent!

2. **Accounting for Machine Types**

Option A: Track job distribution per machine type
- Pros: Accurately reflects the true energy cost per VO.
- Cons: Adds complexity; might lead to disputes if a VO is often assigned to less efficient machines.

Option B: Average out hardware efficiency
- Pros: Simple and less controversial. Treat the cluster as a "black box" delivering aggregated services.
- Cons: Loses granularity in reporting efficiency.

Suggestion: Option B for simplicity. But we consider expanding it to include machine-specific efficiencies and PUE (“bill” the site, which should push site level optimization?).

Conclusions & Outlook

- Improving on the methodology and developing an automated Analysis framework:
  - energy measurement now integrated in **HEP-Score**
  - **HS23/Watt** metric soon to be included in the **HEP-Score DB** (one stop shop for hardware rating!)
  - (maybe) add the frequency dependence in the **DB** for better characterization of hardware?
- Implement a prototype for WLCG global CO_2 accounting - possibly, not in isolation
  - work harder on developing a robust Power Accounting strategy!
  - iron out the details in collaboration with **VOs** & **WLCG sites** (idle, PUE, hardware efficiency)
  - talk to experts on Power grids to attach the CO_2 cost to power
- Keep looking at more energy efficient hardware solutions (x86, ARM, RISC-V, GPUs)

end

Emanuele Simili, Gordon Stewart, Samuel Skipsey Albert Borbely & David Britton (**ScotGrid Glasgow**)

in-House (production)

2xIntel40ht: Dual Socket Intel XEON E5-2630 v4 (HP) ~ 1.5k cores

CPU: 2 * x86 Intel(R) Xeon(R) E5-2630 v4, 10C/20HT @ 2.2GHz (TDP 85W)
RAM 160GB (4 x 32GB + 4 x 8GB) DDR4 2400 MHz → 4 GB/core
HDD: 2TB disk SATA @ 7200 RPM

2xAMD64ht: Dual Socket AMD EPYC 7513 (DELL) ~ 5k cores

CPU: 2 * x86 AMD EPYC 7513 (Milano), 32C/64HT @ 2.6GHz (TDP 200W)
RAM: 512GB (16 x 32GB) DDR4 3200MT/s → 4 GB/core
HDD: 3.84TB SSD SATA Read Intensive

2xAMD64ht: Dual Socket AMD EPYC 7452 (DELL) ~ 7.5k cores

CPU: 2 * x86 AMD EPYC 7452 (Roma), 32C/64HT @ 2.35GHz (TDP 200W)
RAM: 512GB (16 x 32GB) DDR4 3200MT/s → 4 GB/core
HDD: 3.84TB SSD SATA Read Intensive

2*ARM80c: Dual Socket Ampere Altra Q80-30 (Ampere) ~ 2k cores

CPU: 2 * ARM Ampere Q80-30, 80C @ 3GHz (TDP 210W)
RAM: 512GB (32 x 16GB or 16 x 32GB) DDR4 3200MT/s → 3.2 GB/cor
HDD: 2 * 1TB NVMe

ARM128c: Single Socket Ampere Altra Max M128-30 (SuperMicro) ~ 2k cores

CPU: ARM Ampere M128-30, 128C @ 3GHz (TDP 250W)
RAM: 512GB (8 x 64 GB) DDR4 3200MHz → 4 GB/core
HDD: 8TB NVMe

in-House (testing)

AMD96ht: Single AMD EPYC 7003 (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

2xAMD48ht+GPU: Dual Socket AMD EPYC 7443 (DELL)

CPU: 2* AMD EPYC 7443, 24C/48HT @ 2.3GHz (TDP 200W)
GPU: 2* NVIDIA A100 PCIe 80GB (TDP 300W)
RAM: 256GB (16 x 16GB) DDR4 3200MHz → 2.7 GB/core
HDD: 480GB SSD SATA + 5TB SSD SCSI

ARM80c: Single socket Ampere Altra Q80-30 (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

Grace144c: Dual Socket* NVidia Grace (SuperMicro)

CPU: NVidia Grace 144-Core 480GB DDR5 @ 3.4GHz (TDP 500W)
RAM: 480GB (on chip) DDR5 4237MHz → 3.3 GB/core
HDD: 1TB NVMe + 4TB NVMe

And, we also have a **RISC-V** test box …

Remote Testing

2*AMD256ht: Dual Socket AMD EPYC 9754 (SuperMicro)
CPU: 2 * x86 AMD EPYC 9754 (Bergamo), 128C/256HT @ 3.1GHz (TDP 360W)
RAM: 1.536TB (24 x 64GB) DDR4 3200MHz → 3 GB/core
HDD: 512GB NVMe + 3.84TB SSD

Super Micro

AMD128ht: Single Socket AMD EPYC 8534P (SuperMicro)
CPU: AMD EPYC 8534P (Siena), 64C/128HT @ 3.1GHz (TDP 200W)
RAM: 576GB (6 x 96GB) DDR5 3200MT/s → 4.5 GB/core
HDD: 1TB NVMe Storage

2xAMD192ht: Dual Socket AMD EPYC 9654 96-Core (…)
CPU: AMD EPYC 9654 (Genoa), 96C/184HT @ 3.7GHz (TDP 340W)
RAM: 1TB (…) → 5GB/core
HDD: …

@ RAL

ARM128c: Single Socket Ampere Altra Max M128-28 (XMA)
CPU: ARM Ampere M128-28, 128C @ 2.8GHz (TDP 250W)
RAM: 512GB (8 x 64GB) DDR4 3200MHz → 4 GB/core
HDD: 1TB NVMe Storage

XMA

ARM192c: Single Socket AmpereOne A192-32x (SuperMicro)
CPU: ARM AmpereOne A192-32x, 192C @ 3.2GHz (TDP 350W)
RAM: 1Tb (…) → 5 GB/core
HDD: 1TB NVMe Storage

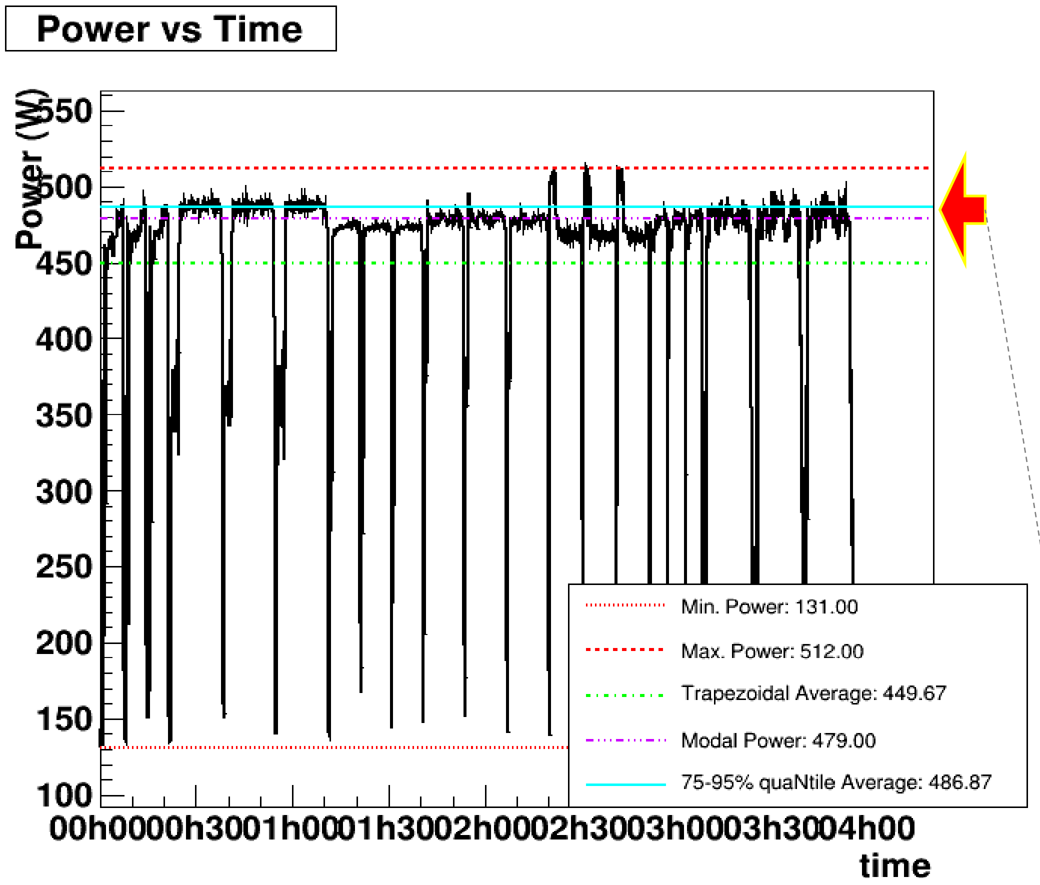
Super Micro

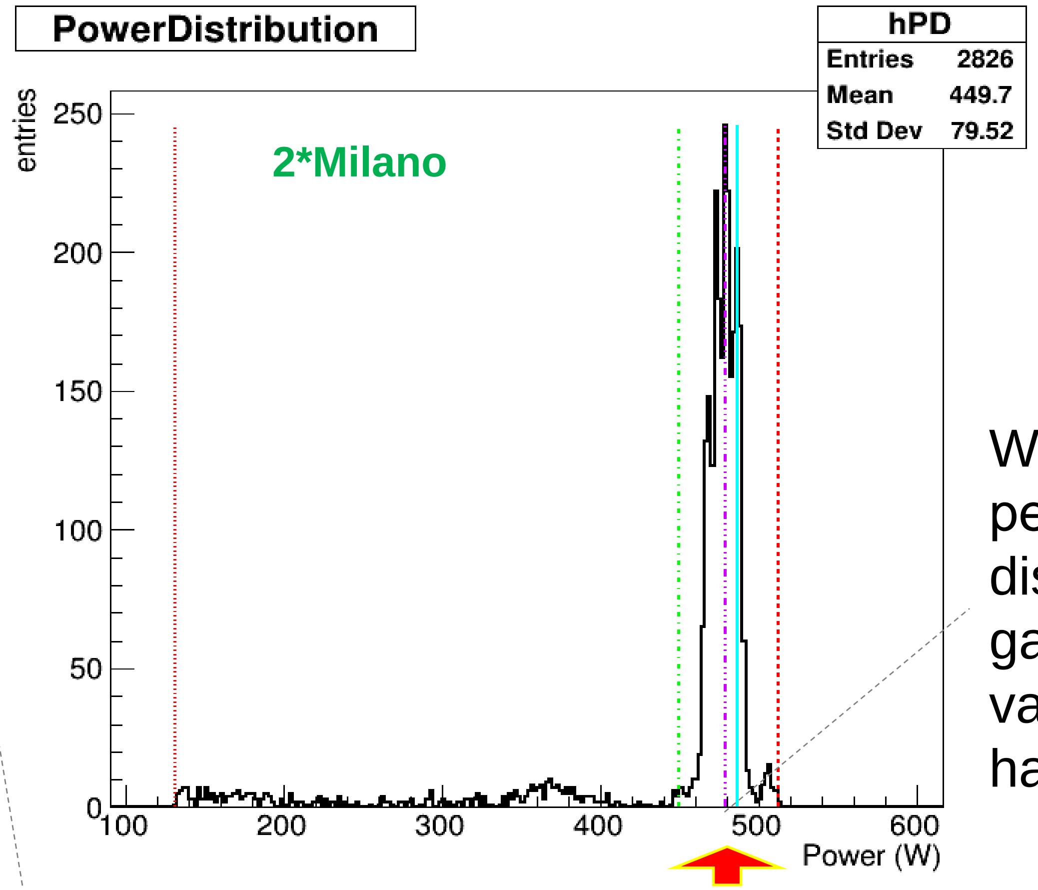
We have expressed our interest in testing new hardware to a few vendors, and from time to time we get remote access to new machines. We have also gathered data from other WLCG sites (**RAL**).

But, our machine sample is negligible compared to what is available in the **HEP-Score DB** …

What Watt

We wish to extract an accurate **Figure of Merit** (**FoM**) of power usage for a standard HEP workload from smaller **HEP-Score** containerized jobs, which is easy to implement and consistent across hardware.





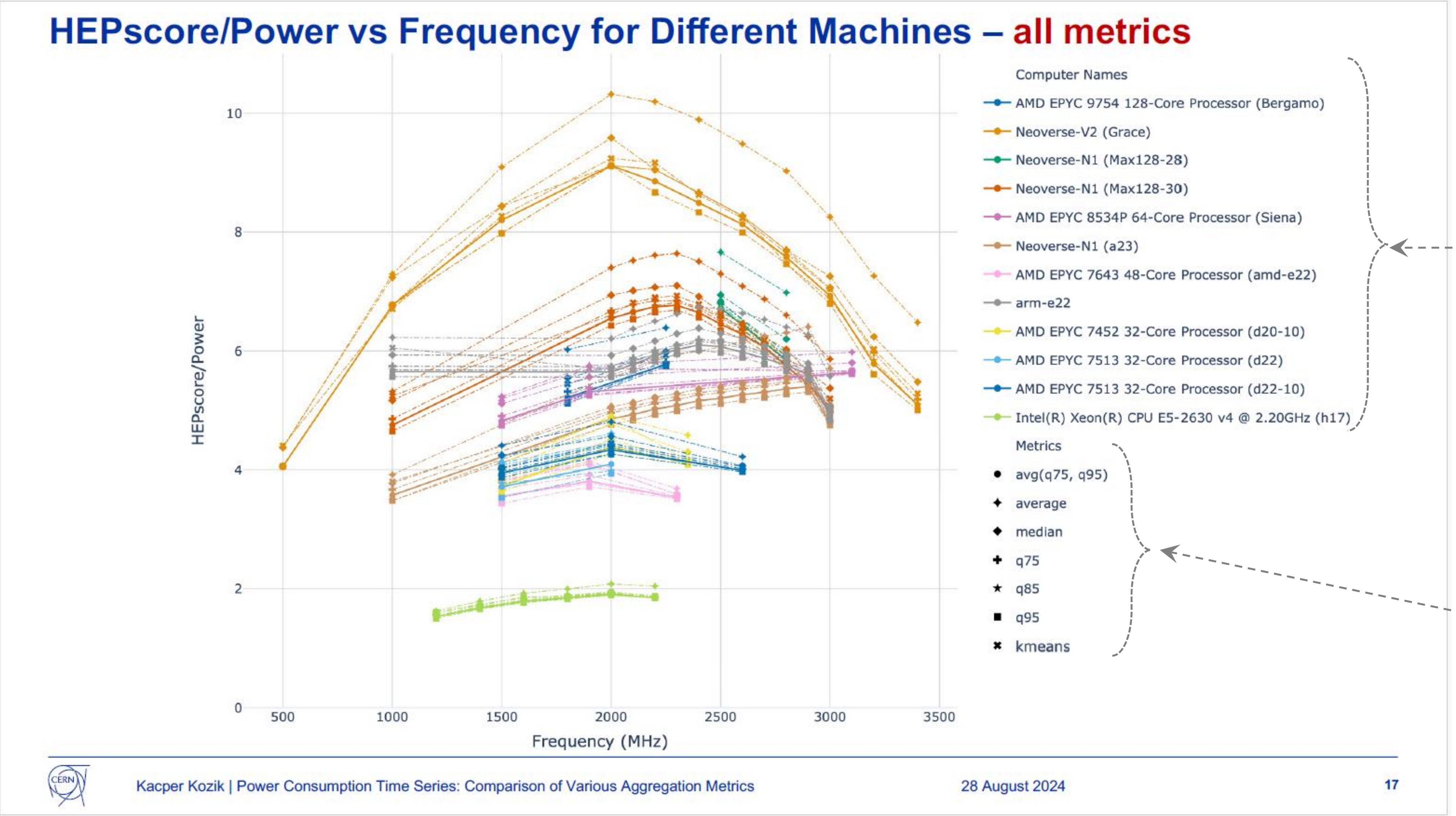
We could fit this peak, but … the distribution is not gaussian and varies across hardware.

<75-95%> Blue line sits nicely in the plateau !

By arranging the data in power order we can perform an upper quartile average, but discard the top 5% of data to remove isolated peaks. This we call 75-95% quantile average.

What Watt (reprise)

The **HEPiX Benchmark Working Group** has also studied various statistical proxies for power usage.
In particular, see the presentation by Kacper Kamil Kozik: https://indico.cern.ch/event/1433496/



The machines are the same from the previous slide, but labels are slightly different.

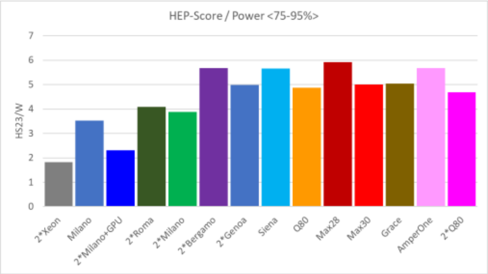
The "average power" is estimated by using different statistical proxies (metrics), see legend.

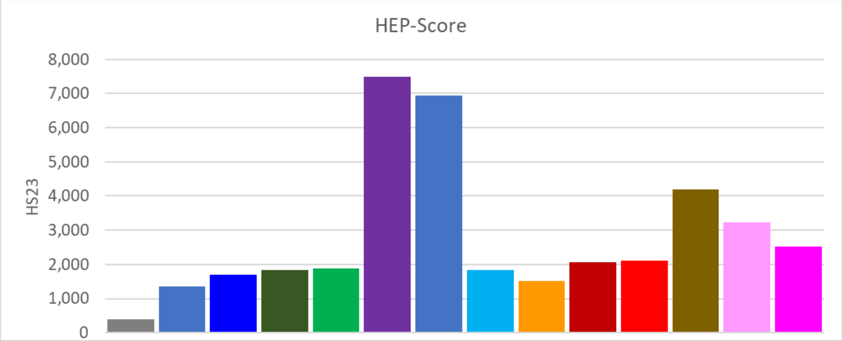
more HS23/Watt

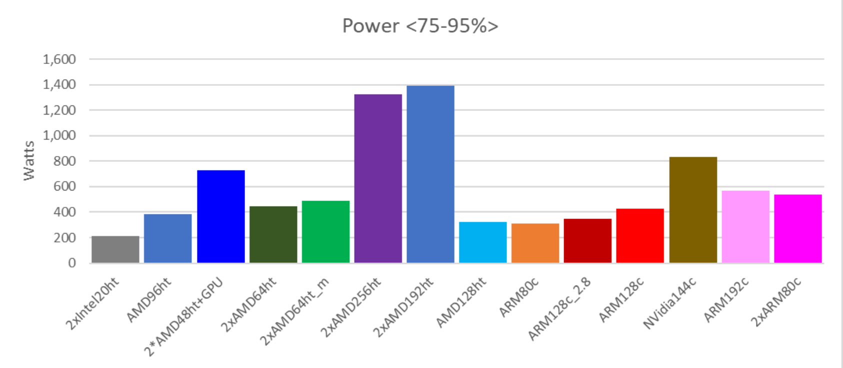
Using our **FoM**, we measure performance per watt as: **HS23 / Power<75-95%>**

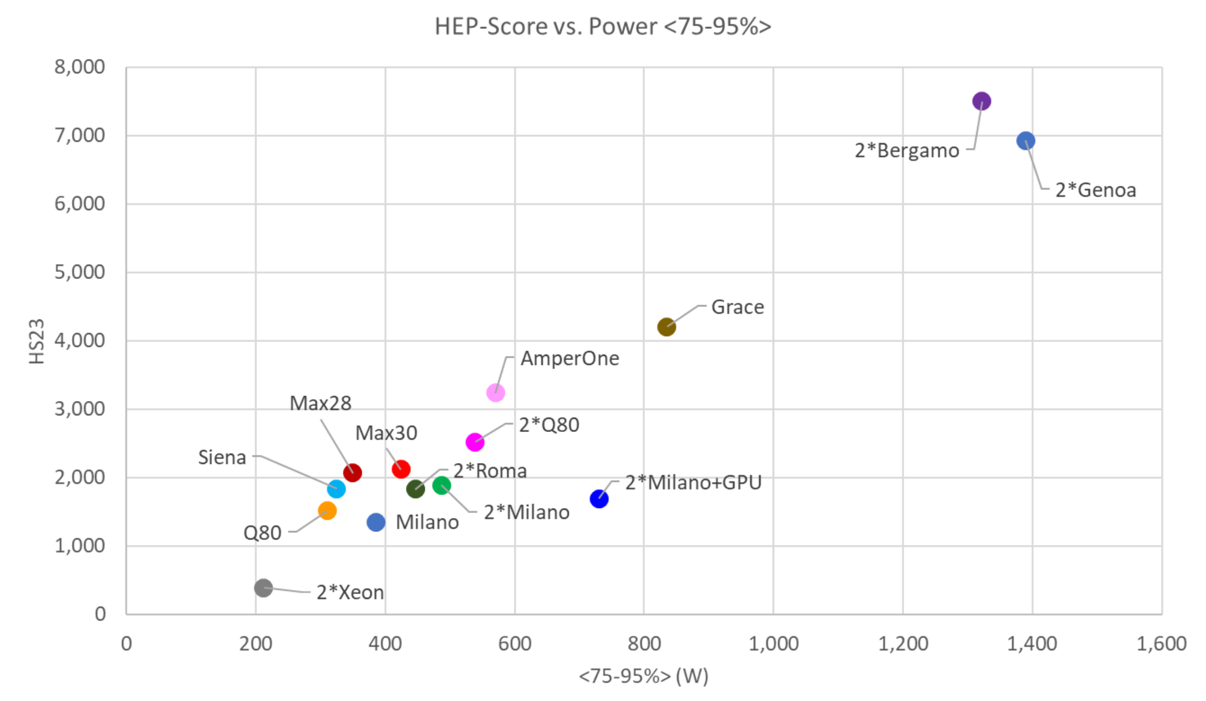
GPU not used

Nickname	Machine	CPU	Arch	HT	Threads	Governor	Max Freq.	HS23/W	<75-95%>	Watt/core
*2*Xeon*	2xIntel20ht	2 * Intel XEON 10-Core CPU E5-2630 v4	2*x86_64	on	40	conservative	2.2	1.822	211.7	5.3
Milano	AMD96ht	AMD EPYC 7643 48-Core Processor	x86_64	on	96	conservative	2.3	3.525	384.7	4.0
*2*Milano+GPU*	2*AMD48ht_gpu	2 * AMD EPYC 7443 24-Core Processor + ~~2* NVIDIA A100 PCIe 80GB~~	2*x86_64	on	96	conservative	4.0	2.318	730.1	7.6
*2*Roma*	2xAMD64ht	2 * AMD EPYC 7452 32-Core Processor	2*x86_64	on	128	conservative	3.3	4.092	446.3	3.5
*2*Milano*	2xAMD64ht_m	2 * AMD EPYC 7513 32-Core Processor	2*x86_64	on	128	conservative	2.6	3.876	486.9	3.8
*2*Bergamo*	2xAMD256ht	2 * AMD EPYC 9754 128-Core Processor	2*x86_64	on	512	conservative	3.1	5.670	1,322.2	2.6
*2*Genoa*	2xAMD192ht_cor	2 * AMD EPYC 9654 96-Core Processor	2*x86_64	on	384	conservative	3.7	4.991	1389	3.6
Siena	AMD128ht	AMD EPYC 8534P 64-Core Processor	x86_64	on	128	conservative	3.1	5.659	324.9	2.5
Q80	ARM80c	Ampere Altra Q80-30	aarch64	//	80	conservative	3.0	4.881	309.9	3.9
Max28	ARM128c_2.8	Ampere Altra Max M128-28	aarch64	//	128	conservative	2.8	5.911	349.9	2.7
Max30	ARM128c	Ampere Altra Max M128-30	aarch64	//	128	conservative	3.0	5.002	424.1	3.3
Grace	NVidia144c	NVidia Grace 144-Core 480GB DDR5	2*aarch64	//	144	conservative	3.4	5.035	835.2	5.8
AmperOne	ARM192c	AmperOne 192-Core	aarch64	//	192	conservative	3.2	5.673	570.1	3.0
*2*Q80*	2xARM80c	2 * Ampere Altra Q80-30	2*aarch64	//	160	conservative	3.0	4.681	538.5	3.4

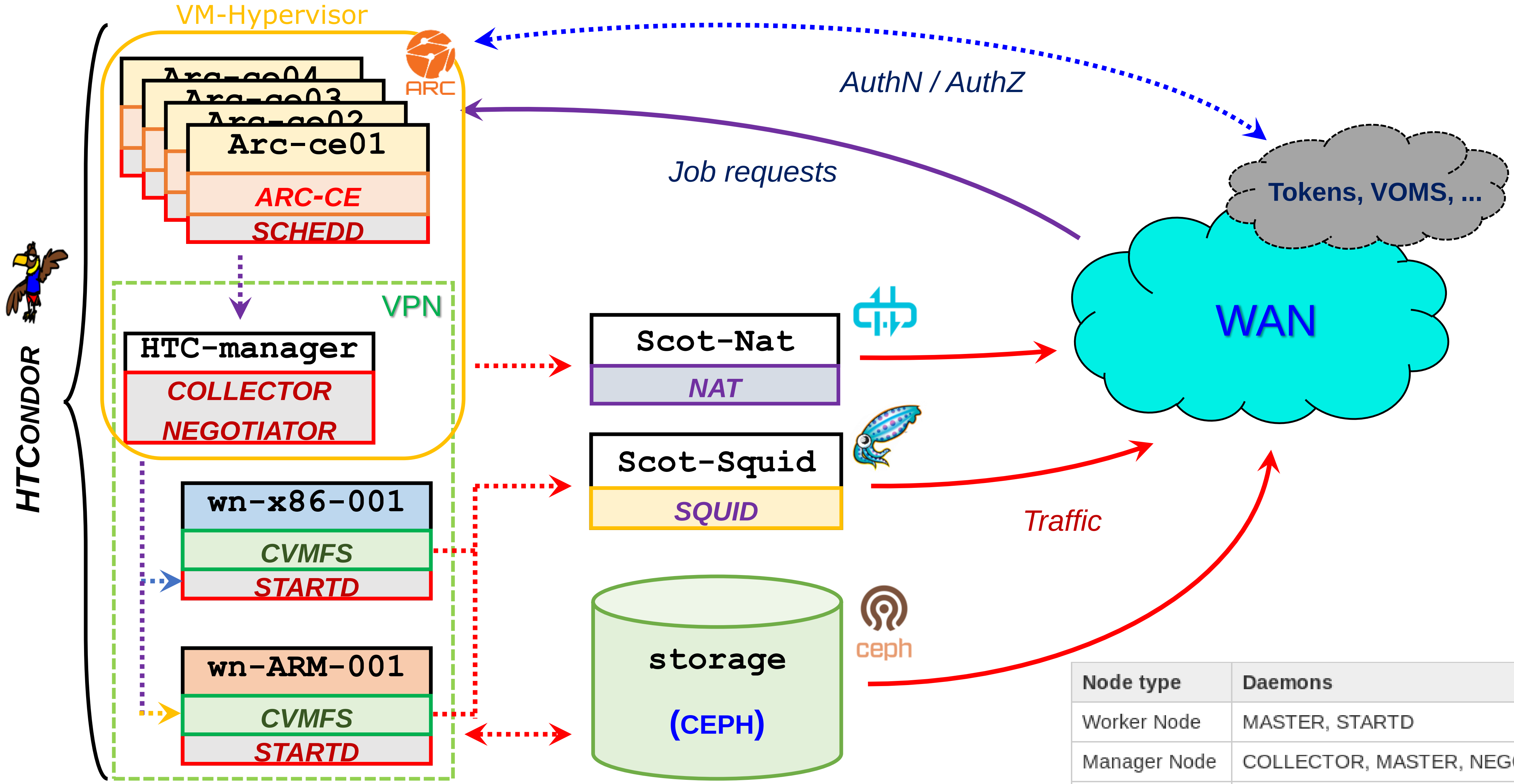








ScotGrid Tier2 Cluster Overview



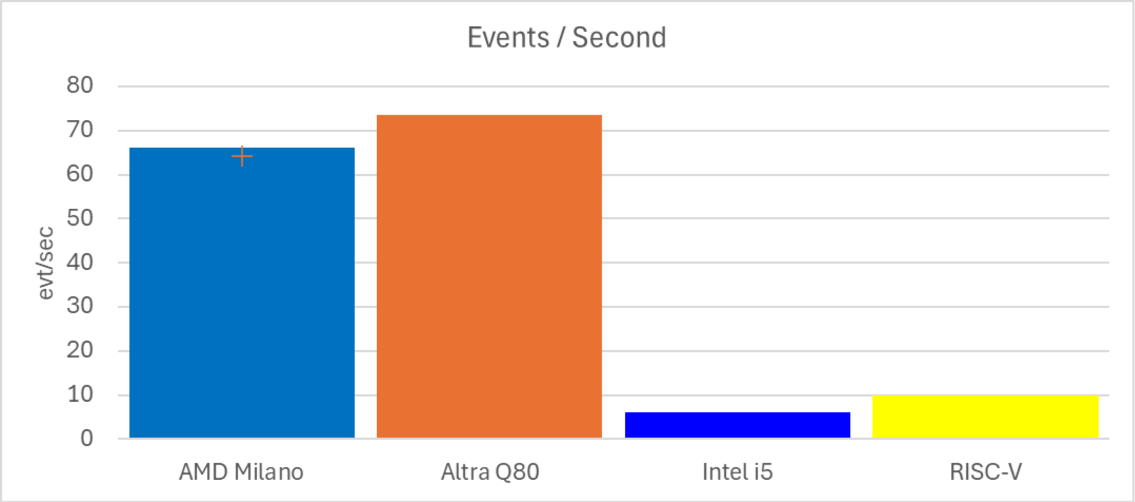
Node type	Daemons
Worker Node	MASTER, STARTD
Manager Node	COLLECTOR, MASTER, NEGOTIATOR, SCHEDD
CE Node	MASTER, SCHEDD

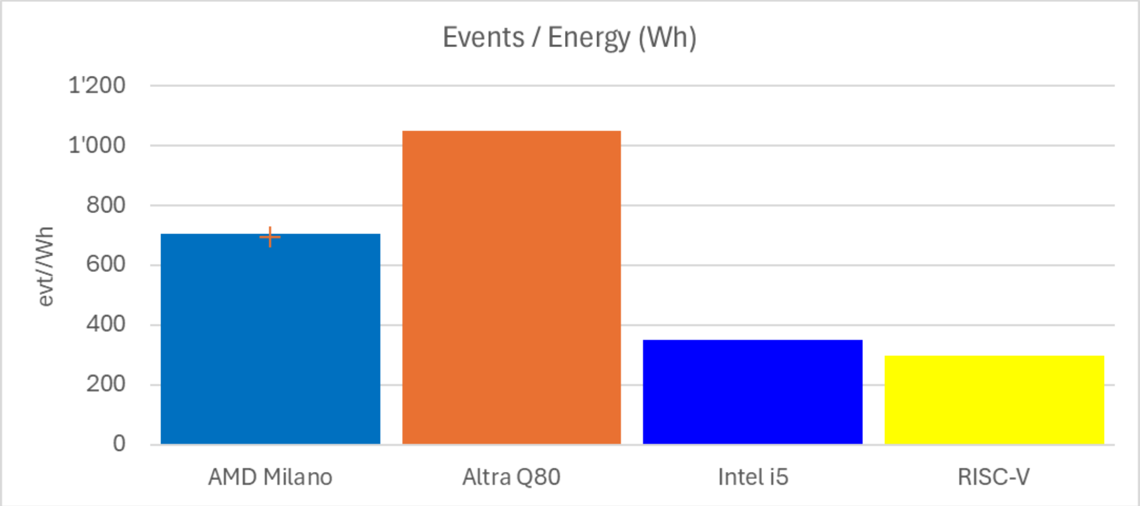
Preliminary look at RISC-V

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 kernel in 2022, the 64-bit variant (riscv64) in 2023.

We recently acquired a fully integrated RISC-V desktop PC running Fedora 38 (for ~2.5k £) …

Milk-V Pioneer





We ran a **Geant4** full detector simulation as benchmark, and compared the RISC-V performance with 3 types of hardware (desktop i5 PC, AMD EPYC and ARM Ampere Q80 servers).

We evaluated the performance as **Events/Second** and **Events/Sec/Watt**. (≈ **Events/Energy**).

Our results show that, at present, the **RISC-V** PC performs slightly better than a 2017 desktop, but the hardware is not mature enough to compete with server grade **x86** and **ARM** CPUs (and there are currently no RISC-V server solutions).

With growing interest for the RISC-V architecture (e.g., **EPI** https://www.european-processor-initiative.eu/), the landscape can rapidly evolve, and we want to be prepared!