ARM compute @ Glasgow & Power Update

Outline

- Update on Glasgow cluster & new kit testing
  - heterogeneous computing!
  - doubling our **ARM farm** with new AltraMax

- Updates benchmarks results:
  - new **Figure of Merit** to estimate power usage (HEP-Score → standard job)
  - **HEPscore/Watt** comparison extended to more machines
  - new **Frequency Scan**
  - idle vs. frequency

- ARM **validation campaign**
  - High level summary of experiment tests
  - Validation results

- Outlook …

in-House (production)

2xIntel40ht: Dual Socket Intel XEON 10-Core CPU E5-2630 v4 (HP)

CPU: 2 * Intel(R) Xeon(R) 10-core CPU E5-2630 v4 @ 2.2GHz (TDP 85W)
RAM: 160GB (4*32GiB + 4*8GiB) DDR4 2400 MHz → 4 GB/core
HDD: 2TB disk SATA 6Gb/s @ 7200 RPM (SEAGATE)
OS: CentOS 7.9 → 🗑

2xAMD64ht: Dual Socket AMD EPYC 7513 32-Core Processor (DELL)

CPU: 2 * x86 AMD EPYC 7513, 32C/64HT @ 2.6GHz (TDP 200W)
RAM: 512GB (16 x 32GB) DDR4 3200MT/s → 4 GB/core
HDD: 3.84TB SSD SATA Read Intensive
OS: CentOS 7.9 → Alma 9

2xAMD64ht: Dual Socket AMD EPYC 7452 32-Core Processor (DELL)

CPU: 2 * x86 AMD EPYC 7452, 32C/64HT @ 2.32GHz (TDP 200W)
RAM: 512GB (16 x 32GB) DDR4 3200MT/s → 4 GB/core
HDD: 3.84TB SSD SATA Read Intensive
OS: CentOS 7.9 → Alma 9

2*ARM80c: Dual Socket Ampere Altra Q80-30 80-Core Processor (Ampere)

CPU: 2 * ARM Q80-30 80C @ 3GHz (TDP 210W)
RAM: 512GB (32 x 16GB or 16 x 32GB) DDR4 3200MT/s → 3.2 GB/core
HDD: 2 * 1Tb NVMe (INTEL + SAMSUNG)
OS: Rocky 9.2

~ 2k cores Single Socket Ampere Altra Max M128-30 coming soon …

in-House (testing)

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 (SAMSUNG)
OS: Alma 9

2xAMD96ht: Single AMD EPYC 7003 48-Core Processor (DELL)

CPU: 2* AMD EPYC 7443 24-Core Processor @ 4GHz @ 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 (DELL)
OS: Rocky 8

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

CPU: ARM Q80-30 80C @ 3GHz (TDP 210W)
RAM: 256GB (16 x 16GB) DDR4 3200MHz → 3.2 GB/core
HDD: 3.84TB SSD SATA (SAMSUNG)
OS: Alma 9

Grace144c: Dual Socket NVidia Grace 144-Core Processor (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 (SAMSUNG)
OS: Alma 9

+ a recently purchased RISC-V desktop PC (see next slides)

Tested Remotely

2*AMD256ht: Dual Socket AMD EPYC 9754 128-Core Processor (SuperMicro)

CPU: 2 * x86 AMD EPYC 9754, 128C/256HT @ 3.1GHz (TDP 360W)
RAM: 1.536TB (24 x 64GB) DDR4 3200MHz → 3 GB/core
HDD: 512GB NVMe + 3.84TB SSD
OS: Rocky 9.2

SuperMicro

AMD128c: Single Socket AMD EPYC 8534P 64-Core Processor (SuperMicro)

CPU: AMD EPYC 8534P @ 3.1GHz (TDP 200W)
RAM: 576GB (6 x 96GB) DDR5 3200MT/s → 4.5 GB/core
HDD: 1Tb NVMe Storage
OS: Rocky 8.8 Green Obsidian

SuperMicro

ARM128c: Single Socket Ampere Altra Max M128-28 128-Core Processor (XMA)

CPU: ARM M128-28 @ 2.8GHz (TDP 250W)
RAM: 512GB (64 x 8GB) DDR4 3200MHz → 4 GB/core
HDD: 1Tb NVMe Storage
OS: Rocky 8.8 Green Obsidian

XMA

More coming soon. We notified vendors about our interest.
Looking for: **AmpereOne**, Grace Hopper, Blackwell …

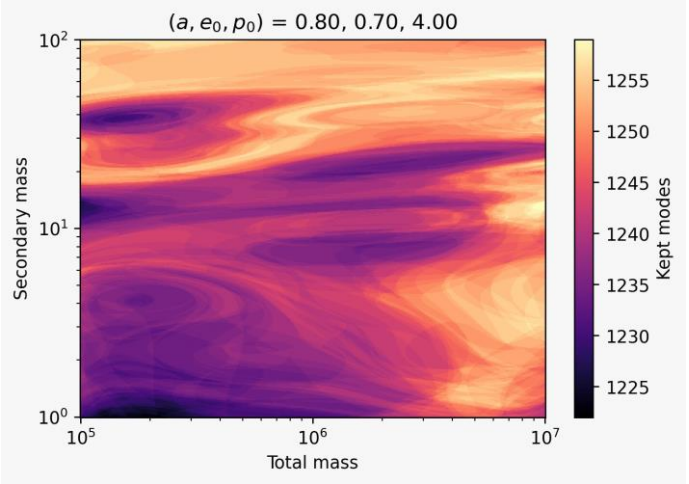
GPU testing

We had our GPU node for a while now, so far it has been used:

- by local users, as **A.I. playground**
  (from early **GPT**s to the latest **SDXL**)

- by **Gravitational Waves** groups (LIGO/LISA) for running a Neural-Network interpolator (EMRI SNR function)

https://academic.oup.com/mnras/article/522/4/6043/7159736



- by Bruno for trying out the **Celeritas** CPU+GPU framework

https://github.com/celeritas-project/celeritas

first goal: reproduce the ATLAS EM-Calorimeter simulation

RISC-V testing

Milk-V Pioneer : Single socket RISC-V 64-Core Processor (Milk-V)
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
OS: Fedora 38

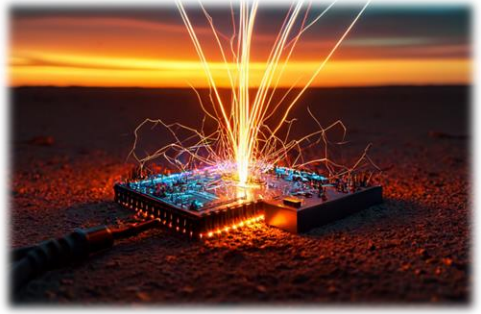
We managed to install **ROOT** for RISC (https://github.com/hahnjo/root.git), **CVMFS** and **XRootD** by compiling from source.

We could run **ROOT stress tests**: all 64 cores seem to do work and the peak power consumption is ~140W … but this is a desktop PC.

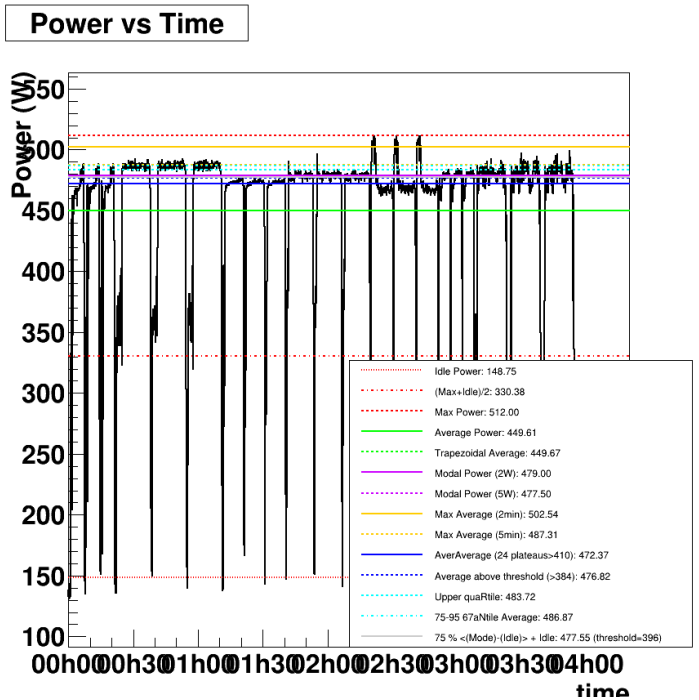
We have been talking with Tommaso (**INFN**), who also has a couple of RISC-V machines and had plan to try porting **CMS** …

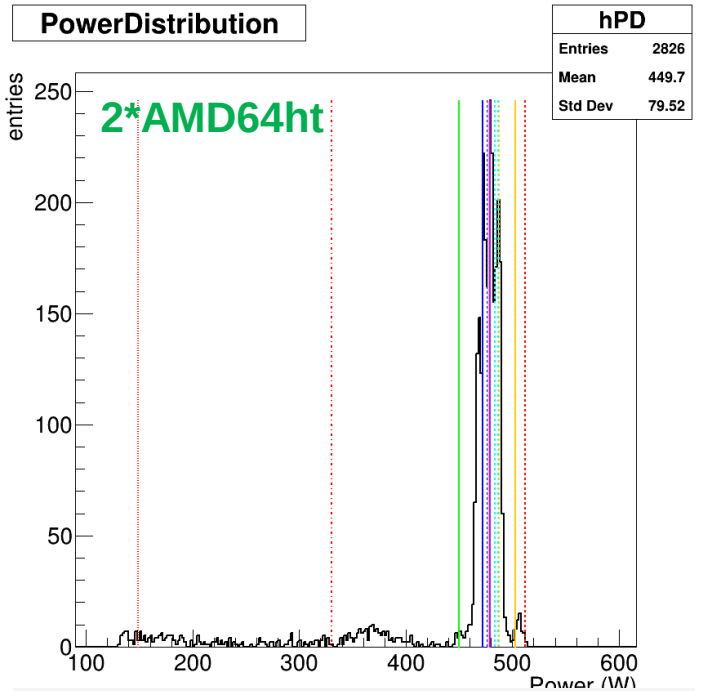
What Watt

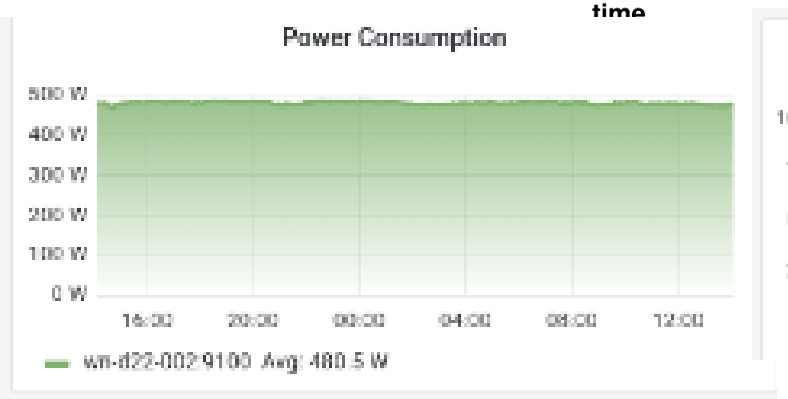
We benchmark using the **HEP-Score suite** (i.e., 3 x 7 workloads of *10-30 min.* each, including start-up phase).

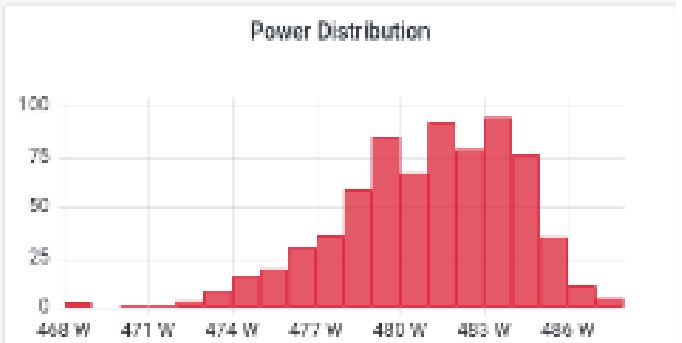


This does not well represent the standard running conditions at grid sites, where typical pilots last *24-48h* and multiple jobs run simultaneously, keeping the CPU at ~100% all time.









So … what is the most representative figure for of power usage?
Or, in other words: **What Watt** ?

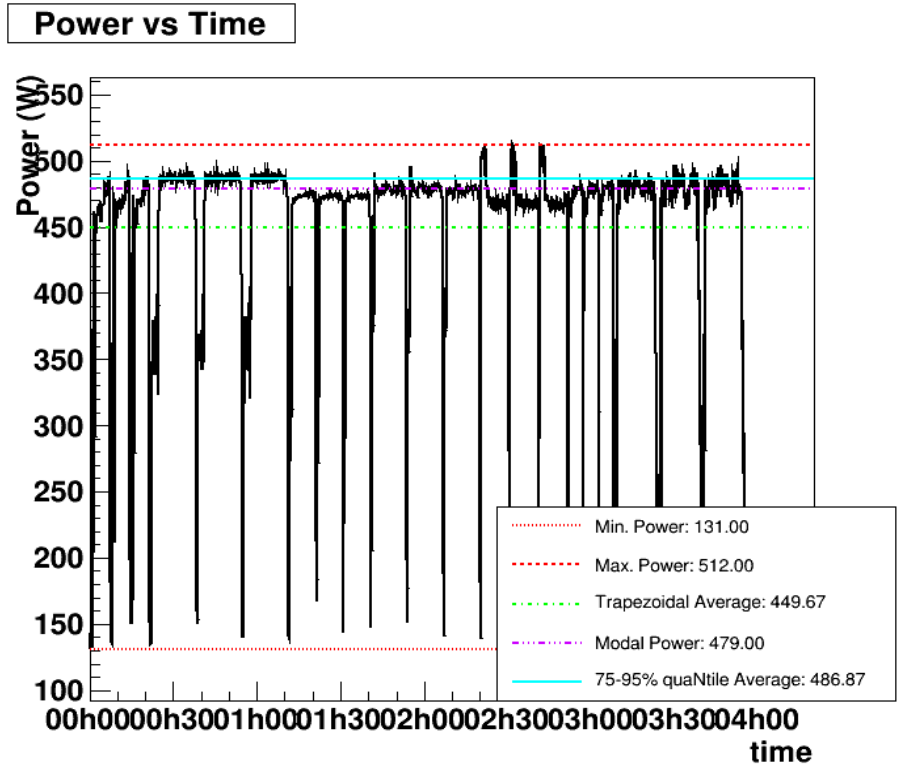
What Watt (2)

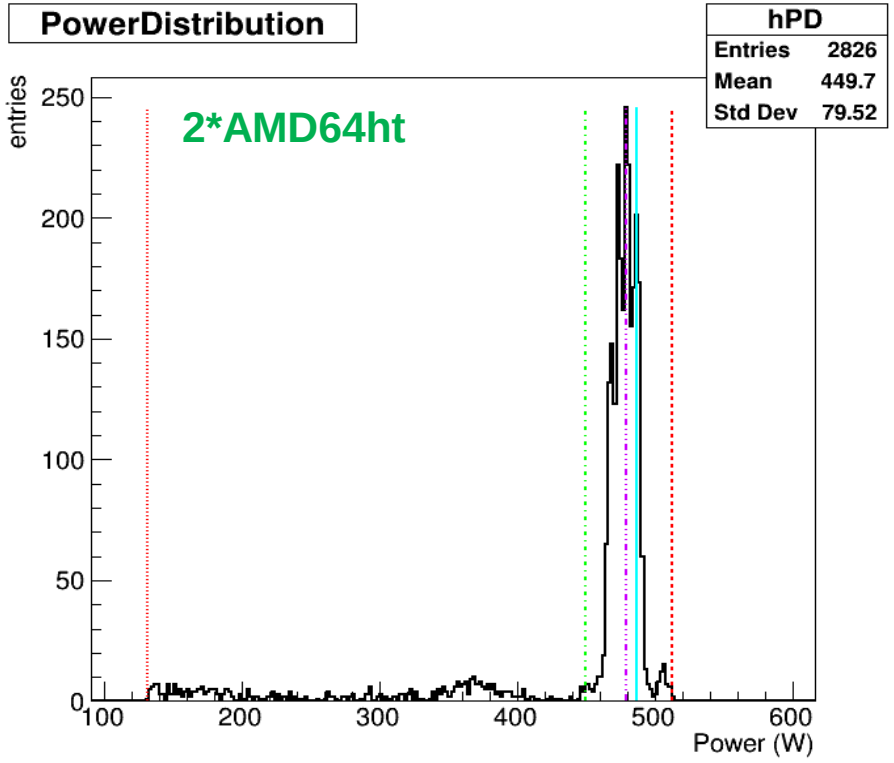
New **Figure of Merit** (**FoM**), i.e., the best proxy of power usage for standard HEP workloads:

FoM should be tuned to the power profile of **typical** runs:

⇒ Runs start with low power while staging and then run in a plateau for most of the run.

⇒ Don't want to be swayed by small spikes above the plateau or regions of time where power is low between runs.





Maximum (red dashed) – **Overestimates** usage

Average (green dashed) – **Underestimates** usage

Notice that power usage during HEP-Score run follows a peaked distribution.

FoM should reflect typical power usage, so we want to be inside the peak.

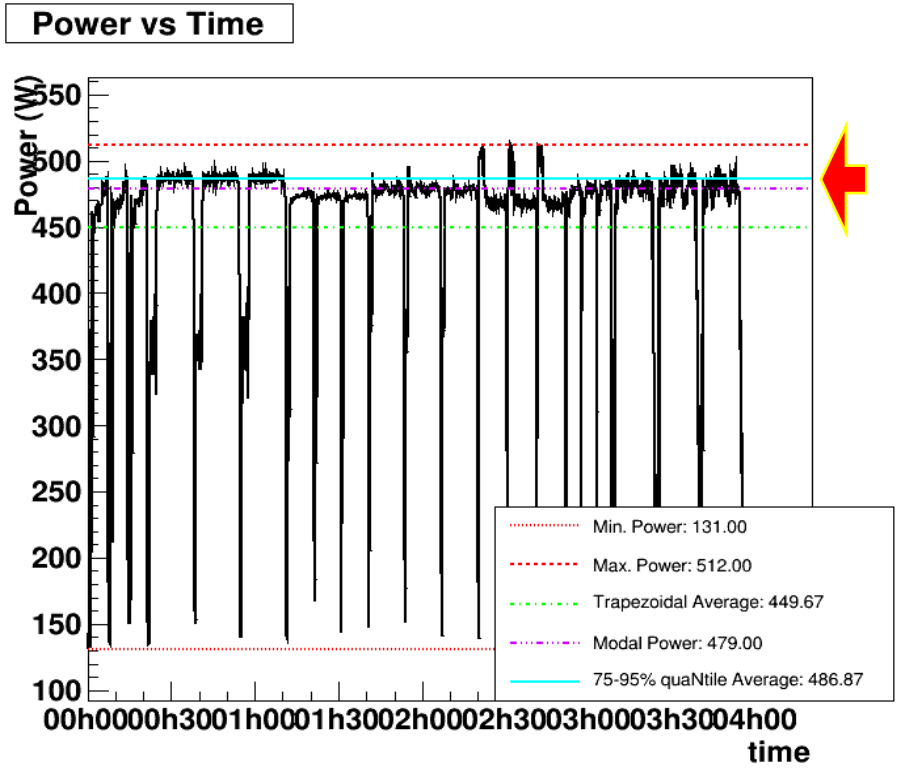
What Watt (3)

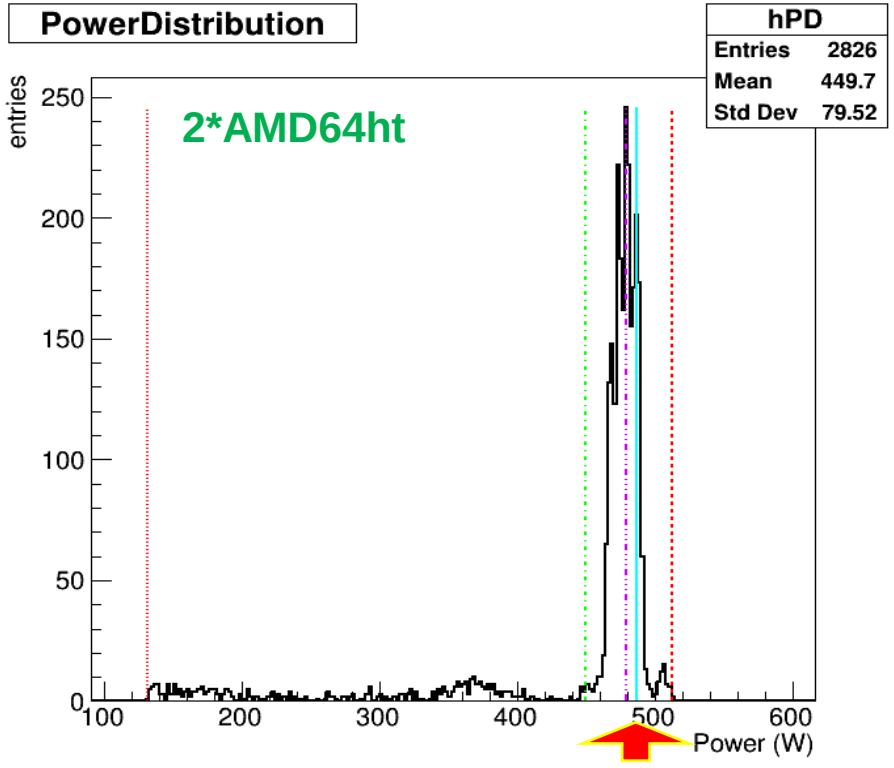
New **Figure of Merit** (**FoM**), i.e., the best proxy of power usage for standard HEP workloads:

FoM should be easy to implement, we could fit this peak, but there are other ways of doing it.

Arrange the data in power order and perform an upper quaRtile average, but removing the top 5% of data

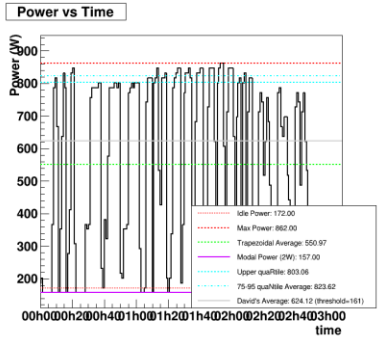
75-95% quaNtile average*

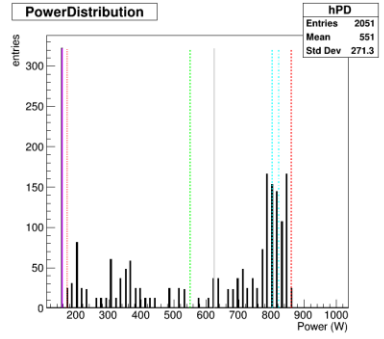


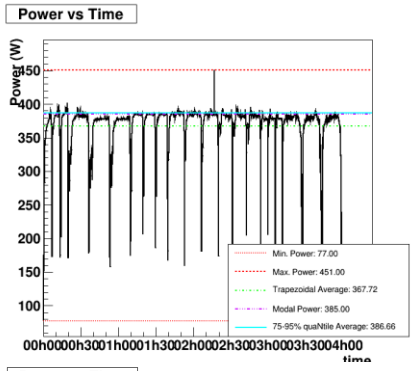


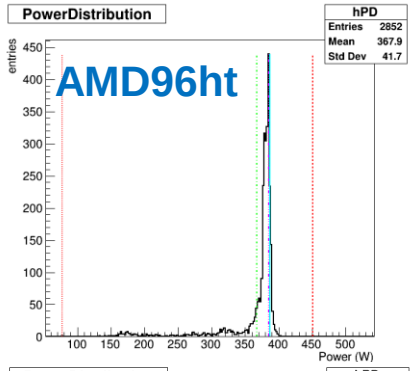
<75-95%> Blue line sits nicely in the plateau

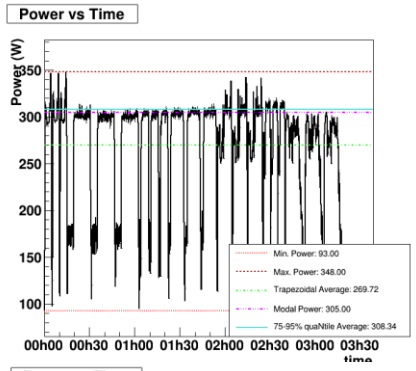
Modal power also sits in the plateau – but we found edge cases where this breaks (e.g., Grace has a very steady idle state)

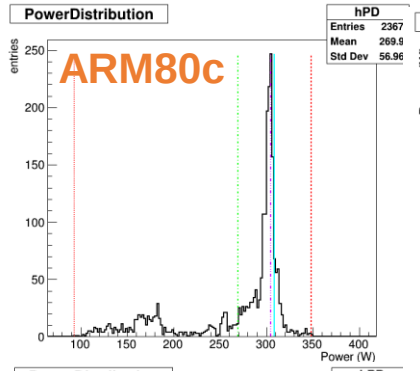


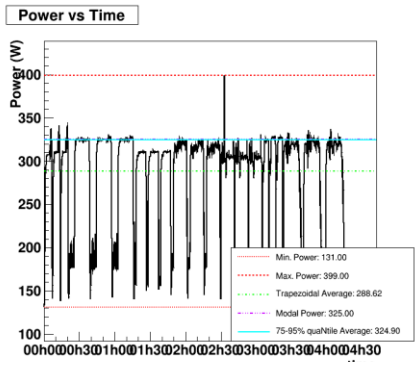


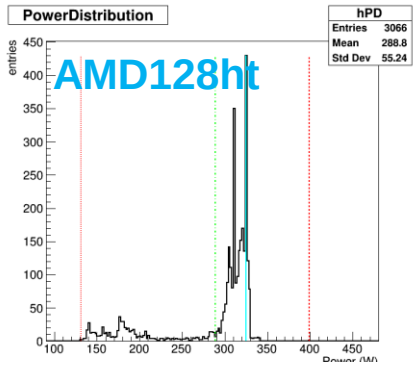


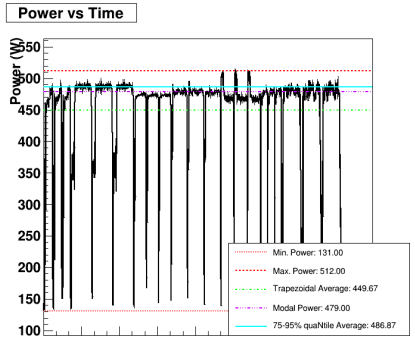


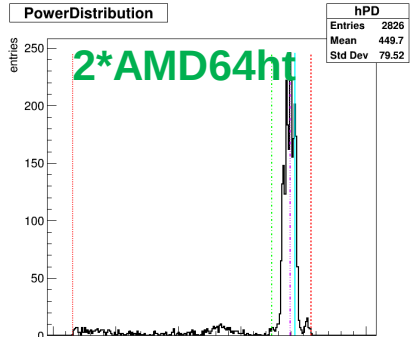


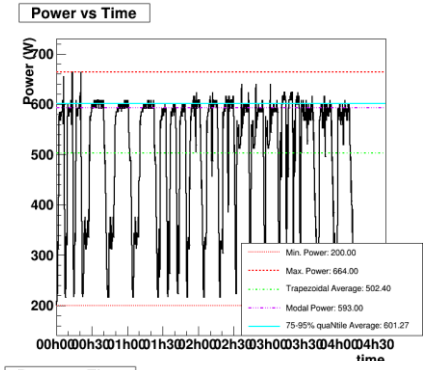


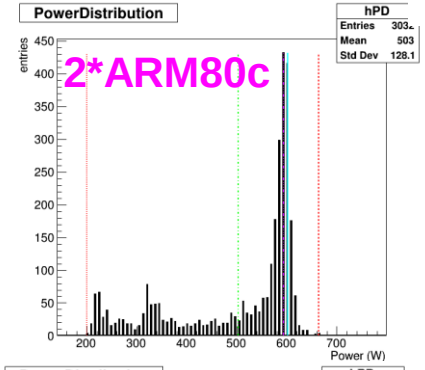


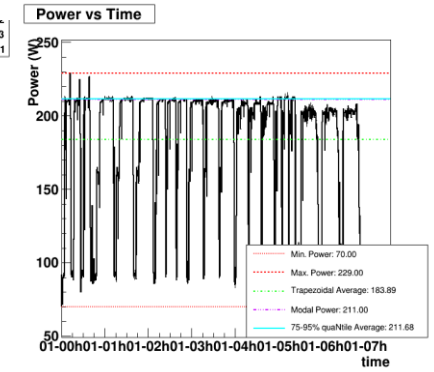


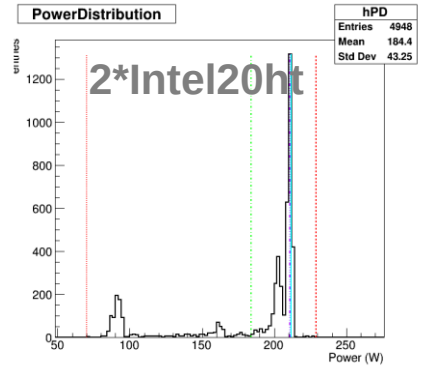


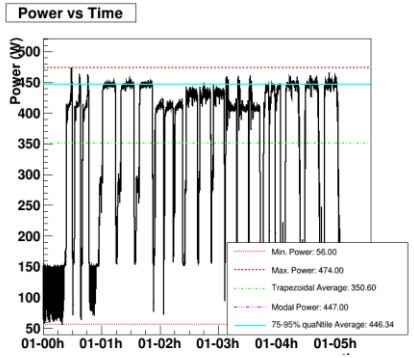


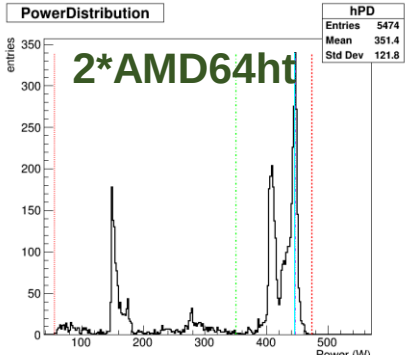


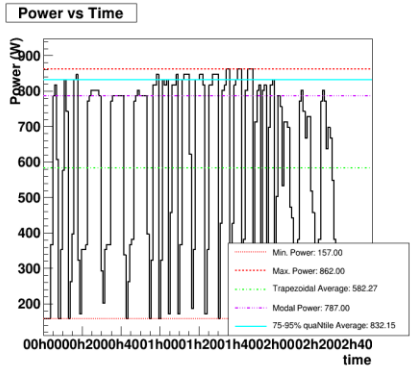


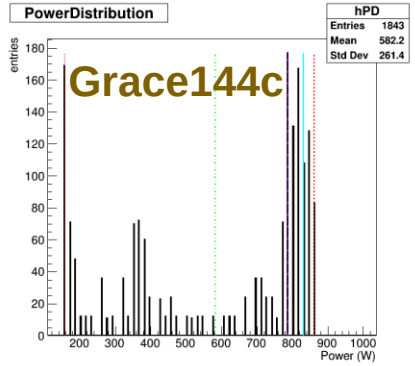


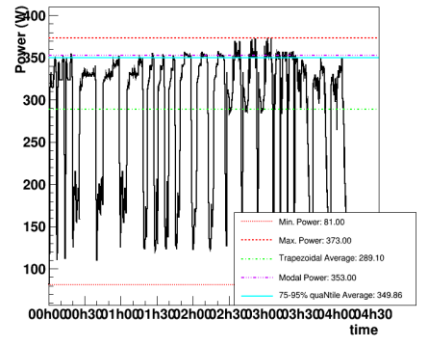


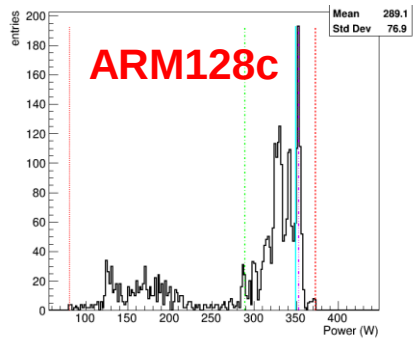


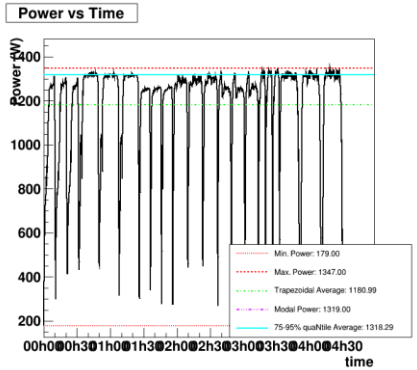


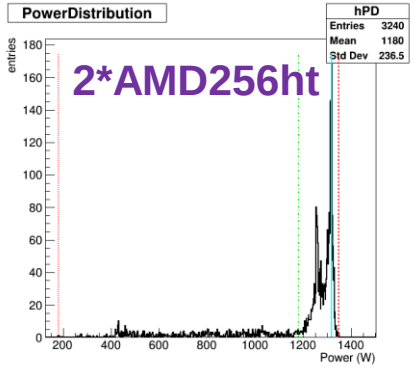








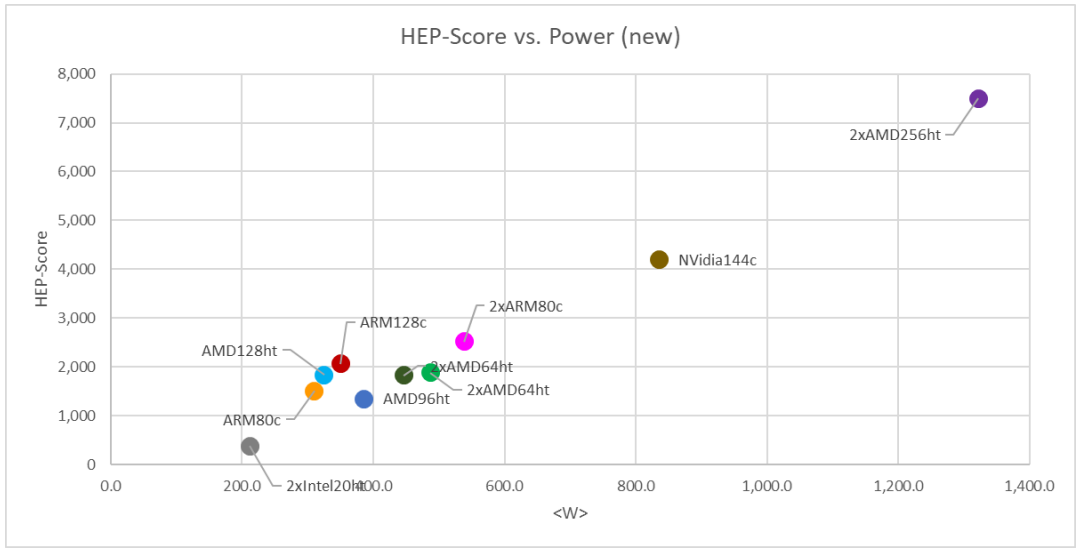


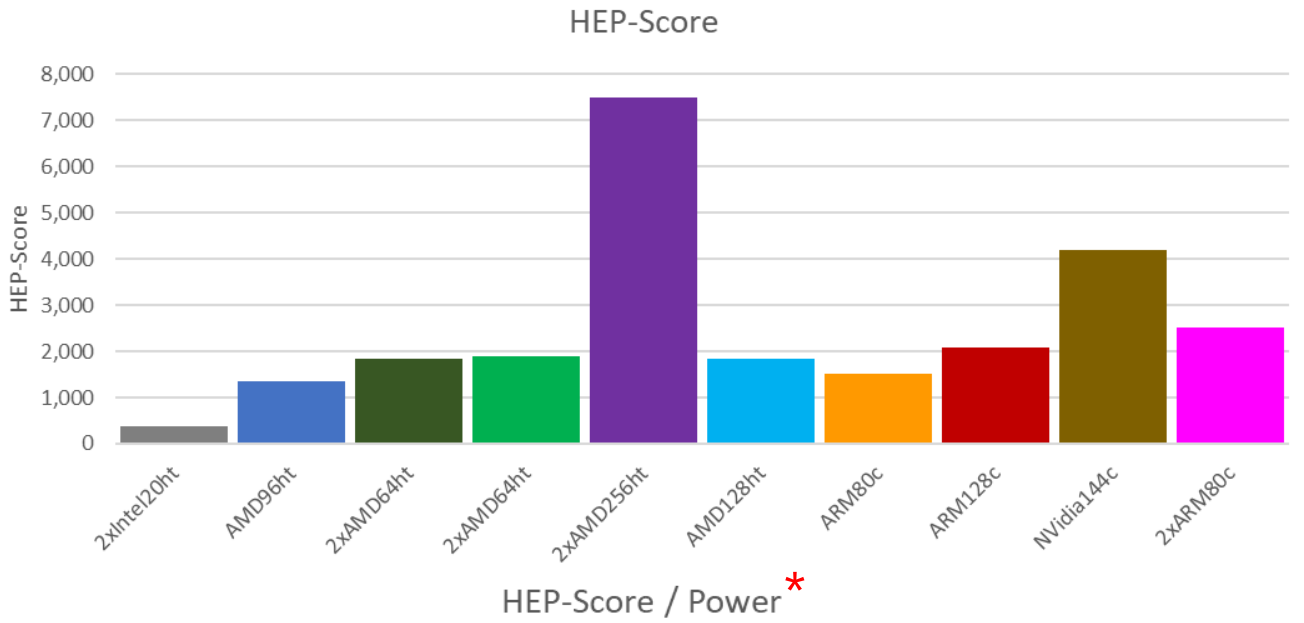


You can convince yourself that the new **FoM** well fits the power usage plateau on all tested machines …

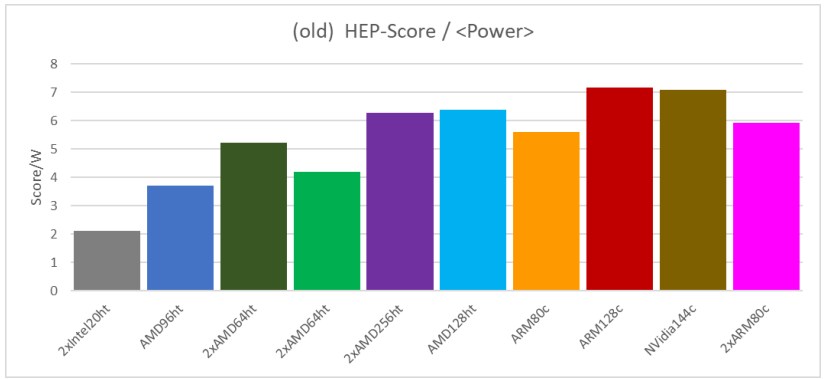
HEPscore/Watt

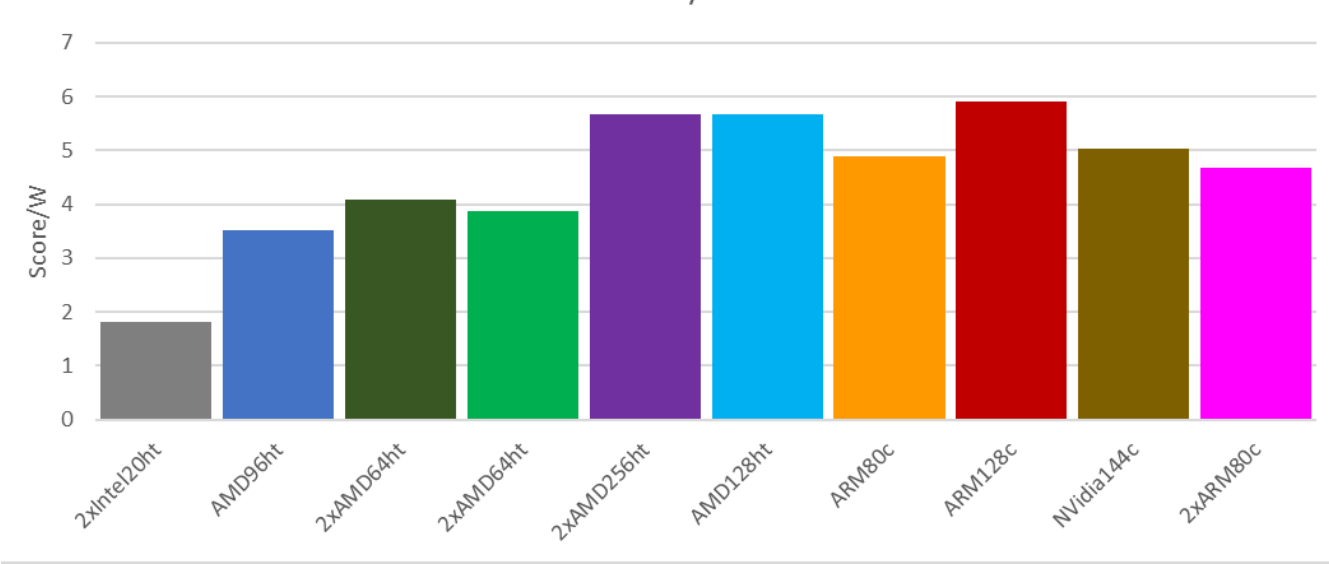
Using this new **Figure of Merit**, we have tested few more machines, and re-calculated the charts ...





The ranking does not change w.r.t. to the standard average, but distances do:

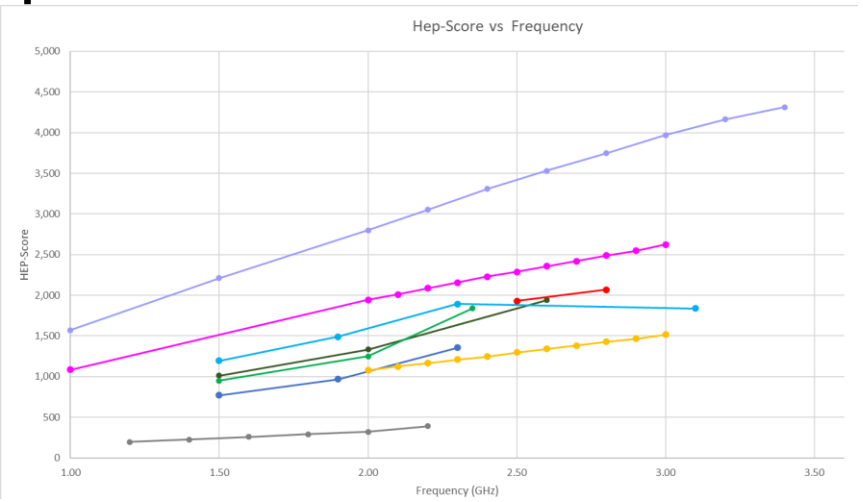


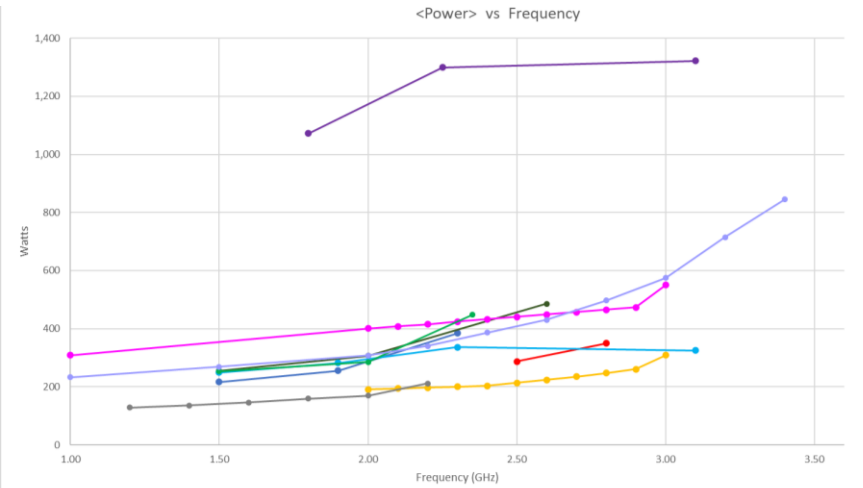


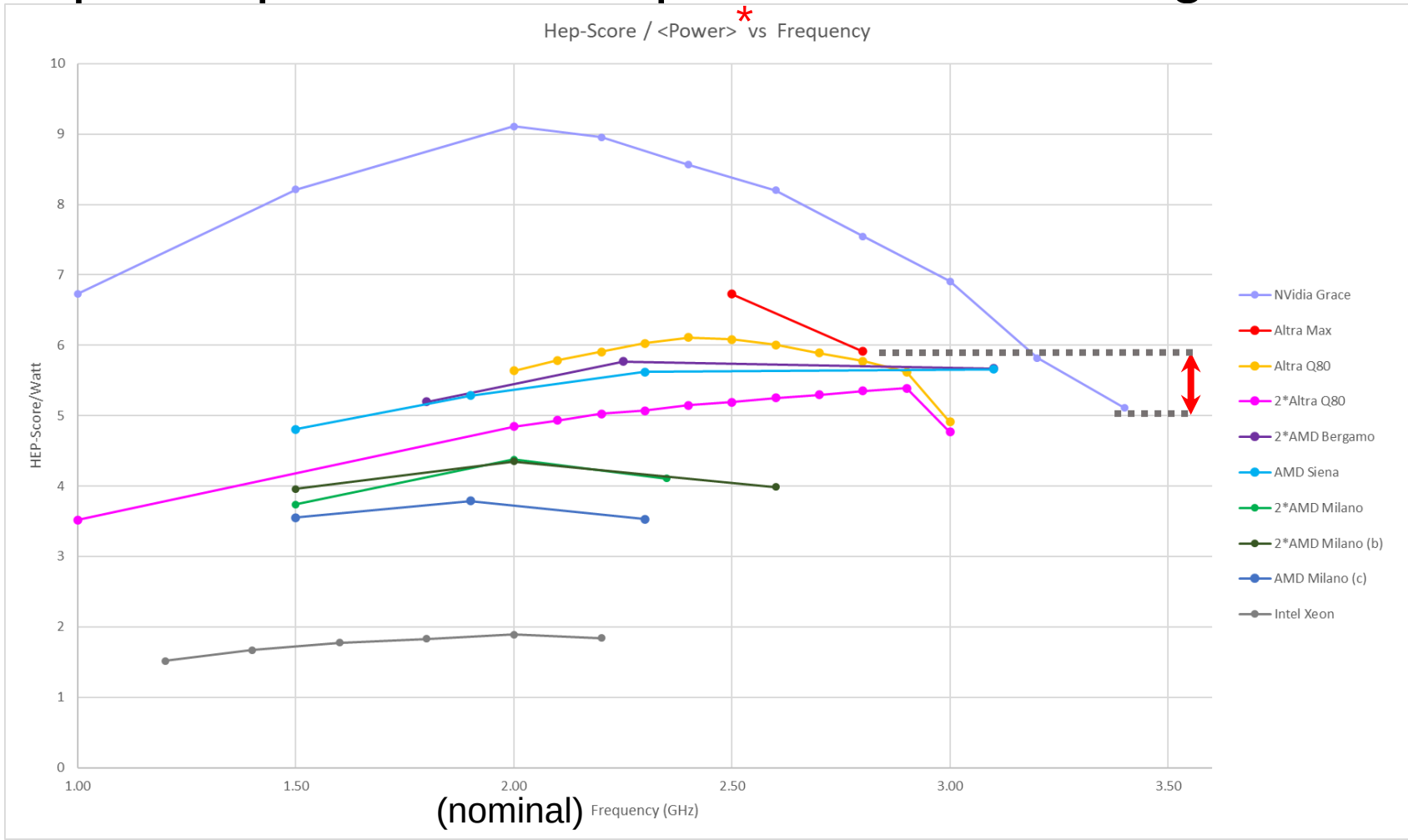
Frequency Throttling

HEP-Score increases with frequency (indeed), as power usage does too ...

HEP-Score/Watt vs. **CPU Frequency** gives a better picture of hardware potentials, and also shows optimal performances per watt at mid range.

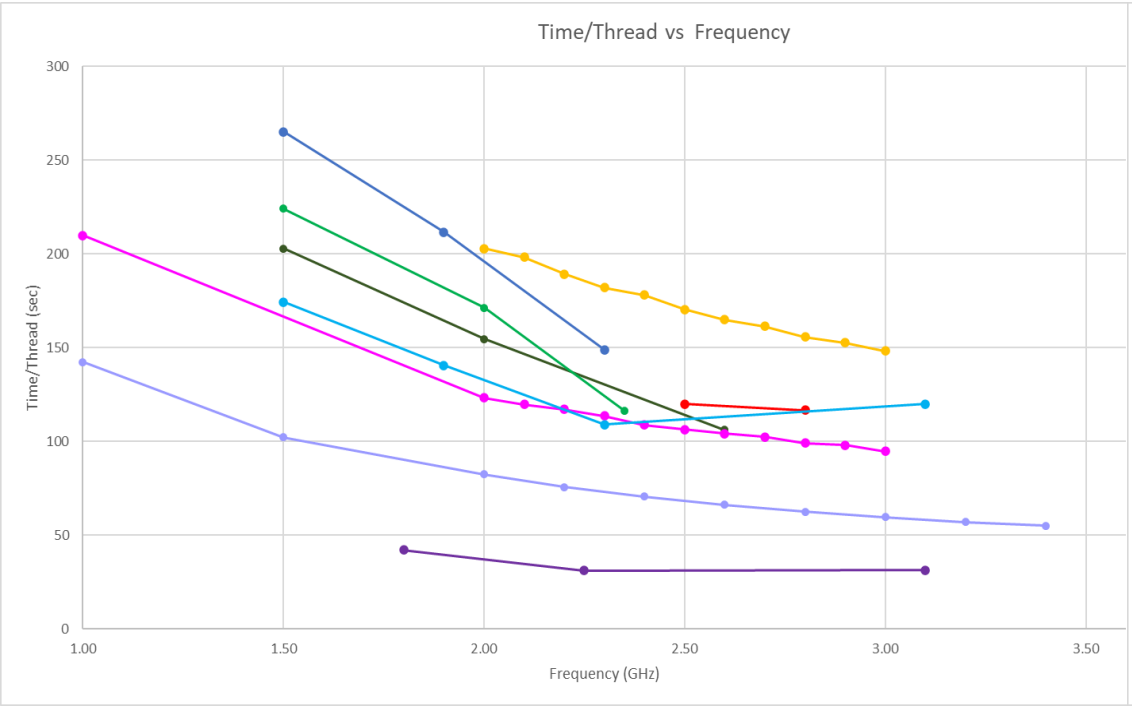


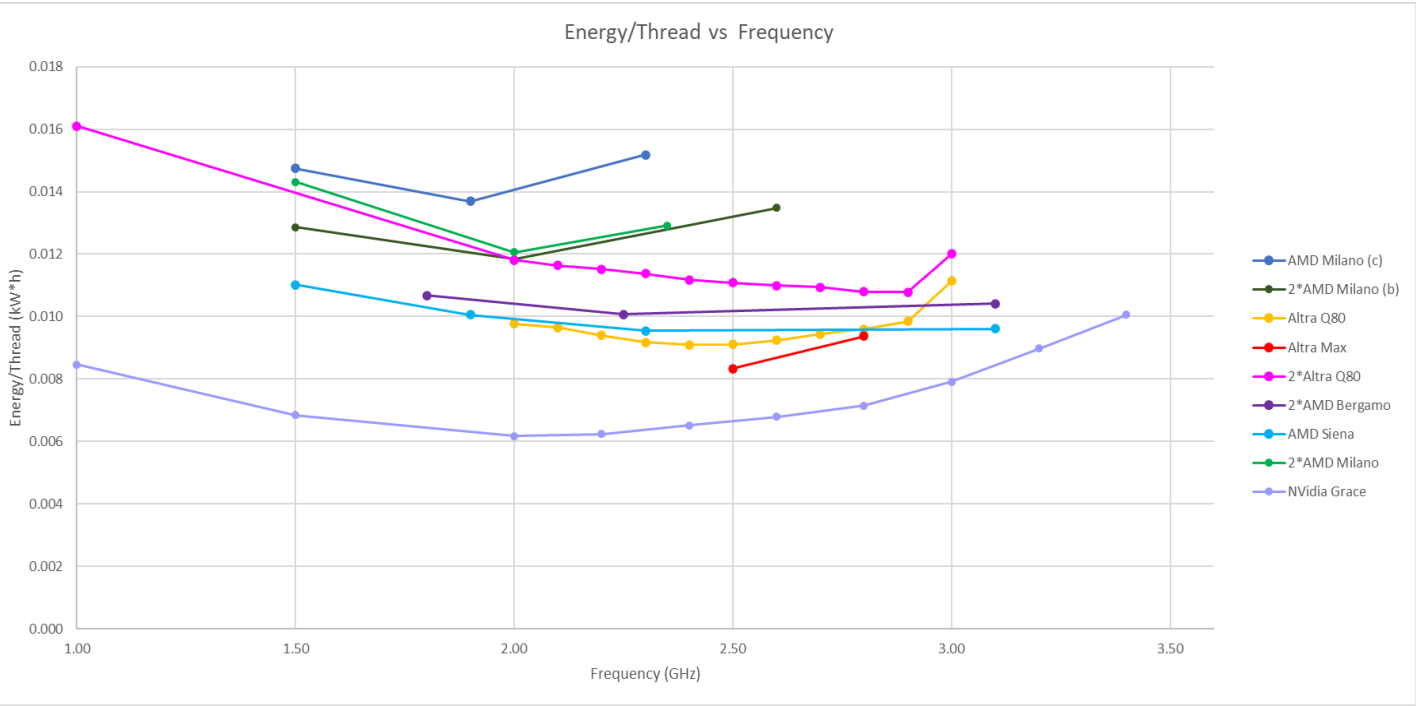




Frequency Throttling (2)

Execution time decreases almost linearly with frequency, while the total energy per job has a minimum below max. frequency (on all hardware).



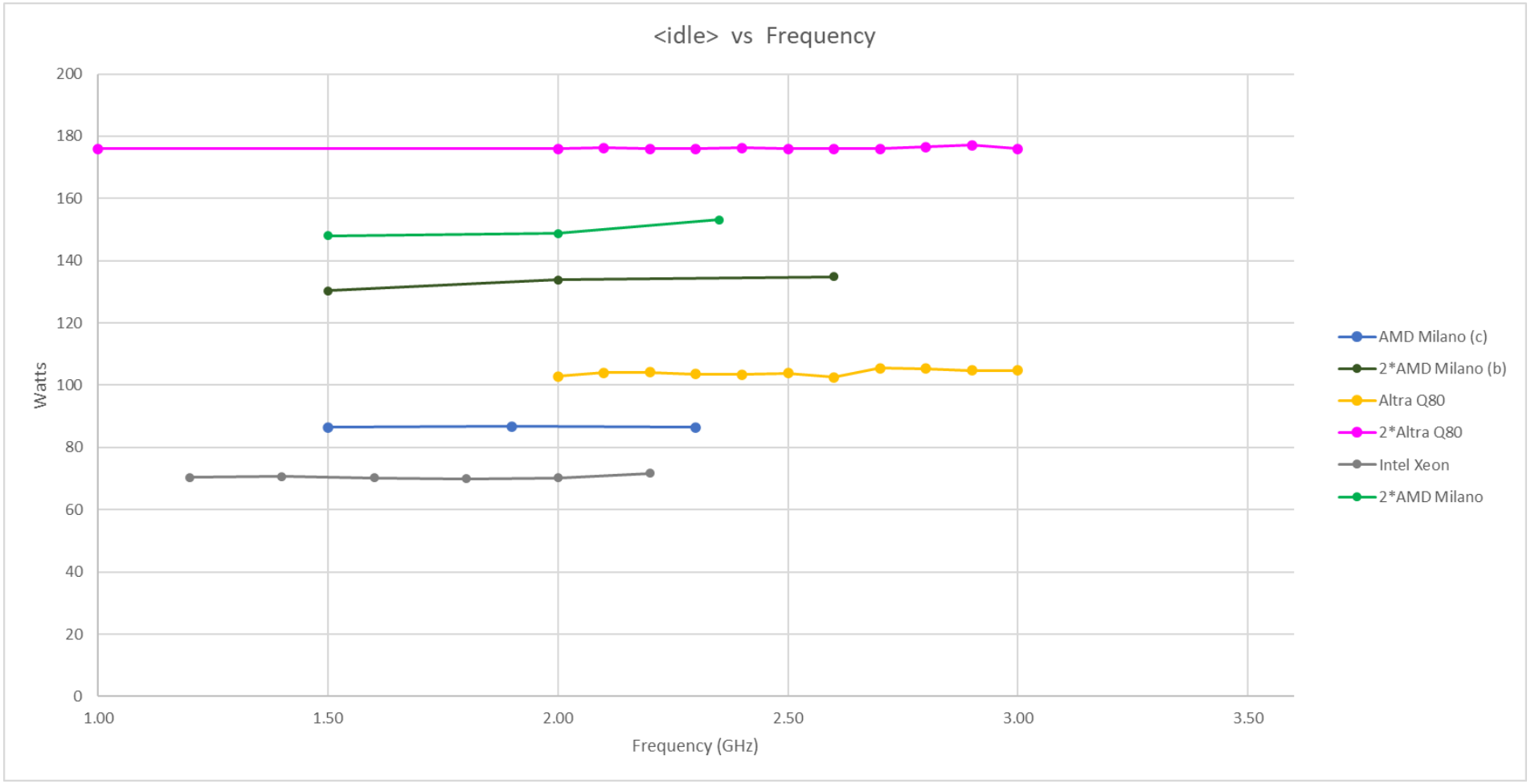


What is clear is that tuning the frequency down a notch can save quite some energy at the cost of a small increase in time, but how much time and energy savings greatly depends on the hardware!

In the end, it is up to each sites to find a compromise ...

Idle

When power measurements are taken during a pure '*sleep*' job, the **idle power drain** is independent from the CPU frequency !



This was done out of curiosity, after we observed a small correlation between CPU frequency and idle consumption …

… which turned out to come from the tail of the workloads, as the idle measurement was taken between workloads.

ARM Farm

From these tests, **ARM compute** have shown indications that it could outperform **x86** in terms of energy efficiency for HEP-style workloads.

Following our work in this field, we received a donation of ~2000 cores **ARM Q80-30** machines from **Ampere** and created an ‘ARM farm’ to see how easy it would be to advertise ARM resource on a Tier-2 grid site.

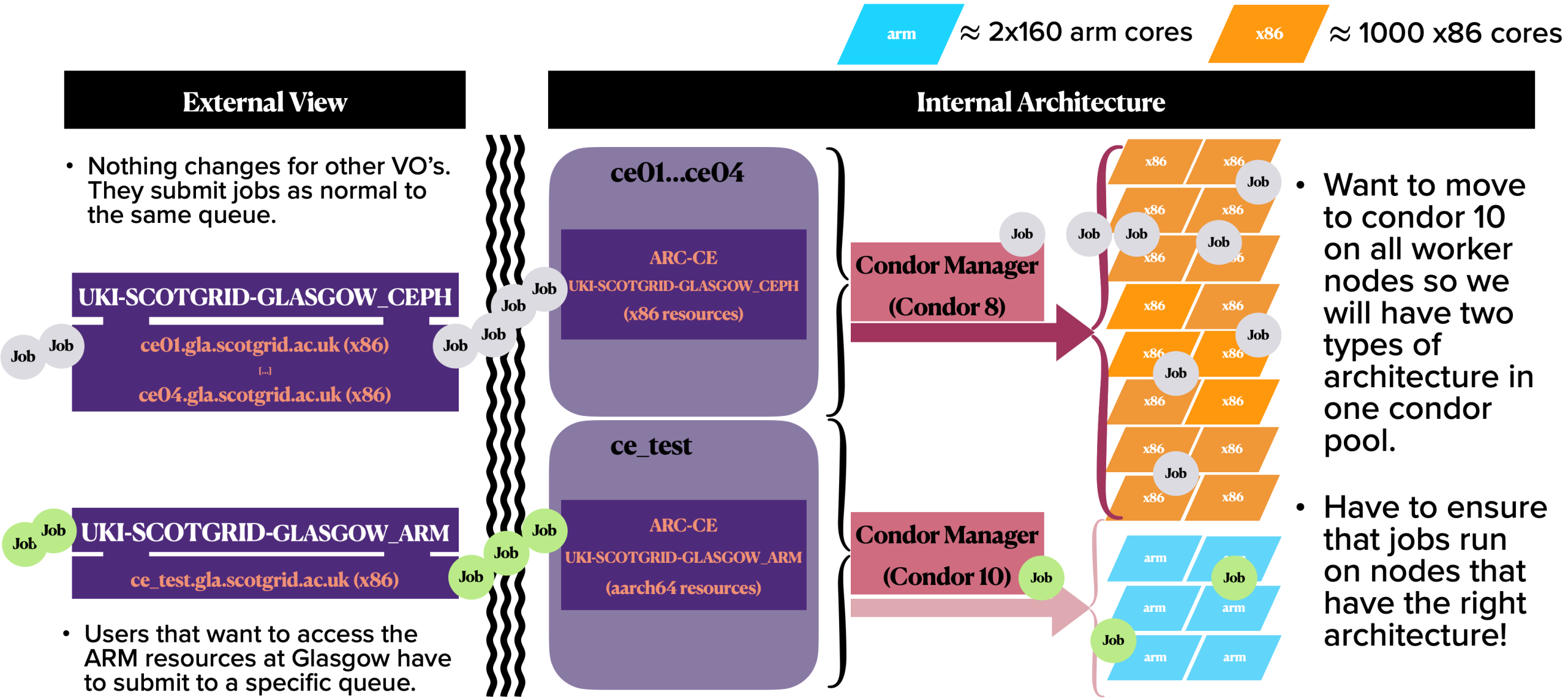
2*ARM80c: Dual Socket Ampere Altra Q80-30 80-Core Processor (MegaRAC)

CPU: 2 * ARM Q80-30 80C @ 3GHz (TDP 210W)
RAM: 512GB (32 x 16GB or 16 x 32GB) DDR4 3200MT/s → 3.2 GB/core
HDD: 2 * 1Tb NVMe (INTEL + SAMSUNG)
OS: Rocky 9.2

Initially this ARM resource was offered out for testing the to 4 main **LHC** experiments, with the idea being that once these 4 main experiments had run and validated their physics outputs, we could move to fully integrating said ARM resource at our grid site.

ARM Farm (2)

Job submission chain @ ScotGrid Glasgow:



ARM Validation

Currently **ATLAS**, **CMS**, and **ALICE**, have finished running extensive campaigns against our ARM cluster.

- **ATLAS**: Successfully ran at our site and physics validated
  ATLAS have already fully validated ARM work
  https://its.cern.ch/jira/browse/ATLPHYSVAL-919
- **CMS**: ARM work was run for physics validation purposes …
  Ran into problems with AAA – not a CMS site so data not local
  … but, as Katy said, the Physics Validation was unsuccessful ☹
- **ALICE**: Successfully ran work and it is physics validated (see next slide).
- **LHCb**: Should start running after the Easter Break.

Expanding the ARM farm:

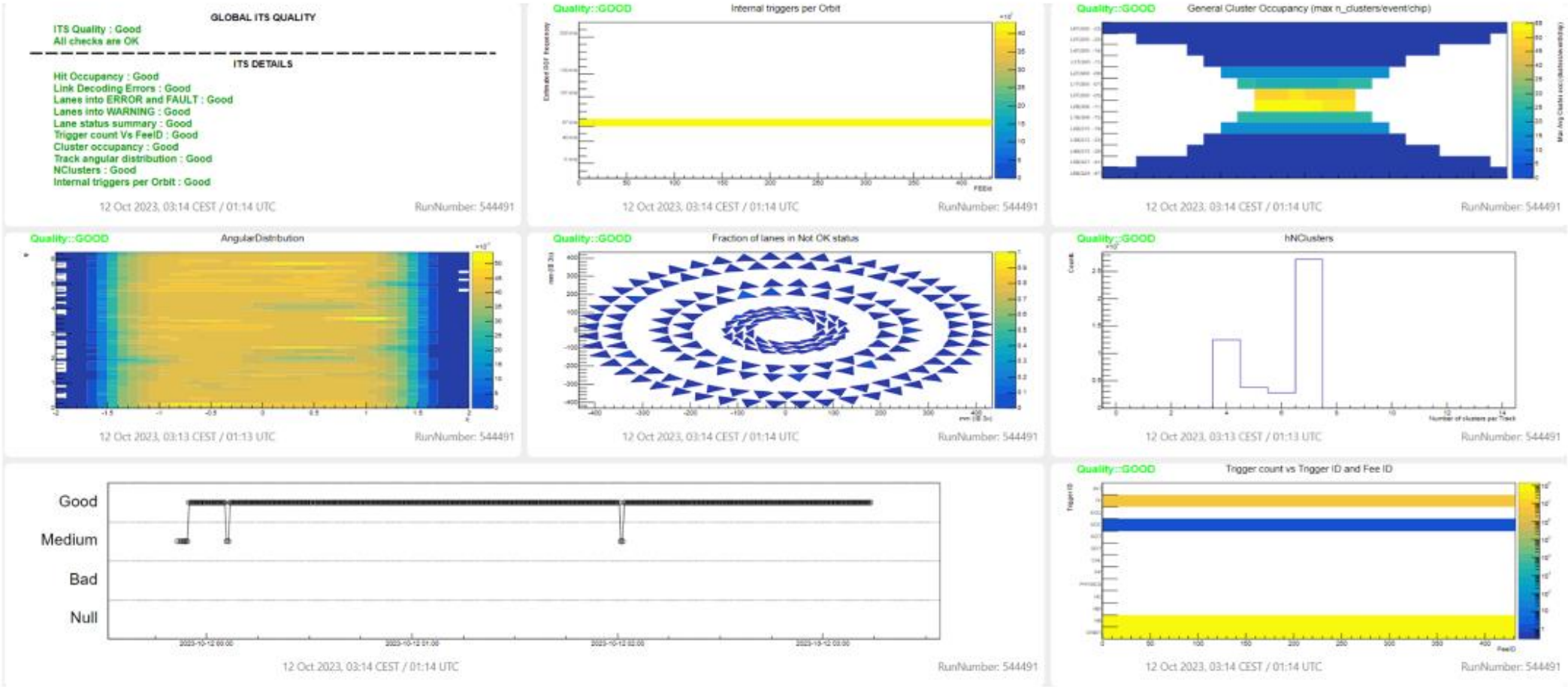
we have purchased ~2000 cores **AltraMax M128-30** to double our ARM provision!

ALICE Validation

A Large Ion Collider Experiment

ALICE

Single detector QA dashboard



ARM plots here are made in comparison to a prior production on **x86**.

Good means that they are equivalent.

Snapshot of the ALICE Inner Tracking System – by Latchezar Betev (**ALICE**)

Outlook

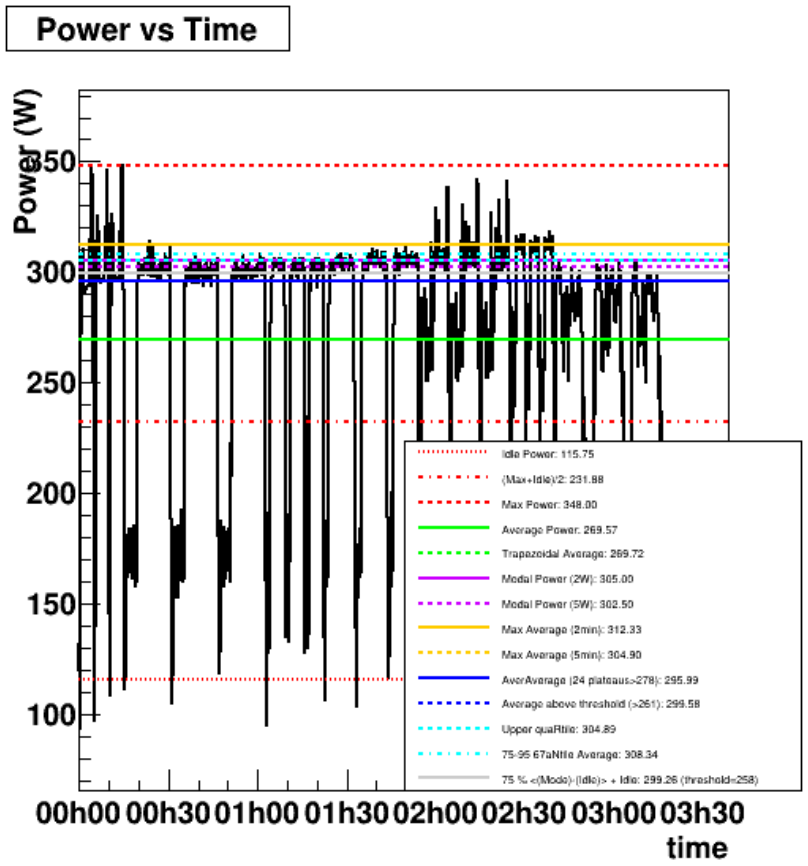
- Improve on the calculation, consolidate results, **streamline** the process from benchmarking to a reference table/excel (a lot of copy/paste still done by hands, and details are still changing).
  - energy measurement is now integrated as HEP-Score plugin (**v2** ?)
  - port/improve the analysis routine as yet another plug-in (**v2** +1 ?)

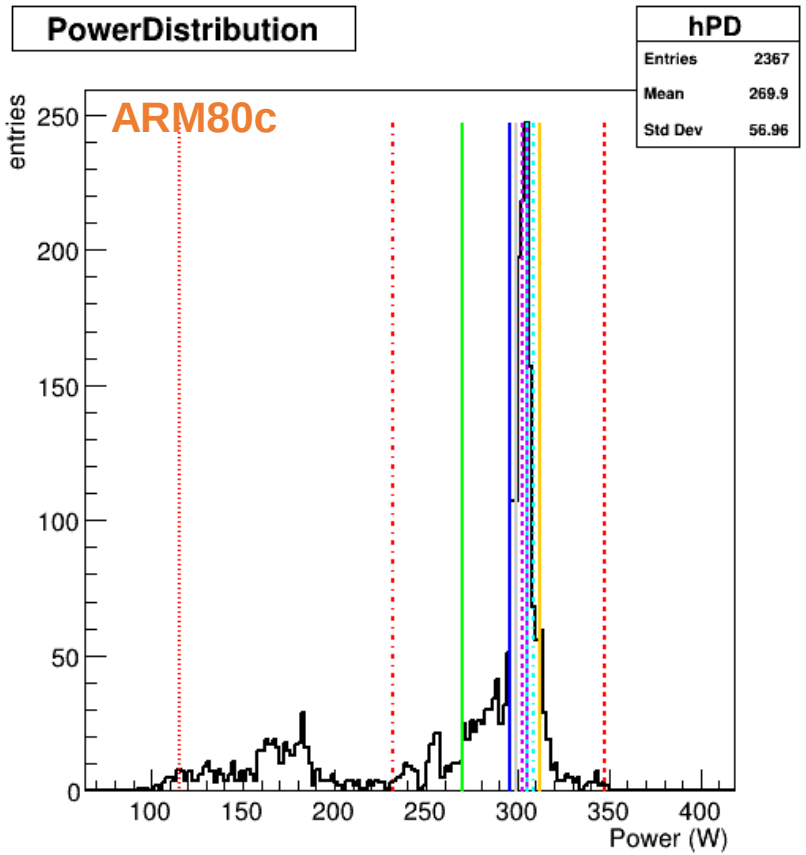
- Study the effect of **frequency throttling** at cluster level: we have an Ansible script to tune the frequency remotely!
  - How to use in production (e.g., daytime)?
  - We are testing the potential savings on a simulated cluster (Dwayne)

- Keep testing new hardware with old and new benchmarks:
  - benchmark **GPU**+**CPU** using Celeritas (Bruno)
  - benchmark **RISC-V** with ROOT & look for new developments (CMS)

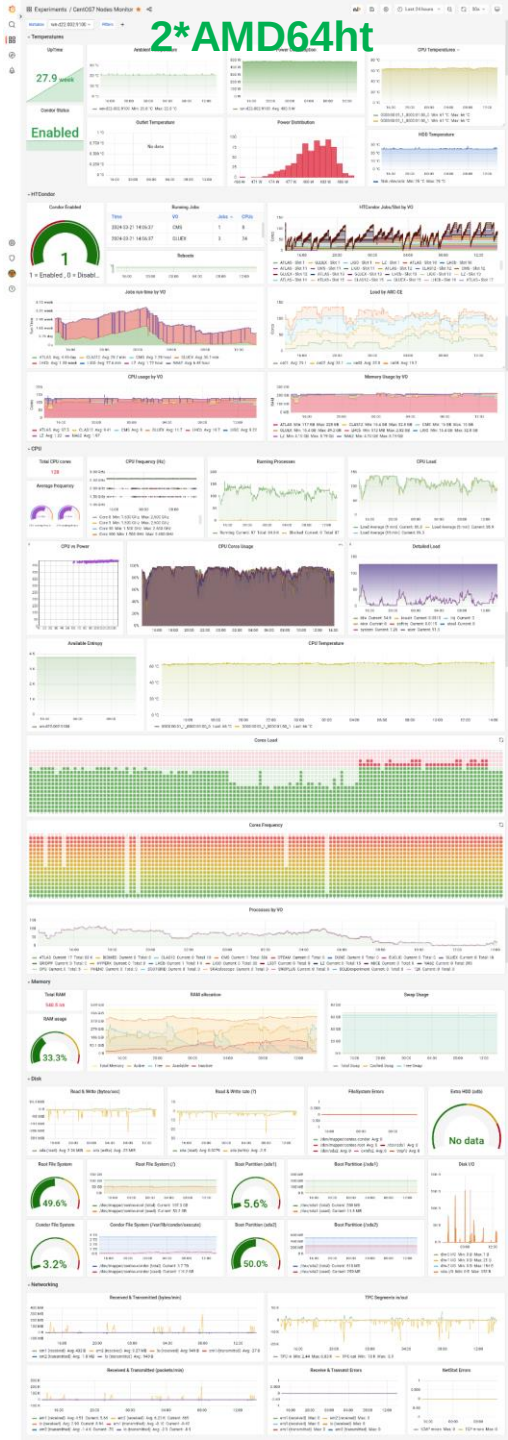
end

Leftovers …


Power vs Time
Power (W)
350
300
250
200
150
100
00h00 00h30 01h00 01h30 02h00 02h30 03h00 03h30
time
Idle Power: 115.75
(Max+Idle)/2: 231.88
Max Power: 348.00
Average Power: 269.57
Trapezoidal Average: 269.72
Modal Power (2W): 305.00
Modal Power (5W): 302.50
Max Average (2min): 312.33
Max Average (5min): 304.90
AverAverage (24 plateaus>278): 295.99
Average above threshold (>261): 299.58
Upper quaRtile: 304.89
75-95 67aNtile Average: 308.34
75 % <(Mode)-(Idle)> + Idle: 299.26 (threshold=258)


PowerDistribution
ARM80c
hPD
Entries 2367
Mean 269.9
Std Dev 56.96
entries
250
200
150
100
50
0
100 150 200 250 300 350 400
Power (W)


Power Consumption
500 W
450 W
400 W
350 W
300 W
250 W
200 W
150 W
100 W
50 W
0 W
14:00 16:00 18:00 20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00


2*AMD64ht
27.9
Enabled
No data
49.6%
5.6%
3.2%
50.0%
33.3%

My IPMI collector & analyser

IPMI

jdump.sh

(**root**'s crontab)

/tmp/ipmidump.json

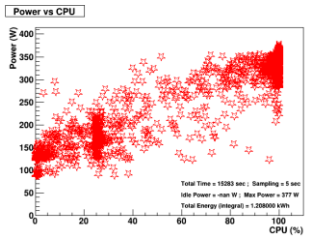
jget.sh + Job

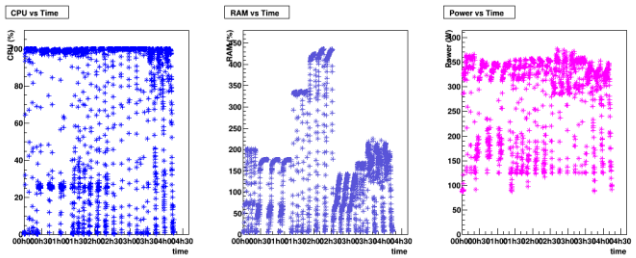
(**user**'s script)

ipmi_runtime.csv

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

date_yyyymmdd	time_hhm	cpu_usage	memory_u	cpu_frequ	ipmi_powe
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	89
08/06/2023	23:38:29	0.7	5.54	1	89
08/06/2023	23:38:34	0.8	5.55	1	92
08/06/2023	23:38:39	0.7	5.53	1	92
08/06/2023	23:38:44	1.3	5.55	1	89
08/06/2023	23:38:49	1.4	5.56	1	89
08/06/2023	23:38:54	0.8	5.55	1	92
08/06/2023	23:38:59	0.7	5.56	1	92
08/06/2023	23:39:04	0.9	5.55	1	88
08/06/2023	23:39:09	0.9	5.55	1	88
08/06/2023	23:39:14	0.6	5.55	1	88
08/06/2023	23:39:19	0.8	5.55	1	88
08/06/2023	23:39:24	0.9	5.55	1	89
08/06/2023	23:39:29	1.1	5.55	1	89
08/06/2023	23:39:34	1	5.55	1	90
08/06/2023	23:39:39	0.8	5.56	1	90
08/06/2023	23:39:44	1.1	5.56	1	91
08/06/2023	23:39:49	1.2	5.56	1	91
08/06/2023	23:39:54	32.1	5.99	1	104
08/06/2023	23:39:59	26.9	6.96	1	104
08/06/2023	23:40:04	64.5	11.59	2.8	122
08/06/2023	23:40:09	96.1	29.26	2.8	122
08/06/2023	23:40:14	97.2	50.6	2.8	285

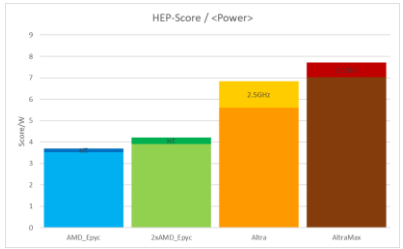




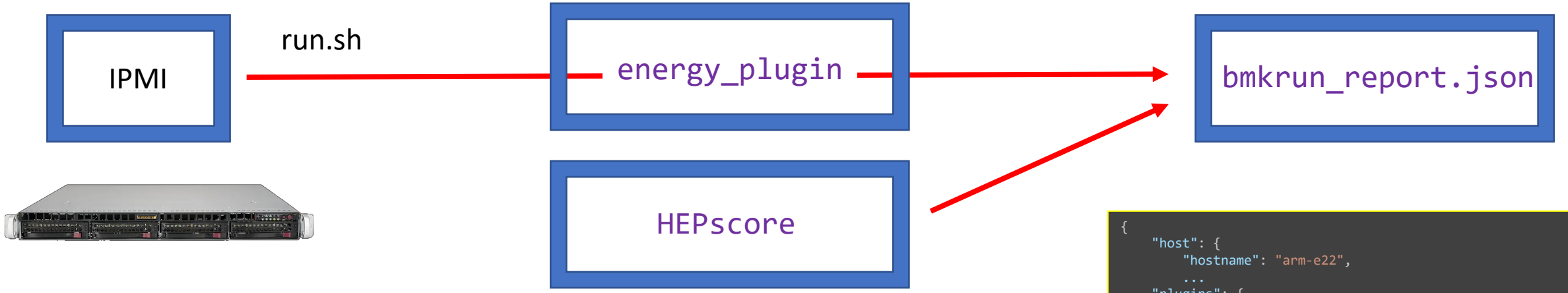
ipmi2root.C

(ROOT script)

Excel



HEPscore Energy Plug-in



Values can be exported back to CSV file via custom script, and from there follow the same analysis chain …

ipmi2root.C

about 500 lines


```
{
    "host": {
        "hostname": "arm-e22",
        ...
    "plugins": {
        ...
            "hepscore": {
                "cpu-frequency": {
                    ...
                "power-consumption": {
                    "start_time": "2023-11-03T15:55:11.310751Z",
                    "end_time": "2023-11-03T19:11:11.293188Z",
                    "values": [
                        115.0,
                        303.0,
                        ...
                        169.0
                    ],
                    "statistics": {
                        "min": 108.0,
                        "mean": 269.0862944162437,
                        "max": 344.0
                    },
                    "config": {
                        ...
                },
                ...
            "score": 1509.9384,
            "status": "success"
        }
    }
}
```