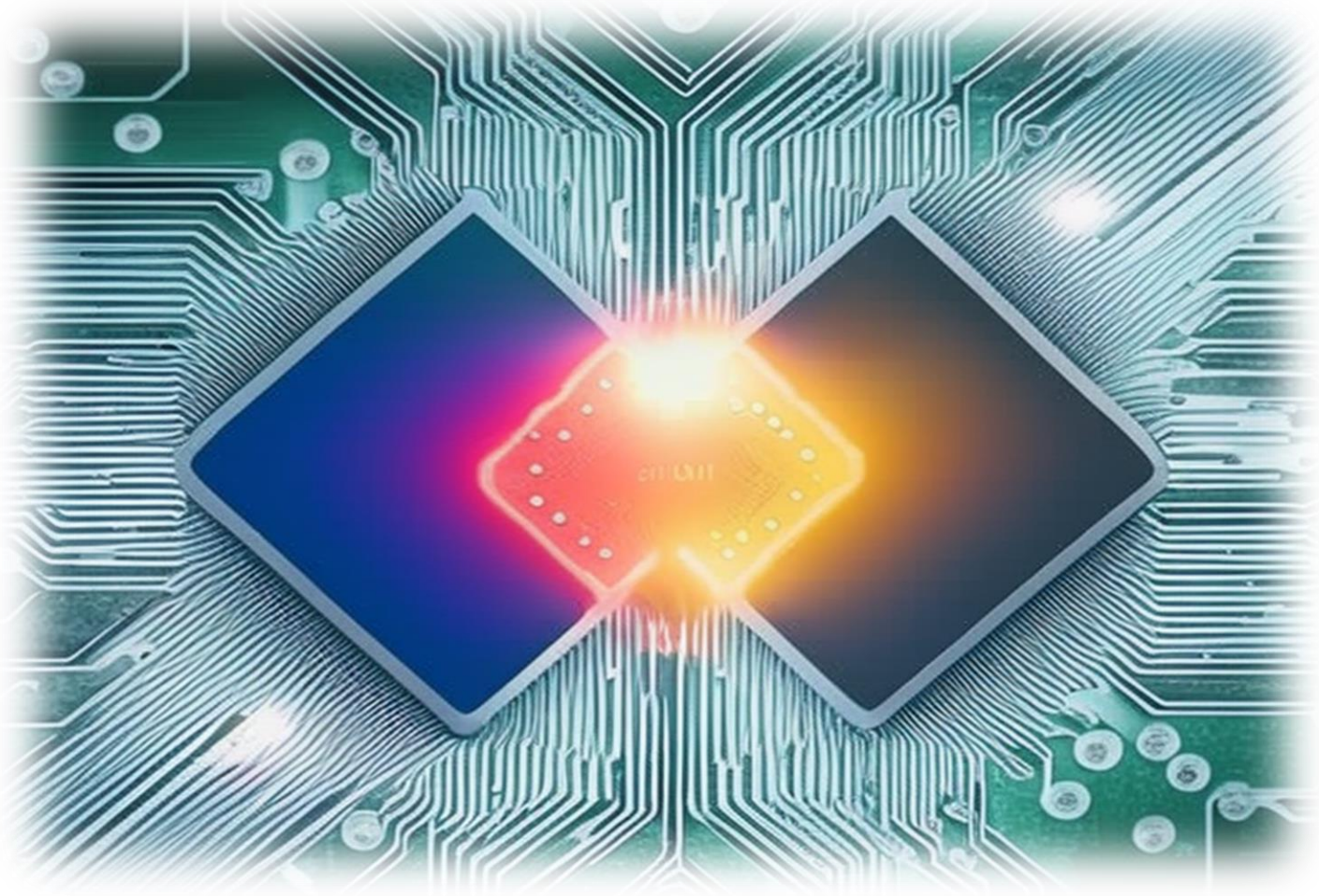
Energy measurement plugin for the HEP Benchmarking Suite



Introduction

- The power consumption of computing is coming under intense scrutiny worldwide, driven both by concerns about the carbon footprint, and by rapidly rising energy costs.

- We want to enable the HEP community to make informed choices when purchasing hardware, not only in term of performances, but also of energy usage.

- Integrating power measurement into the HEP Benchmark Suite is the obvious way to go ... it has been teased in previous talks, and finally it is happening!

- With the HEP-Score being now available for **arm64** CPUs, it will become even easier to compare the energy consumption of different architecture over identical HEP workloads.

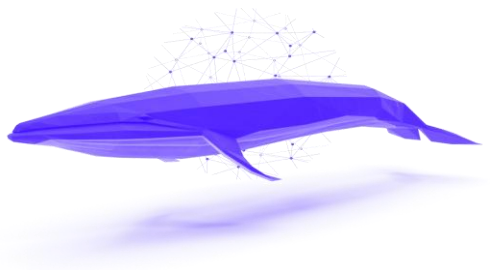
ScotGrid @ Glasgow: Emanuele Simili, Gordon Stewart, Samuel Skipsey, Dwayne Spiteri, David Britton

HEPiX @ CERN: Domenico Giordano, Gonzalo Menendez Borge, Johannes Elmsheuser

Power Readings (how it was)

CPU, RAM usage metrics and IPMI power readings were extracted and logged by two custom scripts:

- The 1st script is a cron job (by **root**) that every 10 seconds exports IPMI power readings with a timestamp to `/tmp/ipmidump.txt` (… because *IPMItool* requires **root** privileges).
- The 2nd script is executed by the **user** and it takes in the job to be benchmarked as argument. When executed, it starts grabbing the IPMI readings from the dump file, attaches few more info (CPU, RAM) and appends them to a CSV file. After 1 min. sleep (so to measure idle power), it runs the given job, and then waits another min. after the job is finished before quitting.
- After the job is done, the CSV file is exported locally and processed in ROOT (time profiles plots and power integration). Cumulative results are then visualized in Excel.

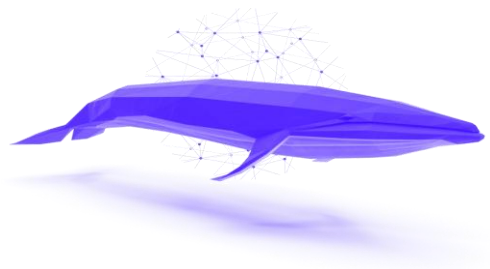


In addition, all servers are running a *node_exporter* client, and we use *Prometheus* and *Grafana* easy visualization and general monitoring purposes.

Power Readings (how it will be)

CPU, RAM usage metrics, and IPMI, GPU power readings will be extracted by a Python plug-in within the HEP Benchmarking Suite:

- A 1st Python module will run along with the workload and log hardware metrics and power readings to a CSV file every N seconds. Before running the actual workload and at the end of the run, the scripts collects idle power for 1 minute (or any custom interval).
- At the end of the workload, a 2nd Python module will read the exported metrics from the CSV and calculate averages and the total energy consumption (power integration).
- Cumulative results will then be reported together with the HEP-Score result in the *json* file.



In addition, nodes can run their own *node_exporter* or any other client for general monitoring purposes.

Power Readings (how it is)

For now there is a beta version of both the exporter and analyser script, both in python.
They run standalone and they are not yet integrated within the suite, but work is in progress …

- 1st Python module:

  ipmi_exporter.py

```
def grabmetrics(f,starTime):
    # Run BASH commands to grab metrics
    cmd_cpu  = r""" top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" """
    cmd_freq = r""" cpupower frequency-info -f | grep "current CPU frequency:" """
    cmd_ram  = r""" free | grep "Mem:" """
    cmd_ipmi = r""" ipmitool dcmi power reading | grep "Instantaneous power reading:" """
    cmd_gpu  = r""" nvidia-smi --query-gpu=power.draw --format=csv,noheader,nounits | awk '{s+=$1} END {print s}' """

    # Store metrics in runtime variables
    timeNow = dt.datetime.now()
    runtime = int((timeNow - starTime).total_seconds())
    cpu = 100 - float(runcommand(cmd_cpu))
    frq = getnumbers(runcommand(cmd_freq),0) / 1000000
    ram = getnumbers(runcommand(cmd_ram),1) /1024 /1024
    powa = getnumbers(runcommand(cmd_ipmi),0)
    gpu = float(runcommand(cmd_gpu))

    # Concatenate metrics in CSV rows and append these rows to a CSV file
    row = f"{runtime}, {round(cpu,1)}, {round(frq,1)}, {round(ram,1)}, {round(powa,1)}, {round(gpu,1)}"  # no units
    f.write(row + " \n")
```

- 2nd Python module:

  ipmi_analyser.py

```
    # Custom functions that reads system metrics from CSV file and calculates stuff
    def readmetrics(self):
        ...
                # Clean up exported metrics
                row = line.strip()
                cols = row.split(",")
                ...

                # Calculates max/min values
                if(powa > self.maxPower):  self.maxPower = powa
                ...
                # Calculates idle/job time integrals
                if(not isIdle):
                    self.totalEnergy = self.totalEnergy + powa
                    ...
                else:
                    self.idlePower = self.idlePower + powa

         # Prints all imported metrics
         print(f"{counter}):  {tsec}sec, {round(cpu,1)}%, {frq}GHz, {round(ram,1)}Gb, {round(powa,1)}W, {round(gpu,1)}W ... {status}")
```

Standalone Example

```
$ sudo python3 ipmi_exporter.py

Starting IPMI exporter ...
* 0, 0.2, 1.6, 11.0, 248.0, 101.5 *
* 1, 0.2, 3.8, 11.0, 363.0, 101.8 *
…
* job is running: sleep 100 ...
* 60, 0.2, 1.8, 11.0, 249.0, 101.8 *
* 61, 0.1, 2.3, 11.0, 246.0, 101.5 *
...
* Finished job. Waiting 60 seconds before quitting ...
* 120, 0.2, 2.1, 11.0, 257.0, 101.7 *
…
```

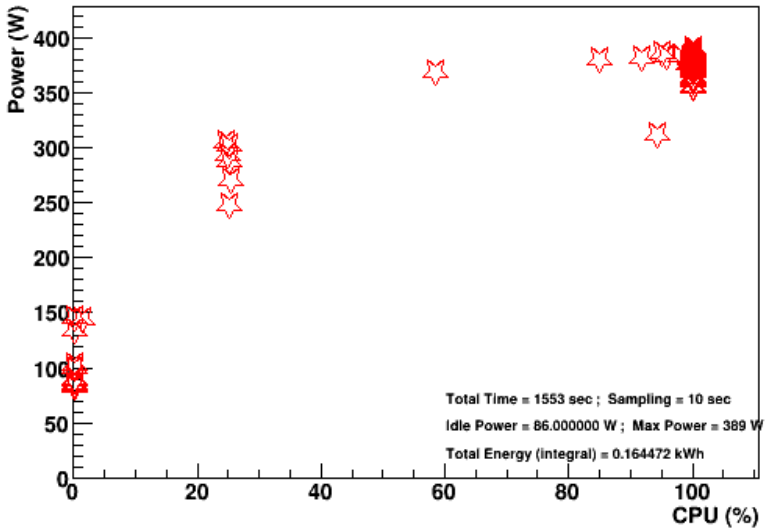
```
$  python3 ipmi_analyser.py

Starting IPMI analyser ...
0):  0sec, 0.2%, 1.6GHz, 11.0Gb, 248.0W, 101.5W ... [idle]
…
60):  10sec, 0.2%, 1.8GHz, 11.0Gb, 249.0W, 101.8W ... [busy]
…

*** Totals ***
* Total Time = 29 sec , Sampling = 1 sec ;  Idle time = 2 * 10 sec
* Average Idle Power = 282.8 W ;  Max Power = 374.0 W ;  Average Power = 275.9 W
* Total Energy (integral) = 0.0023 kWh
* Total RAM used = 11.0 GB ;  Min. RAM = 11.0 , GB ;  Installed RAM = 256 GB
* Frequency = 1.5 GHz - 3.2 GHz
* GPU average power = 101.8 W ;  GPU total energy = 0.00085 kWh
```

Job Profiles (from CSV)

Power vs CPU



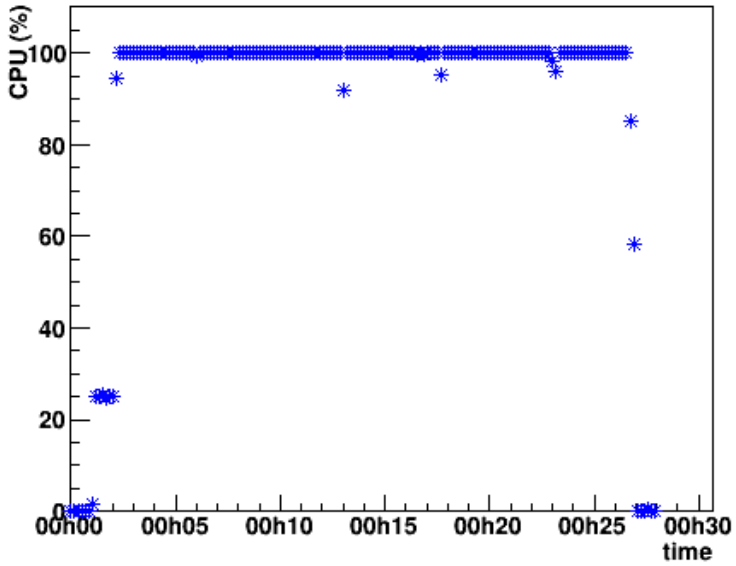
x86_64: Single AMD EPYC 7003 series (GigaByte)

CPU:	AMD EPYC 7643 48C/96T @ 2.3GHz (TDP 300W)
RAM:	256GB (16 x 16GB) DDR4 3200MHz
HDD:	3.84TB Samsung PM9A3 M.2 (2280)

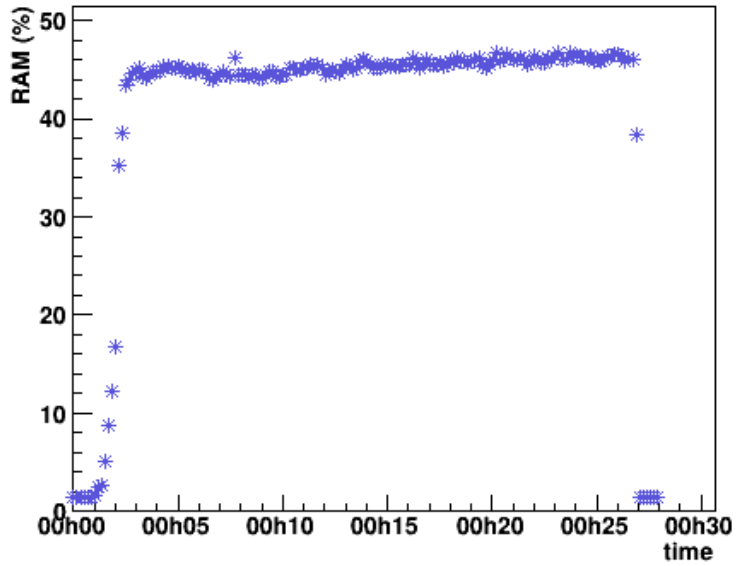
Frequency vs Time

...

CPU vs Time



RAM vs Time



Power vs Time