



Measuring Science per Watt

Hardware-Aware Power Profiling for Heterogeneous HPC



David Goz

INAF – Osservatorio Astronomico di Trieste

ICSC – Centro Nazionale di Ricerca in HPC, Big Data e Quantum Computing

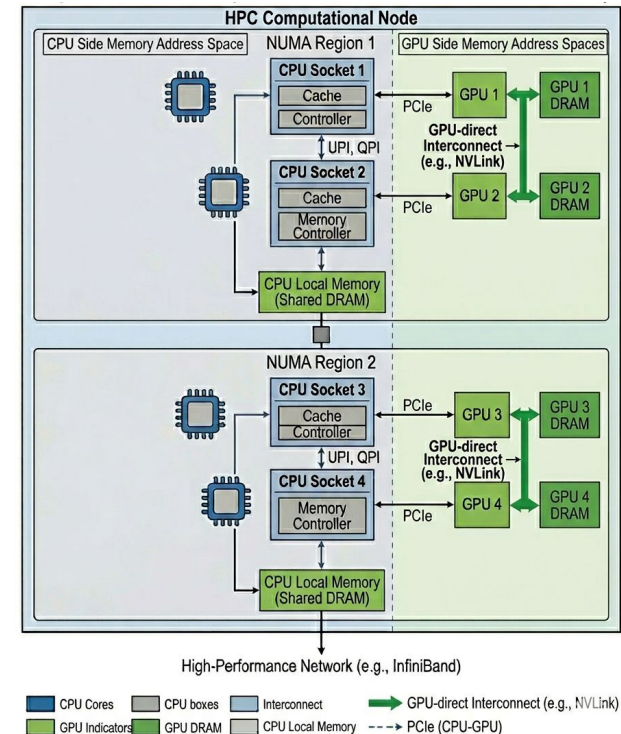
PPMT · INAF GitLab · Open Source

Why Topology-Aware Power Profiling

A modern HPC node packs 4 CPU sockets across 2 NUMA regions and up to 4 GPUs -- each with its own power domain, its own HBM, its own thermal envelope.

A flat, single-number power reading cannot tell you which one is actually burning the watts. Without topology, energy measurements become spatially ambiguous -- and you optimize the wrong thing.

"The most important metric is no longer how fast you compute -- it's how much science you get per watt."



Independent power domains per socket and per GPU -- this is what PPMT measures against

From PMT to PPMT

THE FOUNDATION

PMT

Power Measurement Toolkit - ASTRON

- Lightweight C++ library, single process
- Direct vendor APIs: RAPL, NVML, rocm-smi
- Background thread samples power in real time



THIS WORK

PPMT

Parallel Power Measurements Toolkit - INAF

- Concurrent, synchronized measurement across MPI processes
- Single process to a parallel, scalable framework
- Same low-overhead, sensor-accurate sampling underneath

PPMT keeps PMT's hardware-accurate sensor access and adds a topology-aware MPI orchestration layer on top

Core API & Design Goals

1. Header inclusion

```
// Import the native C++ wrapper
#include <ppmt/ppmt.hpp>
```

```
// Import the core C API
#include <ppmt/ppmt_c.h>
```

2. Initialization

```
// C++ Application (Stack Object)
// CPUs = true, GPUs = true, GPU IDs vector
PPMT ppmt(true, true, gpuIDs);
```

```
// C Application (opaque struct)
// factory needs all config args
ppmt_c *profiler = ppmt_create(1,0,NULL,0,0,0);
// free explicitly when done in C!
ppmt_destroy(profiler);
```

3. Instrumenting regions

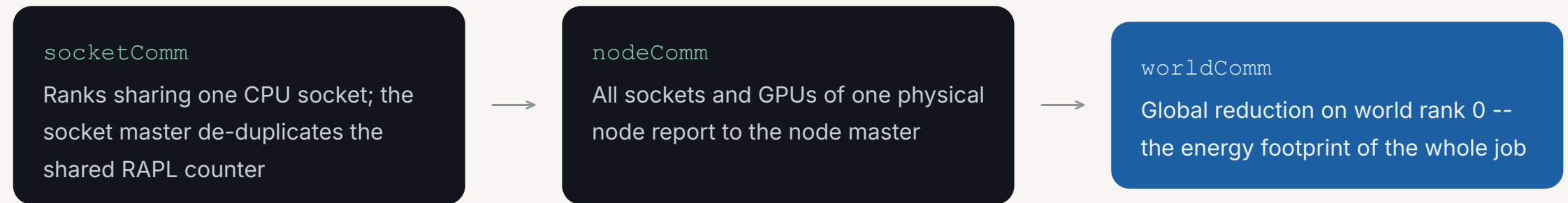
```
ppmt.startCPU("cpu_tag_name");
// ... heavy computation here ...
ppmt.stopCPU("cpu_tag_name");
ppmt.startGPU("gpu_tag_name", devID);
// ... GPU kernel here ...
ppmt.stopGPU("gpu_tag_name", devID);
```

```
ppmt_startCPU(profiler, "cpu_tag_name");
// ... heavy computation here ...
ppmt_stopCPU(profiler, "cpu_tag_name");
ppmt_startGPU(profiler, "gpu_tag_name", devID);
// ... GPU kernel here ...
ppmt_stopGPU(profiler, "gpu_tag_name", devID);
```

WHAT'S NEW

Topology-Aware Energy Metrics

At start-up PPMT builds three nested MPI communicators -- automatically, no configuration -- so every joule is traced to the socket, node or GPU that spent it, then rolled up to one job total.



One run, four views: per rank · per socket · per node · whole job

Two Devices, Two Aggregation Rules

Real run, tag [main]: 16 MPI ranks on 4 nodes, each node 1 CPU socket + 4 GPUs

CPU -- RAPL: MAX per socket, then SUM

One counter per socket, shared by its ranks: all 4 read ~342.0 kJ.
MAX de-duplicates, then sockets are summed.

Job: 1.368 MJ · 999 W

Naive sum over ranks: 5.47 MJ -- 4x over-count

GPU -- NVML: SUM across devices

One counter per GPU, independent power rails: 105.0 + 105.9 +
106.3 + 101.6 kJ on node 0, then nodes are summed.

Job: 1.722 MJ · 1258 W

Additive by physics -- summing is correct

Written automatically by one `ppmt.show()` call



VALIDATION

PPMT v0.1.2 vs LIKWID 5.5.2 -- same RAPL counter, independent implementations

Validated Against [LIKWID tool](#)

1.7%

package-energy difference

inside PPMT's own 100 ms sampling uncertainty ($\leq$ 2.4% for this region)

Socket PKG	LIKWID	PPMT	Δ
Energy	75.30 J	74.00 J	-1.7%
Power	35.90 W	35.18 W	-2.0%
Job-level total	no	socket · node · world	

Twin MPI benchmarks, differing only in instrumentation; kernel compiled to identical instructions (objdump-verified). Single-socket Xeon Silver 4108, 2 ranks.

Single-node pilot. Full campaign (repeatability, rank count, imbalance, region length, overhead) in progress for v0.1.2.

Open Source, Production-Ready

AVAILABILITY

INAF GitLab

[Release v0.1.2 -- September 2026](#)

Apache License 2.0 · strict C++20

FOUR BUILD VARIANTS

g++
serial CPU

nvc++
serial GPU

mpic++
MPI (CPU)

mpic++
MPI + GPU

AUTHORS AND CONTRIBUTIONS

[David Goz](#) @INAF-OATs

Conceptualization · Methodology · Software: development · Repository: maintainer

[Giovanni Lacopo](#) @INAF-OATs

Conceptualization · Methodology · Repository: developer

[Cristiano Urban](#) @INAF-OATs (IA2)

Software: CI infrastructure development, automation pipelines and deployment environment · Repository: maintainer

[Giuliano Taffoni](#) @INAF-OATs

Conceptualization · Methodology

[Cristina Knapic](#) @INAF-OATs (IA2)

Software: CI infrastructure development and GitLab provider

What's next

GFLOP/s/W via PAPI --
science-per-watt, made literal

Per-thread OpenMP profiling

A Python interface (PyPPMT)

Time-resolved aggregation
under load imbalance

RICK on Leonardo -- for Radio Astronomy

Radio Imaging Code Kernels (RICK) is a massively parallel radio-interferometric imaging pipeline for the SKA era -- gridding, w-stacking and FFT. We instrumented RICK v2.0 with PPMT v0.1.1 on CINECA's Leonardo.

DCGP -- CPU only

- 2x Intel Xeon Platinum 8480+, 112 cores/node
- No GPU

Booster -- CPU + GPU

- 1x Xeon 8358 + 4x NVIDIA A100 64GB
- NVLink 3.0, 200 GB/s

REFERENCES

E. De Rubeis et al., "Accelerating radio astronomy imaging with RICK," Astronomy and Computing 50 (2025) 100895 · [doi:10.1016/j.ascom.2024.100895](https://doi.org/10.1016/j.ascom.2024.100895) · [arXiv:2411.07321](https://arxiv.org/abs/2411.07321)

G. Lacopo et al., "Accelerating radio astronomy imaging with RICK: a step towards SKA-Mid and SKA-Low," Astronomy and Computing 55 (2026) 101074 · [doi:10.1016/j.ascom.2026.101074](https://doi.org/10.1016/j.ascom.2026.101074) · [arXiv:2601.19714](https://arxiv.org/abs/2601.19714)

The Gridding Paradox

2x

DCGP (CPU) more energy-efficient than Booster (GPU) on the same kernel, same 8 nodes

DCGP (CPU) -- Gridding

S_E = 0.59

Booster (GPU) -- Gridding

S_E = 0.27

Gridding's irregular memory access lets CPU sockets idle down under memory stalls -- GPUs cannot, and pay for it twice: slower, and less efficient.

A flat, node-wide energy counter would never have shown this -- PPMT's per-kernel, per-device view surfaced it automatically.

Source: D. Goz, G. Lacopo, C. Urban, G. Taffoni, C. Knapic, "PPMT: A Scalable and Hardware-Aware Power Measurement Toolkit for Distributed HPC Applications – Evaluating Energy-to-Solution on Radio Astronomy Pipelines," submitted to ITADATA 2026 Workshop, Scientific HPC in the pre-Exascale era (3rd ed.)

Wrapping Up PPMT

- **Topology-aware:** energy per rank, socket, node and whole job -- automatic, physically correct aggregation
- **Validated:** within 2% of LIKWID on CPU package energy
- **In production:** RICK on Leonardo -- gridding 2× more energy-efficient on CPUs than on GPUs
- **Open:** INAF GitLab, Apache 2.0, C++ and C interfaces

Thank you

Questions?

david.goz@inaf.it · INAF-OATs · ICSC
PPMT: INAF GitLab,
v0.1.2