Master in High Performance Computing

Energy efficiency and application-level energy profiling

Moreno Baricevic

SISSA
Scuola Internazionale Superiore di Studi Avanzati

The Abdus Salam International Centre for Theoretical Physics

Master in High Performance Computing

Summary

- background

- software tools
- testbed

- benchmarks and methods
- results

- observations

Why energy efficiency is a hot topic?

- due to energy cost and sustainability, **power consumption** of data centers is one of the rising problems, as well as the **main limiting factor** for the development of **exascale** systems

- “green” computing more often coupled to HPC, introducing concept of **“high-efficiency” computing** (Top500 complemented by Green500)

- the **cost in energy** during cluster life cycle may be **comparable to its acquisition cost**

What the growing interest in energy efficiency is leading to?

- growing need for **energy-aware resource management and scheduling**

- energy profiling of applications gives important information about the **“cost in energy”** of running a specific code on a specific machine (using a specific library)

- hardware/software probes reporting **near-real-time power consumption**, and interface software to access this information

How to reduce the power consumption of HPC resources?

- policy-based automatic **power management**
  (idle nodes into power saving modes, power on/wake nodes for new workload, ...)
- exploit **hardware capabilities**: DVFS / power-saving states / performance states / turbo mode
- **power capping** policies (maximum amount of overall admitted power consumption)
- assign workload to highest **performance-per-watt** resources first
- **energy-aware** resource management systems and schedulers able to exploit all of the above, implementing **out-of-band and unattended energy assessment capabilities**

How to measure the power consumption?

- **external** hardware-dependent probes and sensors, PDUs, power grid counters

- **local** machine hardware sensors

- metrics obtained accessing hardware counters available on most major **microprocessors**

RAPL (Running Average Power Limit) (MSR by INTEL)

  – monitors, controls, and gets notifications on SoC power consumption (platform level power capping, monitoring and thermal management)

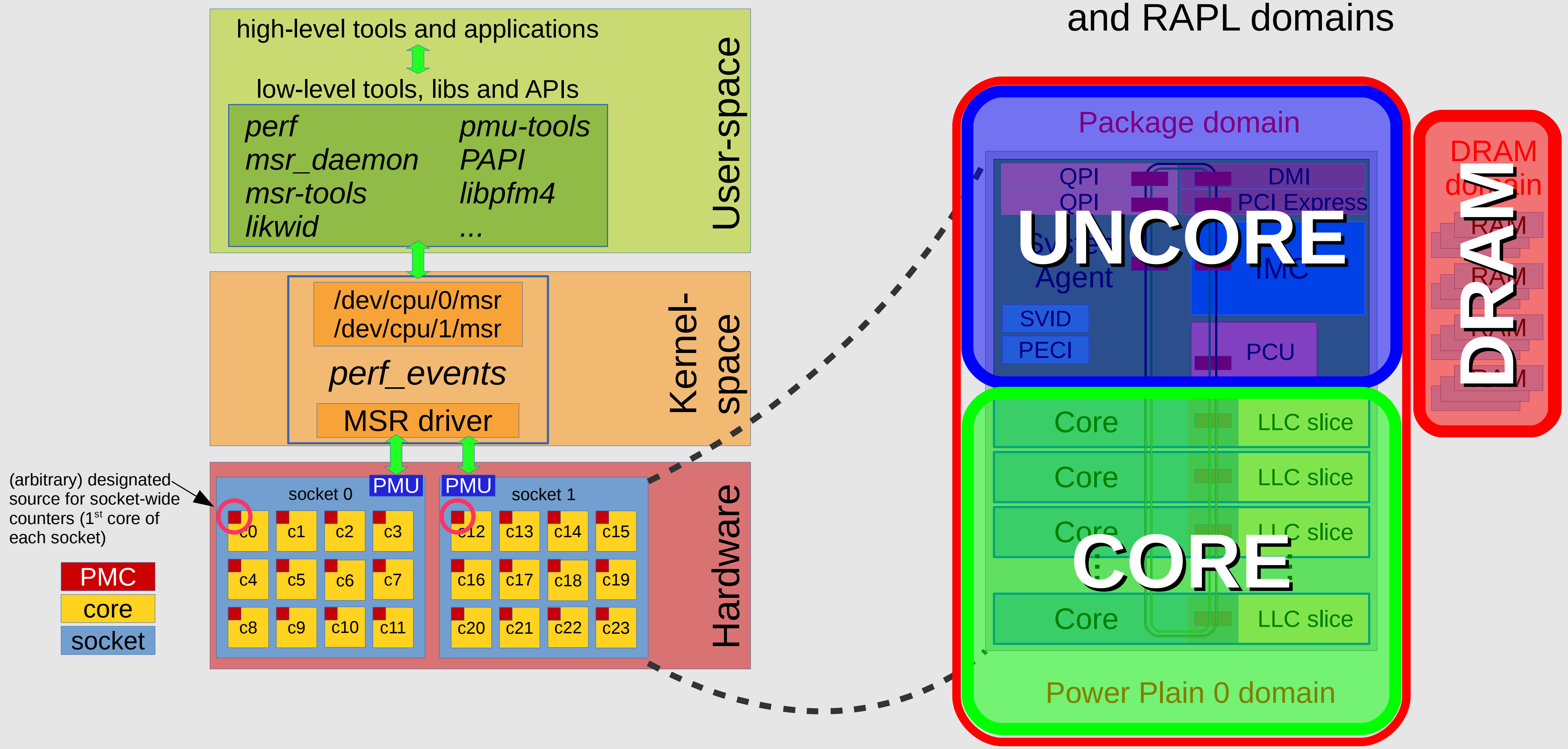
Power Management and Acquisition Software

- *Eurora Monitoring Framework* (Micrel Lab's sw developed for Eurora)
  - **msr-statd**: MSR/RAPL acquisition software
    (C program revised, improved, now linked to *hwloc)*
  - **gpu-statd**: GPU status info
    (Python script interfaced to NVidia Management Library (*nvml*))
- linux **perf** in order to access and collect performance counters (used as well for the top-down characterization)
- **cpufreq-utils** in order to modify the CPU frequency scaling governor and CPU frequencies (require super-user's privileges)
- various ad-hoc parsers and wrapper scripts (bash, awk, sed, perl, python, C)

Software stack, CPU sub-systems & RAPL domains

Software stack

Ivy Bridge sub-systems and RAPL domains



Testbed

C3E (CO.S.IN.T. Cloud and Cluster Environments)

- 2 Aurora Chassis
  - 6 Aurora computing nodes, 24 cores each
  - 4 Aurora computing nodes, 24 cores & 2x NVidia K20 each
- SLURM cluster: 1 VM hosting a dedicated masternode + 2 (physical) computing nodes

Each Aurora blade:

- Intel Xeon *Ivy Bridge* EP E5-2697 v2 @ 2.70 GHz, 12 cores, 30MB L3, in a dual-socket server configuration (*Romley*)
- 64GB RAM

Benchmarks

- 2 well-known benchmarks
  - HPL (CPU-bound app.)
  - HPCG (memory-bound app.)
- 2 real-world scientific applications
  - Quantum ESPRESSO (Palladium simulation)
  - LAMMPS (Lennard-Jones liquid benchmark)

from scheduler perspective, the code is a black-box

Benchmarking and Analysis

- benchmarks tuning runs
- energy profiling of well-known applications (HPL, HPCG) and real-world applications (QE, LAMMPS)
- comparison through top-down characterization
- comparison through performance counters
- comparing performance and energy efficiency changing frequency and problem size (looking for a trade-off for memory-bound applications)
- comparing different BLAS libraries (Netlib, ATLAS, OpenBLAS, MKL) using HPL through all the aforementioned methods
- HPL and frequency scaling coupled to GPU

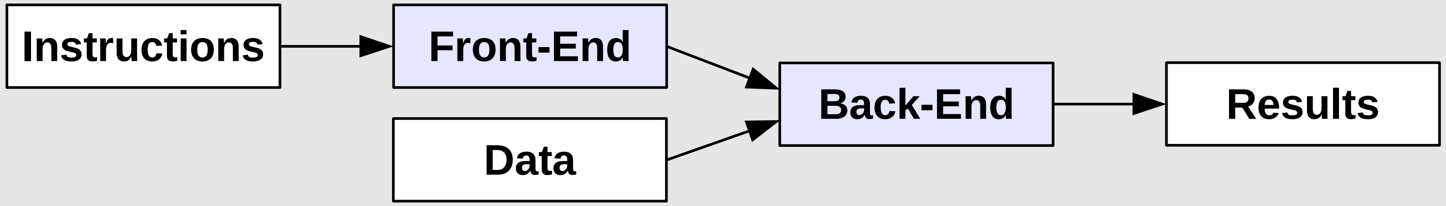
Benchmarking and Analysis

category	test	HPL				HPCG	QE	LAMMPS
		Netlib	ATLAS	OpenBLAS	MKL			
energy efficiency (RAPL)	basic analysis & tuning (*)	x	x	x	x	x	x	x
	frequency scaling	(**)	x	x	x	x	x	x
	problem size scaling	(**)	x	x	x	x		x
	frequency scaling + GPU				x			
	multinode (*)				x			
performance counters (*)	top-down analysis	x	x	x	x	x	x	x
	cache-miss, branch mispr.	x	x	x	x	x	x	x
	FLOPS from counters				x	x	x	x
	exp. cache-miss (***)	x	x	x	x			

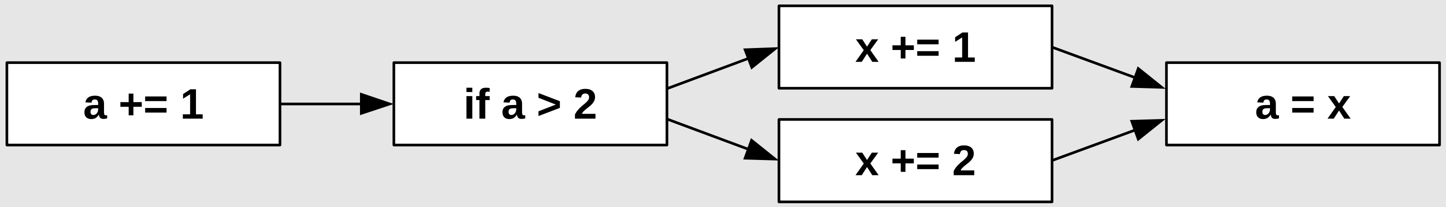
(*) with ondemand governor (automatic frequency scaling w/ Turbo Boost)
(**) only for memory size 1/4 and 1/8
(***) experimental tests using various combinations of cache-related performance events

Top-down characterization and performance counters analysis

- TOP-DOWN characterization of pipeline slots
  - stalled: *front-end bound*, *back-end bound*



  - not stalled: *bad speculation*, *retiring*



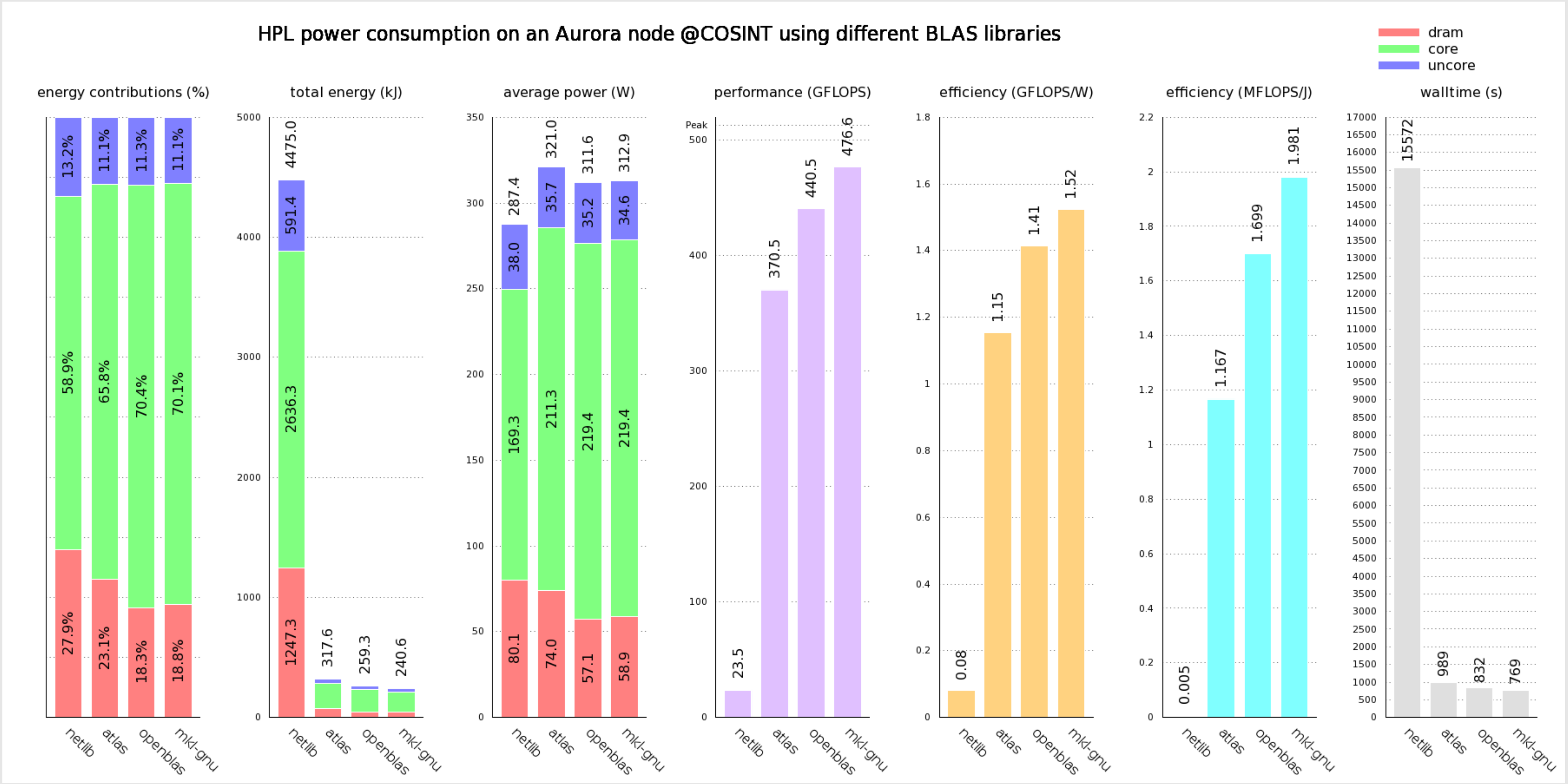
- performance counters analysis
  - *cache-miss ratio* (top-down BE-bound)
  - *branch-misprediction* (top-down bad-speculation)

Why comparing BLAS libraries?

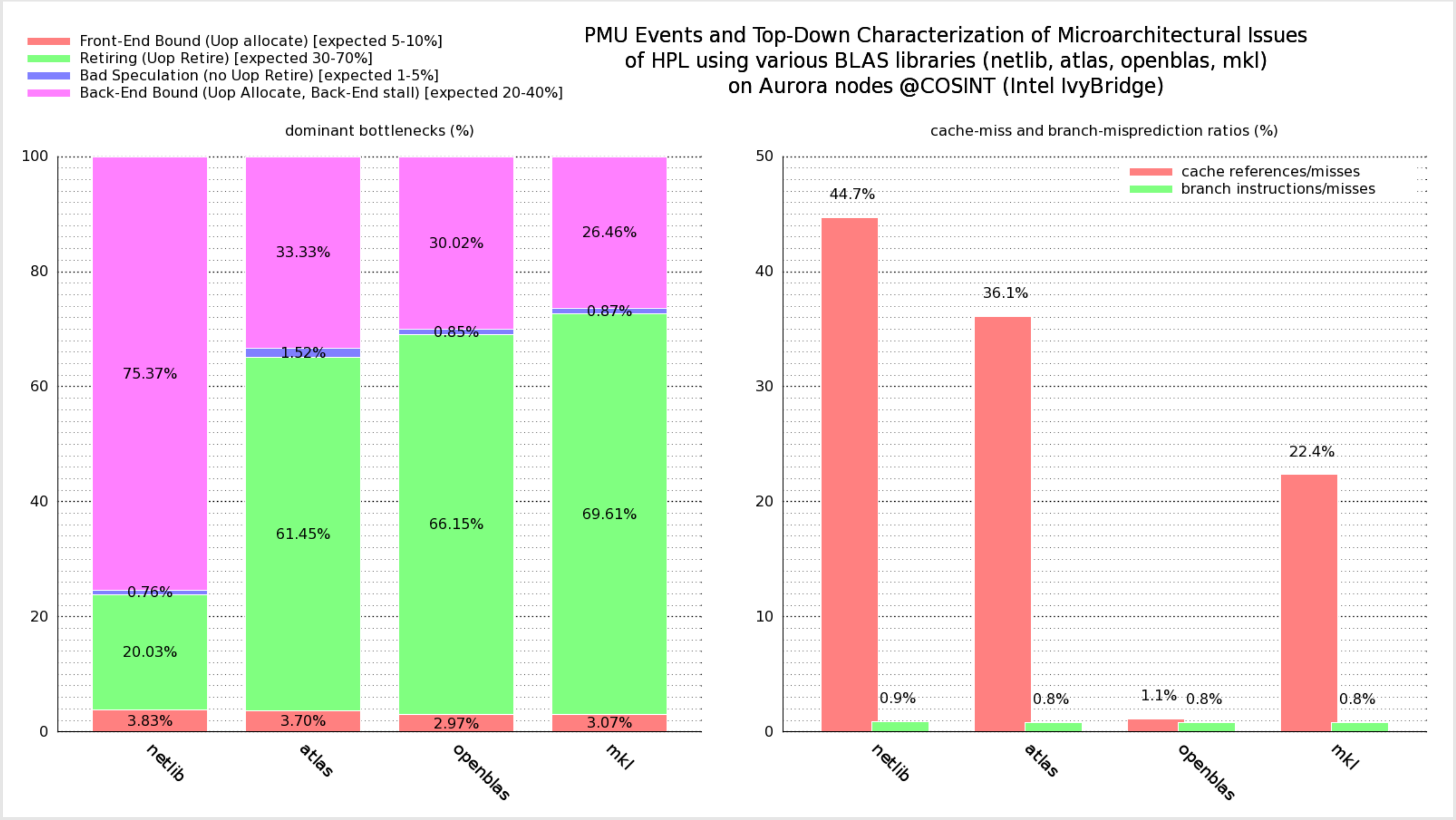
- several implementations available
  - NETLIB (reference library)
  - ATLAS (compile-time automagic optimization)
  - MKL (INTEL proprietary libraries)
  - OpenBLAS (free/open high-perf implementation)

- as the implementation changes, the achieved performance can be very different, energy efficiency must be different too (FLOPS/Watt - FLOPS/Joule)

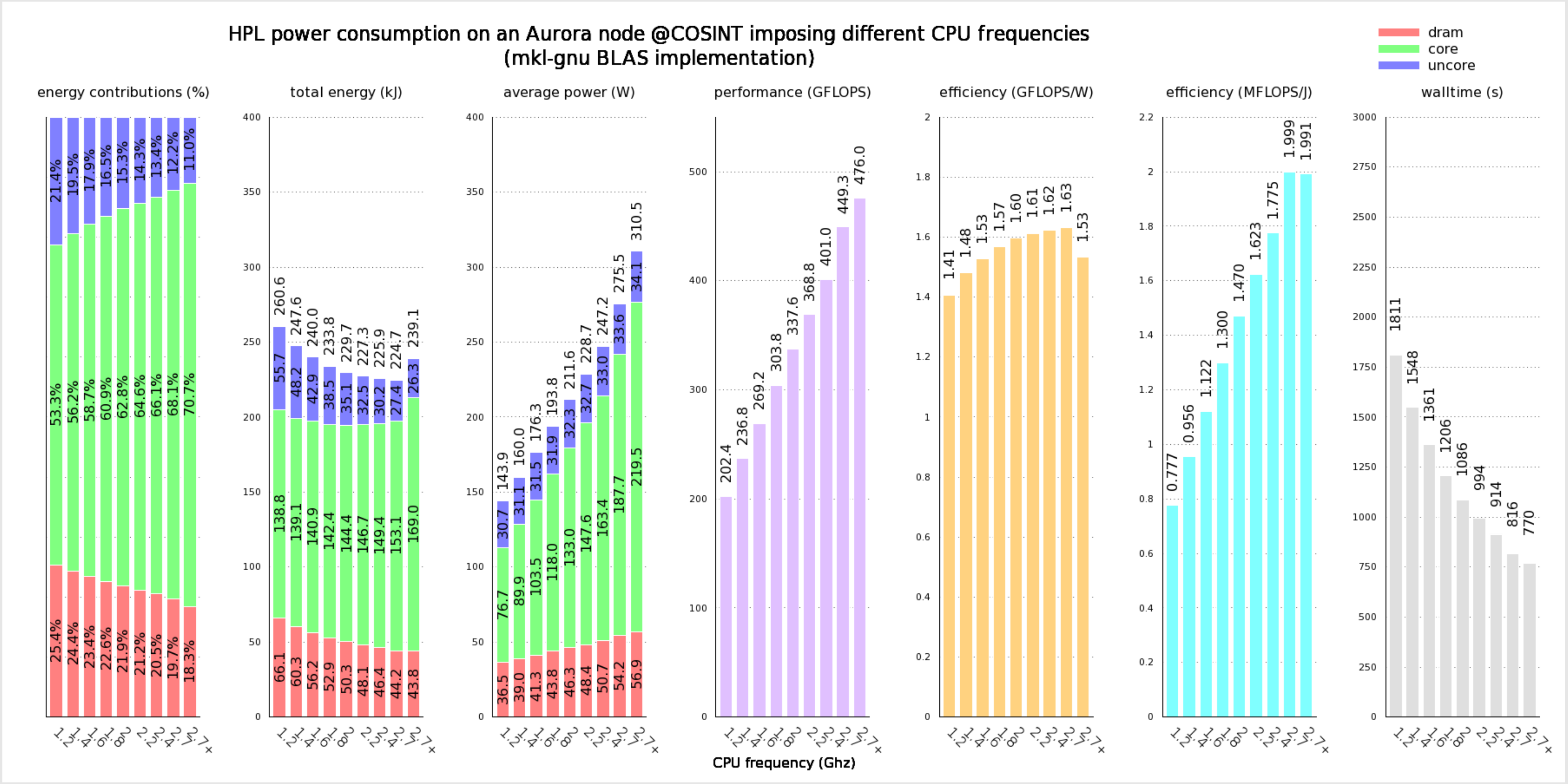
Comparing BLAS implementations: energy-efficiency



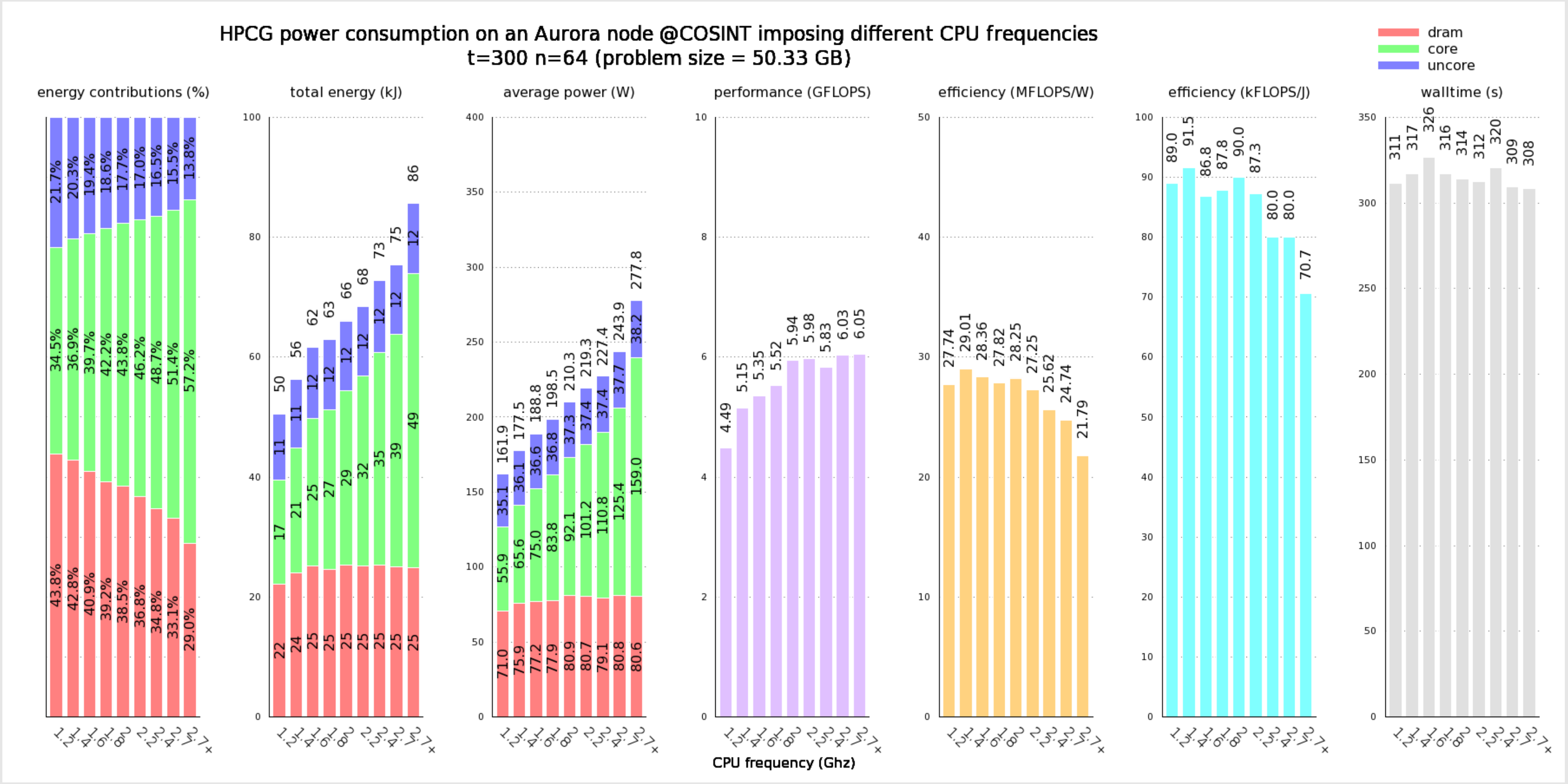
Comparing BLAS implementations: top-down and perf. counters



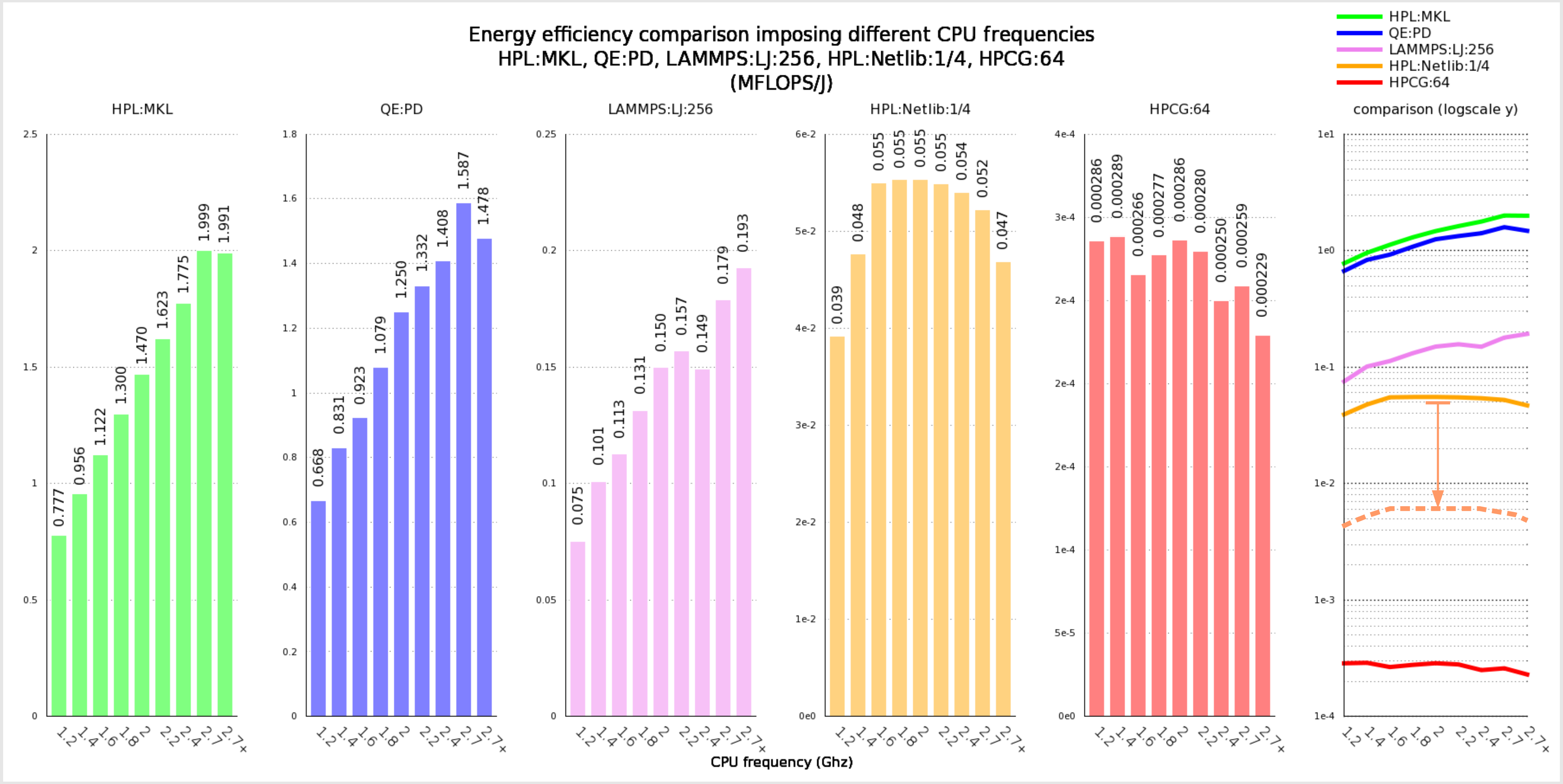
HPL+MKL: frequency scaling



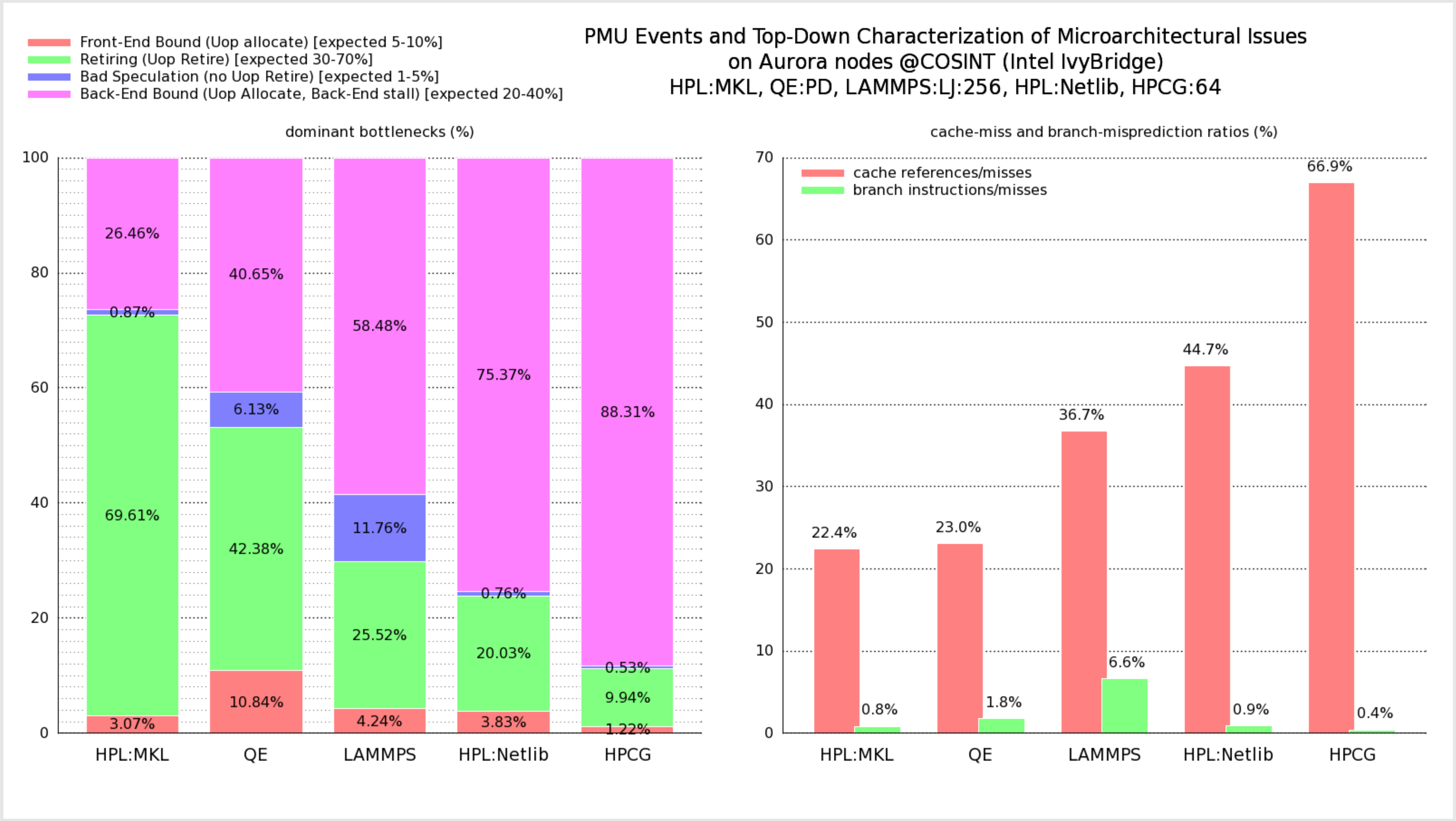
HPCG: frequency scaling



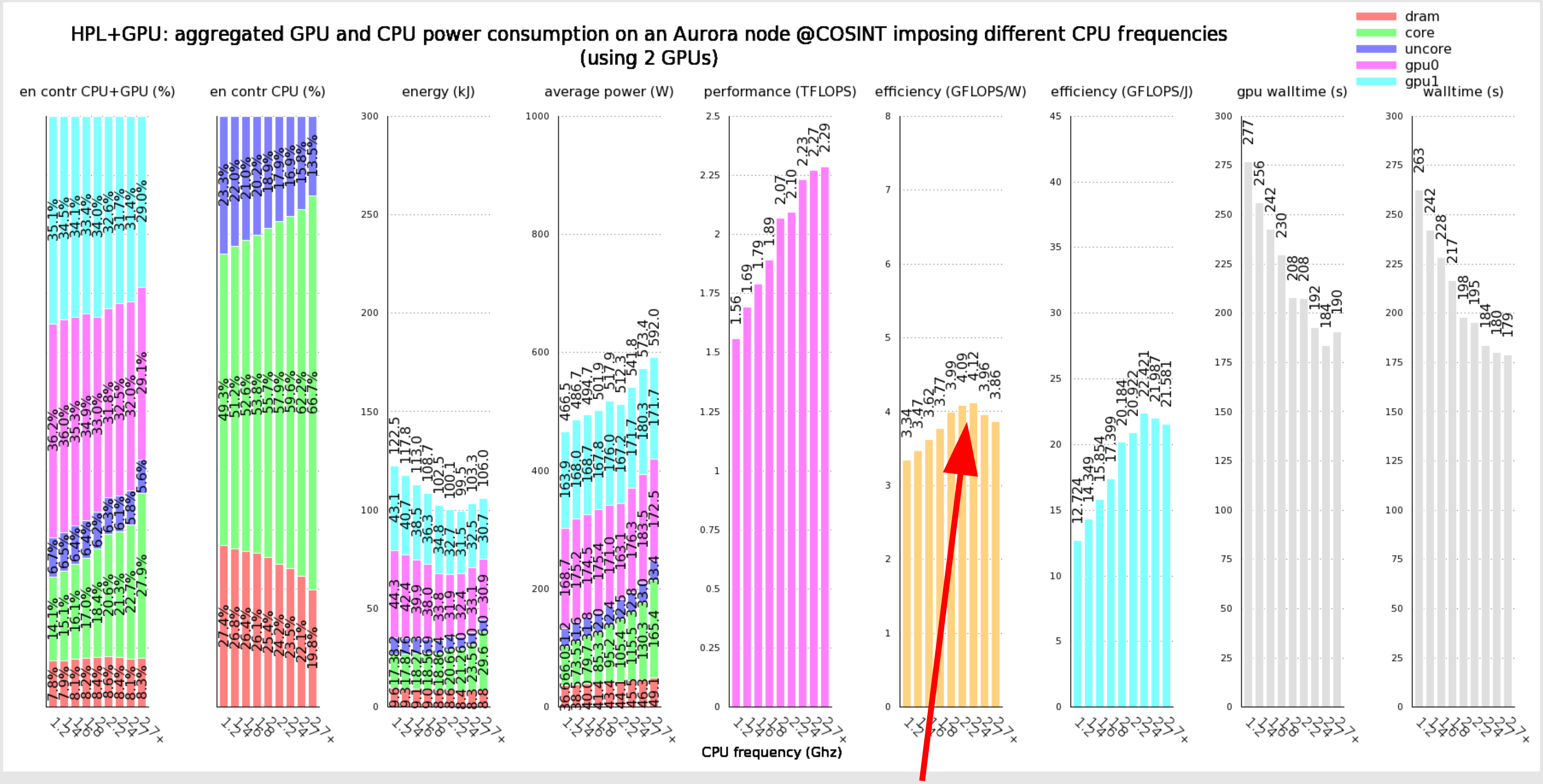
Comparison: energy efficiency w/ freq. scaling



Comparison: top-down and perf. counters



HPL+MKL: DVFS & GPU



4th position in the Green500!

Results summary

- **HPL** and **HPCG** represent **extremes**, **real-world** lies **in-between**
- **unoptimized** codes/libraries may lead to **catastrophic** results (both for performance and energy-efficiency)
- **OpenBLAS** and **MKL** appear to be almost **comparable**
- **energy-efficiency** for **memory-bound** applications **benefits** from **down-clocking**, useful under power capping constraints
- **GPU performance** unexpectedly **driven** by **CPU clock**

Master in High Performance Computing

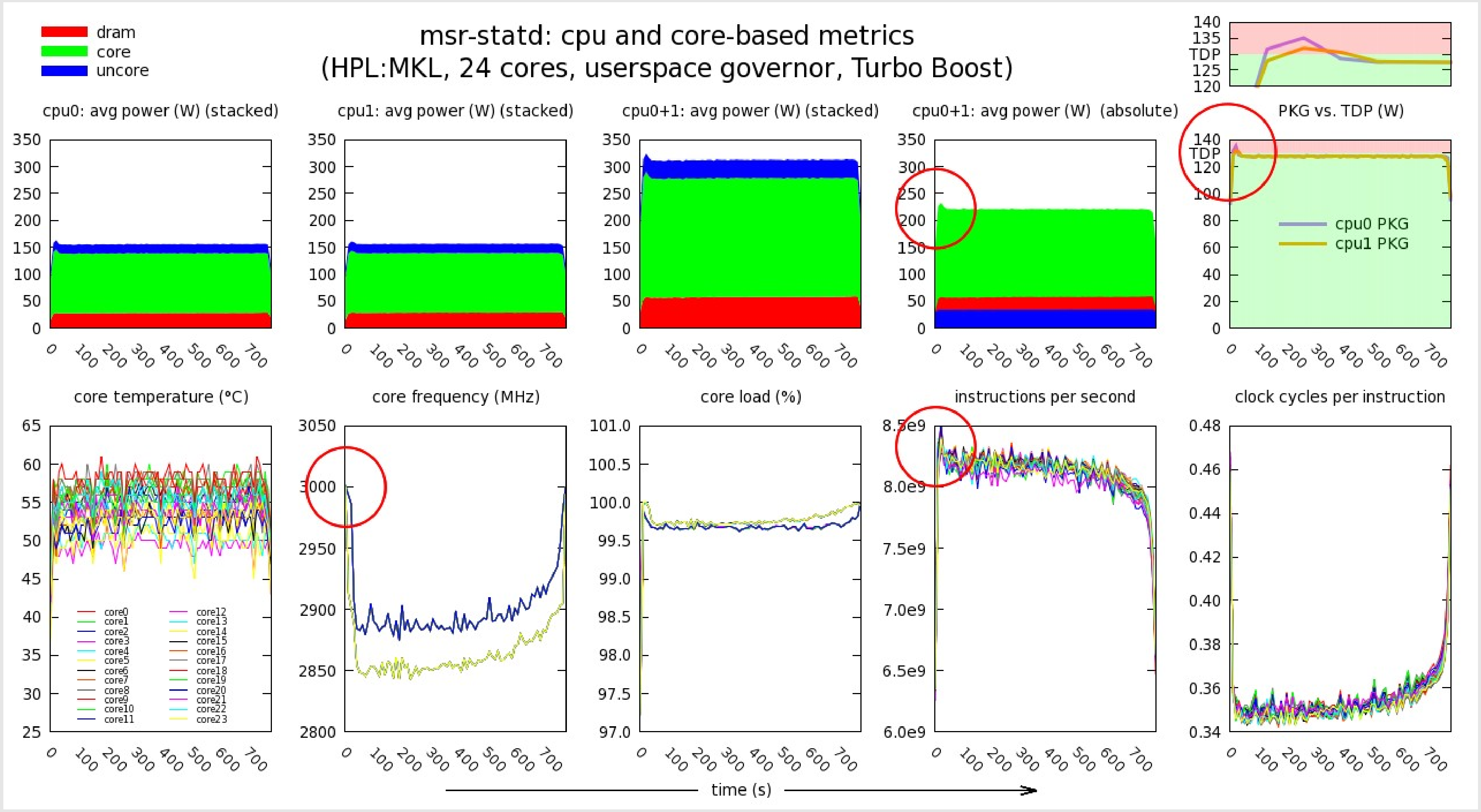
Observations

- **HPL** is **not** necessarily a **wise** metric **for choosing HW**
- **RAPL** and **performance counters** represent **powerful tools** for out-of-band/unattended profiling and monitoring (energy-aware scheduling)
- performance counters are **complex and difficult to handle** properly, several **hidden caveats** make them difficult to be widely exploited, further study is required

Future perspectives

- **better understand performance counters** reliability and profiling capabilities
- investigate **GPU/MIC** performance counters and DVFS effects on power consumption and energy-efficiency
- **integration** of out-of-band energy consumption monitoring **in production environment** (PBS/torque) with real-world usage, as well as **energy-aware scheduling** (Micrel Lab)
- **investigate** power profiling capabilities of **other devices**, platforms and infrastructures (overall and multi-node power consumption and energy efficiency)

Ignored data and possible further analysis



Thanks for your attention!

Any question?