Combining Uncore Frequency and Dynamic Power Capping to Improve Power Savings

Amina Guermouche

University of Bordeaux, *INRIA*
France

4^{th} Workshop on Resource Arbitration for Dynamic Runtimes (RADR)
June 03, 2022

EuroHPC
Joint Undertaking

Introduction and motivation

- Computational power increase

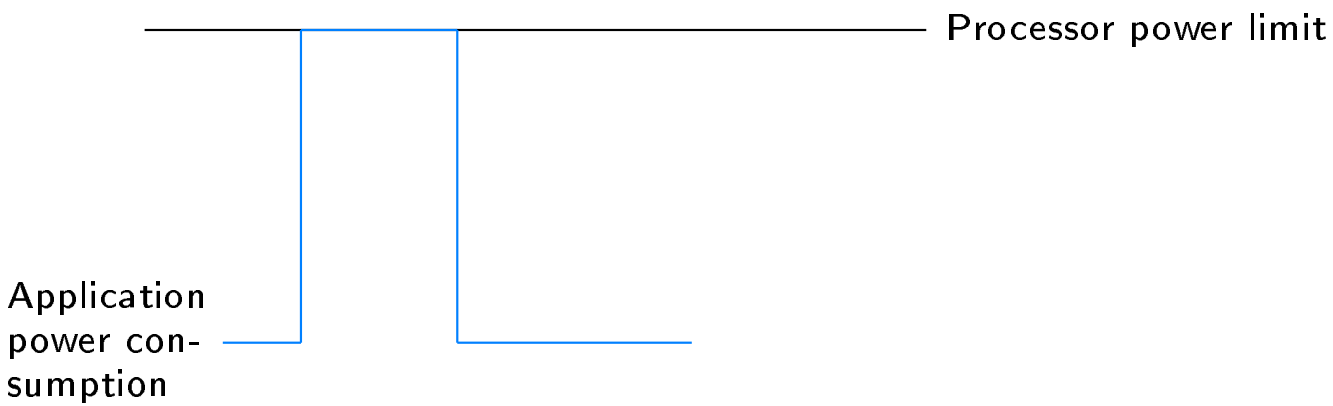
  Frontier 1.1 EFlop/s

- Large power consumption

  Frontier 21.1 MW
  - $\sim$ 12000 households

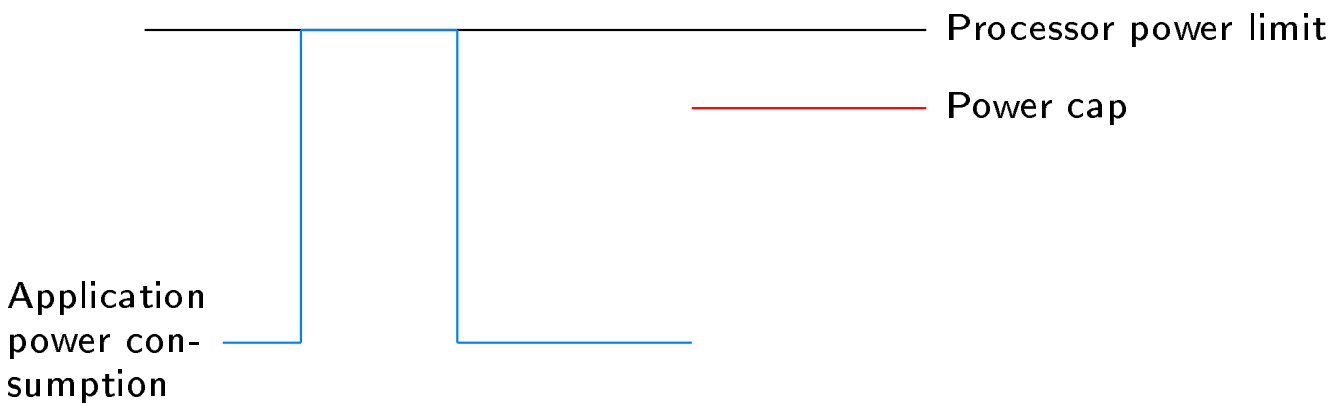
- DoE limit on power budget for exascale machines ($20-30$ MW)
  - Techniques to reduce power consumption (DVFS, UFS, ...)
  - Power capping

Power capping



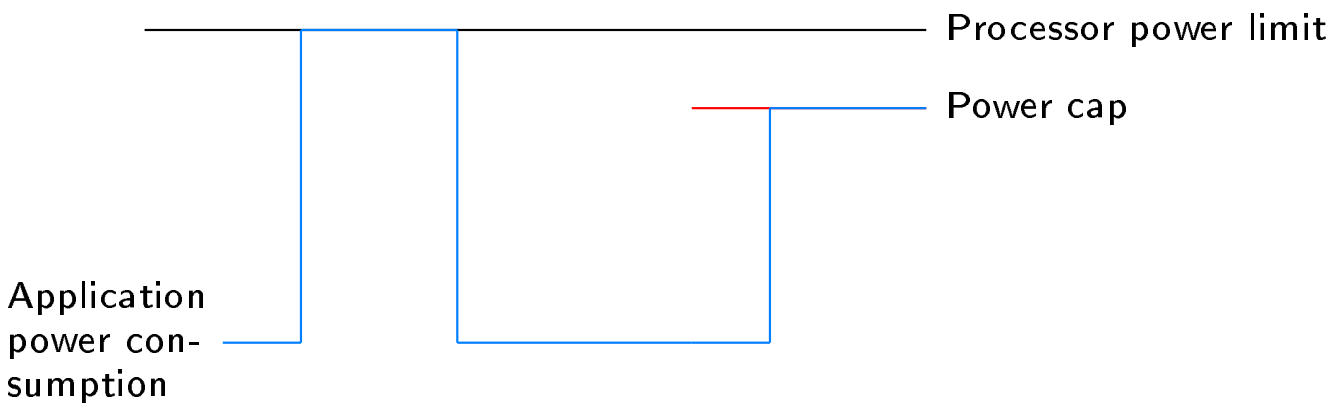
- A processor has a limited power budget (to avoid any damage)

Power capping



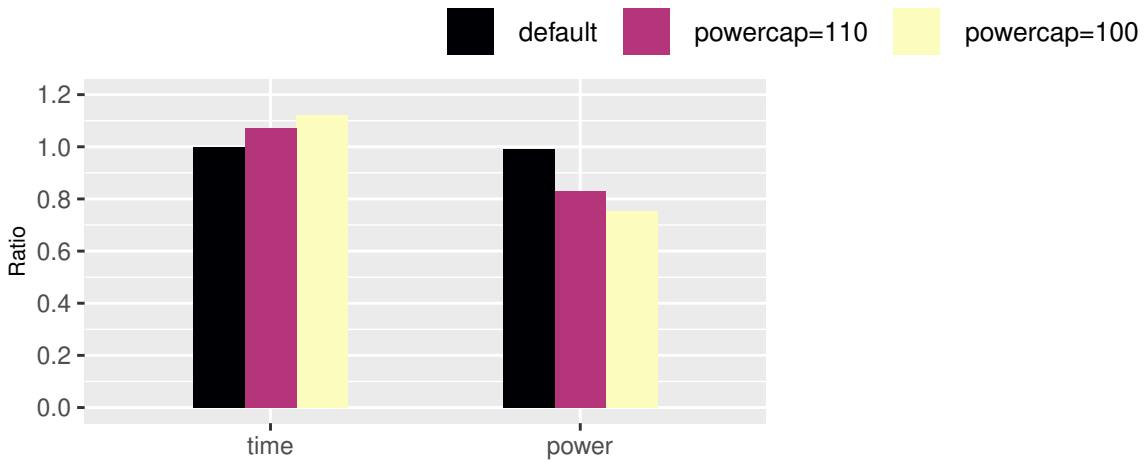
- A processor has a limited power budget (to avoid any damage)

Power capping



- A processor has a limited power budget (to avoid any damage)
- One can reduce the power budget:
  - Per processor
  - Per DRAM (not available on the used platform)

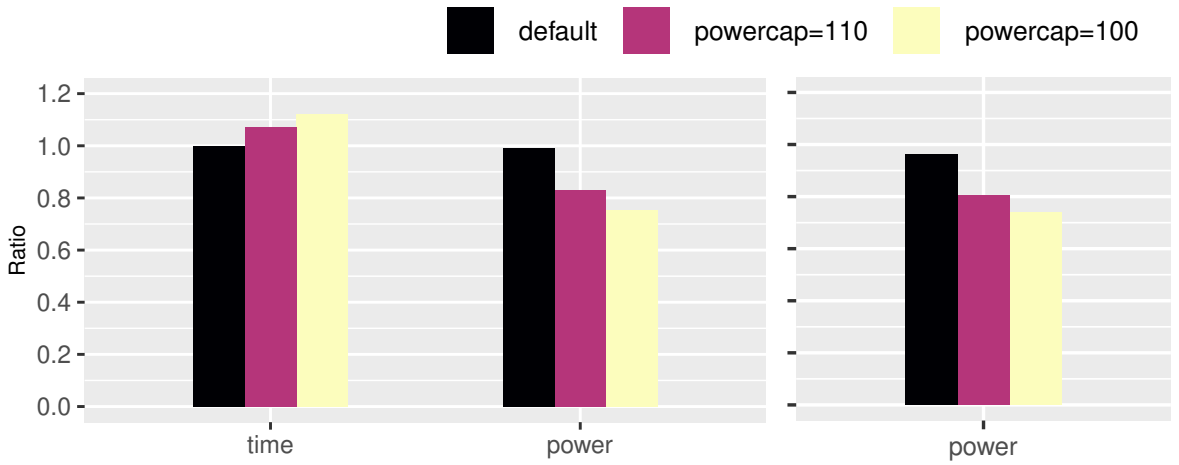
Impact of power capping on application performance and power consumption



Impact of power capping on the power consumption and execution time of CG (from the NAS Parallel Benchmarks) on a 4x16 cores Intel Xeon Gold. Default budget is 125 W.

- Applying power cap throughout the execution of an application introduces an overhead

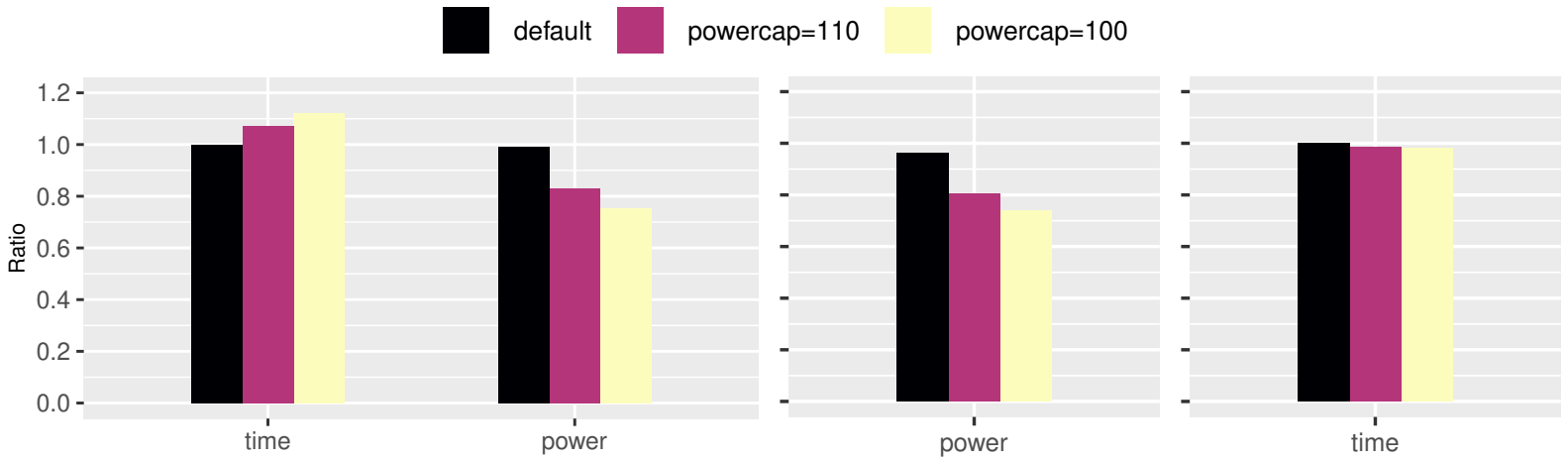
Impact of power capping on application performance and power consumption



Impact of power capping on the power consumption and execution time of CG (from the NAS Parallel Benchmarks) on a 4x16 cores Intel Xeon Gold. Default budget is 125 W.

- Applying power cap throughout the execution of an application introduces an overhead
- Applying power capping only on the beginning of the application provides power savings

Impact of power capping on application performance and power consumption



Impact of power capping on the power consumption and execution time of CG (from the NAS Parallel Benchmarks) on a 4x16 cores Intel Xeon Gold. Default budget is 125 W.

- Applying power cap throughout the execution of an application introduces an overhead
- Applying power capping only on the beginning of the application provides power savings with no impact on performance

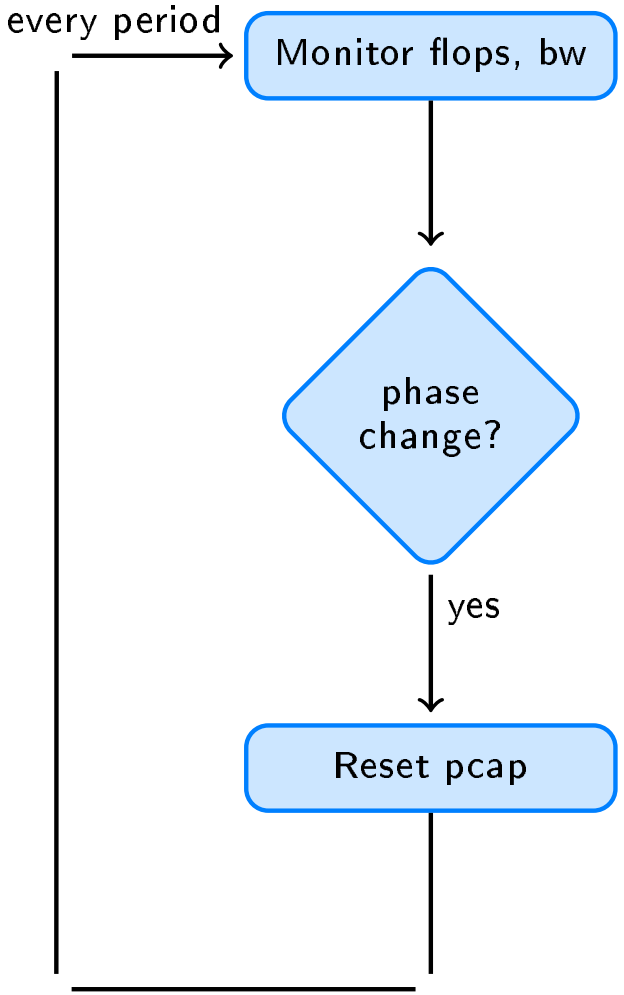
Outline

Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown

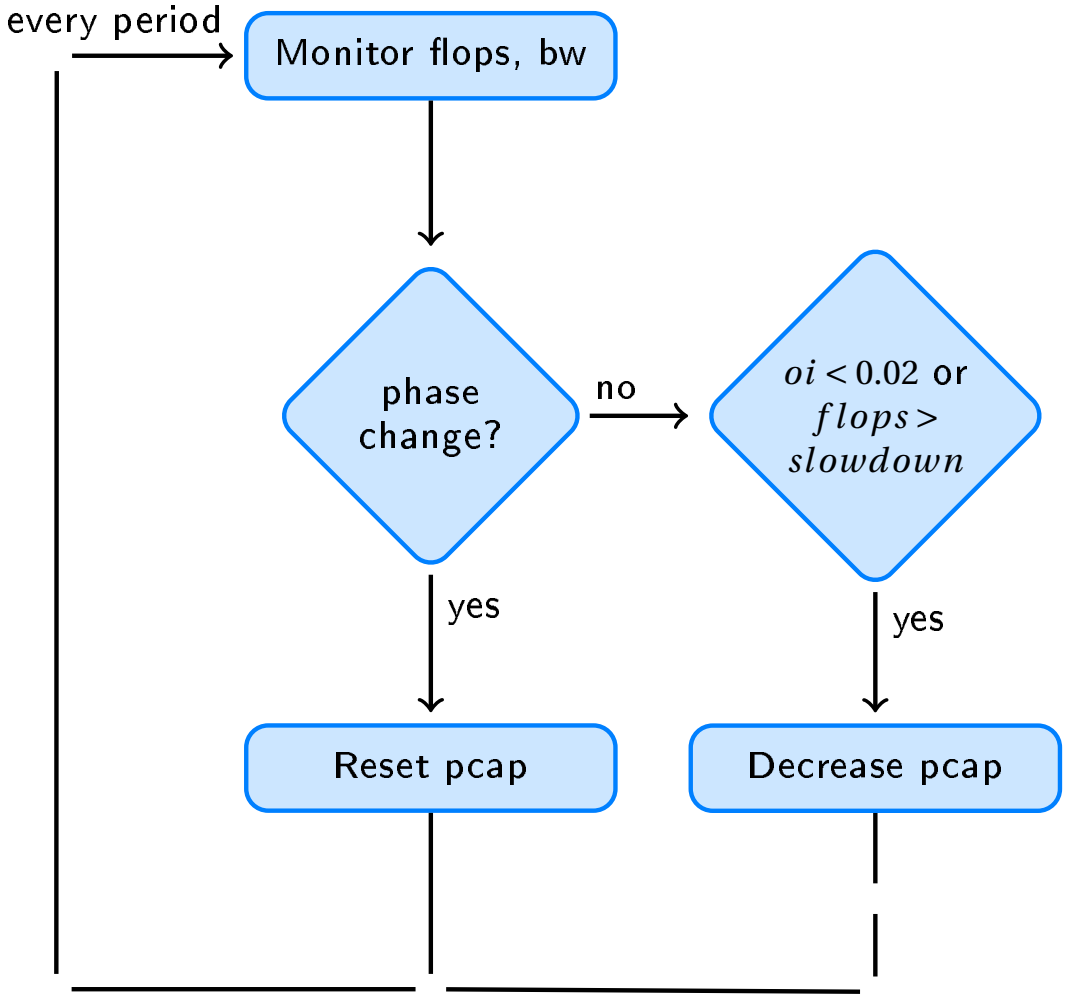
Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown



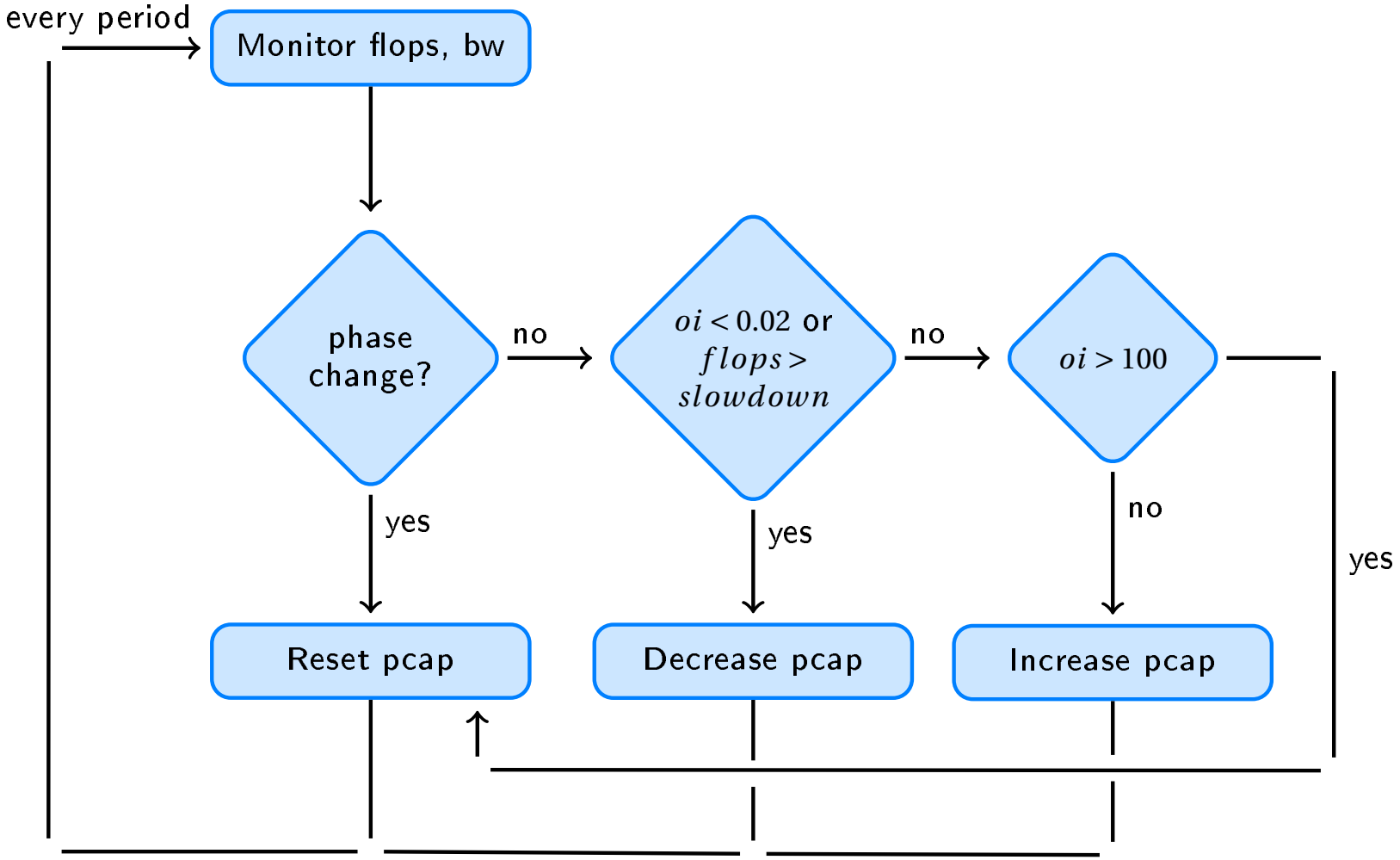
Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown



Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown



Dynamic power capping

- Basic algorithm:
  - Decrease power cap as long as the flops are within the tolerated slowdown (increase otherwise)
  - Reset the power cap whenever when the behavior of the application changes

- Goals:
  - Respect the user-defined tolerated slowdown
  - ⚠ We may not be able to save energy (since power capping impacts performance). As a consequence, the goal is to save power without impact on energy consumption.

Uncore frequency scaling (UFS)

- Frequency of the L3 cache, the memory controllers, ...
- The default UFS does not always adapt to the application needs

Uncore frequency scaling (UFS)

- Frequency of the L3 cache, the memory controllers, ...
- The default UFS does not always adapt to the application needs

→ DUF: Dynamic uncore frequency scaling

  - Adapts to the application characteristics (computations, memory accesses)
  - Allows for a user-defined tolerated slowdown
  - Improvements compared to the default UFS:

Uncore frequency scaling (UFS)

- Frequency of the L3 cache, the memory controllers, ...
- The default UFS does not always adapt to the application needs

→ DUF: Dynamic uncore frequency scaling

- Adapts to the application characteristics (computations, memory accesses)
- Allows for a user-defined tolerated slowdown
- Improvements compared to the default UFS:

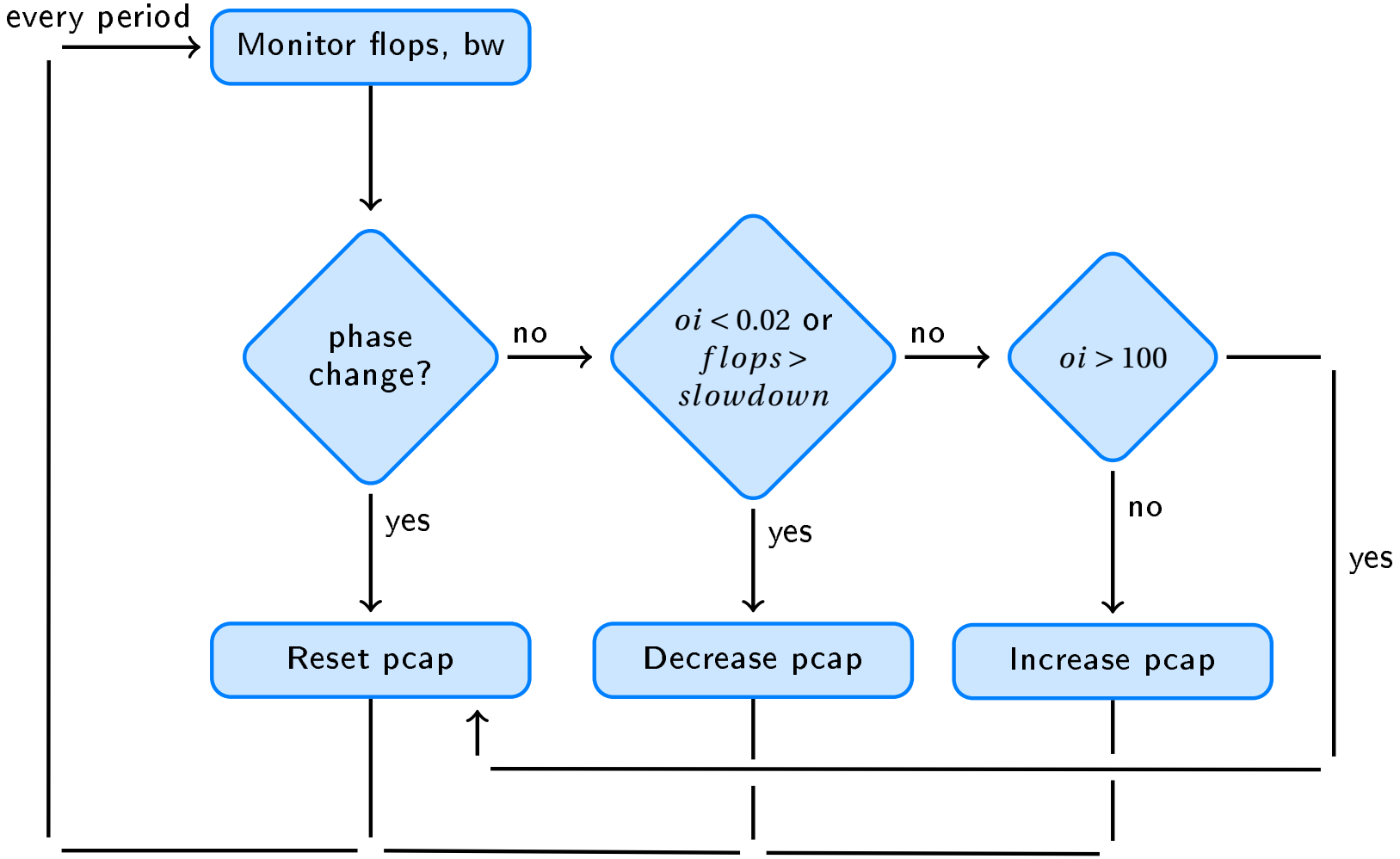
→ Power savings: by up to **15.6 %** with **no performance loss**

→ Power savings: by up to **7.46 %** with **less than 5 % performance loss**

→ Performance improvement: by up to **11.90 %** under power capping

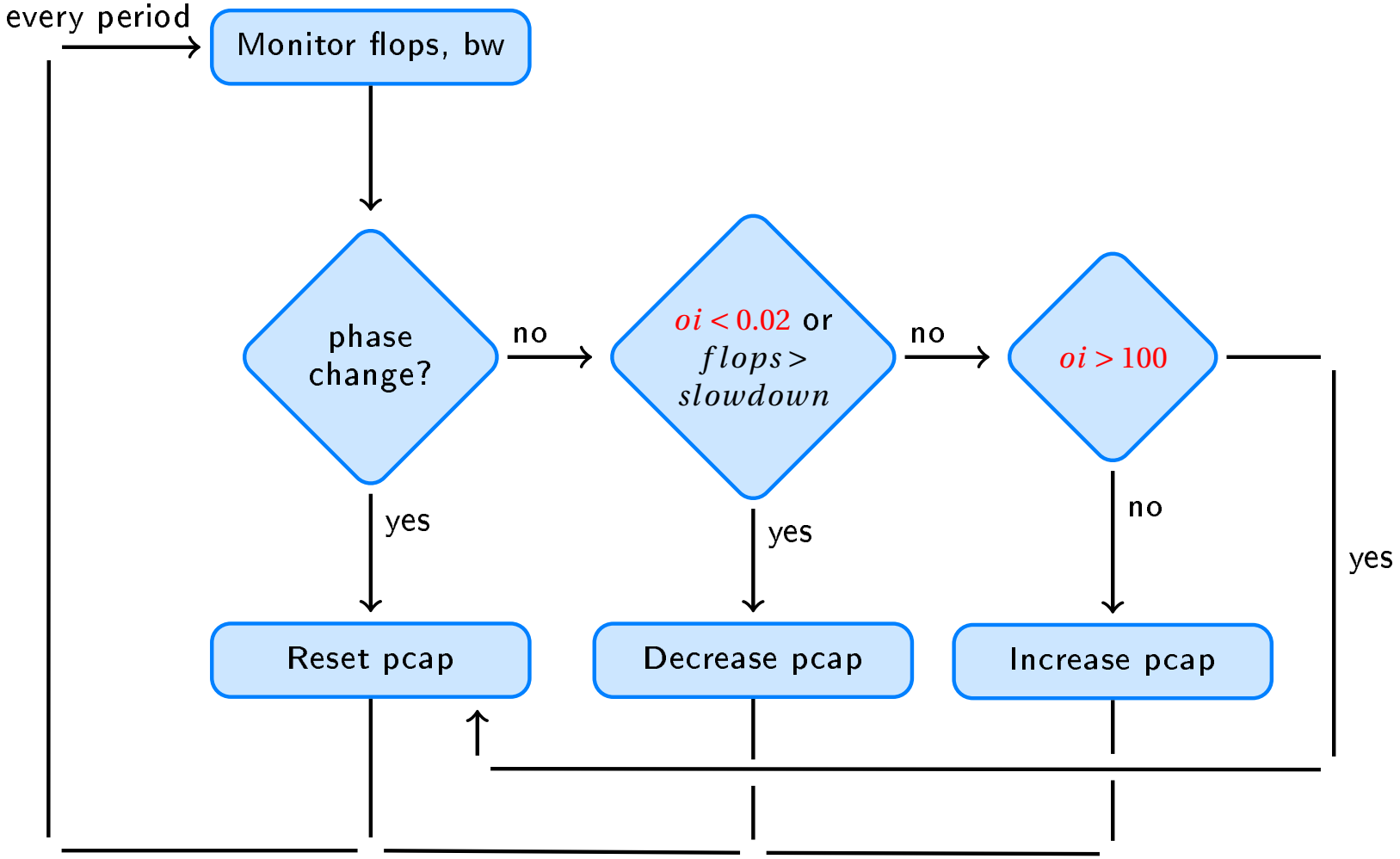
Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown



Dynamic power capping (DUFP) algorithm

- Adapt the power cap to the application needs
- Handle a user-defined tolerated slowdown



Outline

Target architecture and measurement framework

- Grid'5000 Grenoble Yeti: Intel Xeon Gold 6130 (Skylake)
  - 4 Processors
  - 16 cores/ Processor
  - Default power cap 125 W

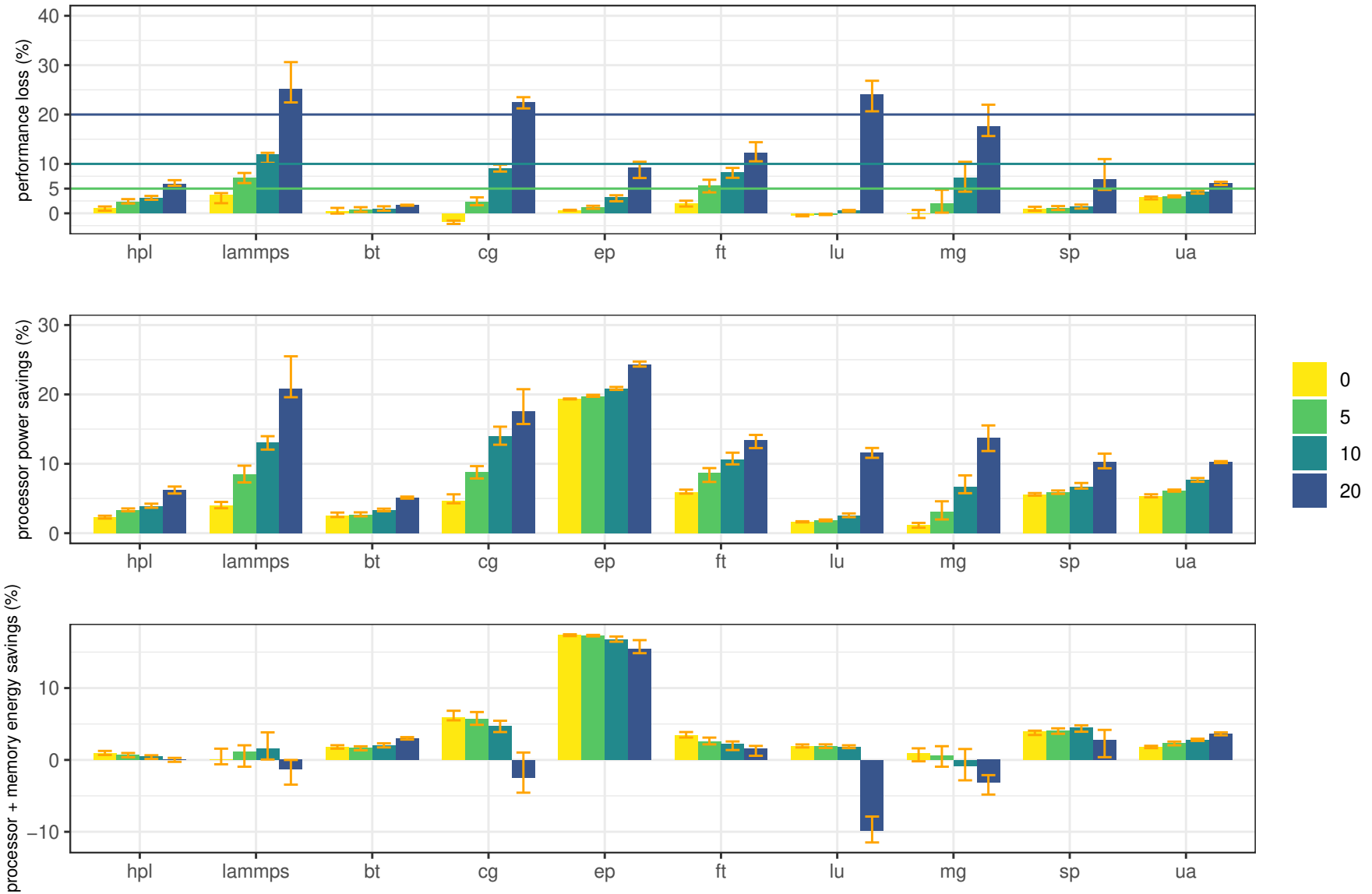
- Measurement framework:
  - PAPI library for all measurements (FLOPS/s, memory bandwidth, processor power consumption, memory power consumption)
  - Powercap library to set the power cap

Target applications and configurations

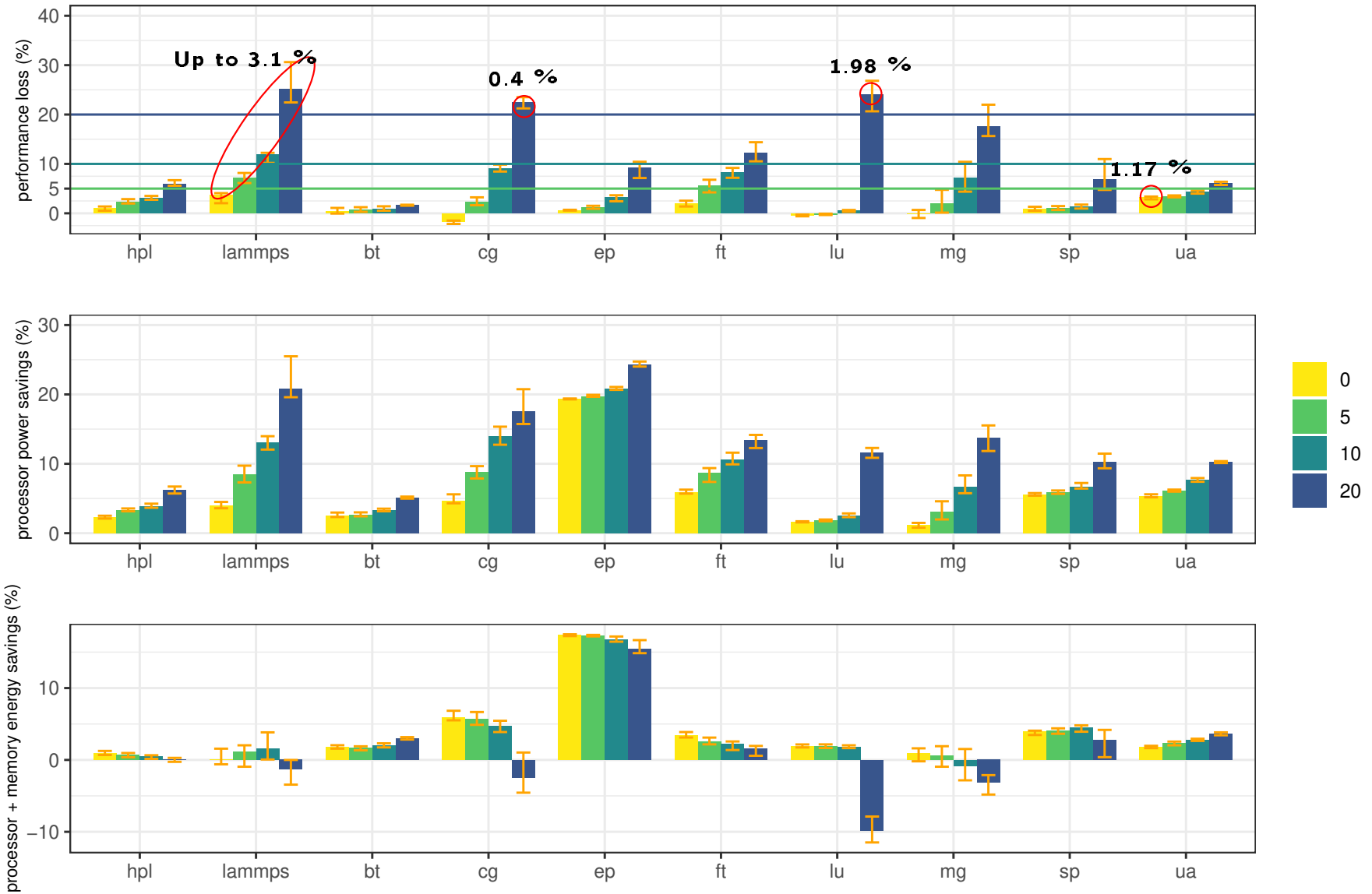
- Target applications
  - Nas Parallel Benchmarks BT, CG, EP, FT, LU, MG, SP, UA
  - HPL
  - lammps

- Configurations:
  - Period = 200ms
  - Tolerated slowdown: 0 %, 5 %, 10 % and 20 %
  - Measurement error: 2 %
  - Uncore frequency scaling is also handled

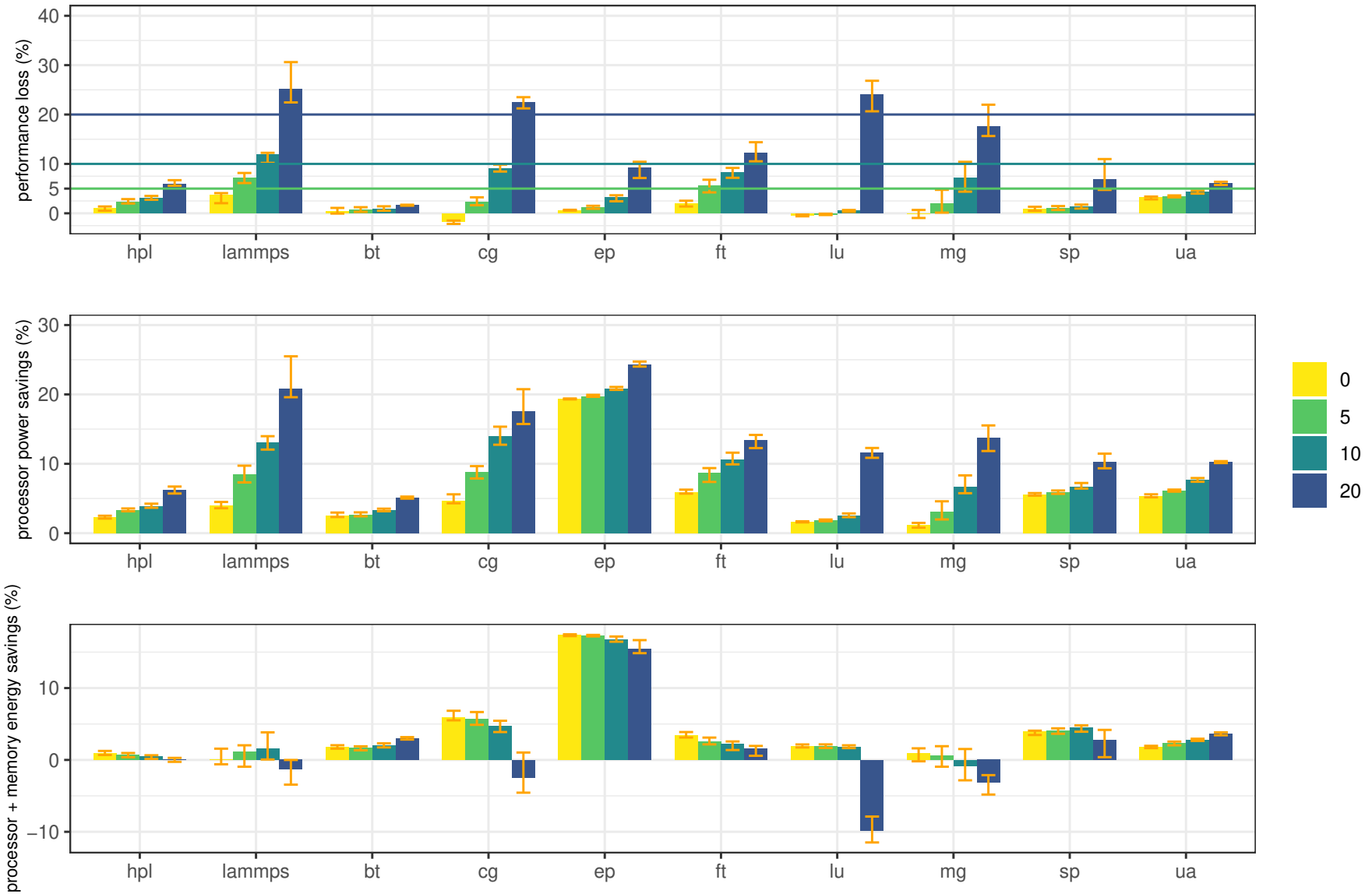
Impact on execution time, power and energy consumption



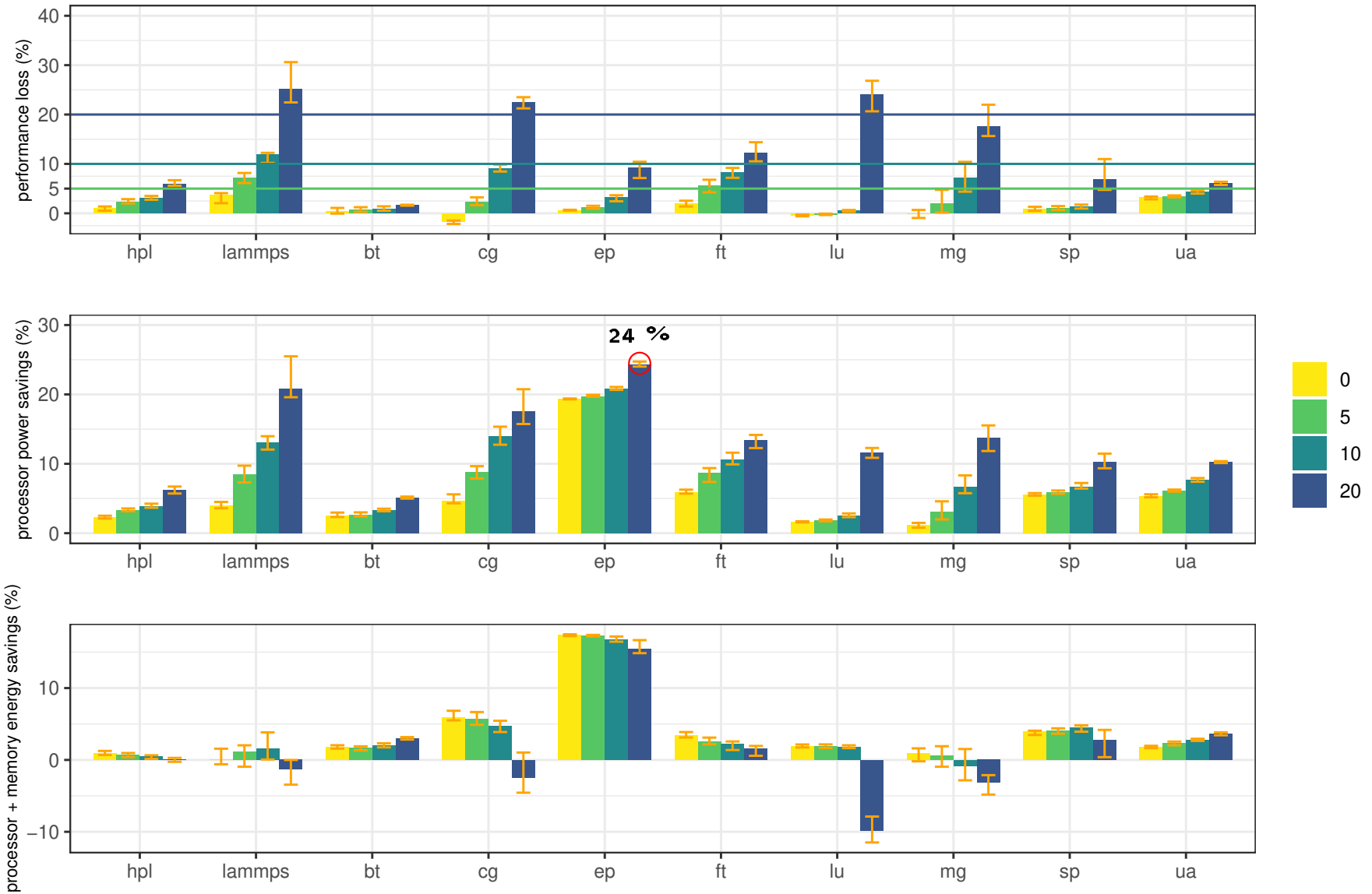
Impact on execution time, power and energy consumption



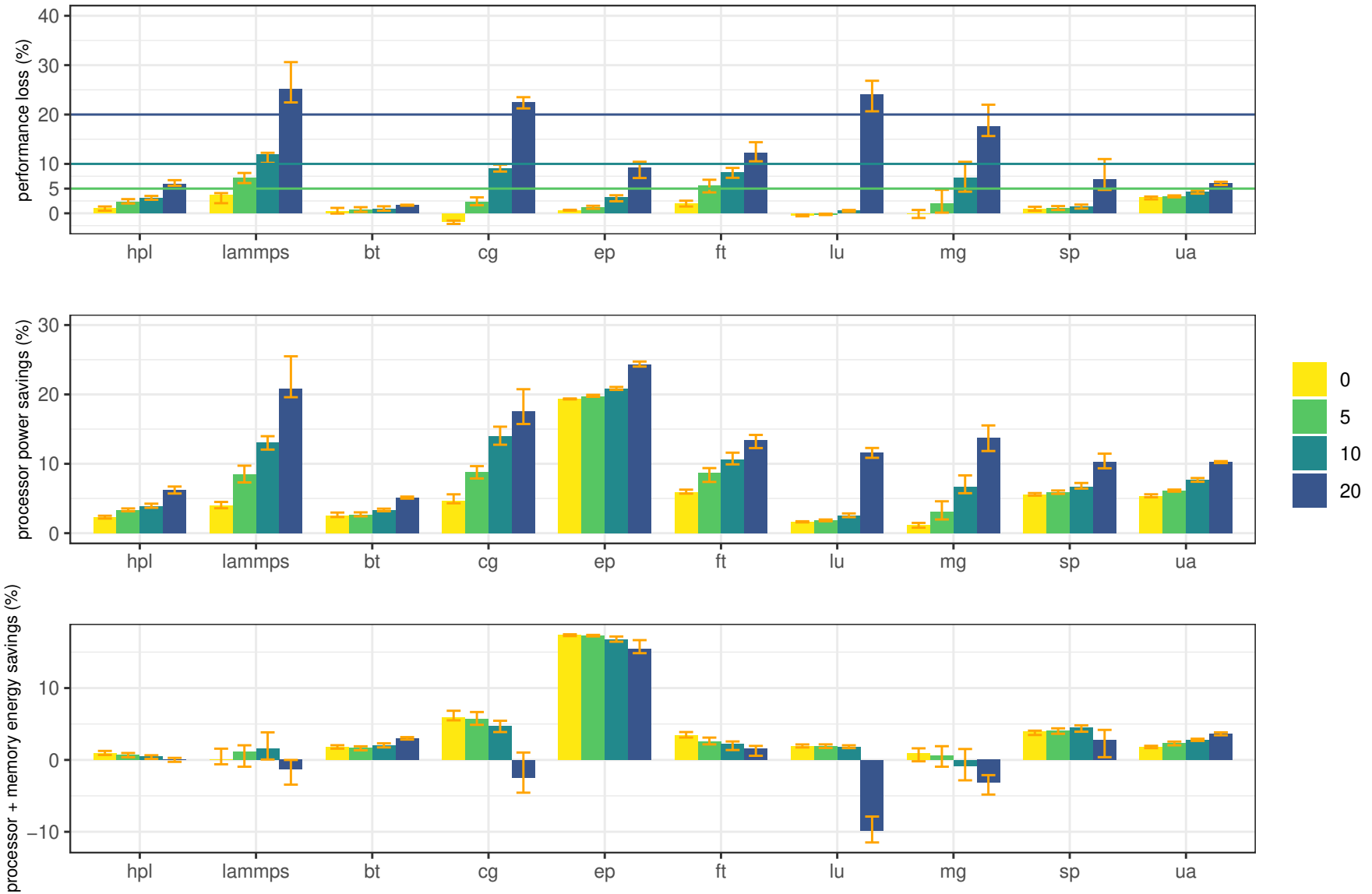
Impact on execution time, power and energy consumption



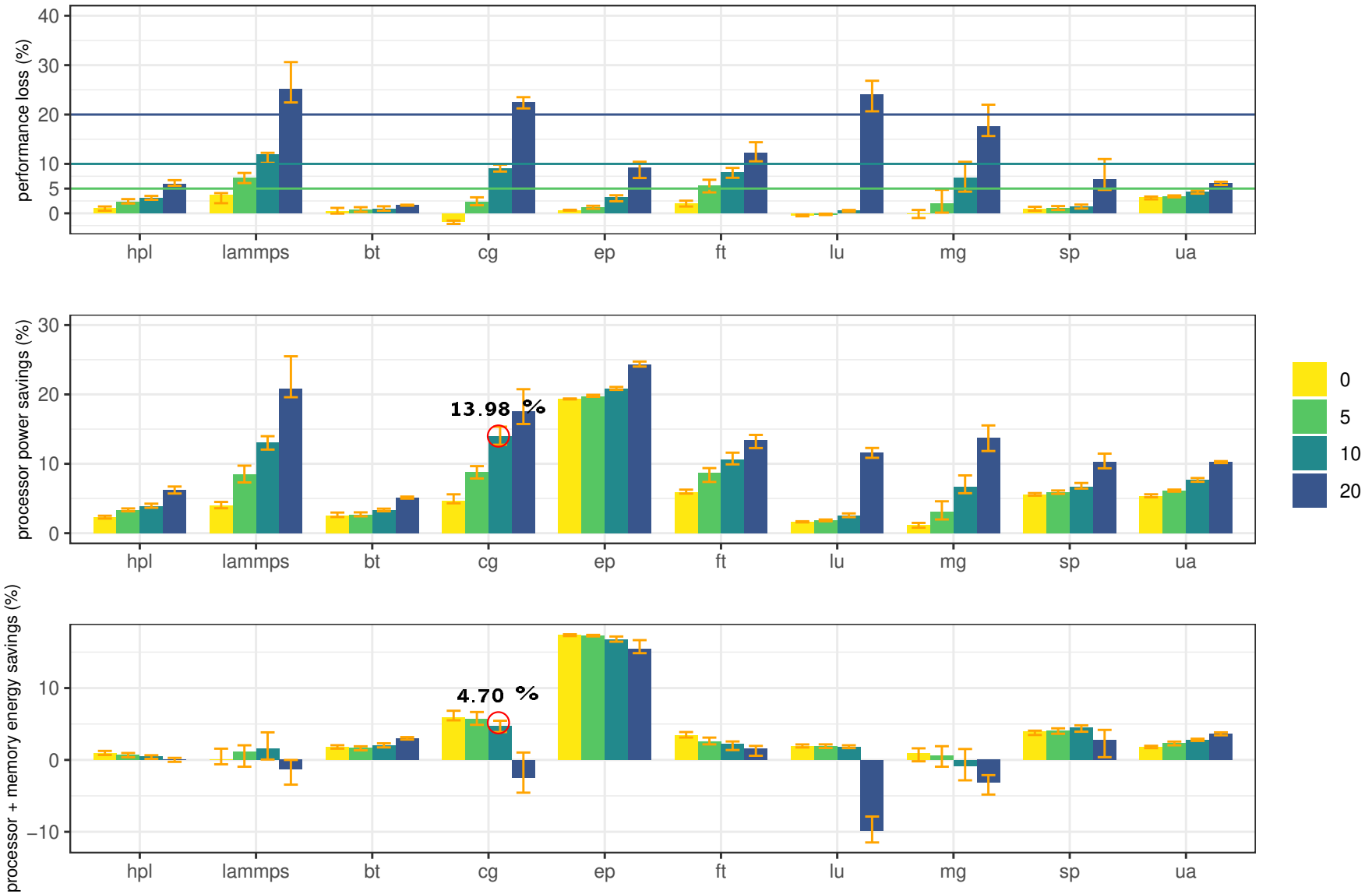
Impact on execution time, power and energy consumption



Impact on execution time, power and energy consumption



Impact on execution time, power and energy consumption



Impact on execution time, power and energy consumption

Impact on execution time, power and energy consumption

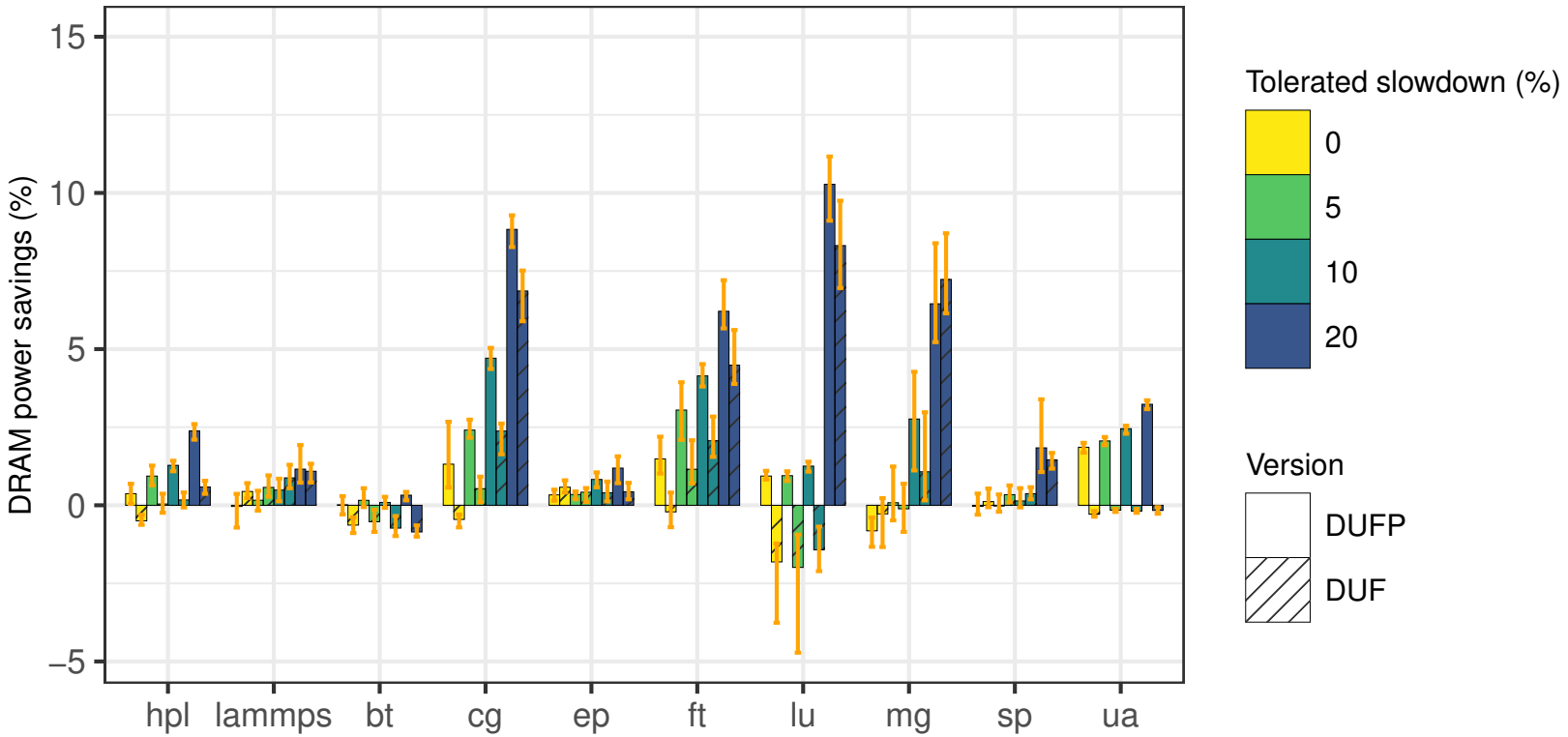
Impact on execution time, power and energy consumption

Impact on execution time, power and energy consumption

Impact on execution time, power and energy consumption

Impact on execution time, power and energy consumption

Impact on memory power consumption



Summary of the results

- Up to 5.56 % power savings with less than 1 % performance loss (0.85 % slowdown)
- Up to 8.76 % power savings with less than 5 % performance loss (2.32 % slowdown)
- Best energy savings with 0 % tolerated slowdown
- Best power savings with no energy loss with 10 % tolerated slowdown
- Additional power savings with DUFP compared to DUF

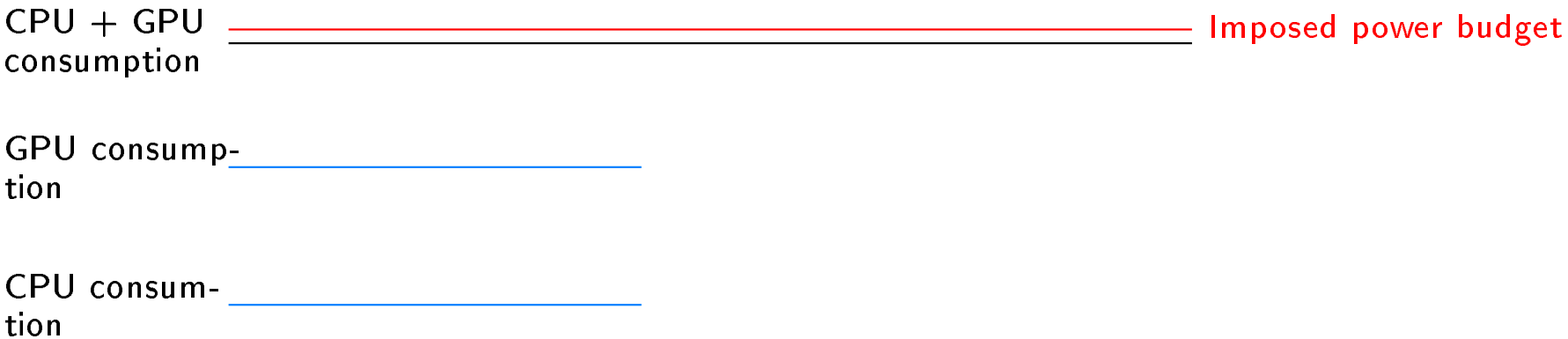
Related work

- Uncore frequency + Dynamic power capping
  - Reinforcement learning to get the best energy consumption
    - No user-defined tolerated slowdown

- Dynamic power capping
  - DNPC: Dynamic power capping with user-defined tolerated slowdown
    - Performance model
  - CoPPer: which power cap to apply to meet user-defined performance?
    - Performance model
    - Application instrumentation

Conclusion and future work

- Conclusion: Using power capping, can we reduce the power consumed by an application with a limited impact on its energy consumption?
  - Power savings with no energy loss for all applications (up to 10 % tolerated slowdown)
  - Tolerated slowdown respected for most configurations
- Future work
  - Manage CPU frequency
  - Use learning techniques to get the best configuration
  - Use dynamic power capping for dynamic CPU/GPU scheduling



Conclusion and future work

- Conclusion: Using power capping, can we reduce the power consumed by an application with a limited impact on its energy consumption?
  - Power savings with no energy loss for all applications (up to 10 % tolerated slowdown)
  - Tolerated slowdown respected for most configurations
- Future work
  - Manage CPU frequency
  - Use learning techniques to get the best configuration
  - Use dynamic power capping for dynamic CPU/GPU scheduling