Compas'2015 — 2015-07-03

Software power estimation on multicore systems *.

Maxime Colmant, *Romain Rouvoy, Lionel Seinturier.*

* *Published article: Colmant (M.), Kurpicz (M.), Huertas (L.), Rouvoy (R.), Felber (P.) et Sobe (A.). – Process-level Power Estimation in VM-based Systems. – In European Conference on Computer Systems (EuroSys), Bordeaux, France, avril 2015*

ADEME
Agence de l'Environnement et de la Maîtrise de l'Energie

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Université de Lille 1 SCIENCES ET TECHNOLOGIES

PARA DIME

IIUN Institut d'informatique Université de Neuchâtel

Introduction

- Increasing usage of ICT devices
  - 1.43 GtCO2 in 2020 (6%) [1]
- Complexity of modern processors
  - Limited power-aware interfaces [2, 3]
- Software power estimation, a cornerstone
  - Identify the largest power consumers, make informed decisions
  - Architecture-agnostic solution is needed

[1] The Climate Group. *"SMART 2020: Enabling the low carbon economy in the information age"*
[2] Marcus Hähnel, et al. *"Measuring energy consumption for short code paths using RAPL"*
[3] Yan Zhai, et al. *"Happy: Hyperthread-aware power profiling dynamically"*

Motivation

- In general, performance > energy efficiency
- ICT has a huge impact on the world CO2 emissions
- Main power consumer: processor (increasingly complex)
- Multi-core CPUs are widely used nowadays
- On the hardware side (e.g. SMT, DVFS, C-states)
- On the software side?

- **Software power efficiency: can play a deterministic role!**

Approaches

- Hardware-centric approach
  - Coarse-grained
  - Expensive
- Software-centric approach
  - Fine-grained
  - Awkward

Software-centric approach

- Needs
  - Efficient and accurate power models
  - Trade-off between accuracy / overhead
- Existing solutions
  - Specific softwares and architectures [1, 2, 3, 4]
  - As an example, Intel with RAPL [4, 5]
- Our goal
  - Provide an architecture-agnostic solution
  - Identify green patterns as methodological guidelines

[1] Ramon Bertran, et al. *"Decomposable and responsive power models for multicore processors using performance counters"*
[2] William Lloyd Bircher, et al. *"Complete system power estimation: A trickle-down approach based on performance events"*
[3] Vasileios Spiliopoulos, et al. *"Power-sleuth: A tool for investigating your program's power behaviour"*
[4] Yan Zhai, et al. *"Happy: Hyperthread-aware power profiling dynamically"*
[5] Marcus Hähnel, et al. *"Measuring energy consumption for short code paths using RAPL"*

Methodology

- Power models
  - Mostly linear [1], trustfully represent the power consumption
  - Component metrics are gathered with power consumption
- CPU metrics
  - CPU load [2]
  - Hardware Performance Counters (HPC) [3, 4, 5, 6, 7]
- HPCs
  - Architecture-dependent
  - Considered by state-of-the-art as the most accurate metrics

[1] John McCullough, et al. *"Evaluating the effectiveness of model-based power characterization"*
[2] Daniel Versick, et al. *"Power consumption estimation of CPU and peripheral components in virtual machines"*
[3] Ramon Bertran, et al. *"Decomposable and responsive power models for multicore processors using performance counters"*
[4] William Lloyd Bircher, et al. *"Complete system power estimation: A trickle-down approach based on performance events"*
[5] Min Yeol Lim, et al. *"Softpower: Fine-grain power estimations using performance counters"*
[6] Vasileios Spiliopoulos, et al. *"Power-sleuth: A tool for investigating your program's power behaviour"*
[7] Yan Zhai, et al. *"Happy: Hyperthread-aware power profiling dynamically"*

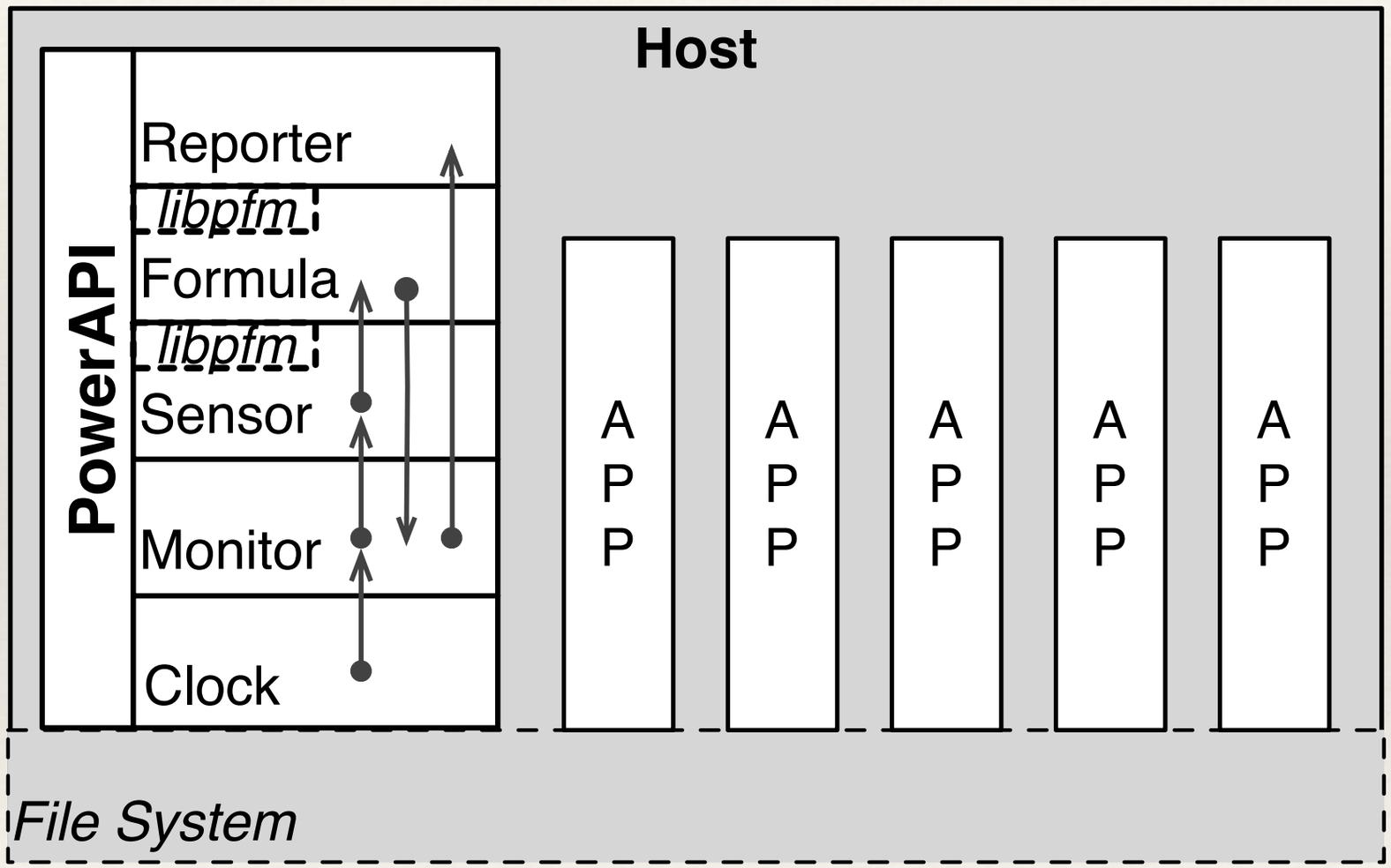
Statements

- Problems
  - Most of power models are architecture and software dependents
  - Lack of information, difficult to adapt and to reproduce
- Solutions
  - Criteria selection for HPCs: Availability, exploitation overhead, evolution
  - Architecture-agnostic power models

Contribution

- PowerAPI, under AGPL v3 license
  - Scala / Akka
  - Actor model
  - Modulable
  - Software-defined power meter
  - Available on GitHub

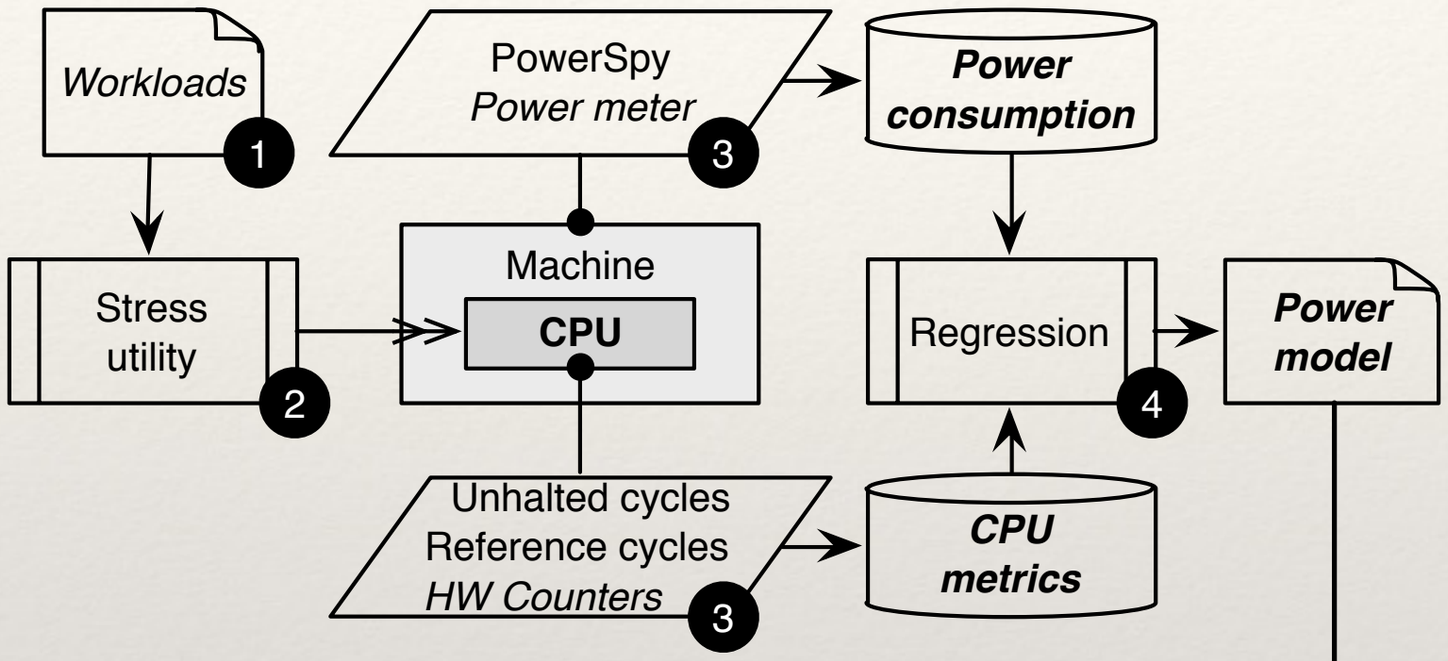
PowerAPI: Architecture



PowerAPI: Basics

- First step: Learning the CPU power model
- Second step: Software power estimation at runtime

1) Learning the CPU power model

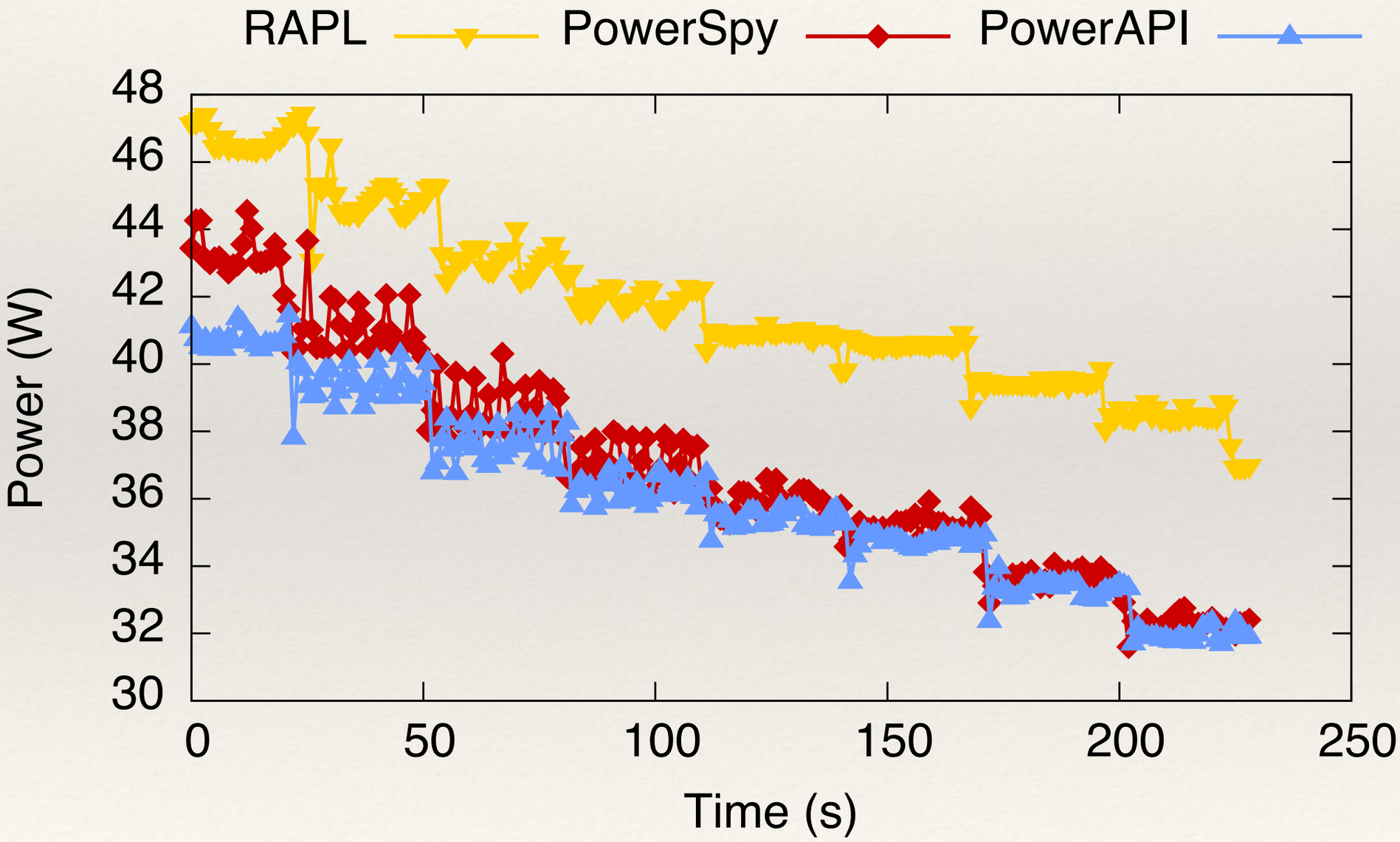


Power Model

$$P_{host}(f) = P_{idle}(f) + \sum_{pid \in PIDs} P_{cpu}(f, uc^1_{pid} ... uc^N_{pid})$$

$$P_{cpu}(f, uc^1_{pid} ... uc^N_{pid}) = \sum_{n=1}^{N} P_f(uc^n_{pid})$$

Why an external power meter?



Evaluation: Setup

PARSEC

SPECjbb

Intel Xeon W3520

Intel i3 2120 (x 3)

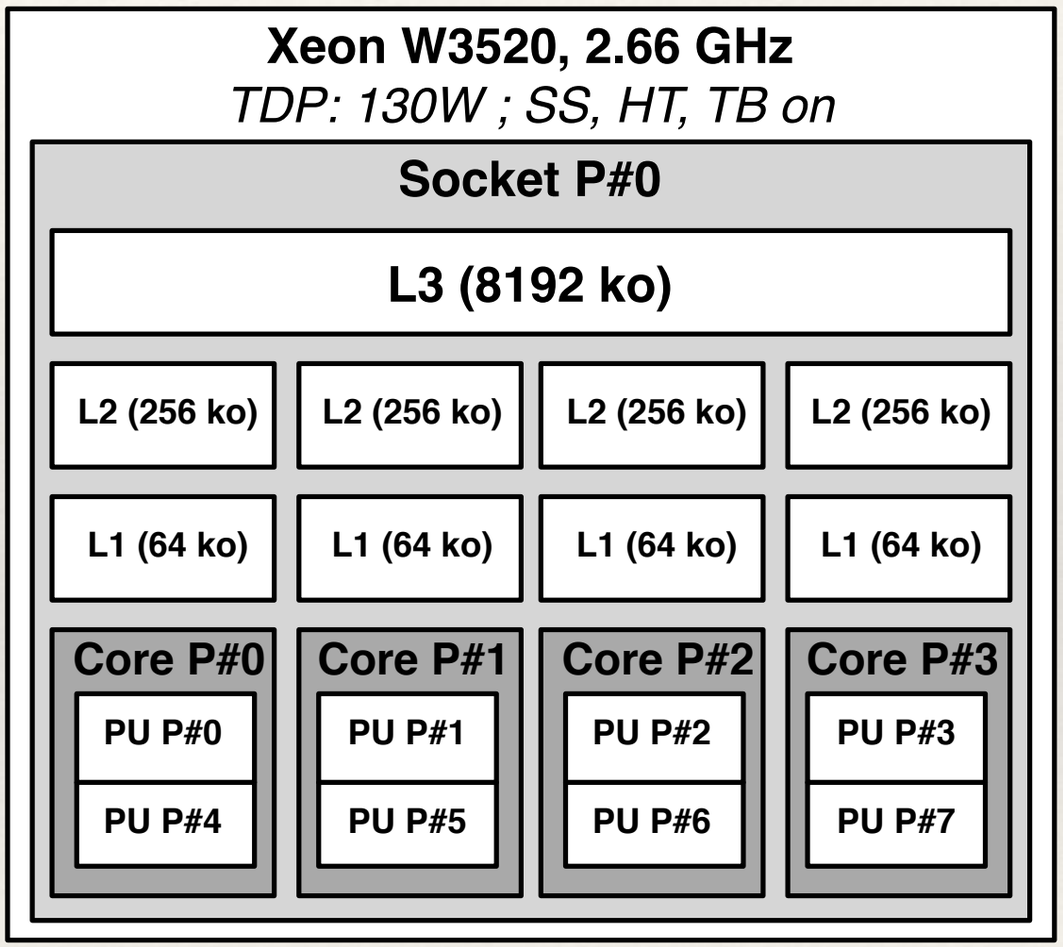
Evaluation: Setup (2)

PARSEC

SPECjbb

Intel Xeon W3520

Intel i3 2120 (x 3)



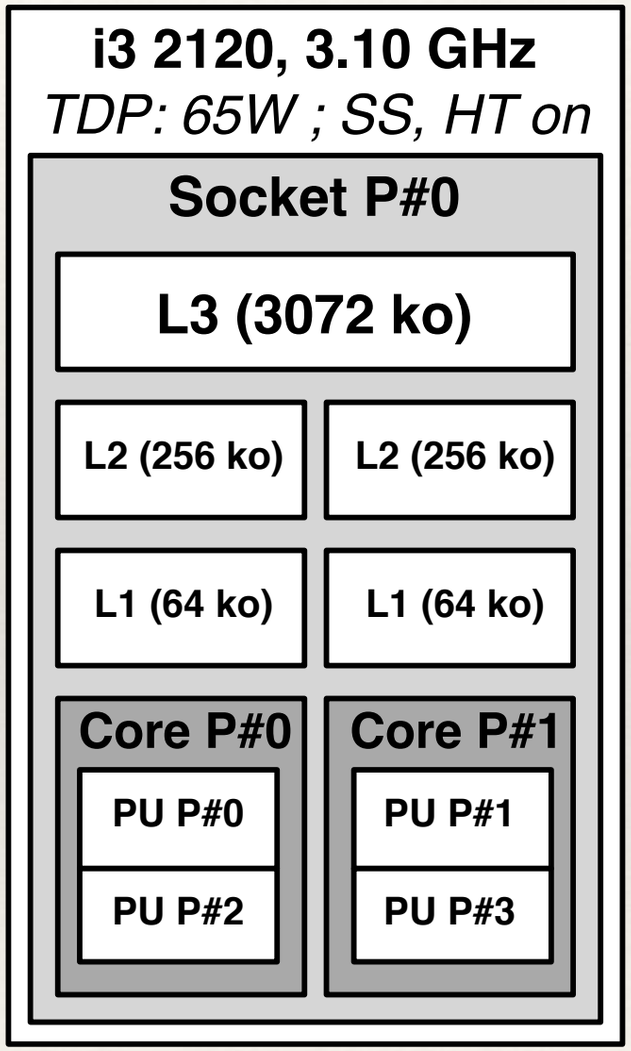
Evaluation: Setup (3)

PARSEC

SpecJBB

Intel Xeon W3520

Intel i3 2120 (x 3)



Evaluation: Setup (4)

PARSEC

SPECjbb

Intel Xeon W3520

Intel i3 2120 (x 3)

- Designed for multi-core architectures
- Multi-threaded
- CPU & memory intense

Evaluation: Setup (5)

PARSEC

SPECjbb

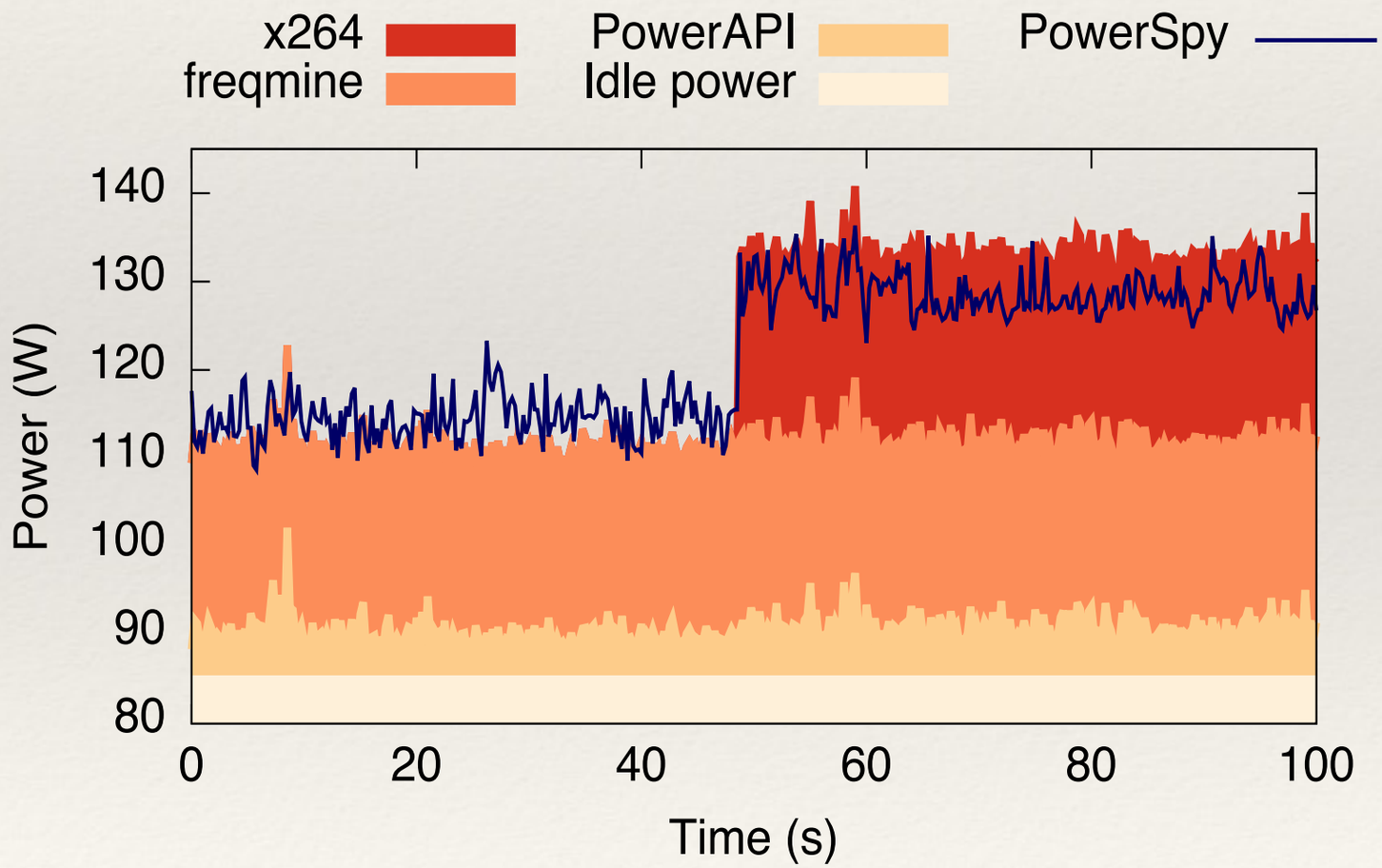
Intel Xeon W3520

Intel i3 2120 (x 3)

- Real world
- Multi-threaded
- Distributed

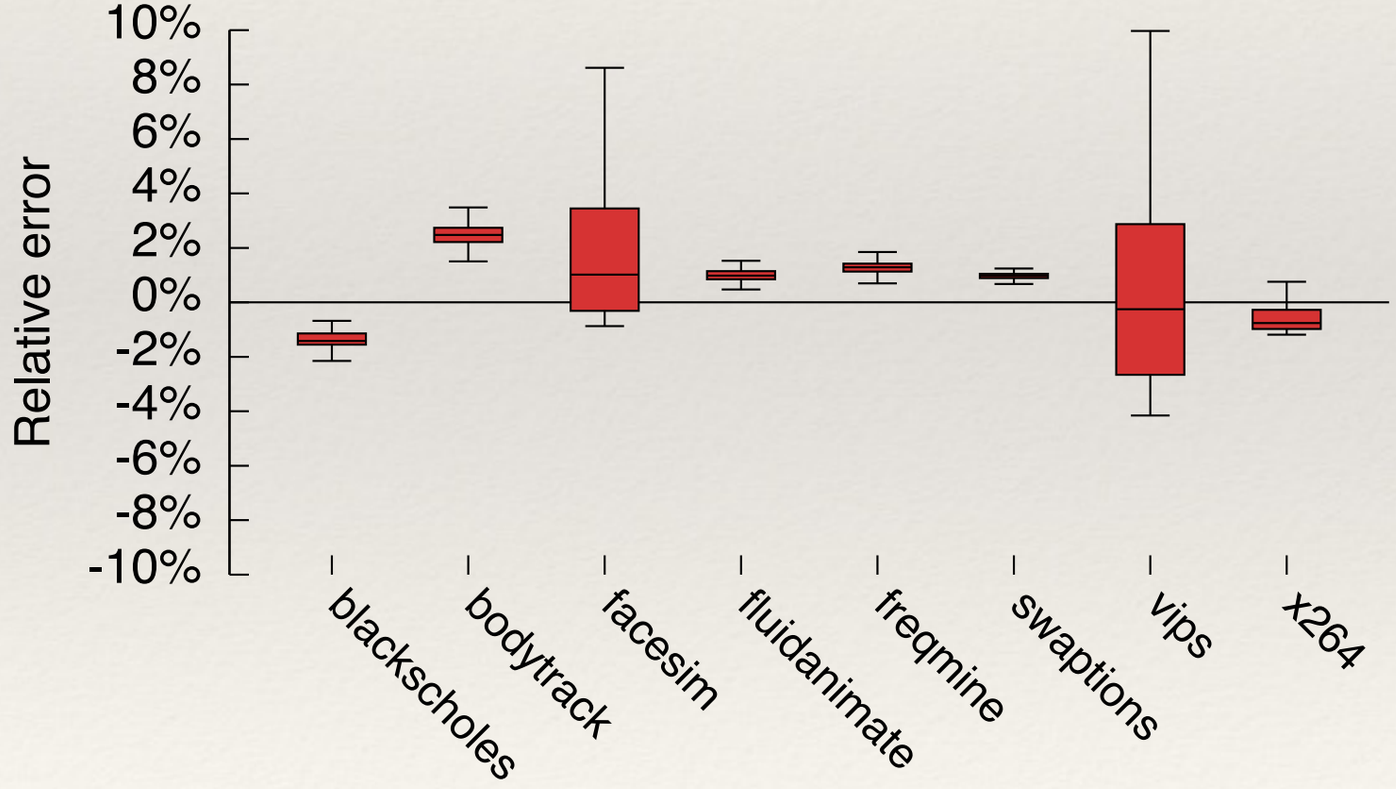
Evaluation: Multi-processes monitoring

PARSEC benchmarks (multi-threaded, CPU & memory intense)



Evaluation: PARSEC on the Xeon W3520

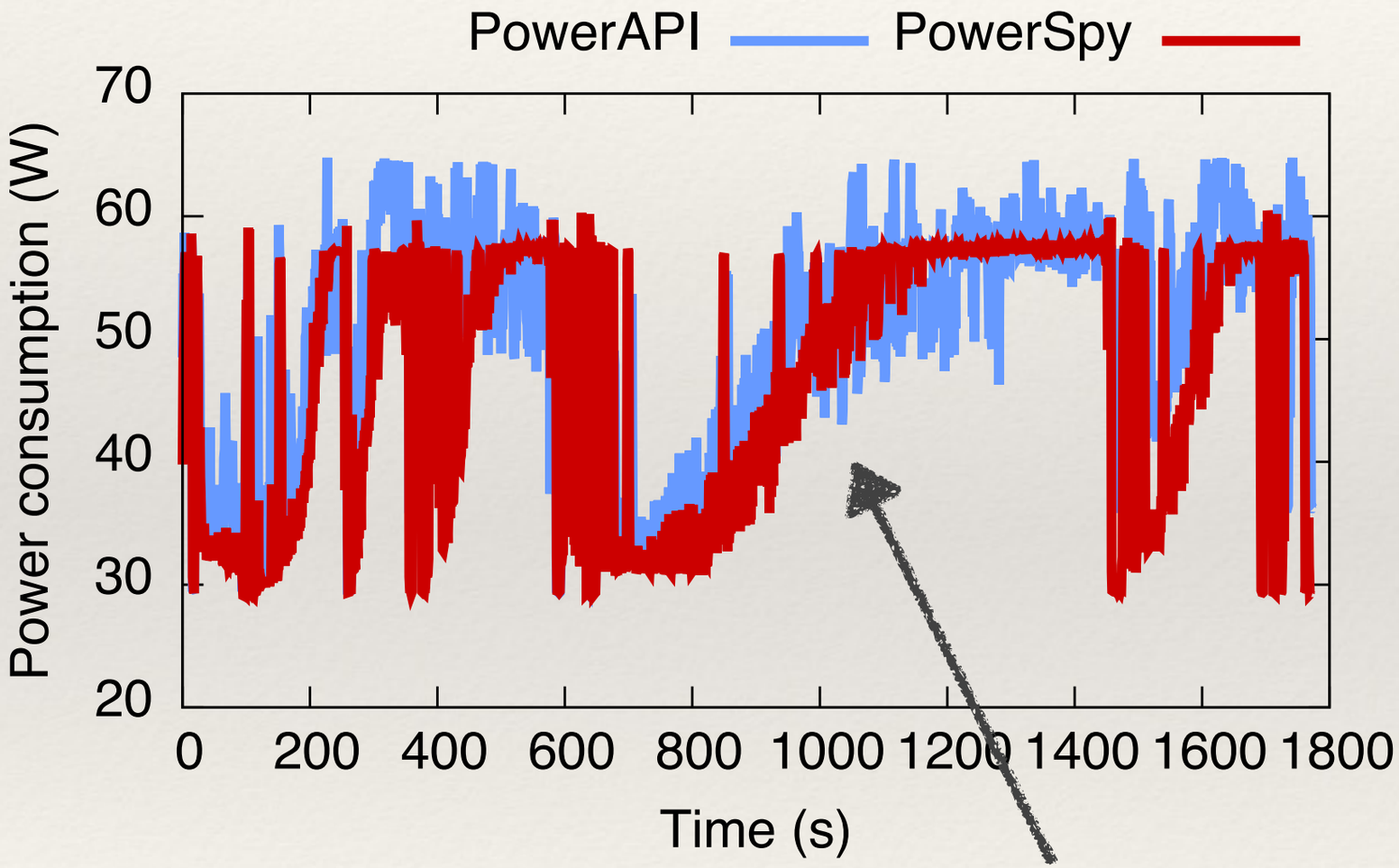
PARSEC benchmarks (multi-threaded, CPU & memory intense)



Evaluation: SPECjbb2013

- Supermarket company
- Distributed warehouses
- Online purchases
- Management operations (data mining)

Evaluation: SPECjbb2013 on the i3 2120



Real world application: load variations

Conclusion

- PowerAPI: middleware toolkit for process-level power estimation
- Hardware features agnostic
  - Turbo, HyperThreading, SpeedStep, ...
- Distributed monitoring support
- No dedicated hardware

Thanks for your attention.

Questions?

Contact: maxime.colmant@inria.fr