PowerAPI: *Building Software-Defined Power Meters*

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

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"*

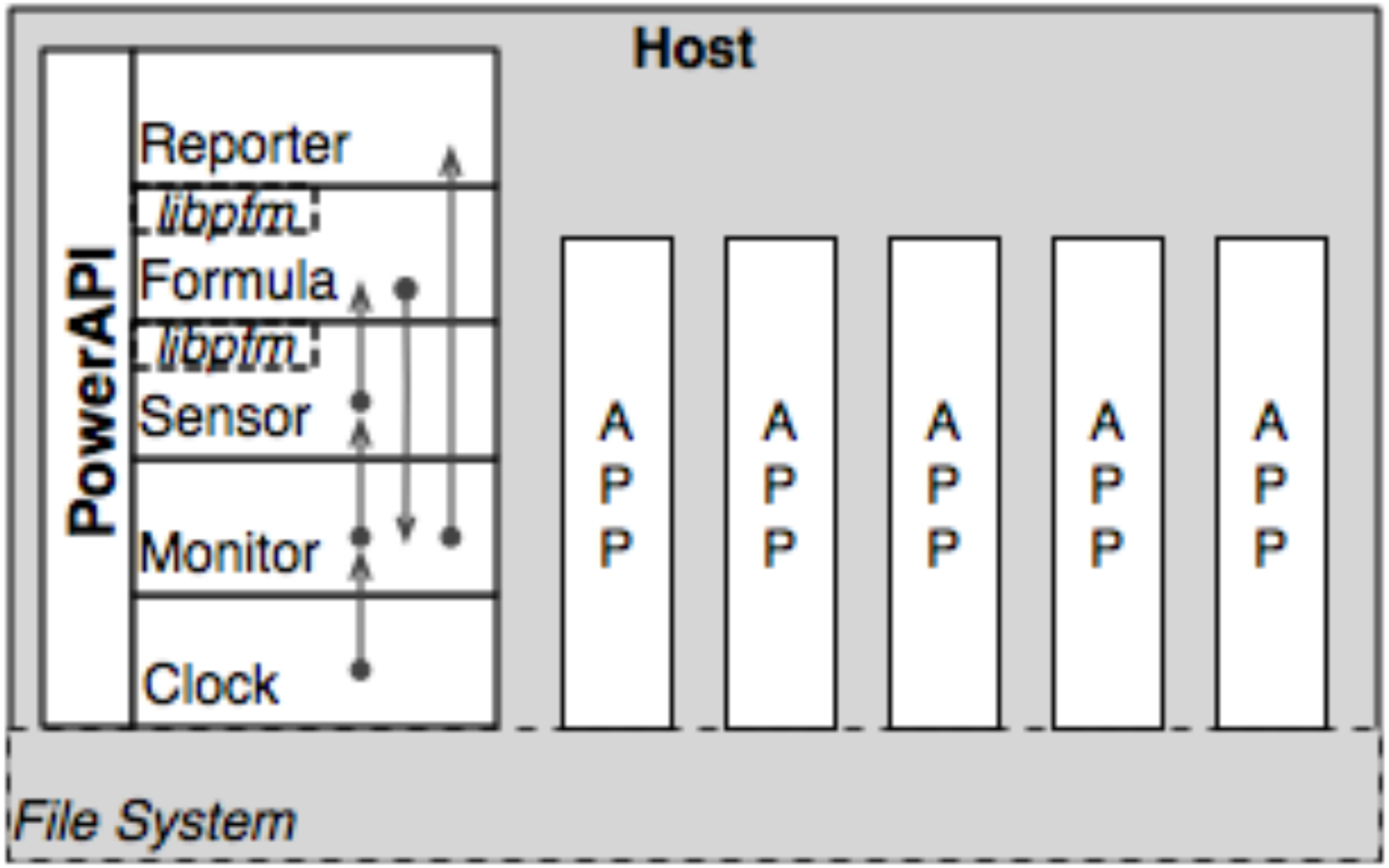
Statements

- 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
  - Toolkit for Software-defined power meter
  - Scala / Akka
  - Actor model
  - Modular
  - Available on GitHub (http://powerapi.org)

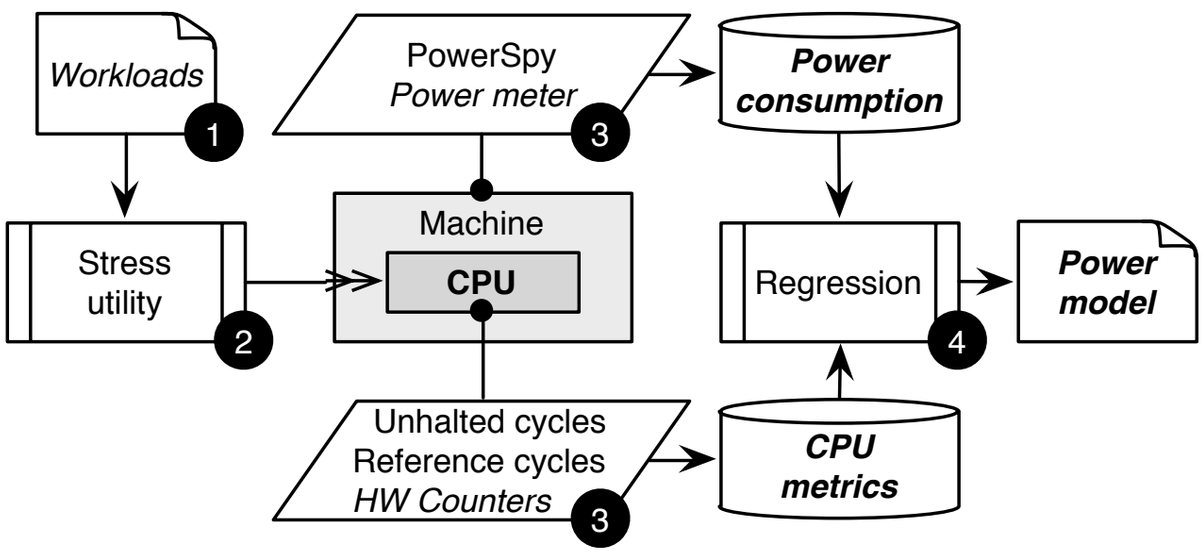
PowerAPI: Architecture

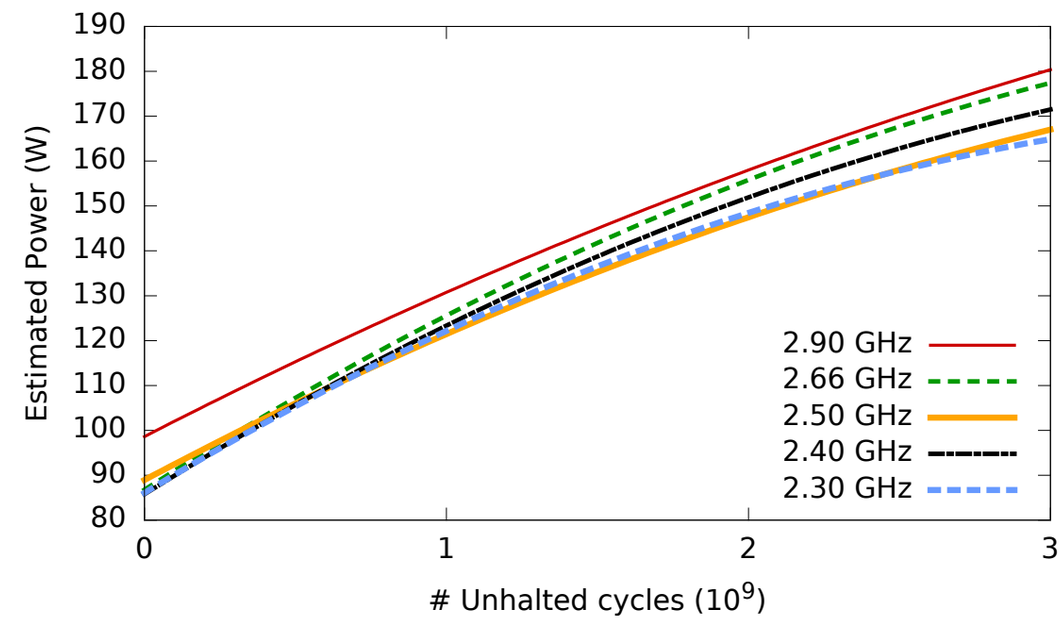


PowerAPI: Basics

- 1st step: Learning the CPU power model
- 2nd step: Software power estimations at runtime

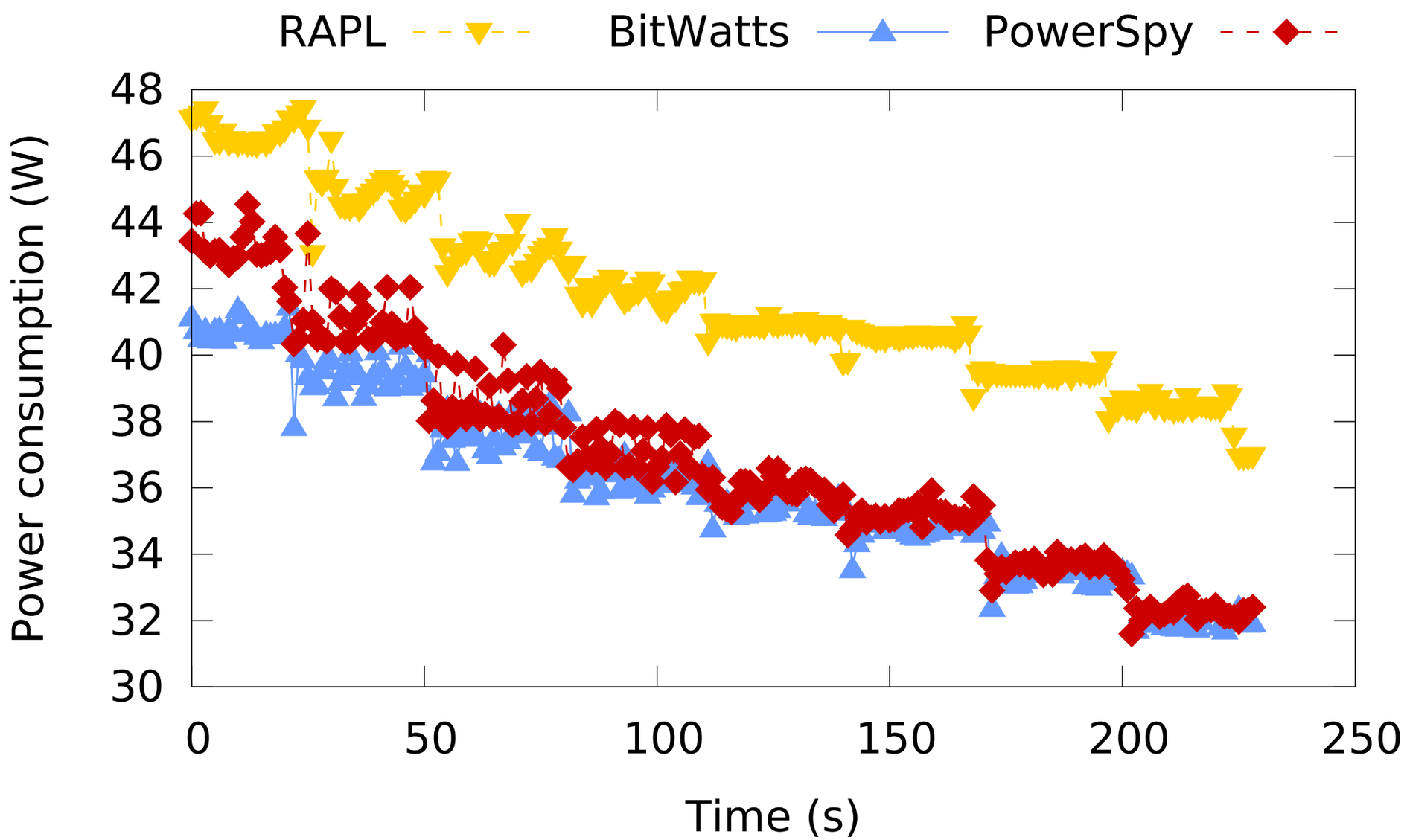
1) Learning the CPU power model





$$P_{2.90}(uc_{pid}) = \frac{8.64 \cdot uc_{pid}}{10^9} - \frac{6.10 \cdot uc_{pid}^2}{10^{18}}.$$

Why an external power meter?



Evaluation: Setup

PARSEC	SPECjbb
Intel Xeon W3520	Intel i3 2120 (x 3)

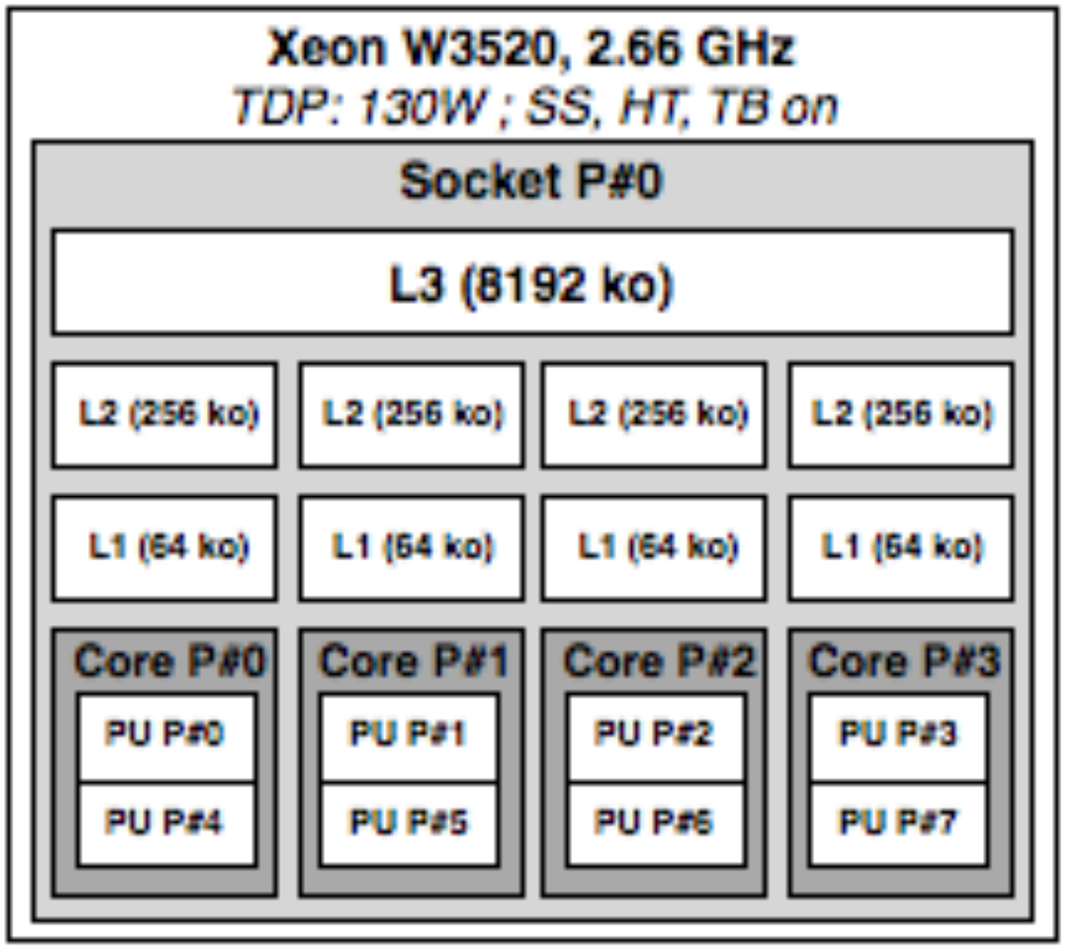
Evaluation: Setup (2)

PARSEC

SPECjbb

Intel Xeon W3520

Intel i3 2120 (x 3)



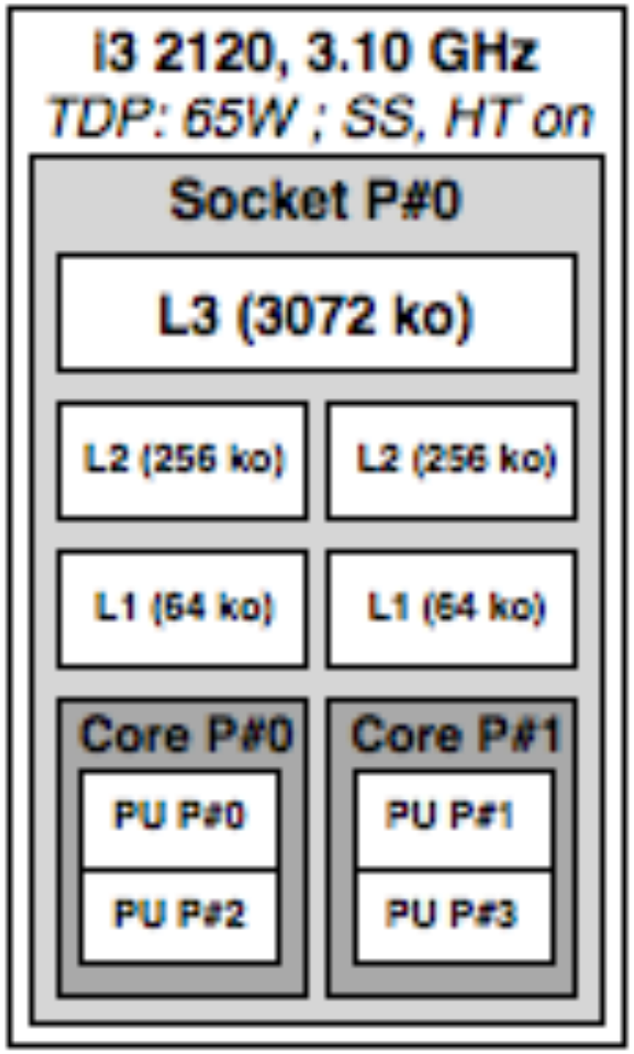
Evaluation: Setup (3)

PARSEC

SpecJBB

Intel Xeon W3520

Intel i3 2120 (x 3)

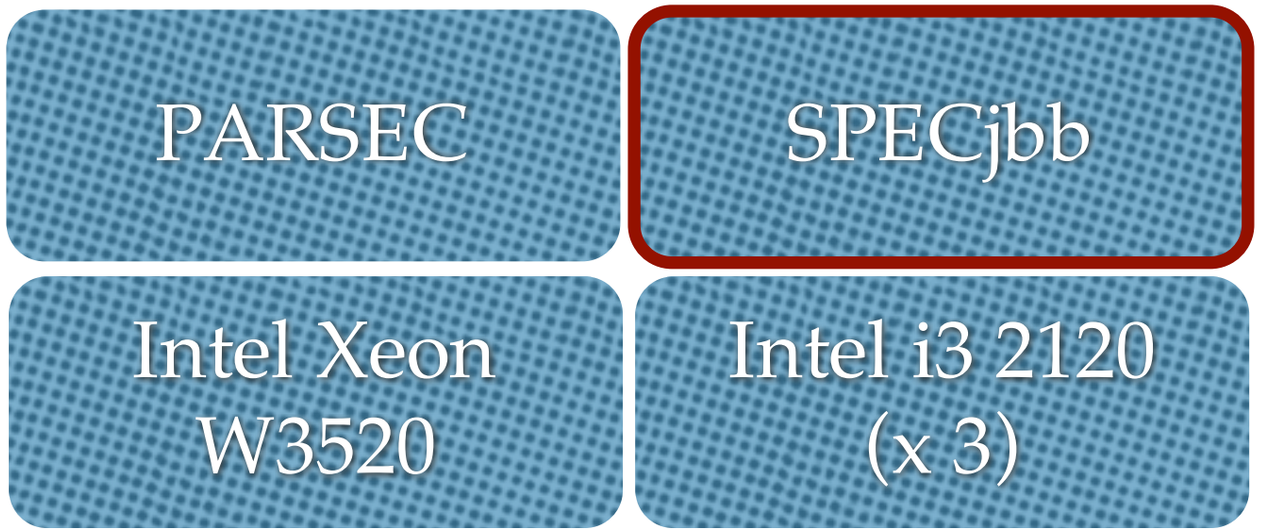


Evaluation: Setup (4)

PARSEC	SPECjbb
Intel Xeon W3520	Intel i3 2120 (x 3)

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

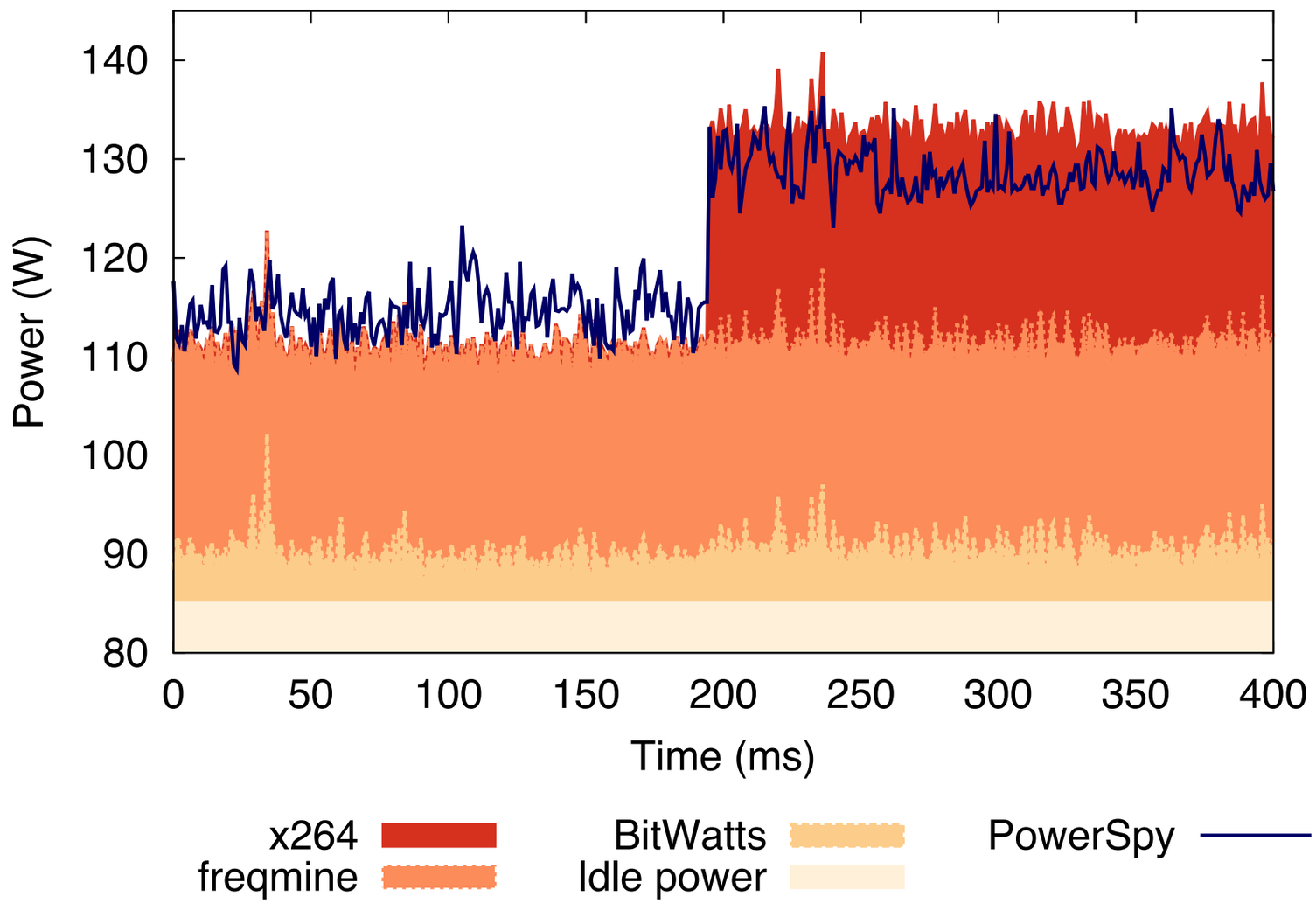
Evaluation: Setup (5)



- Real world
- Multi-threaded
- Distributed

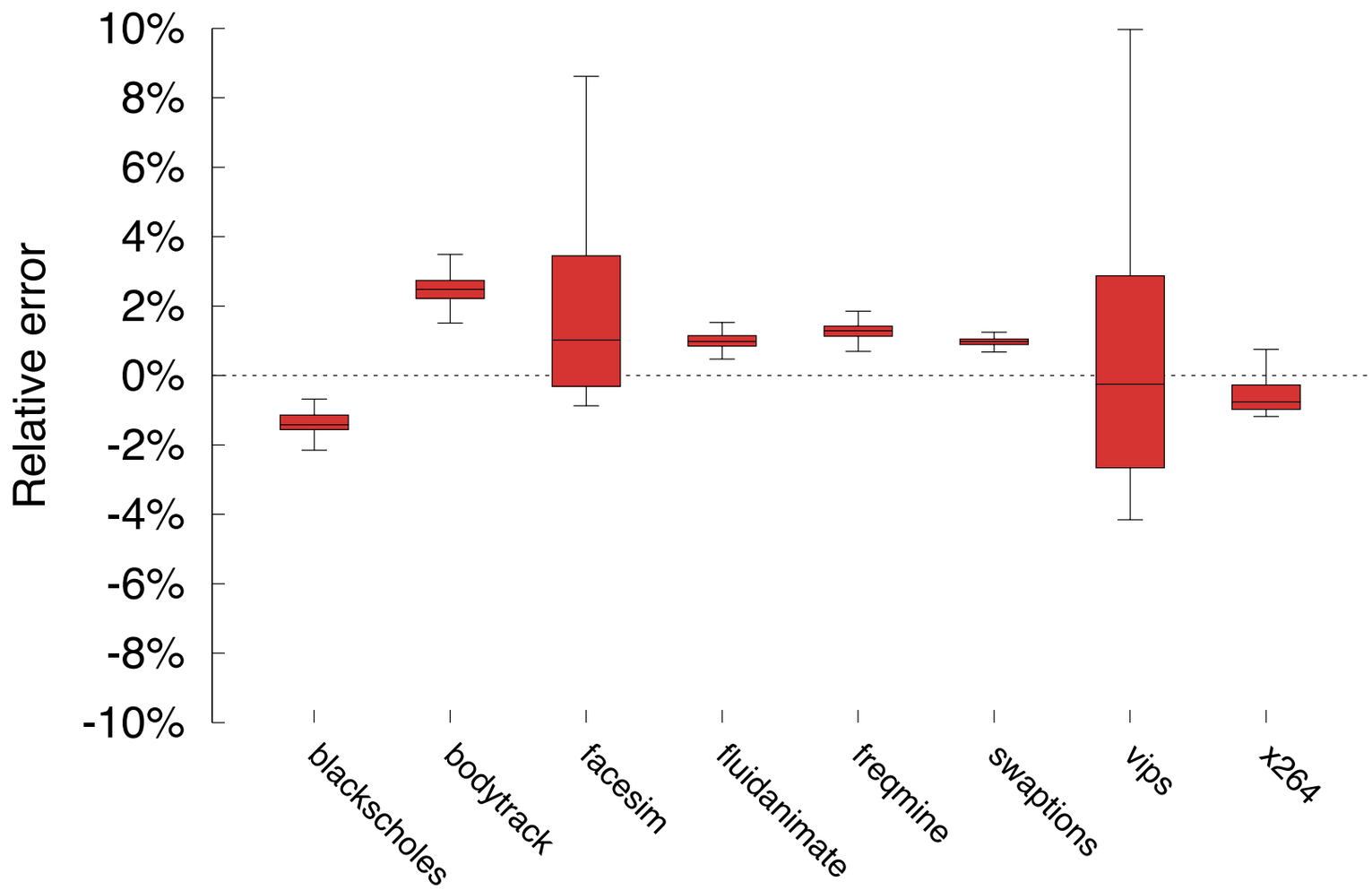
Evaluation: Multi-processes monitoring

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



Evaluation: PARSEC on the Xeon W3520

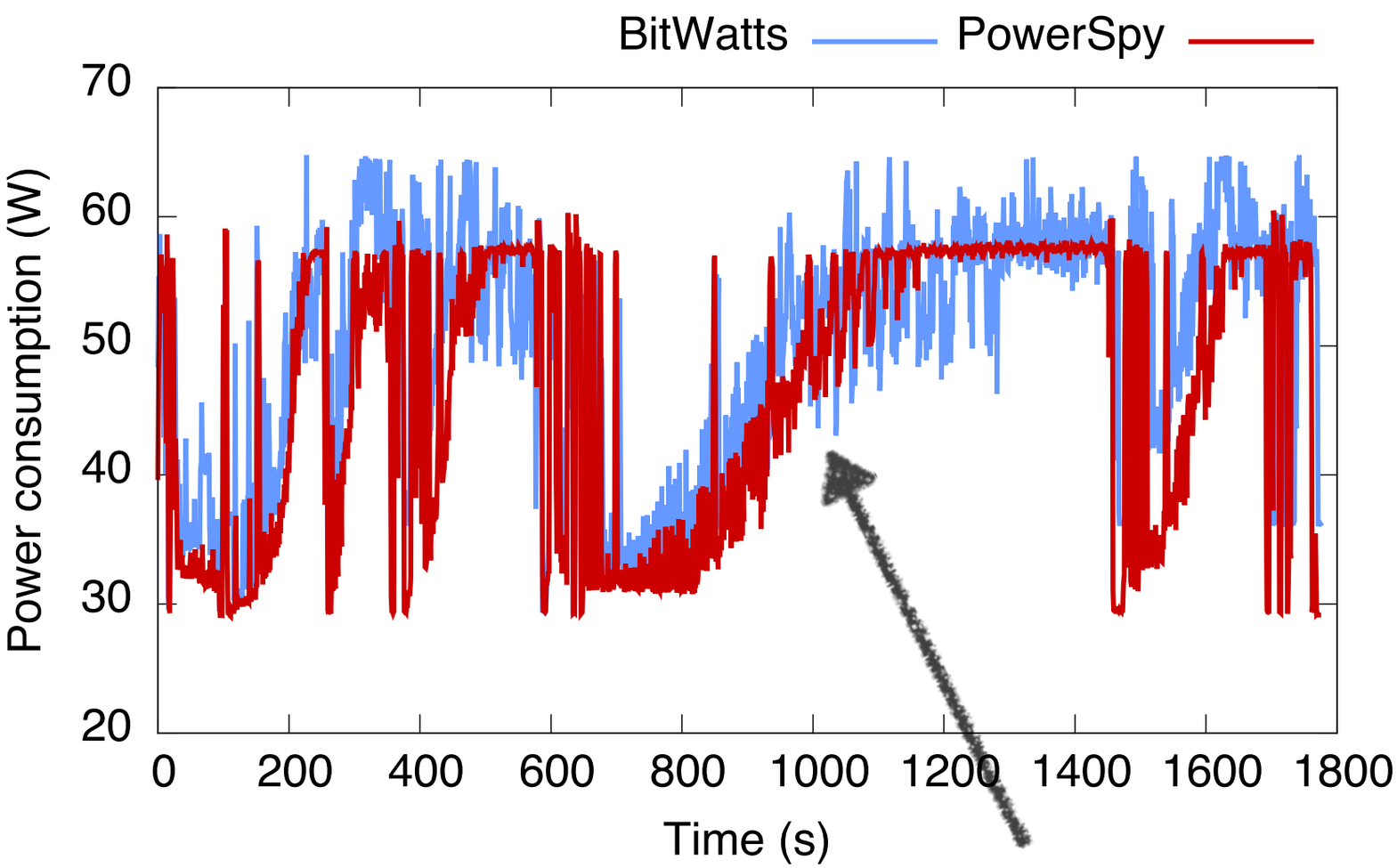
PARSEC benchmarks (multi-threaded, CPU & memory intense)



Evaluation: SPECjbb2013

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

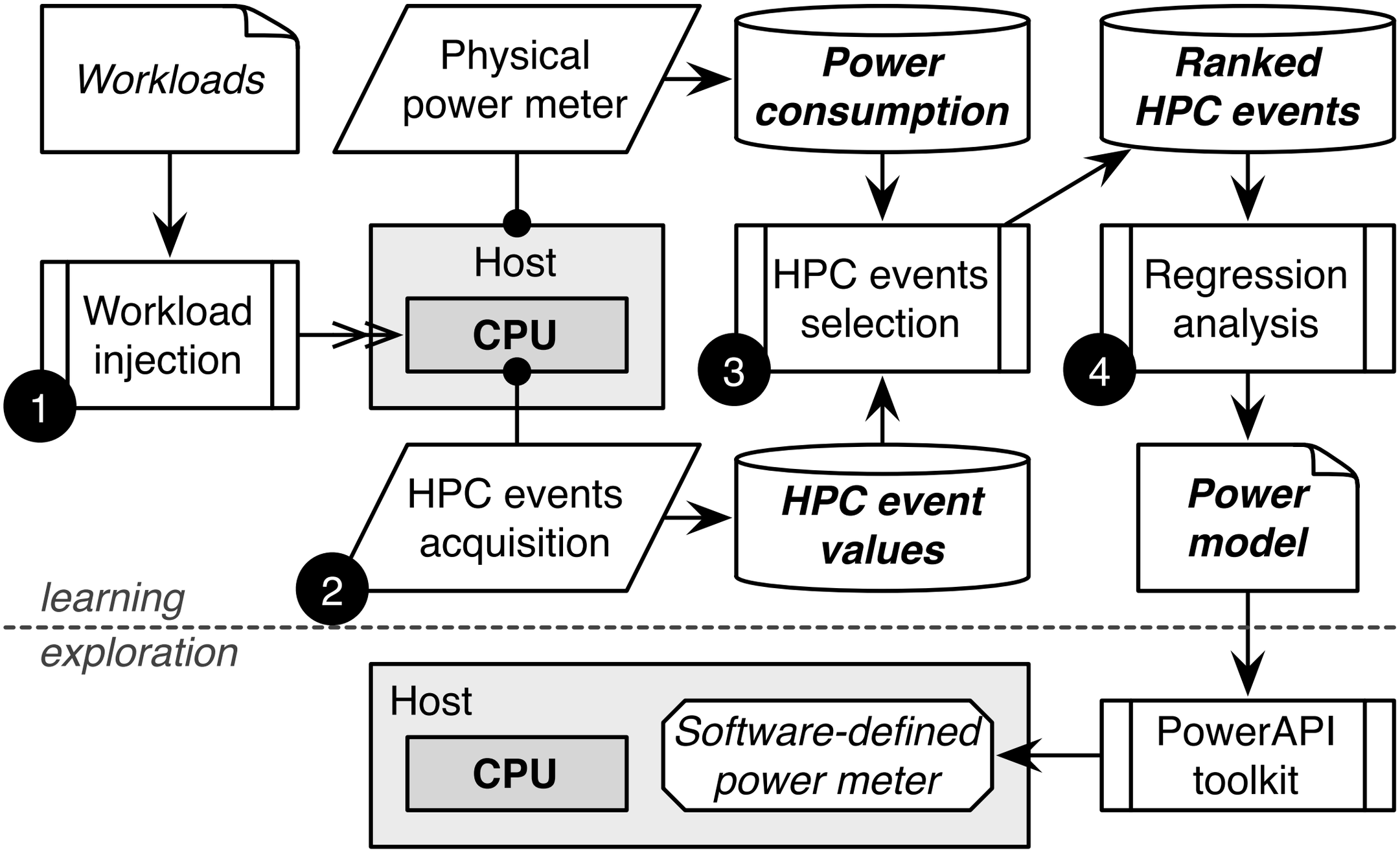
Evaluation: SPECjbb2013 on the i3 2120



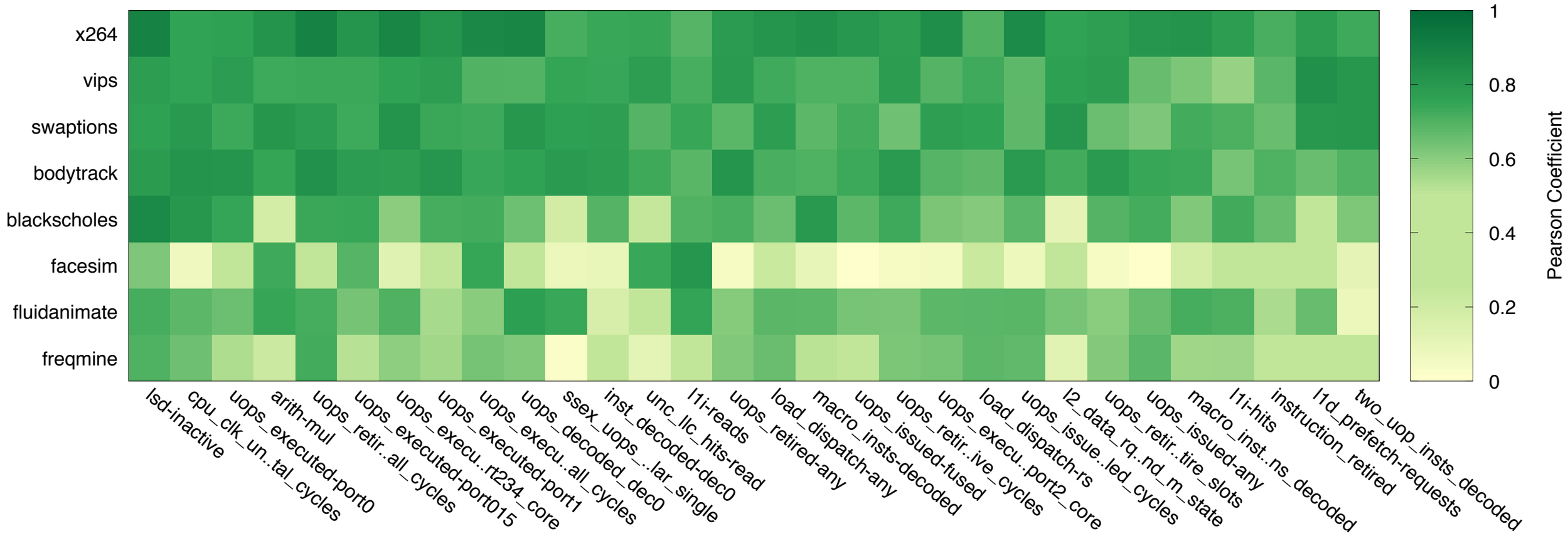
Real world application: load variations

Author(s)	**Processor(s)**	**Feature(s)**	**Regression(s)**	**Benchmarks**	**Error(s)**
A.Aroca *et al.* [2]	Xeon E5606 & W3530, Opteron 6276	HW sensors	polynomial, multiple linear	*eval.:* Hadoop App.	$< 7\%$ of total energy
Bertran *et al.* [7]	Core 2 Duo	14 HPC regrouped by component	multiple linear by component	*sampl.:* μ-benchs *eval.:* SPEC CPU 06	5%
Bircher *et al.* [10]	Pentium 4	μ-ops trace-cache, micro-code ROM	multiple linear	*sampl.:* SPEC CPU 00 *eval.:* SPEC CPU 00	2.5%
Colmant *et al.* [11]	Xeon W3520 & i3 2120	non-halted cycles reference cycles	polynomial	*sampl.:* stress *eval.:* PARSEC, SPEC JBB	3%
Contreras *et al.* [12]	XScale PXA255	5 HPCs	multiple linear	*eval.:* SPEC CPU 00, Java CDC/CLDC	4%
Dolz *et al.* [13]	Xeon E3-1275	3 HPCs HW sensors	linear	*sampl.:* linpack, stream, iperf, IOR *eval.:* Quantum Espresso	3 W 70 W max.
Economou *et al.* [14]	Turion, Itanium 2	HW sensors	multiple linear	*sampl.:* Gamut *eval.:* SPECs, Matrix, Stream	5%
Isci *et al.* [18]	Pentium 4	15 HPC	multiple linear	*eval.:* μ-benchs, AbiWord, Mozilla, Gnumeric	3 W
Li *et al.* [24]	8-way issue superscalar	IPC	linear	*sampl./eval.:* DB, email, SPEC JVM 98, SPEC INT 95	1% off. 6% run.
Rivoire *et al.* [35]	Core 2 Duo & Xeon, Itanium 2, Turion	HW sensors HPCs	multiple linear	*sampl.:* calibration suite *eval.:* SPECs, Stream, ClamAV	$< 5\%$
Yang *et al.* [42]	Xeon E5620 & E7530	7 components 91 preselected	support vector	*sampl.:* NPB, IOzone, CacheBench *eval.:* SPEC CPU 06, IOzone	4.7%
Zamani *et al.* [43]	Opteron	3 to 5 HPCs	ARMAX	*sampl./eval.:* BT.C, CG.C, LU.C, SP.C	0.1% – 0.5% offline
Zhai *et al.* [44]	Sandy Bridge	non-halted cycles	linear	*eval.:* Google, SPEC CPU 06	7.5%

Learning the relevant HPC



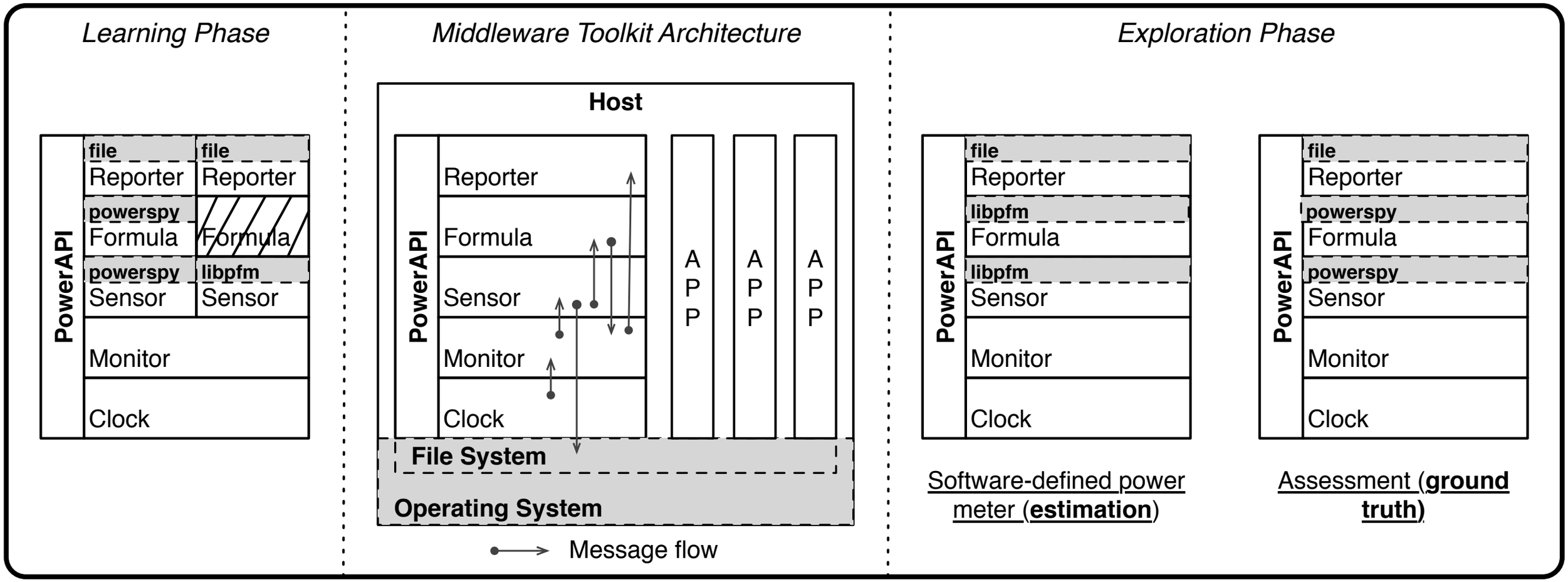
Correlating HPC & Power

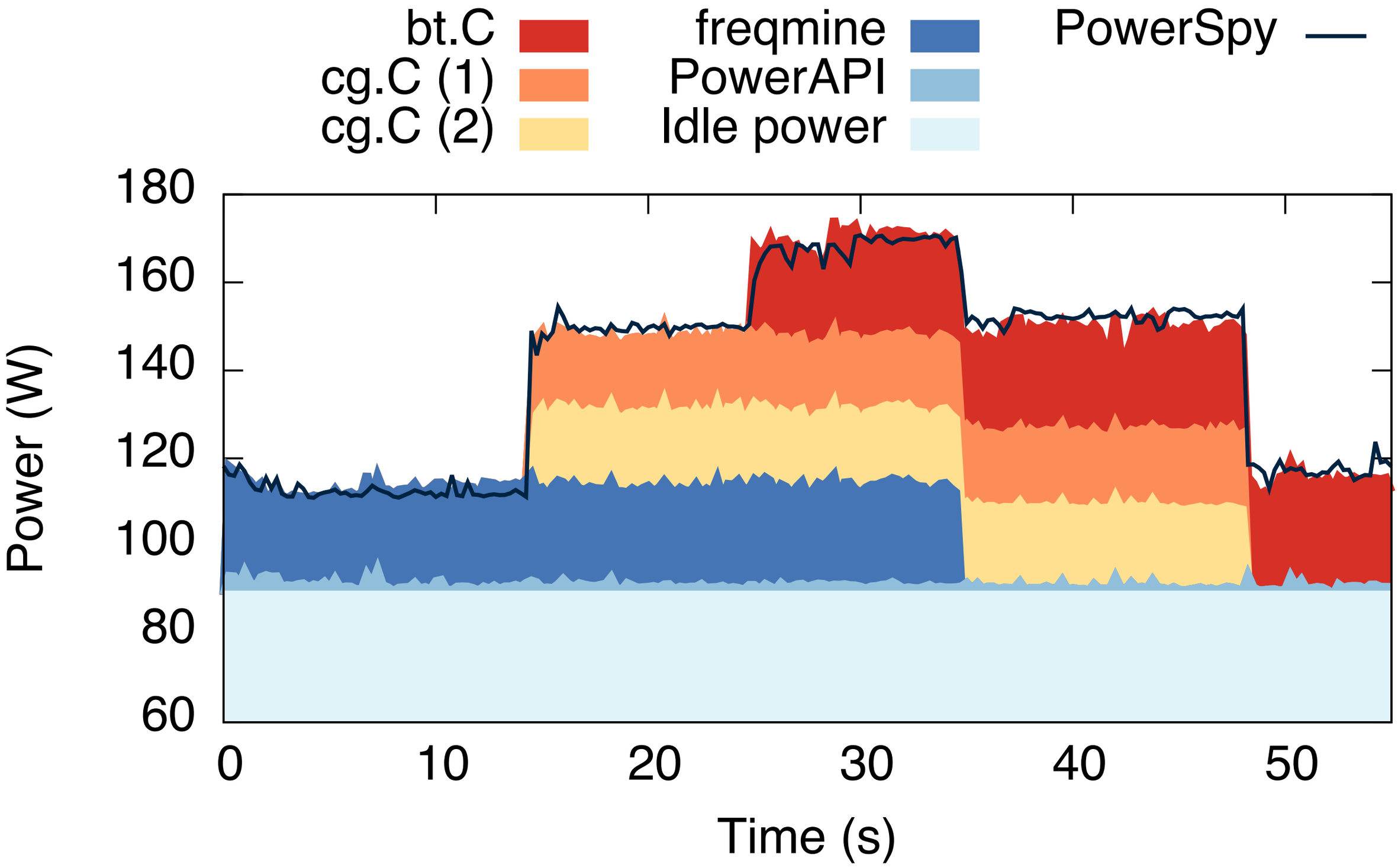


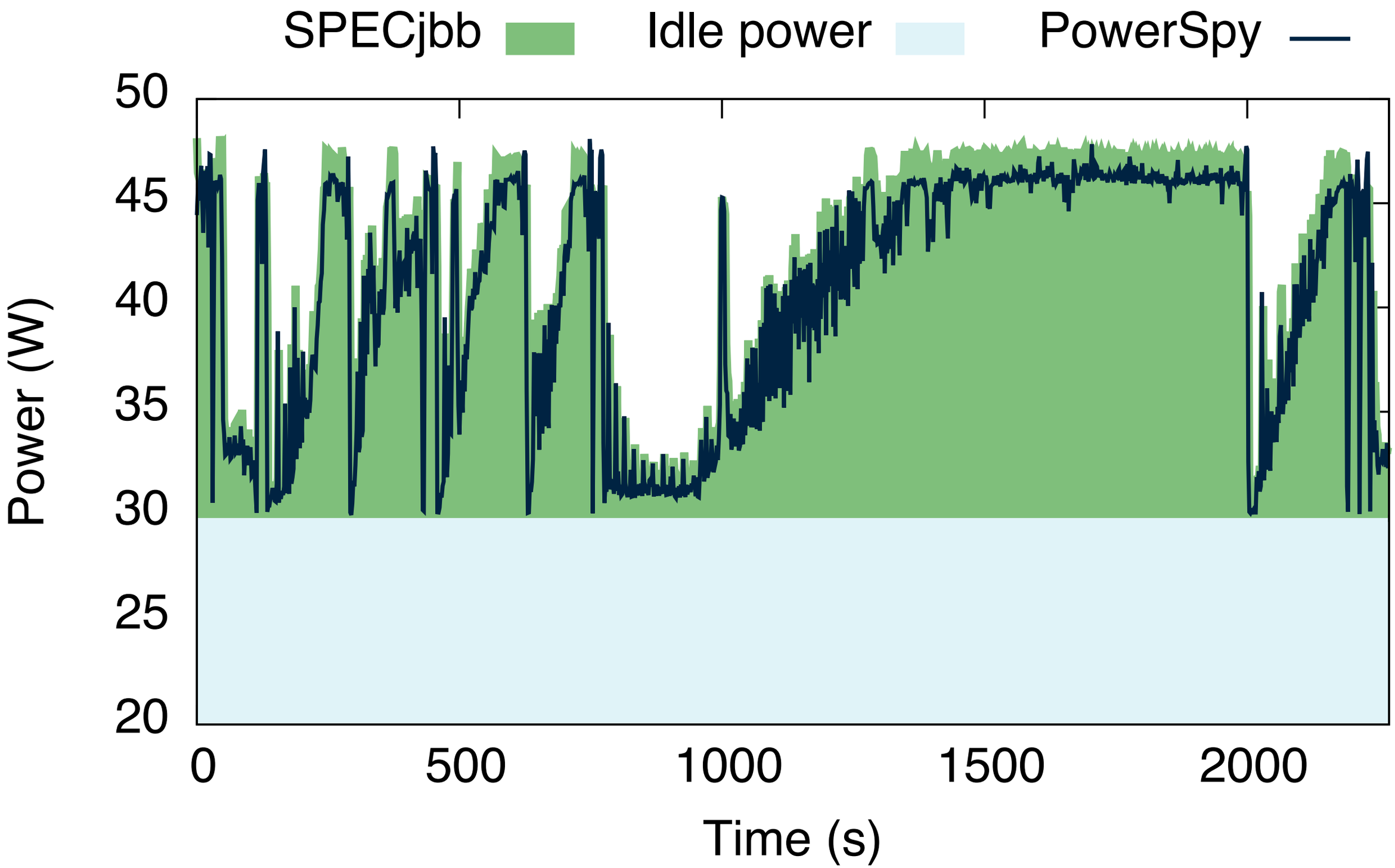
Manuf.	CPU	PMU	#Gen.	#Ev.
Intel	Xeon W3520	nhm	4	338
		nhm_unc	8	176
	i3 2120	snb	4	336
		snb_unc_cbo0	2	19
		snb_unc_cbo1	2	18
AMD	Opteron 8354	fam10h_barcelona	3	421
ARM	Cortex A15	arm_ac15	6	67

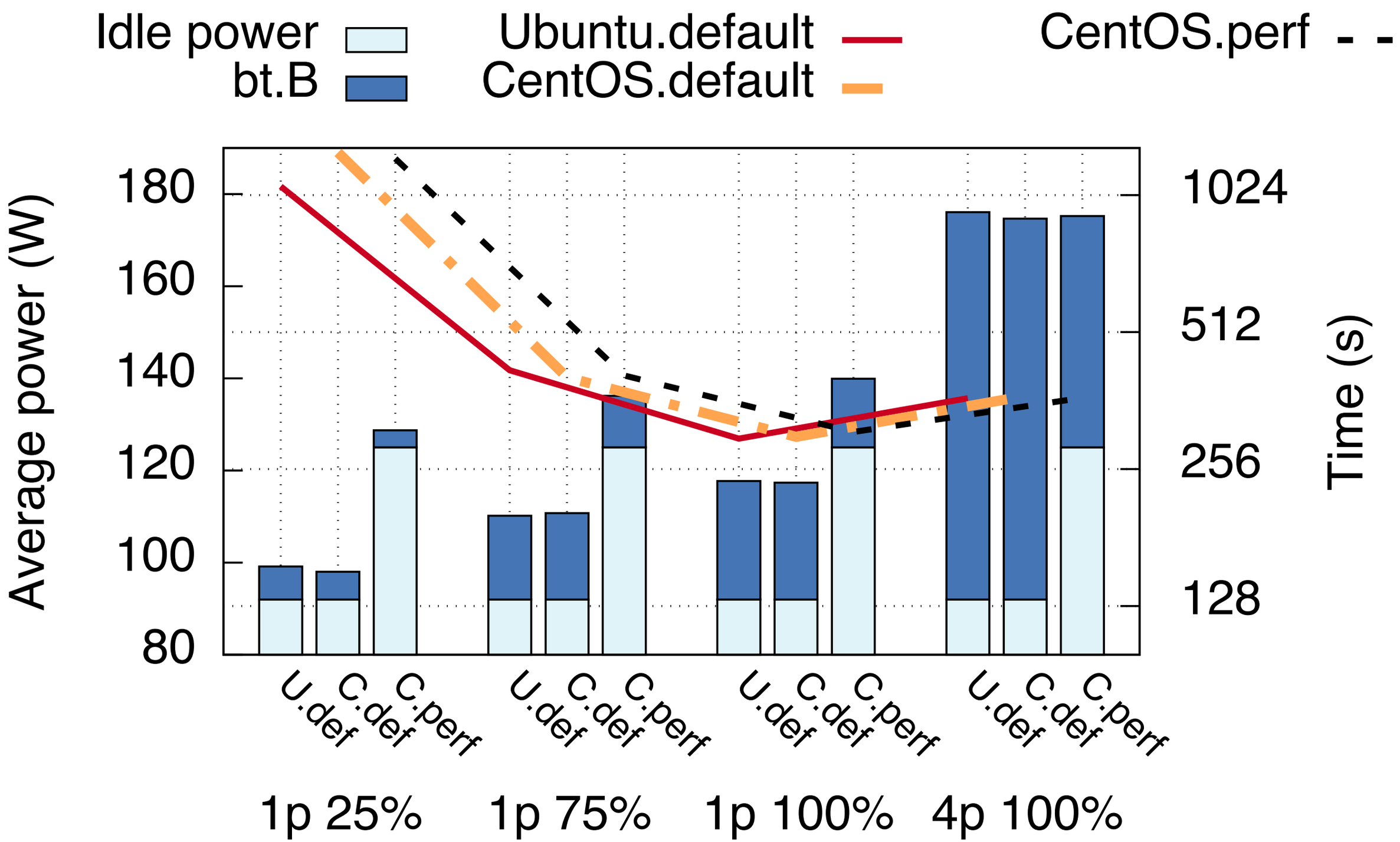
$$w = \left\lceil \sum_{p \in PMUs} \frac{|E_p|}{|C_p|} \right\rceil \times |W| \times i$$

PowerAPI: Middleware Toolkit




bt.C
cg.C (1)
cg.C (2)
freqmine
PowerAPI
Idle power
PowerSpy
Power (W)
180
160
140
120
100
80
60
0
10
20
30
40
50
Time (s)

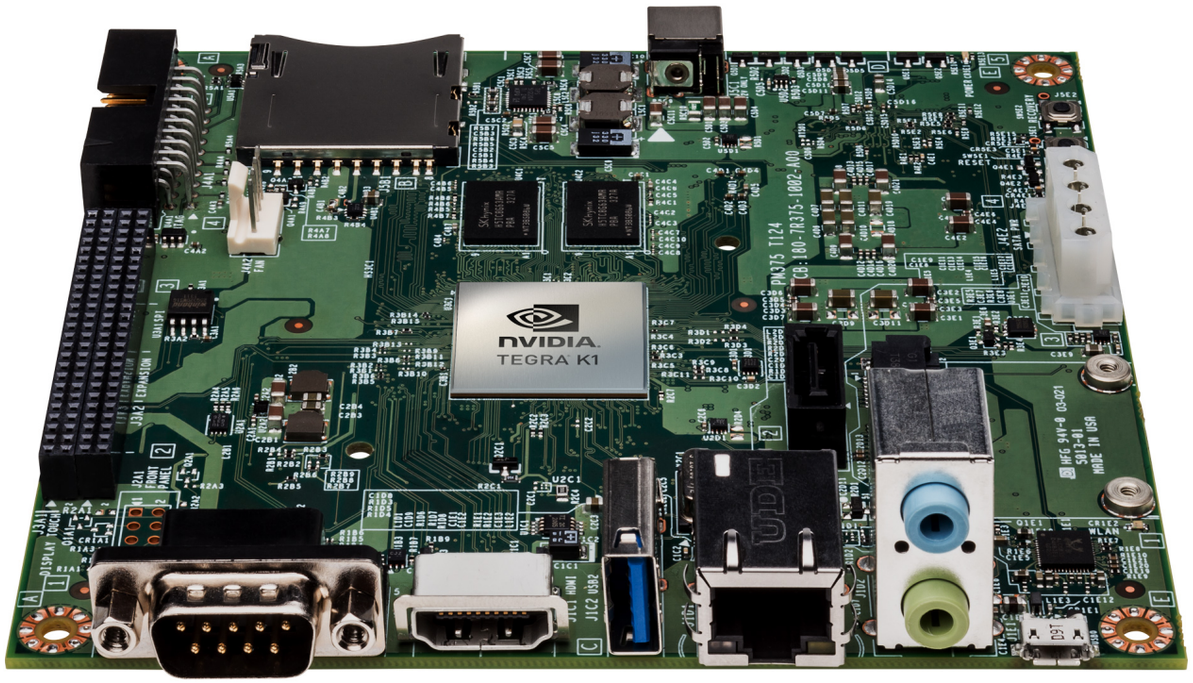

SPECjbb
Idle power
PowerSpy
50
45
40
35
30
25
20
Power (W)
0
500
1000
1500
2000
Time (s)

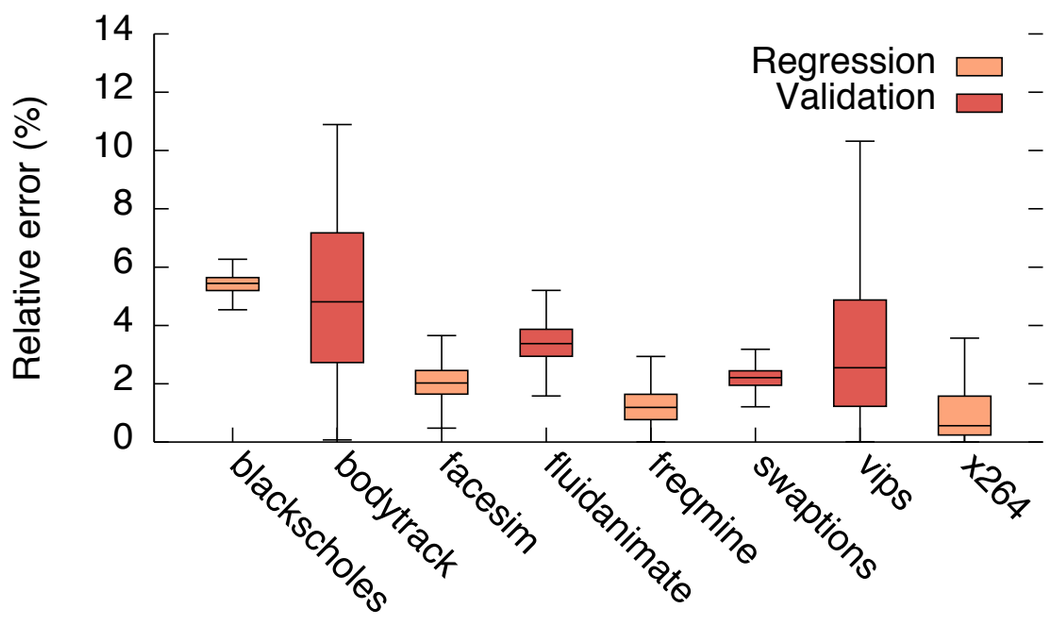

Idle power
bt.B
Ubuntu.default
CentOS.default
CentOS.perf
Average power (W)
180
160
140
120
100
80
1024
512
256
128
Time (s)
U.def
C.def
C.perf
1p 25%
1p 75%
1p 100%
4p 100%

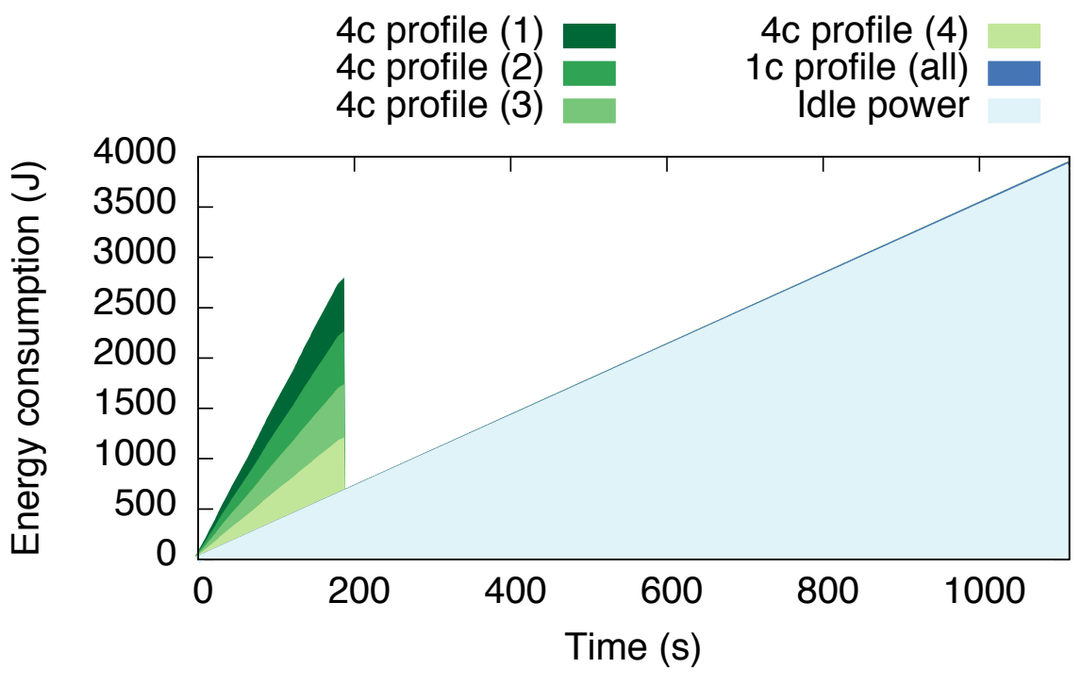
Beyond Intel CPUs

Nvidia Jetson TK1 Dev Kit:

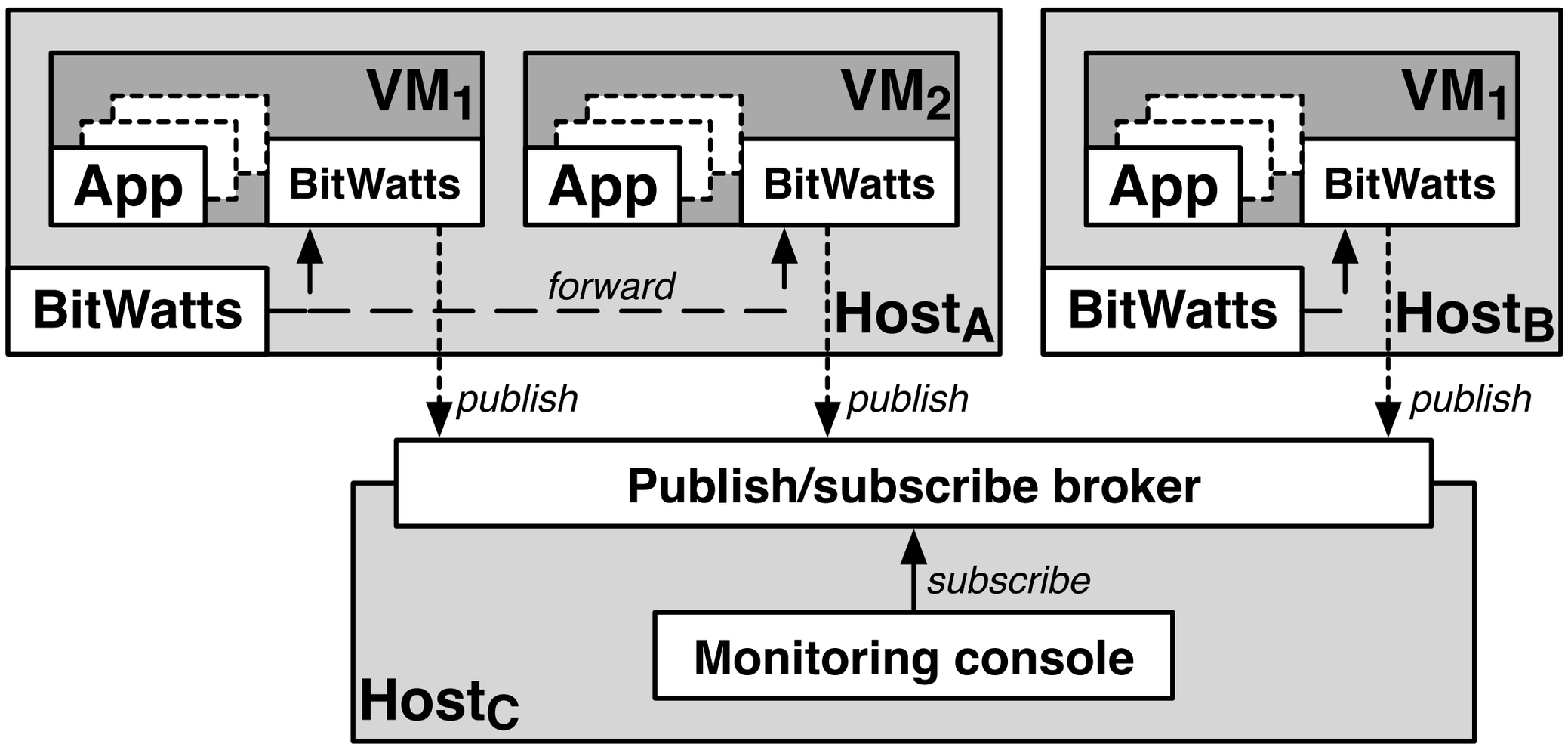
- Tegra K1 SOC, Kepler GPU w/ 192 Cores,
- NVIDIA 4-Plus-1 quad-core ARM Cortex-A15 CPU,
- 2GB Mem,
- 16GB eMMC



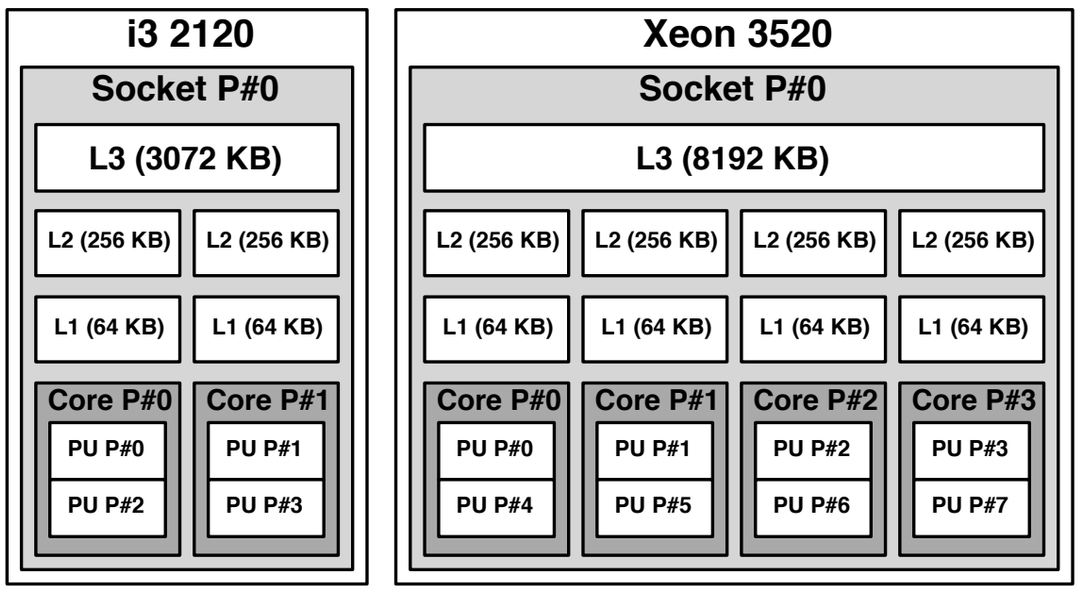




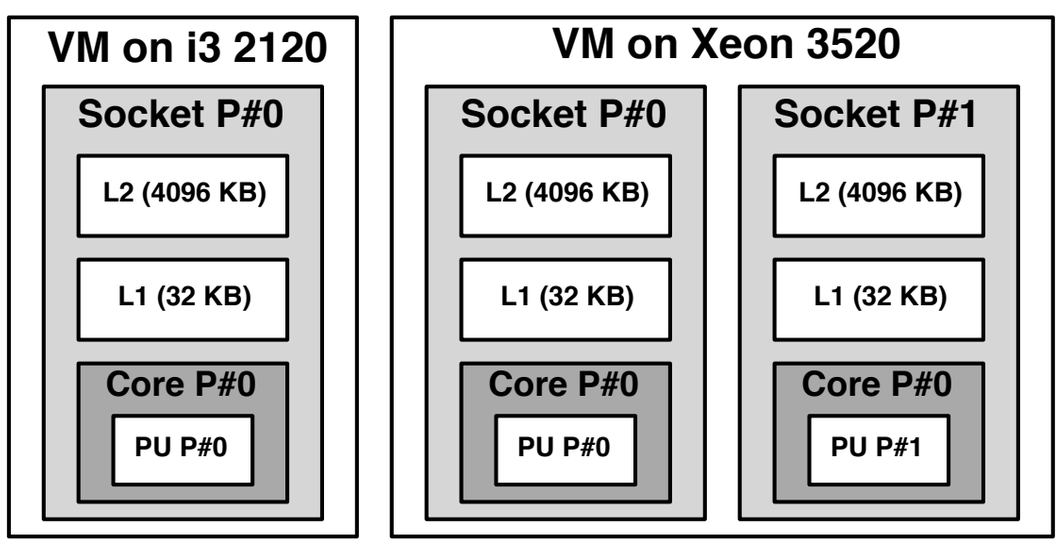
BitWatts: PowerMeter-as-a-Service



CPU Architectures

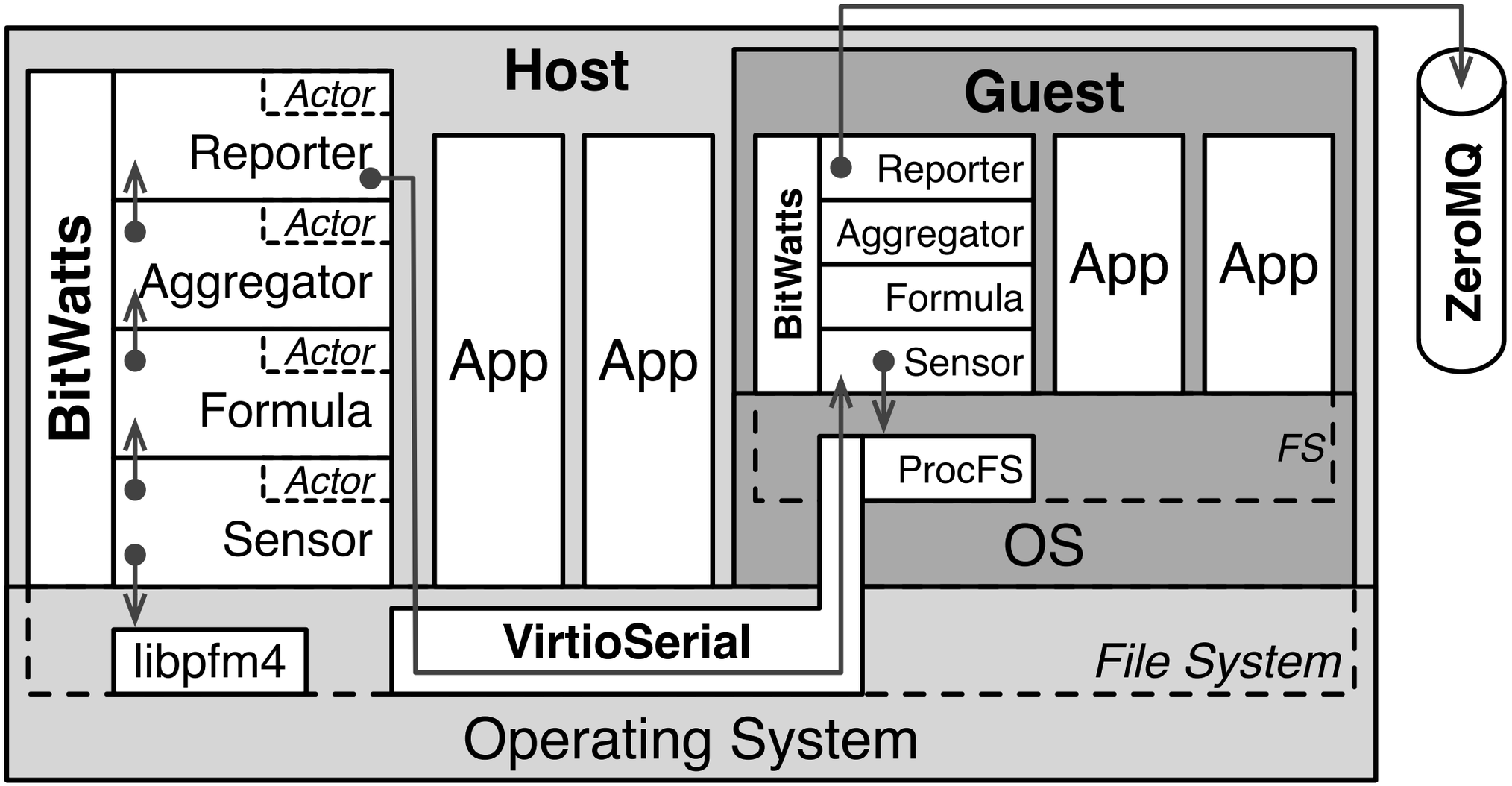


$$P_{cpu}(f, uc^1_{pid} ... uc^N_{pid}) = \sum_{n=1}^{N} P_f(uc^n_{pid}).$$



$$P_{vm}(app) = P_{cpu}(f, uc^1_{vm} ... uc^N_{vm}) \cdot \frac{U_{vm}(app)}{U_{vm}(total)}.$$

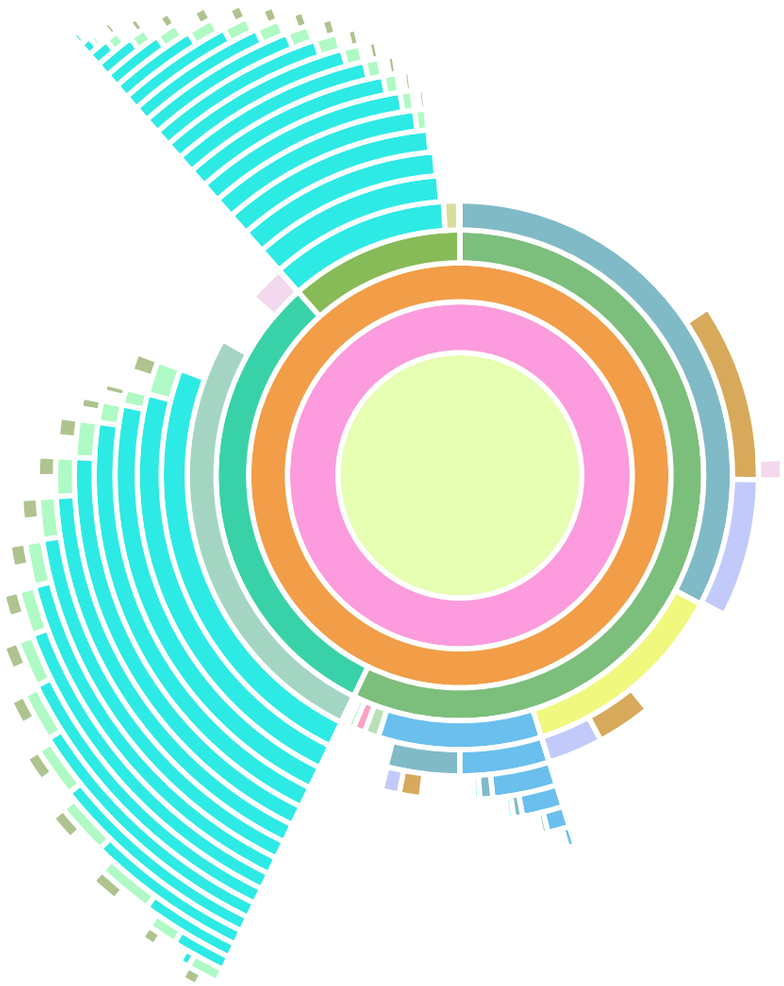
BitWatts: Architecture

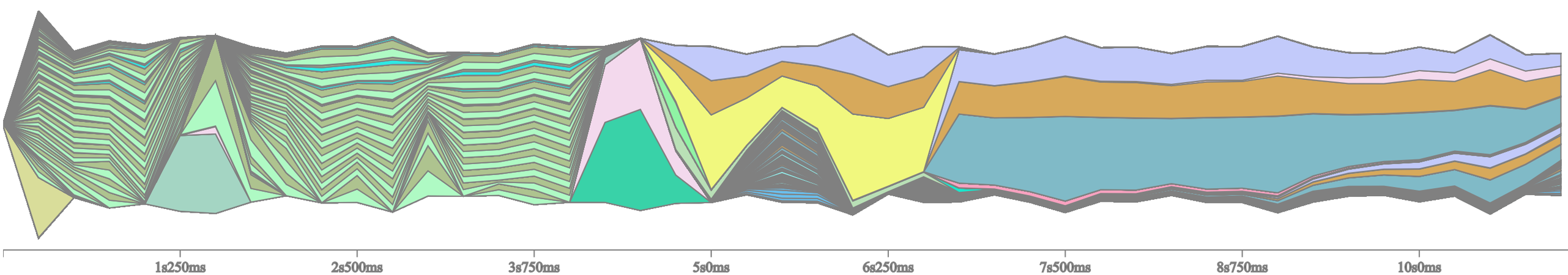


Perspectives: CodEnergy

What about code-level power measurements ?

- Energy debugging (hotspots)
- Including third-party components (disks)





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: romain.rouvoy@univ-lille1.fr