Fine-Grained Energy Attribution for Multi-Tenancy

Hongyu Hè, Michal Friedman, Theodoros Rekatsinas

Systems Group Seminar at ETH Zurich

20 October 2023

ETH zürich

Context

Was a side project sprang from our passion for sustainable computing …

- Preliminary results published at HotCarbon ’23 in July
- On-going collaboration in academia
- Sparked BSc/MSc thesis initiatives
- Adopted by a startup based in Seattle
- Covered by a podcast from the Green Software Foundation

Overview

(1) A theoretical model for thread-level, NUMA-aware energy attribution in multi-tenant environments

(2) Preliminary results on model validity, effectiveness, and robustness to noisy-neighbor effect

(3) Live demo!

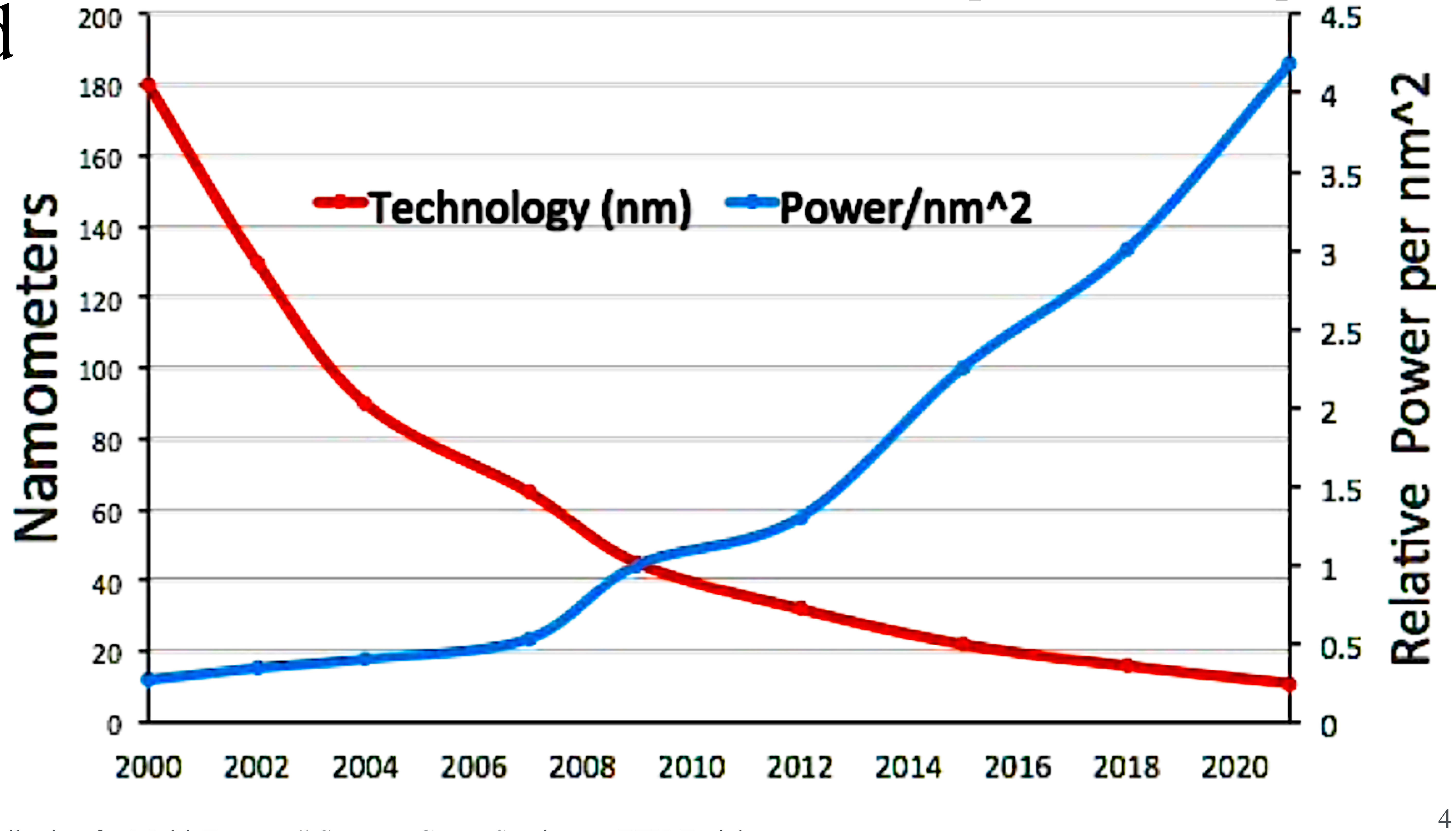
(4) Opportunities and challenges towards energy-aware clouds

(5) Continued efforts to improve ML energy efficiency

Growing Environmental Footprints of Computing

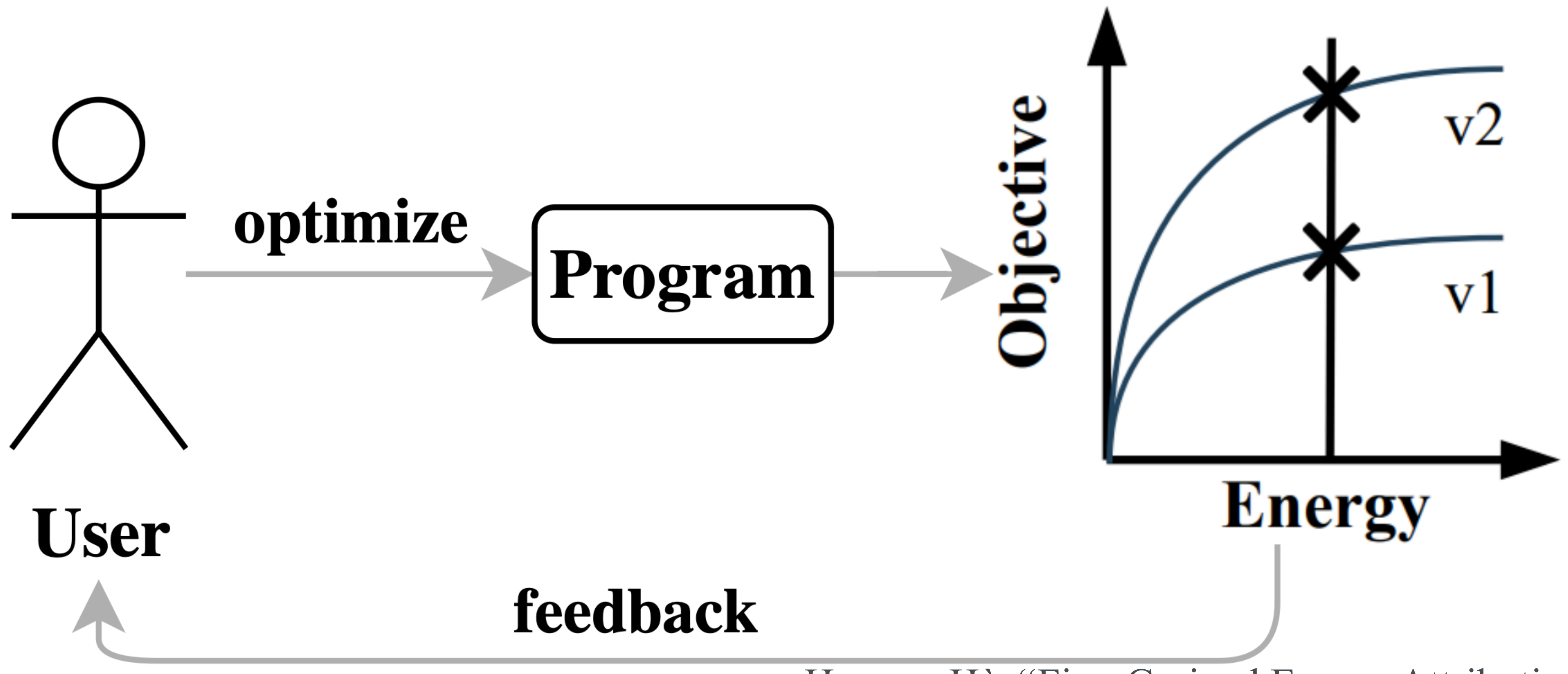
- End of Dennard Scaling → Uncurbed power density
- 900 million tons of CO_2 ≡ Entire aviation industry [Gupta ‘22]
  - 2-4% of worldwide emissions
- Increasing computing demand
  - Networks
  - Machine learning (!)

[Patterson ‘19]



What's wrong with pursuing best performance?

- Performance ≠ Energy efficiency
  - Power (P) $\propto C \cdot V^2 \cdot F + P^{\text{idle}}$, and Energy $= P \cdot T$
  - Race-to-halt ❌ → DVFS
    - Time (T): linear effect
    - Frequency (F) increases with voltage (V): quadratic effect ❗
- Improve **observability**

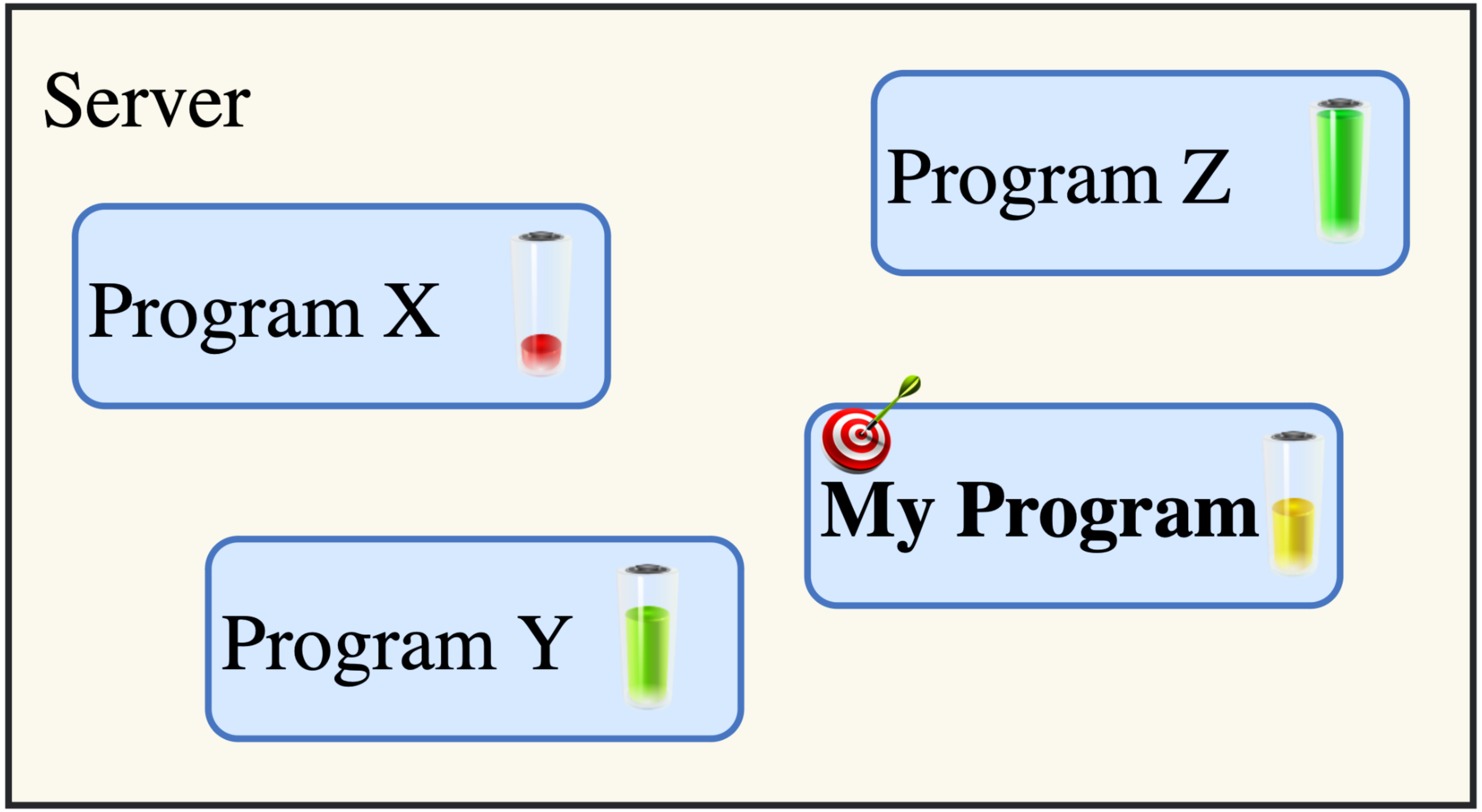


⇒ Per-workload energy attribution

(Focus of this work: CPU and DRAM)

Fine-Grained Software Energy Attribution

- Determine the energy of the **target application** and its subtasks (aka. energy provenance)
- **Exclude** the energy used by collocated jobs (aka. “noisy neighbors”)



Challenges

(1) Nonlinearity in energy modeling

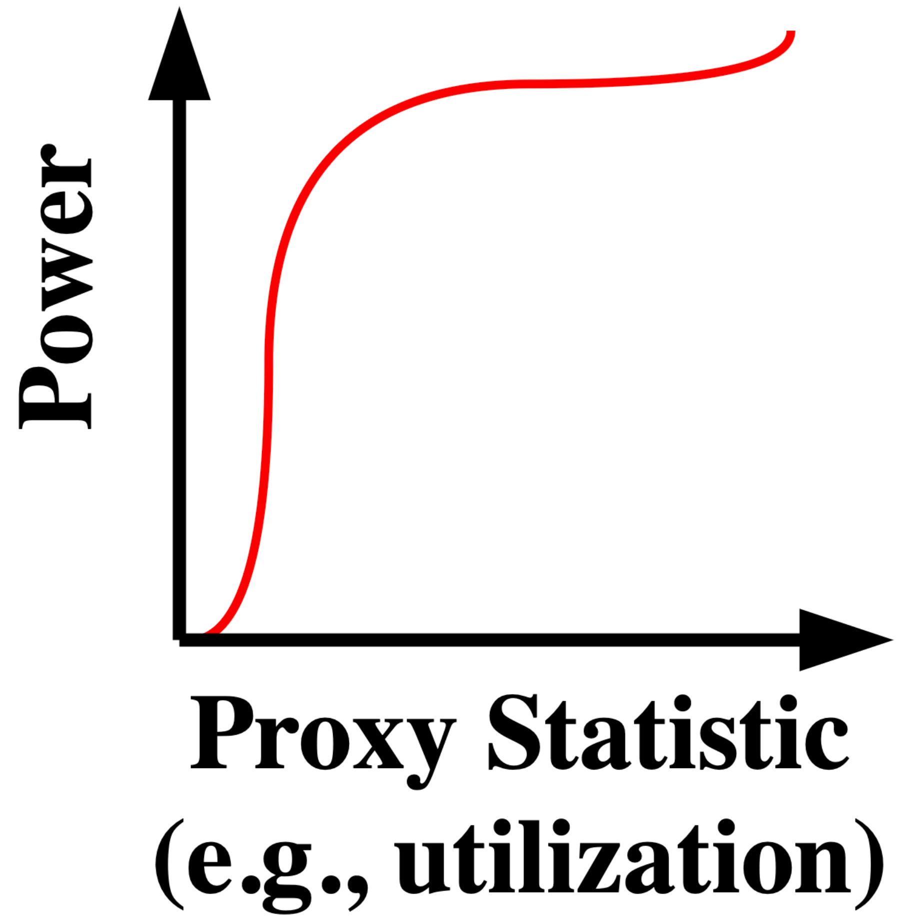
(2) NUMA architecture

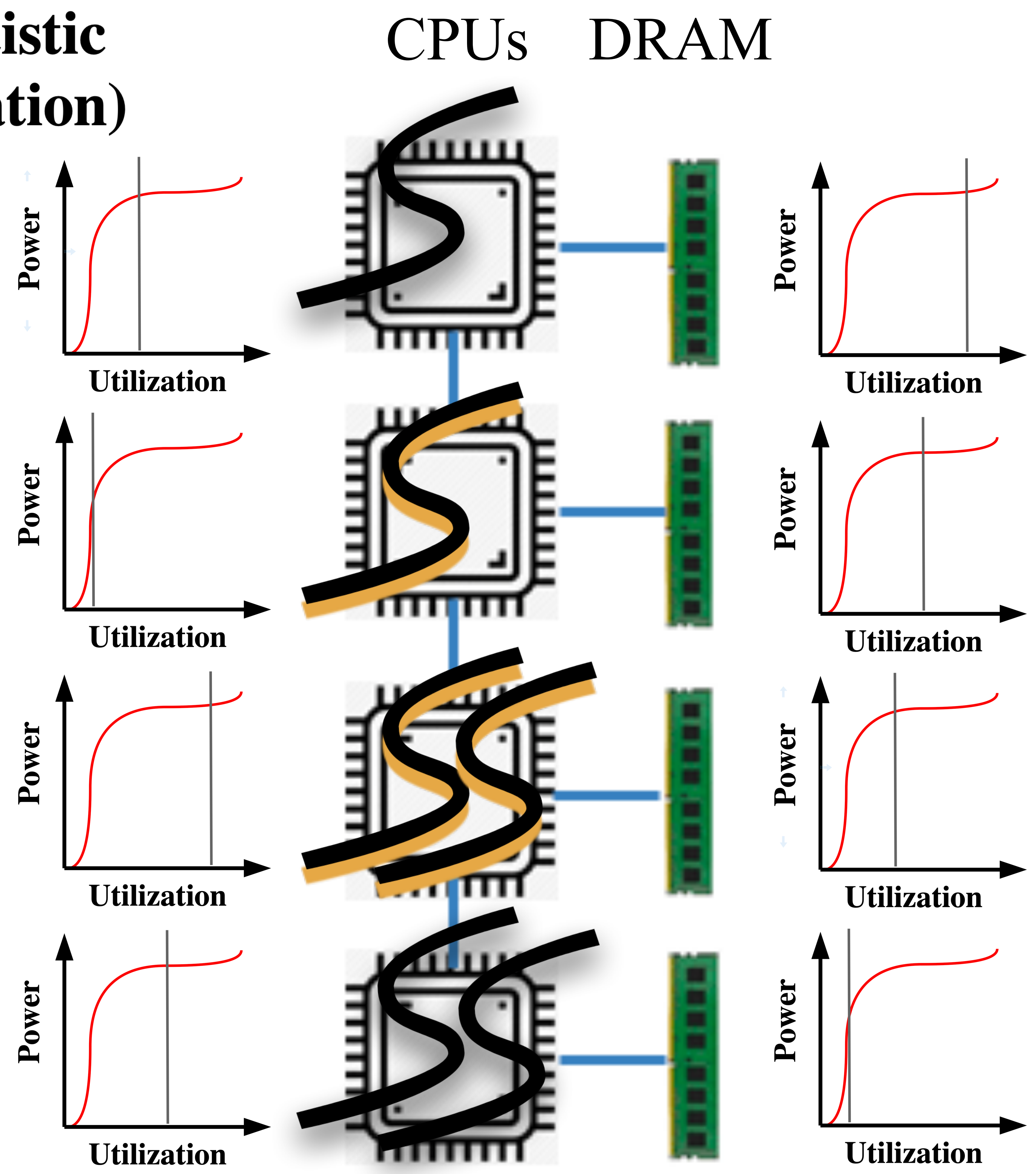
(3) Multithreading

(4) Multi-tenancy

(5) Runtime dynamics

(6) Low-cost measurement





Gaps in Existing Work: Coarse-grained models

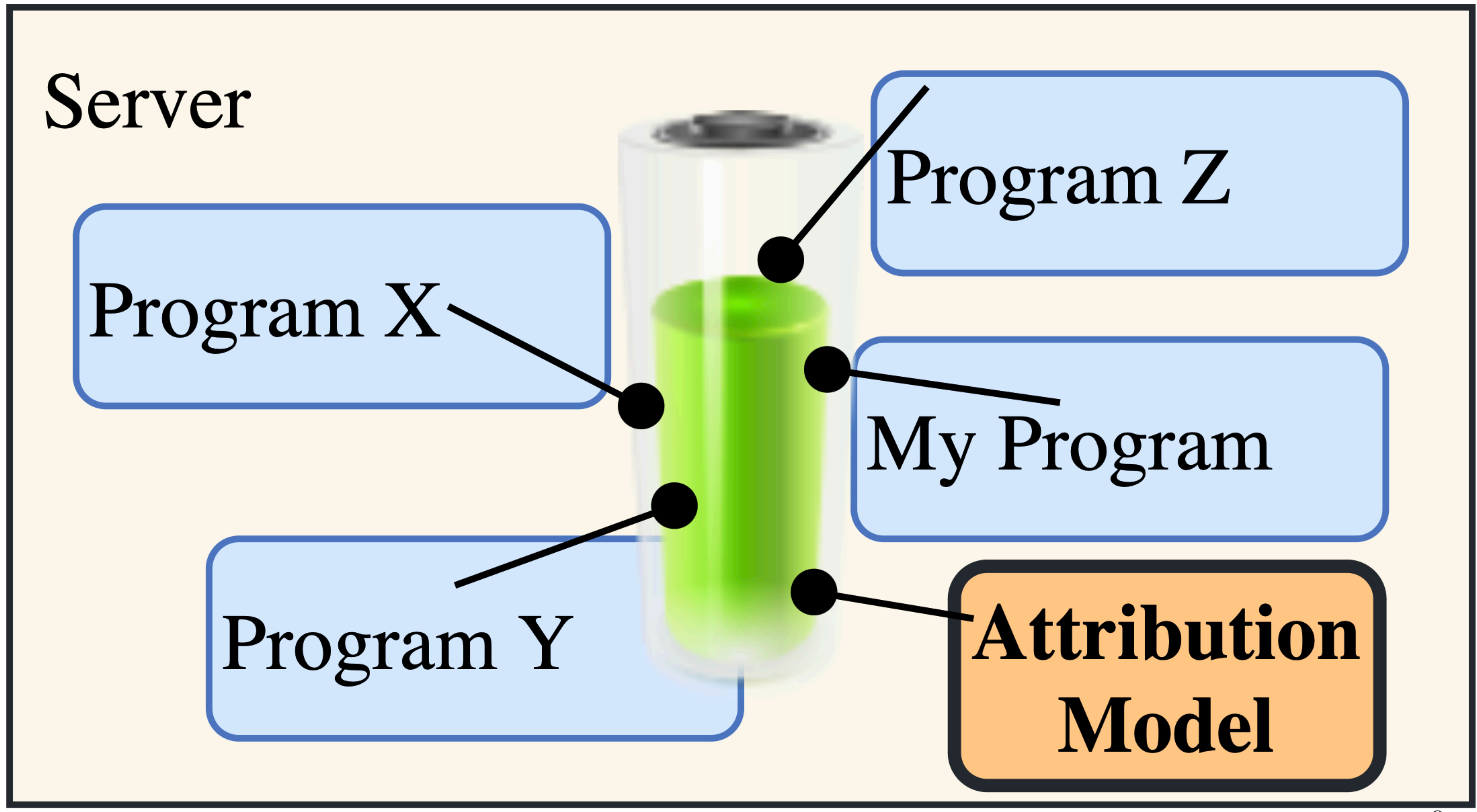
(1) Device-level aggregation → NUMA effects ❌

(2) Process-level accounting → Multithreading ❌

(3) Static tracking → Runtime dynamics ❌

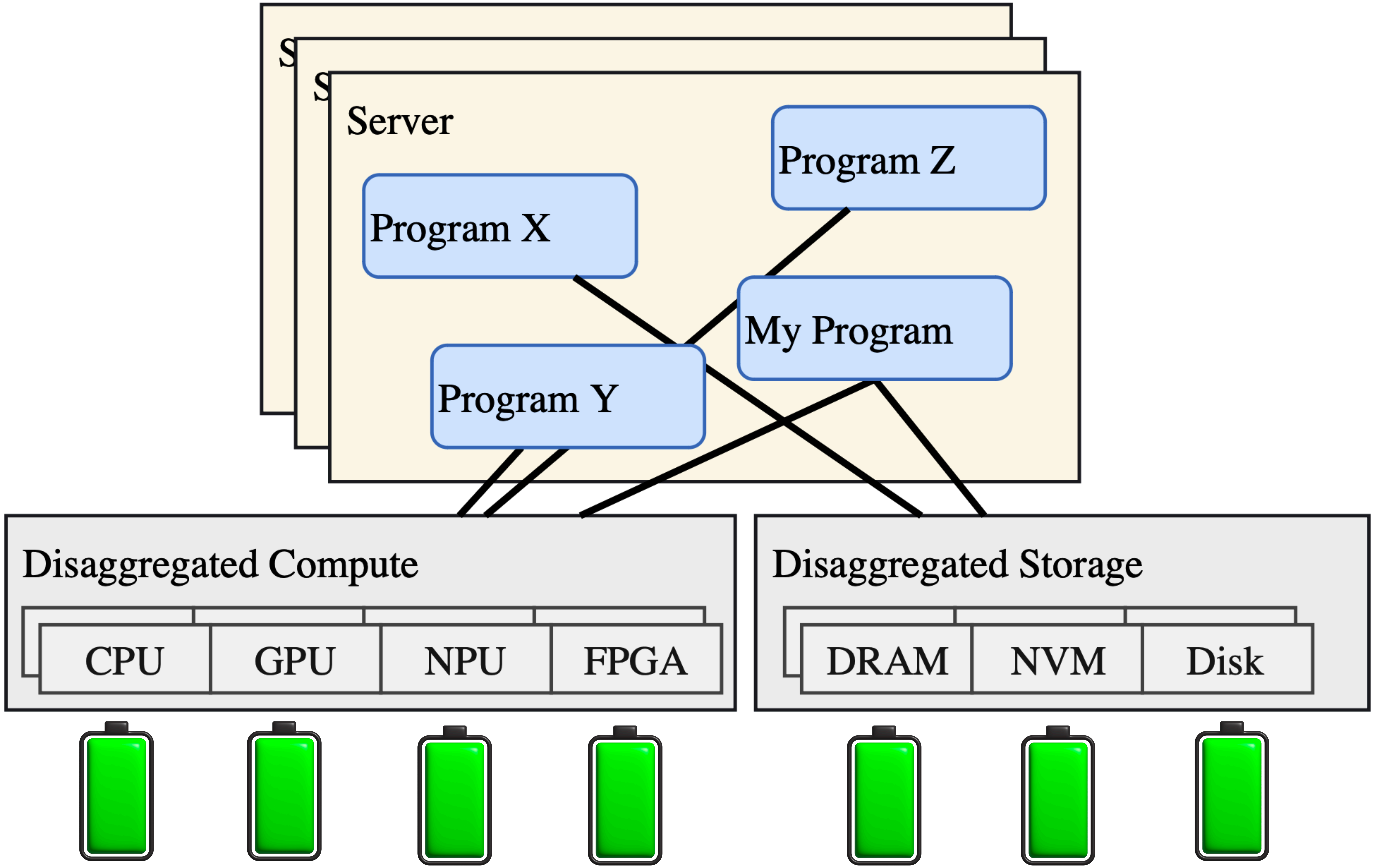
(4) Model energy cost mixed with measurement ❌

(5) Susceptible to noisy-neighbor effect ❌



Challenges ++: Coarse hardware support

Device/user-level energy reporting from hardware



Fine-grained software energy attribution is feasible even with **coarse-grained** hardware support!

Our Fine-Grained Method

(1) Per-socket accounting → NUMA effects ✅

(2) Thread-level attribution → Multithreading ✅

(3) Dynamic runtime tracking ✅

(4) Robust to noisy-neighbor effect → Multi-tenancy ✅

(5) Separation between model energy cost and measurement ✅

NUMA-Aware Thread-Level Model for Multi-Tenancy

$$\left(P^D_{\text{static}}\right)^s = (\text{Sample energy value of } D \text{ for } T_{\text{static}})\,/T_{\text{static}}. \quad (3)$$

$$\left(E^D_{\text{static}}\right)^s = \left(P^D_{\text{static}}\right)^s \cdot T_{\text{sample}}. \quad (4)$$

$$\left(E^{\text{CPU}}_{\Delta}\right)^s = \left(E^{\text{CPU}}_{\text{total}}\right)^s - \left(E^{\text{CPU}}_{\text{static}}\right)^s. \quad (5)$$

$$\mathbb{P}^{\text{CPU}}(s \mid a) \approx \left(\int_{t=t'}^{t'+T_{\text{sample}}} \mathbb{1}_{\{a \text{ on } s\}}\, dt\right) \Big/ T_{\text{sample}}, \quad (6)$$

$$\left(T^{\text{CPU}}_{\mathcal{A}}\right)^s = \mathbb{E}\left[T^{\text{CPU}}_{\mathcal{A}} \mid s\right] \approx \sum_{a \in \mathcal{A}} \mathbb{P}^{\text{CPU}}(s \mid a) \cdot T^{\text{CPU}}_a, \quad (7)$$

$$(T_{\text{total}})^s \leftarrow \text{Total CPU time (kernel + user) of } s \quad (8)$$

$$\left(C^{\text{CPU}}_{\mathcal{A}}\right)^s = \left[\left(T^{\text{CPU}}_{\mathcal{A}}\right)^s / \left(T^{\text{CPU}}_{\text{total}}\right)^s\right]^{\gamma}, \quad (9)$$

$$E^{\text{CPU}}_{\mathcal{A}} = \sum_{s \in S} \left(E^{\text{CPU}}_{\Delta}\right)^s \cdot \left(C^{\text{CPU}}_{\mathcal{A}}\right)^s + \left(E^{\text{CPU}}_{\text{static}}\right)^s. \quad (10)$$

$$(M_{\text{total}})^s \leftarrow \text{Total available NUMA memory on } s \quad (11)$$

$$\left(E^{\text{DRAM}}_{\Delta}\right)^s = \left(E^{\text{DRAM}}_{\text{total}}\right)^s - \left(E^{\text{DRAM}}_{\text{static}}\right)^s. \quad (12)$$

$$\mathbb{P}^{\text{DRAM}}(s \mid a) \approx \mathbb{E}\left[\left\{\left(M^{\Delta t}_a\right)^s \Big/ \left(M^{\Delta t}_{\text{total}}\right)^s\right\}^{T_{\text{sample}}}\right], \quad (13)$$

$$(M_{\mathcal{A}})^s = \mathbb{E}\left[M_{\mathcal{A}} \mid s\right] \approx \sum_{a \in \mathcal{A}} \mathbb{P}^{\text{DRAM}}(s \mid a) \cdot (M_a)^s. \quad (14)$$

$$\left(C^{\text{DRAM}}_{\mathcal{A}}\right)^s = \left[(M_{\mathcal{A}})^s \,/\, (M_{\text{total}})^s\right]^{\sigma}, \quad (15)$$

$$E^{\text{DRAM}}_{\mathcal{A}} = \sum_{s \in S} \left(E^{\text{DRAM}}_{\Delta}\right)^s \cdot \left(C^{\text{DRAM}}_{\mathcal{A}}\right)^s + \left(E^{\text{DRAM}}_{\text{static}}\right)^s. \quad (16)$$

NUMA-Aware Thread-Level Model for Multi-Tenancy

$$\left(P_{\text{static}}^{D}\right)^{s} = \left(\text{Sample energy value of } D \text{ for } T_{\text{static}}\right)/T_{\text{static}}. \quad (3)$$

$$\left(E_{\text{static}}^{D}\right)^{s} = \left(P_{\text{static}}^{D}\right)^{s} \cdot T_{\text{sample}}. \quad (4)$$

$$\left(E_{\Delta}^{\text{CPU}}\right)^{s} = \left(E_{\text{total}}^{\text{CPU}}\right)^{s} - \left(E_{\text{static}}^{\text{CPU}}\right)^{s} \quad (5)$$

$$\mathbb{P}^{\text{CPU}}(s \mid a) \approx \left(\int_{t=t'}^{t'+T_{\text{sample}}} \mathbb{1}_{\{a \text{ on } s\}}\, dt\right) \Big/ T_{\text{sample}}, \quad (6)$$

$$\left(T_{\mathcal{A}}^{\text{CPU}}\right)^{s} = \mathbb{E}\left[T_{\mathcal{A}}^{\text{CPU}} \mid s\right] \approx \sum_{a \in \mathcal{A}} \mathbb{P}^{\text{CPU}}(s \mid a) \cdot T_{a}^{\text{CPU}}, \quad (7)$$

$$\left(T_{\text{total}}\right)^{s} \leftarrow \text{Total CPU time (kernel + user) of } s \quad (8)$$

$$\left(C_{\mathcal{A}}^{\text{CPU}}\right)^{s} = \left[\left(T_{\mathcal{A}}^{\text{CPU}}\right)^{s} \Big/ \left(T_{\text{total}}^{\text{CPU}}\right)^{s}\right]^{\sigma}, \quad (9)$$

$$E_{\mathcal{A}}^{\text{CPU}} = \sum_{s \in S} \left(E_{\Delta}^{\text{CPU}}\right)^{s} \cdot \left(C_{\mathcal{A}}^{\text{CPU}}\right)^{s} + \left(E_{\text{static}}^{\text{CPU}}\right)^{s}. \quad (10)$$

$$\left(M_{\text{total}}\right)^{s} \leftarrow \text{Total available NUMA memory on } s \quad (11)$$

$$\left(E_{\Delta}^{\text{DRAM}}\right)^{s} = \left(E_{\text{total}}^{\text{DRAM}}\right)^{s} - \left(E_{\text{static}}^{\text{DRAM}}\right)^{s}. \quad (12)$$

$$\mathbb{P}^{\text{DRAM}}(s \mid a) \approx \mathbb{E}\left[\left\{\left(M_{a}^{\Delta t}\right)^{s} \Big/ \left(M_{\text{total}}^{\Delta t}\right)^{s}\right\}^{T_{\text{sample}}}\right], \quad (13)$$

$$\left(M_{\mathcal{A}}\right)^{s} = \mathbb{E}\left[M_{\mathcal{A}} \mid s\right] \approx \sum_{a \in \mathcal{A}} \mathbb{P}^{\text{DRAM}}(s \mid a) \cdot \left(M_{a}\right)^{s}. \quad (14)$$

$$\left(C_{\mathcal{A}}^{\text{DRAM}}\right)^{s} = \left[\left(M_{\mathcal{A}}\right)^{s} \Big/ \left(M_{\text{total}}\right)^{s}\right]^{\sigma}, \quad (15)$$

$$E_{\mathcal{A}}^{\text{DRAM}} = \sum_{s \in S} \left(E_{\Delta}^{\text{DRAM}}\right)^{s} \cdot \left(C_{\mathcal{A}}^{\text{DRAM}}\right)^{s} + \left(E_{\text{static}}^{\text{DRAM}}\right)^{s}. \quad (16)$$

Our model fits on 1 slide

This talk:
(1) Fine-grained CPU energy attribution
(2) Credit-based per-workload accounting

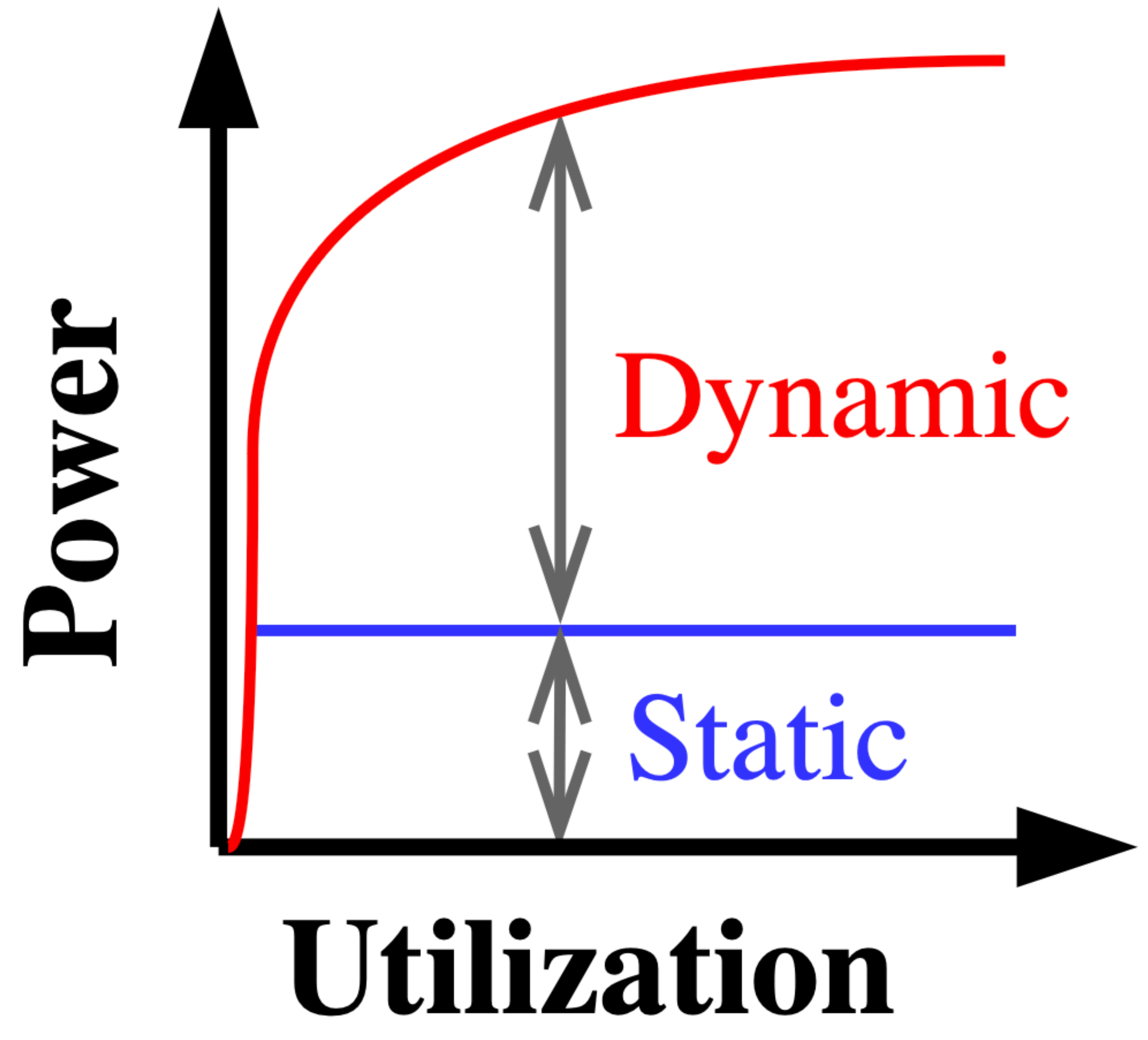
High-Level Intuition

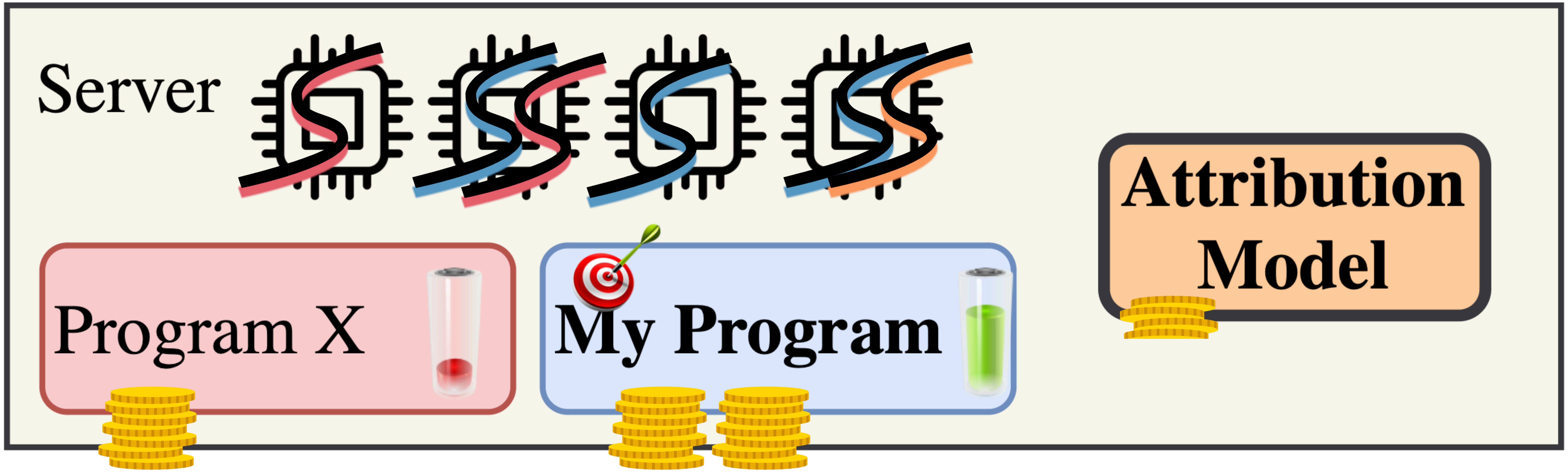
(1) Separate static vs. dynamic power (pitfall)

(2) Per-thread and per-socket accounting

(3) *'Energy credit'* based on *exclusive* resource usage

(4) Separate energy cost of the model itself





Fine-Grained CPU Energy Attribution

(1) Extract dynamic energy (E_Δ^{CPU}) from the total **for each socket** (s):

$$\left(E_\Delta^{\text{CPU}}\right)^s = \left(E_{\text{total}}^{\text{CPU}}\right)^s - \left(E_{\text{static}}^{\text{CPU}}\right)^s$$

(2) Estimate *CPU residence rate* for each **thread/process** (a) of application ($\mathscr{A}$):

$$\mathbb{P}^{\text{CPU}}(s \mid a) \approx \left(\int_{t=t'}^{t'+T_{\text{sample}}} 1\{a \text{ on } s\} dt \right) \Big/ T_{\text{sample}}$$

(3) Approximate CPU time of $\mathscr{A}$ on s with **conditional expectation**:

$$\left(T_{\mathscr{A}}^{\text{CPU}}\right)^s = \mathbb{E}\left[T_{\mathscr{A}}^{\text{CPU}} \mid s\right] \approx \sum_{a \in \mathscr{A}} \mathbb{P}^{\text{CPU}}(s \mid a) \cdot T_a^{\text{CPU}}$$

Per-Workload CPU Energy Credit

(4) Obtain system-wide CPU time (kernel+user) of s: $(T_{\text{total}})^s$

(5) Compute **CPU energy credit** ($C_{\mathscr{A}}^{\text{CPU}}$) for $\mathscr{A}$:

$$C_{\mathscr{A}}^{\text{CPU}} = \sum_{s \in S} \left[\left(T_{\mathscr{A}}^{\text{CPU}}\right)^s / \left(T_{\text{total}}^{\text{CPU}}\right)^s \right]^{\gamma}, \text{ where } \gamma \in [0,1]$$

(6) Attribute the Δ energy to $\mathscr{A}$ ($E_{\mathscr{A}}^{\text{CPU}}$) using $C_{\mathscr{A}}^{\text{CPU}}$:

$$E_{\mathscr{A}}^{\text{CPU}} = \sum_{s \in S} \left(E_{\Delta}^{\text{CPU}}\right)^s \cdot \left(C_{\mathscr{A}}^{\text{CPU}}\right)^s + \left(E_{\text{static}}^{\text{CPU}}\right)^s$$

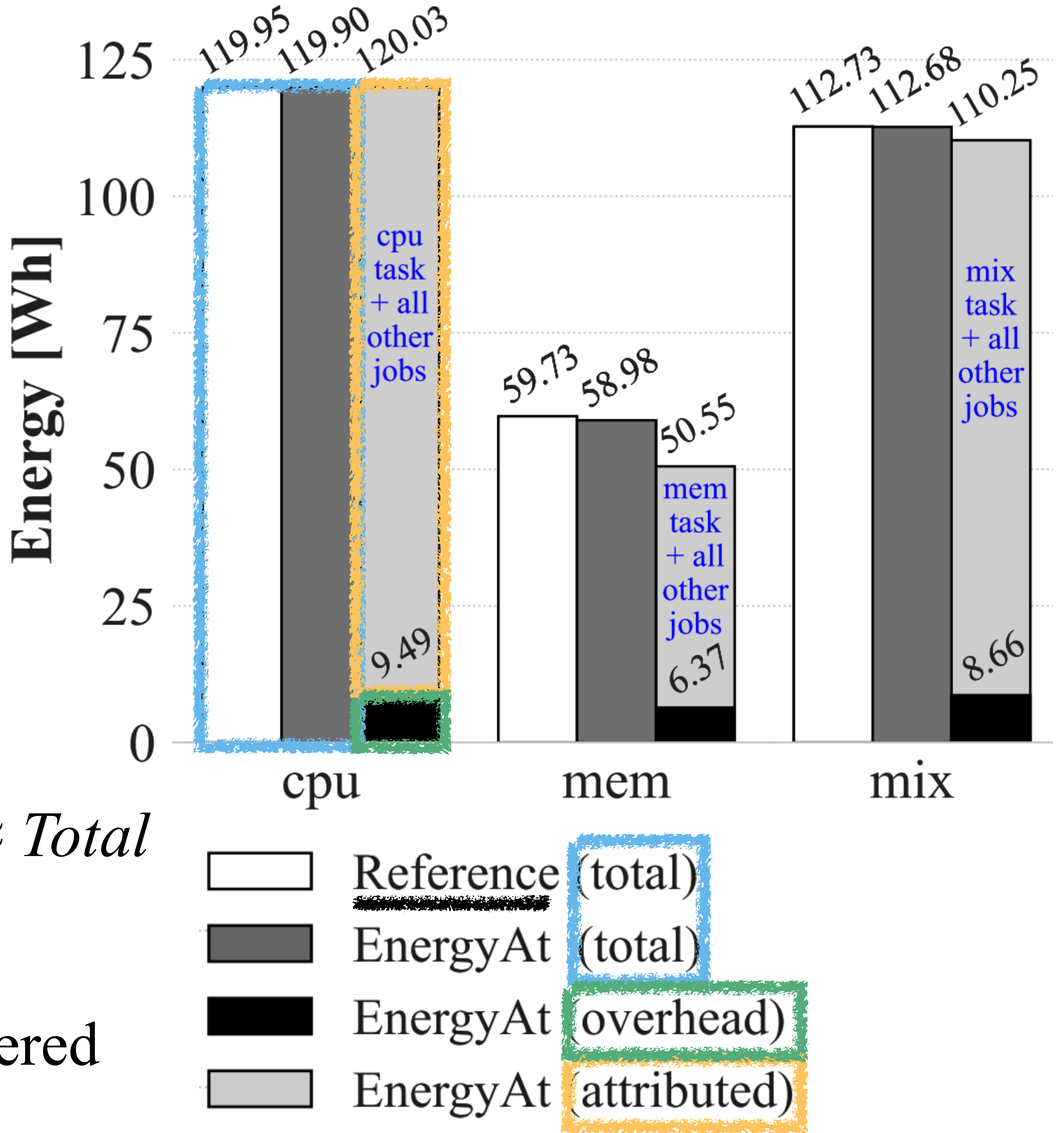
(DRAM model works similarly)

Evaluation Setup

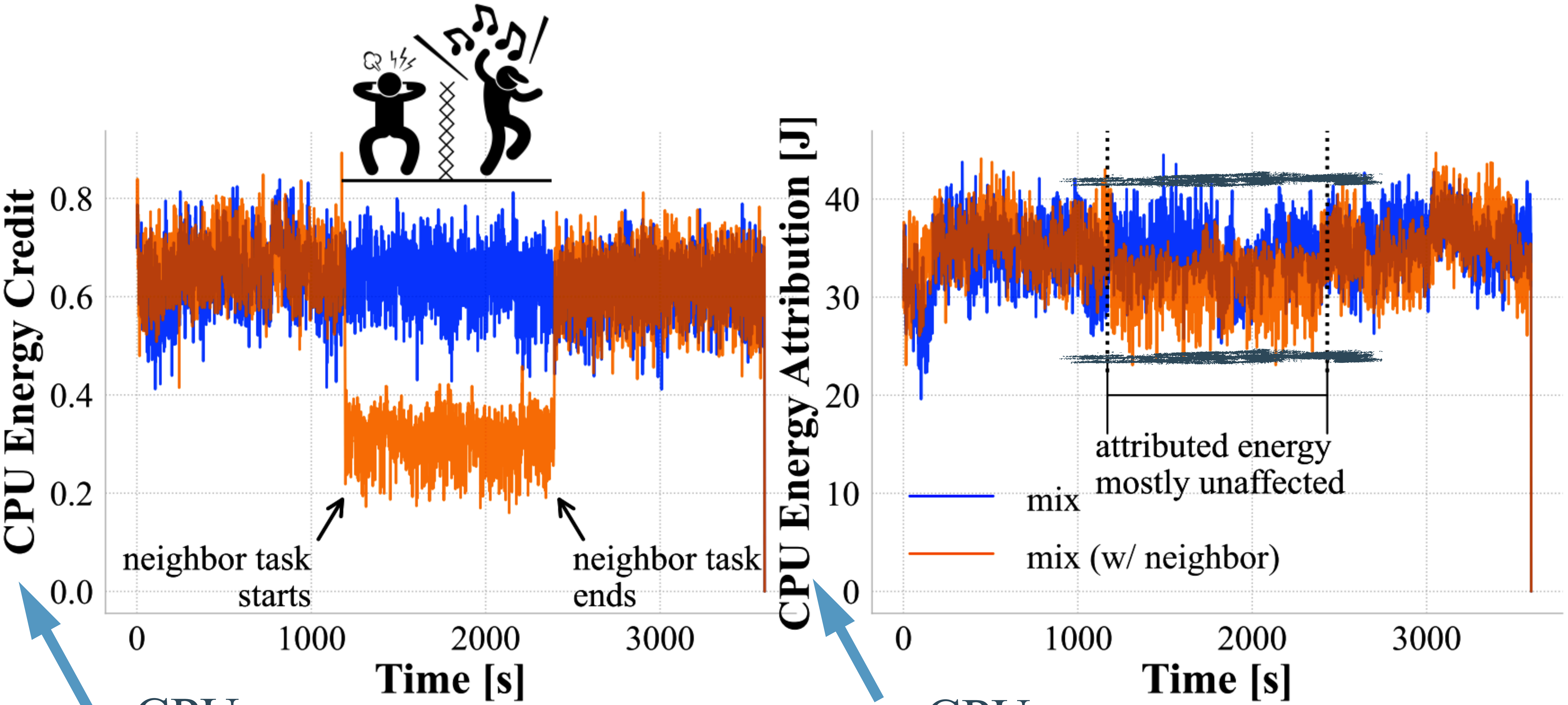
- Prototype: EnergAt (https://github.com/HongyuHe/energat)
- Microbenchmarks (target applications):
  - `cpu`: CPU utilization 0→100% (equal # of threads and processes)
  - `mem`: DRAM usage 0→100% (one process)
  - `mix`: Both CPU and DRAM at ~50% (using `cpu` and `mem` methods)
  - `mix(w/ neighbor)`: 2 `mix` workloads (the target and noisy neighbor)
- Testbed: Intel Xeon E5-2630
  - Dual-socket; 32 (logical) cores in total; 64 GiB of DRAM

Model Validation

- Methodology
  - Validation by summation [Shen ’ 13]
- Reference (total)
  - Modified Firefox plugin
- Observations
  - *Total* value ≈ Reference value
  - Sum of attributed energies + cost ≈ *Total*
  - `mem`: underestimation
    - Only private memories are considered



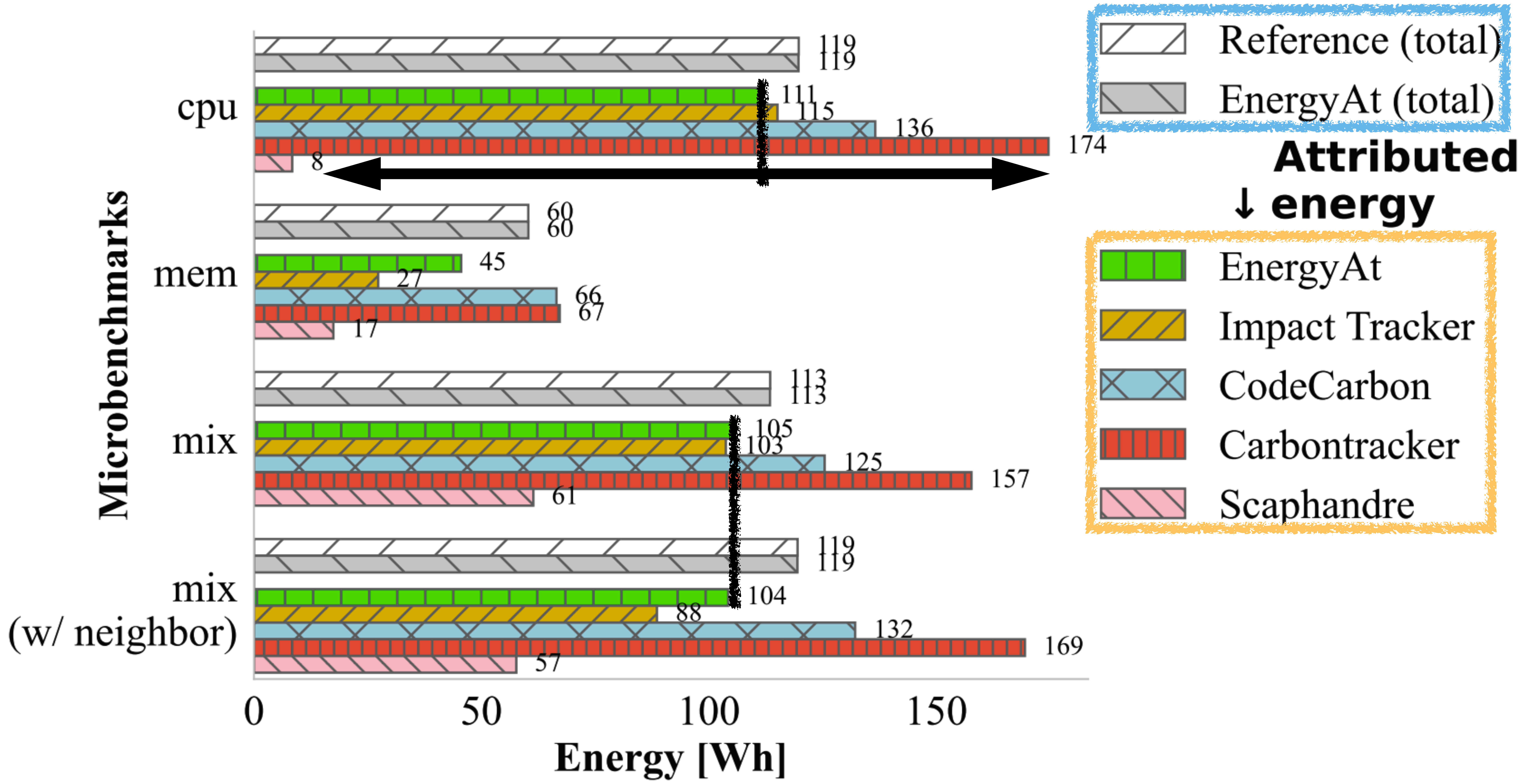
Energy Credit In-Situ



$C_{\mathscr{A}}^{\mathrm{CPU}}$

$E_{\mathscr{A}}^{\mathrm{CPU}}$

Comparing with Prior Work



Live Demo! 🙏

Towards Energy-Aware Heterogeneous Clouds

(1) HW-SW interface for **secure** and **efficient** energy reporting

(2) Energy attribution for **cloud services** (e.g., FaaS and DBaaS)

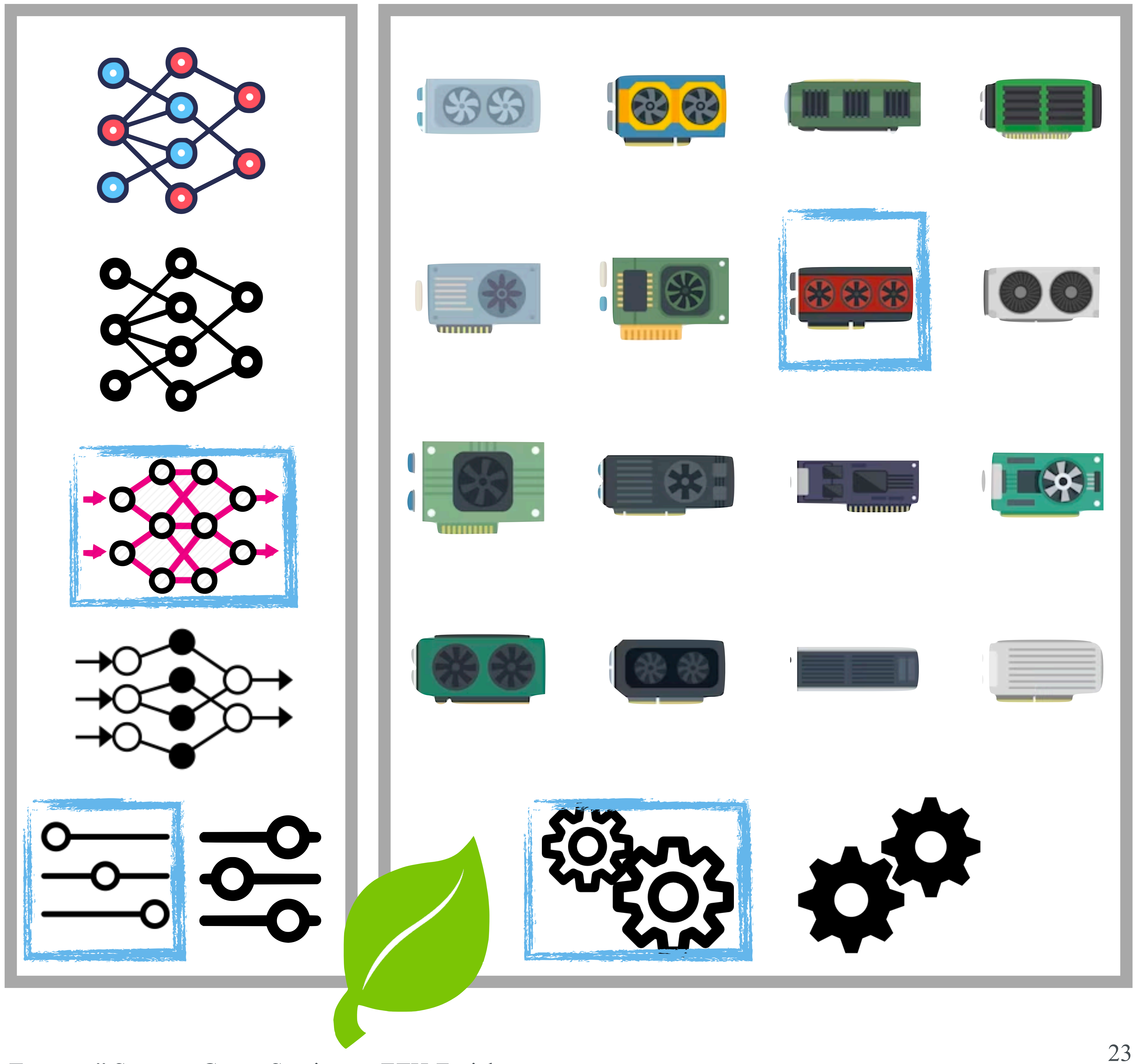
- Virtualization 😈
- Heterogeneous devices
- Energy-based billing

(3) **NUMA-aware** energy optimization

(4) Revisit traditional algorithms in terms of **energy efficiency**

Choosing the 'Best' Configuration for ML Training

- Performance ≠ Energy efficiency
  - Applies to ML training [You '23]
- Most energy-efficient combo of
  - Model and its config
  - GPU and its config
- Automatic decision-making
  - **Without** enumeration



Computer Science
UNIVERSITY OF TORONTO

Summary

- **Fine-grained software energy attribution is feasible even with coarse hardware support!**
- Contributions
  - Thread-level, NUMA-aware energy attribution for multi-tenancy
  - Validation of validity, effectiveness, and robustness to noisy-neighbor effect
  - Opportunities and challenges towards energy-aware clouds
- Code: https://github.com/HongyuHe/energat
  - `sudo pip install energat`
- Paper:

Big shout out to Shail David for his help in revising the paper!

Email: honghe@inf.ethz.ch
Web: hongyu.nl

Backup Slides

Per-Socket Accounting Example

- 2 sockets w/ total CPU times and energies: (100 s, 30 J), (200 s, 50 J)
- 1 task with CPU times on each socket: 20 s and 180 s
- Attribution
  - (20/100 × 30 + 180/200 × 50) J ✅
  - [(20+180)/(100+200) × (30 + 50)] J ❌

⇒ CPU time cannot be the sole proxy, ignoring the NUMA effect

Sampled Hardware Counters

Counters	Metrics
Intel RAPL package and DRAM domains	Accumulated energy consumption of CPU packages and DRAM (through the `sysfs` interface)
Intel RAPL maximum counter values	Maximum ranges of each domain for detecting and mitigating counter overflow
Memory statistics from the `numactl` package	Total, used, and private memory statistics for processes and the operating system on a per-NUMA-node basis
`/proc/*/task/*/stat`	User and kernel times for each task and its children
`CLK_TCK` value	Number of clock ticks per second

Limitations

(1) Not considering other pertinent factors

- E.g., shared memory, I/O, and caches

(2) Validation by summation

- No insight into individual accounting

(3) Non-negligible energy overhead

- Up to 9.5% (when tracing all jobs on a server)

(4) Evaluation on real workloads