Machine Learning for *Energy-Performance*-aware Scheduling

Peter Hu† (zh369), Yifei Shi† (ys690)

MLPW group project

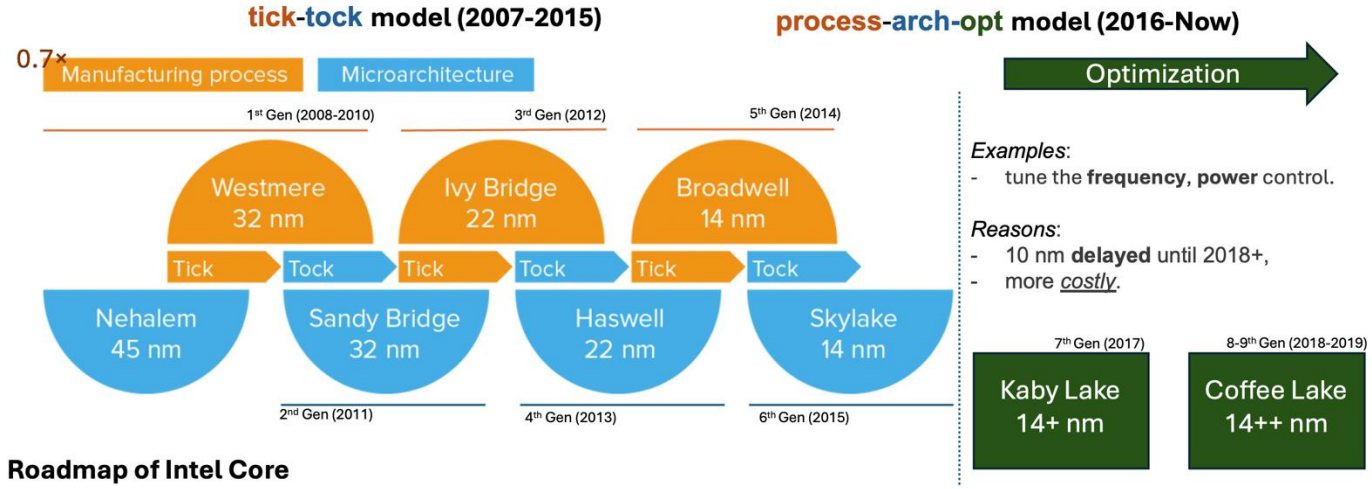
ACS 2025-26

† denotes equal contribution.

Motivation

- This project focuses on **semiconductor processors design**.
- The end of *Dennard scaling* (see table), and more recently *fundamental physical* limitations.
  - power increases due to increased **leakage currents**, leading to *more* architectural design required.
- The research question: Can machine learning autonomously explore the optimal strategy to balance energy and performance in heterogeneous systems?
  - Simulation of processors, tasks and different schedulers,
  - Define the sub-metrics, how to combine to the overall metric,
  - Emulation for the best scheduler policies, processor frequency setup, etc.

Era	Design Reality
Before ~2005	Shrink transistors and performance improves “for free”
After ~2005	Shrinking provides more transistors, but designers must choose how to use them



Roadmap of Intel Core

Simulation: task and processor

Task definition. The system consists of a set of tasks $\mathcal{T} = \{\tau_i\}_{i=1}^{N_{\text{task}}}$.

$$\tau = (t_a, t_f, p, N_{\text{IC}}, E)$$

Table 1: Task attributes and their data type.

	task id	arrival time	instruction count	priority	energy	finish time
Type	int	float	float	int	float	Optional[float]

Heterogeneous multi-core design. The system consists of a set of processors $\mathcal{P} = \{P_i\}_{i=1}^{N_{\text{cores}}}$.

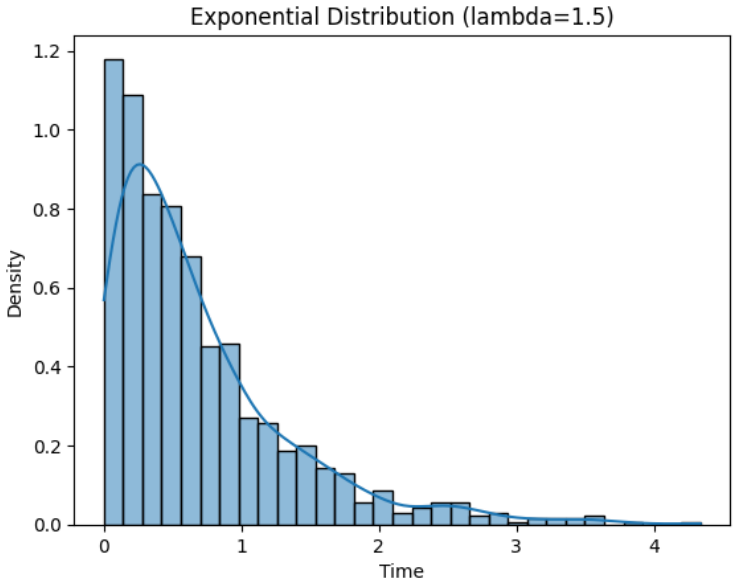
Each processor is set to **a fixed frequency f** during the whole execution
(i.e., Dynamic Voltage and Frequency Scaling not considered here).

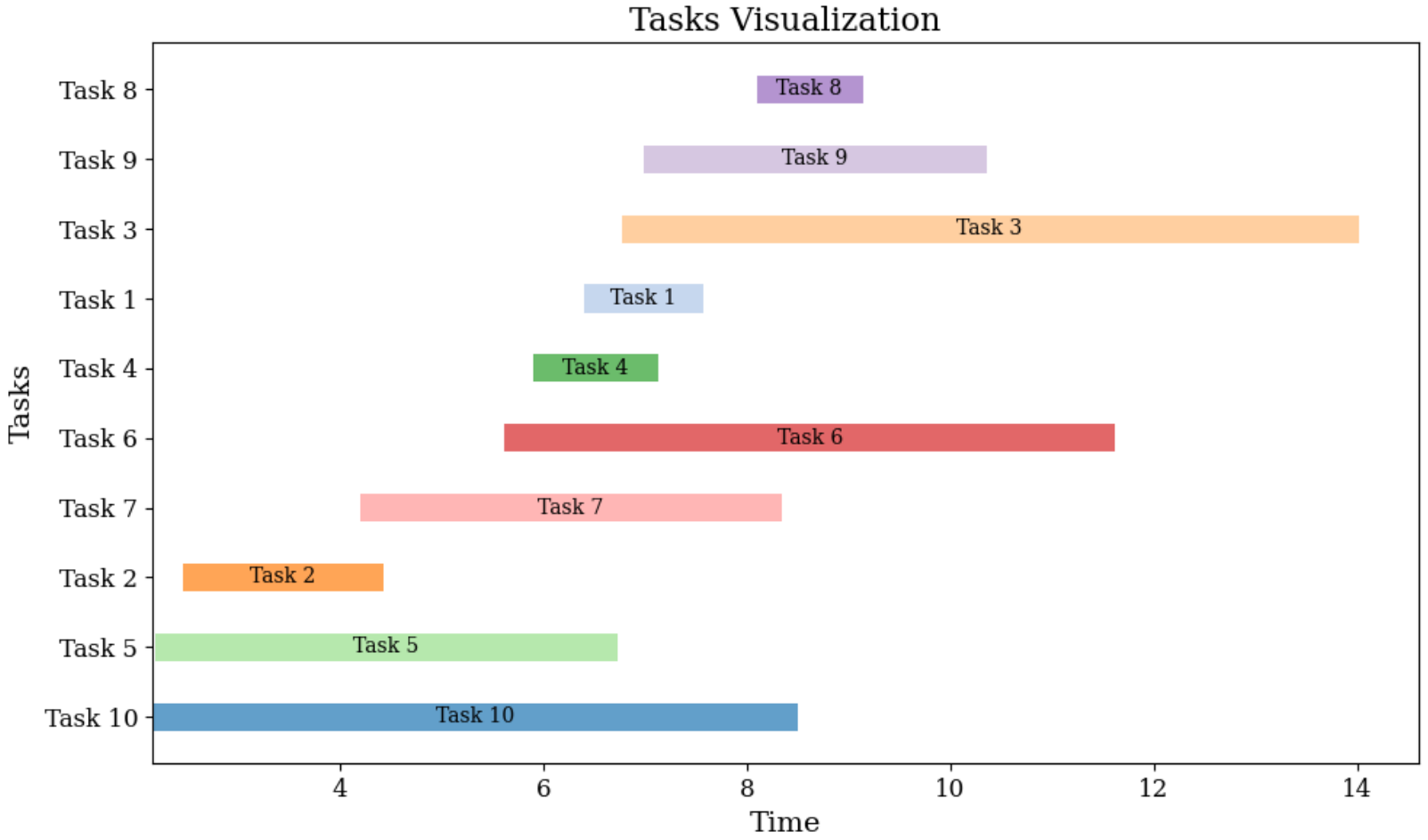
However, there are **different** types of processors {little, Medium, LARGE},
each with its ***own*** operating frequencies that must be explored:
$[f_{little}, f_{Medium}, f_{LARGE}]$.

Simulation: task factory

For each task in the given set, its attributes are **randomly** initialized over a fixed region by a data factory class.

The tasks follow the **Poisson process,** whose waiting time modelled by **exponential** distribution.





Tasks factory visualization.

Simulation: scheduler

Task, Processor, **Scheduler**, Per-run metric.

Scheduler design. The core goal of schedulers is to dispatch tasks to appropriate processors. There are various task assignment algorithms in the literature, which differ by their simplicity, overhead, fairness, and whether they are preemptive. We implement the following scheduling methods (also shown in Table 2),

scheduler choice

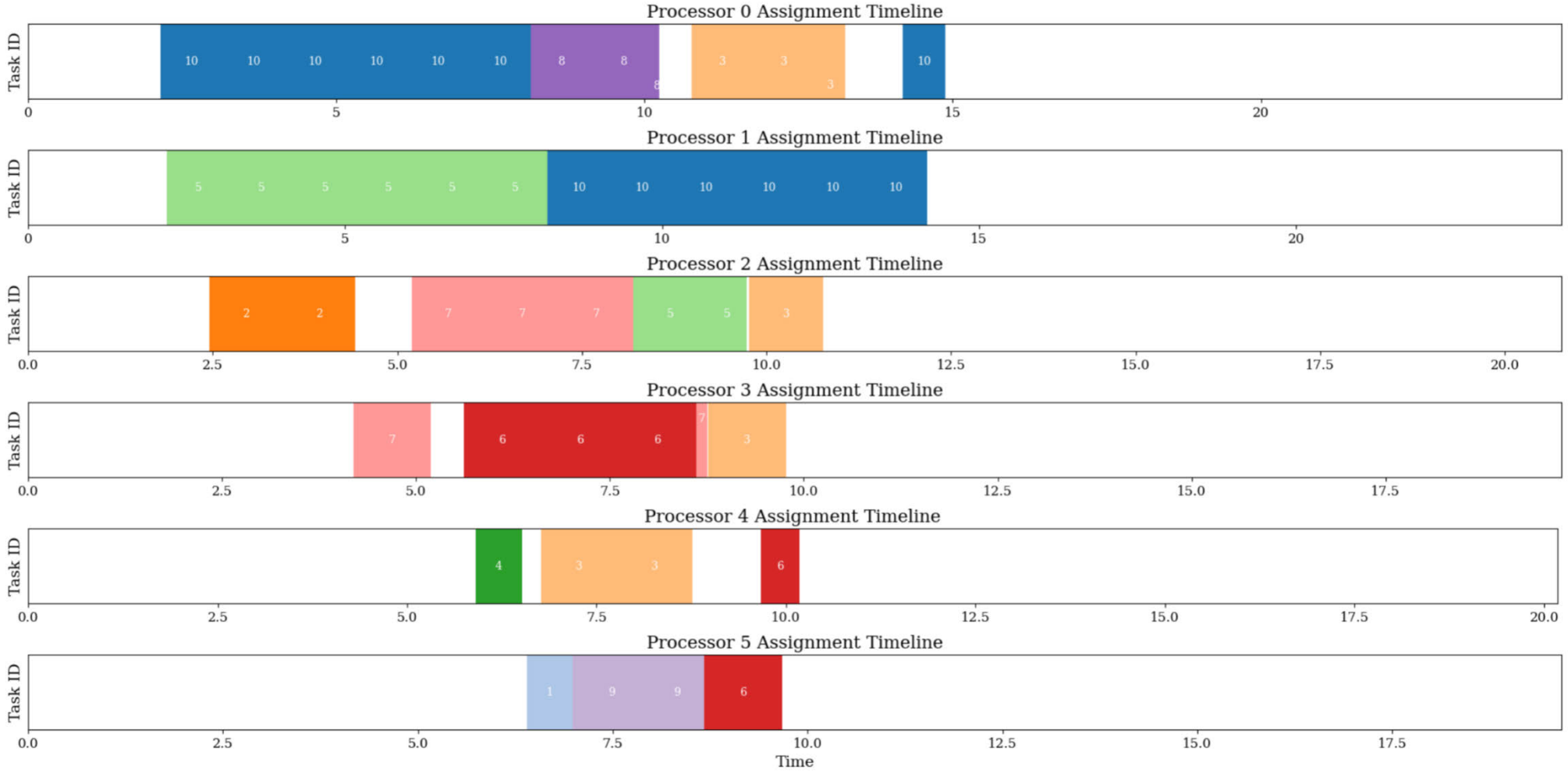
- First-Come, First-Served (FCFS), also known as First-In, First-Out (FIFO). Tasks are executed strictly according to the order of arrival time. The pros are its predictability, starvation-free. However, short tasks that arrive late will wait longer time, which is termed the Convoy effect.
- Round Robin (RR) is a specific implementation of time-sliced or quantum based scheduling (Figure 1b), where tasks are preempted from the processors after the quantum used up. This saves computation time for short tasks, but keeps the average turnaround time larger.
- Priority-based scheduling. Tasks are dispatched based on the priority level. It can either be preemptive or non-preemptive. It favors high-priority tasks, but might starve those low-priority processes.

per-scheduler design choices

Table 2: Comparison of FCFS, Round Robin, and priority-based scheduling algorithms.

Features	**FCFS**	**Round Robin (RR)**	**Priority-based**
decision basis	arrival time	time quantum	priority level
preemptive	×	✓	○
starvation	×	×	✓
context switchg	↓	↑	○
use case	batch processing	time-sharing	real-time / critical

Simulation: scheduler



Round robin scheduler assignments for six heterogeneous processors.

Simulation: env, logging, etc.

Simulation environment and logging. We adopt `SimPy` [13], a process-based discrete-event framework, as our backend simulation environment. It supports waiting, interrupting a process and shared resources. Our simulator is divided into `Task`, `Processors`, `Schedulers`, and `Simulators` packages, where details are described as above. For efficient verification and debugging, we deploy `logging` to track every event of the simulation.

An example log output generated via `logging`.

```
INFO:Task factory:Created <N> tasks. Processor <p_id>:Frequency = <f>.
INFO:Round Robin scheduler:Initialized with quantum = <q>.
INFO:Simulator:Task <t_id> arrived at time <t>, with instruction <count>.
INFO:Round Robin scheduler:Task <t_id> quantum expired, enqueuing again.
```

Simulation: latency metric

Task, Processor, Scheduler, **Per-run metric**.

Latency and frequency modelling.

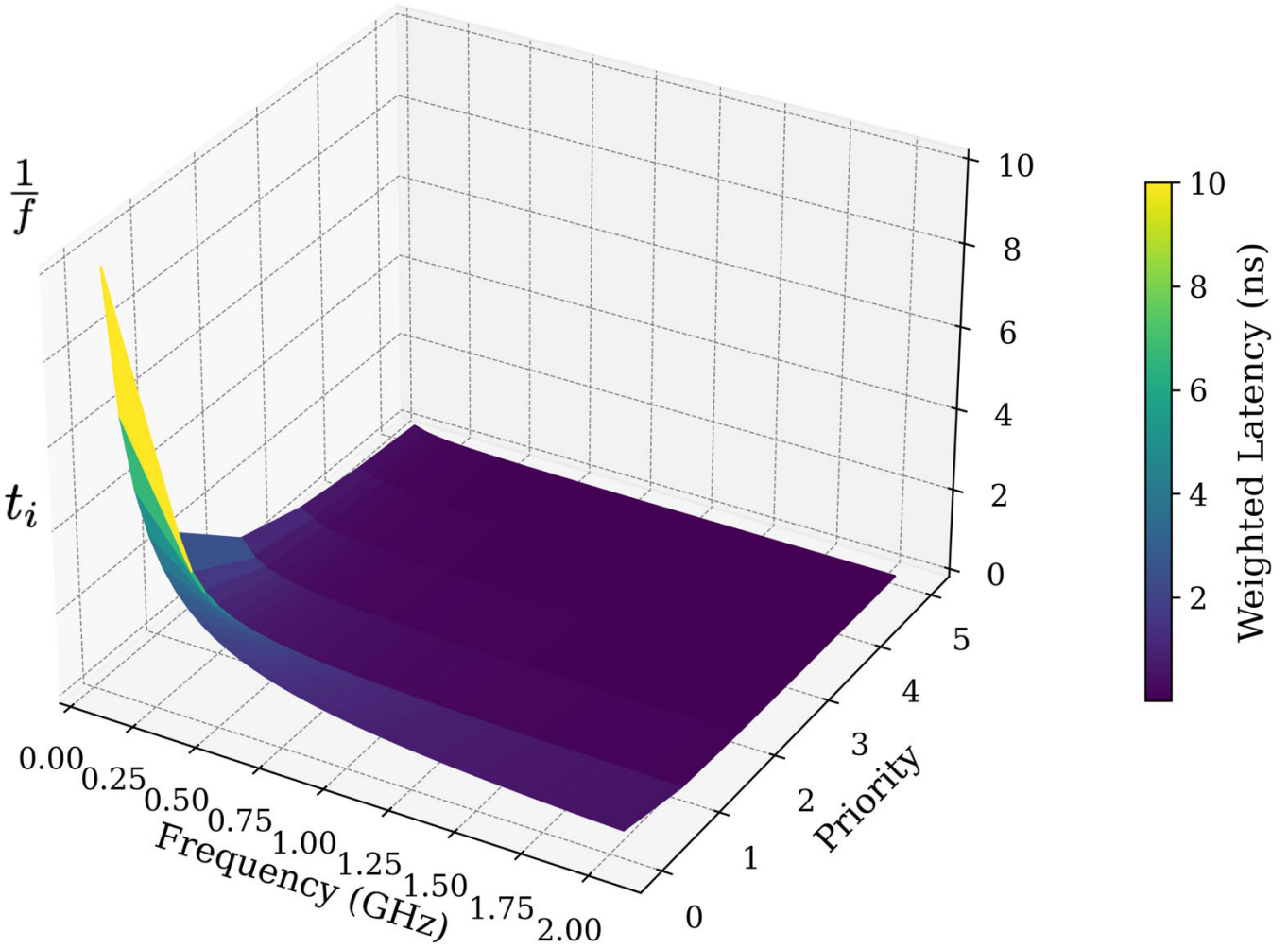
turnaround time is $t = \frac{N_{\text{cycle}}}{f} = \frac{N_{\text{IC}}}{\text{IPC} \cdot f} \propto \frac{1}{f}$

Priority-aware latency.

weighted average given by $t_{\text{aggregated}} = \sum_{i=1}^{N_{\text{task}}} w_i(p) \cdot t_i$

$$w(p) = \frac{1}{(p+1)^2}$$

larger priority level $p \in \mathbb{Z}^+$,
i.e., more **important** tasks,
leads a **higher weight** in latency.



Weight latency vs. frequency and task priority.

Simulation: power/energy metric

Task, Processor, Scheduler, **Per-run metric**.

Energy and frequency modelling.

$$P_{\text{XPU}} = P_{\text{dynamic}} + P_{\text{leakage}}.$$

$$P_{\text{dynamic}} = \alpha\, C \cdot V^2 \cdot f$$

$$P_{\text{leakage}} = V \cdot I_{\text{leakage}}$$

for a fixed period,

$$E(f) = P \cdot \tau = (\alpha C \cdot \frac{1}{k_V^2} f^3 + I_{\text{leakage}} \frac{1}{k_V} f) \cdot \tau.$$

for fixed number of instruction count,

$$= \alpha C \cdot \frac{1}{k_V^2} f^2 + I_{\text{leakage}} \frac{1}{k_V}.$$

Total, idle (leakage), active (dynamic) energy

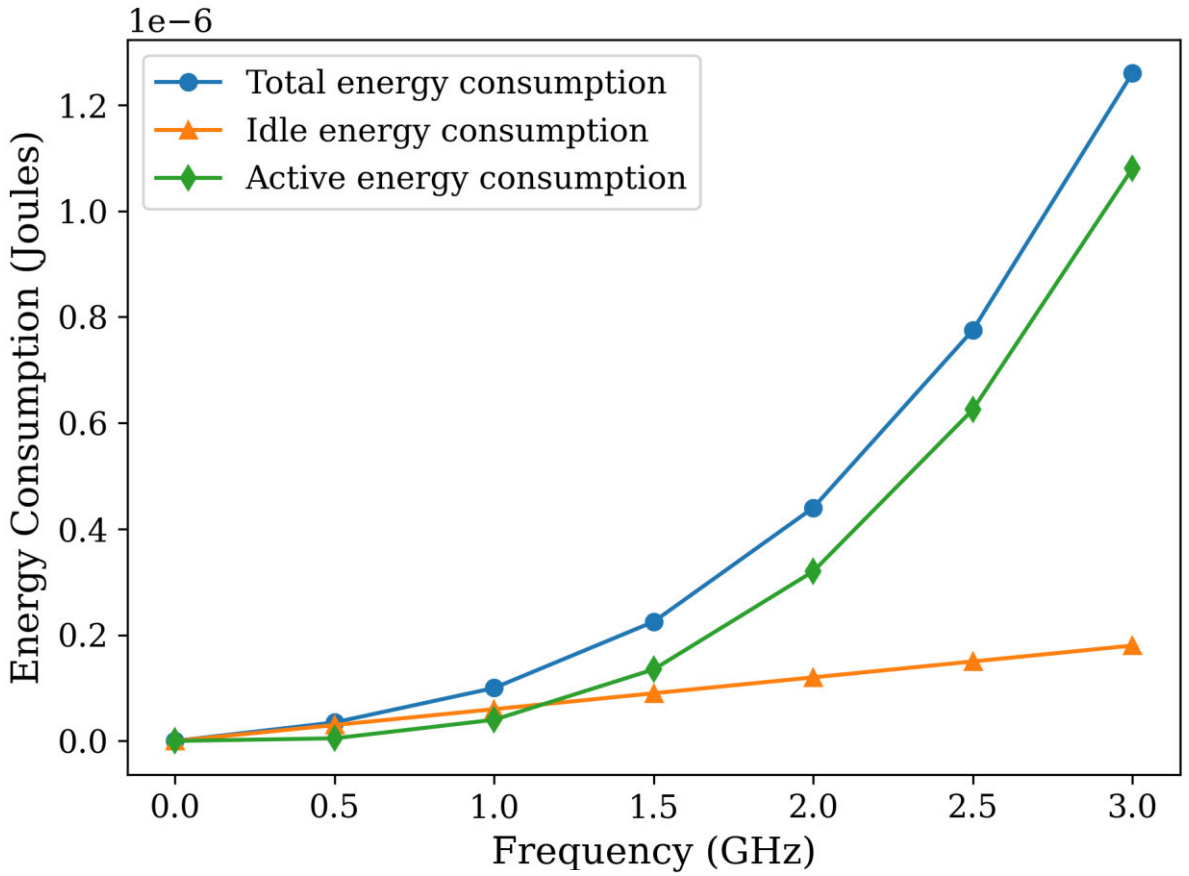


Table 3: Constants used in the processor power and frequency model.

k_V	b_f	C	I_{leakage}	unit of t	unit of f
5×10^9 Hz/V	0 Hz	1×10^{-9} F	3×10^{-1} A	ns ($\times 10^{-9}$ s)	GHz ($\times 10^9$ Hz)

Emulation: Experiment Design

- Motivation:
  - Deploy (both single-objective and multi-objective) bayesian optimization basing on **Energy** and **Time** with optuna.
- Methodology:
  - **Experiment Setting**: default env constants.
    Total Simulation Time: $T_{end} = 1000$ ms
    Total Tasks: $N_{task} = 500$
    Workload Pressure: $\lambda = 1.0$ (tasks/ms)
    Number of Trails: $N_{trail} = 100$
  - **Surrogate Model**: Gaussian Process (GP)
  - **BO Input**: search space for tunable params.
  - **Sensitive analysis** for parameters.

$$\mathcal{L}(\mathbf{x}) = \beta \cdot \ln(E(\mathbf{x})) + \gamma \cdot \ln(T(\mathbf{x}))$$

Joint Optimization Objective Func

Component	Parameter	Type	Range / Options
Little Cores	Frequency (GHz) Count	Continuous Integer	$[0.5, 1.5]$ $\{0, 1, \ldots, 4\}$
Medium Cores	Frequency (GHz) Count	Continuous Integer	$[1.0, 2.5]$ $\{0, 1, \ldots, 4\}$
Big Cores	Frequency (GHz) Count	Continuous Integer	$[1.5, 3.5]$ $\{0, 1, \ldots, 4\}$
Scheduler	Strategy Time Quantum (ms)	Categorical Continuous	{FCFS, RR, Priority} $[0.5, 5.0]$

Search Space as BO Input

Acquisition function
For single-objective: LogEI
For multi-objective: EHVI

Experiment Group 1: Kernel Selection

- Introduce Different **Kernels** for GP -> Determine which kernel fits the optimization landscape
  - **Matern 5/2** -> Twice differentiable -> Moderate smooth
  - Matern 3/2 -> Once differentiable -> Rough
  - RBF -> Infinitely differentiable -> Infinite smooth
  - Random Sampling -> Baseline

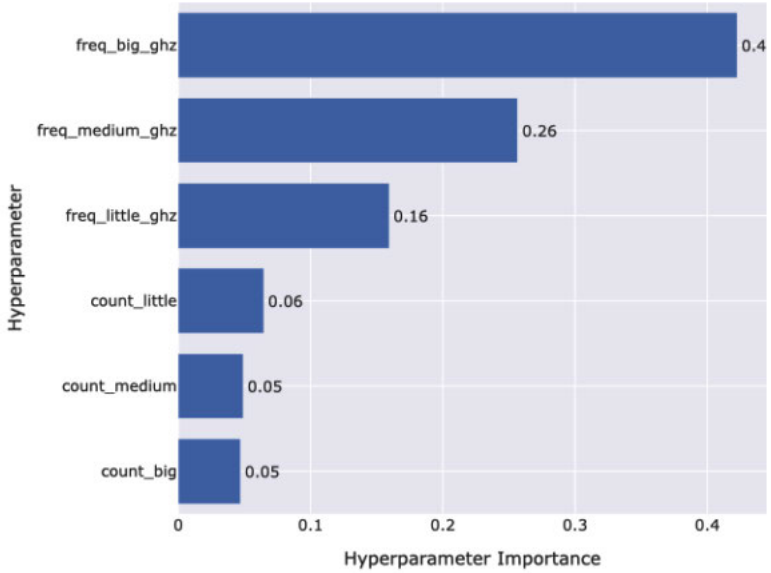
Experiment Group	Scenario / Variant	Core Config (Count × Freq)			Scheduler	Obj. Value
		Little	Medium	Big		
Panel A: Search Strategy Comparison (Balanced, $\lambda = 1.0$***)***						
Algorithm	BO (Matérn 5/2)	3×1.5	-	2×1.5	FCFS	-19.65
	BO (Matérn 3/2)	-	1×1.6	$\mathbf{4 \times 1.5}$	FCFS	-19.65
	BO (RBF)	1×1.5	$\mathbf{4 \times 1.3}$	-	RR (4.1ms)	-19.63
	Random Baseline	$\mathbf{4 \times 1.2}$	2×1.8	-	RR (2.0ms)	-19.57

Landscape is rough, as discrete params have significant impact on Time & Energy

Experiment Group 2: Focuses with different objectives

- Tune the metric weight with Energy or Time to simulate different scenarios
  - Balanced: $w_E = w_T = 1$
  - Energy-First: $w_E = 3, w_T = 1$
  - Time-First: $w_E = 1, w_T = 3$
- Results
  - Energy -> Small/Medium cores efficiency vs. Big cores efficiency-> **Race-to-Idle**

Experiment Group	Scenario / Variant	Core Config (Count × Freq)			Scheduler	Obj. Value
		Little	Medium	Big		
Panel B: Preference Adaptation (Metric Shifts)						
Metric Weights	Balanced	3 × 1.5	-	2 × 1.5	FCFS	-19.65
	Energy-First	-	2 × 2.5	**2 × 3.5**	FCFS	-18.35
	Time-First	3 × 0.7	-	**3 × 1.5**	Priority (0.7ms)	-61.66

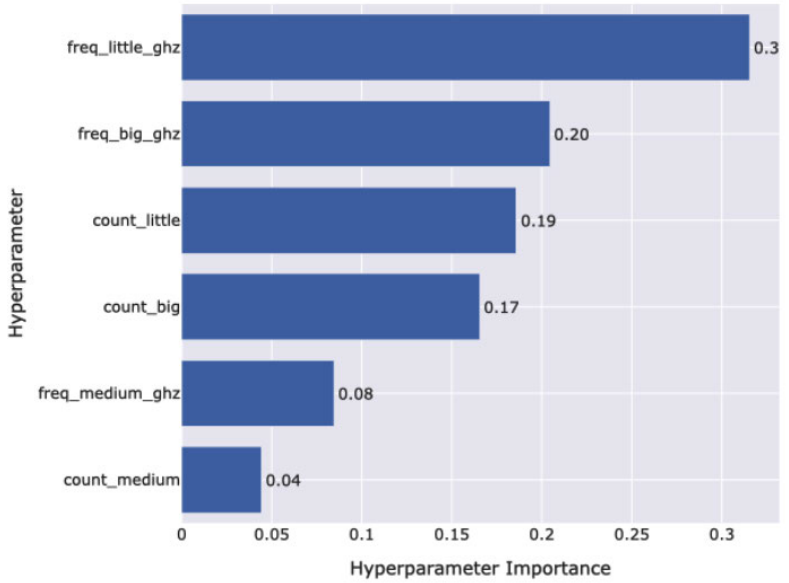


(b) Sensitivity Analysis for Energy-Focused Metric Evaluation.

Experiment Group 3: Different Workload Pressure

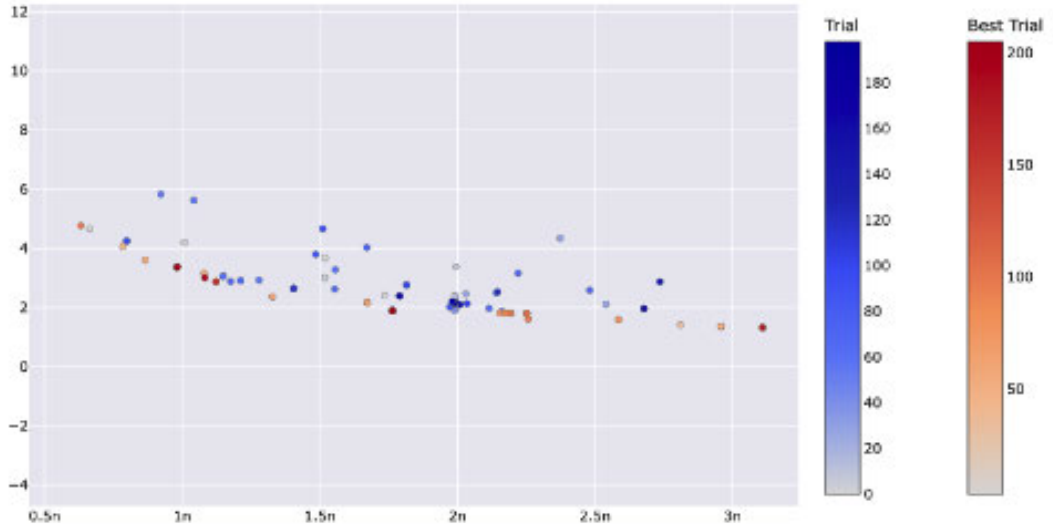
- Workload Pressure: Number of new tasks that are requested to be solved in 1 ms
  - High lambda -> High presure
- Results:
  - Low -> Only little cores -> Save energy (Easy to handle)
  - High -> All cores -> Full-scale out
  - Extreme -> Only little cores -> Save energy (Impossible to optmize Time)

Experiment Group	Scenario / Variant	Core Config (Count × Freq)			Scheduler	Obj. Value
		Little	Medium	Big		
Panel C: Workload Robustness (λ Variation)						
Arrival Rate (λ)	Low ($\lambda = 0.5$)	3×1.4	-	-	Priority (3.8ms)	-19.59
	High ($\lambda = 2.5$)	$\mathbf{4 \times 1.5}$	$\mathbf{4 \times 1.5}$	$\mathbf{4 \times 1.5}$	FCFS	-19.74
	Extreme ($\lambda = 5.0$)	1×0.5	-	-	RR (0.7ms)	-20.63

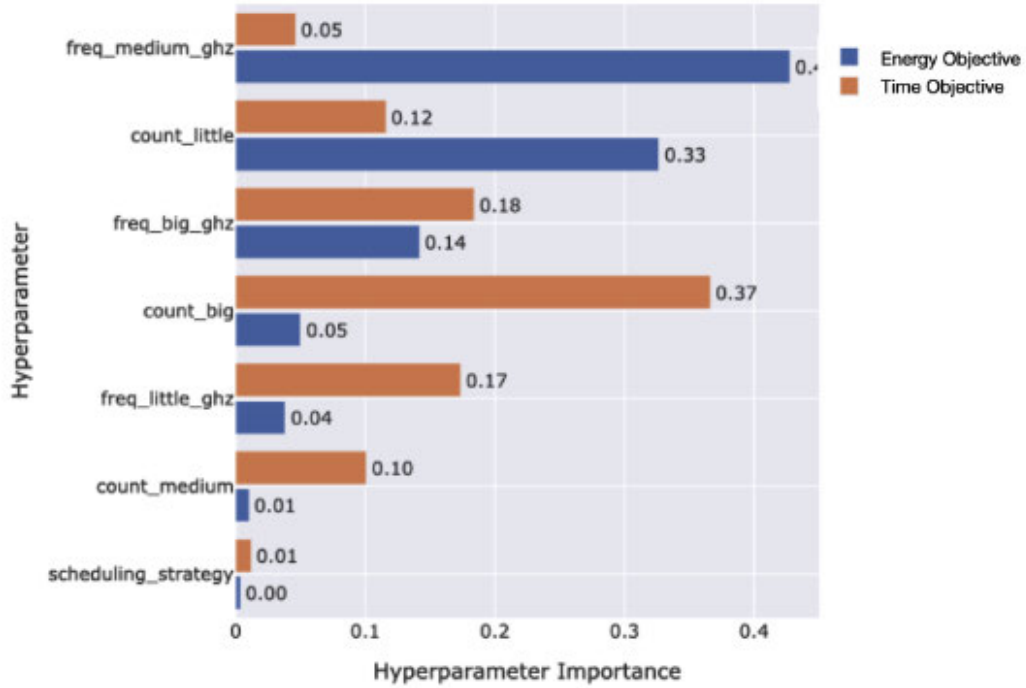


(c) Sensitivity Analysis for $\lambda = 5.0$.

Experiment Group 4: Multi-Objective Optimization



(a) Pareto Frontier Visualization.



(b) Divergent Hyperparameter Importance.

Pareto Front as a curve:
Two objectives are **conflicting** and there is no optimal solution for both of Energy and Time.

Sensitive Analysis:
For Energy - Medium Core Freq, #Little Core
For Time - #Big Core, Big Core Freq

Conclusion

- Question: Can machine learning autonomously explore the optimal strategy to balance energy & performance in heterogeneous systems?
- Design simulator for **CPU scheduling problems**.
- Use Bayesian Optimization to explore the physical properties and factors.
    1. Optimization Landscape is **non-smooth**, as search space is mixed
    2. Discovery on advanced strategies basing on **physical principles**: e.g. Race-to-Idle
    3. **No dominant strategies** for both Time (mainly contribute by *Big* core param) and Energy (mainly contribute by *Medium* and *Little* core param).
- Furture Work: Introduce dependency-based scheduling simulator, investigate properties under different simulator constants, more optimization objectives (e.g., fairness, temperature, etc.)...
- *Finally: Thank you for this rewarding course and for your valuable suggestions on our research!*