

Resource-conscious Scheduling for Energy Efficiency on Multicore Processors

Presented by:

Yu Feng

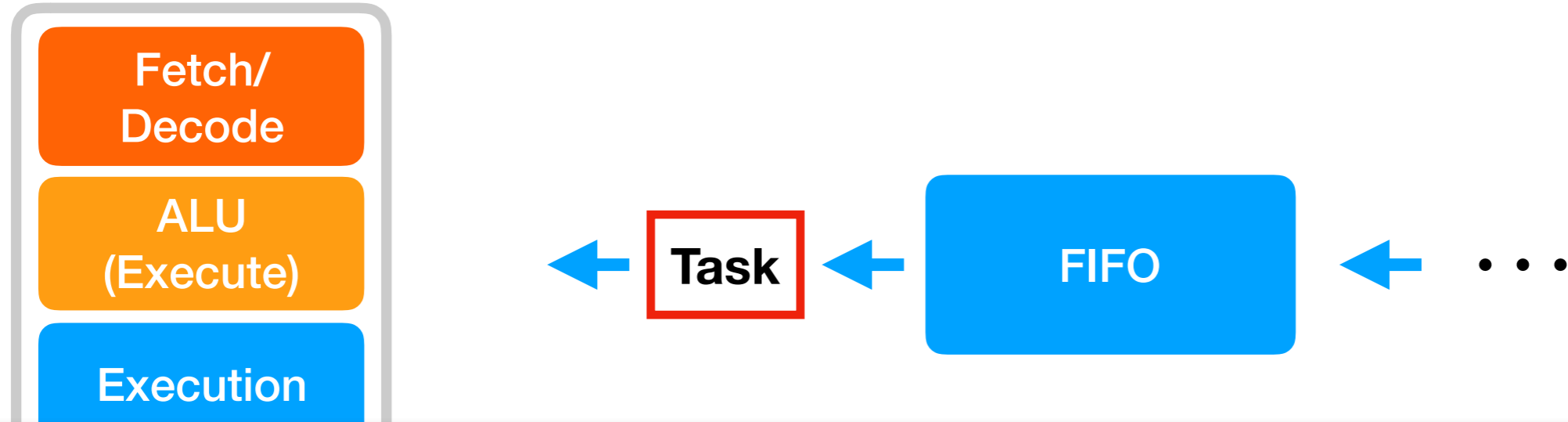
Authors:

Andreas Merkel

Jan Stoess

Frank Bellosa

Example



Which one is more energy-efficient?

For a simple program, assuming the program is perfectly parallelize-able, if we want to achieve 2x speedup:

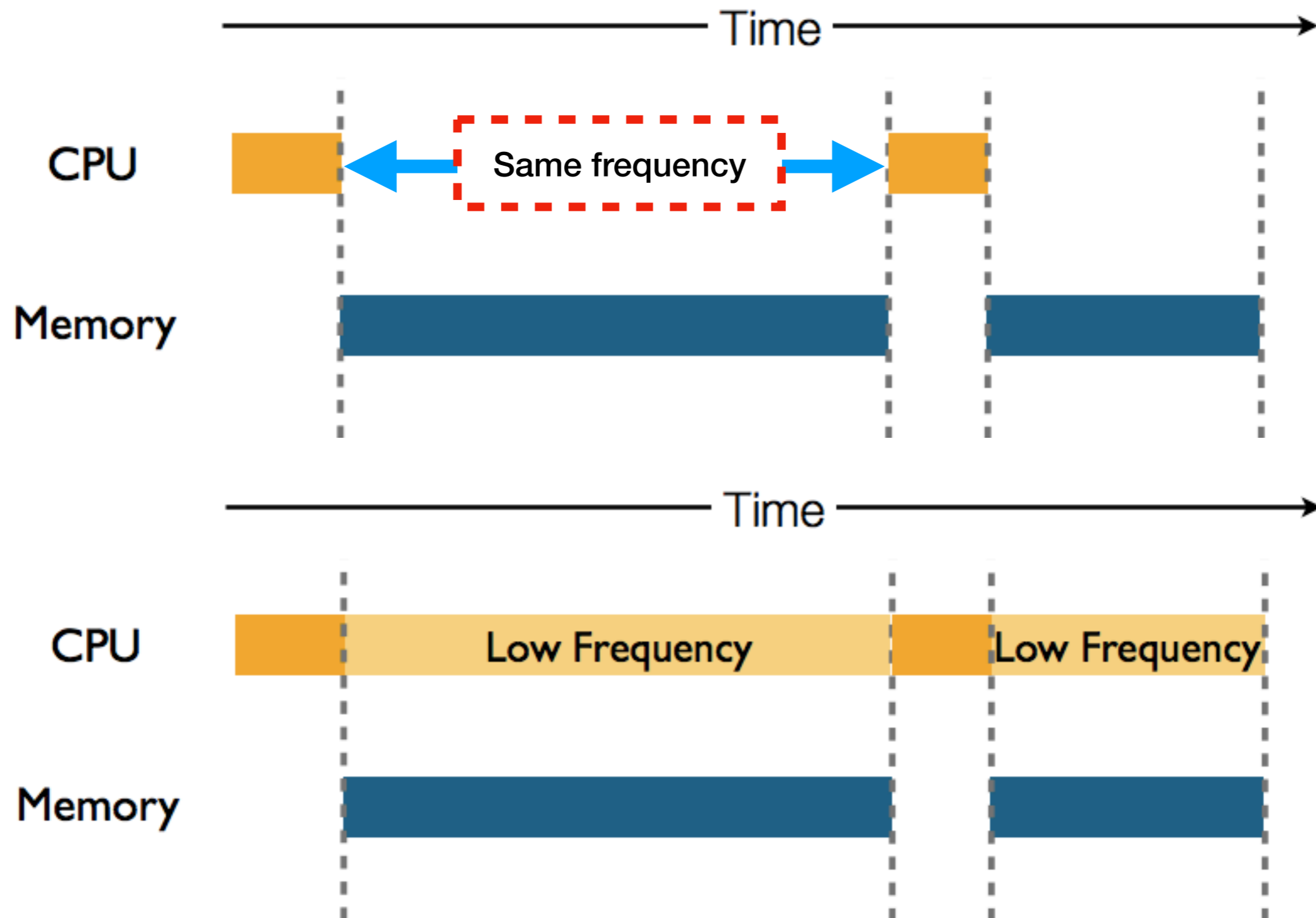
- Switch to a better processor ← **Vertical Scaling**
- Parallel the program by two cores ← **Horizontal Scaling**

Dynamic Power

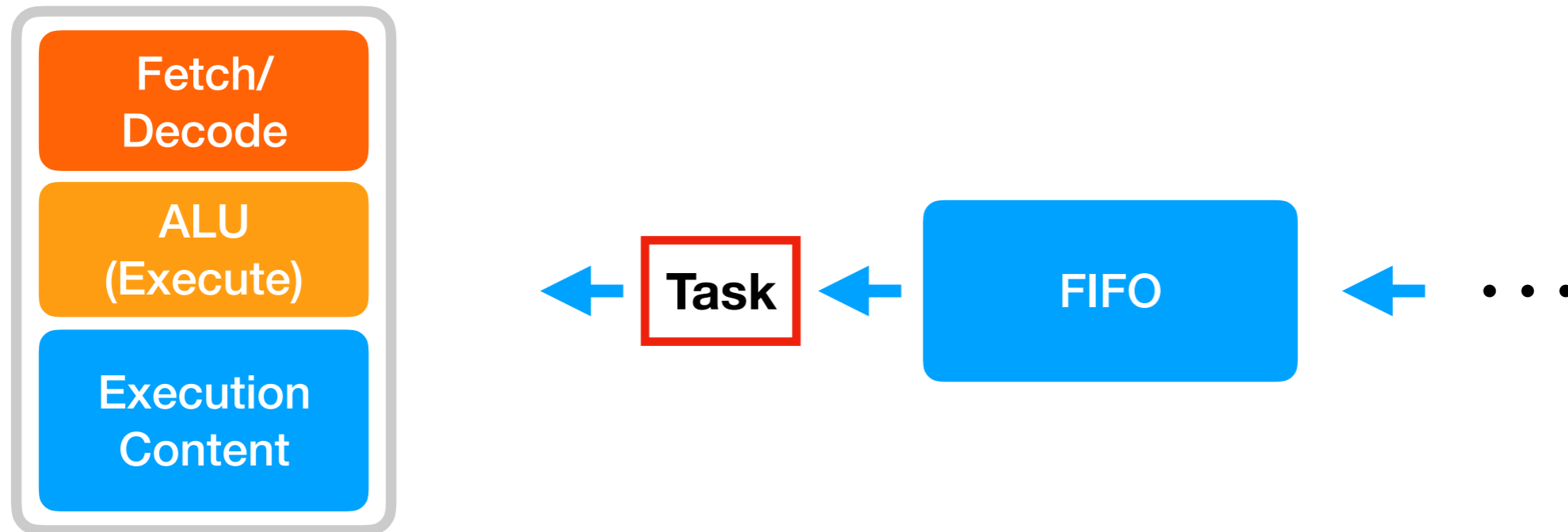
$$P = k C V^2 f$$

- Frequency f is proportional to Voltage V
- Higher voltage moves electrons faster, so a processor can run faster.
- Impact on dynamic power: 0.5 of frequency = 12.5% dynamic power.

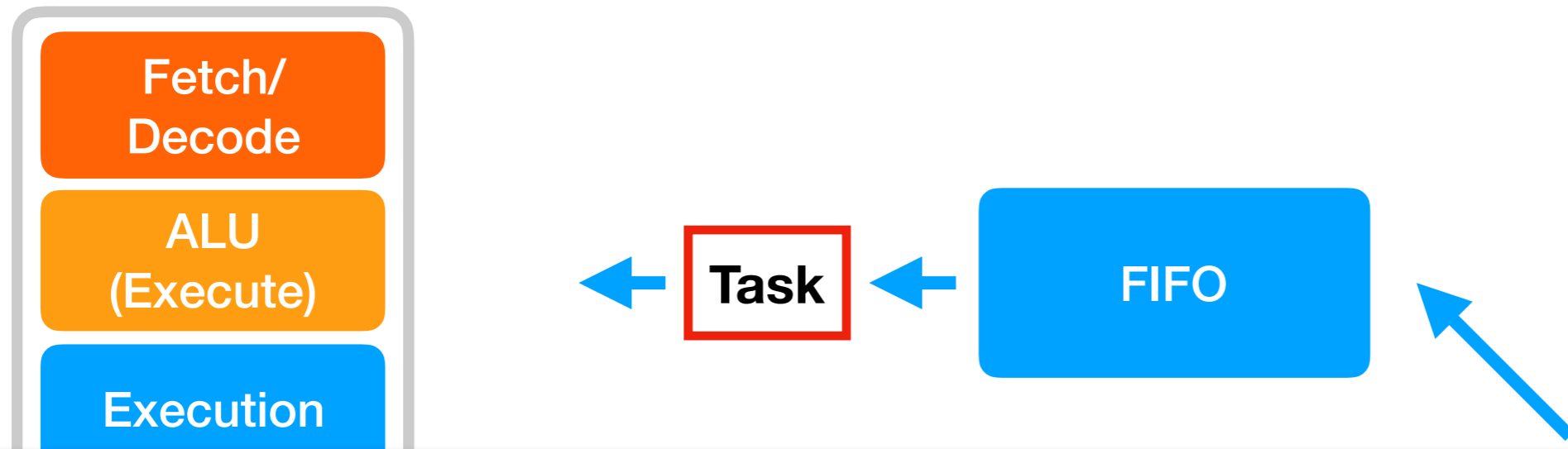
Dynamic Voltage Frequency Scaling



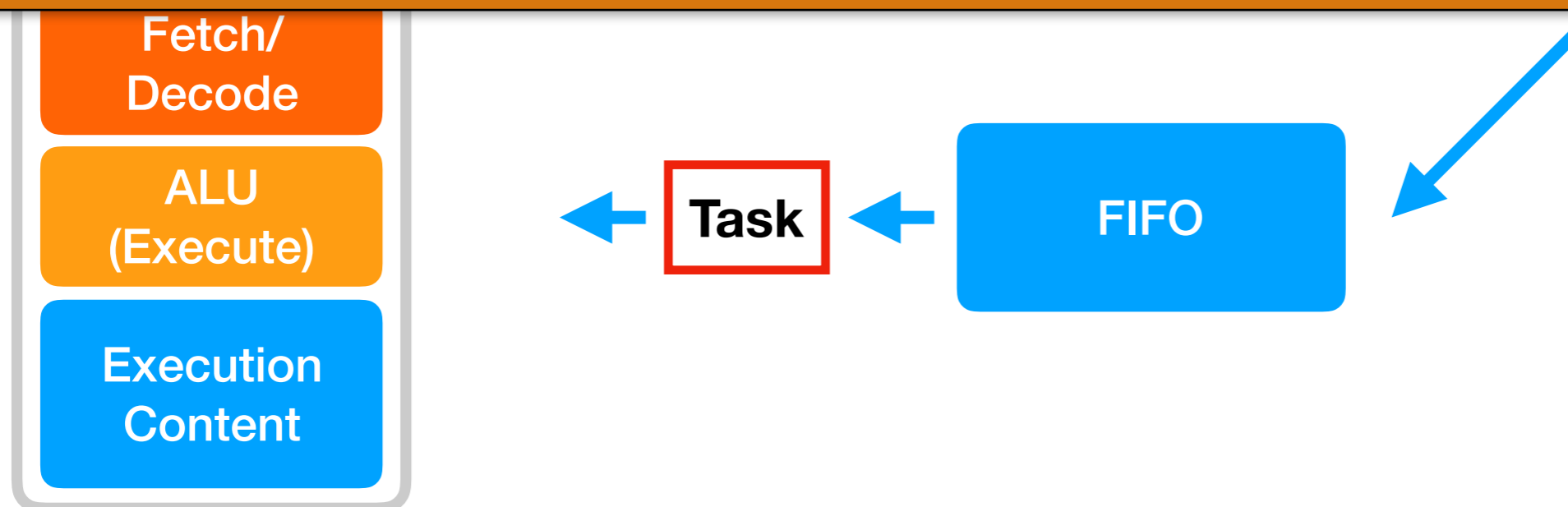
Modern Multi-Core Processor



Modern Multi-Core Processor



What if the task workloads are not identical?



Concerns

- What if the tasks have different characteristics?

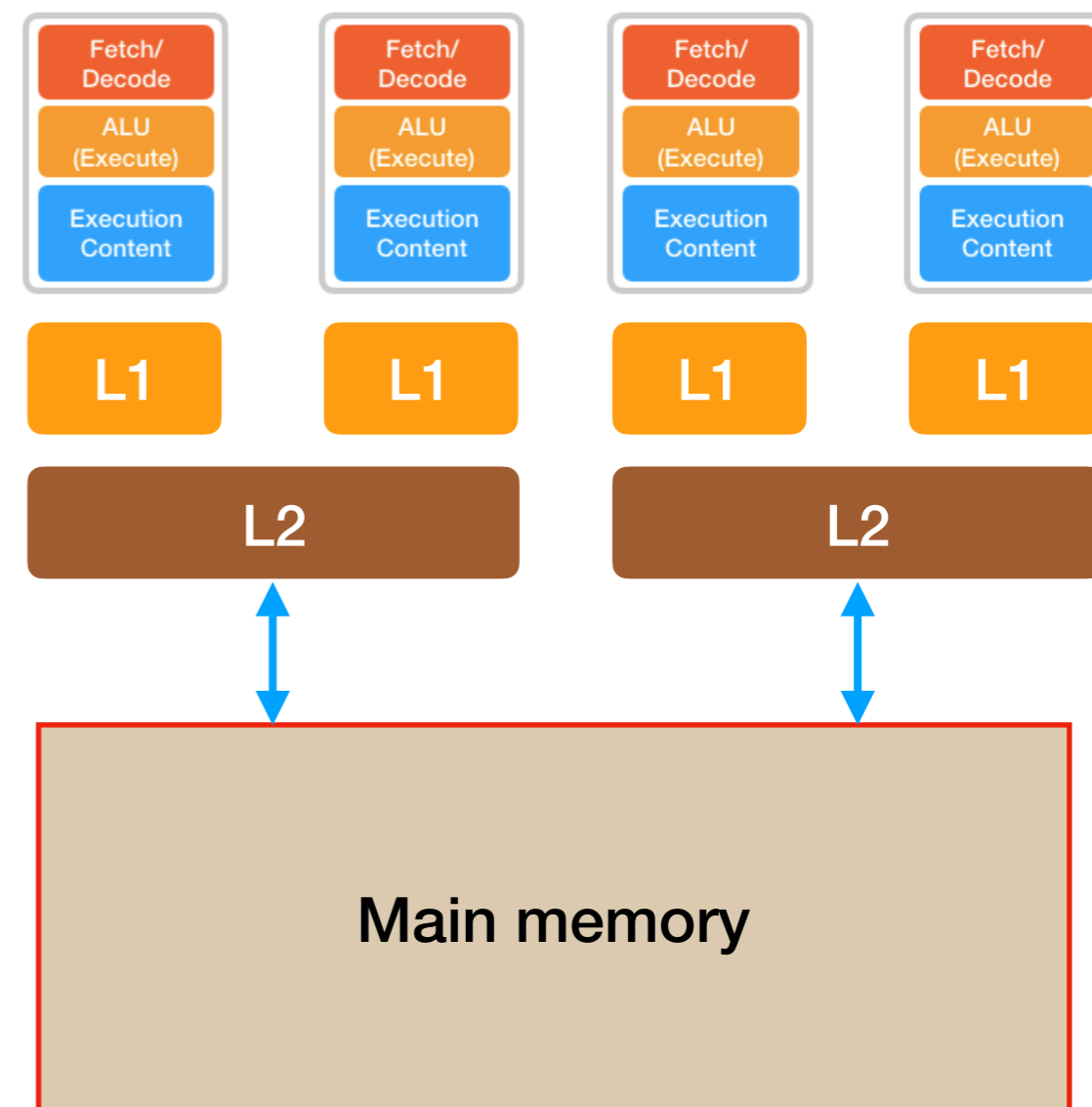
- If some tasks are “compute-bound”?

- If some tasks are “memory-bound”?

Definition

- compute-bound: run more efficiently at a higher frequency

- Memory-bound: executed at lower CPU frequencies without significant slowdown



Problem

Given tasks with varying characteristics, how can a multi-core processor schedule in a more energy-efficient way?

EDP: Energy Delay Product

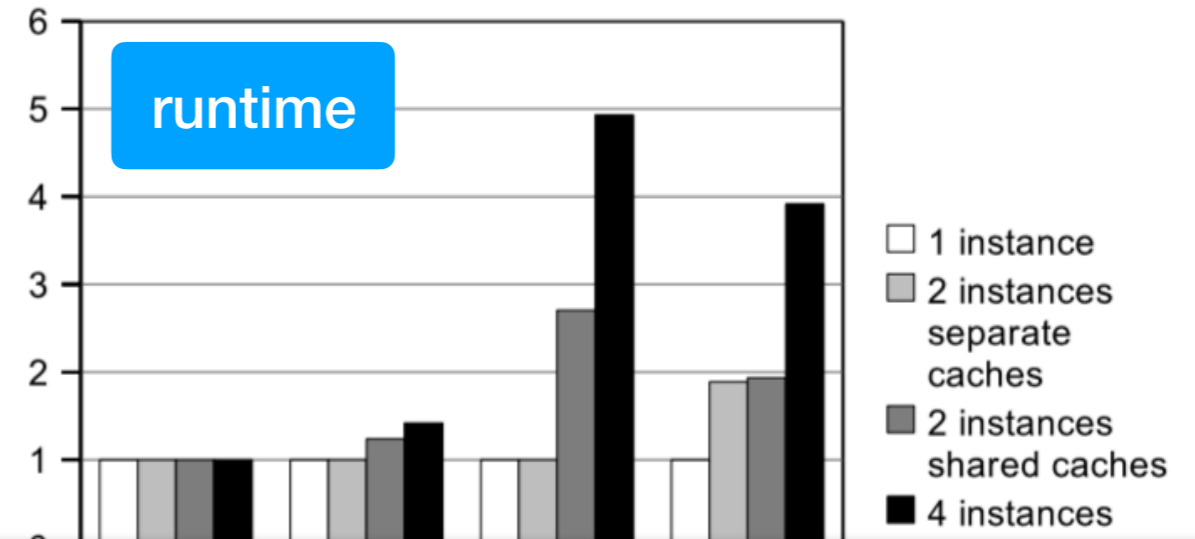
Some definitions:

- **Energy:** Current power x execution time;
- **Delay:** The execution time of a program;
- **EDP:** energy delay product;

What does low EDP mean?

Analysis

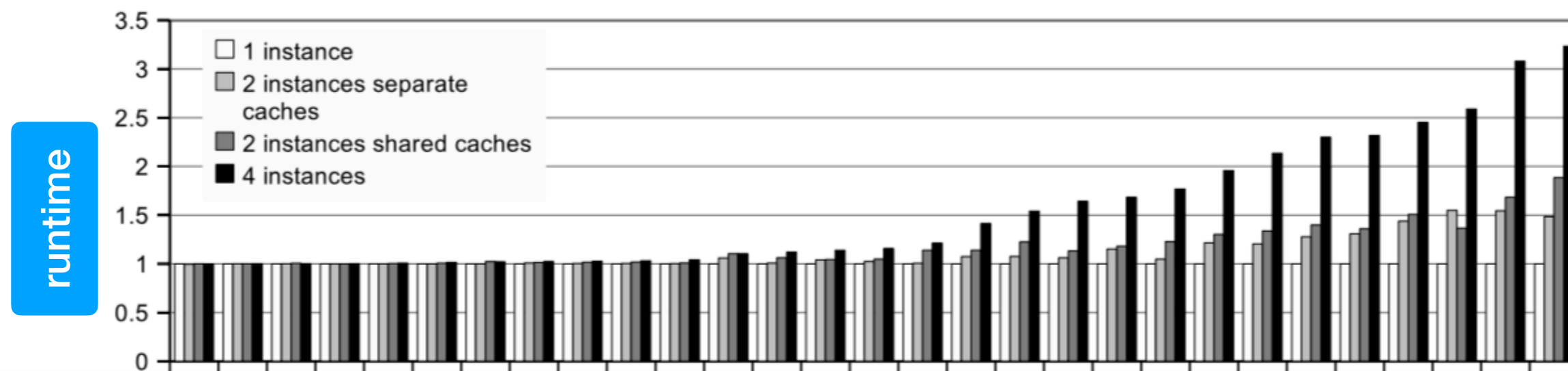
- Experimental Setup: 4-Core CPU
- Tasks: memory-bound stream
compute-bound aluadd



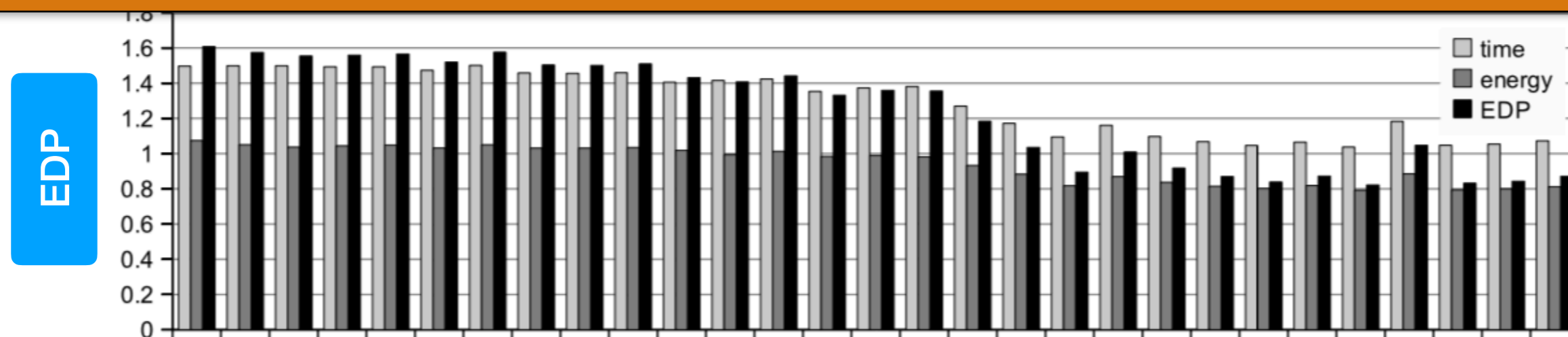
Only memory-bound program can be benefited from DVFS

instances	stream			aluadd			avg. EDP
	time	ener.	EDP	time	ener.	EDP	
4 aluadd	—	—	—	1.49	1.16	1.68	1.68
1 str. + 3 alu.	1.13	0.83	0.93	1.49	1.08	1.63	1.45
2 str. + 2 alu.	1.07	0.77	0.82	1.49	1.10	1.60	1.23
3 str. + 1 alu.	1.09	0.85	0.93	1.49	1.13	1.73	1.13
4 stream	1.04	0.80	0.83	—	—	—	0.83

Benchmark Result



How to schedule the tasks?



Resource-conscious Scheduling

Activity vector: representing demands of a task

- memory bus, L2 cache, and “the rest of the core”

Resource-conscious Scheduling:

- The goal of our balancing policy is to have tasks with different characteristics available on each core, so that a co-scheduling policies has a higher chance of finding a suitable task on each core;
- Steps:
 - 1. Vector balancing;
 - 2. Cross-node migration;
 - 3. Sorted co-scheduling;

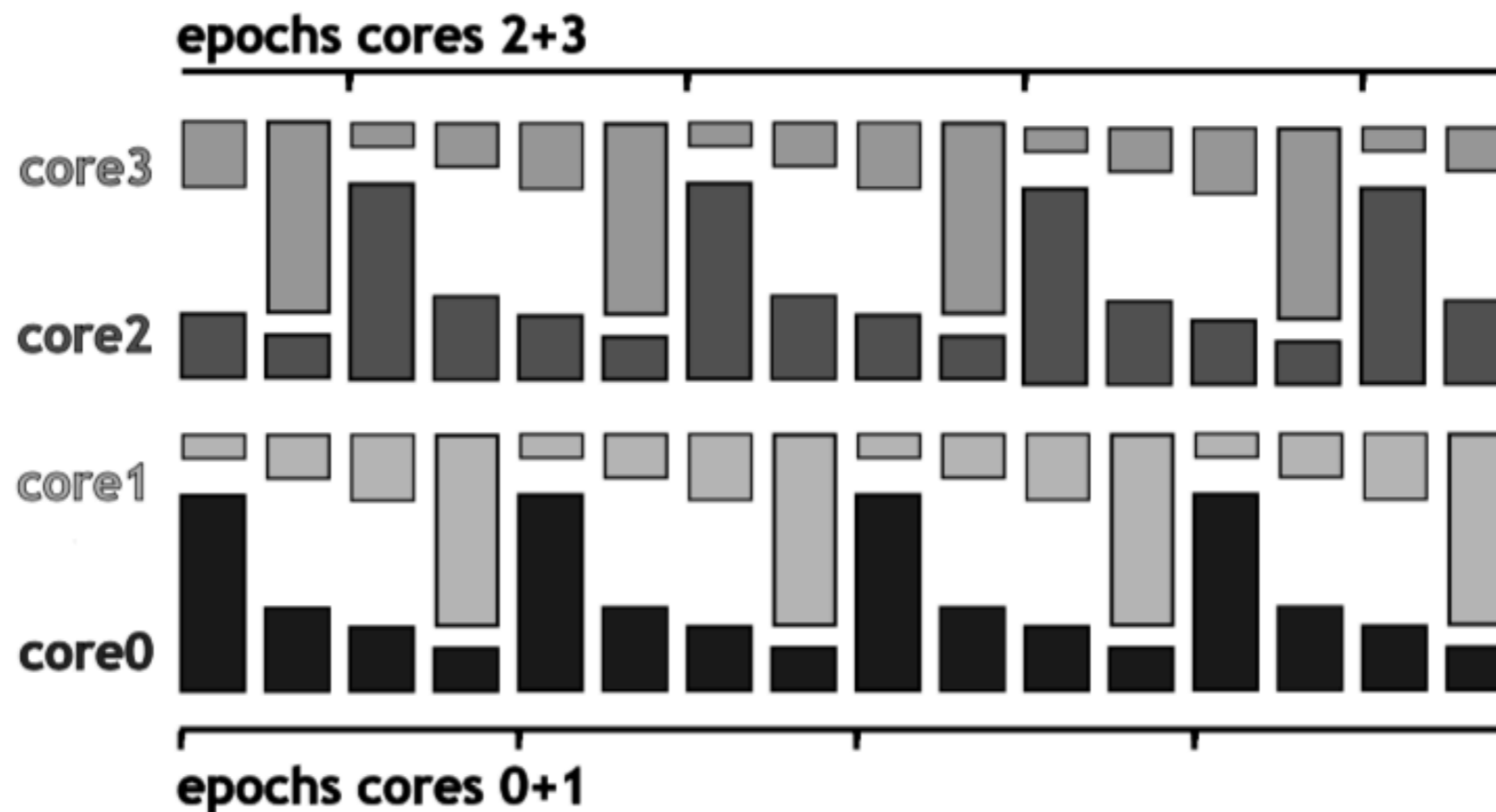
Frequency Heuristic:

- As a fallback when too many memory-bound tasks and avoid contention.

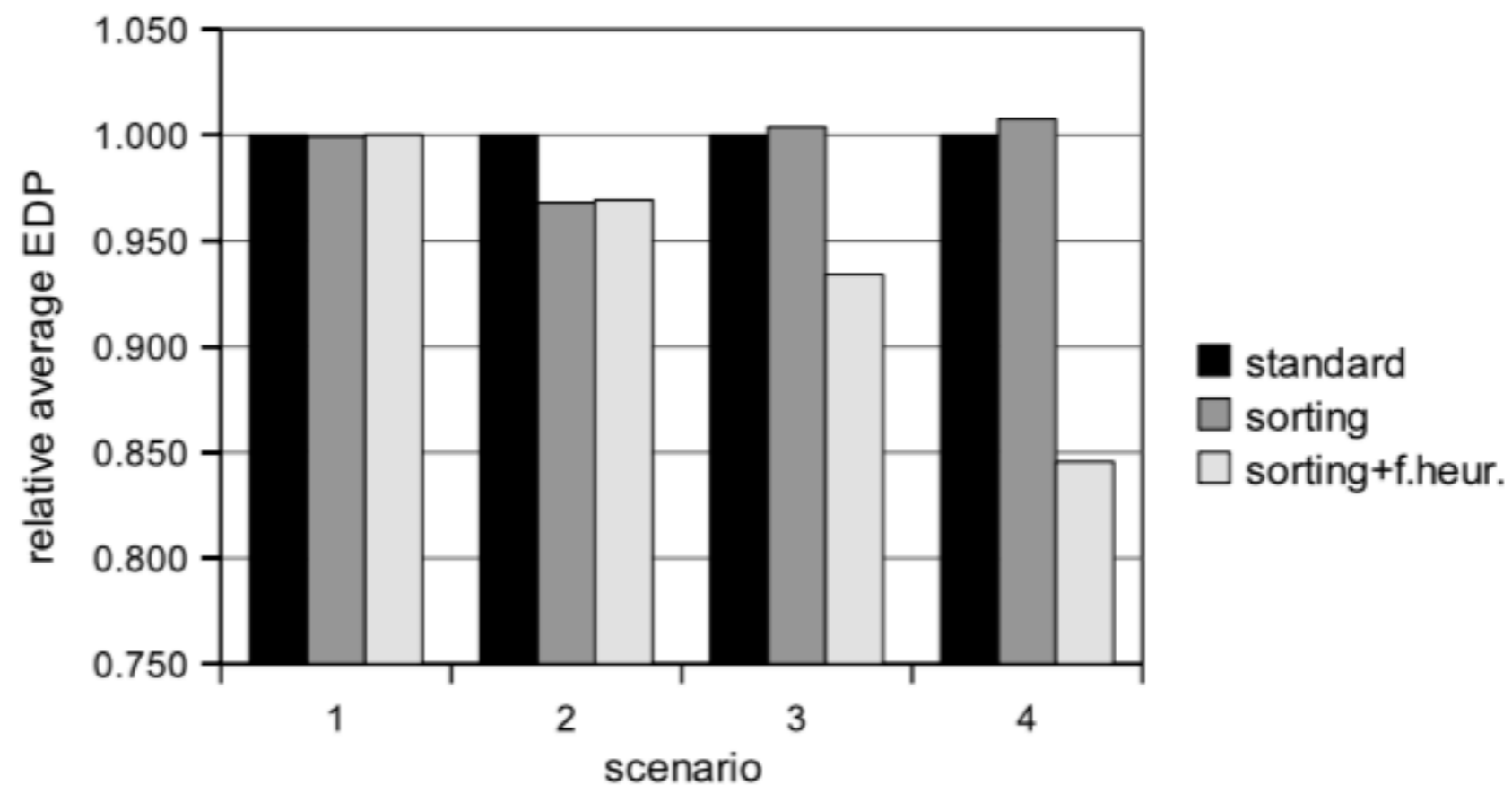
$$\text{EDP factor} = 1.6 - 0.8x$$

Sorted co-scheduling

The idea behind sorted co-scheduling is to group the cores into pairs of two and to execute tasks with complementary resource demands on each of them;



Evaluation



Sorted co-scheduling + frequency heuristics works better.

Takeaways

- ▶ **Co-scheduling can reduce resource contention.**
- ▶ **Scaling down frequency is only beneficial to memory-bound program**
- ▶ **DVFS improve energy efficiency.**