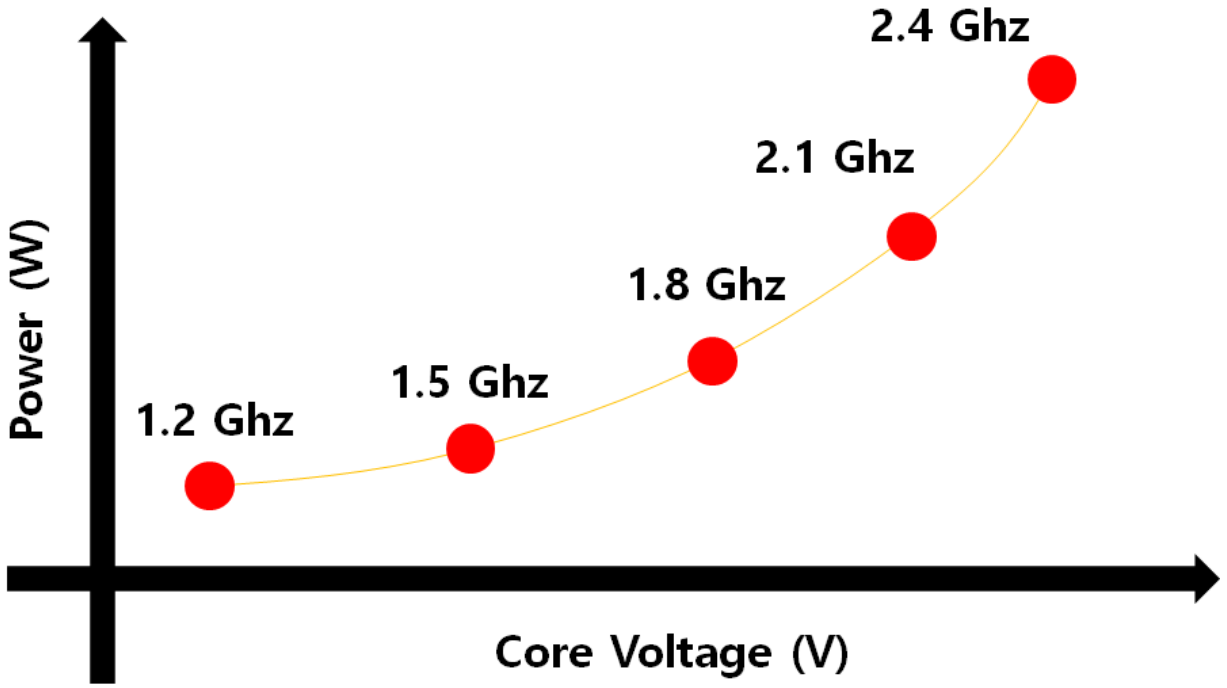
EarDVFS: Environment-Adaptable RL-based DVFS for Mobile Devices

Jaeheon Kwak[†], Sangeun Oh[‡], Jinkyu Lee[§], Insik Shin[*]

Department of Software, Ajou University
Department of Computer Science and Engineering, Korea University
Department of Computer Science and Engineering, Yonsei University
School of Computing, KAIST

Background

- DVFS (Dynamic Voltage Frequency Scaling)
  - Adjusting a processor's voltage & frequency
  - Goal: balancing & improving the processor's performance and power efficiency



Background

- DVFS (Dynamic Voltage Frequency Scaling)
  - Adjusting a processor's voltage & frequency
  - Goal: balancing & improving the processor's performance and power efficiency
  - Governor: an algorithm that determine the voltage & frequency pair

conservative

interactive

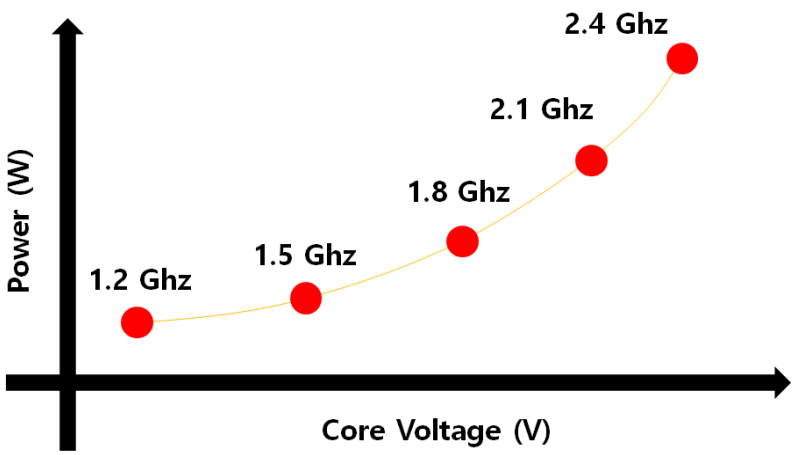
powersave

schedutil

ondemand

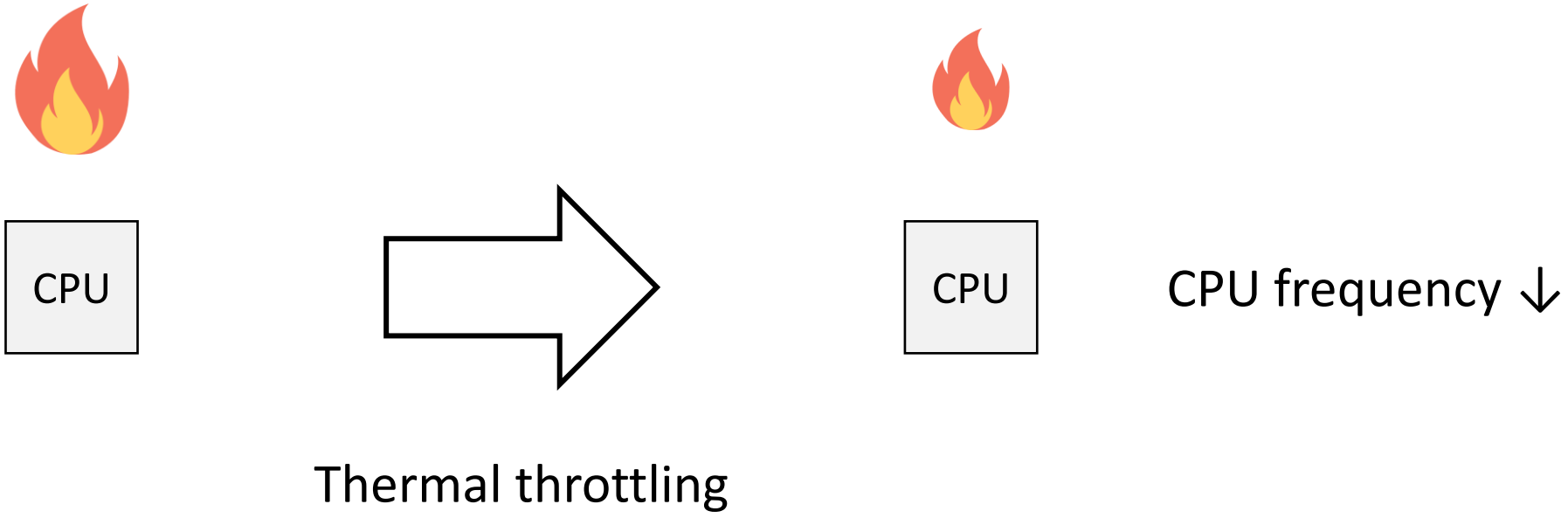
performance

Governors



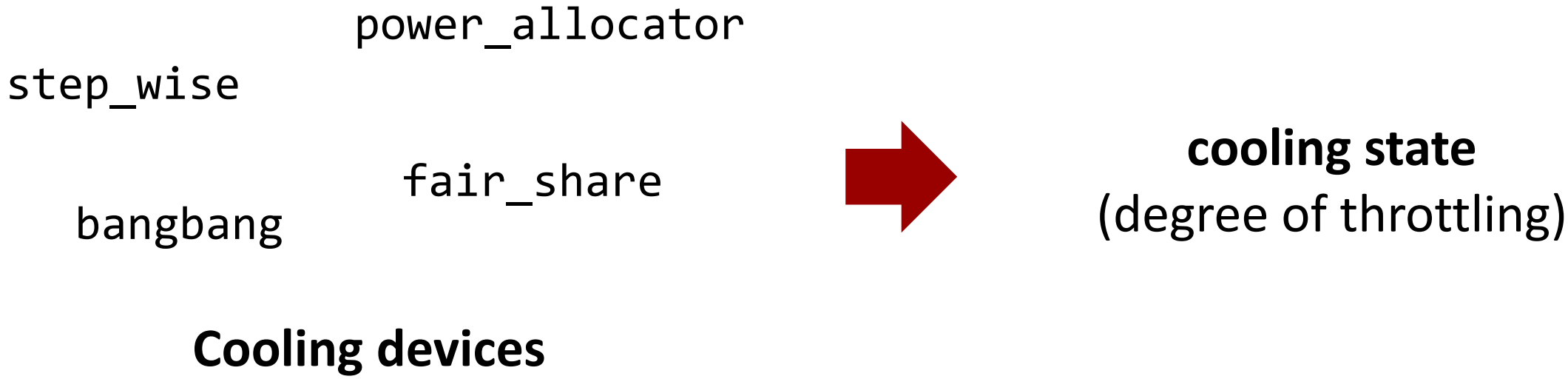
Background

- Thermal throttling
  - Reducing processor's frequency to lower the processor's temperature



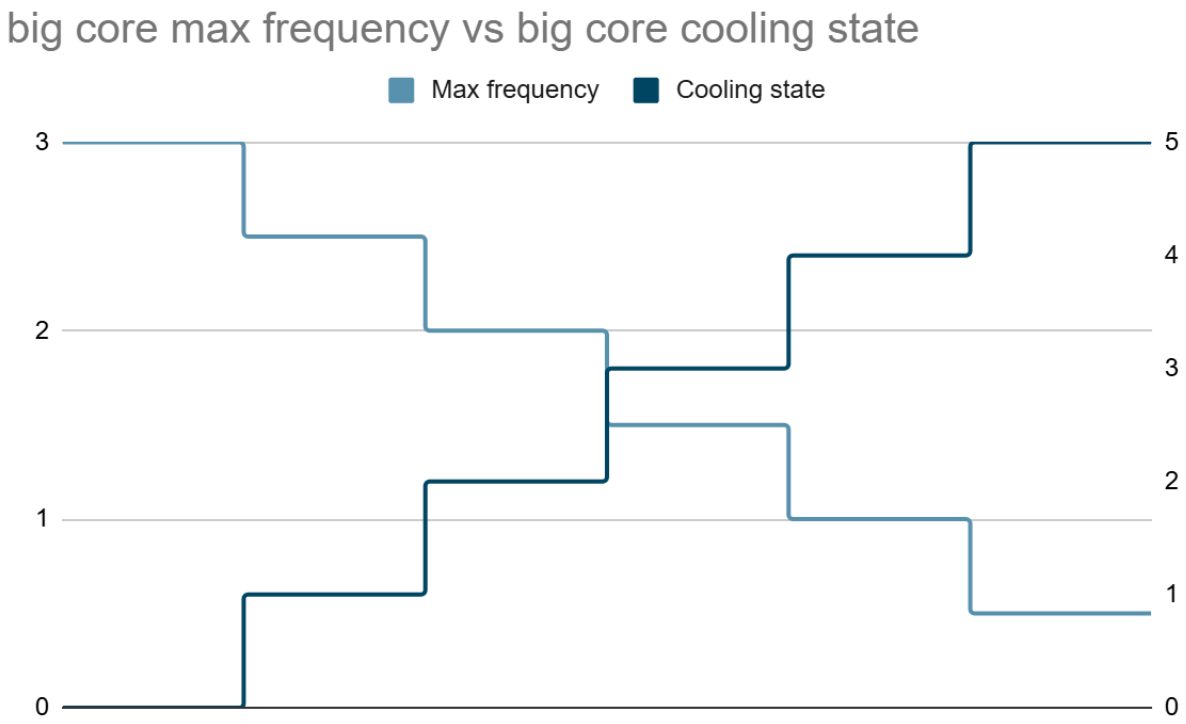
Background

- Thermal throttling
  - Reducing processor frequency to lower the processor temperature
  - Cooling device: a subsystem that determines the degree of throttling



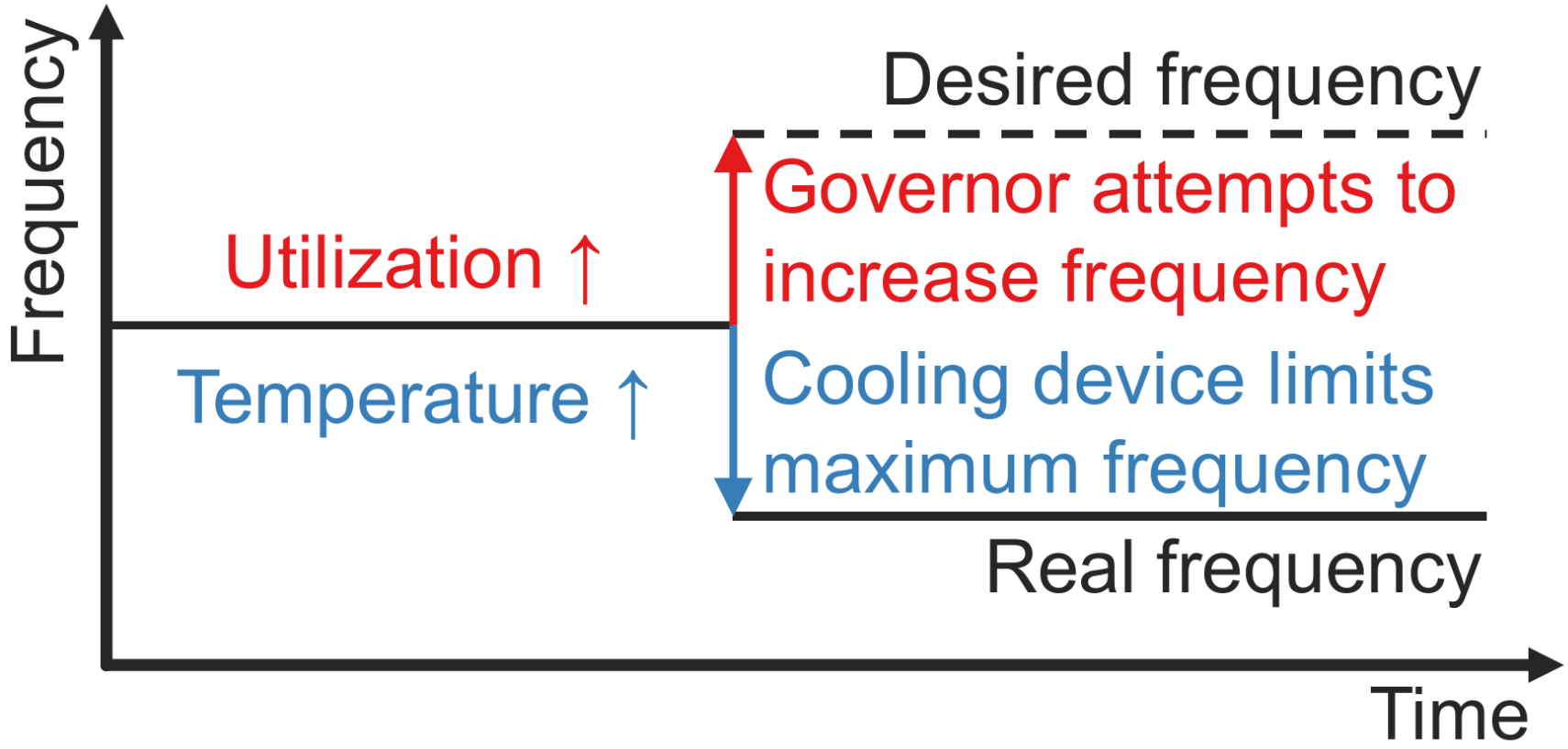
Background

- Thermal throttling
  - Reducing processor frequency to lower the processor temperature
  - Cooling device: a subsystem that determines the degree of throttling
  - Cooling state limits the maximum frequency



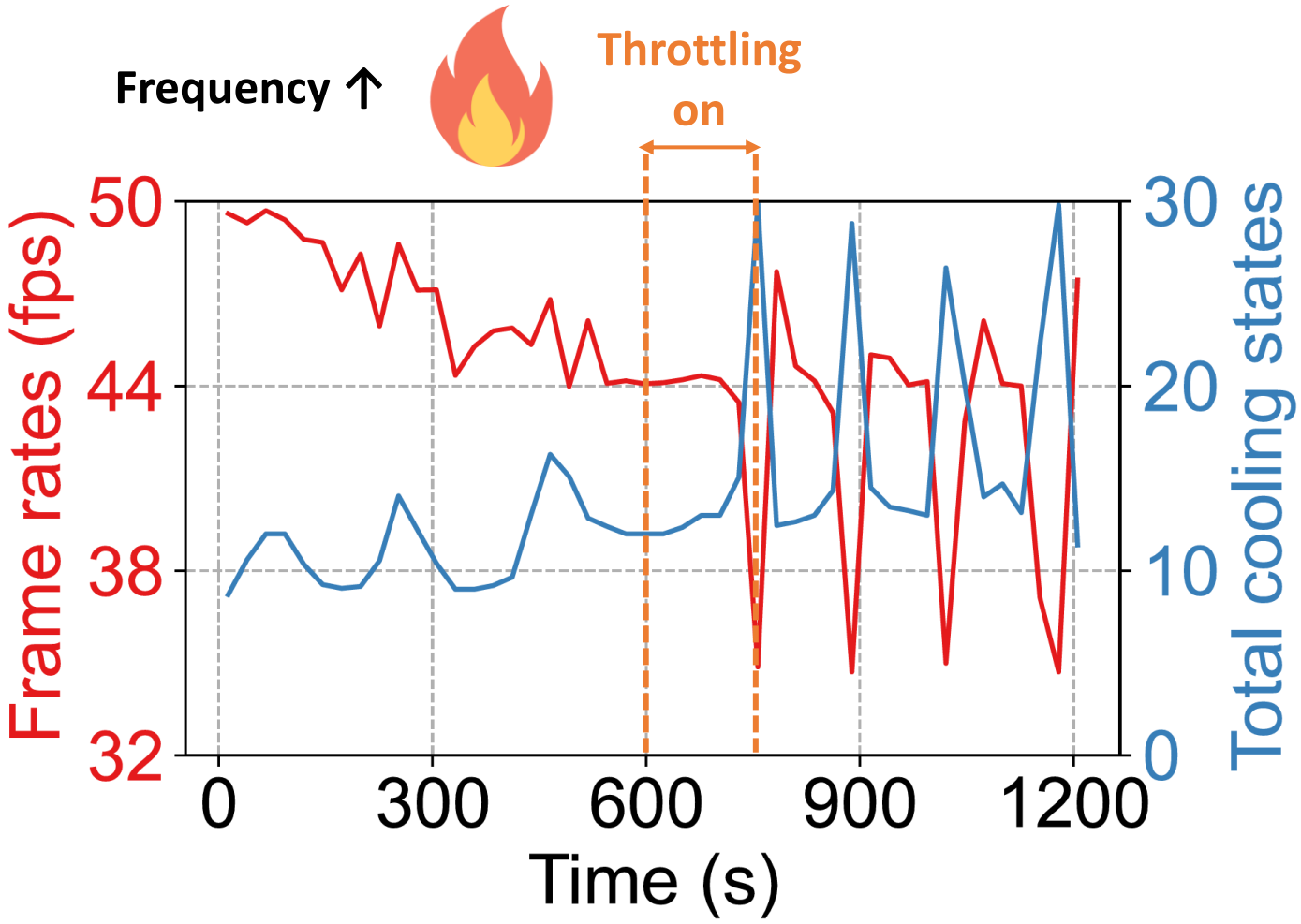
Background

- DVFS governor vs thermal throttling
  - Thermal throttling is stronger than DVFS governor



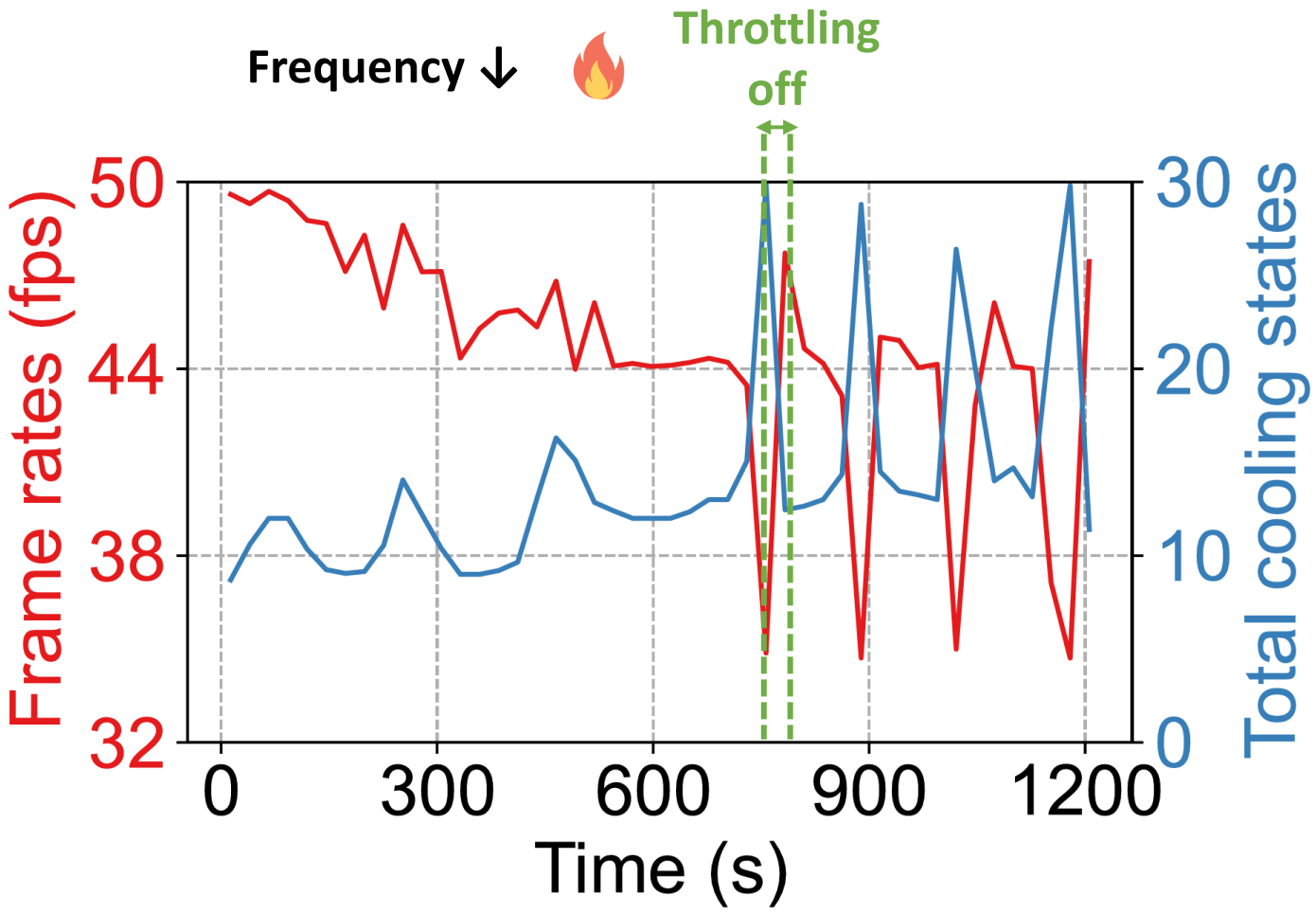
Inefficiency of current DVFS & throttling (1)

- Throttling causes frequency & performance fluctuation!



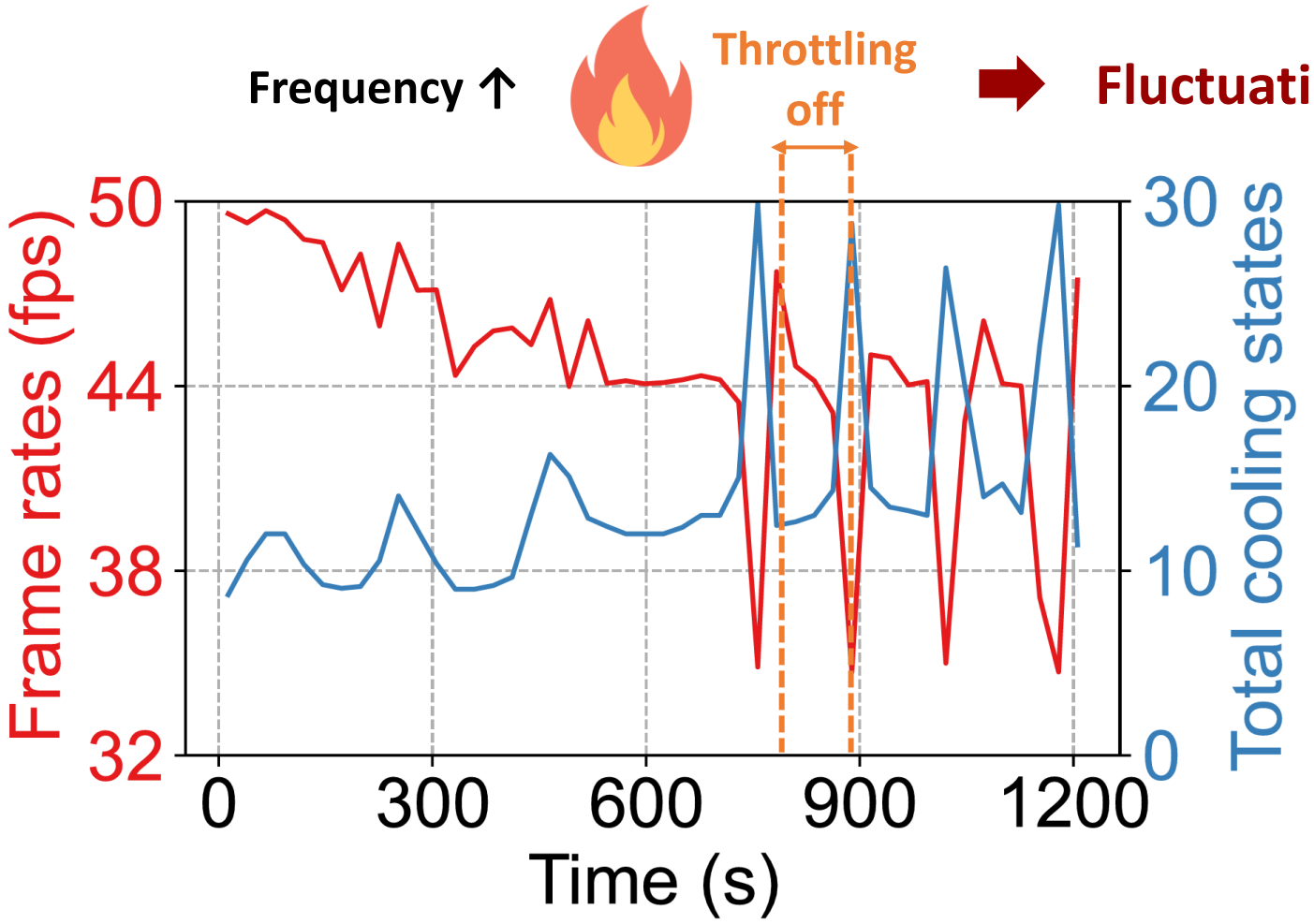
Inefficiency of current DVFS & throttling (1)

- Throttling causes frequency & performance fluctuation!



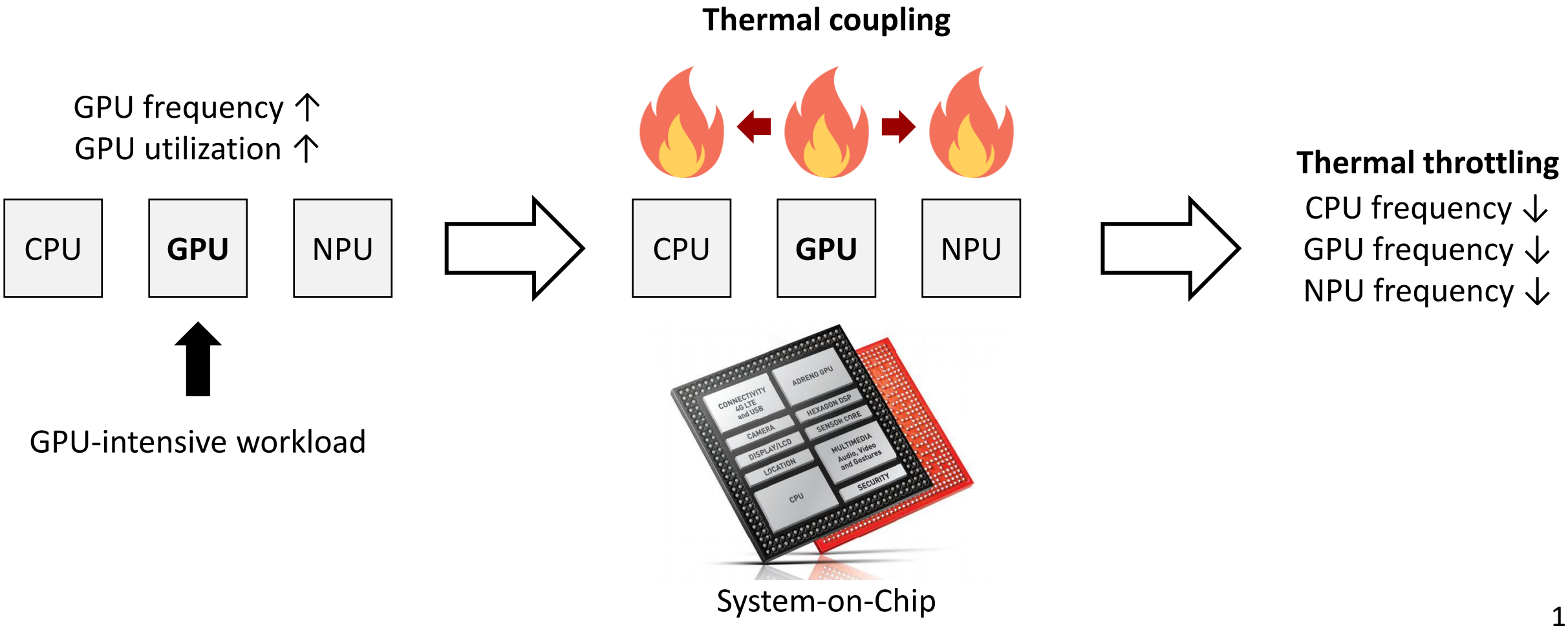
Inefficiency of current DVFS & throttling (1)

- Throttling causes frequency & performance fluctuation!



Inefficiency of current DVFS & throttling (2)

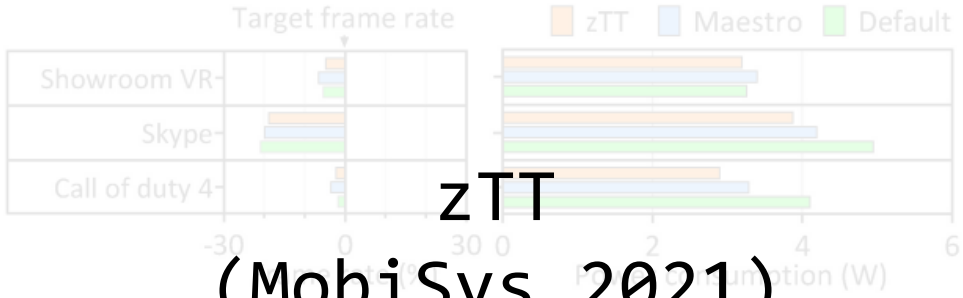
- Lacks consideration of system-on-chip architecture!



Solution: reinforcement learning (RL)-based DVFS

- zTT: Learning-based DVFS with Zero Thermal Throttling for Mobile Devices
  - MobiSys 2021

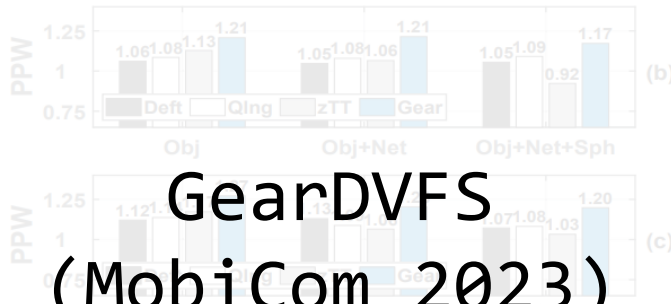
- GearDVFS: A Workload-Aware DVFS Robust to Concurrent Tasks for Mobile Devices
  - MobiCom 2023



zTT
(MobiSys 2021)

(b) Pixel 3a runs each app targeting 60 FPS, which is physically not achievable. While minimizing FPS degradation, zTT saves more power consumption.

Figure 11: Frame rate and total power consumption for mobile apps running on JETSON TX2 and Pixel 3a.

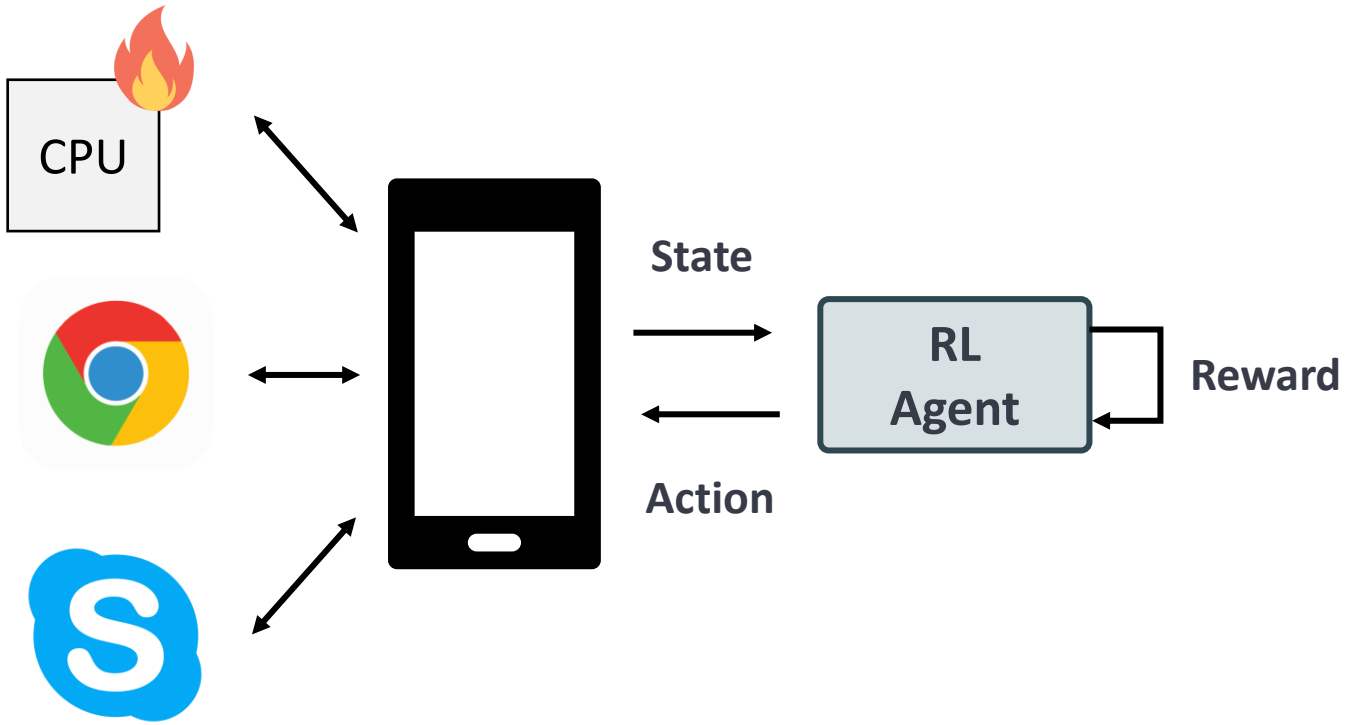


GearDVFS
(MobiCom 2023)

Figure 10: PPW of each method in the mobile applications of (a) self-driving, (b) robot and (c) UAV ground station under various concurrent tasks.

Solution: reinforcement learning (RL)-based DVFS

- RL agents learns the device's operating environment and current workloads
- Then it selects the frequencies of the CPU and GPU to maximize performance



Limitations of existing RL-based DVFS methods & solutions

- L1. Ignoring device-specific configurations
- S1. Proactive throttling

- L2. Insufficiency of considering processor temperature alone
- S2. Leveraging battery temperature

- L3. Fixed environment-dependent parameters
- S3. Environment-robust RL design

L1. Ignoring device-specific configurations

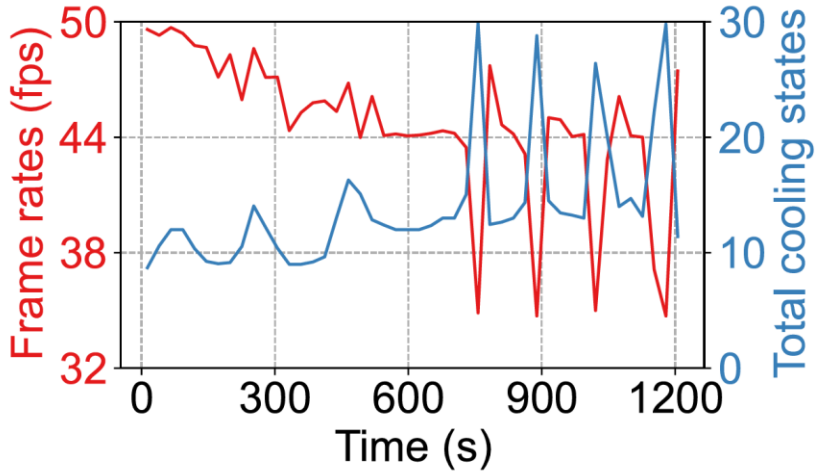
- Device-specific configurations
  - schedutil's target utilization, throttling's temperature thresholds, …
  - Device vendors have carefully engineered these values for each device
  - RL-based DVFS completely ignores these values when making frequency decisions

CFS
↓
PELT CPU utilization
↓
schedutil
↓
Target utilization (e.g., 80%)

Pixel 6 Pixel 7

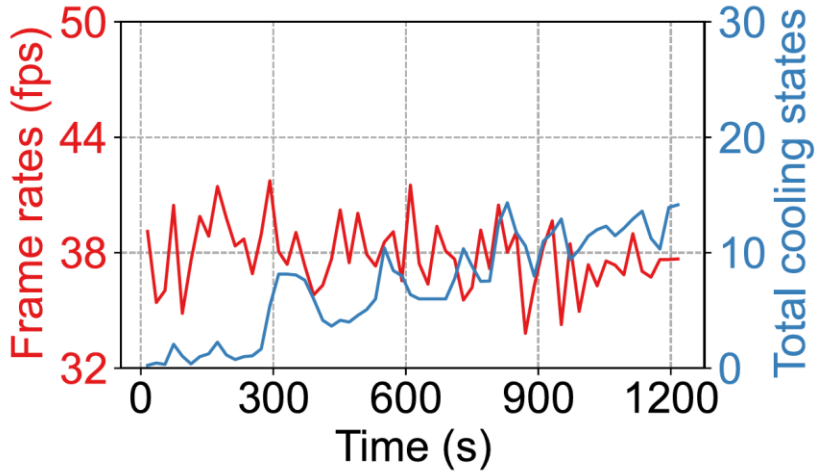
L1. Ignoring device-specific configurations

- GearDVFS
  - Aiming to maintain CPU/GPU utilization at 80%
    (Based on official Linux schedutil documentation)
  - Actual target utilization of Pixel 6 (GS101) is quite lower than 80% (e.g., 15-50%)
    (We confirmed it through reverse engineering)



Better!

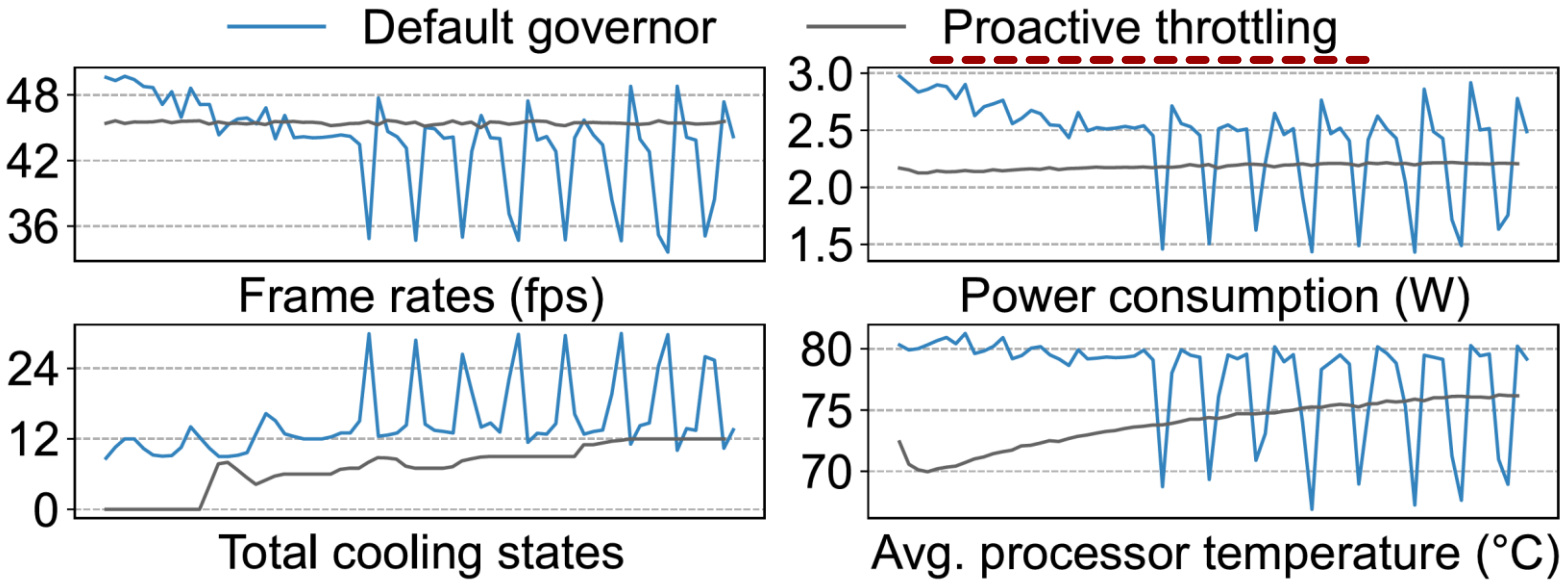
schedutil



GearDVFS

S1. Proactive throttling

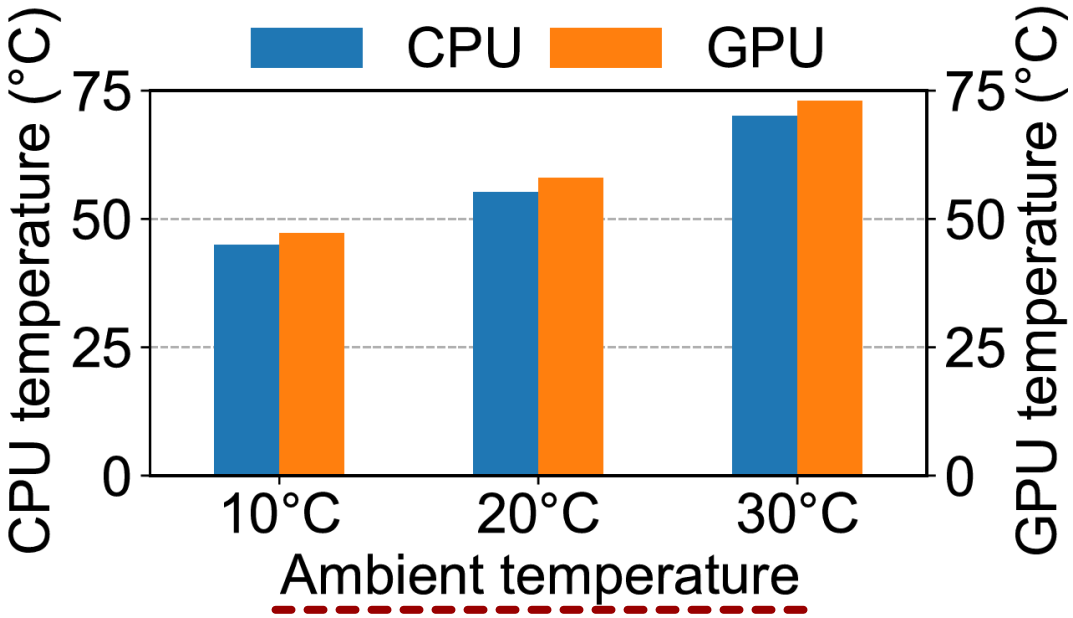
- Limiting the maximum frequency, instead of direct decisions
- Final frequency is still determined by the existing DVFS governor
- Respecting device-specific configurations while mitigating fluctuation issues

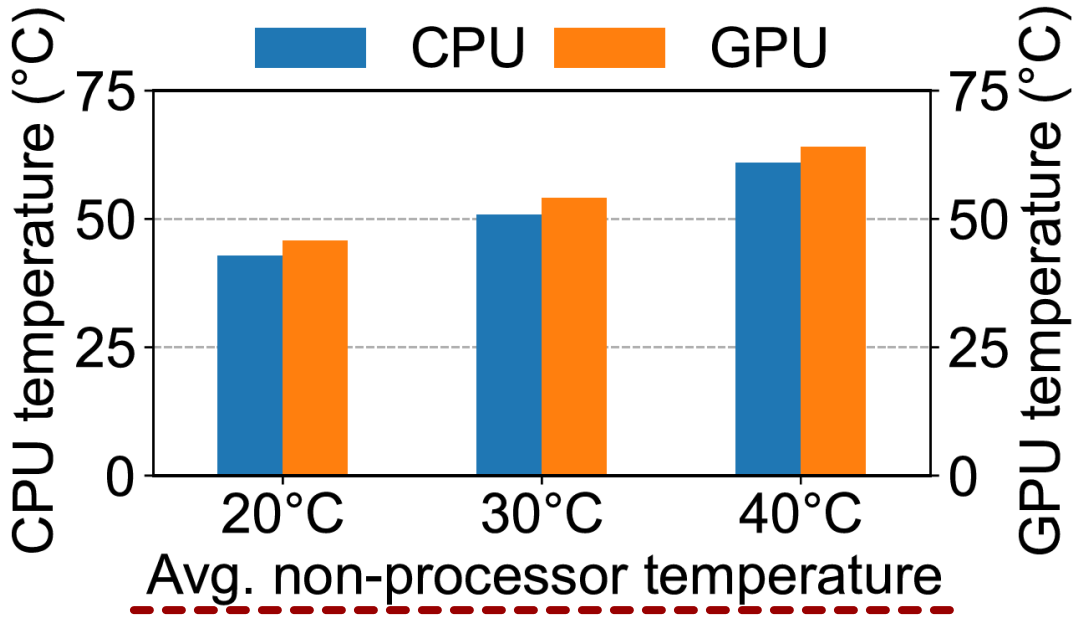


same performance
power consumption ↓
cooling states ↓
temperature ↓

L2. Insufficiency of considering processor temperature alone

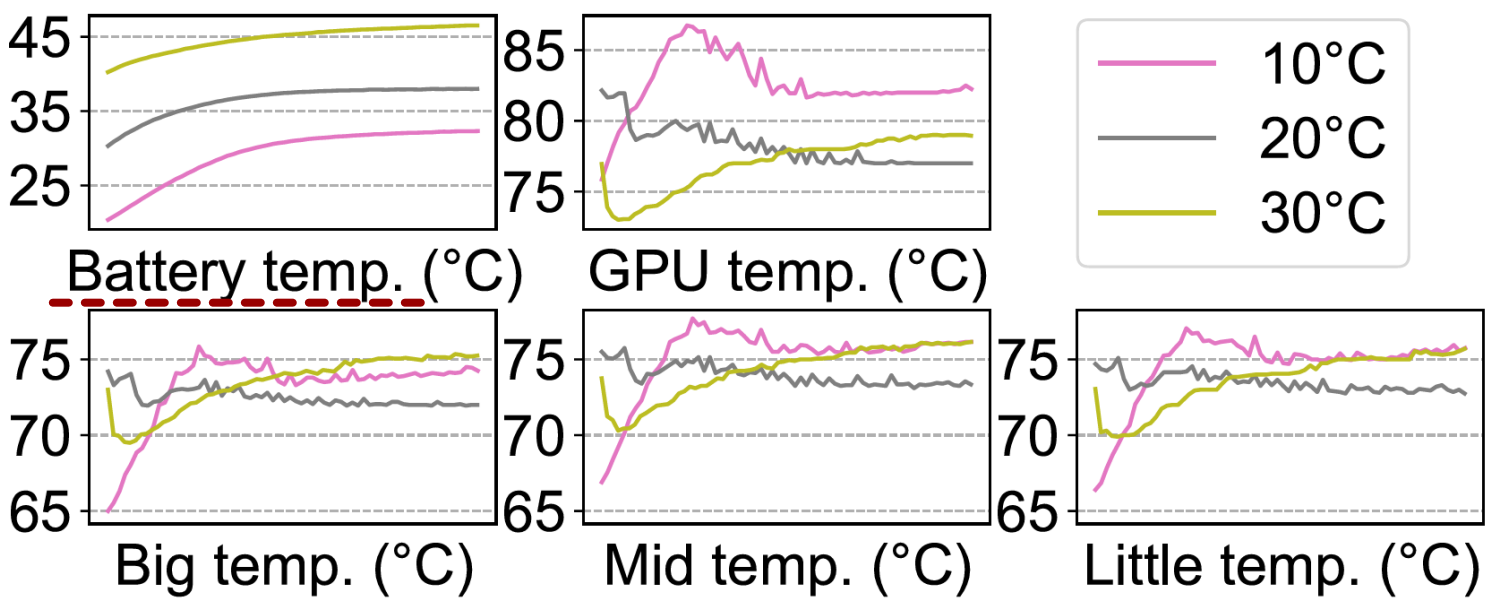
- All existing approaches only consider the temperatures of processors
- Yet, processor temperature is largely impacted by other factors
  - Ambient temperature & temperature of other non-processor components
- But, no ambient temperature sensors & too many non-processor components!

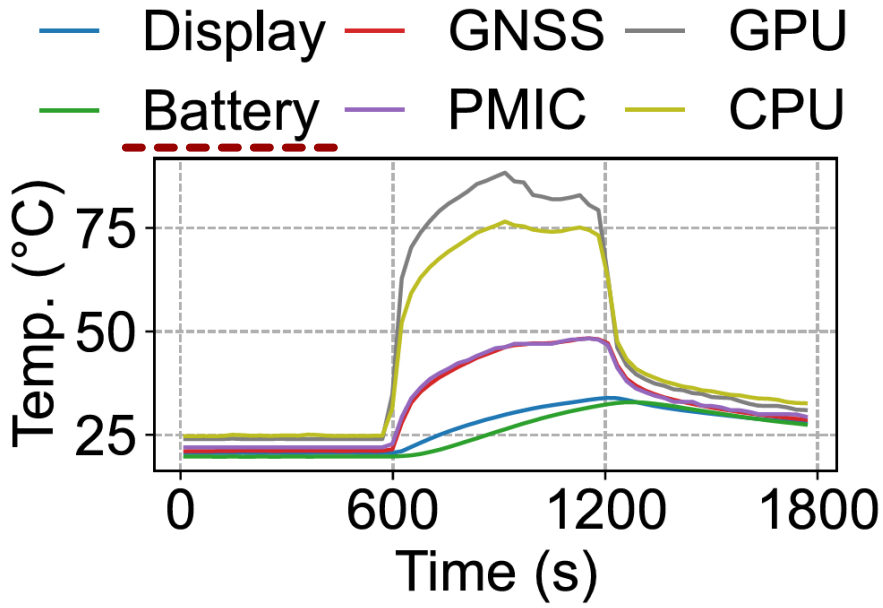




S2. Leveraging battery temperature

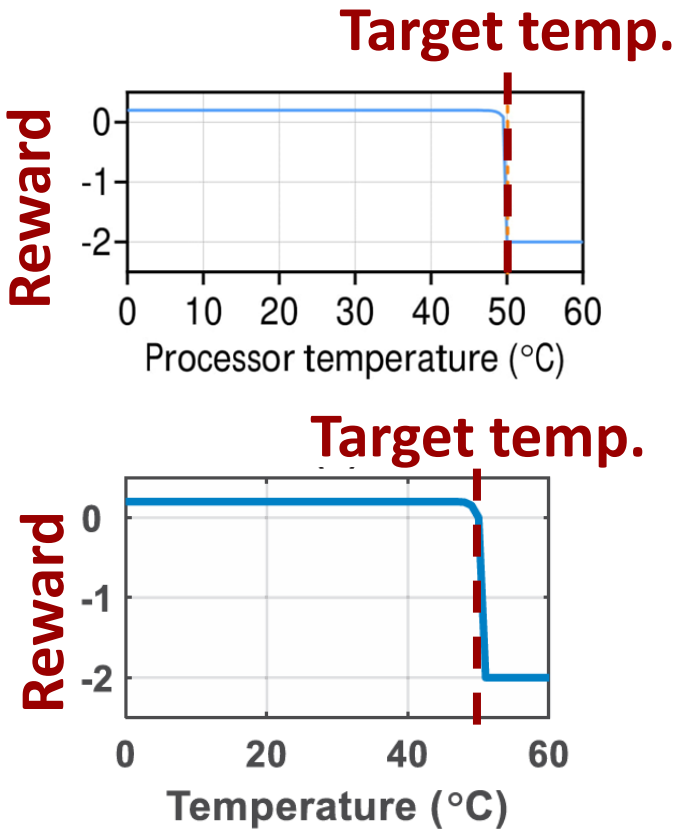
- The battery is the heaviest, largest, and highest heat-capacity component
  - It can track ambient temperature changes
  - It is the key non-processor component driving processor temperature changes

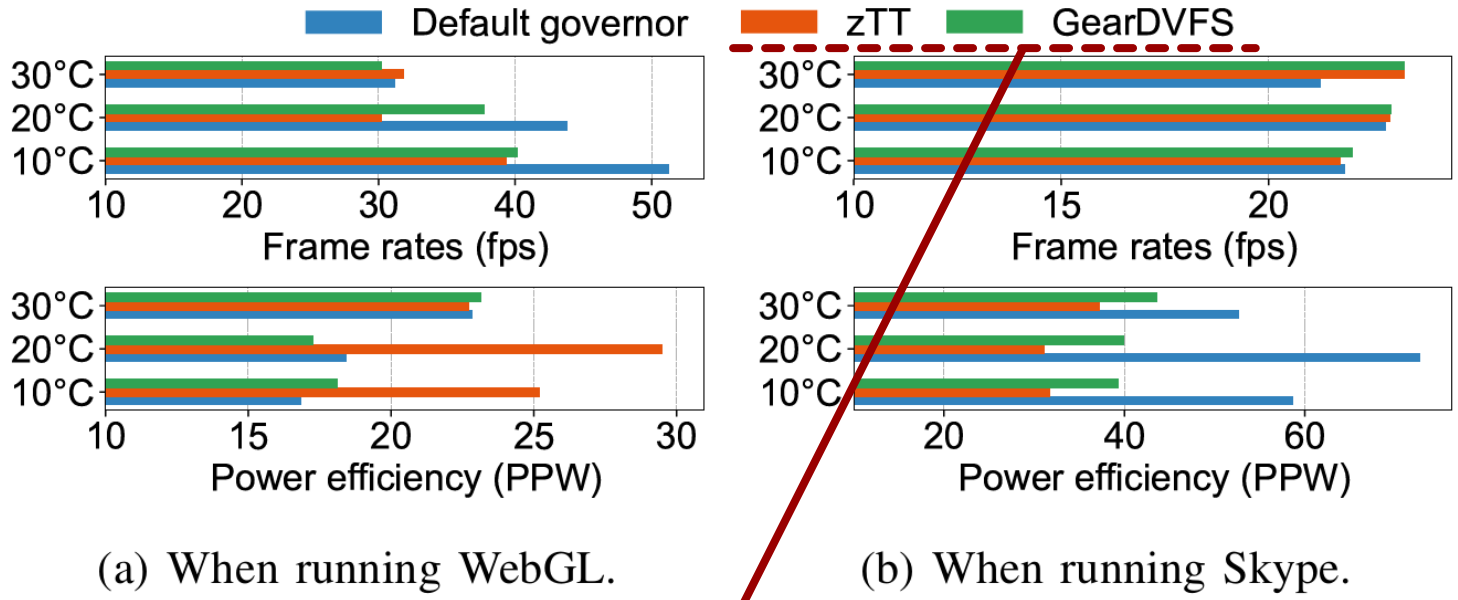




L3. Fixed environment-dependent parameters

- zTT & GearDVFS use a ‘target temperature’ in their reward functions
  - But ideal value depends on the environment, device, and workload!





(a) When running WebGL. (b) When running Skype.

Ineffective under diverse conditions!

S3. Environment-robust RL design

- Our reward function does not include environment-dependent parameters
  - It only includes performance $Q(t)$ and cooling states $C(t)$

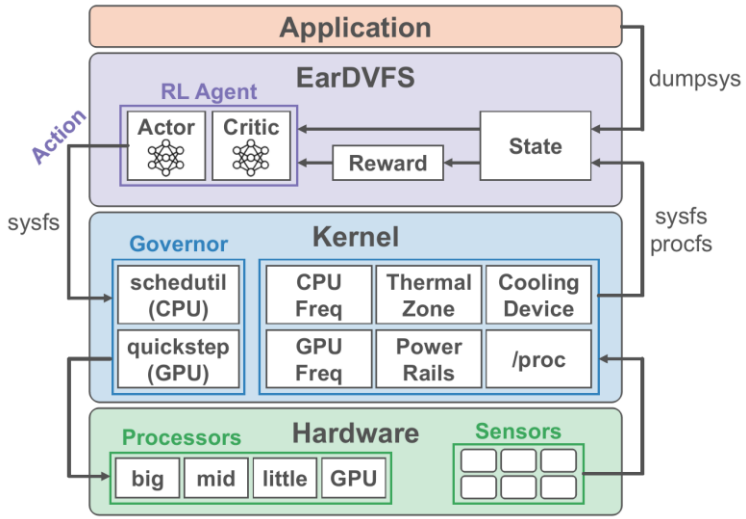
$$reward(t) = Q(t) * \left[\gamma + \left(\frac{M - C(t)}{M}\right)^{\lambda}\right]$$

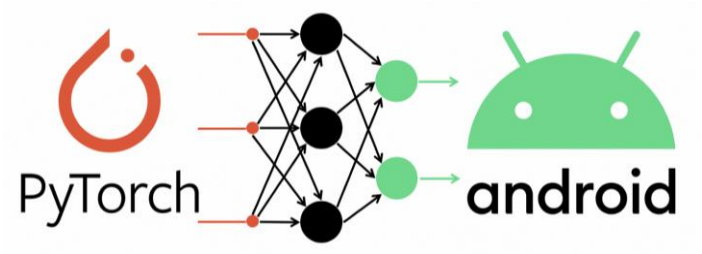
Implementation

- Model
  - Soft Actor-Critic (vs DQN in zTT, GearDVFS) → continuous action space
- Action: (S1) Proactive throttling
  - It determines the freq. of big, mid, little CPU clusters & GPU
- State: (S2) Adding battery temperature
  - Parameters: frequencies, utilizations, temperatures of processors + battery temperature
- Reward: (S3) No environment-dependent parameters
  - Parameters: performance & cooling state

Implementation

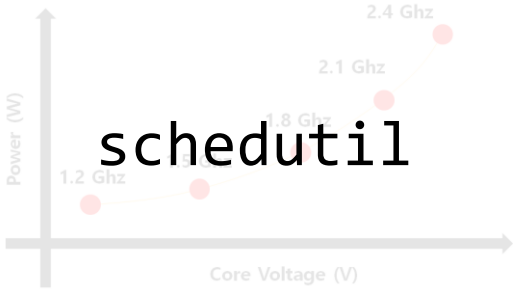
- Implemented in two ways:
  - ADB communication (for training & evaluation)
  - PyTorch Mobile (for verifying deployment feasibility)
- Experiments under strict conditions
  - Controlled ambient temperature
  - All training & testing started at the same thermal state





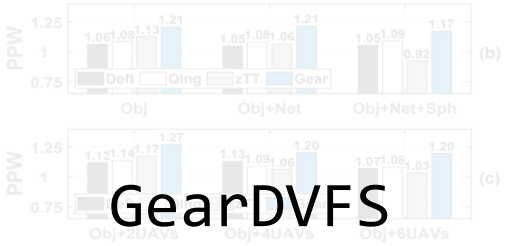
Evaluation

- Compared 4 DVFS methods
  - Method 1. schedutil: Default DVFS governor on Pixel 6
  - Method 2. zTT: RL-based DVFS aiming to completely eliminate thermal throttling
  - Method 3. GearDVFS: RL-based DVFS aiming to maintain 80% processor utilization
  - Method 4. EarDVFS: Our proposed method

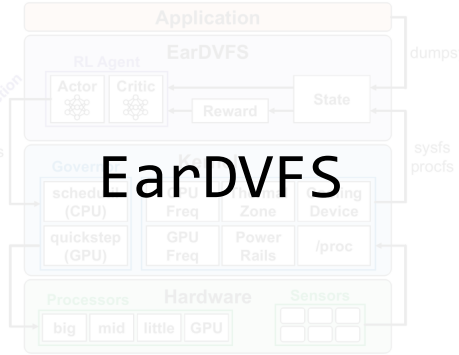


schedutil

zTT



GearDVFS



EarDVFS

(b) Pixel 3a runs each app targeting 60 FPS, which is physically not achievable. While minimizing FPS degradation, zTT saves more power consumption.

Figure 11: Frame rate and total power consumption for mobile apps running on JETSON TX2 and Pixel 3a.

Figure 10: PPW of each method in the mobile applications of (a) self-driving, (b) robot and (c) UAV ground station under various concurrent tasks.

Evaluation

- Tested 4 applications on Pixel 6
  - App 1. WebGL aquarium: CPU-intensive workload
  - App 2. Seascape benchmark: GPU-intensive workload
  - App 3. Genshin Impact: CPU/GPU-heavy workload
  - App 4. Skype: CPU/GPU-light workload

Pixel 6

CPU ↑

App 1

GPU ↑

App 2

CPU & GPU ↑

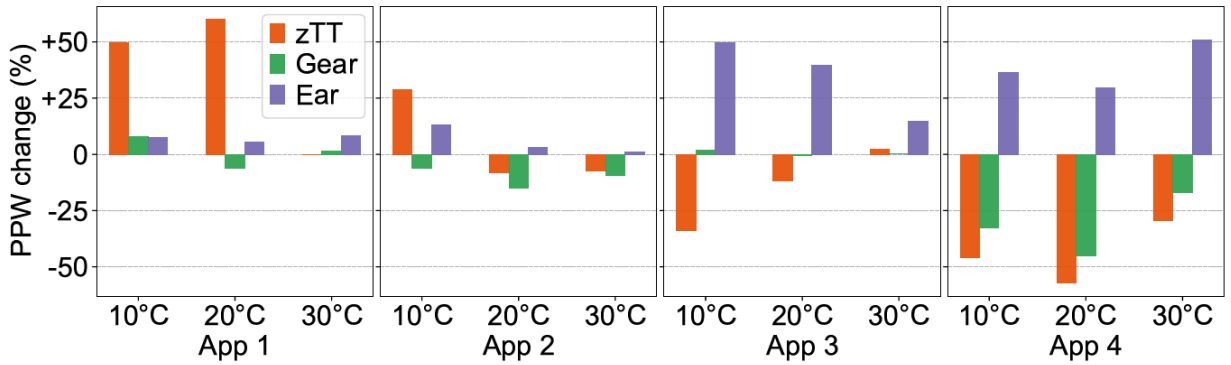
App 3

CPU & GPU ↓

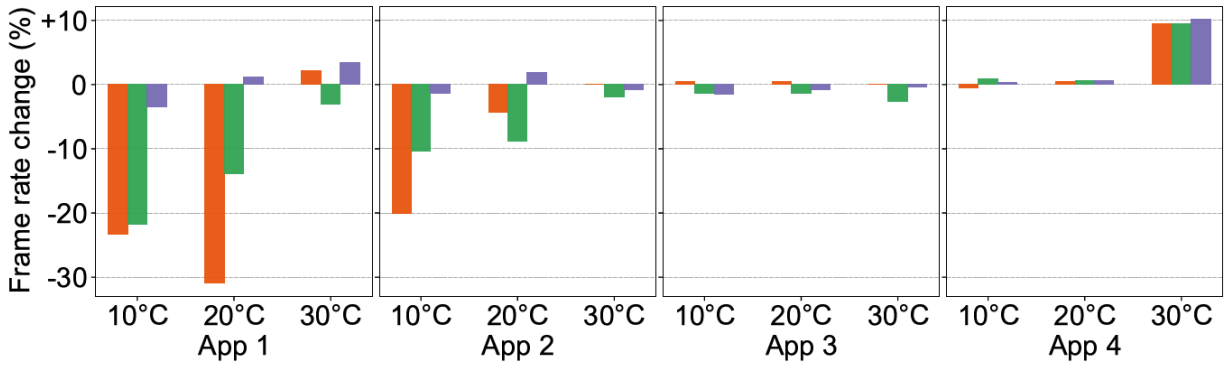
App 4

Evaluation

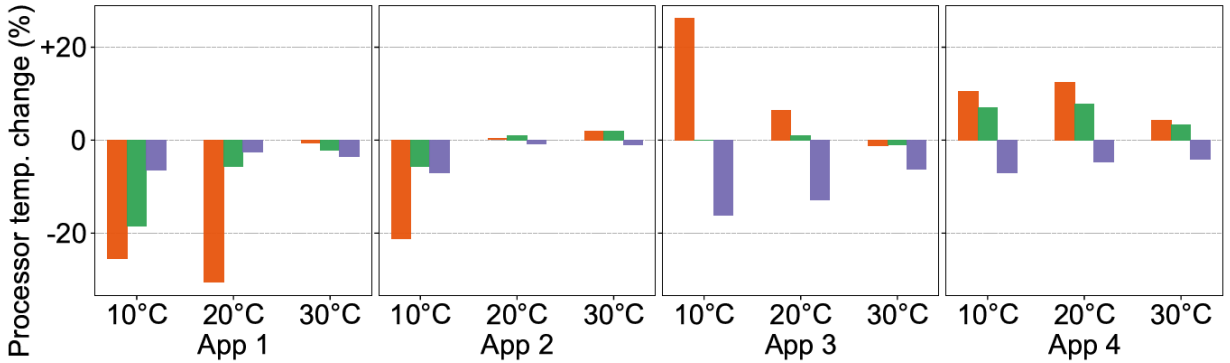
- Overall results
  - Power efficiency 21.6% ↑, performance 0.77% ↑, processor temperature 13.0% ↓



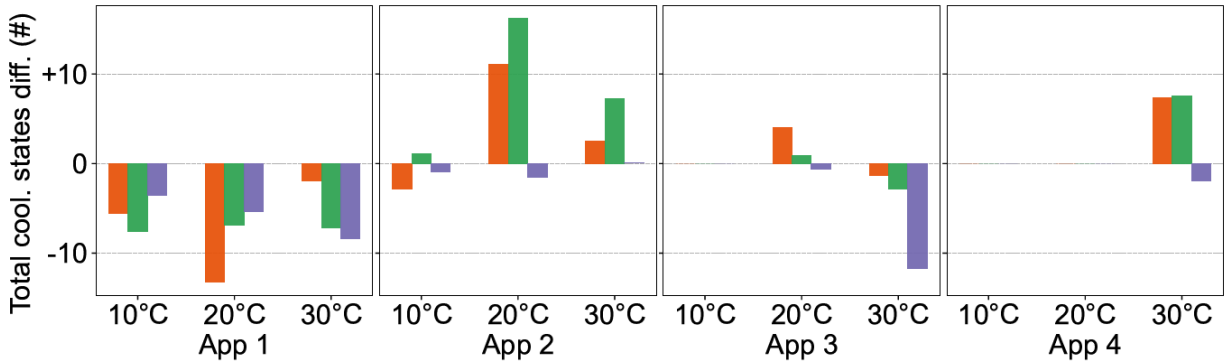
(a) Rate of change in average power efficiency (PPW).



(b) Rate of change in average frame rates.



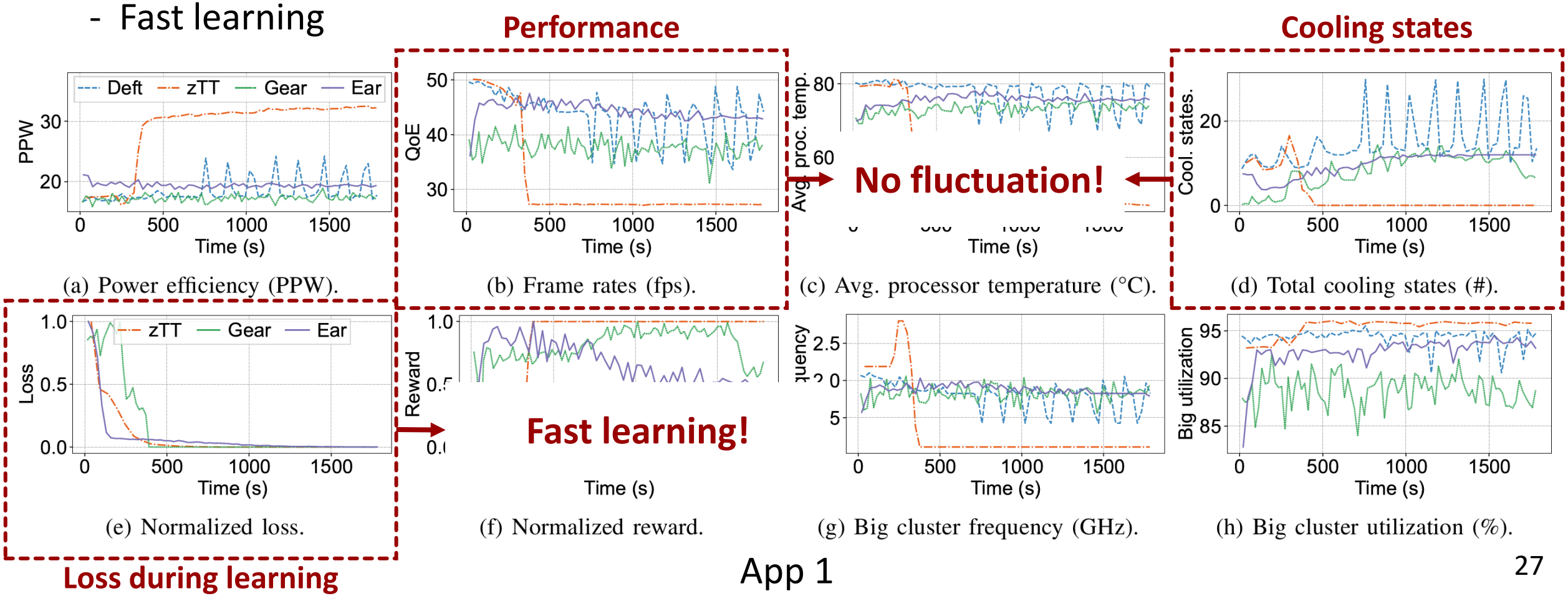
(c) Rate of change in average processor temperature.



(d) Change of average total cooling state levels.

Evaluation

- Run-time behavior
  - No fluctuation
  - Fast learning

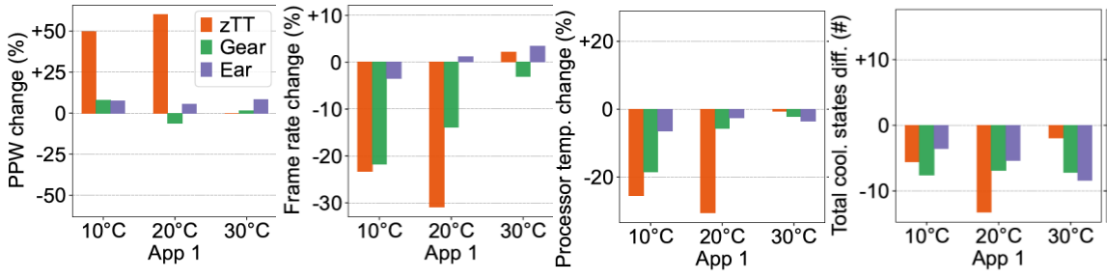


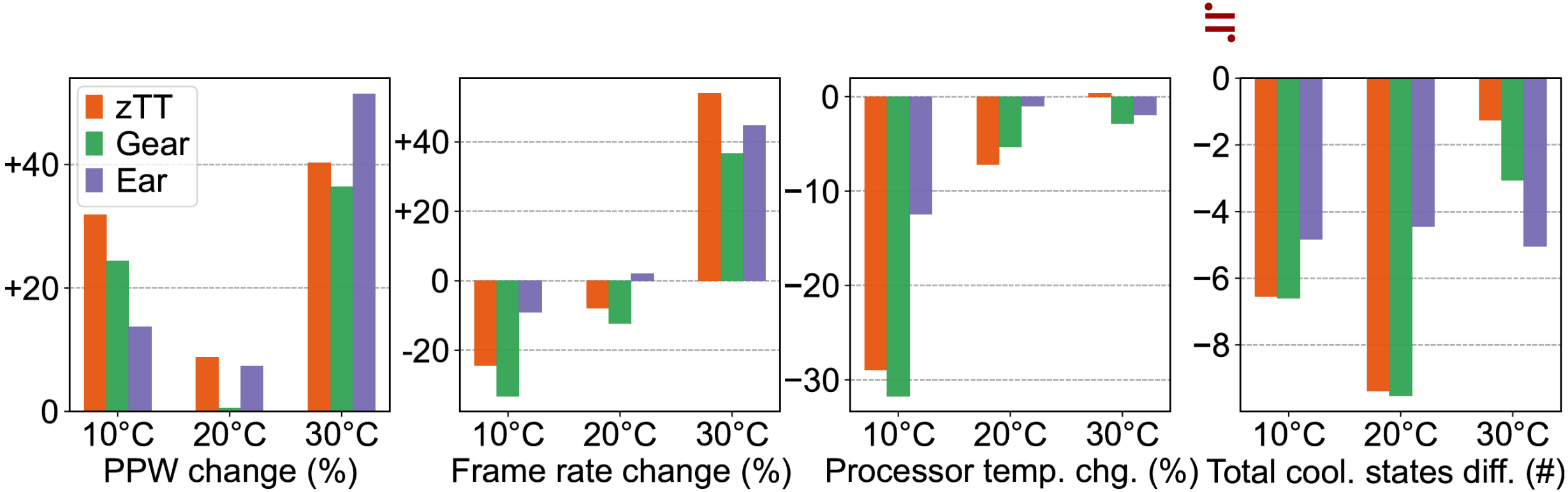
(a) Power efficiency (PPW). (b) Frame rates (fps). (c) Avg. processor temperature (°C). (d) Total cooling states (#).

(e) Normalized loss. (f) Normalized reward. (g) Big cluster frequency (GHz). (h) Big cluster utilization (%).

Evaluation

- Generalization test
  - Tested on Pixel 7
  - Yielded results similar to Pixel 6

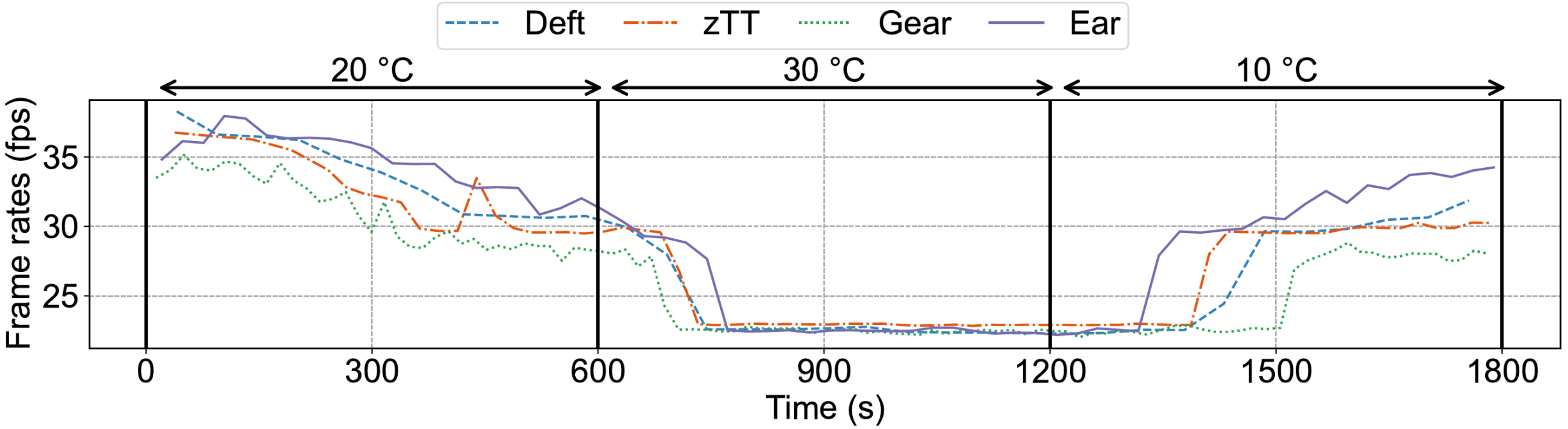




App 1

Evaluation

- Real-time ambient temperature changes
  - Lower CPU frequency selection in GPU-intensive workload (40.8% lower little cluster frequency than the default governor)
  - Quick response to ambient temperature changes



App 2

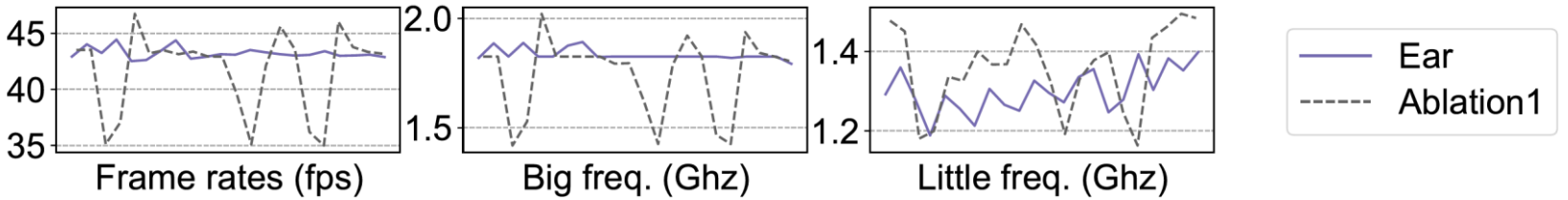
Ablation study

- Ablation study: removes a component to quantify its contribution
  - Action: no proactive throttling (direct frequency decision)
  - State: no battery temperature
  - Reward: no cooling state

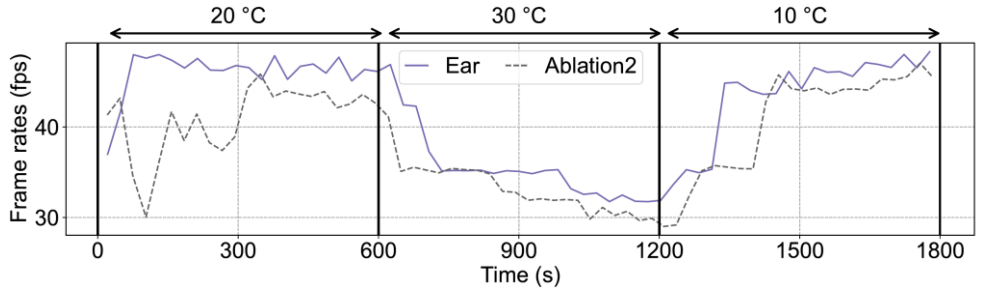
Ablation study

- Ablation study summary
  - Action: No proactive throttling → fluctuation issue
  - State: No battery temperature → slower response to ambient changes
  - Reward: No proactive cooling state → fluctuation issue

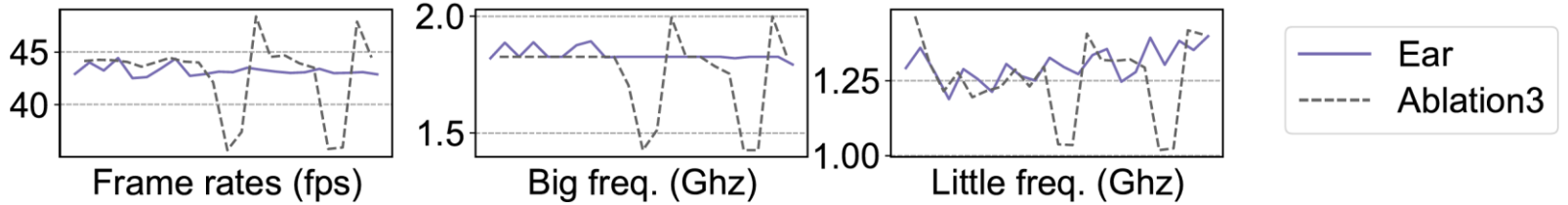
Action



State



Reward



Summary

- EarDVFS: Environment-Adaptable RL-based DVFS
  - Identified limitations of existing RL-based DVFS methods and proposed solutions
    - S1. Proactive throttling
    - S2. Leveraging battery temperature
    - S3. Fixed environment-dependent parameters
  - Evaluated our proposed EarDVFS through rigorous and diverse experiments
    - Maintained performance while improving average power efficiency by 21.6%
    - Validated the necessity of each designed component through ablation study

AJOU UNIV.
1973

Q & A

Jaeheon Kwak

jhkwak@ajou.ac.kr