
# Thermal-Aware Servers for Real-Time Tasks on Multi-Core GPU-Integrated Embedded Systems

Seyedmehdi Hosseinimotlagh and Hyoseung Kim

University of California, Riverside

{shoss007, hyoseung}@ucr.edu

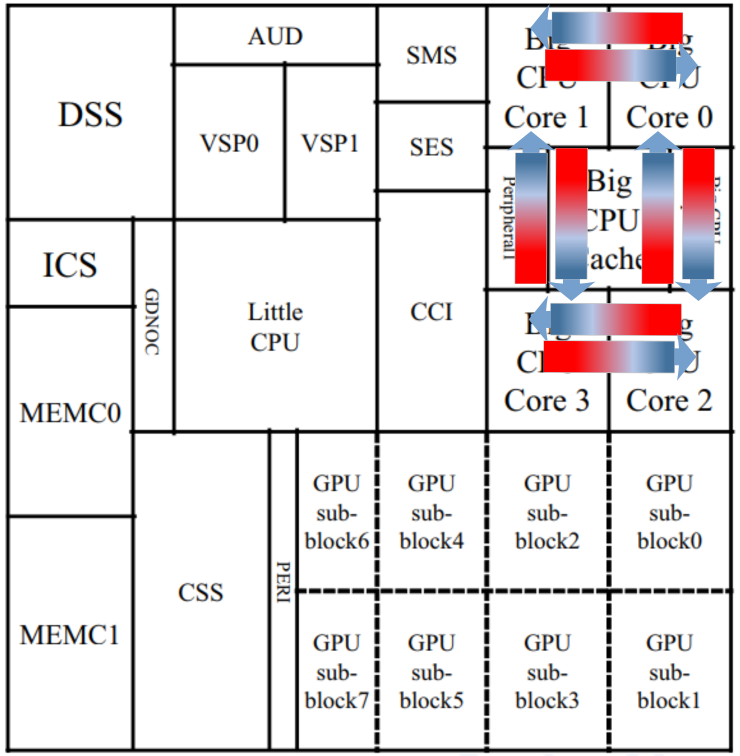
# Motivation

- The recent trend in real-time applications raises the demand for powerful embedded systems with GPU-CPU integrated systems-on-chips (SoCs)

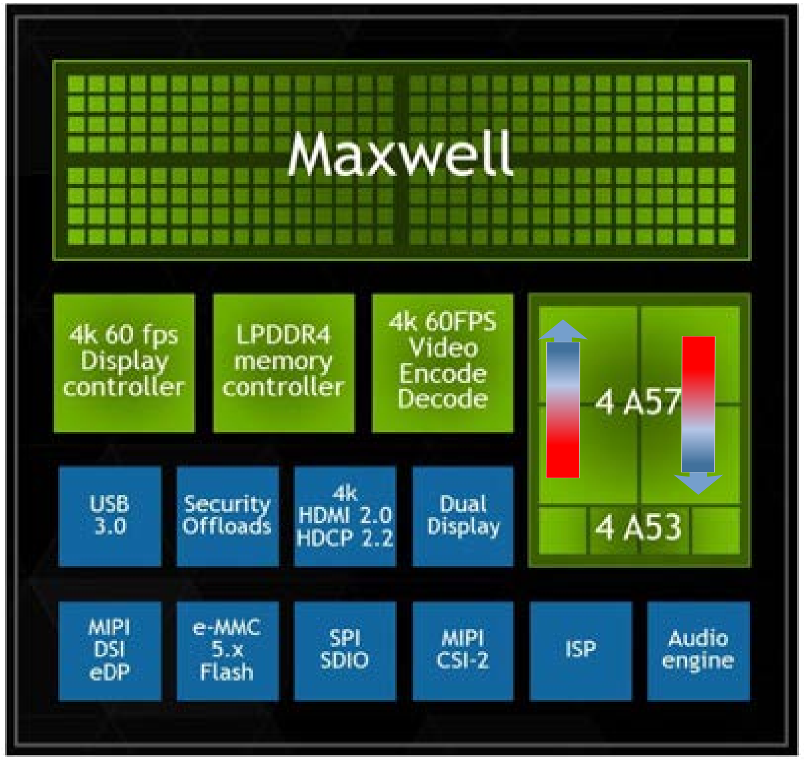

- High power demand ⇒ temperature increase
  - Power increase (rapid battery drain)
  - System reliability
  - Physical harm in real-time implantable medical devices

# Thermal issues in heterogenous multi-core SoCs

- Heat conduction
  - Inter-component heat transfer (between CPUs, GPU, etc.)
  - Intra-component heat transfer (between CPU cores, SMs, etc.)




Exynos 5422 SoC †




Tegra X1 SoC

† Y.H. Gong et. al (2018). Thermal modeling and validation of a real-world mobile ap. *IEEE Design & Test.*

# Thermal management approaches

- Dynamic Thermal Management(DTM)
  - Thermal violation⇒ Frequency throttling or shutdown

    **Timing unpredictability**

- Dynamic Voltage Frequency Scaling (DVFS)
  - Adjusts frequency according to application needs and heat generation

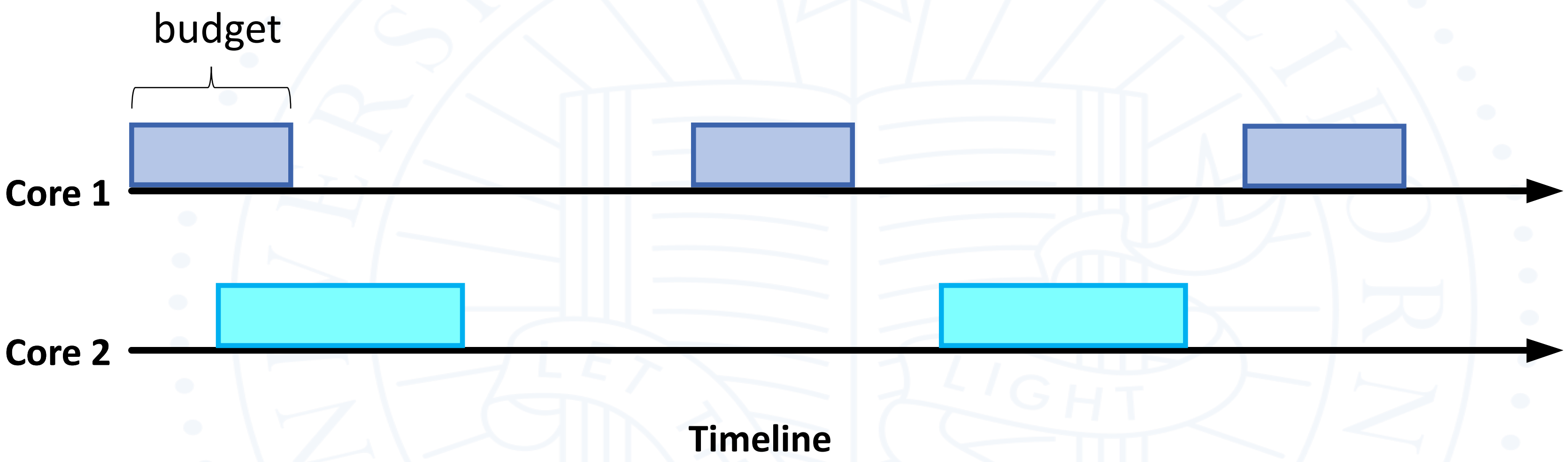
    **Electronic device stress (reliability decrease)** †

    **Not supported by all components of SoCs**

† A.Iranfar et, al. (2018). Thespot: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip. *Computer-Aided Design of Integrated Circuits and Systems*

# Thermal management approaches

- **Thermal Isolation Servers**†: isolate the thermal effect **spatially** and **temporally**
  - Spatial: heat generated from tasks executing from other CPU cores
  - Temporal: heat generated from tasks previously executed on the same CPU core

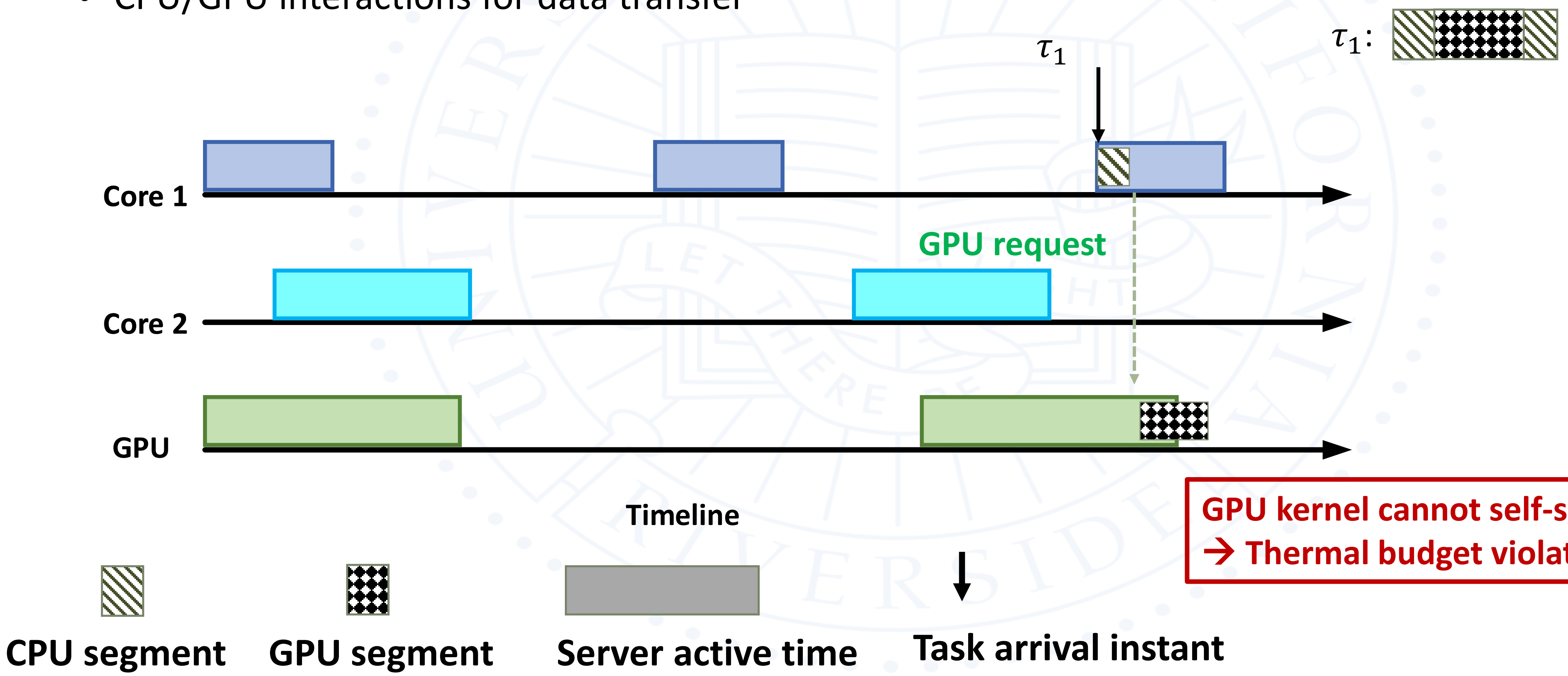



- Benefits:
  - Bounds the temperature increase caused by a set of tasks
  - Guarantees both timing and thermal constraints

† R. Ahmed et. Al (2017). On the design and application of thermal isolation servers. *(TECS)*

# Limitations

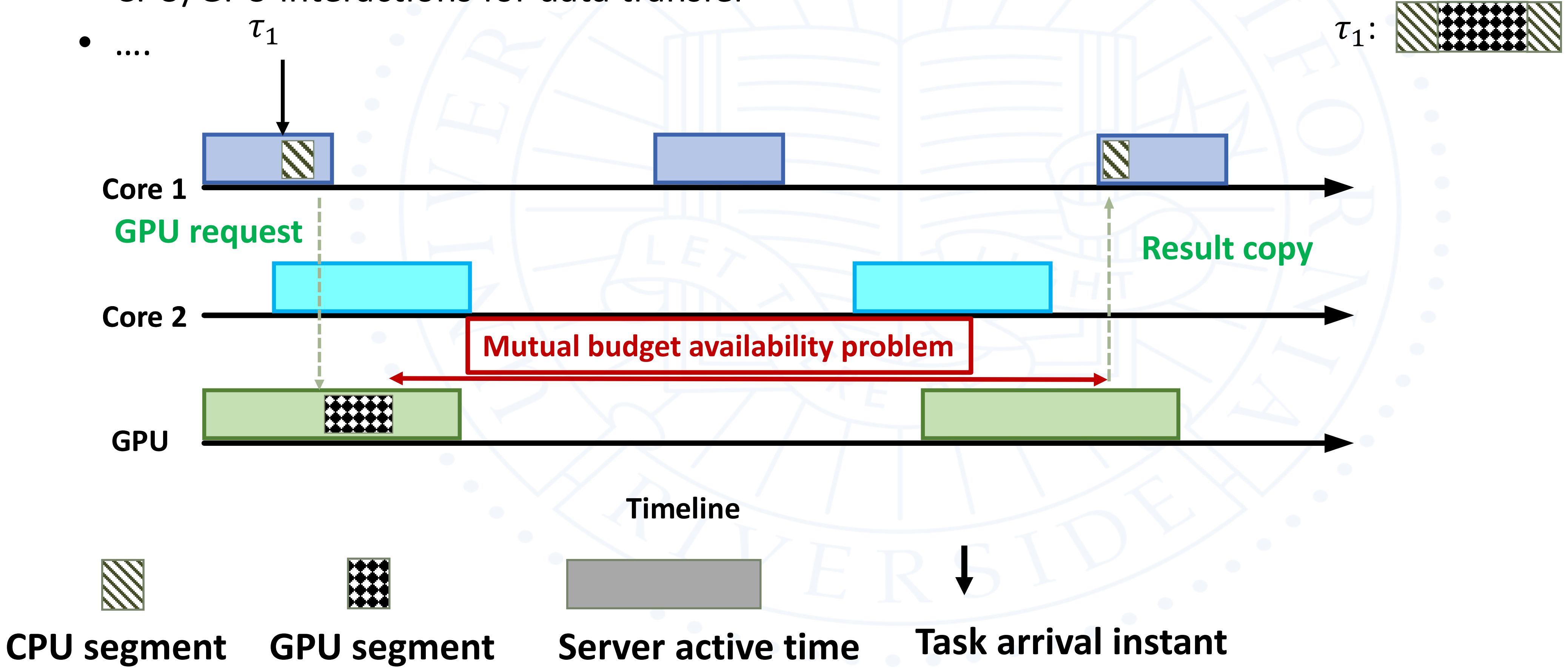
- Supports only periodic server policy for budget replenishment
- Cannot handle GPU-using tasks on *the shared GPU*
  - Non-suspendable GPU kernel execution
  - CPU/GPU interactions for data transfer




**GPU kernel cannot self-suspend**
**→ Thermal budget violation!**

# Limitations

- Supports only periodic server policy for budget replenishment
- Cannot handle GPU-using tasks on *the shared GPU*
  - Non-suspendable GPU kernel execution
  - CPU/GPU interactions for data transfer
  - ….



# Contributions

- **Thermal-aware CPU-GPU server framework**
  - Characterize different timing penalties for various replenishment policies (polling, deferrable, sporadic) for CPU cores
  - GPU thermal-aware server to launch non-suspendable GPU kernels
  - Present a protocol for CPU and GPU thermal-aware servers

- Improvements
  - **Misc. GPU operation budget reservation** for reducing **CPU-GPU data handover delay**
  - Waiting queue design of the GPU server for mitigating **remote blocking**

# Outline

- Introduction & background
- Proposed framework
  - System model
  - CPU/GPU server design
  - Timing analysis
  - Improvement techniques
- Evaluation
  - Server implementation
  - Schedulalibity tests on generated tasksets
- Case study
- Conclusion & future work

# System model

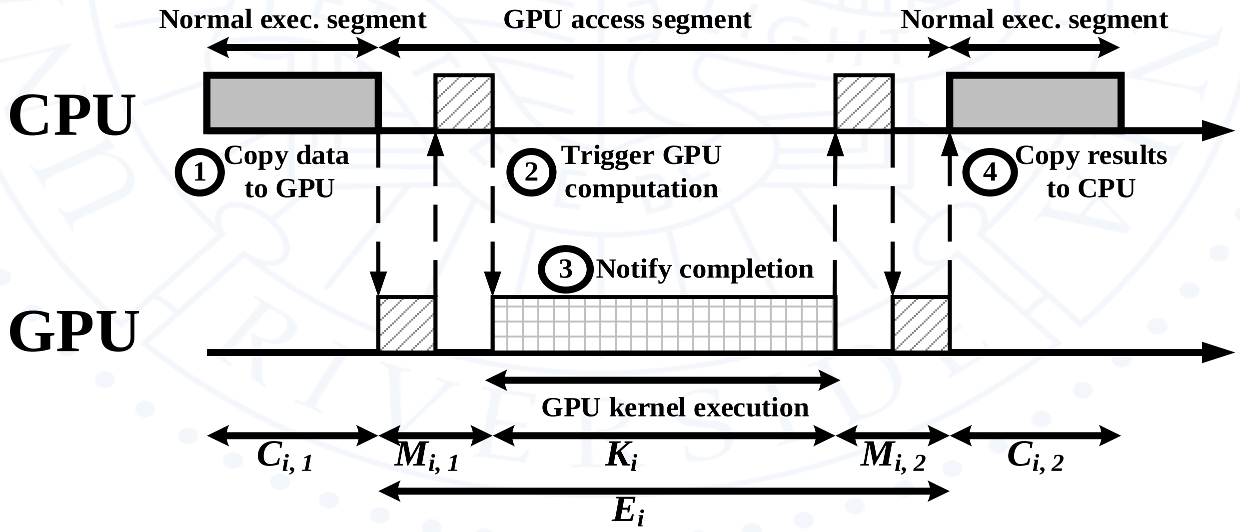
- Partitioned fixed-priority scheduling for sporadic tasks
- Server: $v_i = (C_i^v, T_i^v)$
  - $C_i^v$ : Maximum execution budget
  - $T_i^v$: Budget replenishment period
  - One thermal-aware server for each CPU core and the GPU
- CPU budget replenishment policies:
  - Polling
  - Deferrable
  - Sporadic
- Task $\tau_i = \left((C_{i,1}, E_i, C_{i,2}), T_i, s_i\right)$
  - $C_{i,1}$ : WCET of the first normal CPU execution segment
  - $E_i$ : WCET of the GPU access segment
  - $C_{i,2}$ : WCET of the second normal CPU execution segment
  - $T_i$ : minimum inter-arrival time
  - $s_i$: CPU-only/GPU-using task indicator (0 is CPU-only, 1 is GPU-using task)

# GPU execution model

- GPU segment ($E_i$):
  - $M_{i,1}$: Data copy (CPU-GPU intervention)
  - $K_i$: Pure GPU kernel
  - $M_{i,1}$: Result copy (CPU-GPU intervention)

Miscellaneous operations

- $E_i = M_{i,1} + K_i + M_{i,1}$
- Modeled as a critical section protected by a ***suspension-based*** mutually-exclusive lock
- Pending GPU requests are put in a waiting queue



# Thermal-aware CPU-GPU server protocol

1. Pending GPU requests are inserted to **a priority queue**
2. To handle the GPU request of a task $\tau_i$, there must exist **at least $E_i$ budget** available on the GPU server

   *One complete GPU segment*

   - **Reason**: non-suspending characteristics of the GPU
3. GPU request **boosts the priority** of the corresponding task to the highest-priority level
   - **Reason**: to reduce remote blocking time for other tasks waiting for the shared GPU
4. GPU server uses the **sporadic server** policy (CPU can use polling/deferrable/sporadic)
   - **Reason**: to avoid thermal back-to-back effect & long waiting time

- CPU/GPU Server budget determined analytically

Max temp. threshold of *i*-th core; Conductivity coef.; GPU thermal effect

$$\begin{cases} \theta_M^i = \lambda_i[\alpha + (\theta_s - \alpha)e^{\beta t_{wk}}] + \underbrace{\gamma_{i,g}[\alpha^g + (\theta_s^g - \alpha^g)e^{\beta^g t_{wk}^g}]}_{\text{GPU thermal effect}} \\ \theta_M^g = \alpha^g + (\theta_s^g - \alpha^g)e^{\beta^g t_{wk}^g} + \underbrace{\underbrace{\sum_{j=1}^{m} \gamma_{g,j}}_{\gamma^g}[\alpha + (\theta_s - \alpha)e^{\beta t_{wk}}]}_{\text{CPU thermal effect}} \end{cases}$$

Max temp. threshold of GPU; Steady state temp.; CPU thermal effect

# Task schedulability analysis

Worst-case response time

Local and remote blocking times

CPU-GPU handover delay

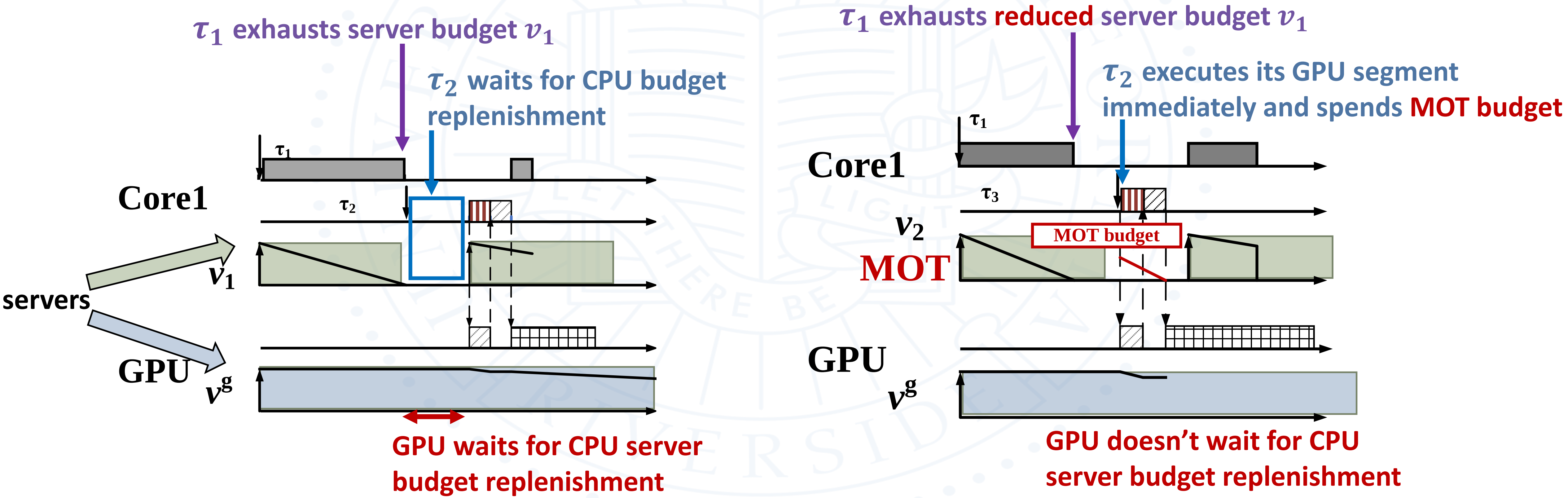
$$W_i^{n+1} = C_i + B_i^l + B_i^r + H_i^{gc} + \left\lceil \frac{W_i^n + C^c - s_i(H_i^{gc} + K_i)}{T^c} \right\rceil (T^c - C^c) + \sum_{\substack{\tau_h \in V(\tau_i) \\ h>i}} \left\lceil \frac{W_i^n + J^c + (W_h - C_h) - s_i(H_i^{gc} + E_i)}{T_h} \right\rceil C_h$$

Server budget and budget replenishment period

Preemption delay of higher-priority tasks on the same core

# Miscellaneous operation time (MOT) reservation

- Reserves a small portion of the CPU server budget for miscellaneous operations ($M_{i,1}$ & $M_{i,2}$)
  - **Reason:** to reduce the **CPU-GPU handover delay**: GPU does not need to **wait for the budget replenishment of the CPU** server during the data transmission phase




Without MOT

With MOT

# Remote blocking enhancement

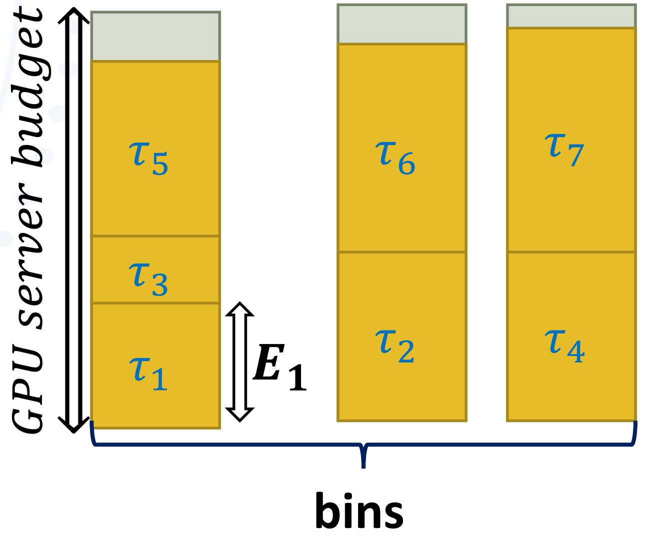
- Problem**: Long remote blocking time -** In the worst case, **every GPU-using segment** causes **GPU budget depletion**

$$B_i^{r,n+1} = \max_{\substack{l<i \\ s_i>0}} W_l' + \sum_{\substack{h>i \\ s_i>0}} \left( \left\lceil \frac{B_i^{r,n}}{T_h} \right\rceil + 1 \right) . W_h'$$

$$W_i' = H_i^{gc} + E_i \qquad H_i^{gc} = s_i(T^g - C^g + 2T^c)$$

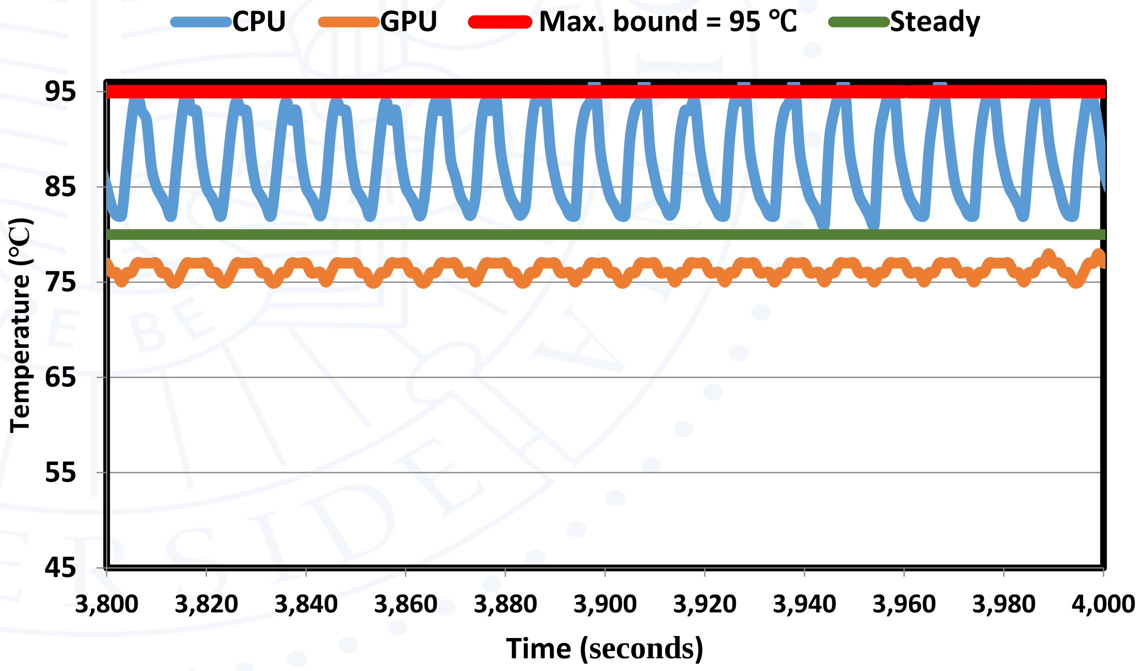
**Large due to non-preemptivity/suspendability of GPU**

- Intuition: Handles multiple small GPU segments with ONE GPU budget
- Solution: Pre-define GPU servicing order w/ a bin-packing approach
  - Each bin represents **ONE** GPU budget replenishment period



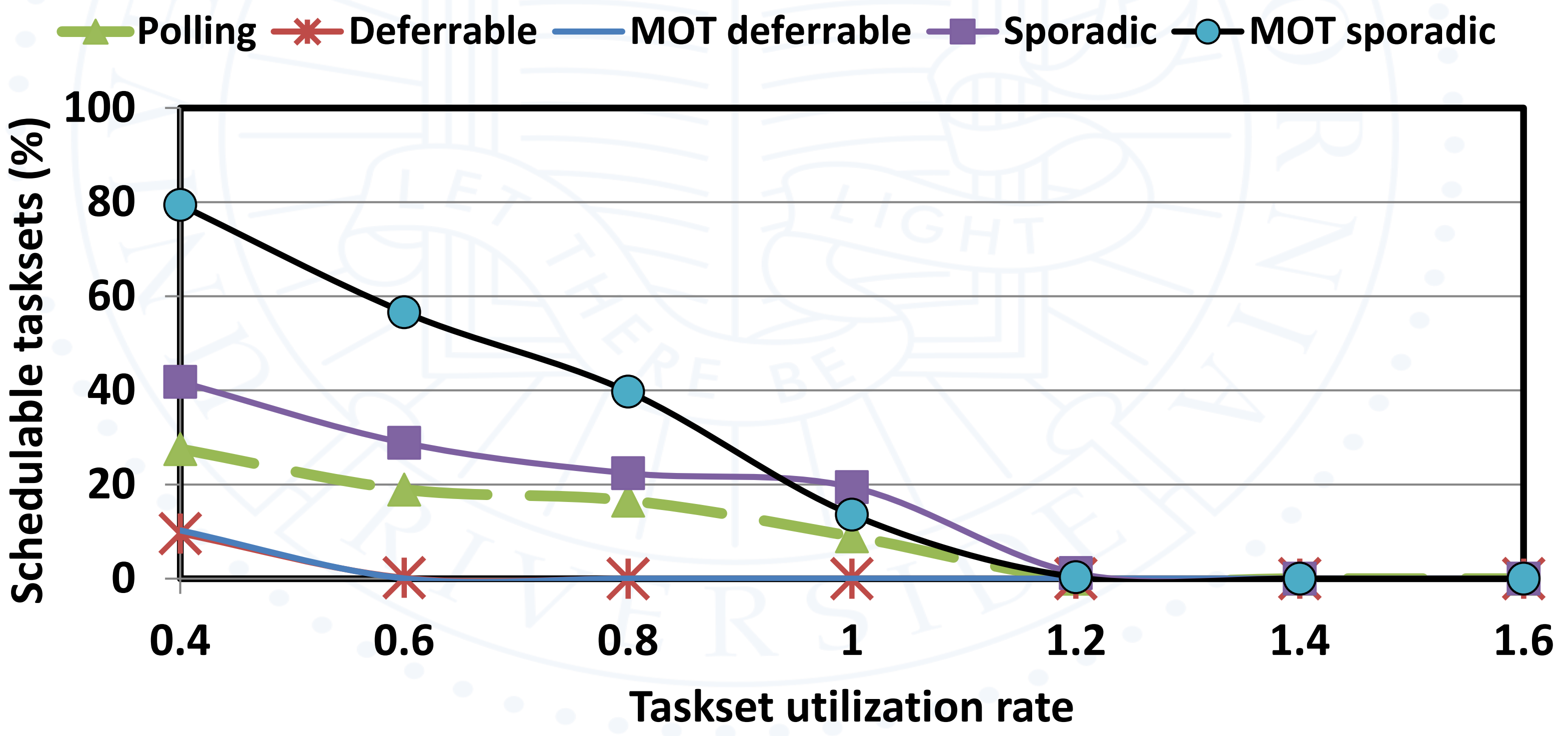
# Evaluation

- Platform ODROID-XU4
  - 4 Cortex-A15 cores (big cluster)
  - 4 Cortex-A7 cores (little cluster)
    - Used for only system maintenance & monitoring processes, etc.
  - Integrated Mali-T628 GPU
  - Built-in temperature sensors in big cluster and the GPU
  - DTM throttles the frequency to 900 MHz
- Benchmark
  - Mali SDK benchmark



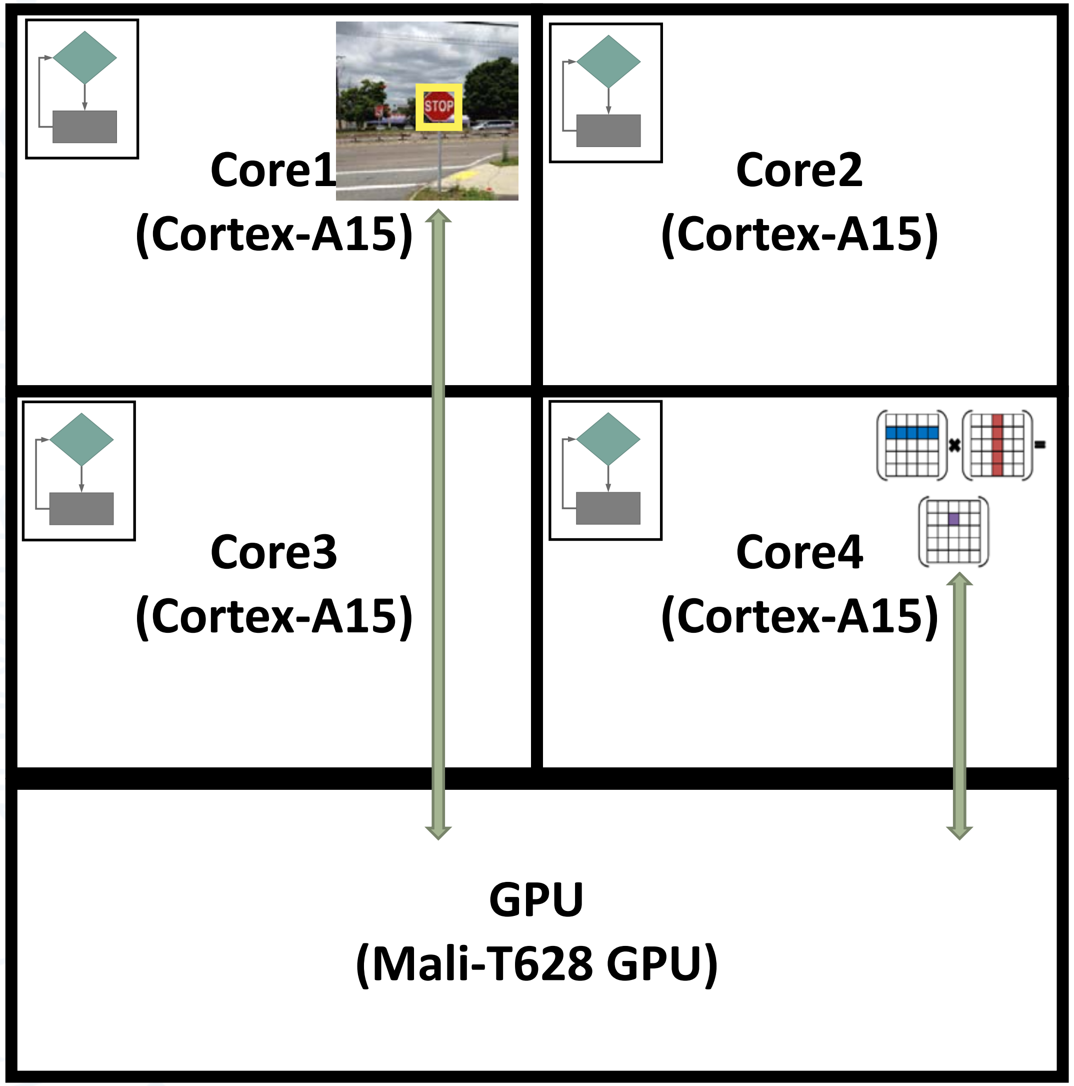
# Schedulability experiments

| Parameters | Values |
| --- | --- |
| Number of CPU cores | 4 |
| Number of tasks | [8, 20] |
| Taskset utilization | [0.4, 1.6] |
| Task period and deadline | [30, 500] ms |
| Percentage of GPU-using tasks | [10, 30] % |
| Ratio of GPU segment len. to normal WCET | [2, 3]:1 |
| Ratio of misc. operations in GPU segment $\frac{M_{i,1}+M_{i,2}}{E_i}$ | [10, 20]% |



# Case study

- 3 types of applications:
  - RT GPU-using task (core 1)
    - Highway workzone recognition application for autonomous driving
    - 800-frame video (rendering repeatedly)
    - 8 frames per second
  - Non-RT CPU-only tasks (all cores)
    - Lowest priority
  - Non-RT GPU-using tasks (core 4)
    - Matrix multiplication



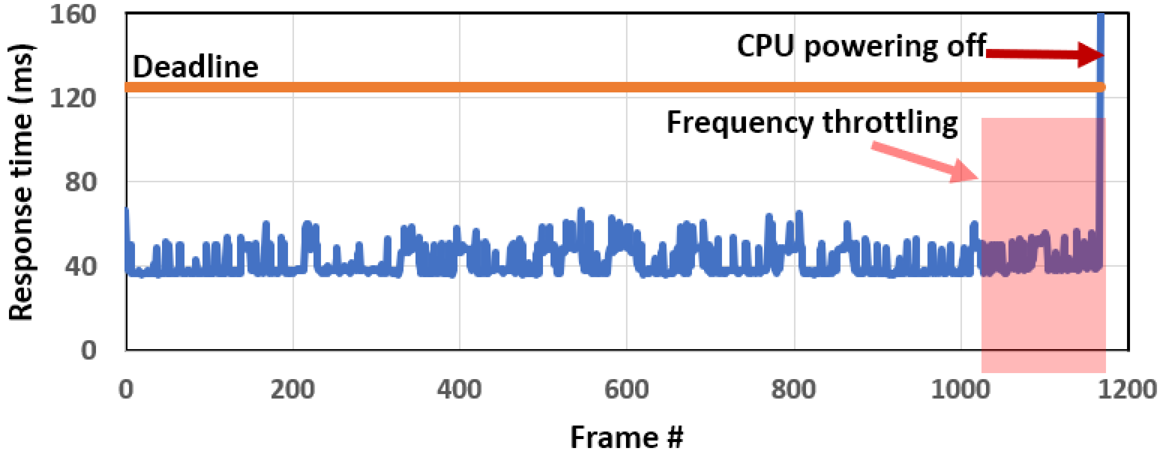
# Baseline *vs.* proposed framework

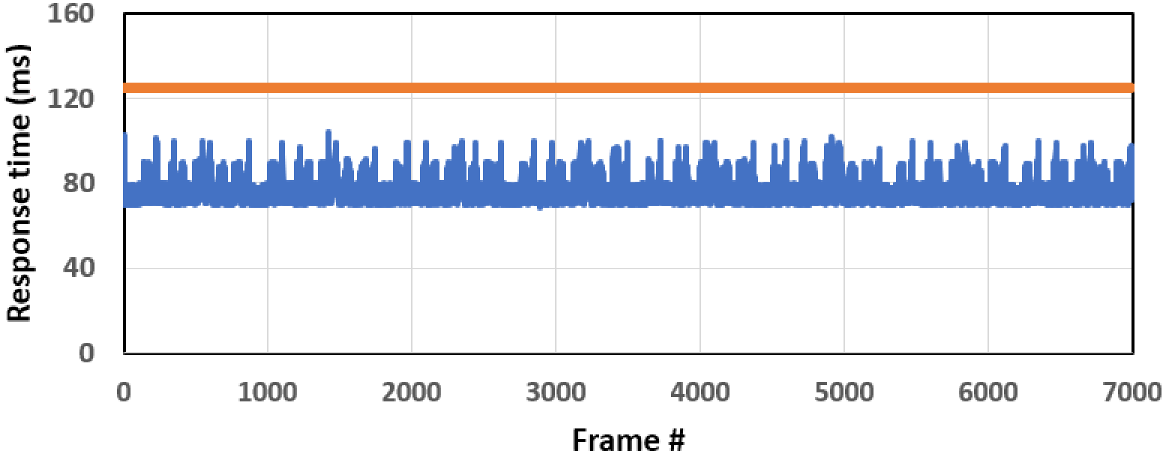
## Baseline

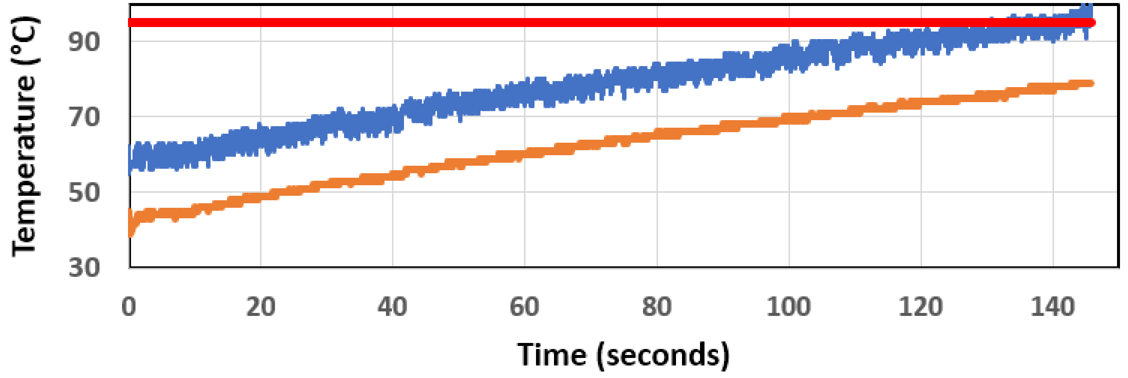
- DTM was triggered after processing around 1040 frames
- CPU powering off after three minutes

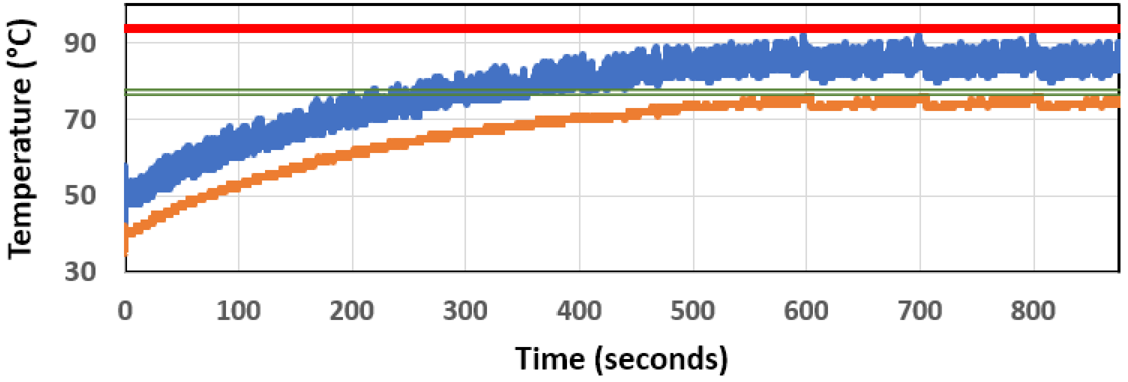
## Proposed framework

- Longer but bounded response time
- Tightly bound of operating temperature by the thermal threshold



# Conclusion

- Proposed a thermal-aware framework to bound the maximum temperature of CPU cores and shared GPU
  - Guarantees real-time schedulability
  - Provides analytical foundations to check both thermal and temporal safety
  - Enhancement designs
    - *Miscellaneous operation time reservation* mechanism
    - Waiting queue design for packed GPU kernel execution
- Future work
  - Thermal behavior of GPU-using tasks based on the type of resources used by their kernels
    - GPU kernel frequently accessing local memory may generate much less heat
  - Multiple preemptible thermal-aware servers on each CPU/GPU core
  - Data-driven characterization of hardware thermal parameters

# Thank You