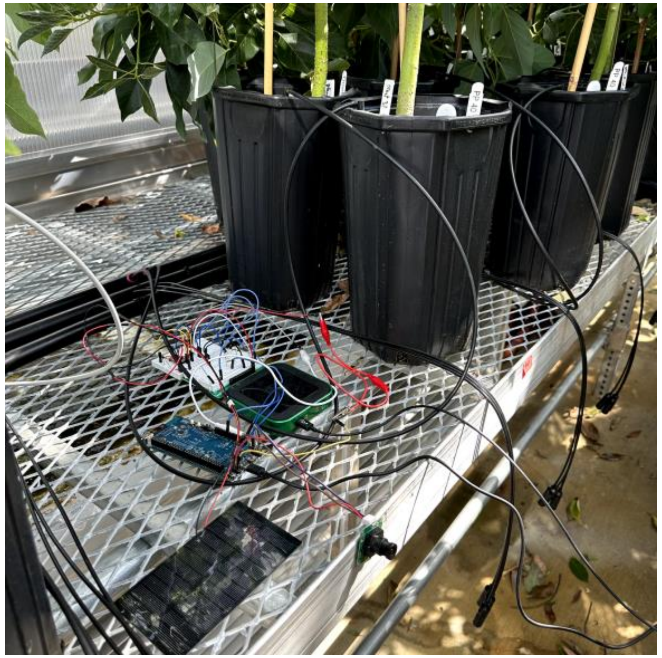
MII: A Multifaceted Framework for Intermittence-aware Inference and Scheduling

Ziliang(Johnson) Zhang, Cong Liu, Hyoseung Kim

University of California, Riverside

ESWEEK 2024

Intermittently Powered Devices (IPDs)



What are IPDs?

- Battery-less devices rely on ambient energy for power
- Intermittence: devices turn on and off constantly

Why IPDs?

- **Adaptability:** can operate in human-inaccessible locations
- **Sustainability:** zero carbon emission and environment-friendly
- **Permanence:** maintenance-free (no batteries)

Inference tasks on IPDs (Intermittent Inference)

Why run inference tasks on IPDs?

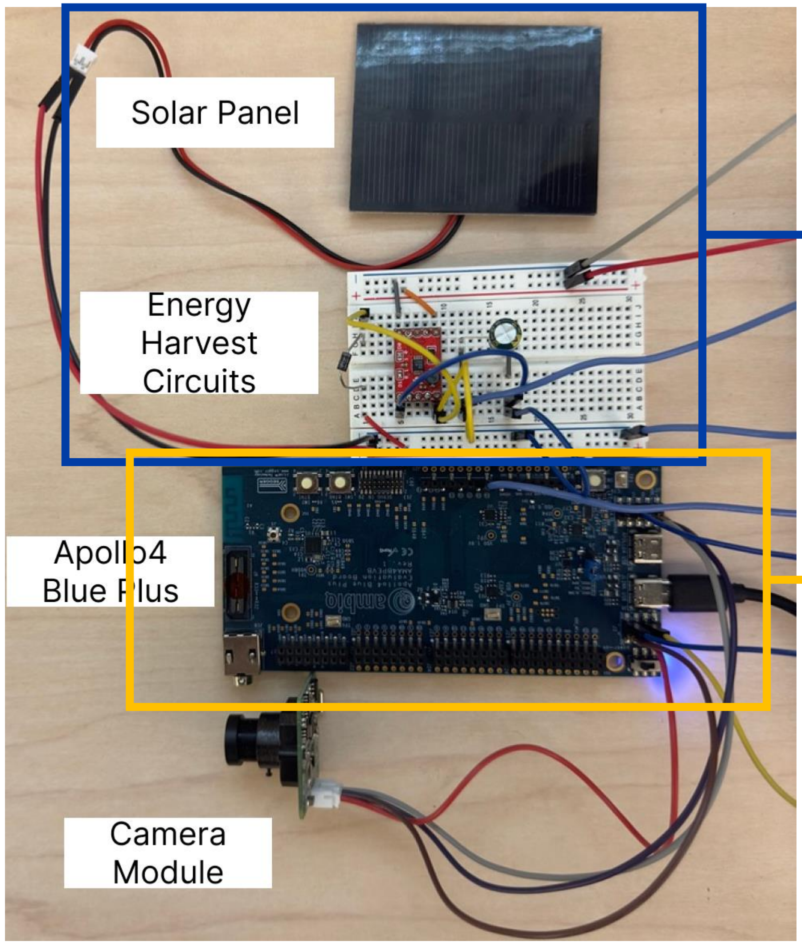
- **Costly Communication**: over 1 hour to send a single MNIST image to server. But 10s to complete the inference locally and send the result to server [1].
- **Data privatization**: keeps data locally for privacy and safety

Why inference on IPDs is challenging?

- **Intermittence:** inference progress lost when device powers off
- **Small SRAM:** cannot fit the entire layer of data
- **Timing constraints:** Tasks run periodically. Periods are arbitrary deadlines

[1] Graham Gobieski, et.al., 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19).

IPD Hardware



IPD Example Setup

Power Source:
- IPD harvests energy (solar, radio wave) and stores it in a super capacitor (~1mF).
- When energy depletes, IPD **turns off**
- When enough energy accumulates, it **turns on**

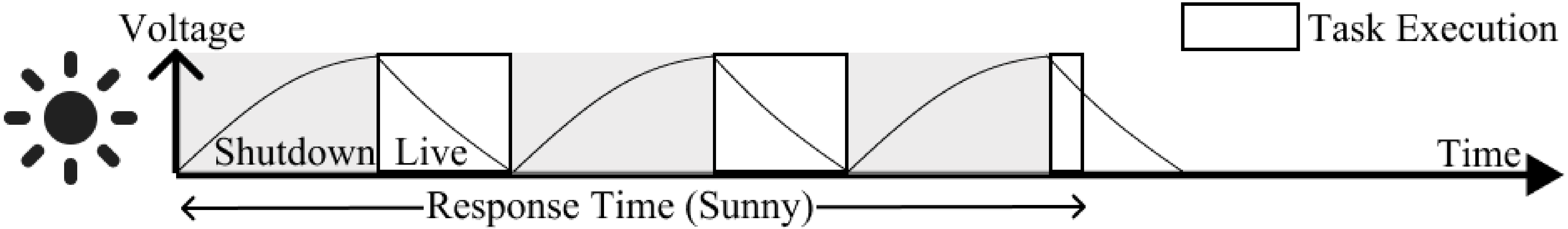
IPD has 2 types of memories*:
- **Volatile Memory (VM)**: fast, small, data lost when powered off;
- **Non-Volatile Memory (NVM)**: slow, large, data maintained when powered off.

*This holds for IPDs that run CPU and VM at higher clock rates than NVM

IPD Execution Model

IPD executes in **power cycles** due to constant power on/offs:

- **Live Time**: IPD turns on and begins program execution
- **Shutdown Time**: IPD shuts down and harvests energy
- **Environmental Effects**: affects energy harvesting rate, leading to different response times for the same task



Prior Work: Checkpointing Mechanisms

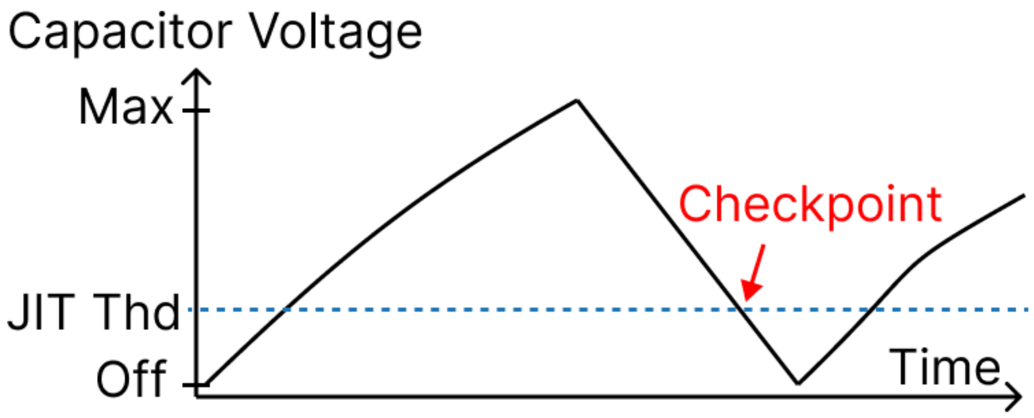
Purpose of Checkpointing: guarantee tasks forward progress during power loss

Existing Checkpointing Mechanisms:

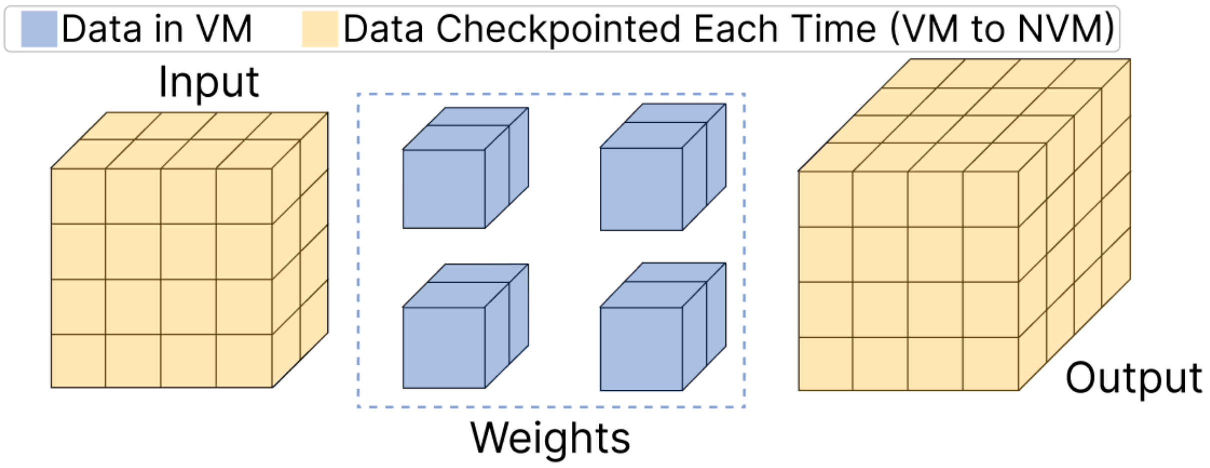
- Just-in-time Checkpointing **(JIT)**
- Static Checkpointing **(ST)**:
  - Static Checkpointing – Layer **(ST-L)**
  - Static Checkpointing – Filter **(ST-F)**
  - Static Checkpointing – Tiled **(ST-T)**

Prior Work: JIT Checkpointing

- **Just-in-Time Checkpointing (JIT)**: makes a checkpoint of the entire system's state to NVM when shutdown is imminent [1]



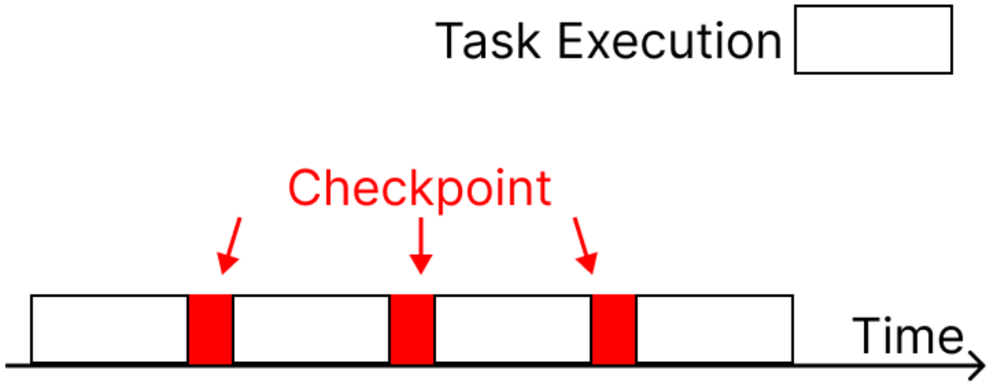
- **Pros**: Fastest because only one checkpointing per shutdown
- **Cons**: Largest in peak memory usage



[1] H. Jayakumar et al., "Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers," J. Emerg. Technol. Comput. Syst., vol. 12, no. 1, aug 2015.

Prior Work: Static Checkpointing

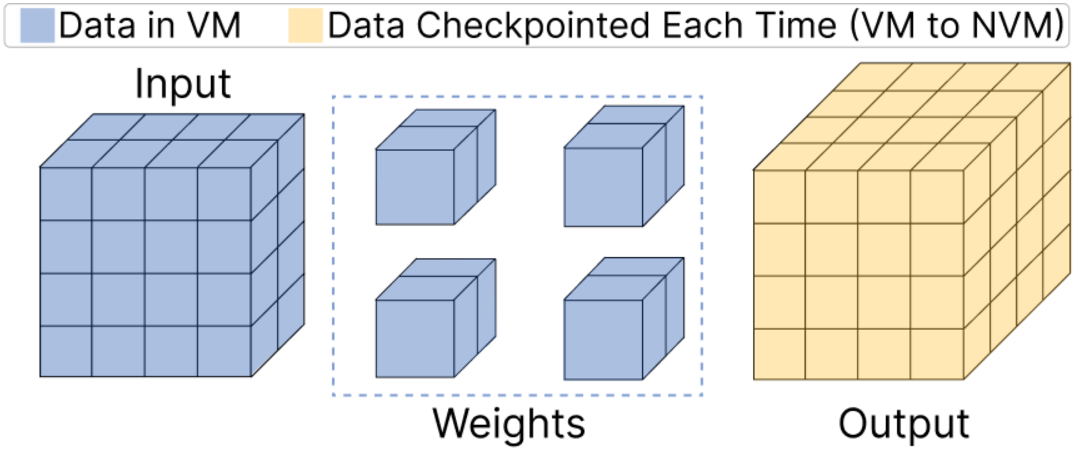
Static Checkpointing (ST): transforms the task into atomic blocks and stores the results to NVM at the end of each block. Re-exe the block upon reboot [1].



Blocks can be written in **different granularities**

Static Checkpointing - Layer (ST-L): Each layer of a DNN can be naturally modeled as an atomic block

- **Pros**: Fastest among ST because only one checkpointing per layer
- **Cons**: Largest in peak memory usage

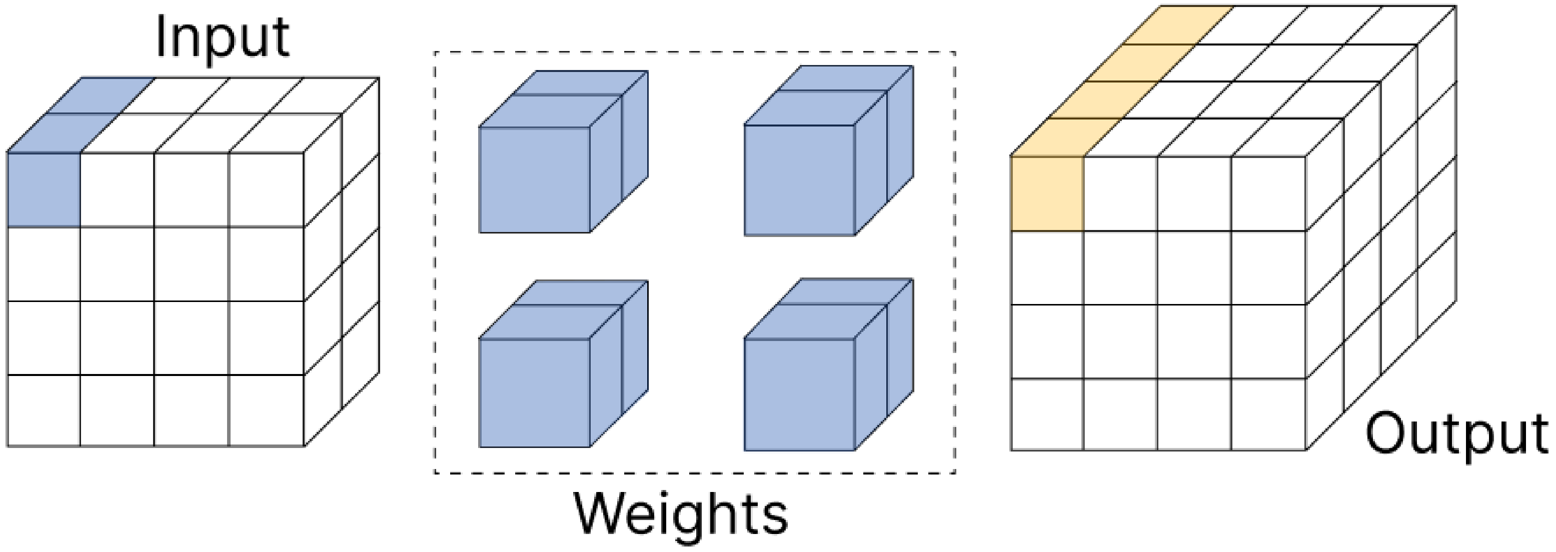


[1] Kiwan Maeng, et al., 2017. Alpaca: intermittent execution without checkpoints. Proc. ACM Program. Lang. 1, OOPSLA, Article, 30 pages.

Static Checkpointing - Filter (ST-F):

Re-writing each filter convolution into a separate atomic block [1,2]

- **Pros**: Only loading partial Input and Output to reduce peak memory
- **Cons**: Slower than ST-L

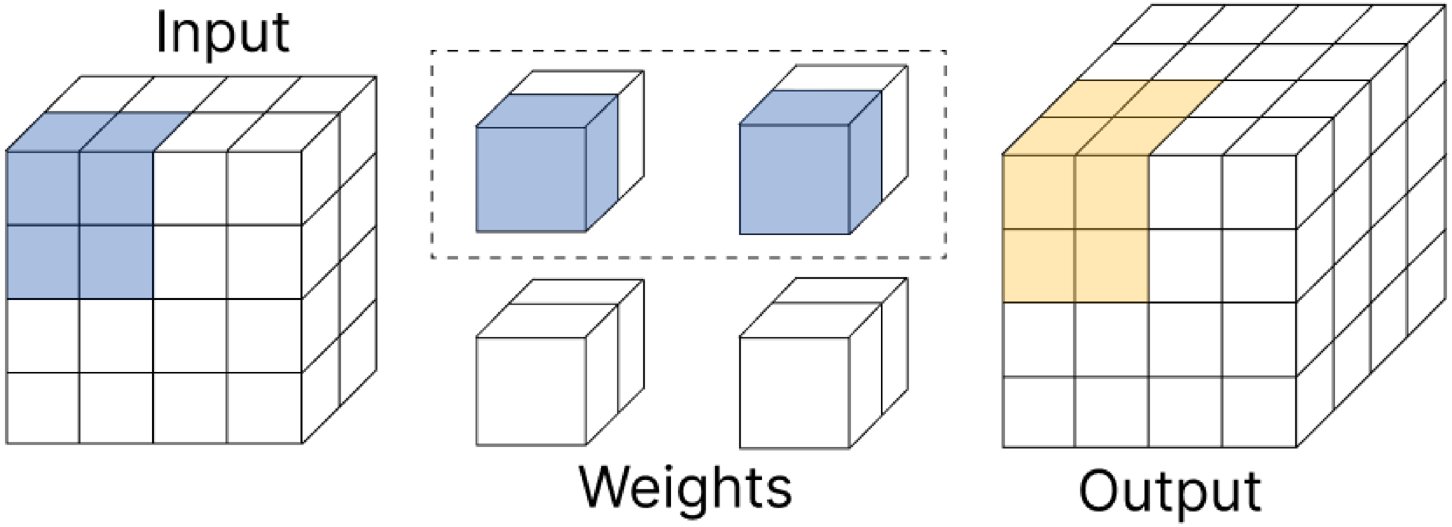


[1] Bashima Islam et al., 2020. Zygarde: Time-Sensitive On-Device Deep Inference and Adaptation on Intermittently-Powered Systems. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4, 3, Article 82 (September 2020), 29 pages.
[2] Graham Gobieski, et.al., 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19).

Static Checkpointing – Tiled (ST-T):

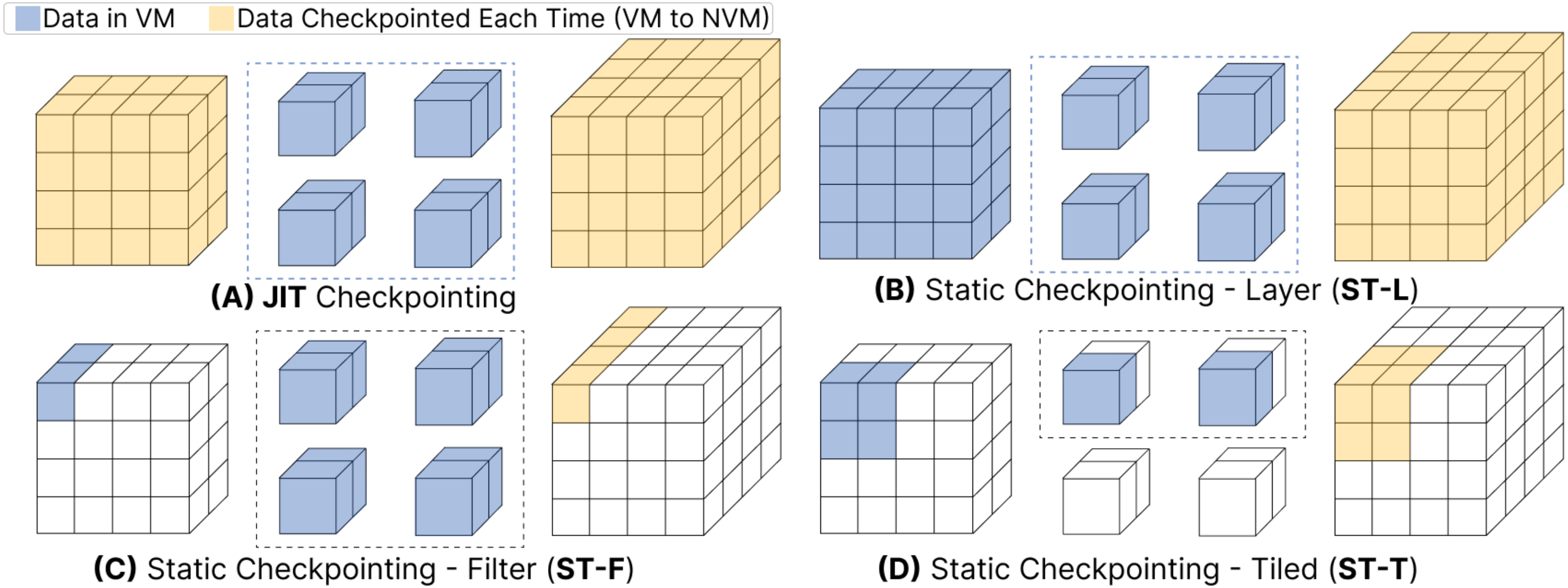
Further breakdown inference by re-organizing into a tiled structure [3]

- **Pros**: Further reduce peak memory by loading partial Weights
- **Cons**: May be slower than ST-F



[3] Hashan Roshantha Mendis et al., 2021. Intermittent-Aware Neural Architecture Search. ACM Trans. Embed. Comput. Syst. 20, 5s, Article 64

Prior Work: Summary



Use **only one CM** for the entire system is problematic: cannot balance peak memory and execution time!

Obs.1: Using a single checkpointing method across all layers is bad!

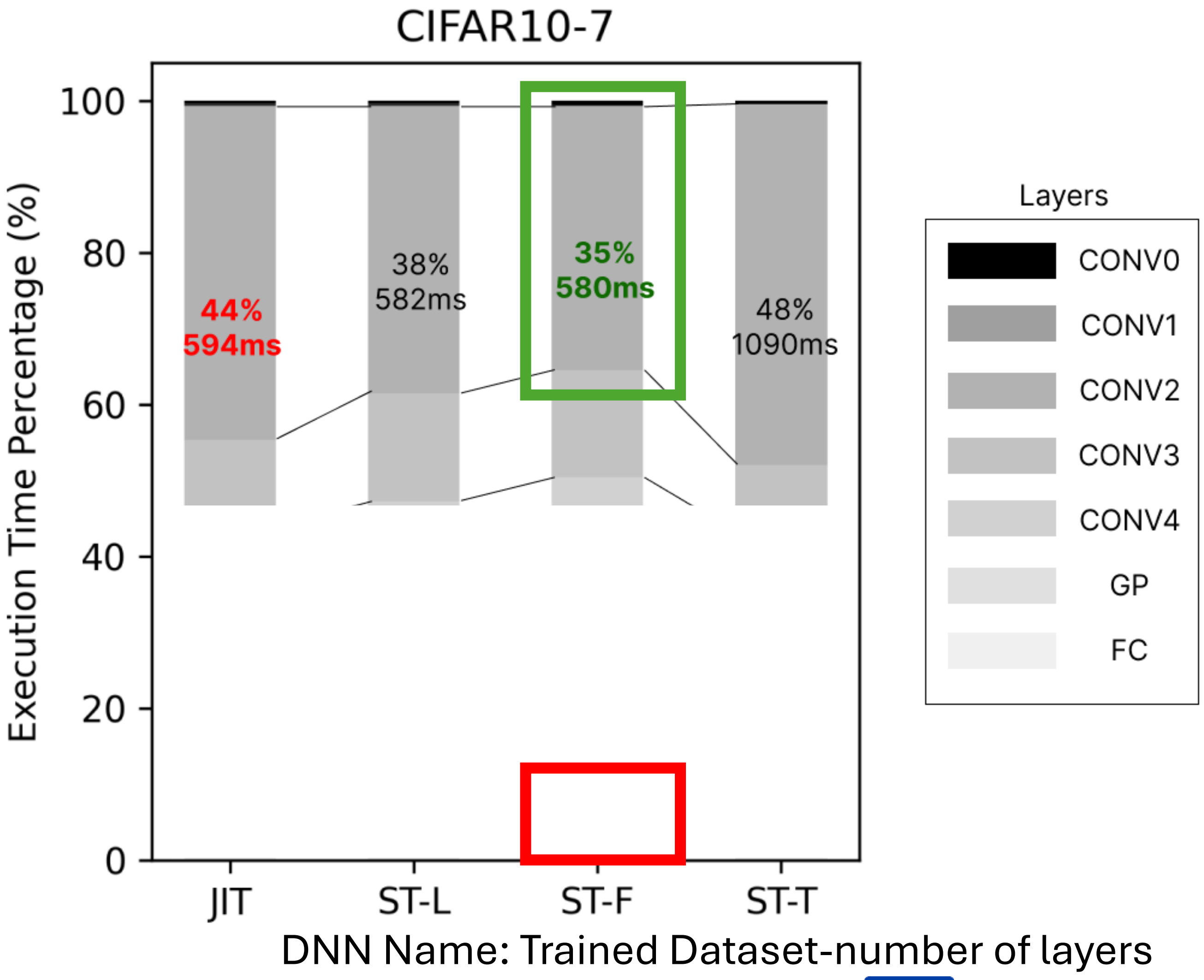
General tradeoff between JIT and ST:

- **JIT**: faster but uses larger peak memory
- **ST**: slower but uses smaller peak memory

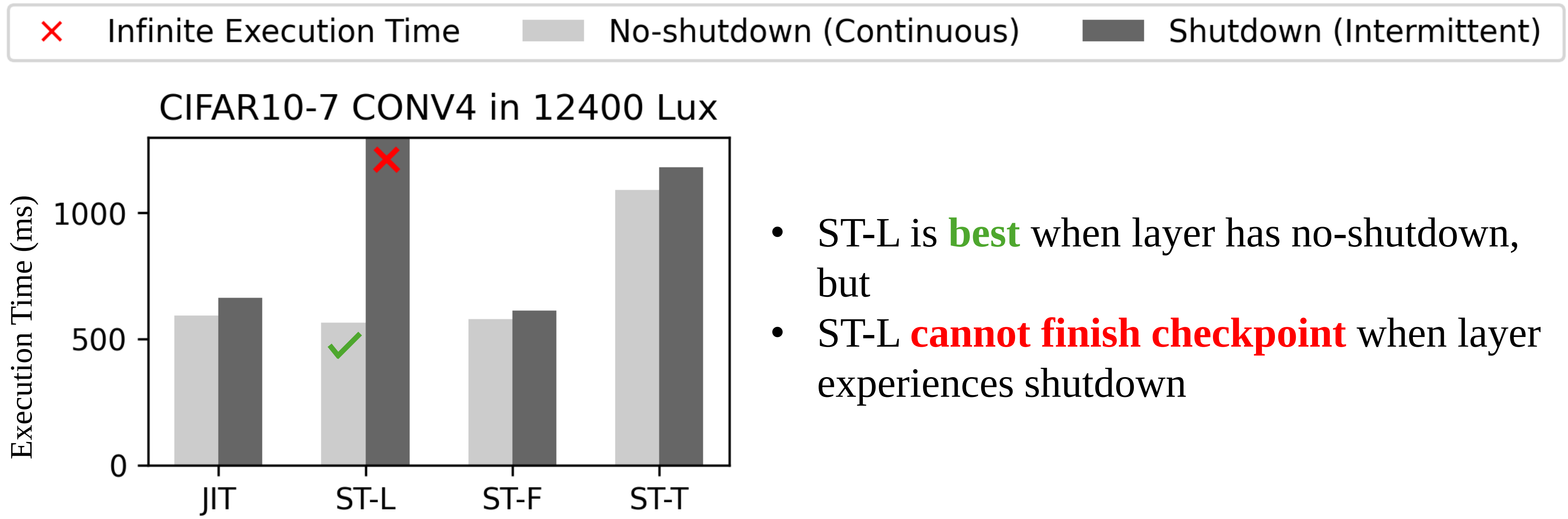
However,
Layer-wise performance is different!
Layer CONV4: **ST-F** is faster than JIT

But Again,
layer CONV0: **ST-F** is the worst!

A **layer-wise** adoption of **different CM** is needed!



Obs. 2: The optimal checkpointing choice changes when it experiences shutdown



- ST-L is **best** when layer has no-shutdown, but
- ST-L **cannot finish checkpoint** when layer experiences shutdown

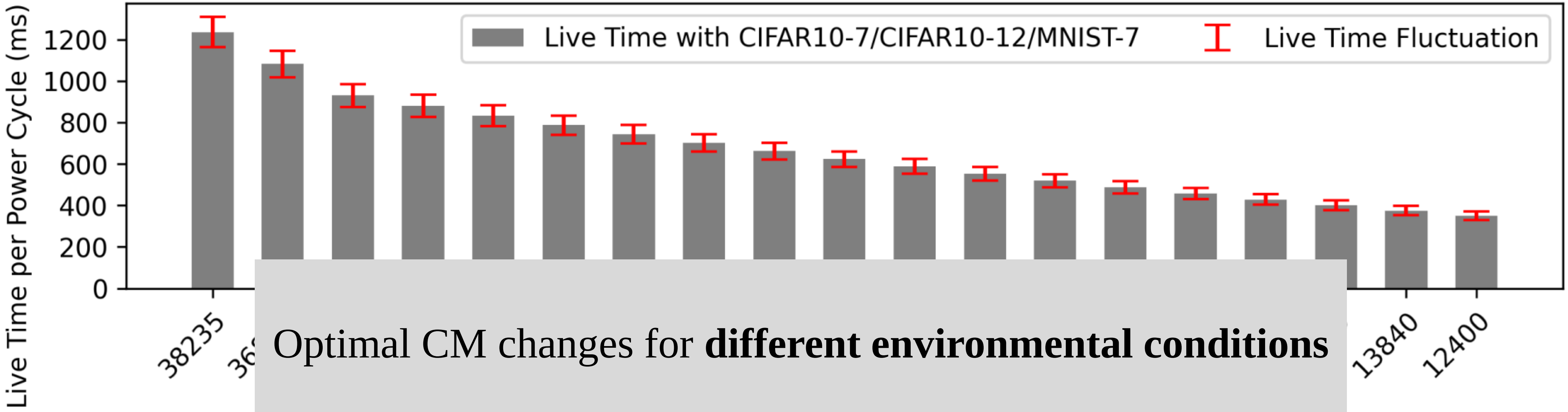
Obs. 3: environments change live time and shutdown layers drastically

Live time varies under different lighting

- Sunny: higher energy harvesting rate
- Cloudy: lower energy harvesting rate

CIFAR10-7 Shutdown Layers

CM	12400 Lux	38235 Lux
JIT	2,4	2,4



Optimal CM changes for **different environmental conditions**

MII Design Overview

Why applying an **offline + online** solution?

- Takes too long to run everything online – Huge runtime overhead!
- Problem is too challenging to address in a single phase

Offline Phase: search for layer-wise optimal CM solution under a given env. (**Obs.1 & 2**)

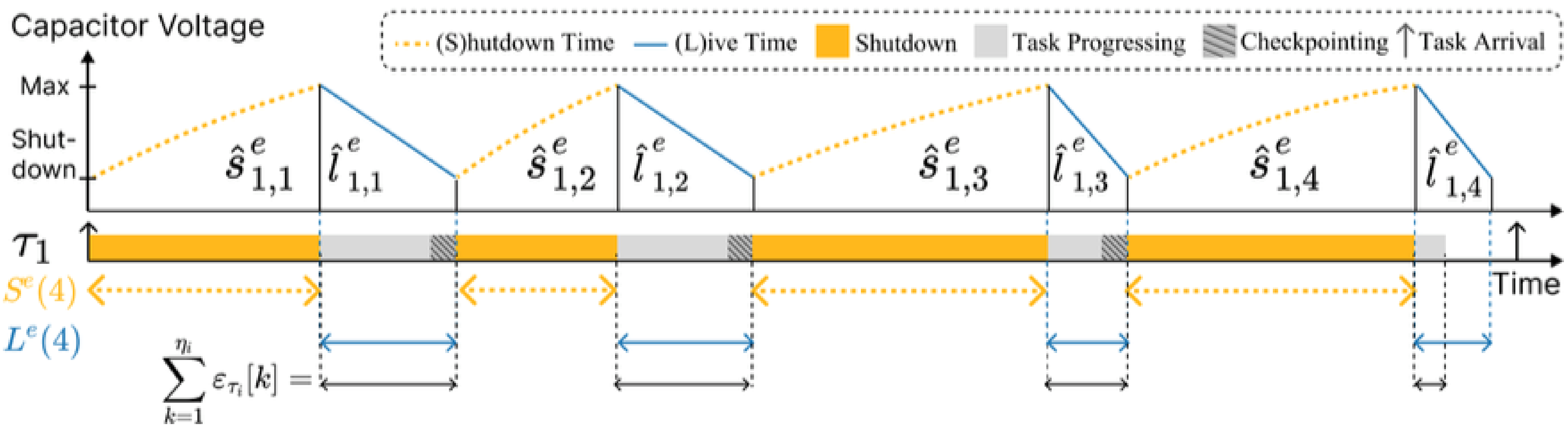
- Energy and Tasks Modeling
- Task-level and System-level (Two-level) CM Search

Online Phase: runtime adaption of CMs to cope with env. Changes (**Obs.3**)

- MII Scheduler: Power Cycle Harmonizing, Scheduling Policy, Proactive Shutdown
- CM Adaption

Offline Phase: Energy Pattern Modeling

Purpose: formulate the energy pattern to calculate the shutdown layers for each task



e: env condition

$S^e(n)$: **cumulative max** shutdown time over n consecutive power cycles

- Purpose: estimate at least how long to charge for the task to be finished

$L^e(n)$: **cumulative min** live time over n consecutive power cycles

- Purpose: estimate at least how much time we can use for task execution

Offline Phase: Two-Level CM search

Why two-level: cannot address all three constraints (Time, Mem., Shutdown) for all tasks in a single run

Task level: Minimize solo execution time of task τ_i

$\varepsilon_{\tau_i}[k][V] =$ τ_i's collective execution time from layers 1 to k, while not exceeding the memory constraint V.

$cm_{\tau_i}[k][V] =$ CMs achieving $\varepsilon_{\tau_i}[k][V]$.

System level: Minimize collective sum of solo execution times of m tasks

Collective sum of solo execution times:

$$E_\Gamma[i][V] = \min_{1 \leq j \leq V-1} E_\Gamma[i-1][j] + \varepsilon_{\tau_i}[\eta_i][V-j]$$

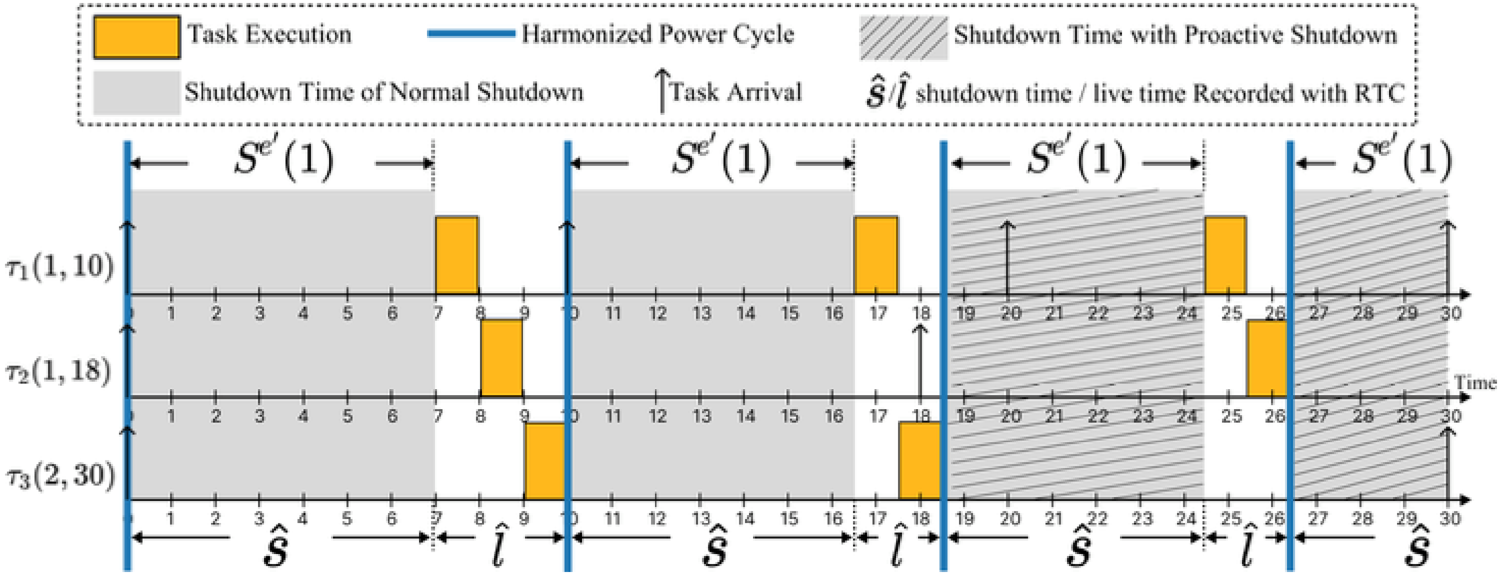
The initial conditions are:

$$E_\Gamma[0][1 \ldots V_{ipd}] = 0, E_\Gamma[1][1 \ldots V_{ipd}] = \varepsilon_{\tau_1}[\eta_1][1 \ldots V_{ipd}], \text{ and}$$
$$CM_\Gamma[0][1 \ldots V_{ipd}] = \emptyset, CM_\Gamma[1][1 \ldots V_{ipd}] = cm_{\tau_1}[\eta_1][1 \ldots V_{ipd}].$$

Online Phase: Tasks Scheduling

Power Cycle Harmonizing

- **Reason**: runtime power cycle deviated from the one used in offline search (Task Arrival)
- Delay task to next power cycle
- Ensures power cycle begins when IPD turns off



MII Scheduler

- Variant of the Least Slack Time (LST)
- Checks slack time at the boundary of each layer

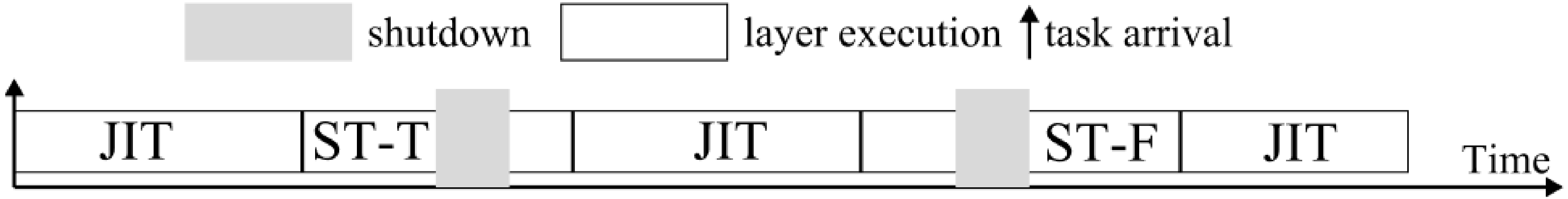
Proactive Shutdown

Shutdown when (1) or (2):

(1) Stored energy not enough to execute next block
(2) JIT threshold met

Online Phase: CM Adaption

Why need to adapt: Environment changes affect the shutdown layers



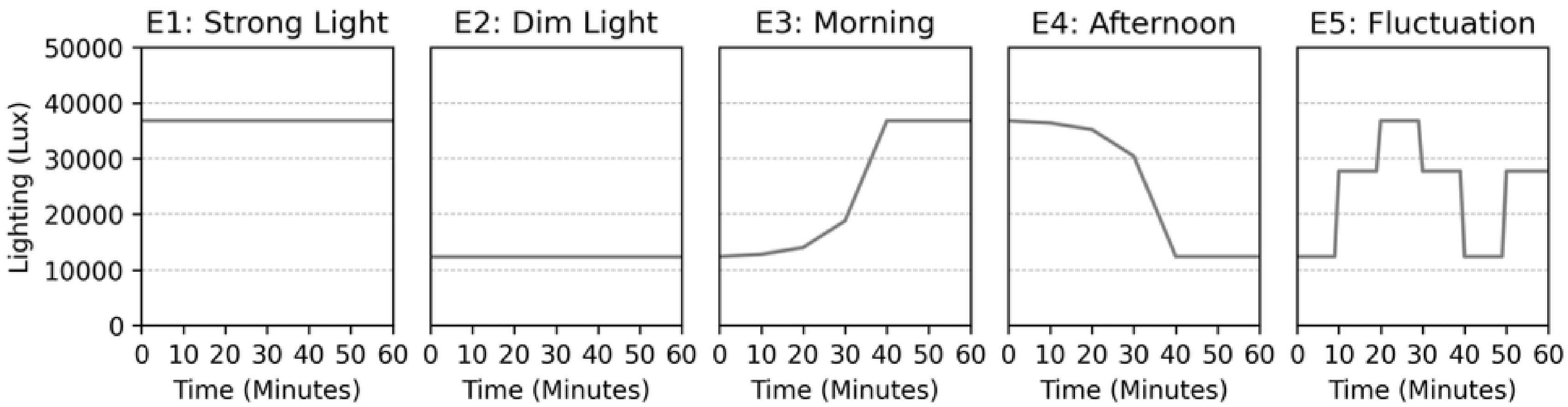
Offline Optimal

Online Environment Change: Red Layers sub-optimal and have to change CM!

CM Adaption: Redo CM search for tasks that are currently running

Evaluation Setup

- **Controlled Environments**: patterns cover 97% of lighting conditions during daytime



- **DNNs**: 8 DNN models trained from 6 datasets. Naming: CIFAR10-7layers == C7
- **DNN Tasksets (TC):** TC1 – C7, C12, M7, H5; TC2 – FC4, AutoEncoder, TC3 – MBV1, DSCNN

- **Hardware**: Apollo4 Blue Plus, 1.5W 8.2V Solar Panel, 1mF capacitor, 512KB VM

Baseline Configuration

QuickRecall [1]

- Standard JIT Checkpointing only system

Zygarde [2]

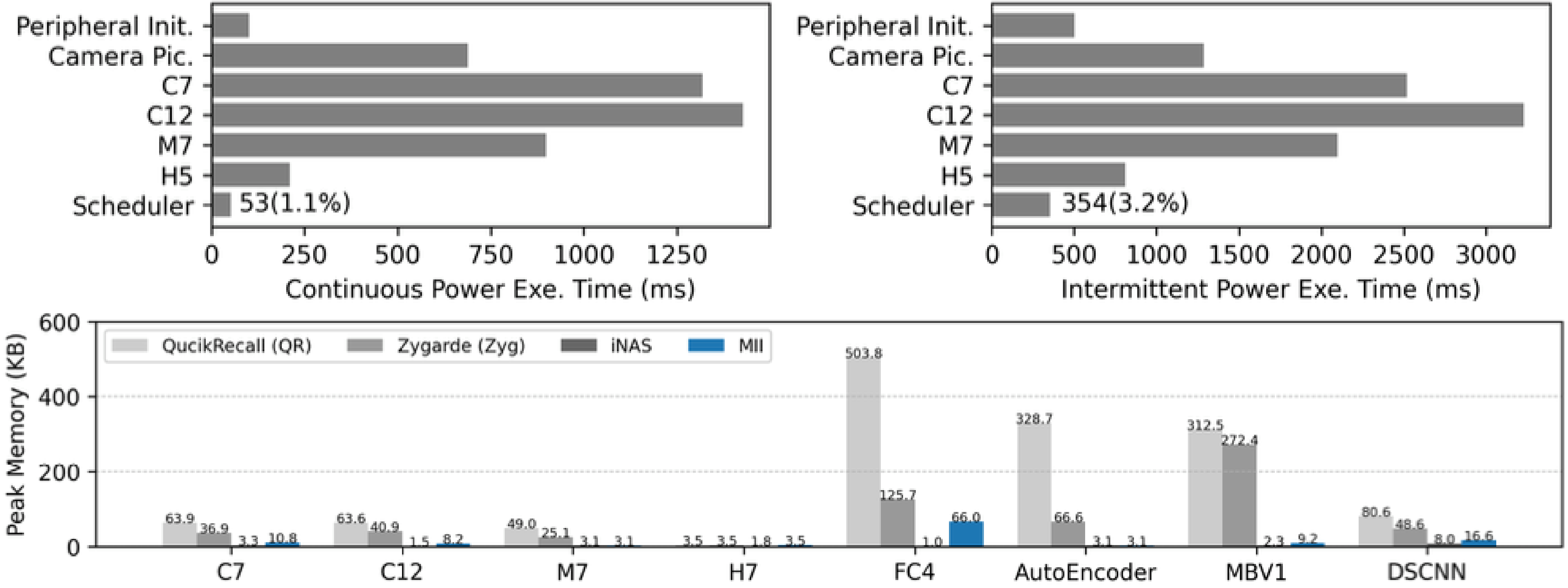
- Uses Static Checkpointing – Filter
- Uses early-exitable DNN models and early exit when less energy is available

iNAS [3]

- Uses Static Checkpointing – Tiled
- Design the tile sizes offline with a Neural Architecture Search algorithm

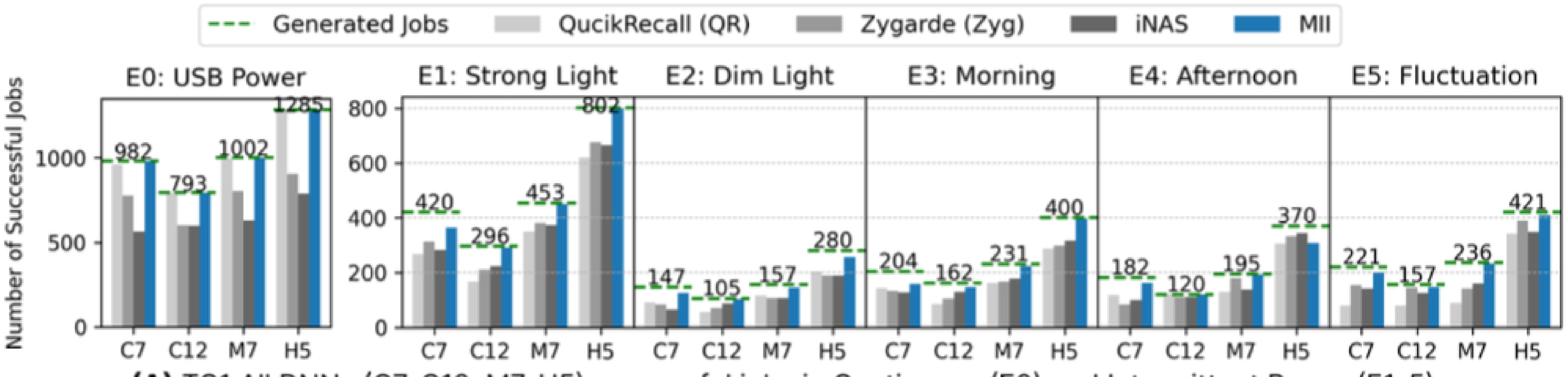
[1] H. Jayakumar et al., “Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers,” J. Emerg. Technol. Comput. Syst., vol. 12, no. 1, aug 2015.
[2] Bashima Islam et al., 2020. Zygarde: Time-Sensitive On-Device Deep Inference and Adaptation on Intermittently-Powered Systems. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4, 3, Article 82 (September 2020), 29 pages.
[3] Hashan Roshantha Mendis et al., 2021. Intermittent-Aware Neural Architecture Search. ACM Trans. Embed. Comput. Syst. 20, 5s, Article 64

Efficiency: runtime breakdown



MII successfully keeps peak memory below VM constraints while only uses **3.2%** of runtime overhead

Effectiveness: Successful Jobs Increases



(A) TC1 All DNNs (C7, C12, M7, H5) successful jobs in Continuous (E0) and Intermittent Power (E1-5)

- **Successful jobs**: jobs that complete execution before deadline
- **Why successful jobs**: measures the quality of service. Higher successful jobs means more valid results are generated
- On average **21%** successful jobs increase in stable energy conditions (E1-E2). **39%** increases under dynamic energy conditions (E3-E5)

Thank you

https://izenderi.github.io/

RTEN Real-time Embedded and Networked Systems Laboratory

UC RIVERSIDE