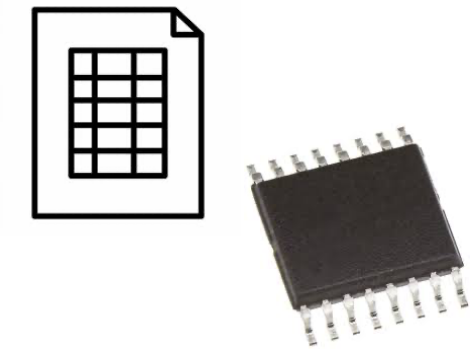
Quantifying Context Switch Overhead of Artificial Intelligence Workloads on the Cloud and Edges

Kun Suo, Yong Shi, Chih-Cheng Hung, Patrick Bobbie

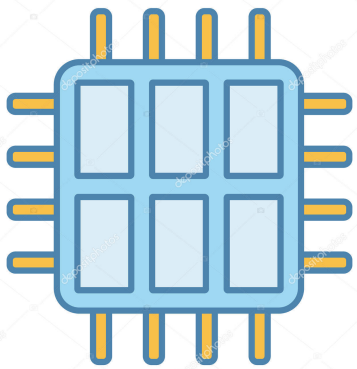
Kennesaw State University

What Does the Context Switch Overhead Include?

- **Direct overhead of context switch**
  - Global page table directory switching
  - Kernel stack switching
  - Hardware context switching
    - Loading data into the registers
  - Translation lookaside buffer (TLB) refresh

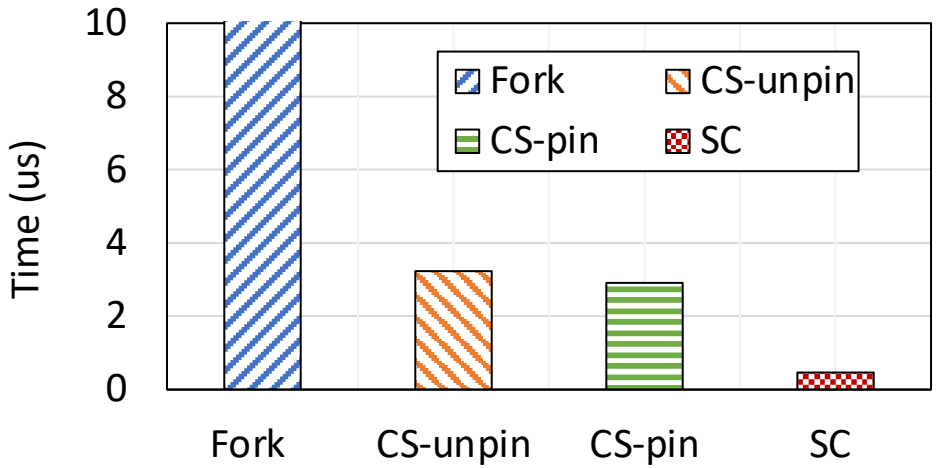


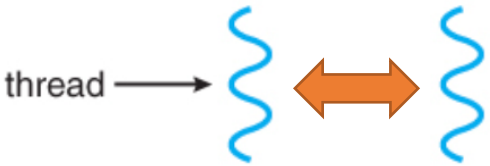
- **Indirect overhead of context switch**
  - Shared data between multi-core caches



How Much is the Overhead of Context Switch?

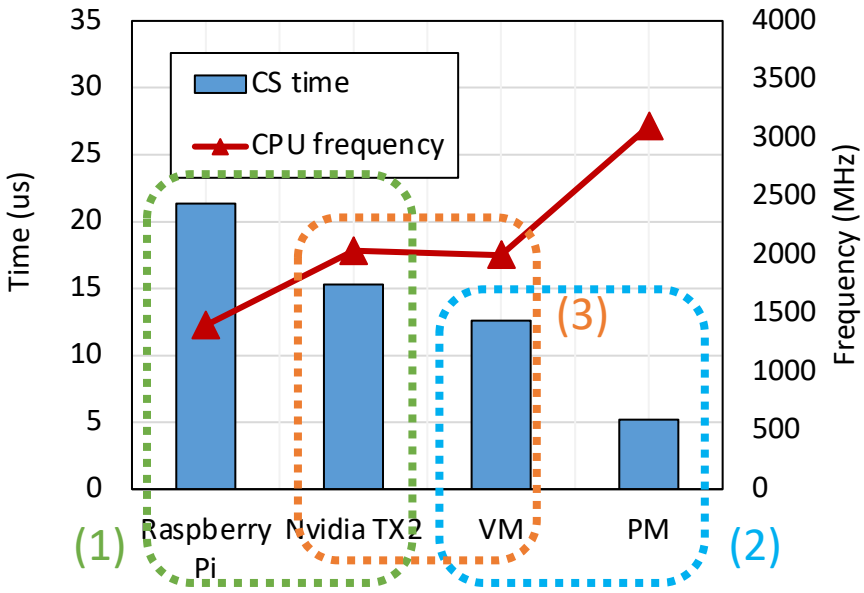
- **Measuring the CPU time cost of one context switch using pipe**
  - Two processes in Linux and passed a token between them
  - Manually set the processes with maximum priority under real-time scheduling policy SCHED_FIFO
  - Compared with system calls, folk
  - ***Conclusion: Fork > CS-unpin > CS-pin > System Call***

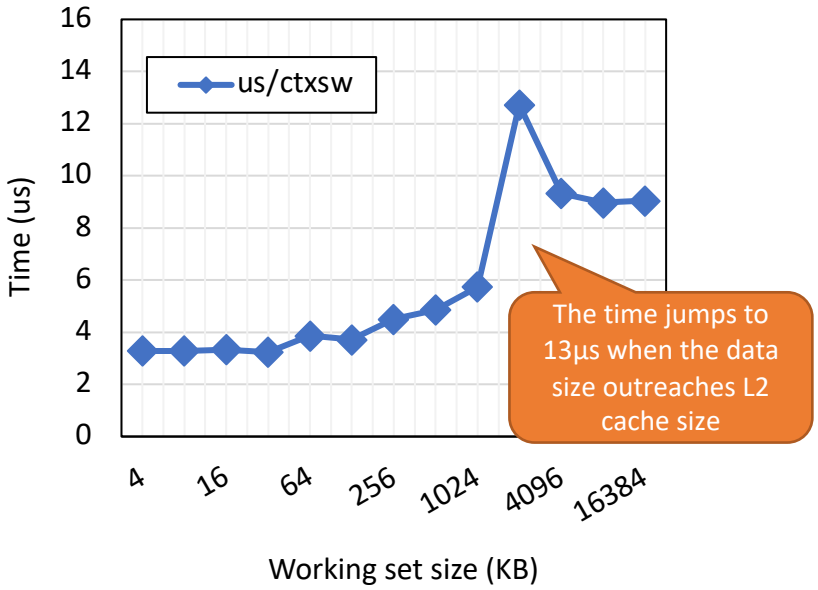




How Much is the Overhead of Context Switch?

- **Evaluate the overhead of context switch using microbenchmark contextswitch**
  - We adopted edge or IoT devices such as Raspberry Pi 4B and Nvidia TX2
    1) With faster CPU speed, the delay introduced by context switch could be greatly reduced
    2) Context switch overhead on VMs is 2.4× of that on physical machines even though the CPU frequency of physical nodes is just 50% faster
    3) The context switch overhead of VM is much less than edge devices due to its large L1 and L2 cache

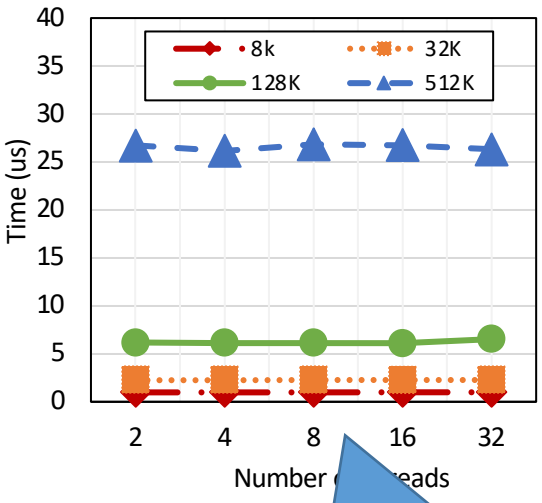




How Much is the Overhead of Context Switch?

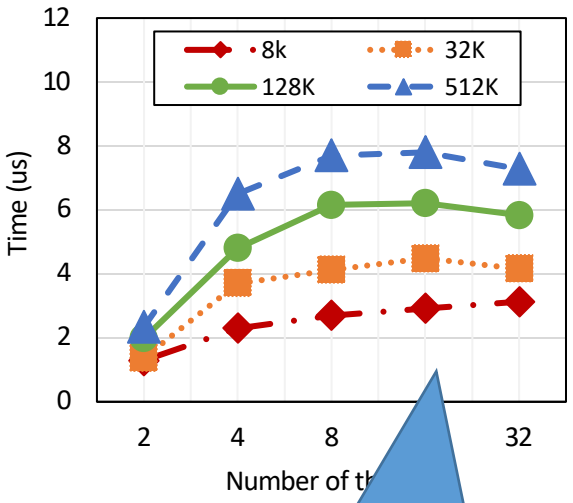
- **Measure application overhead using lat_ctx in lmbench**
  - Create parent-child processes sending and receiving from pipe
  - Precisely control the intensity of process switching and the number of processes of switching through parameters

(a) Non-Context switch overhead



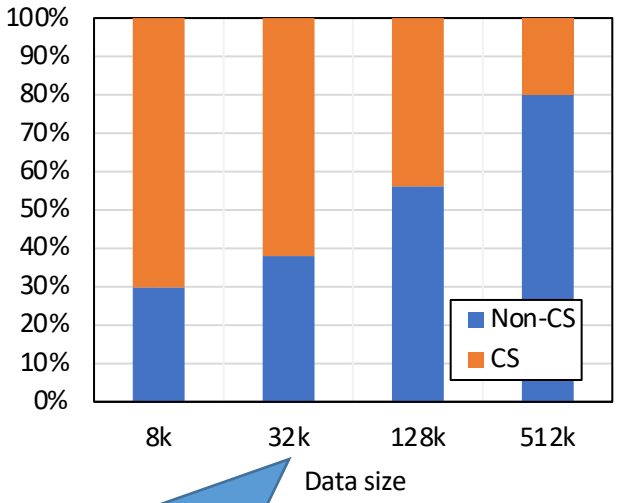
noncontext switching overhead is consistent with different number of processes or threads but proportional to the amount of data

(b) Context switch overhead



context switching overhead is grows significantly with different number of processes and proportional to the amount of data

(c) Overhead percentage with 4 threads



non-context switch overhead is dominant when the amount of data among processes is large while context switch takes the majority of overhead when the data size is small

Existing Efforts

- **Context Switch Cost Analysis**
  - Analyzing factors that affect context switch cost and application performance [PACT-08]
  - Industry: providing dedicated resources, preemptive scheduling, prefetching the hot data, etc.

  ✗ CS overhead of AI workloads

- **Optimizing Context Switch Overhead**
  - Fixed priority preemptive scheduling [RTAS-18]
  - Context-switch aware TLB algorithm [MICRO-17]

  ✗ Application analysis

- **Infrastructure Efficiency on Edge AIs**
  - Optimize underlying infrastructure performance [HPDC-19, HotCloud-19]

  ✗ Not focus on context switch

Experimental Settings & Methodology

- **Hardware**
  - Raspberry Pi 4B with quad-core CPU and 4GB of RAM
  - Nvidia Jetson TX2, a quad-core A57 processor, a dual-core Denver processor, a 56-core GPU, and 8GB of RAM
  - VM (quad-core vCPU and 16GB of memory) and PM (24-core Intel Xeon E5-2670 v3 CPU and 504 GB memory)

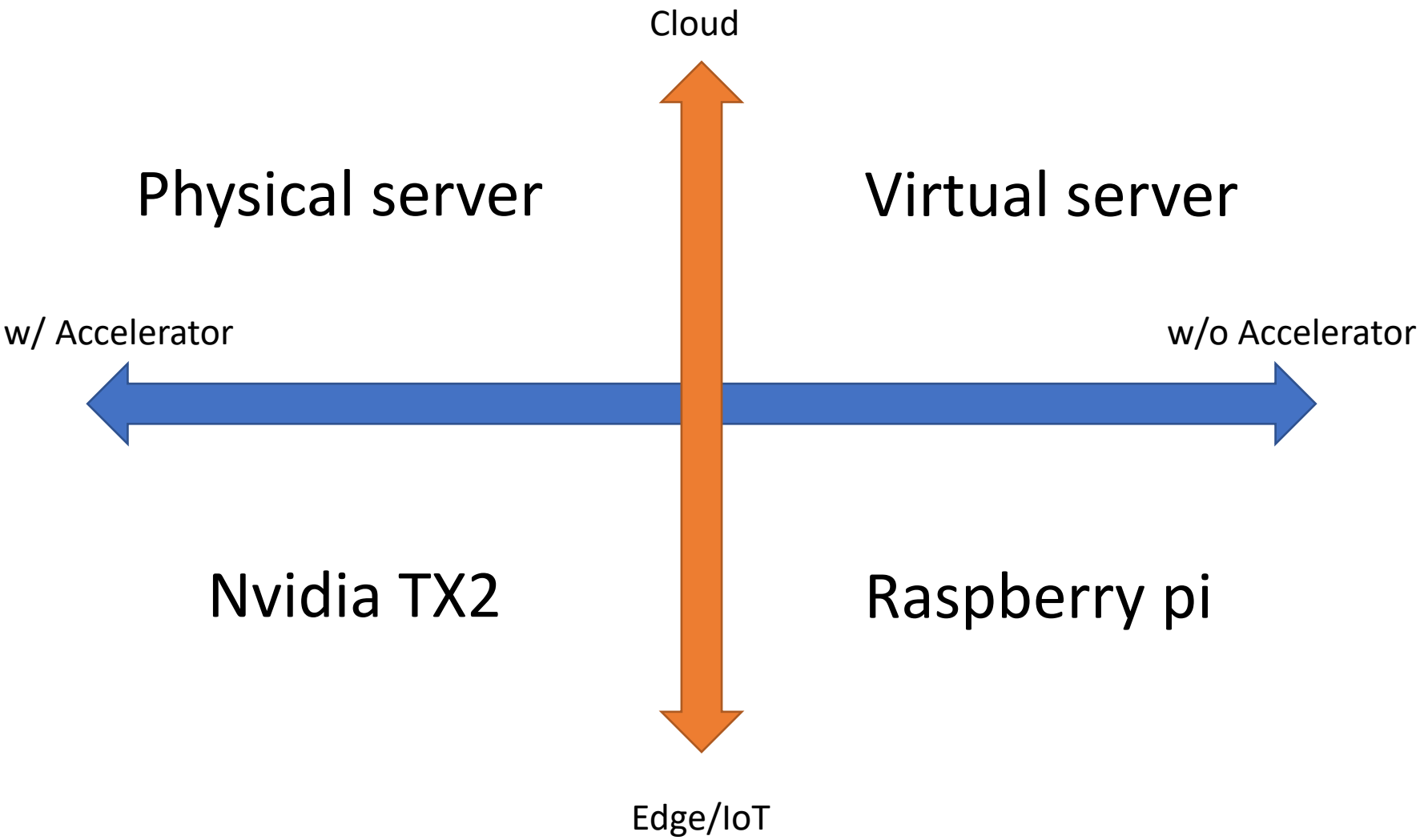
- **Software**
  - TensorFlow 1.14
  - AI Benchmark: 42 tests and 16 sections

- **Methodology**
  - Linux nmon capture the data
  - Convert into HTML file using nmonchart

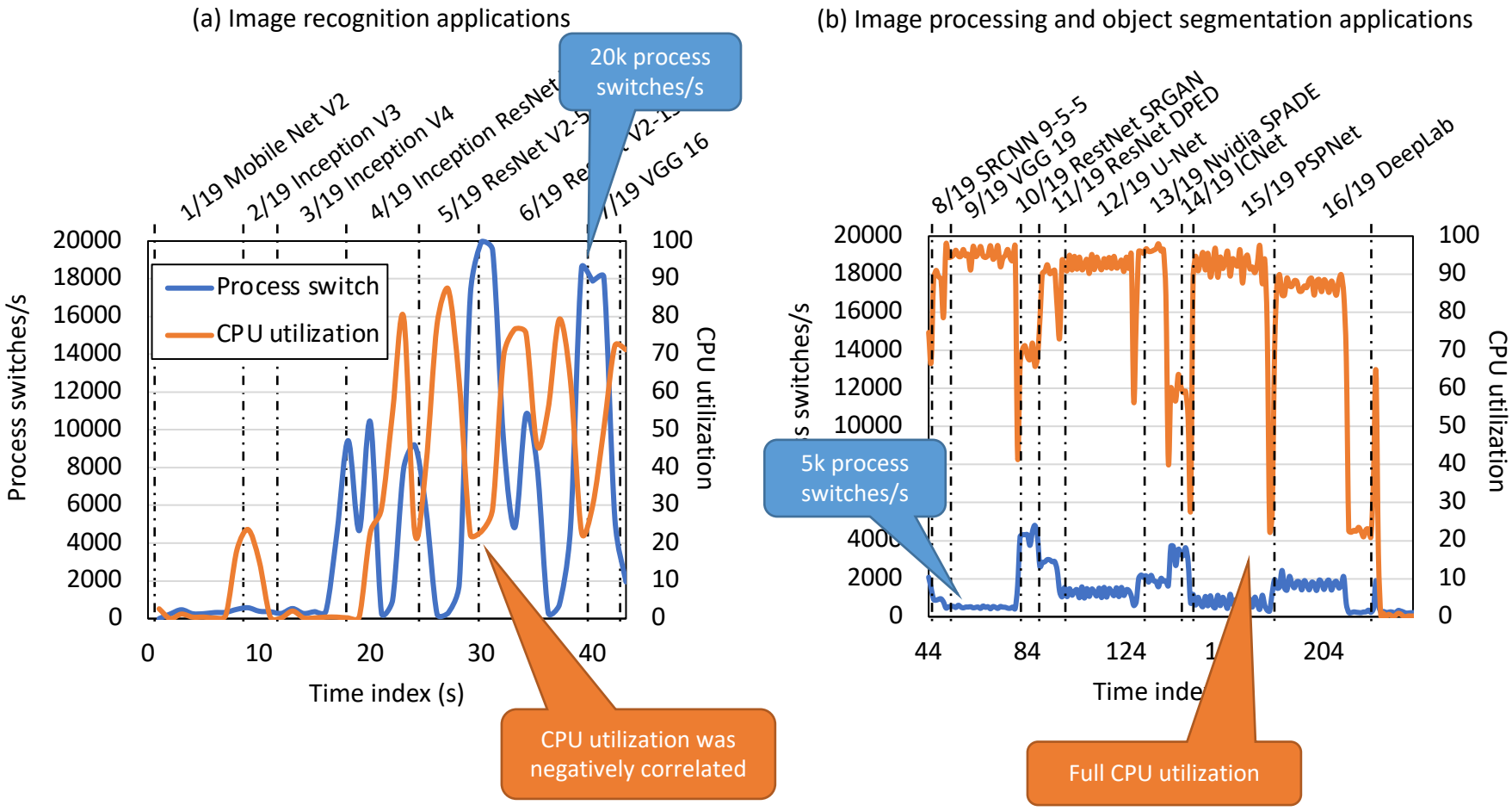
Specifications of AI workloads

ID	Task	Neural Network / Model
1	Object Recognition	MobileNet v2
2	Classification	Inception v3
3	Classification	Inception v4
4	Facial Recognition	Inception-ResNet v2
5	Classification	ResNet-50 v2
6	Classification	ResNet-152 v2
7	Classification	VGG-16
8	Super-Resolution	VGG-19
9	Super-Resolution	ResNet-SRGAN
10	Image Deblurring	SRCNN 9-5-5
11	Image Enhancement	ResNet-DPED
12	Bokeh Simulation	U-Net
13	Semantic Image Synthesis	Nvidia-SPADE
14	Image Segmentation	ICNet
15	Image Segmentation	PSPNet
16	Image Segmentation	DeepLab v1

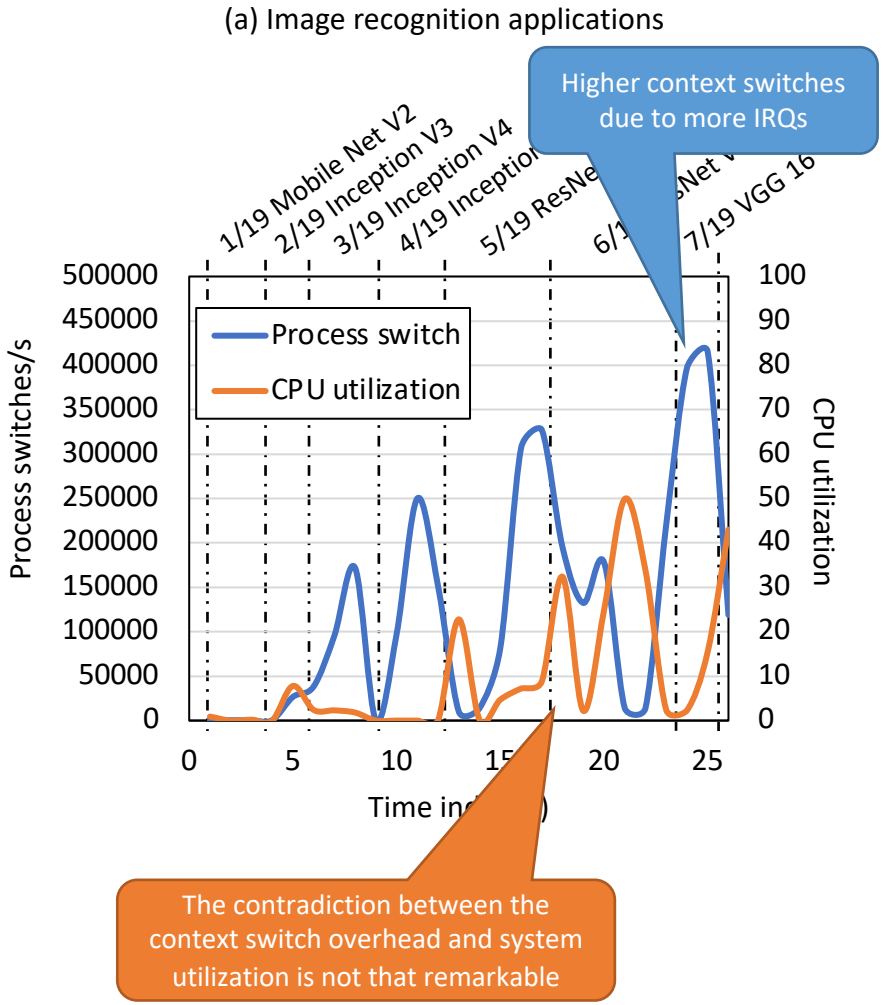
Context Switches of AI workloads Evaluation

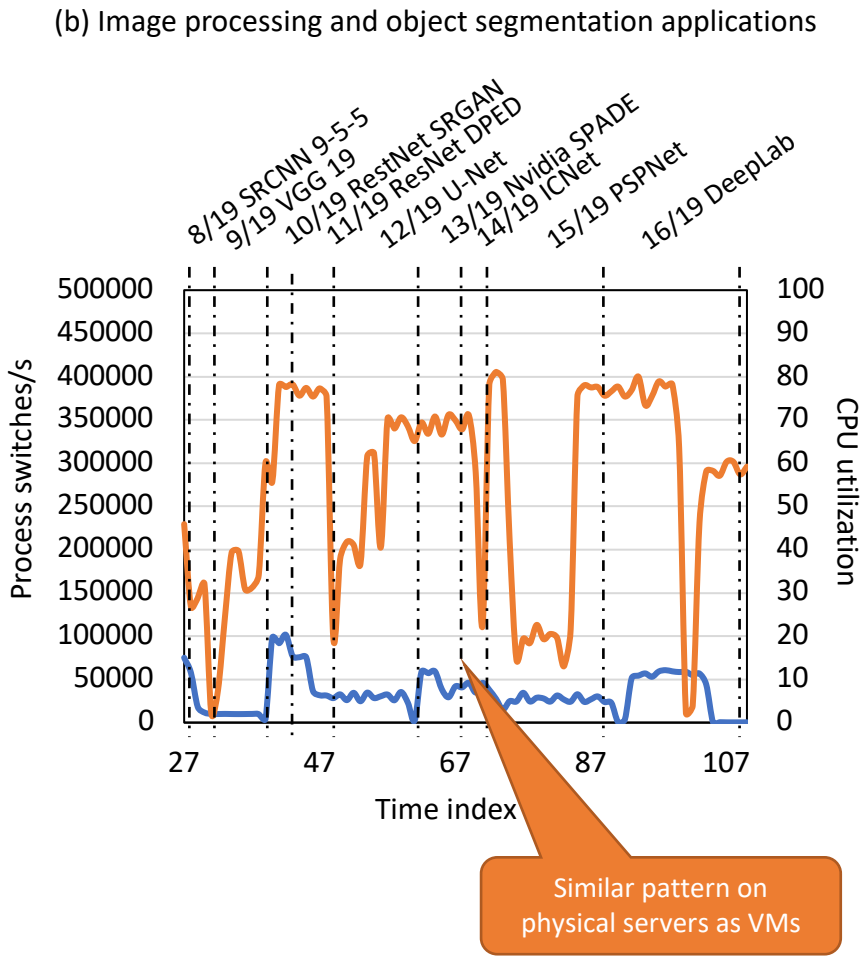


AI workloads on cloud VMs

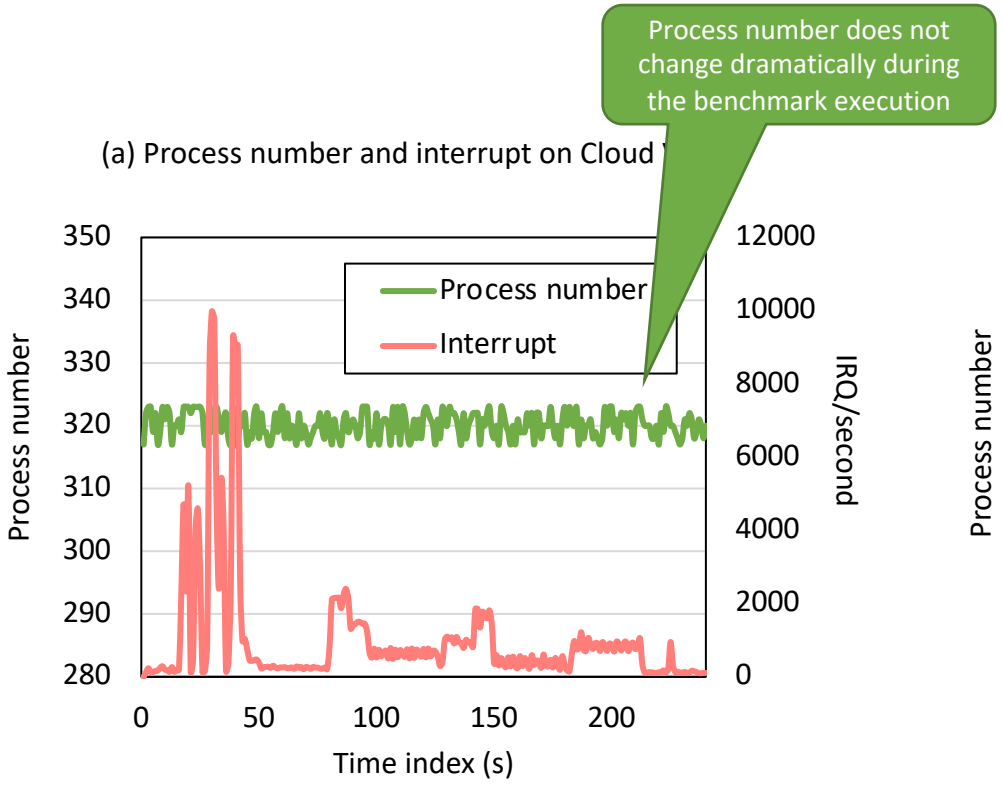


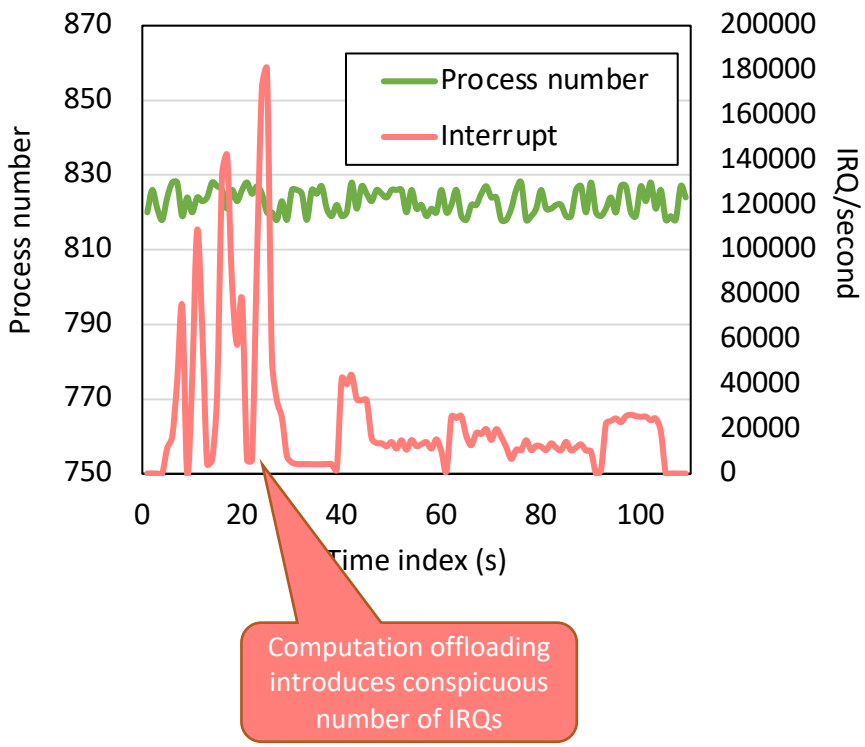
AI workloads on physical servers





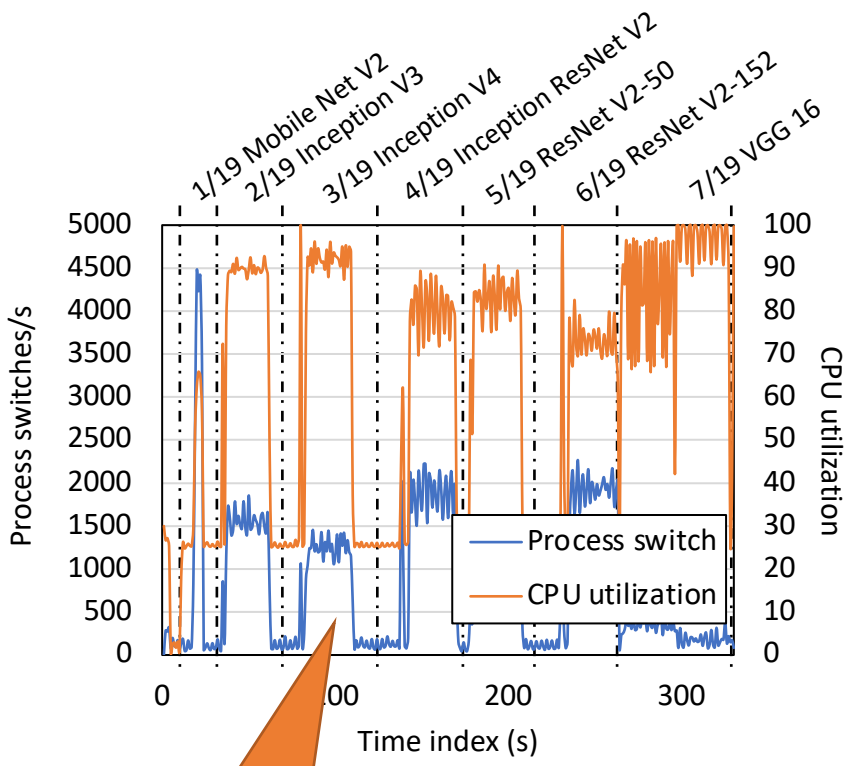
Process number and interrupt





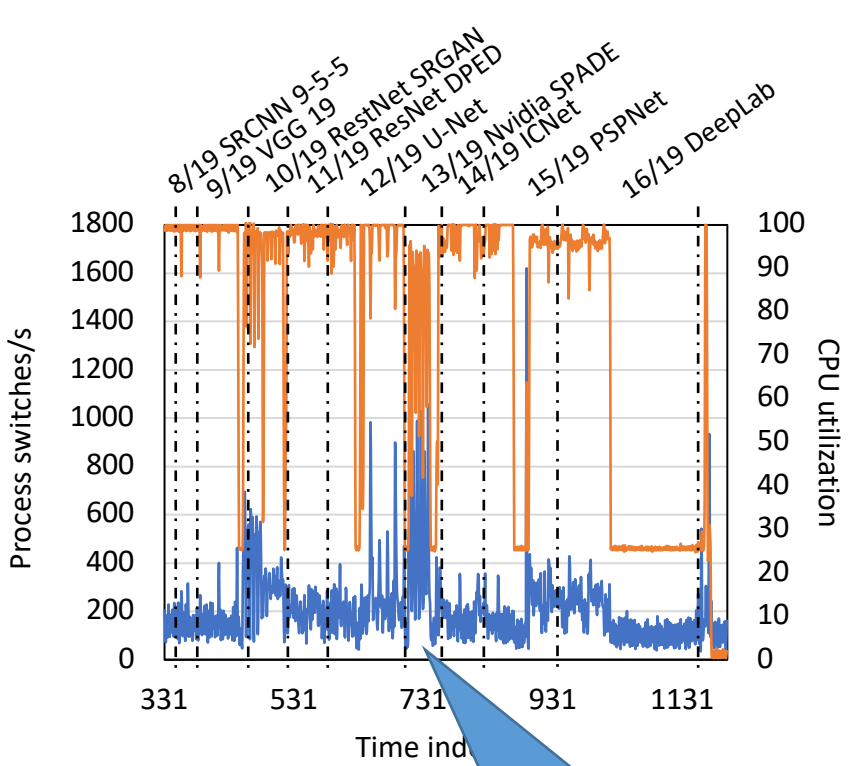
AI workloads on Raspberry Pi

(a) Image recognition applications



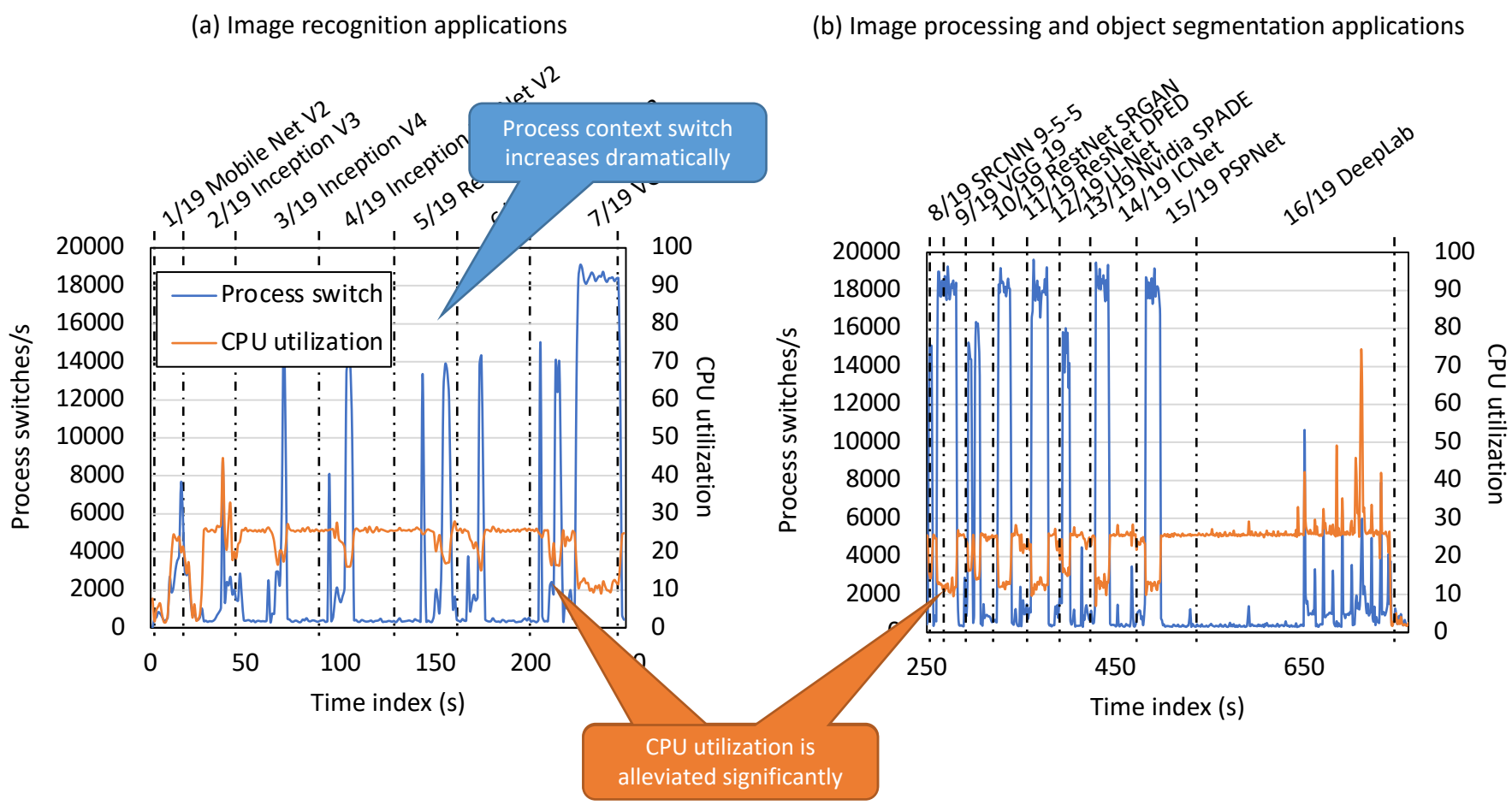
CPU utilization increases to 100% when number of context switches goes up

(b) Image processing and object segmentation applications



Context switching goes up, we observe the reduction of CPU utilization and performance

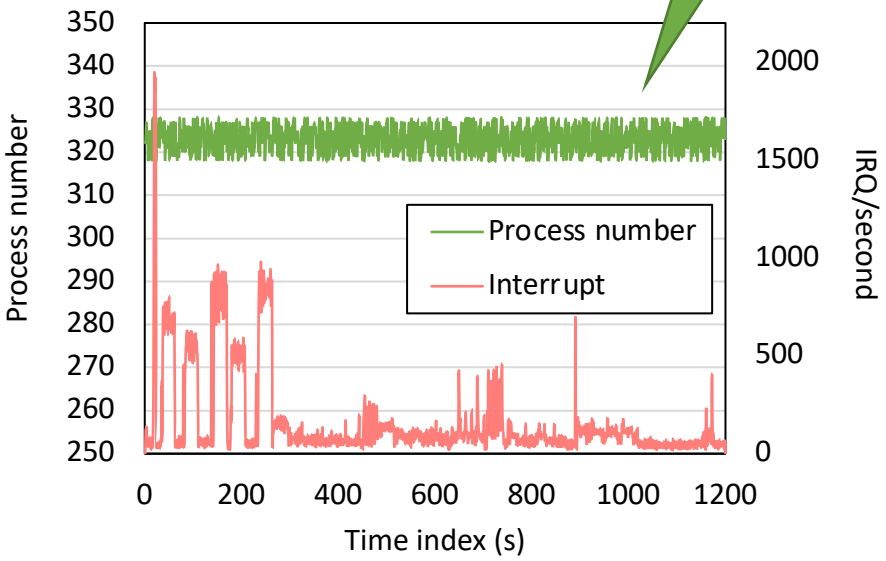
AI workloads on Nvidia TX2 (w/ GPU)



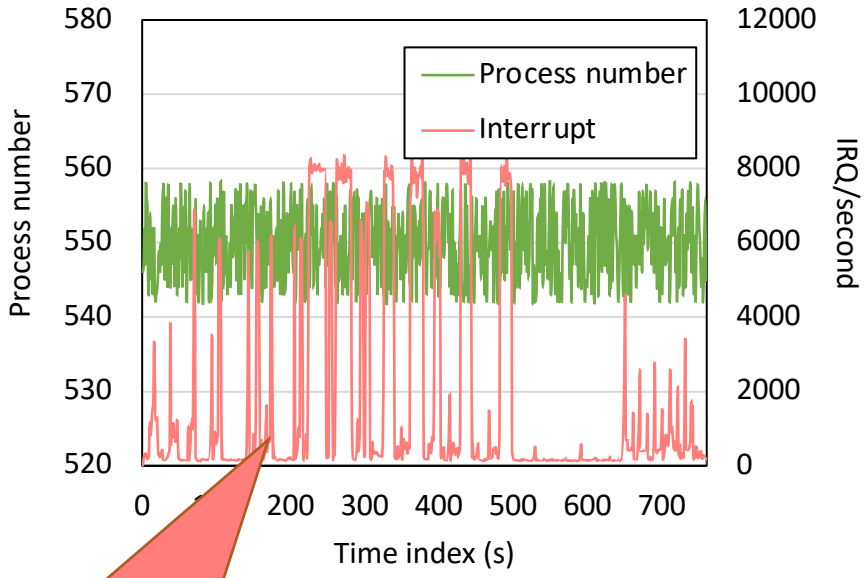
Process number and interrupt

Process number does not change dramatically during the benchmark execution

(a) Process number and interrupt on Raspber



(b) Process number and interrupt on Nvidia Jetson TX2



IRQs involving dramatic interactions between the CPUs and the accelerators

Conclusion

- We systematically studied the overhead of context switches under different system settings
- We identified such the overhead are highly related with hardware platforms, application behaviors, data set sizes and number of process threads
- We investigate different AI workloads on data center machines and IoT devices with traditional or heterogeneous architectures

Thoughts

1. Is it feasible to eliminate the context switches overhead by adjusting the factors (e.g., application behaviors, data set size, etc.)?

2. How can the kernel perform a better isolation between different computation devices especially with heterogeneous accelerators to mitigate the context switch overhead?

3. Can the system apply different policies flexibly, such as interrupt coalescing, multicore scheduling, etc., to mitigate context switches overhead of AI workloads with various patterns?

Thank you !

Questions?