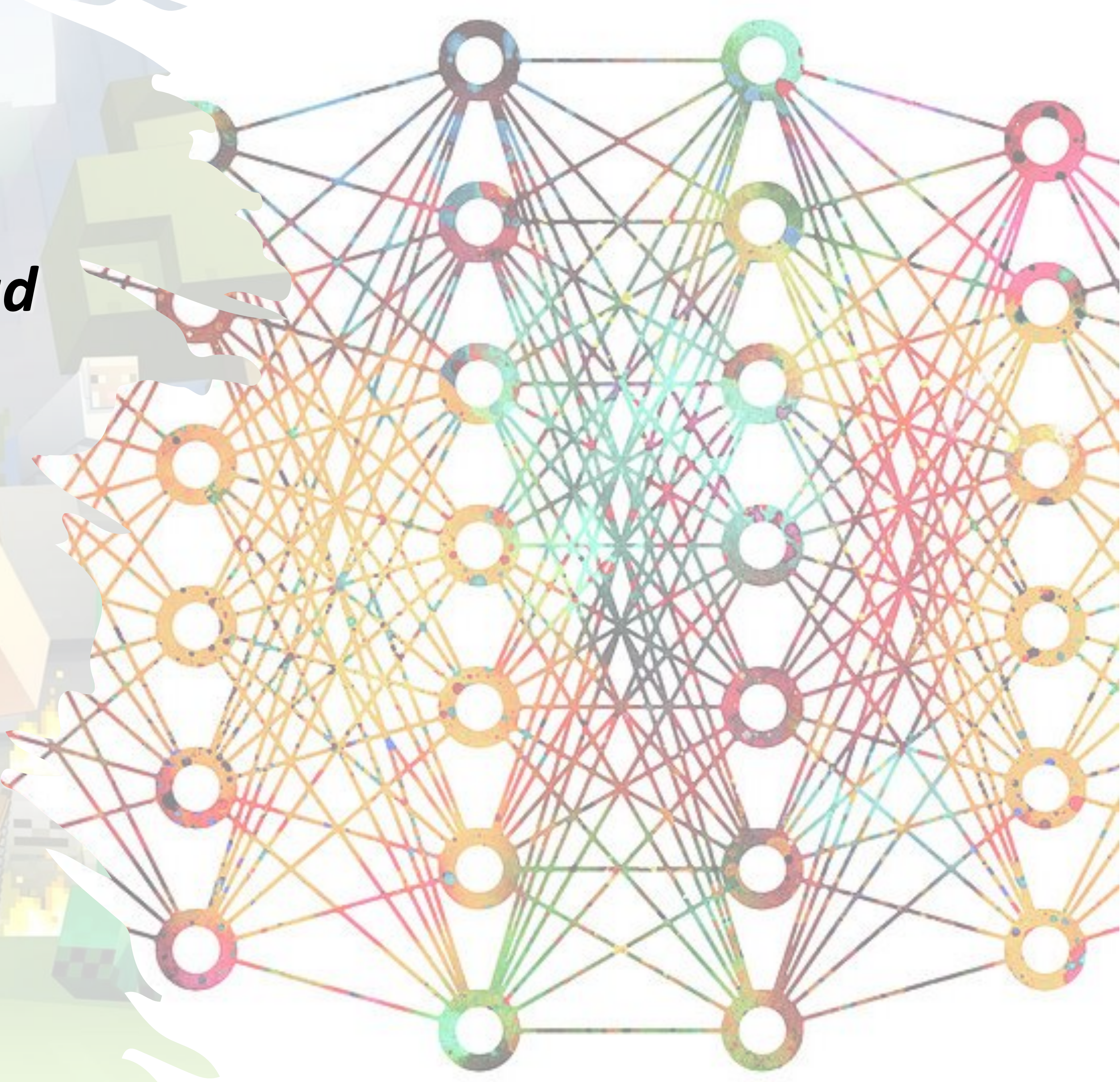
PilotFish: Harvesting Free Cycles of Cloud Gaming with Deep Learning Training

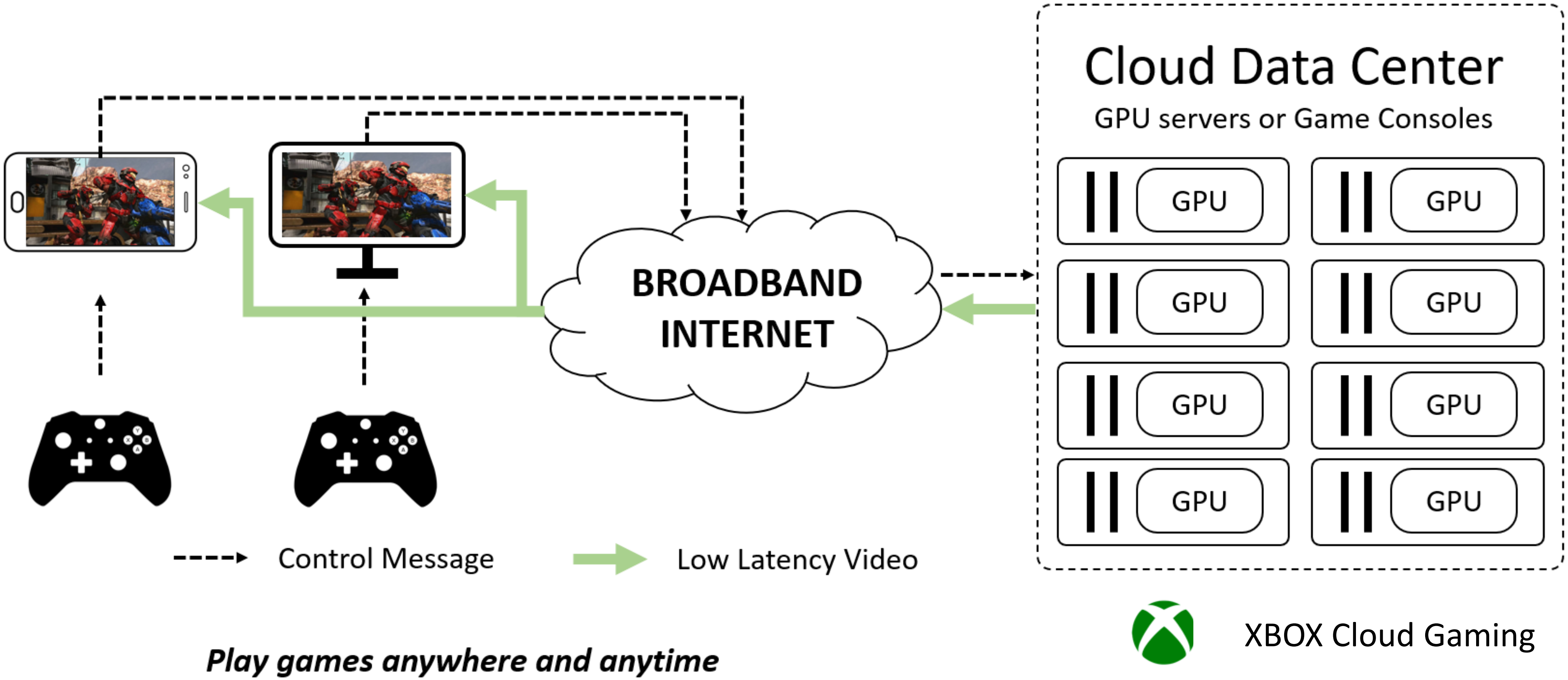
*Wei Zhang, Binghao Chen, **Zhenhua Han**, Quan Chen, Peng Cheng, Fan Yang, Ran Shu, Yuqing Yang, Minyi Guo*

Microsoft Research

Shanghai Jiao Tong University

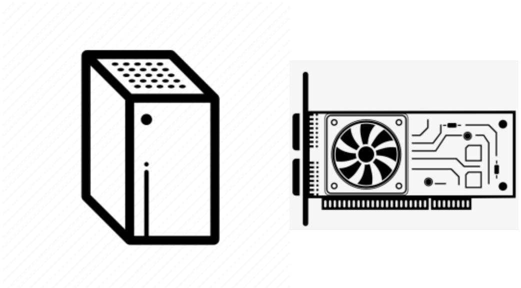


Cloud Gaming



Play games anywhere and anytime

Low GPU Util. of Cloud Gaming



Modern GPUs
Run games at 4K 60 FPS (frames/s)

Video Streaming over Internet

Network limitation

- *40Mbps for 4K @ 60 FPS*
- *25Mbps for 1080p @ 60 FPS*
- *40 ms latency*

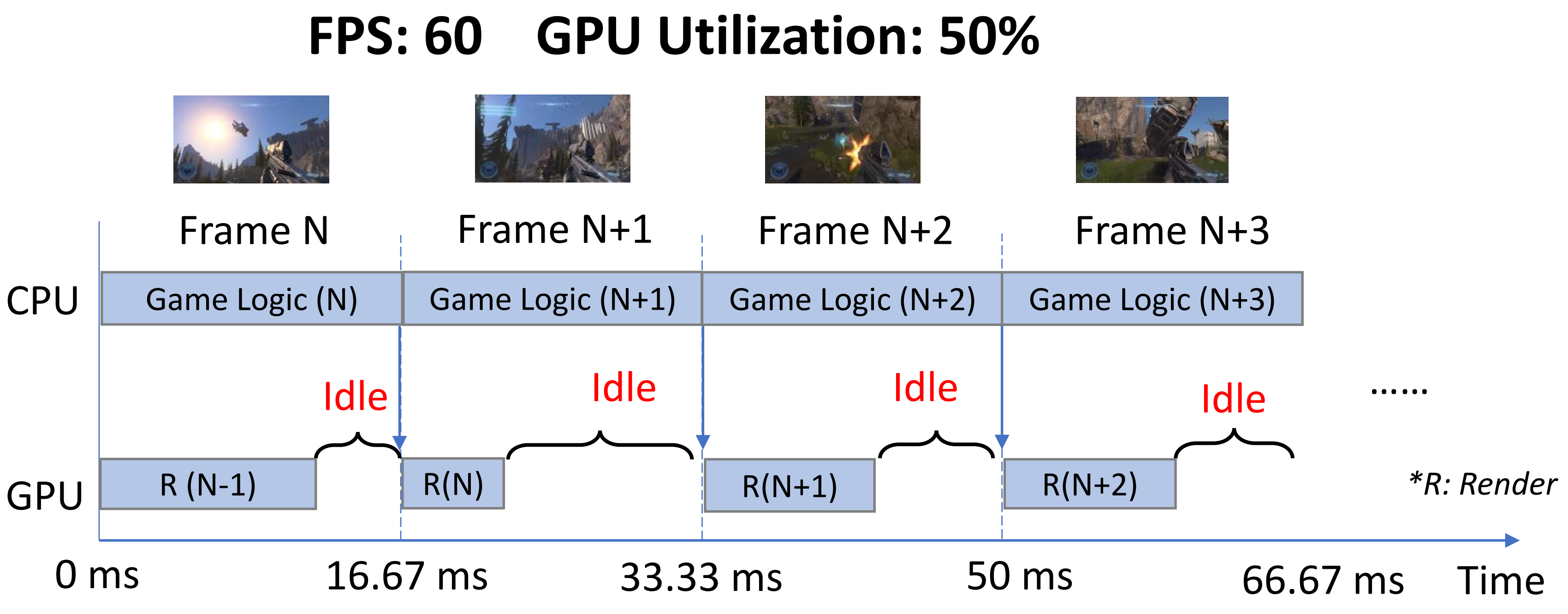
Device Limitation

- *Screen resolution*
- *HW acceleration for decoding*

Game	GPU Util.	VRAM (GB)	FPS	Lock FPS
Dota 2	38.2%	1.61	59.9	Yes
League of Legends	26.9%	1.16	59.8	Yes
PUBG	40.6%	4.05	60	Yes
CS:GO	45.0%	2.6	201	No
Civilization 5	32.3%	1.11	59.8	Yes
Assassin's CO	69.15%	2.39	59.6	Yes

1080p@60FPS on Nvidia RTX 2060 6.4 TFLOPS (comparable to XBOX's cloud gaming GPU)

GPU Rendering 101

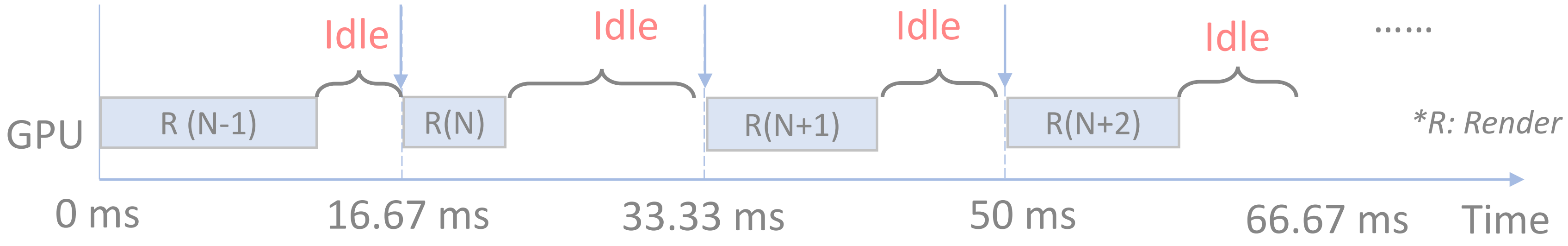


1. Game scenes are rendered frame by frame in a pipelined manner
2. The rendering time varies for different frames due to scene complexity
3. Idle GPU periods appear when GPU is underutilized

GPU Rendering 101

FPS: 60 GPU Utilization: 50%

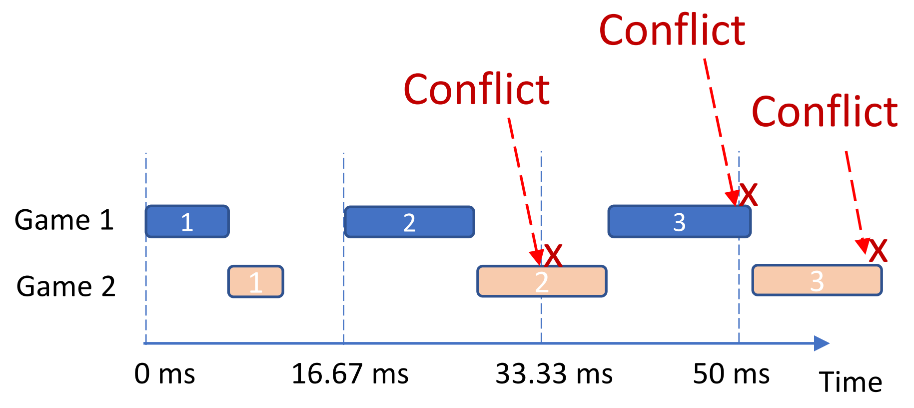
How can we harvest the idle GPU periods to improve GPU utilization?



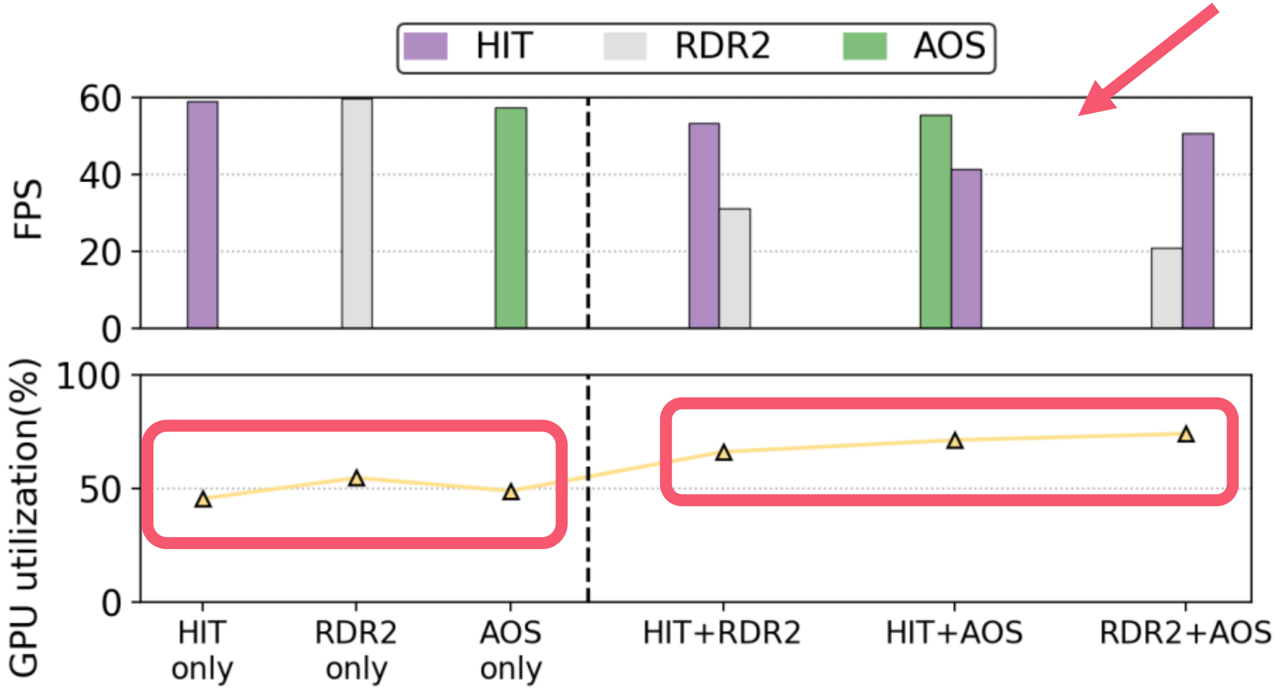
1. Game scenes are rendered frame by frame in a pipelined manner
2. The rendering time varies for different frames due to scene complexity
3. Idle GPU periods appear when GPU is underutilized

Run multiple games on single GPU?

- Games are too random
- High variation of rendering time
- Frequent conflicts



GPU timeline



Co-location multiple games does not improve much utilization but lead to severe FPS drop

HIT: HITMAN 3 RDR2: Red Dead Redemption 2 AOS: Ashes of Singularity

Requirements for Co-location with Games

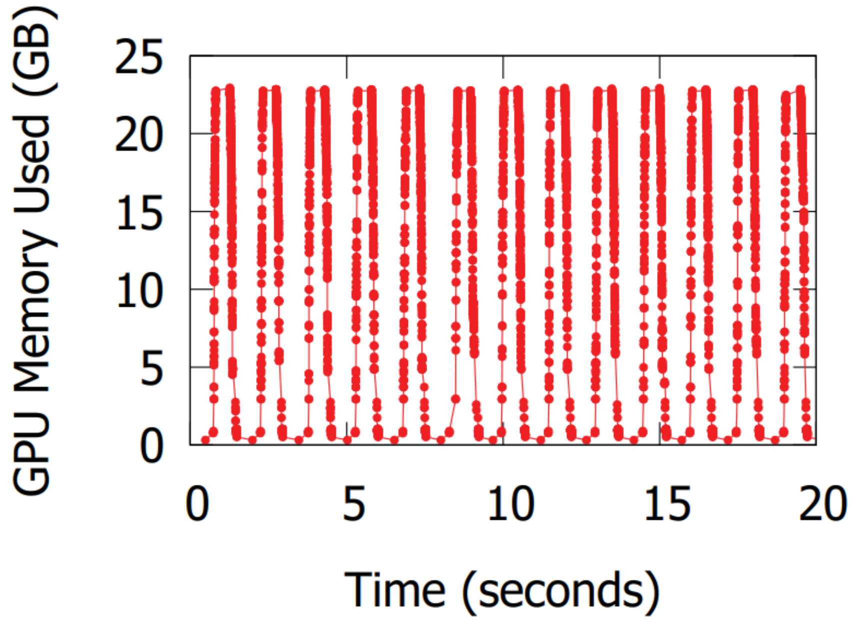
Quickly capture idle GPU periods

Predictable workload for co-location

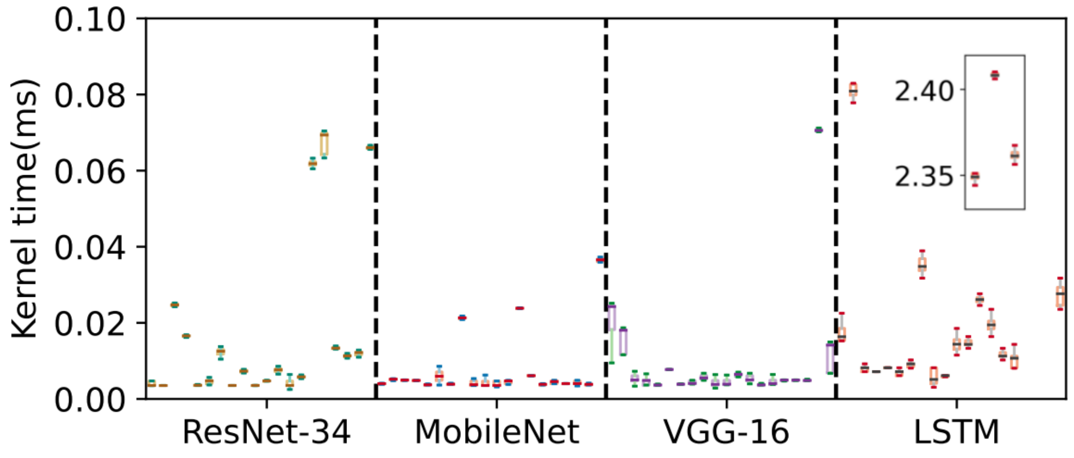
Quick preemption for straggler

DL Training is a Good Choice for Co-location

- Deep learning training is a stable and predictable workload
  - Repetitive and iterative pattern
  - Stable execution time and GPU memory usage
  - Fine-grained GPU kernels

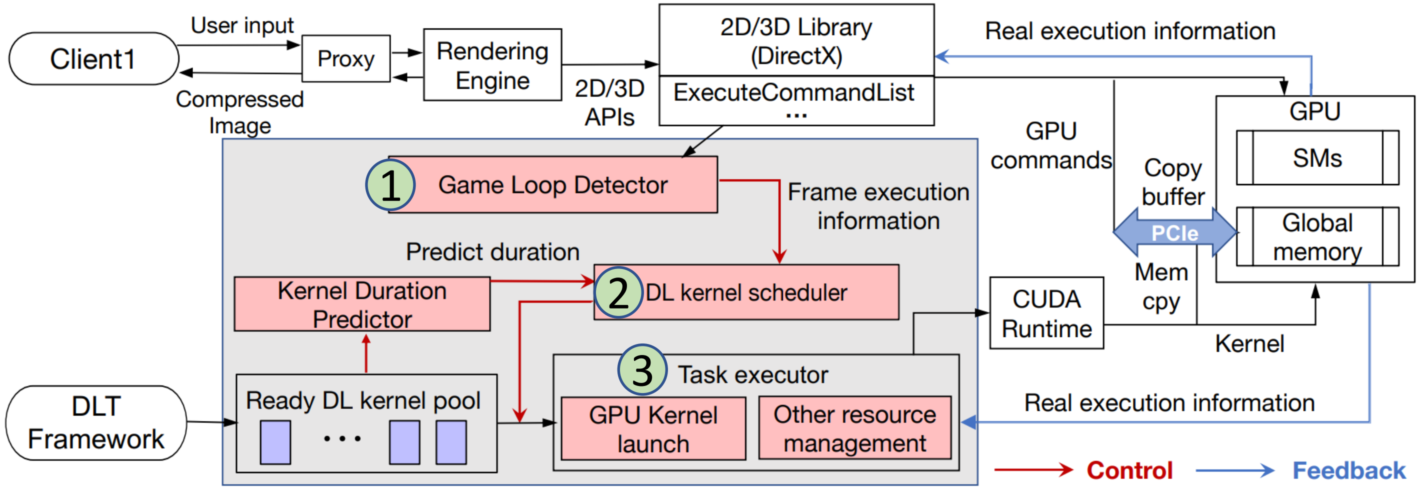


Stable iteration time and memory usage of each iteration [Gandiva, OSDI'18]



Stable and fine-grained GPU kernel

PilotFish System Design



Pilotfish Runtime System

1. Instrument rendering APIs to capture idle GPU periods
2. Fine-grained scheduling DL training kernels
3. Managing task execution to avoid potential interference

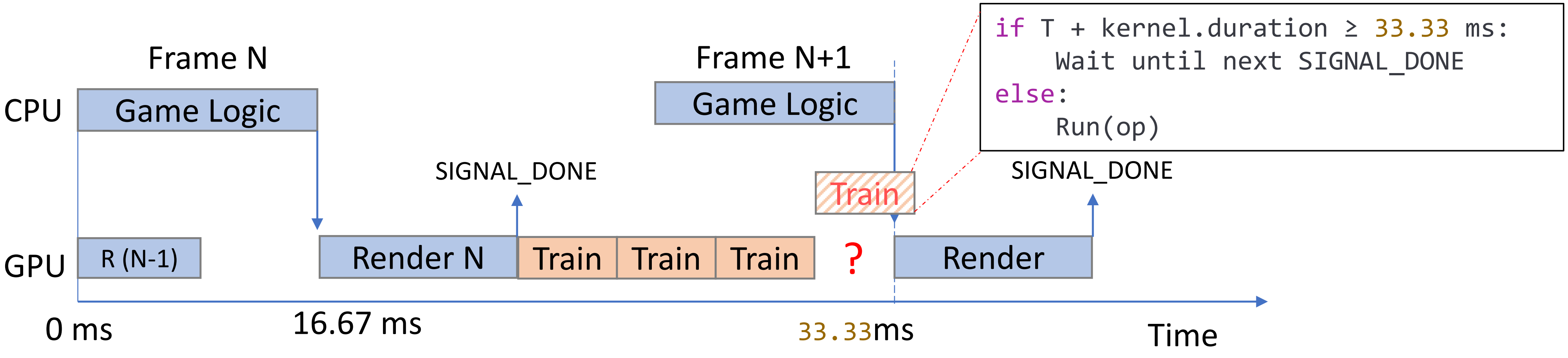
Real-time capturing idle GPU periods

- Rendering commands are compiled to GPU kernels via graphic libraries
  - E.g., DirectX 12 uses *ExecuteCommandLists* for submission
  - *Present():* an async call at the end of each frame
- Hook these APIs to monitor rendering task submission
- Insert a Signal to notify frame completion
- Do not require game modification

```
// Render the scene.
void D3D1211on12::OnRender()
{
    // Record all the commands we need to render the scene into the command list.
    PopulateCommandList();

    // Execute the command list.
    ID3D12CommandList* ppCommandLists[] = { m_commandList.Get() };

    m_commandQueue->ExecuteCommandLists(_countof(ppCommandLists), ppCommandLists);

    RenderUI();

    // Present the frame.
    ThrowIfFailed(m_swapChain->Present(1, 0));

    MoveToNextFrame();
}
```

A common procedure for game rendering

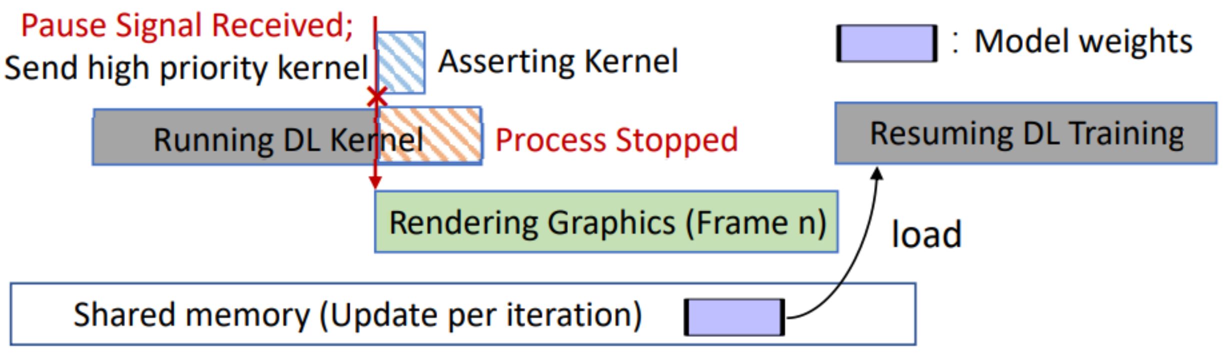
Fine-grained Scheduling of DL operation



Coordinated scheduling to avoid GPU interference

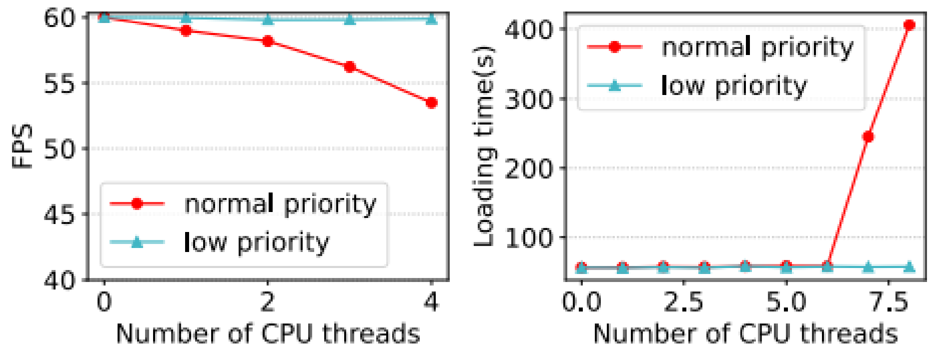
DL Training Task Executor

- Straggler kernels may execute longer than expected
  - Hard guarantee: preempt the DL training job immediately if next frame starts
  - Soft guarantee: allow slight FPS drop (1-2 FPS) to not preempt straggler kernels
- Fast preemption: 0.7 ms preemption latency
  - Using two GPU streams
    - Low-priority stream runs DL training kernels
    - high-priority stream only receives “asserting kernels”



Avoding contention on other resources

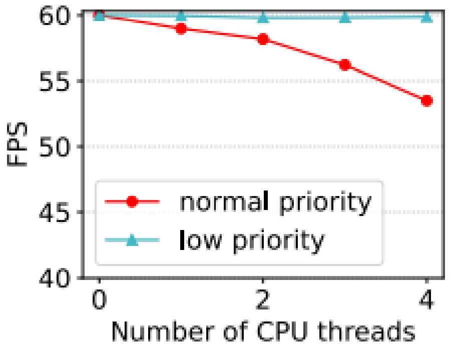
- CPU: thread priority



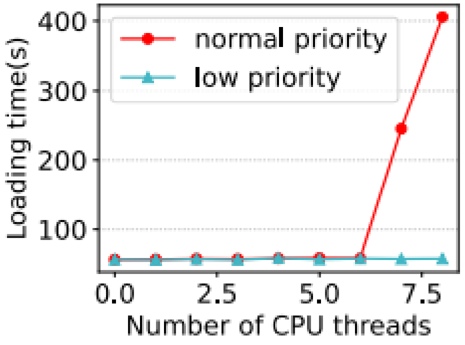
(a) FPS (b) Loading time

Avoding contention on other resources

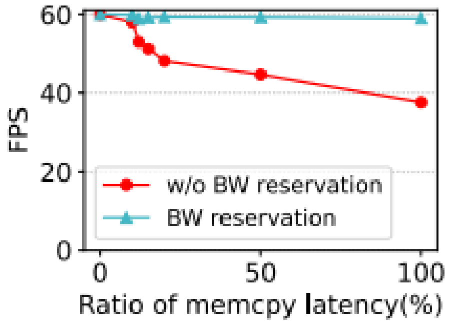
- CPU: thread priority
- PCI-e: Baymax* for PCI-e bandwidth reservation
- Disk I/O: namespace isolation and I/O priority



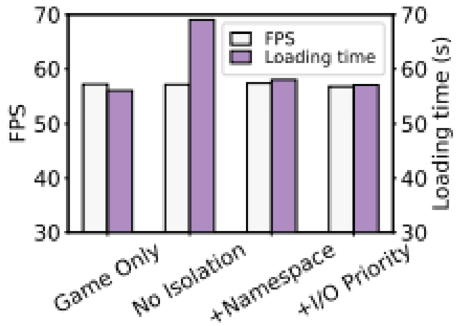
(a) FPS



(b) Loading time



(a) PCIe bus



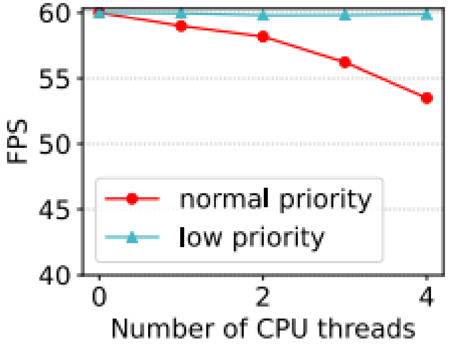
(b) Disk I/O

*Baymax: QoS-awareness and increased utilization for non-preemptive accelerators in warehouse scale computers

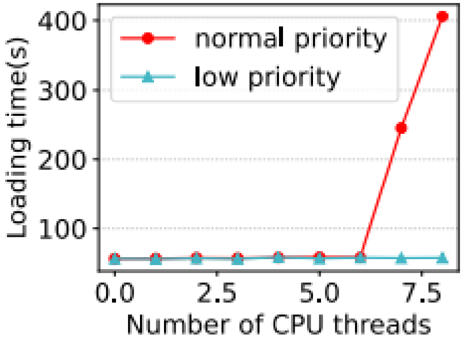
Avoding contention on other resources

- CPU: thread priority
- PCI-e: Baymax* for PCI-e bandwidth reservation
- Disk I/O: namespace isolation and I/O priority

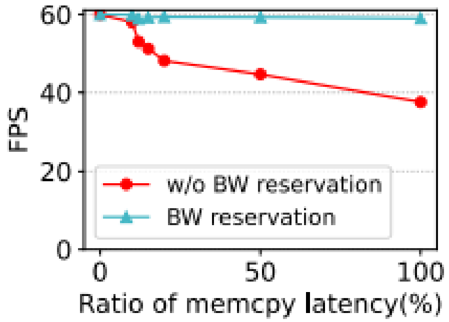
- GPU memory and cache
  - sum of peak GPU memory <= total GPU memory
  - No observed contention on GPU cache



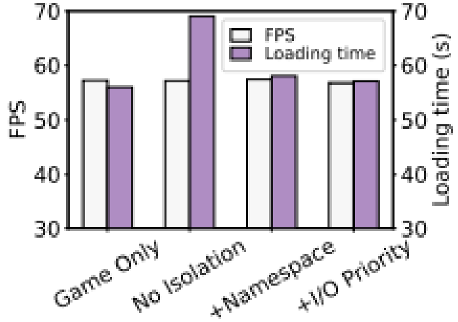
(a) FPS



(b) Loading time



(a) PCIe bus

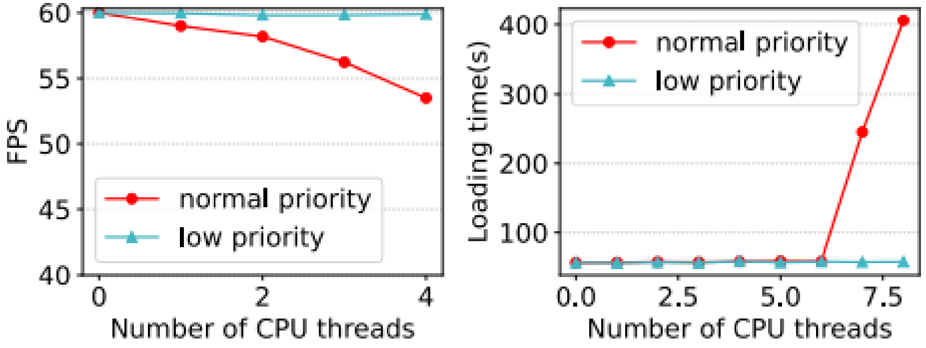


(b) Disk I/O

*Baymax: QoS-awareness and increased utilization for non-preemptive accelerators in warehouse scale computers

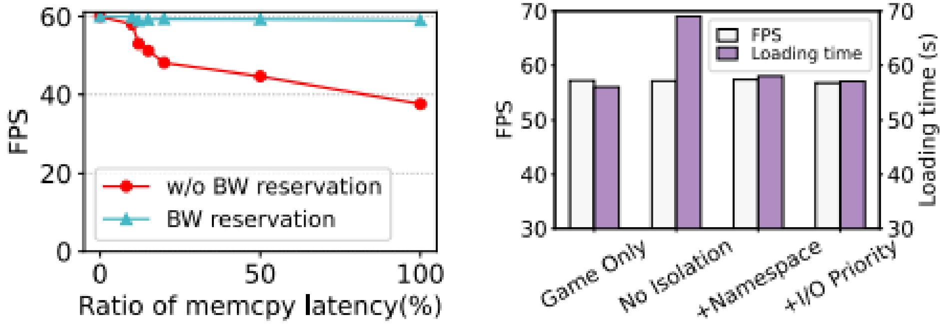
Avoding contention on other resources

- CPU: thread priority
- PCI-e: Baymax* for PCI-e bandwidth reservation
- Disk I/O: namespace isolation and I/O priority



(a) FPS (b) Loading time

- GPU memory and cache
  - sum of peak GPU memory <= total GPU memory
  - No observed contention on GPU cache



(a) PCIe bus (b) Disk I/O

- Network and video streaming encoding
  - No contention for seperated network and dedicated hardware encoder

*Baymax: QoS-awareness and increased utilization for non-preemptive accelerators in warehouse scale computers

Evaluation

- Game server
  - Intel i7-7700+RTX2060, Windows 10, CUDA 11, DirectX 12, PyTorch 1.8.1
- Games and DL models

Benchmarks	Workloads
Ashes of the Singularity	Crazy quality on 2560*1440; FPS: 60 GPU focused benchmark
Red Dead Redemption 2	Favor performance quality on 2560*1440; FPS: 60
Shadow of the Tomb Raider	High quality on 2560*1440; FPS: 60
F1 2021	Medium quality on 1920*1080; FPS: 60
HITMAN3	Ultra quality on 2560*1440; FPS: 60
DL Training	ResNet-34 (RS) [28]; VGG-16 [42] ; MobileNet (MN) [29]; LSTM [43]; Dataset: ImageNet-1k, Wikitext-2

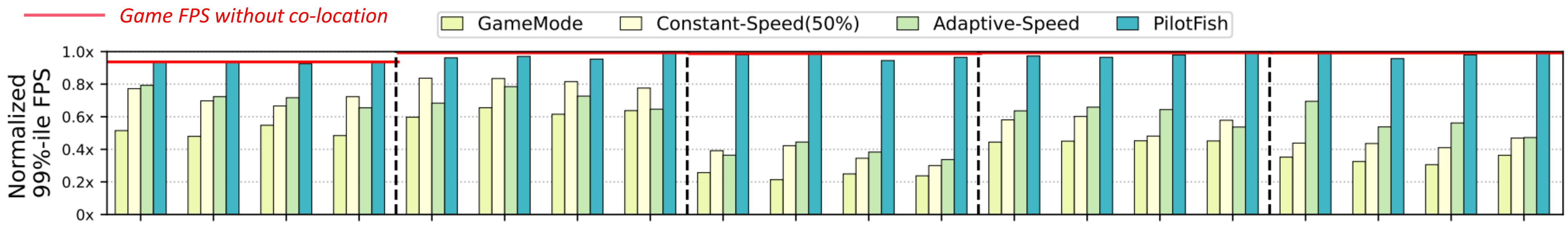
Evaluation (cont.)

- Harvest Ratio: the percentage of idle GPU cycles harvested

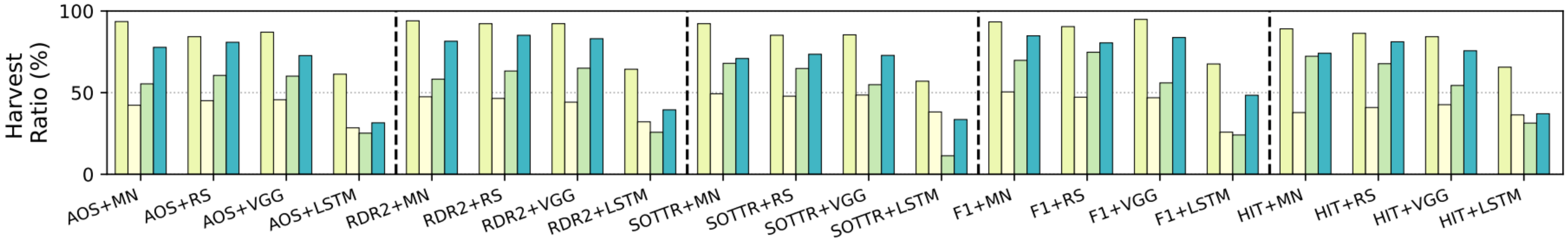
$$\text{Harvest Ratio} = \frac{\text{GPUUtil}_{\text{co}} - \text{GPUUtil}_{\text{Game}}}{100\% - \text{GPUUtil}_{\text{Game}}},$$

- Baselines:
  - Windows GameMode: only prioritizing CPU of game processes
  - Constant-Speed: controls the submission speed of DL kernels at a constant speed
  - Adaptive-Speed: using PresentMon to adaptively control DL kernel submission
    - If FPS < 60: speed = speed/2;
    - else: speed = speed*1.2.

Evaluation Result

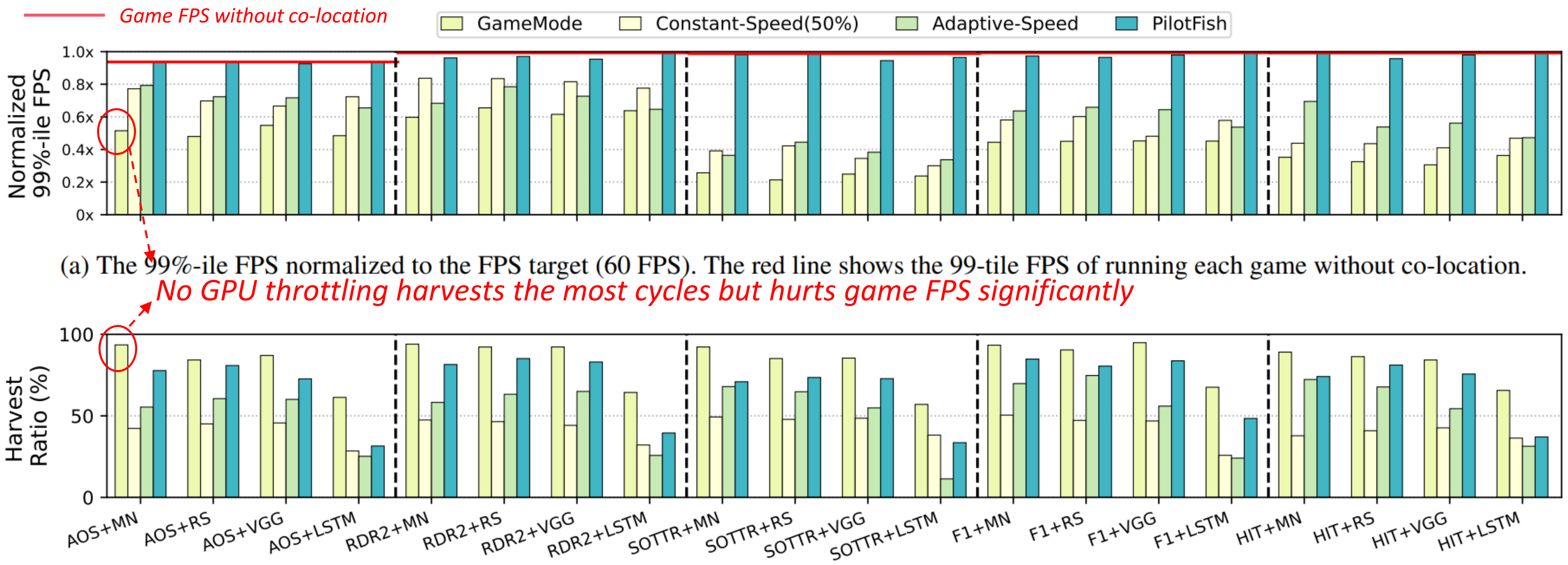


(a) The 99%-ile FPS normalized to the FPS target (60 FPS). The red line shows the 99-tile FPS of running each game without co-location.



(b) The harvest ratio of idle GPU time of cloud games.

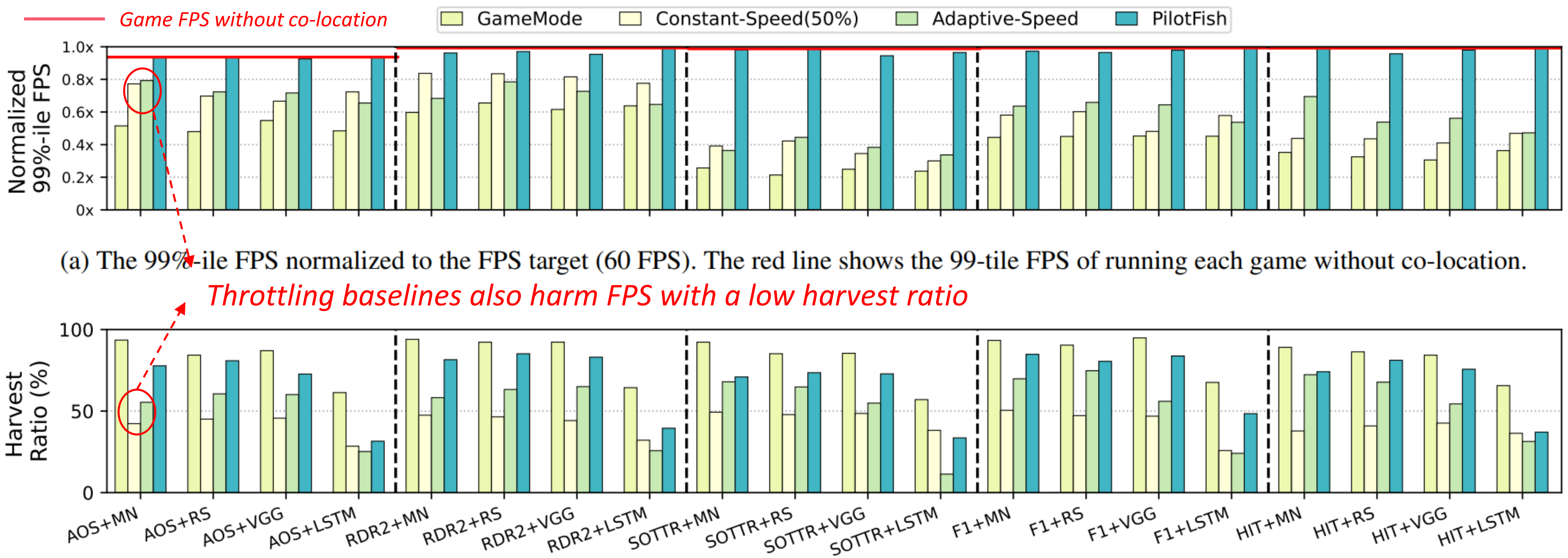
Evaluation Result



(a) The 99%-ile FPS normalized to the FPS target (60 FPS). The red line shows the 99-tile FPS of running each game without co-location.

(b) The harvest ratio of idle GPU time of cloud games.

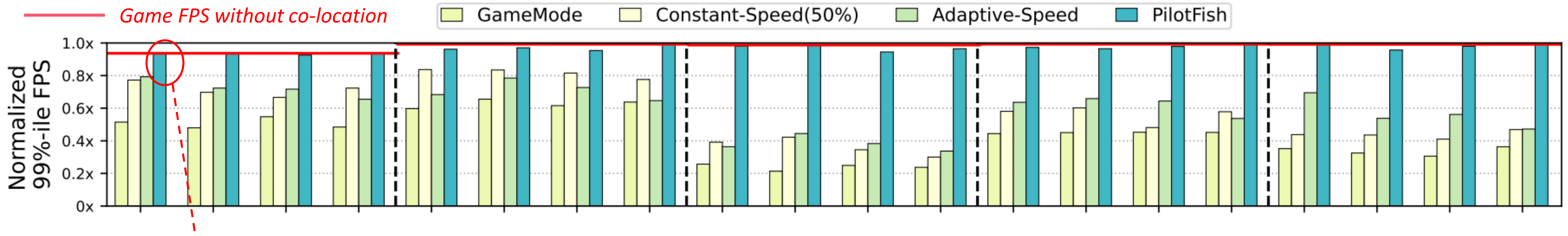
Evaluation Result



(a) The 99%-ile FPS normalized to the FPS target (60 FPS). The red line shows the 99-tile FPS of running each game without co-location.

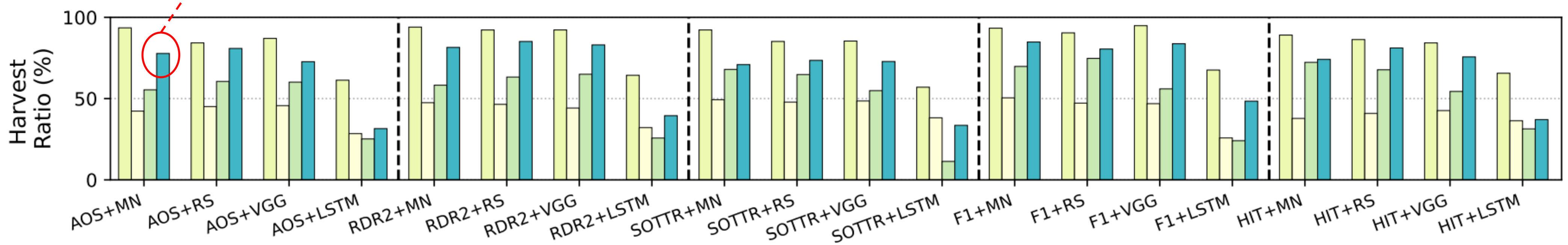
(b) The harvest ratio of idle GPU time of cloud games.

Evaluation Result



(a) The 99%-ile FPS normalized to the FPS target (60 FPS). The red line shows the 99-tile FPS of running each game without co-location.

PilotFish guarantees no interference to FPS with high harvest ratio

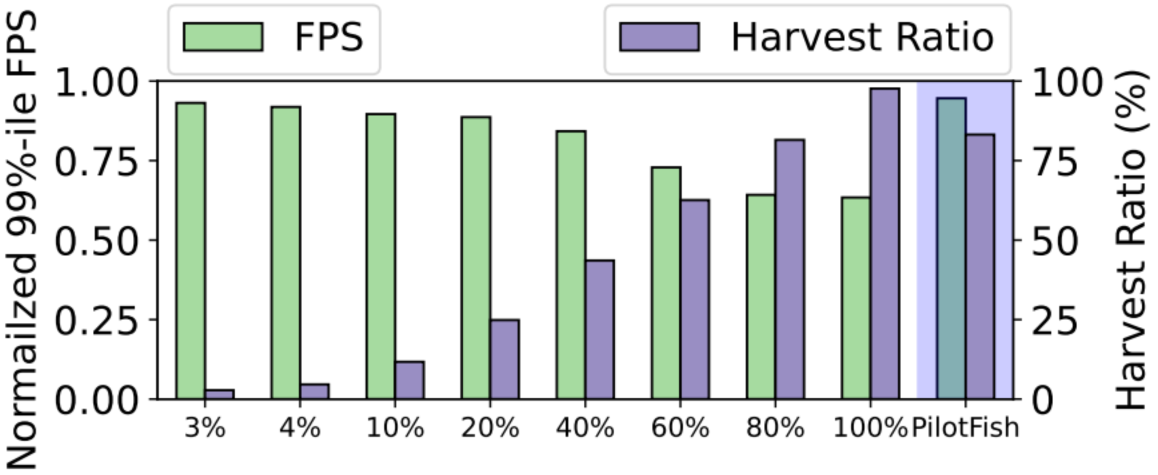


(b) The harvest ratio of idle GPU time of cloud games.

Source of improvement

dynamic scheduling

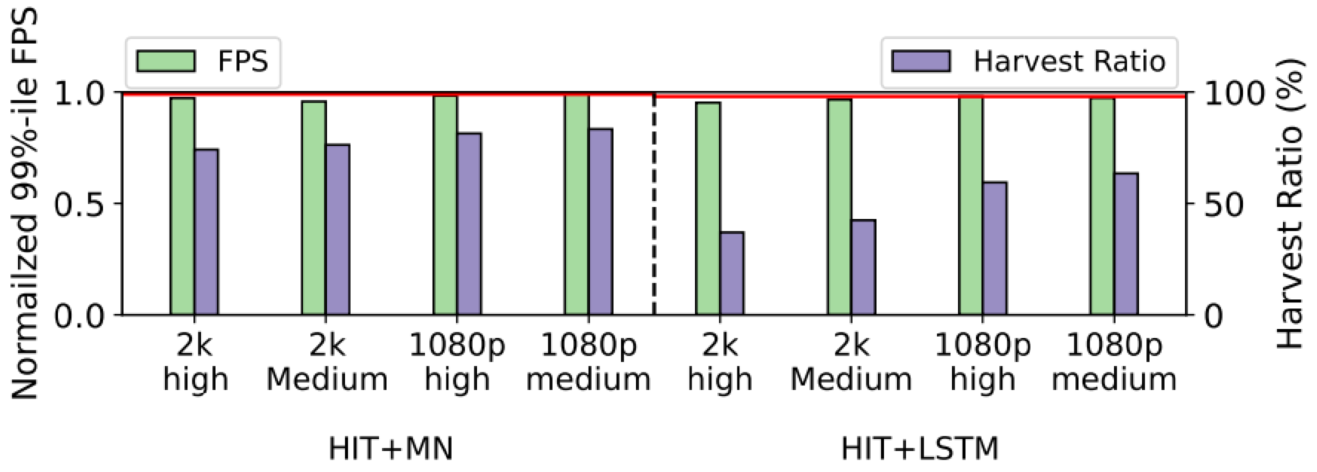
- Constant-Speed will not impact FPS only when its speed is ≤ 3%
- PilotFish harvests the idle GPU cycles as Constant-Speed(80%) without impacting FPS

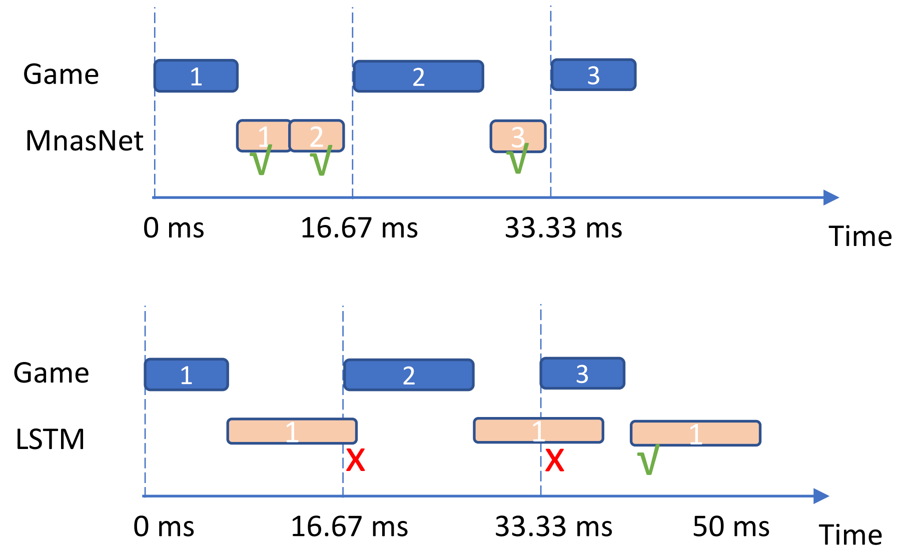


The Red Dead Redemption 2 + ResNet34

Different harvest ratios for different models

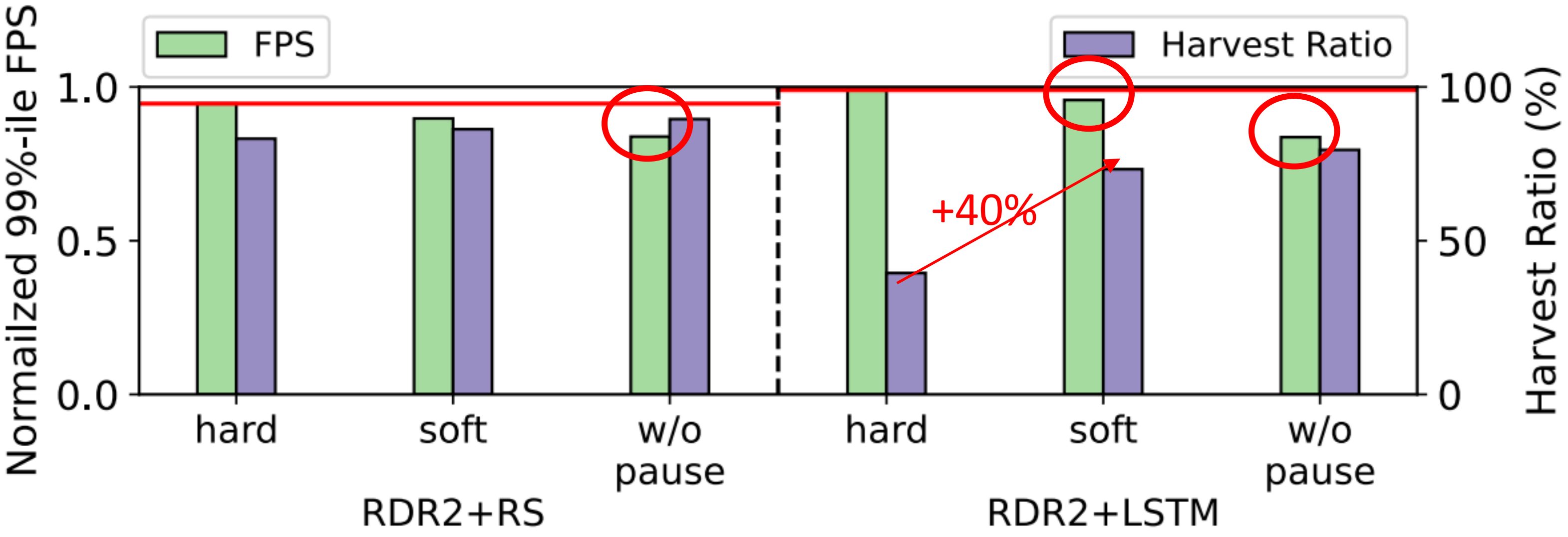
- LSTM has more long running kernels than MnasNet
  - Harder to find safe scheduling opportunity for LSTM





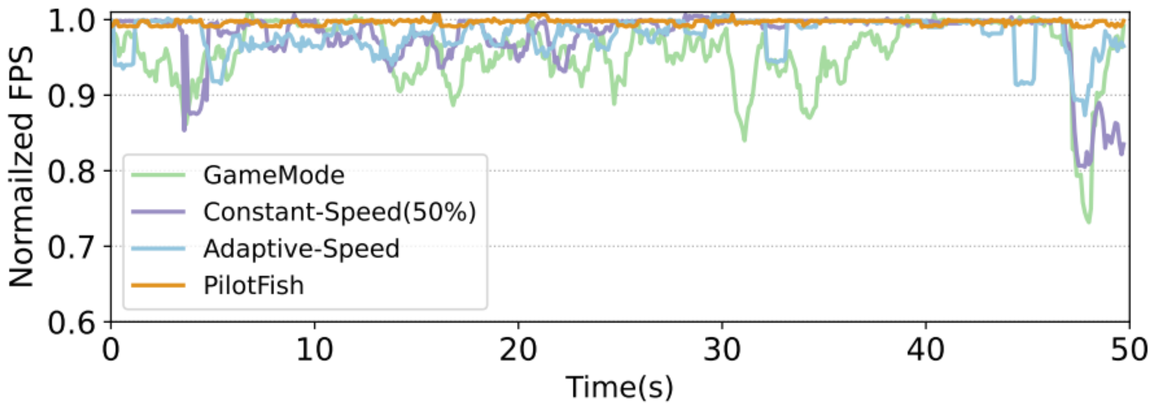
Soft/Hard Guanratee to Games

- Soft guarantee is useful for models with long kernels like LSTM
- Pausing is necessary for preempting straggler kernels



Game FPS over time when co-location

- Co-located with ResNet-34 (batch size = 8)
- The FPS drop in baselines may lead to reduced rendering quality



PilotFish Game Only Baseline co-location

Demo

Game:
- Tom Clancy's The Division 2
- FPS locks at 60
- Resolution: 1920*1080
- Quality: Highest

DL Training:
- Model: ResNet-50
- Dataset: cifar-10
- Batch Size: 16

Co-location PilotFish | **Game Only** | **Co-location No Throttling**

FPS

Video Link:
https://github.com/Chen-Binghao/PilotFish

Conclusion of PilotFish

- Cloud gaming has low-utilization due to limited streaming quality on powerful GPUs

- PilotFish: harvesting free GPU cycles of cloud gaming w/ DL training
    - Quickly capture GPU idle periods via API instrumentation
    - Leverage DL training's predictability to safely schedule computation kernels
    - Low-overhead pausing mechanism to prevent interference from stragglers

- PilotFish can harvest up to 85.1% idle GPU cycles without interfere to games

- Thanks.
- Please feel free to raise your questions

- Contact:
  - Zhenhua Han (Zhenhua.Han@microsoft.com)