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
Cloud Gaming

Cloud Data Center
GPU servers or Game Consoles
- GPU
- GPU
- GPU
- GPU
- GPU

BROADBAND INTERNET

Control Message
Low Latency Video

Play games anywhere and anytime

XBOX Cloud Gaming
Low GPU Util. of Cloud Gaming

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

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

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

1080p@60FPS on Nvidia RTX 2060 6.4 TFLOPS (comparable to XBOX’s cloud gaming GPU)
GPU Rendering 101

FPS: 60  GPU Utilization: 50%

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

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

```c
// 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”
Avoiding contention on other resources

• CPU: thread priority
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

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

*Baymax: QoS-awareness and increased utilization for non-preemptive accelerators in warehouse scale computers
Avoiding 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
- Network and video streaming encoding
  - No contention for separated 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

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ashes of the Singularity</td>
<td>Crazy quality on 2560*1440; FPS: 60 GPU focused benchmark</td>
</tr>
<tr>
<td>Red Dead Redemption 2</td>
<td>Favor performance quality on 2560*1440; FPS: 60</td>
</tr>
<tr>
<td>Shadow of the Tomb Raider</td>
<td>High quality on 2560*1440; FPS: 60</td>
</tr>
<tr>
<td>F1 2021</td>
<td>Medium quality on 1920*1080; FPS: 60</td>
</tr>
<tr>
<td>HITMAN3</td>
<td>Ultra quality on 2560*1440; FPS: 60</td>
</tr>
<tr>
<td>DL Training</td>
<td>ResNet-34 (RS) [28]; VGG-16 [42] ; MobileNet (MN) [29]; LSTM [43]; Dataset: ImageNet-1k, Wikitext-2</td>
</tr>
</tbody>
</table>
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%-ile FPS of running each game without co-location.

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

---

**Game FPS without co-location**

- No GPU throttling harvests the most cycles but hurts game FPS significantly.

---

(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.

---

No GPU throttling harvests the most cycles but hurts game FPS significantly.
Evaluation Result

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

Throttling baselines also harm FPS with a low harvest ratio.

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

(a) The 99%-tile 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 \( \leq 3\% \)
• PilotFish harvests the idle GPU cycles as Constant-Speed(80%) without impacting FPS
Different harvest ratios for different models

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

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)