# AGILE: Lightweight and Efficient Asynchronous GPU-SSD Integration

**Zhuoping Yang**∗, Jinming Zhuang∗, Xingzhen Chen∗, Alex K. Jones†, and Peipei Zhou∗

Brown University*; Syracuse University†;

zhuoping_yang@brown.edu
peipei_zhou@brown.edu

https://peipeizhou-eecs.github.io/

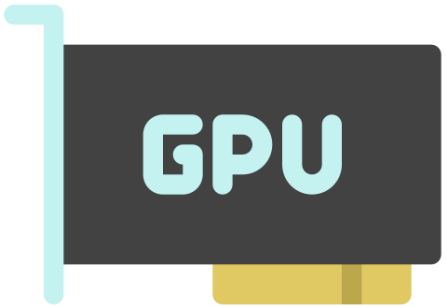
https://github.com/arc-research-lab/AGILE


School of Engineering
BROWN UNIVERSITY

Deep Learning

Computer Vision

Autonomous Driving



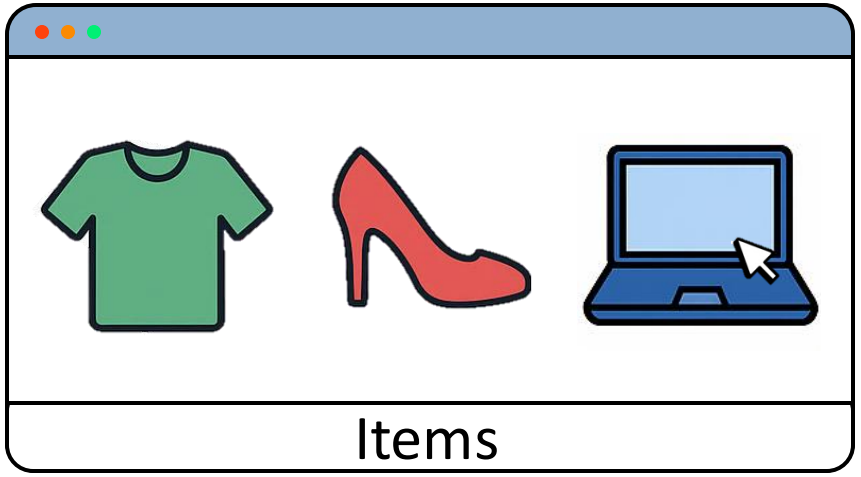

Finance / Risk Modeling

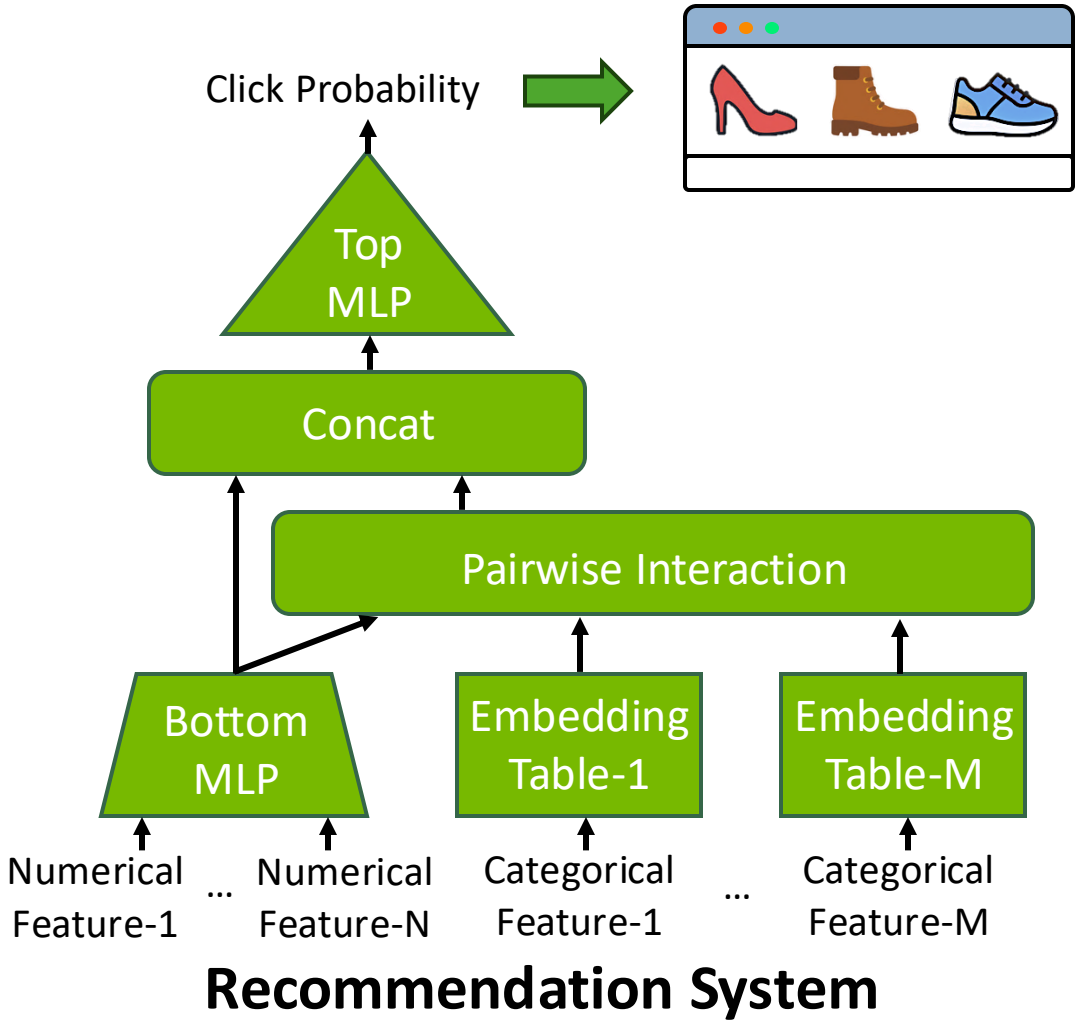
**GPGPU: The engine behind modern applications.**

Graph Analytics

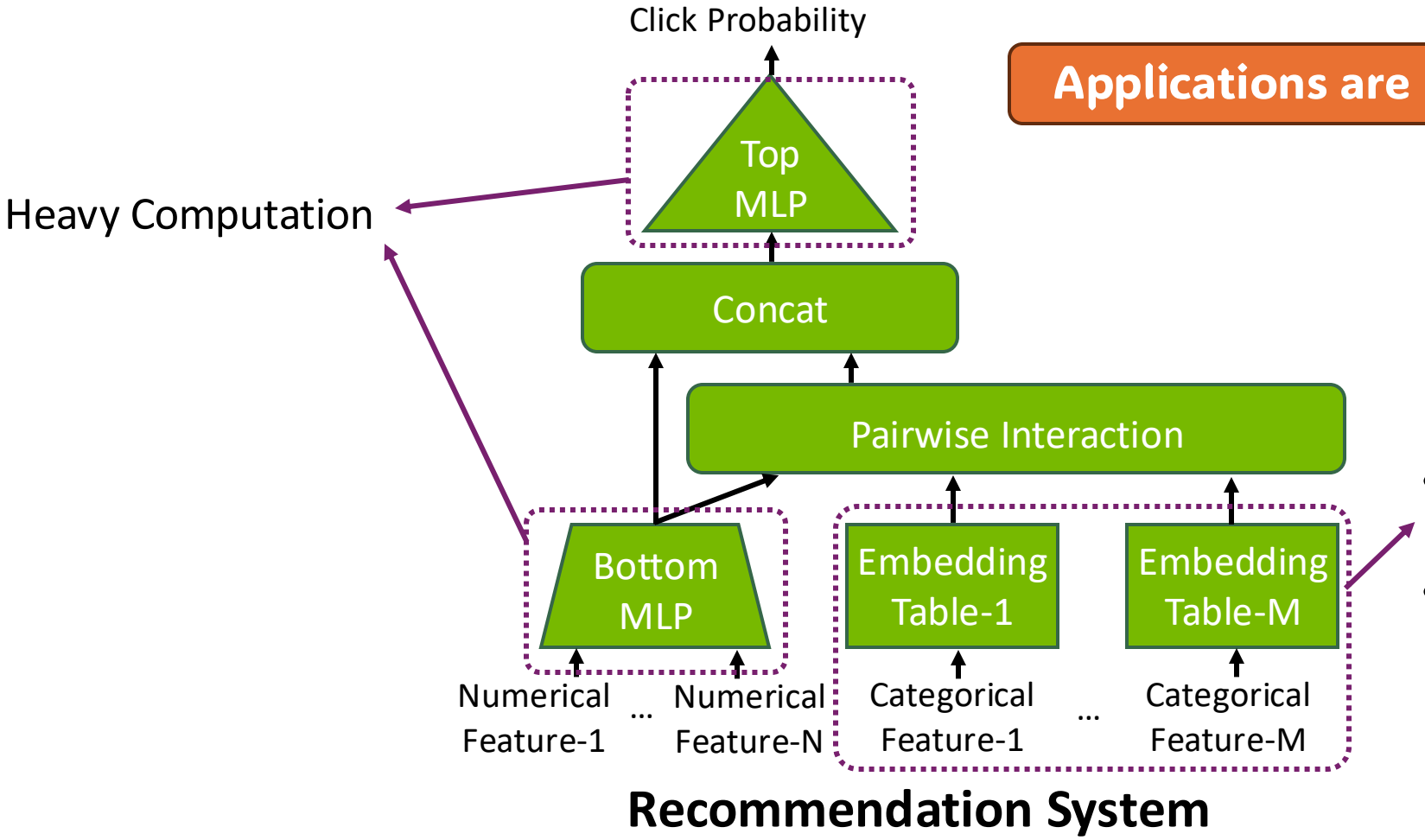
Scientific Simulation

Video Processing



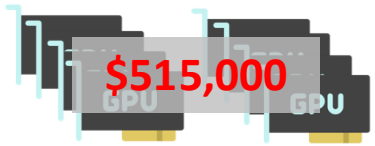

**Recommendation System**




**Applications are more than just Computation!**

- Huge embedding tables **far beyond the GPU's memory.**
- Irregular Access.

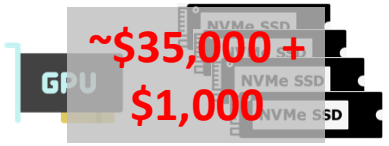
School of Engineering
BROWN UNIVERSITY



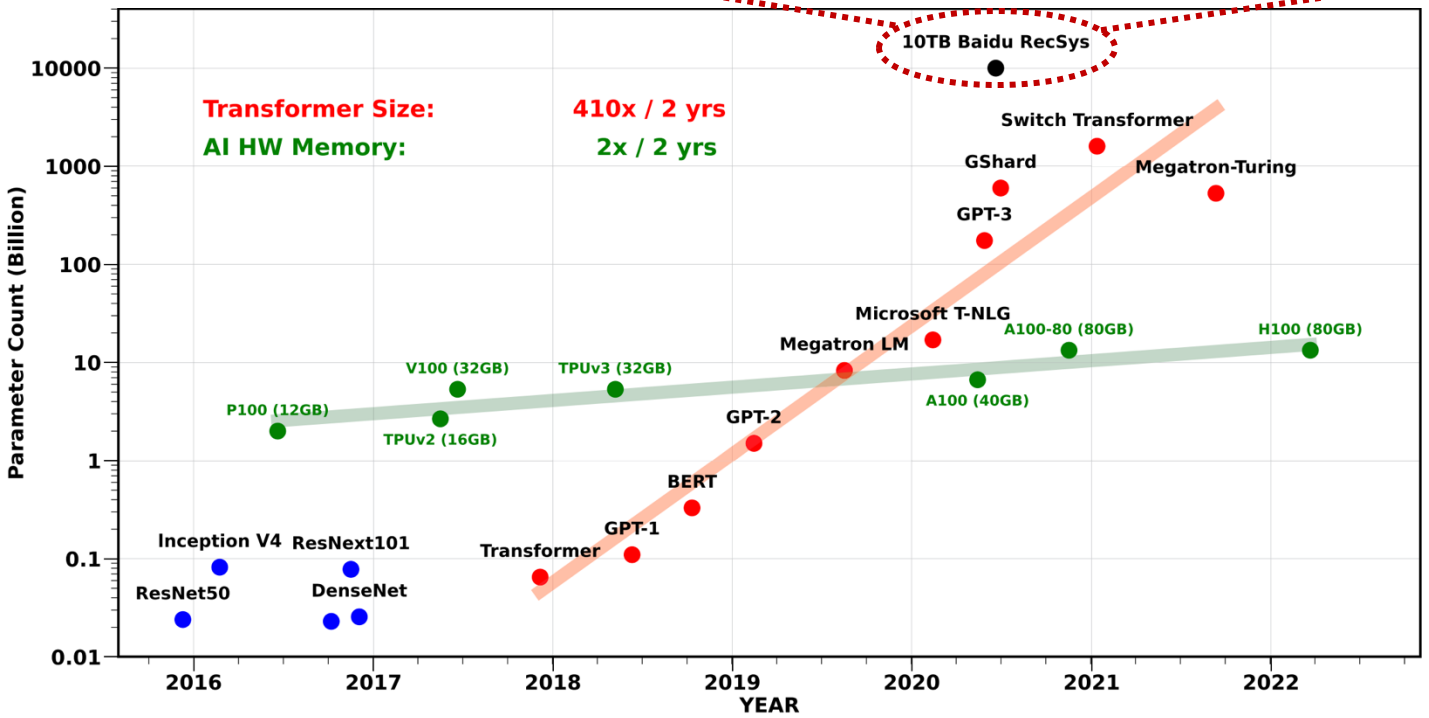

NVIDIA DGX B200
(8x B200 GPUs)

V.S.

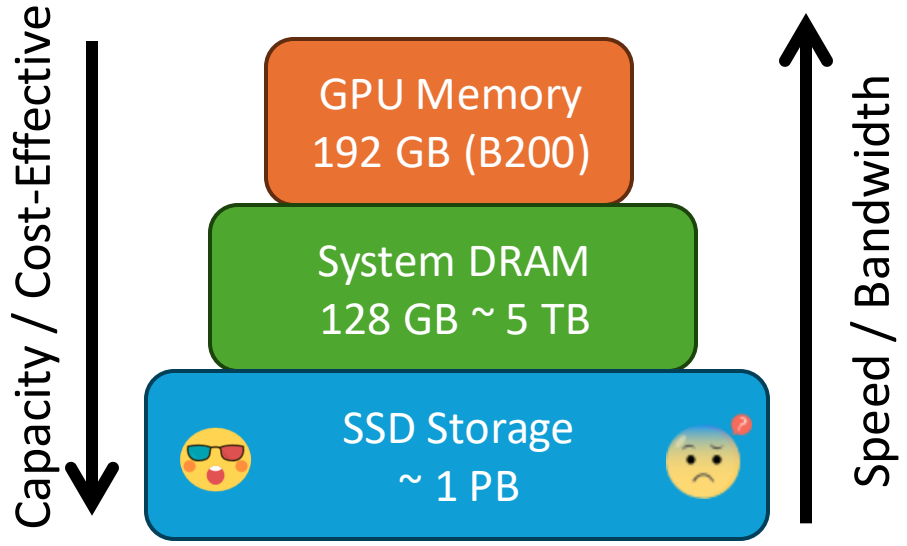
🤩 14x Cheaper!




NVIDIA B200 +
4x 4TB Samsung 990 EVO Plus




AI and Memory Wall[1]




Hierarchical memory in modern computing systems

**GPUs are increasingly constrained by the memory wall**

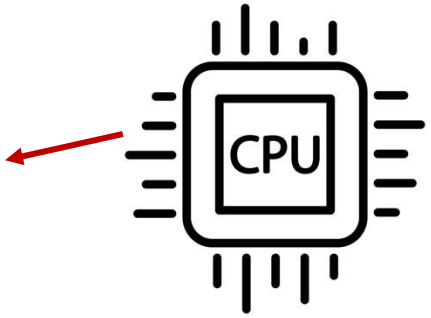
[1] Gholami, Amir, et al. "Ai and memory wall." *IEEE Micro* 44.3 (2024): 33-39.

# BaM[1] (A GPU-Centric Storage Access Model)

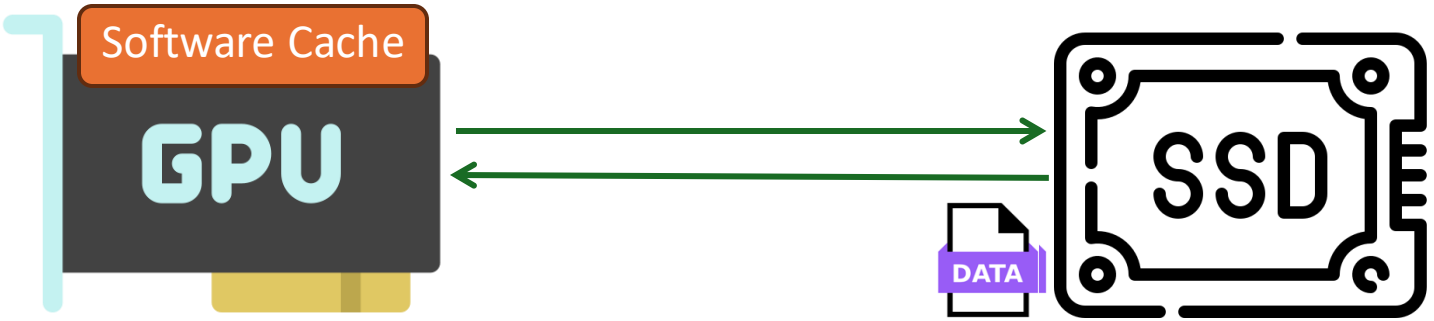
School of Engineering
BROWN UNIVERSITY

CPU is bypassed completely in both the control plane and the data plane.

- No extra copies or synchronization overheads




Can we further improve the GPU-centric storage access model?




[1] Zaid Qureshi, Vikram Sharma Mailthody, et al. "GPU-initiated on-demand high-throughput storage access in the BaM system architecture." Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2023.

# Programming Models for GPU-Centric I/O

School of Engineering
BROWN UNIVERSITY

## Async I/O enables overlapping the slow transfer with other tasks.

```
1 def kernel_block_access
2     a = read_block(..)
3     b = read_block(..)
4     compute(a, b)
5     ...
```

Synchronous I/O Model (BaM)

- Return after the slow transfer is finished.
- GPU thread must wait for the slow transfer.

```
1 def kernel_async_access
2     a = read_async(..)
3     b = read_async(..)
4     ...
5     wait(a)
6     wait(b)
7     compute(a, b)
8     ...
```

Asynchronous I/O Model

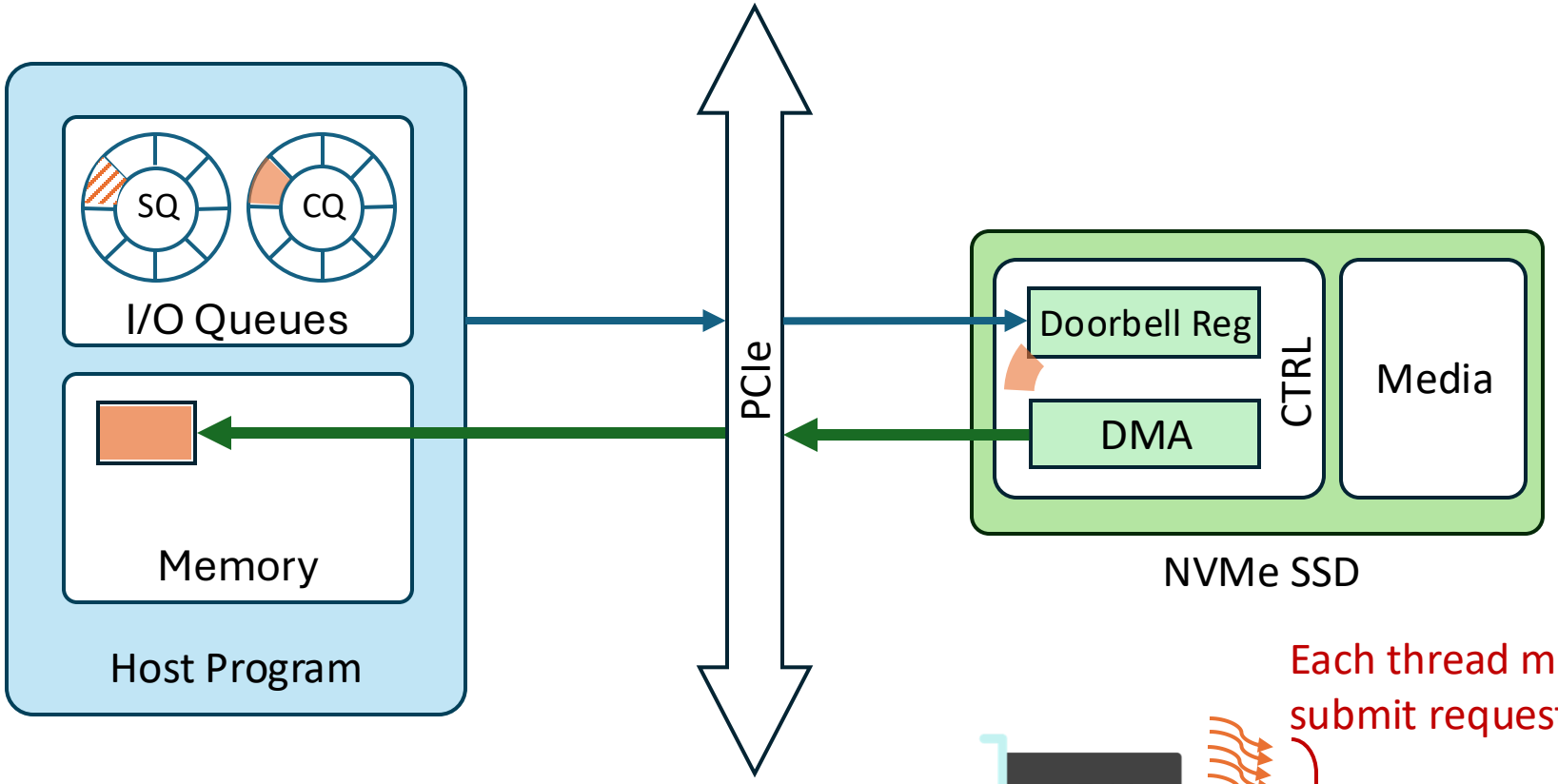
- Return quickly after the request is issued.
- Switch to other tasks during the slow transfer.
- Continue computation once the requested data is ready.

How can we support an async GPU-centric storage access model efficiently?

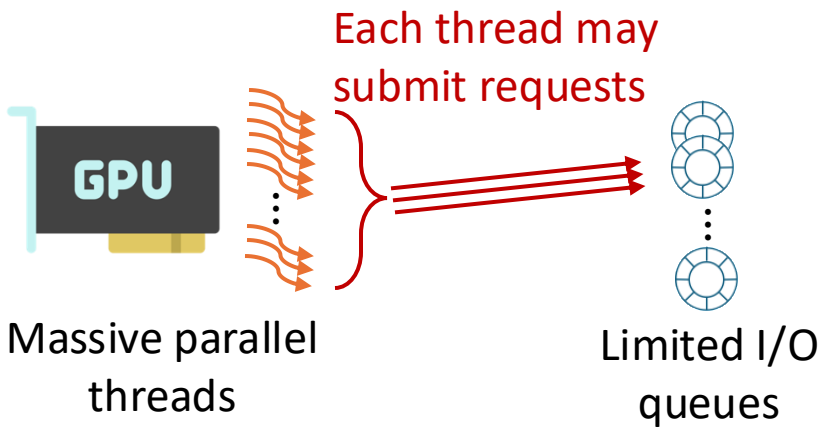
➢ Use AGILE! (this work) 🤩

# Background of NVMe Protocol






**The NVMe protocol is fundamentally asynchronous**

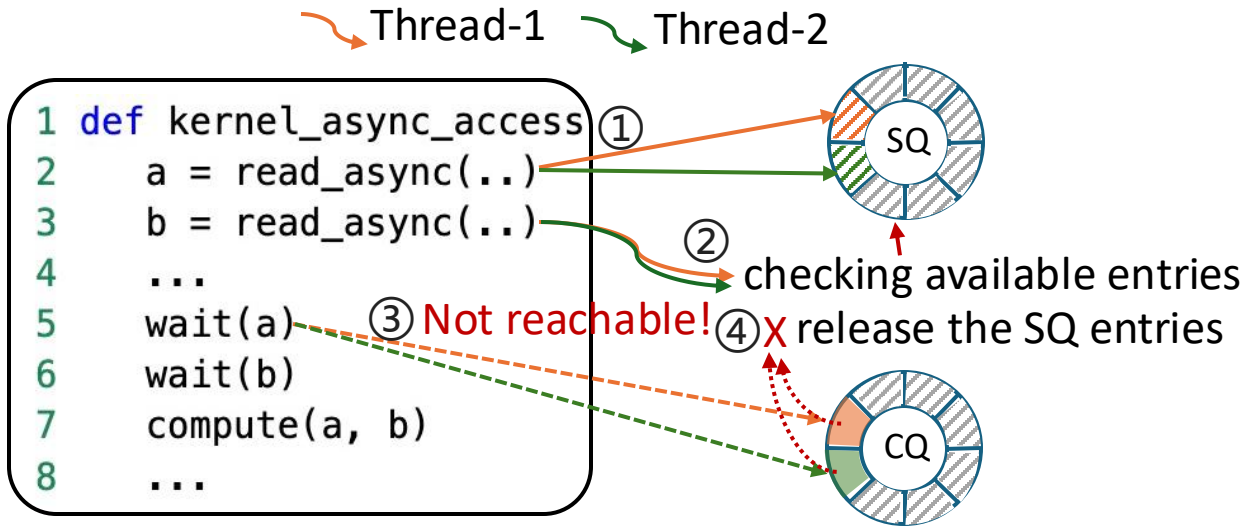


# Deadlock Risks in Async GPU-Centric NVMe I/O

School of Engineering
BROWN UNIVERSITY

Async GPU-centric NVMe I/O can easily lead to a deadlock.

- A deadlock example in GPU-centric NVMe read:



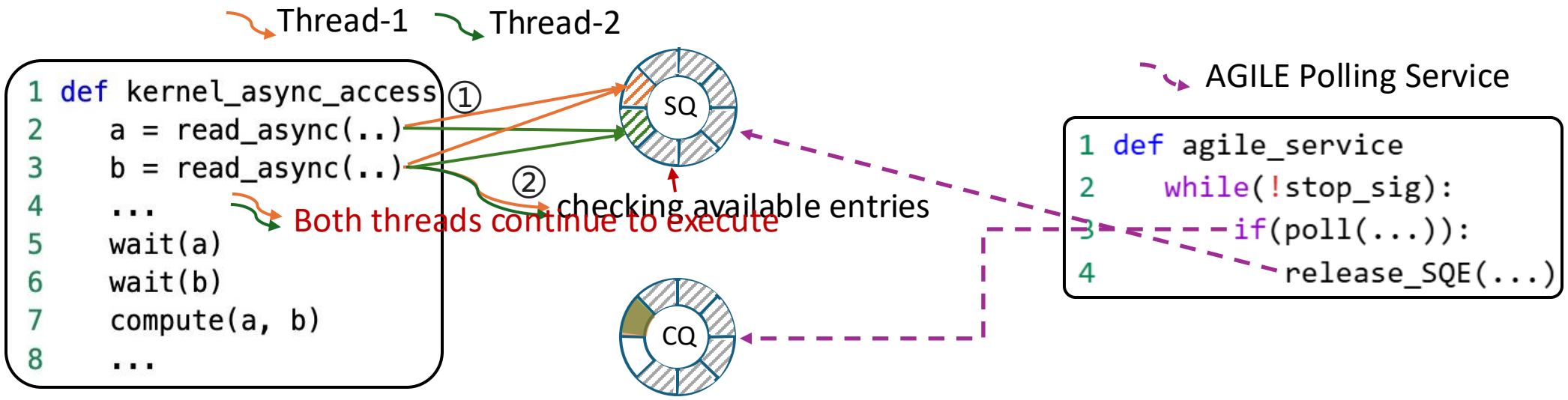

- The root cause of this deadlock is allowing user threads to **hold multiple locks** that can block other user threads. (I/O queues, software cache lines)

# AGILE Polling Service

School of Engineering
BROWN UNIVERSITY

To avoid deadlock from asynchronous NVMe I/O:

- A warp-centric AGILE polling service running in the background



# AGILE Software Cache

School of Engineering
BROWN UNIVERSITY

To eliminate deadlock from the software cache:

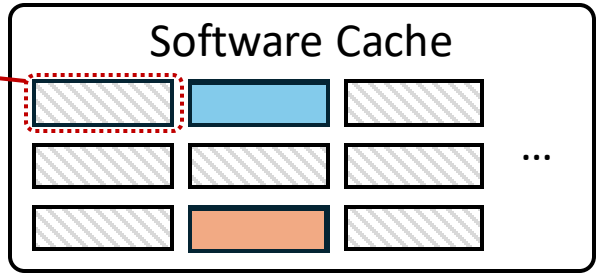
- AGILE does not allow user threads to avoid cache line eviction.

➢ Avoid Deadlocks at the cost of additional I/Os.

```
1 def kernel_async_access
2 →   a = read_async(..)
3 →   b = read_async(..)
4 →   ...
5 →   wait(a)
6 →   wait(b)
7 →   compute(a, b)
8       ...
```

Other threads request new software cache lines

a is evicted at line 6 to avoid blocking others




Unavailable again

➢ Use an extra synchronous read when accessing ‘a’

# Integrate User-managed Buffers into AGILE

School of Engineering
BROWN UNIVERSITY

By default, all accesses are proxied by AGILE software cache.

- Requested data may be unavailable again to eliminate deadlocks.

➢To address this problem, AGILE allows user-managed buffers.

```
1 def kernel_async_access
2     a = read_async(..)
3     b = read_async(..)
4     ...
5     wait(a)
6     wait(b)
7     compute(a, b)
8     ...
```

Access via AGILE software cache
(maybe require additional reads)

User-managed buffers

```
1 def kernel_async_access( \
      a_user_buf, b_user_buf):
2     read_async(a_user_buf)
3     read_async(b_user_buf)
4     ...
5     a_user_buf.wait()
6     b_user_buf.wait()
7     compute(a_user_buf, b_user_buf)
8     ...
```

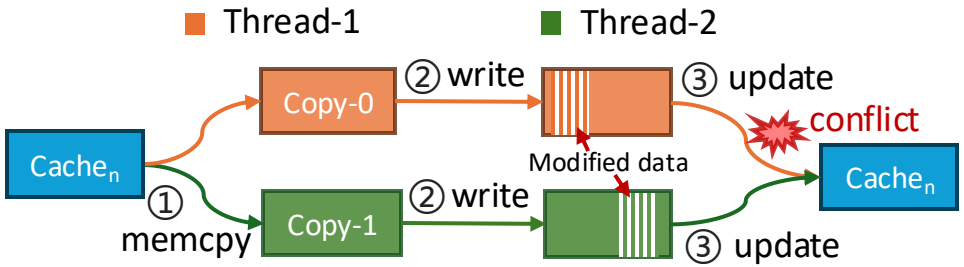
Data will not be evicted once ready.

Data is guaranteed to be ready.

# Integrate User-managed Buffers into AGILE

School of Engineering
BROWN UNIVERSITY

When multiple threads request the same data, data conflicts may occur.




AGILE provides a compile-time optional mechanism (share-table) to let user threads access the same user-managed buffer.

When the share-table is enabled at compile time:

- user-managed buffer (L1; managed by user; registered to share-table)
- Software cache (L2; managed by AGILE)
- SSDs (L3; managed by AGILE)

# AGILE's API

School of Engineering
BROWN UNIVERSITY

Customizing software cache policy

```
class GPUCache:public GPUCacheBase<GPUCache>{...};
#define AGILE_CTRL AgileCtrl<GPUCache, ShareTable>
__global__
void kernel(AGILE_CTRL * ctrl, void * data){
  ...
  AgileLockChain chain;
  // Method-1: AGILE prefetch
  ctrl->prefetch(dev_idx, blk_idx, chain);
  // Method-2: AGILE async_issue
  AgileBufPtr buf(data + tid * ctrl->line_size);
  ctrl->asyncRead(dev_idx, blk_idx, buf, chain);
  buf.wait();
  ctrl->asyncWrite(dev_idx, blk_idx, buf, chain);
  // Method-3: AGILE array-like synchronous API
  auto agileArr = ctrl->getArrayWrap<int>(chain);
  int val = agileArr[dev_idx][idx];
}
```

Debug option for detecting deadlock

```
int main(int argc, char** argv){
    // GPU Configurations
    AGILE_HOST host(...);
    // Policy Configurations
    SHARE_TABLE_IMPL s_table(...);
    GPU_CACHE_IMPL g_cache(...);
    host.setGPUCache(g_cache);
    host.setShareTable(s_table);
    // Add and open target SSDs in the program
    host.addNvmeDev("/dev/AGILE-xxx", ...);
    host.addNvmeDev("/dev/AGILE-xxx", ...);
    host.initNvme();
    // Initialize AGILE controller
    host.initializeAgile(...);
    // CUDA kernel parallelism configurations
    host.configKernelParallelism(...);
    host.queryOccupancy(kernel);
    // Start the lightweight AGILE service
    host.startAgile();
    // Execute the CUDA kernel
    host.runKernel(kernel, args...);
    // Stop AGILE service
    host.stopAgile();
    // Close the opened SSDs
    host.closeNvme();
}
```

Init AGILE & Start AGILE service

Start user kernel

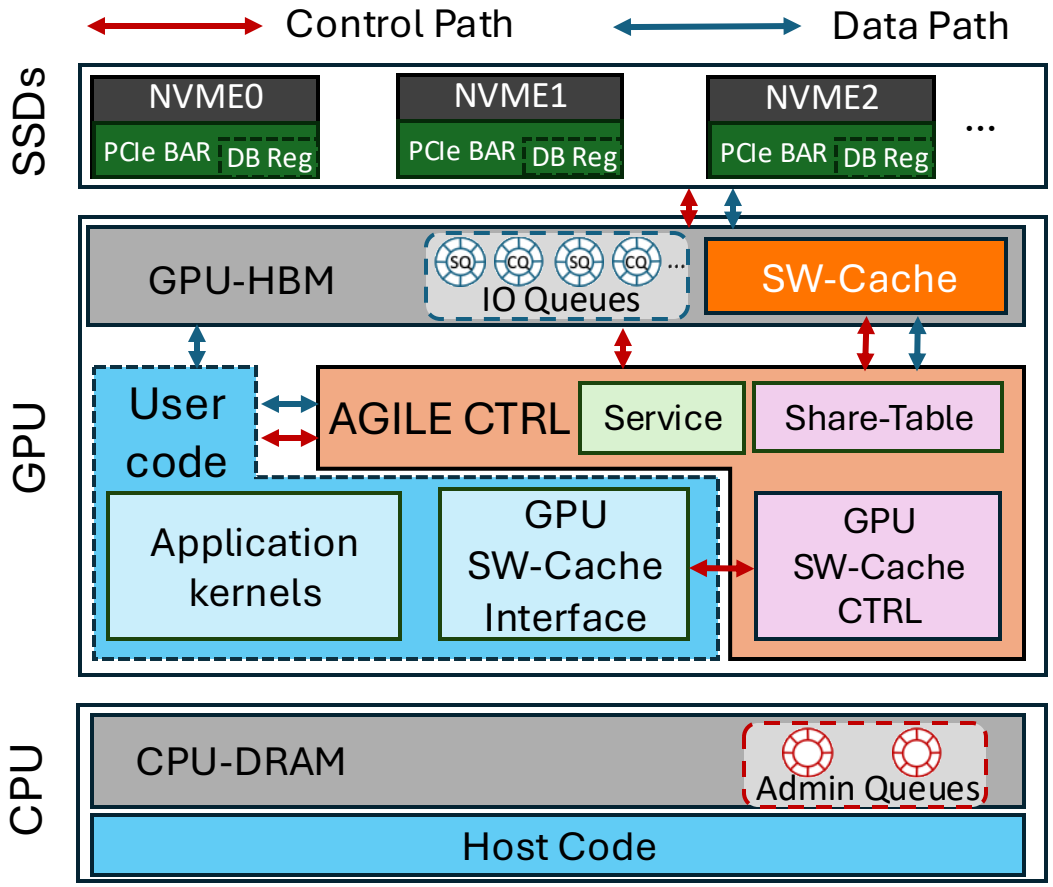
Stop AGILE service & close SSDs

# Summary of AGILE

School of Engineering
BROWN UNIVERSITY

Feature Highlights:

1. AGILE is the first async GPU-centric I/O model, allowing GPU threads to access SSDs asynchronously.
2. AGILE eliminates the deadlock risks in asynchronous NVMe I/O via AGILE polling service.
3. AGILE allows users to customize software cache policies.
4. AGILE allows user-maintained buffers to be integrated into the software cache hierarchy.




System overview of AGILE

# Experimental Setup

School of Engineering
BROWN UNIVERSITY

- Dell R750 Server
  - Ubuntu 20.04 (Linux 5.4.0-200-generic)
- Nvidia RTX 5000 Ada GPU (PCIe Gen4 x16)
- NVMe SSDs
  - Dell Ent NVMe AGN MU AIC 1.6TB SSD (PCIe Gen4 x4)
  - Two Samsung 990 PRO 1TB SSDs (PCIe Gen4 x4)
- Software Cache Policy
  - Clock replacement algorithm[1] (keep the same with BaM)

[1] F. J. Corbato. 1968. A Paging Experiment With The Multics System. Technical Report, Massachusetts Institute of Technology, Cambridge, Project MAC (1968).

# Experimental Evaluation

School of Engineering
BROWN UNIVERSITY

1. Compute and communication overlap.
   - 1 thread block issues 64 NVMe commands, and uses fetched data for compute.

➢AGILE can hide slow communication with computation effectively.

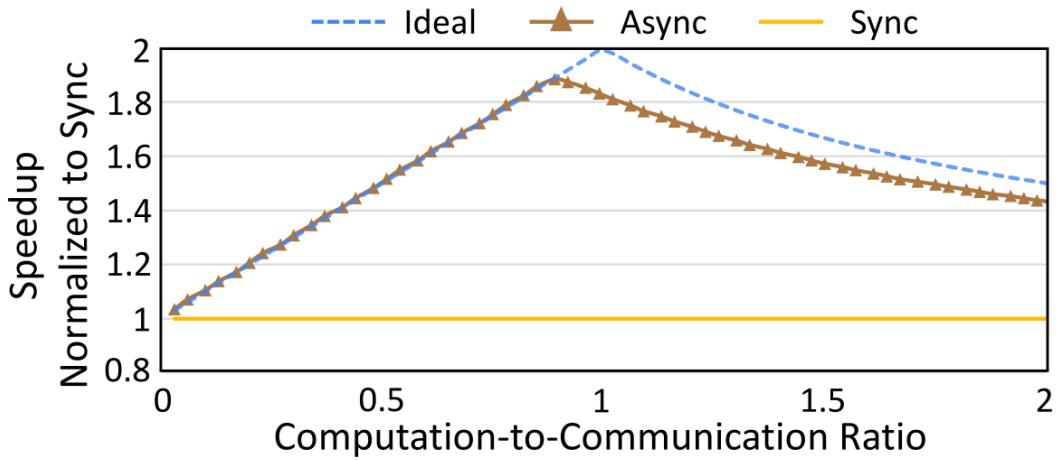



**Figure 4: Speedup comparison of asynchronous I/O over synchronous I/O on workloads with different Computation-to-Communication Ratio (CTC).**

# Experimental Evaluation

School of Engineering
BROWN UNIVERSITY

2. Scalability in 4KB random read/write
   - Access different SSDs in an interleaving fashion.

➢AGILE shows good scalability.

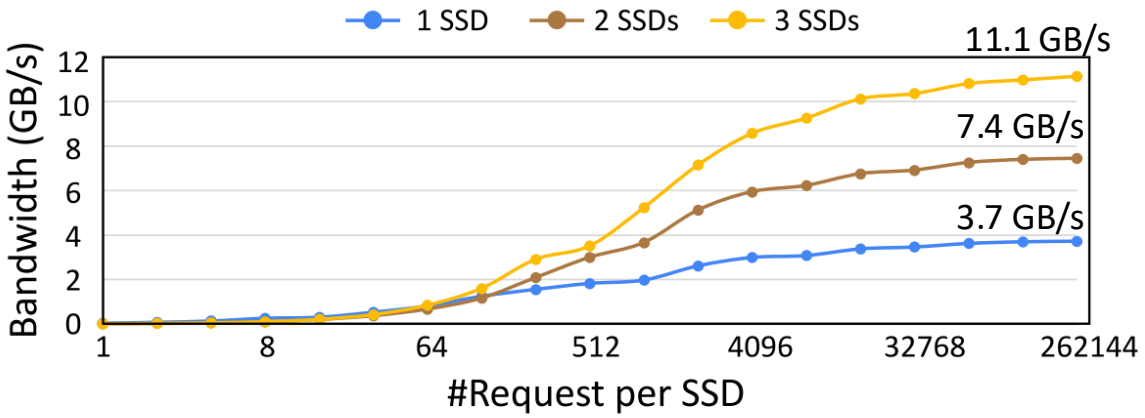



**Figure 5: AGILE 4KB random read on multiple SSDs**

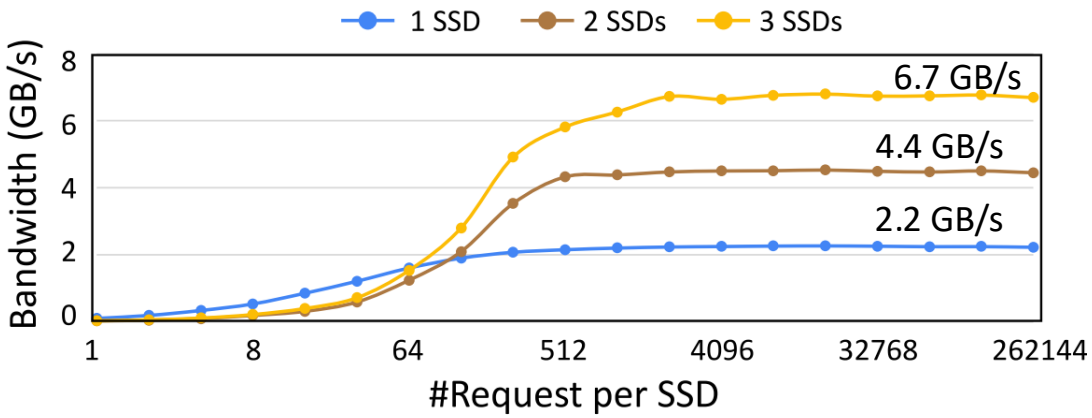



**Figure 6: AGILE 4KB random write on multiple SSDs**

# Experimental Evaluation

School of Engineering
BROWN UNIVERSITY

## 3. Comparison with BaM on DLRM inference MicroBenchmark.

- The DLRM model is adopted from [1] with Criteo 1TB Click Logs dataset[2].
- cuBLAS is used for all matrix multiplications.
- BaM and AGILE are used for fetching data.
- AGILE is used in both synchronous mode and asynchronous mode

Default parameters:

- Batch Size: 2048
- #I/O queues: 128
- Software cache size: 2 GB

- Sweep key parameter: Batch Size & the Number of I/O Queues.

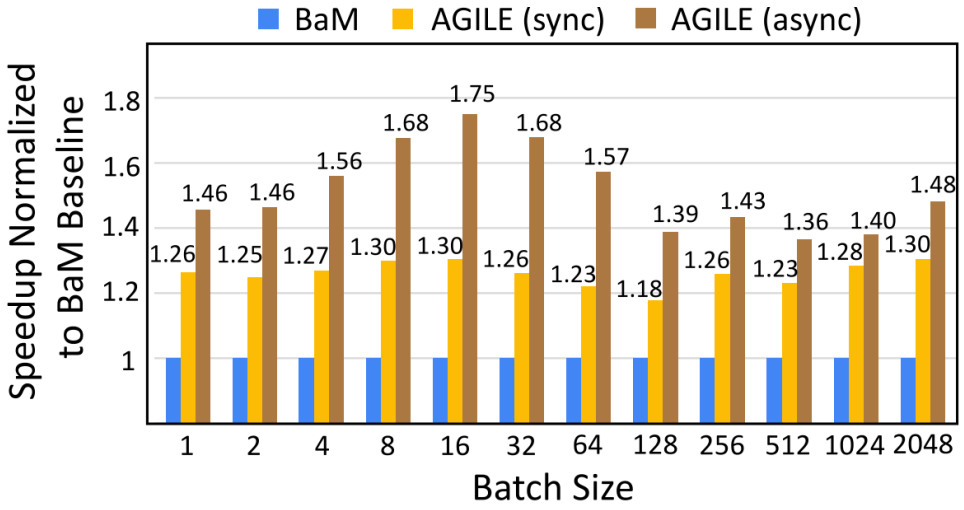



**Figure 8: Speedup comparison of AGILE (async and sync modes) and BaM across varying batch sizes in DLRM inference.**

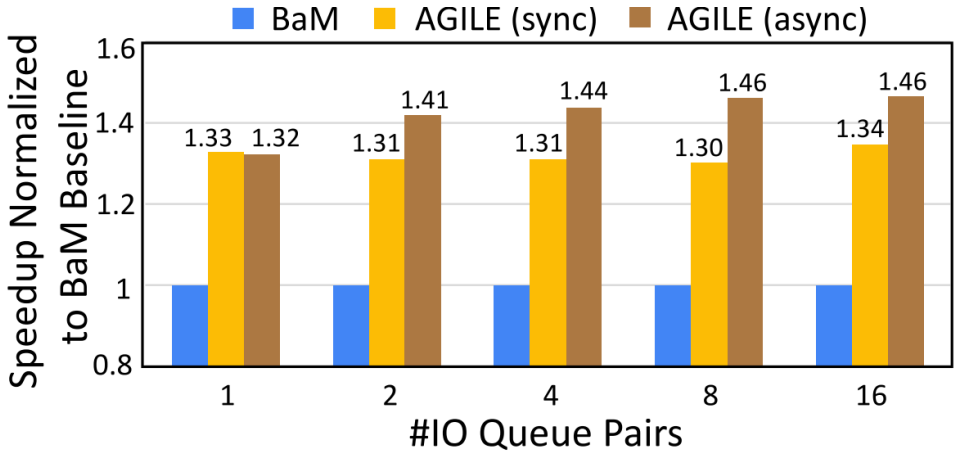



**Figure 9: Speedup comparison of AGILE (async and sync modes) and BaM under varying numbers of I/O queue pairs in DLRM inference.**


[1] Naumov, Maxim, et al. "Deep learning recommendation model for personalization and recommendation systems." *arXiv preprint arXiv:1906.00091* (2019).
[2] Criteo AI Lab. 2025. Download Criteo 1TB Click Logs dataset - Criteo AI Lab. https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/

# Experimental Evaluation

School of Engineering
BROWN UNIVERSITY

## 3. Comparison with BaM on DLRM inference MicroBenchmark.

- The DLRM model is adopted from [1] with Criteo 1TB Click Logs dataset[2].
- cuBLAS is used for all matrix multiplications.
- BaM and AGILE are used for fetching data.
- AGILE is used in both synchronous mode and asynchronous mode

Default parameters:

- Batch Size: 2048
- #I/O queues: 128
- Software cache size: 2 GB

- Sweep key parameter: Software Cache Size

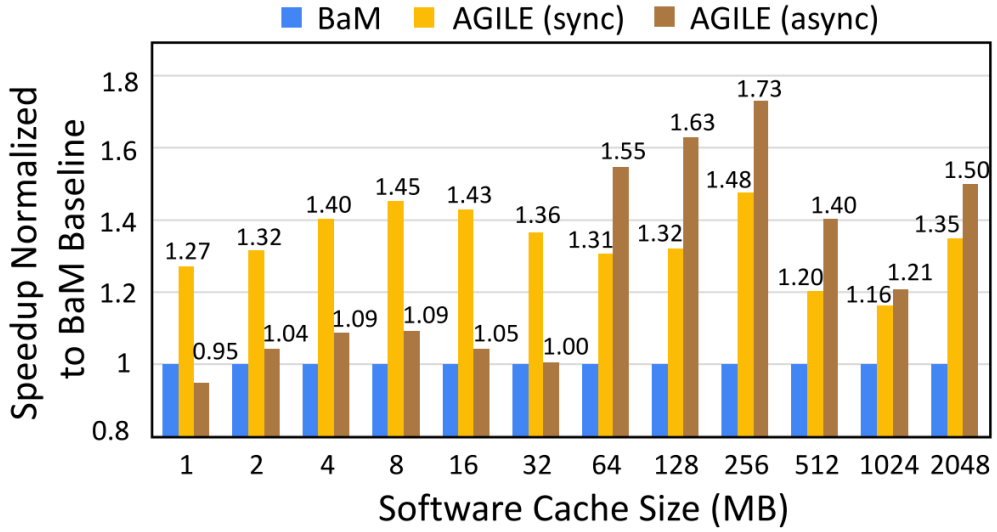



**Figure 10: Speedup comparison of AGILE (async and sync modes) and BaM under varying software cache sizes in DLRM inference.**


[1] Naumov, Maxim, et al. "Deep learning recommendation model for personalization and recommendation systems." *arXiv preprint arXiv:1906.00091* (2019).
[2] Criteo AI Lab. 2025. Download Criteo 1TB Click Logs dataset - Criteo AI Lab. https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/

# Thank You!

# Questions?

zhuoping_yang@brown.edu
peipei_zhou@brown.edu

https://peipeizhou-eecs.github.io/

https://github.com/arc-research-lab/AGILE

BROWN UNIVERSITY

Syracuse University