40th IEEE International Performance Computing and Communications Conference

Diagnosing the Interference on CPU-GPU Synchronization Caused by CPU Sharing in Multi-Tenant GPU Clouds

Presenting Author

Youssef Elmougy
Hofstra University
New York, USA
yelmougy1@pride.hofstra.edu

Weiwei Jia
Xiaoning Ding
New Jersey Institute of Technology
New Jersey, USA
wj47@njit.edu
xiaoning.ding@njit.edu

Jianchen Shan
Hofstra University
New York, USA
jianchen.shan@hofstra.edu

Building Multi-Tenant GPU Clouds is Still a Challenge

Powered by virtualization, resources, like GPUs and CPUs, can be shared among instances!

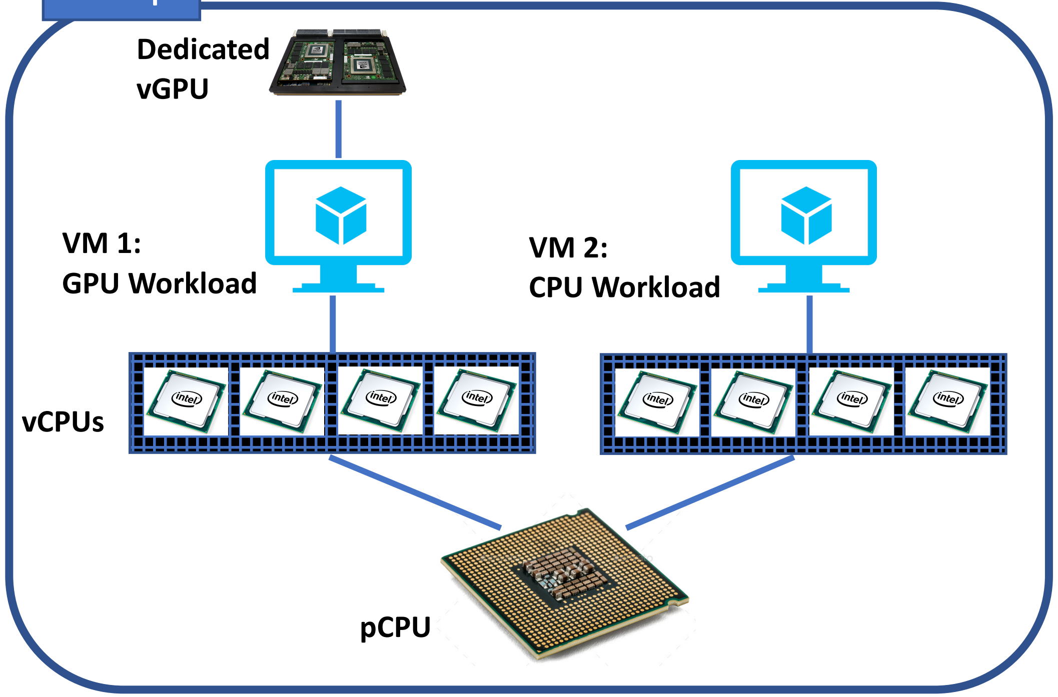
Workloads suffer from **poor performance isolation** and **low utilization** when instances **share GPUs.** Zhang et al. [TPDS '13], Qi et al. [TACO '14], Xue et al. [USENIX ATC '16], Zhang et al. [TPDS '18], Lu et al. [TPDS '19]

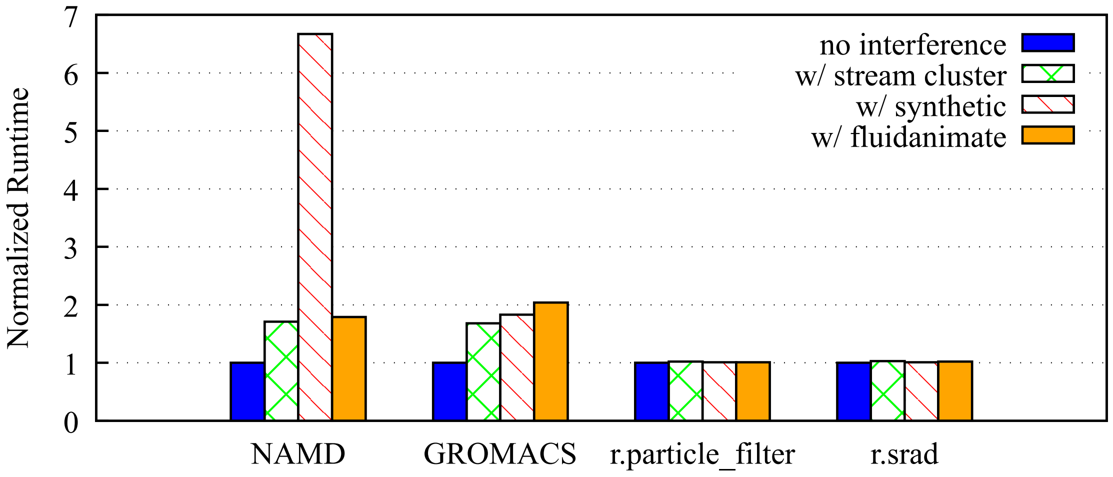
How sharing CPUs among GPU instances could affect the workload performance?

Examining Interference Caused by CPU Sharing

Workloads may suffer from **poor** and **unpredictable performance**!

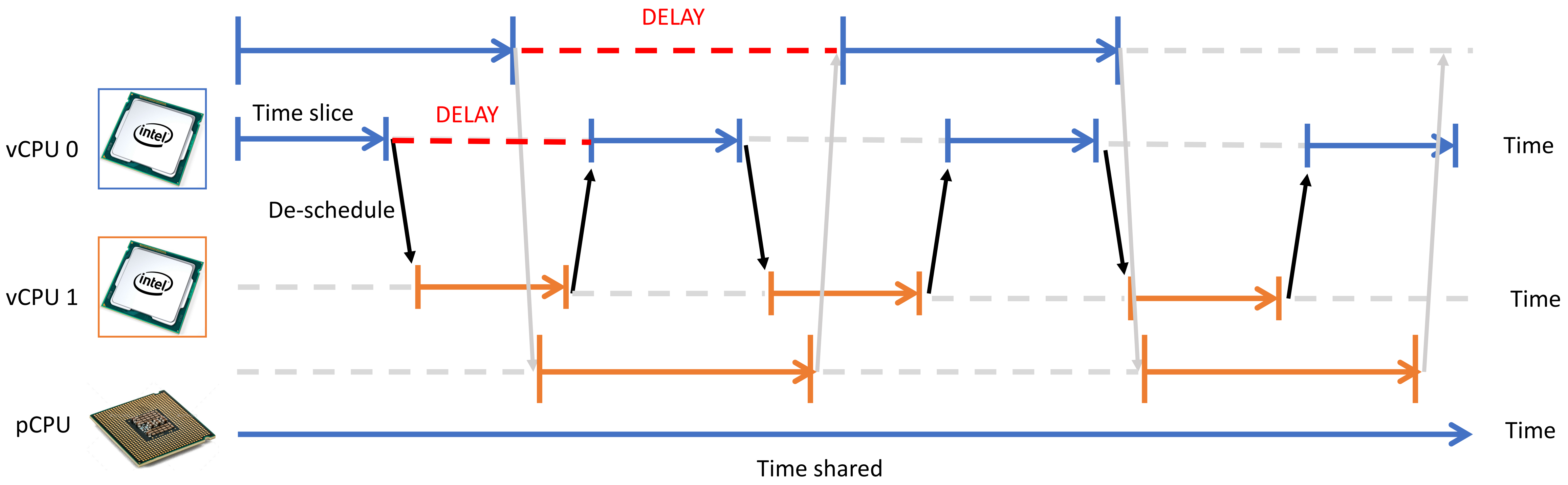
Setup





Result of complex interplay between vCPU sharing and the characteristics of the GPU workload.

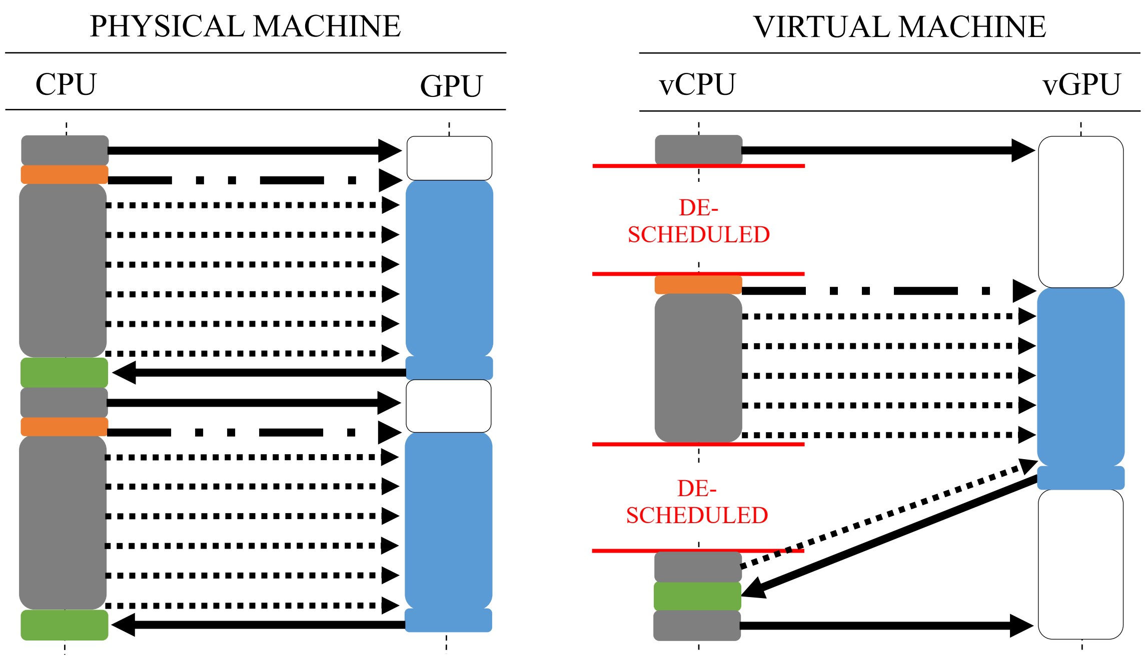
vCPU Discontinuity Caused by vCPU Sharing



How vCPU discontinuity interacts with GPU workloads remains understudied!

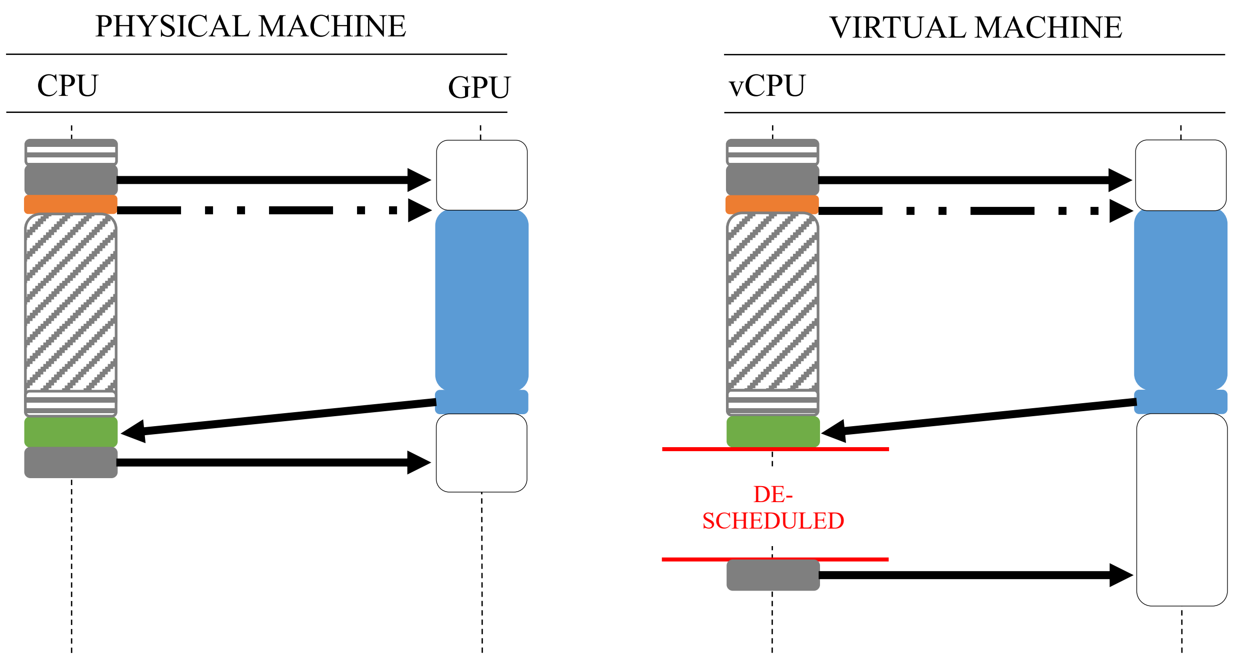
vCPU Discontinuity leads to Performance Degradation and vCPU Underutilization

Polling CPU-GPU synchronization

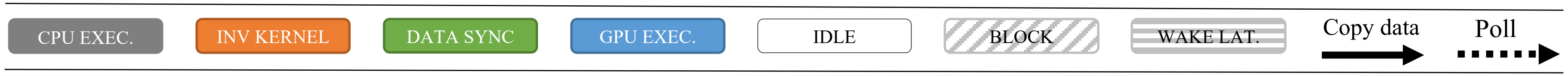


cudaEventQuery

Blocking CPU-GPU synchronization



cudaEventSynchronize

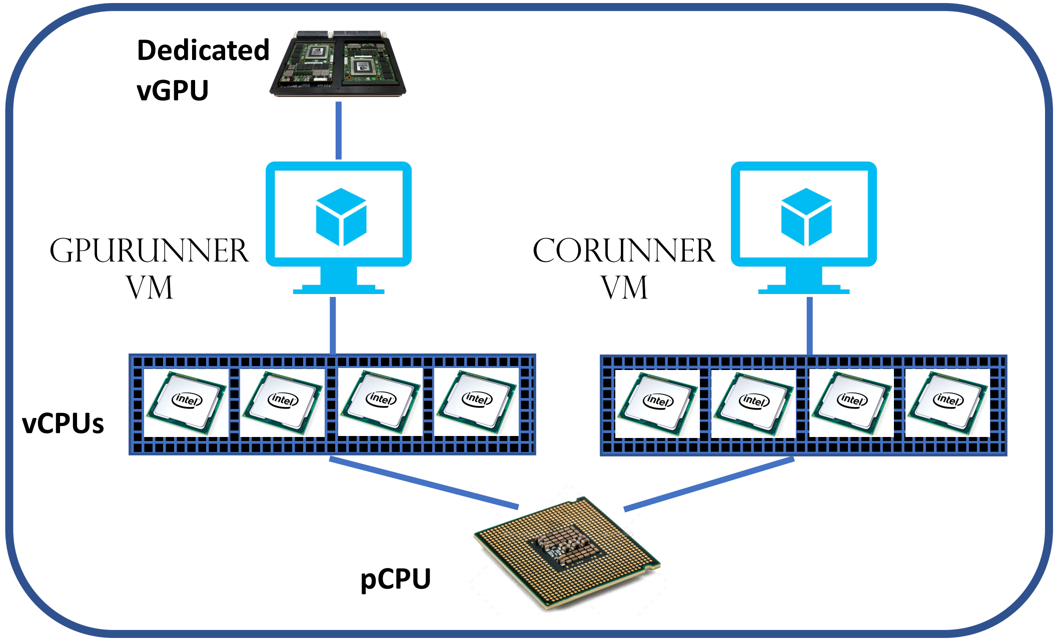


***Synchronization-intensive workloads** are most vulnerable in the GPU cloud!*

Measuring Performance Interference

Host system: **HPE ProLiant DL385Gen10 server** with **256GB memory** using **Ubuntu Server 20.04.2 (kernel 5.8.0), 4 Intel Xeon Gold 6138 20-core processors**, and **2 NVIDIA Tesla P100 GPUs**

Virtualized system: VMM is **KVM**, **Ubuntu Server 20.04.2 (kernel 5.8.0)**, 1 GPU attached using **PCI Passthrough** with driver **CUDA 10.1**



Measuring Performance Interference continued...

CPU-intensive Benchmarks:

- Synthetic program [while(1) loop]
- matmul

PARSEC
['pär-''sek] A unit of measure

[stream_cluster, dedup, x264, fluidanimate]

GPU-intensive Benchmarks:

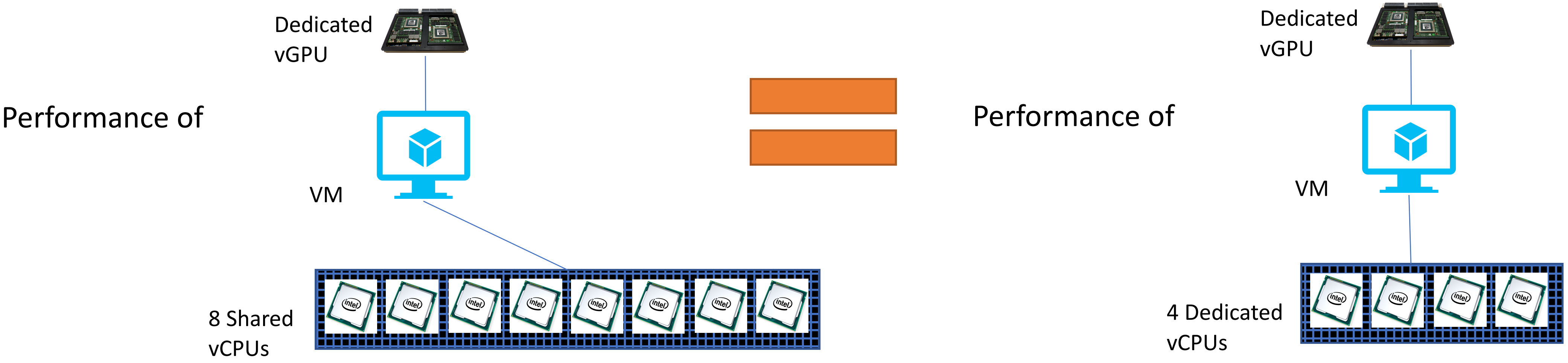
NAMD
Scalable Molecular Dynamics

GROMACS
FAST. FLEXIBLE. FREE.

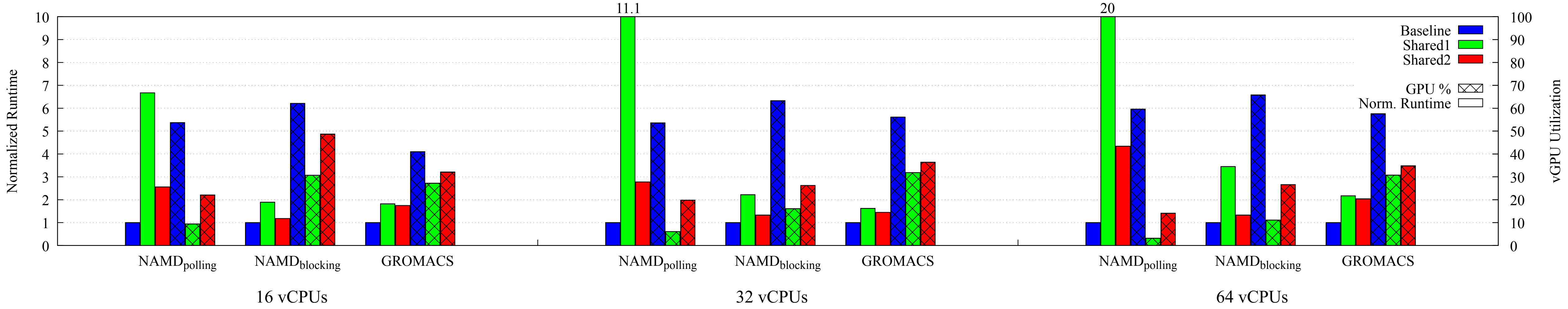
Measuring Performance Interference continued...

Experiment	corunner	GPUrunner	Perf. Iso.	#vCPUs
Shared1	✓	✓	✗	16, 32, 64
Shared2	✓	✓	✓	16, 32, 64
Baseline	✗	✓	N.A	8, 16, 32

Assuming there is no degradation issue from sharing vCPUs:

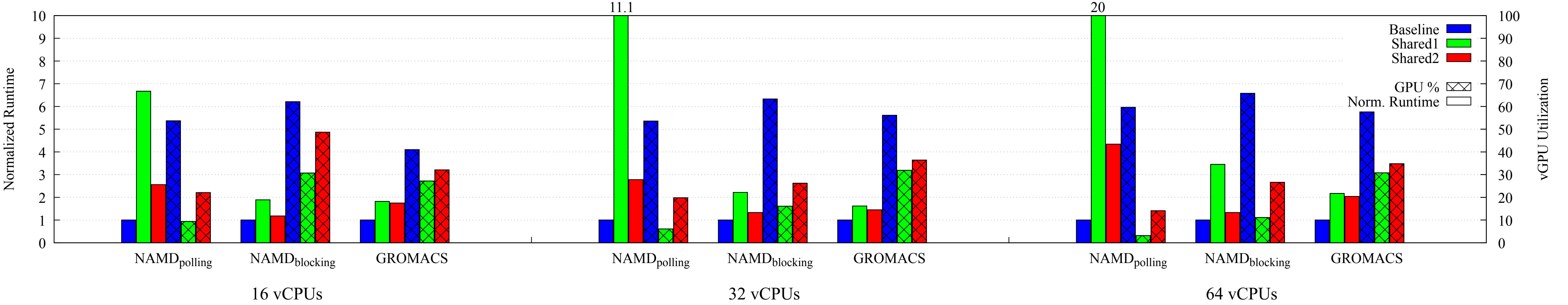


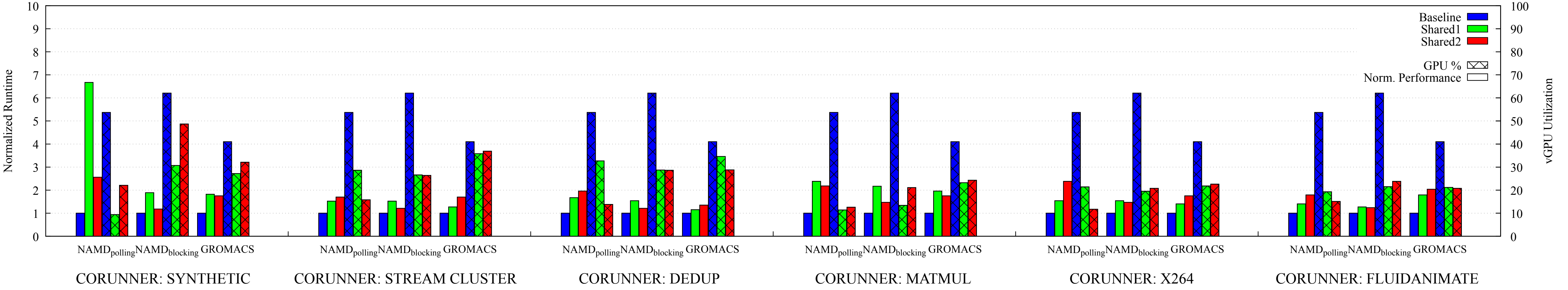
Analyzing the Performance under vCPU Discontinuity Interference



1. Degradation in Runtime Performance
2. vGPU Under-utilization
3. Increased Degradation Under Varied #vCPUs
4. Unpredictability in Performance
5. Increased Degradation Under Varied #VMs

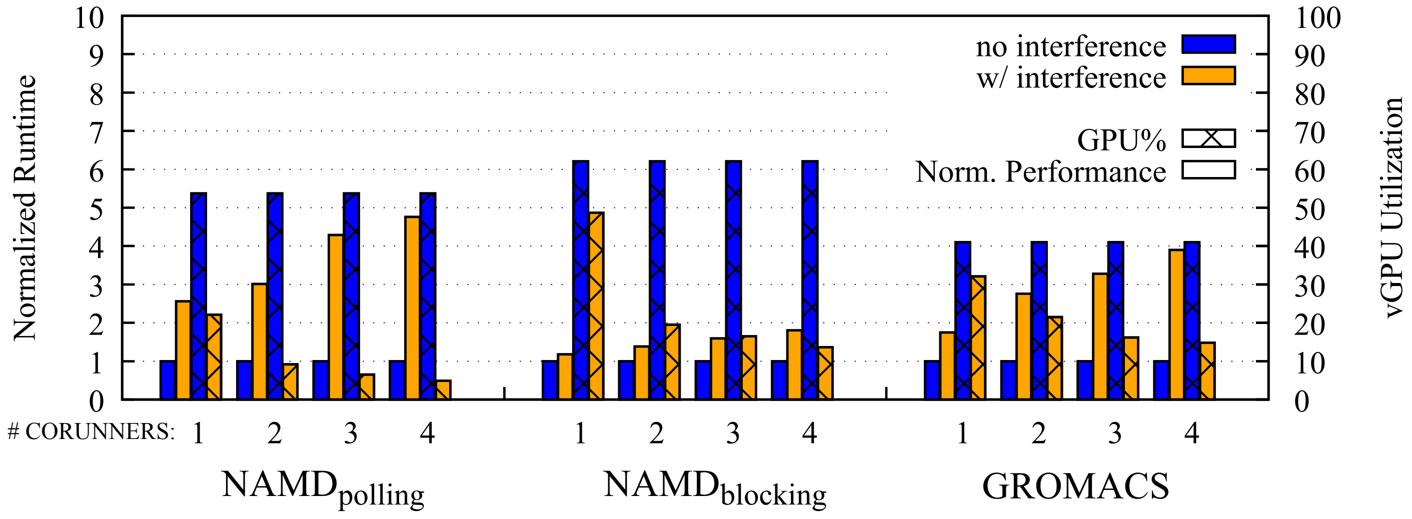
Analyzing the Performance under vCPU Discontinuity Interference





1. Degradation in Runtime Performance
2. vGPU Under-utilization
3. Increased Degradation Under Varied #vCPUs
4. **Unpredictability in Performance**
5. Increased Degradation Under Varied #VMs

Analyzing the Performance under vCPU Discontinuity Interference



1. Degradation in Runtime Performance
2. vGPU Under-utilization
3. Increased Degradation Under Varied #vCPUs
4. Unpredictability in Performance
5. Increased Degradation Under Varied #VMs

Diagnose: Case Study of NAMD

Scalable Molecular Dynamics

Fine-grained profiling done using the performance analysis tool: NVIDIA® Nsight™

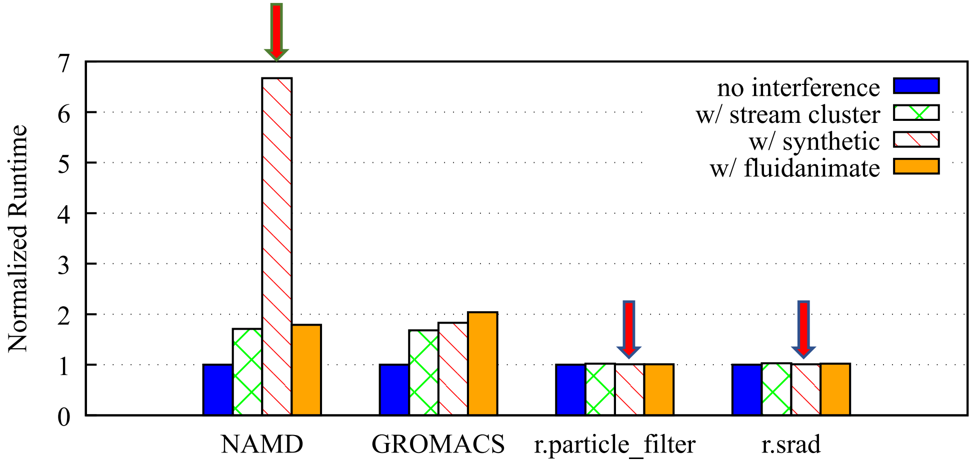
We found that the major sources of latencies are:

1. **During kernel invocation**
2. **Receiving completed kernels**

Workloads with more frequent number of kernels are more vulnerable to performance degradation!

r.particle_filter & r.srad	1 kernel / sec
NAMD	370 kernels / sec

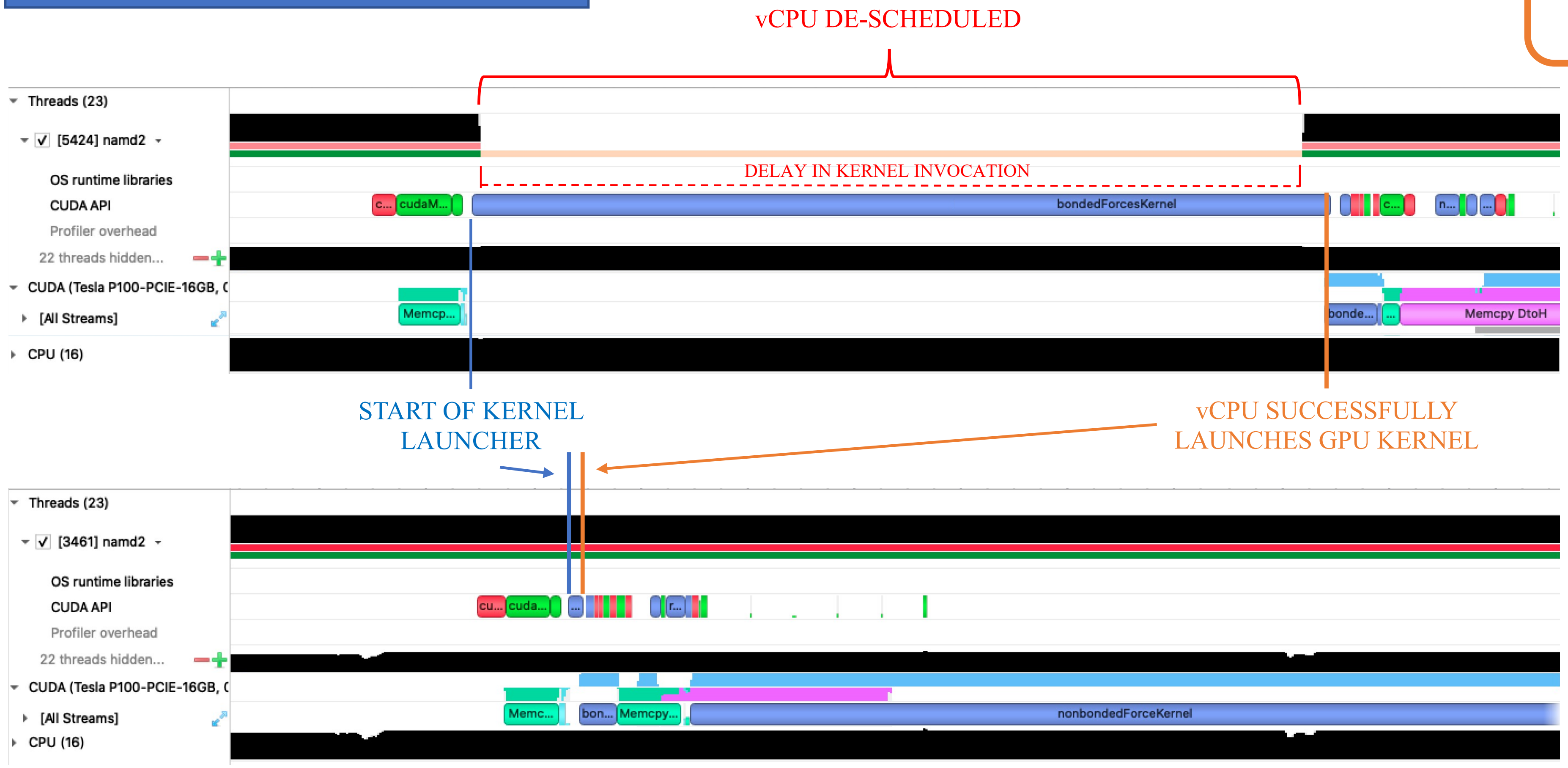
synchronization-intensive workload



Latency in Kernel Invocation

Polling CPU-GPU synchronization

Latency ≈ 29.79ms



Affected by vCPU Discontinuity:

95 kernels / sec

Normal:

171 kernels / sec

Blocking CPU-GPU synchronization

Negligible Latency.

Latency in Receiving Completed Kernels

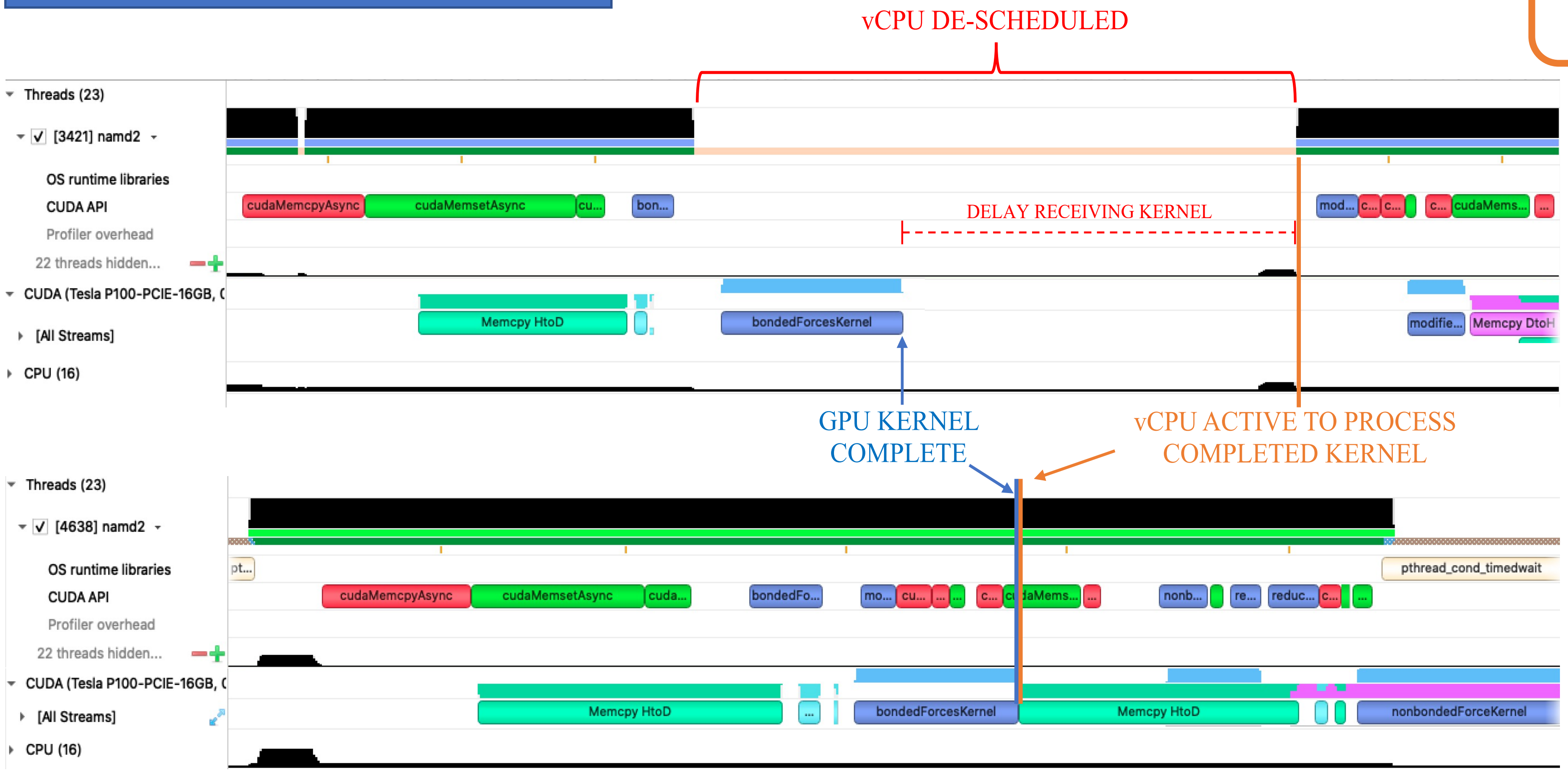
Polling CPU-GPU synchronization

Latency ≈ 28.65ms

Latency in Receiving Completed Kernels continued...

Blocking CPU-GPU synchronization

Latency ≈ 13.58ms



Affected by vCPU Discontinuity:

191 kernels / sec

Normal:

248 kernels / sec

Optimizing Performance in GPU Clouds

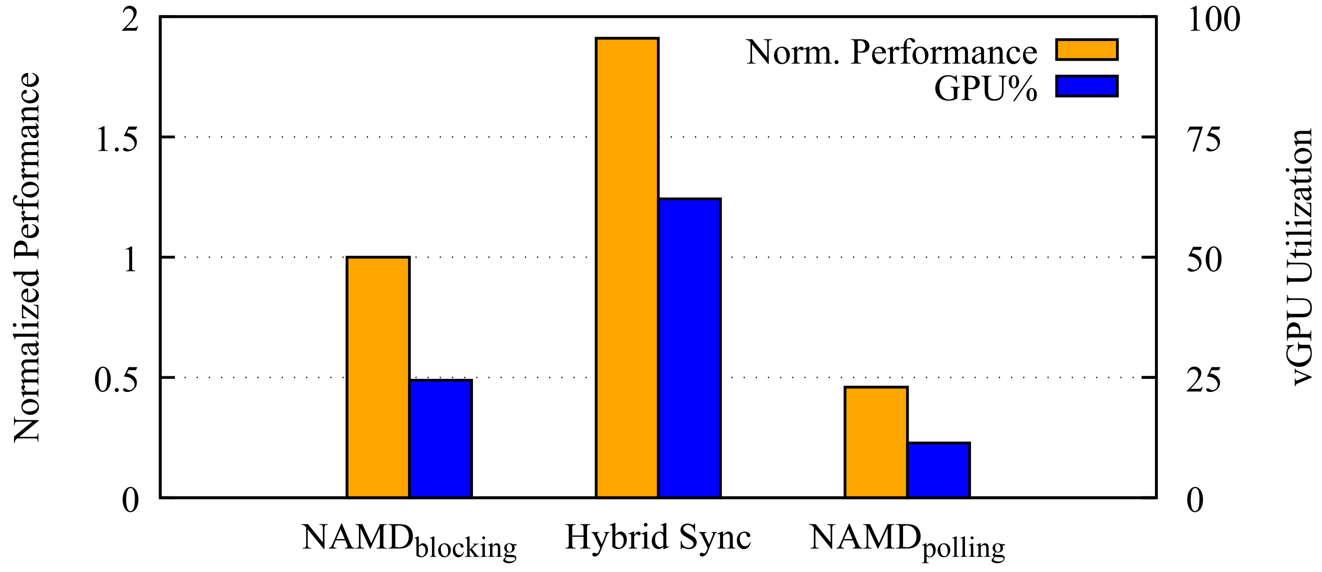
Continuous device communication substantially wastes vCPU time slice

Kernel Execution time is consistent and predictable (± **2%-5%** for NAMD)

We propose a novel
polling-then-blocking CPU-GPU synchronization primitive

This hybrid primitive reduces vCPU time slice waste and reduces kernel latency. It is portable and scalable to other applications!

Polling-then-Blocking Synchronization Results



	CPU-GPU Synchronization Techniques		
	Polling	Blocking	Hybrid
Invoking Kernel	High Latency (29.79*ms*)	Negligible Latency	Low Latency (5.42*ms*)
Receiving Kernel	High Latency (28.65*ms*)	High Latency (13.58*ms*)	Low Latency (1.49*ms*)

1 Execution Cycle ≈ 4.5ms

Conclusion

We diagnosed the effect of vCPU discontinuity on GPU workloads

Performance interference caused by vCPU discontinuity:

- Inefficient CPU-GPU synchronization
- Increases vulnerability of vCPUs to be descheduled
- Unpredictable GPU workload performance
- ↑ # shared vCPUs, ↑ # VMs $\Longrightarrow$ ↑ GPU workload degradation

We proposed a *polling-then-blocking hybrid synchronization technique*

The diagnose should provide a guideline for users and cloud providers using multi-tenant GPU cloud instances sharing CPU resources!

40th IEEE International Performance Computing and Communications Conference

Thank you for your attention!
Any questions?

Diagnosing the Interference on CPU-GPU Synchronization Caused by CPU Sharing in Multi-Tenant GPU Clouds

Youssef Elmougy
Hofstra University
New York, USA
yelmougy1@pride.hofstra.edu

Weiwei Jia
Xiaoning Ding
New Jersey Institute of Technology
New Jersey, USA
wj47@njit.edu
xiaoning.ding@njit.edu

Jianchen Shan
Hofstra University
New York, USA
jianchen.shan@hofstra.edu