

GPU CONCURRENCY

ROBERT SEARLES



EXECUTION SCHEDULING & MANAGEMENT

Pre-emptive scheduling

Processes share GPU through time-slicing Scheduling managed by system



Concurrent scheduling

Processes run on GPU simultaneously User creates & manages scheduling streams



CUDA CONCURRENCY MECHANISMS

	Streams	MPS	MIG
Partition Type	Single process	Logical	Physical
Max Partitions	Unlimited	48	7
Performance Isolation	No	By percentage	Yes
Memory Protection	No	Yes	Yes
Memory Bandwidth QoS	No	No	Yes
Error Isolation	No	No	Yes
Cross-Partition Interop	Always	IPC	Limited IPC
Reconfigure	Dynamic	Process launch	When idle

MPS: Multi-Process Service MIG: Multi-Instance GPU

CUDA STREAMS

STREAM SEMANTICS

- 1. Two operations issued into the same stream will *execute in issueorder*. Operation B issued after Operation A will not begin to execute until Operation A has completed.
- 2. Two operations issued into separate streams have *no ordering prescribed by CUDA*. Operation A issued into stream 1 may execute before, during, or after Operation B issued into stream 2.
- Operation: Usually, cudaMemcpyAsync or a kernel call. More generally, most CUDA API calls that take a stream parameter, as well as stream callbacks.

STREAM EXAMPLES

Host/Device execution concurrency:

Kernel<<<b, t>>>(...); // this kernel execution can overlap with
cpuFunction(...); // this host code

Concurrent kernels:

Kernel<<<b, t, 0, streamA>>>(...); // these kernels have the possibility
Kernel<<<b, t, 0, streamB>>>(...); // to execute concurrently

- In practice, concurrent kernel execution on the same device is hard to witness
- Requires kernels with relatively low resource utilization and relatively long execution time
- There are hardware limits to the number of concurrent kernels per device
- Less efficient than saturating the device with a single kernel

INTRODUCTION TO CUDA's MULTI-PROCESS SERVICE (MPS)

MPI DECOMPOSITION

Spread work across GPUs

Very common in HPC

Designed to **concurrently** execute multiple MPI ranks on different GPUs

Used when there is **too much work** for a single GPU.



MULTI-PROCESS SERVICE (MPS) OVERVIEW

First available on Kepler architecture (SM 3.5 - K20/K40/K80)

Significantly enhanced on Volta (SM 7.0)

Designed to **concurrently** map multiple MPI ranks onto a single GPU

Used when each rank is **too small** to fill the GPU on its own

On Summit, use *-alloc_flags=gpumps* when submitting a job with *bsub*



PROCESSES SHARING GPU WITHOUT MPS

No Overlap





PROCESSES SHARING GPU WITHOUT MPS

Additional small overhead arising from pre-emptive context switch



PROCESSES SHARING GPU WITH MPS

Maximum Overlap



roject Explorer	× default.qdi	rep 🗙 mps.qdrep 🗙							
default.qdre	ep 📑 Time	line View 🔹				₽ 1x □	u 18	-	\land 3 warnings, 22 messag
		15	+108ms		+108.2ms	+108.4m	s	+108.6	6ms +108.8ms
	* In	reads (10)							
		🖊 [61852] MPI Rank 0 👻							
		MPI							1
		CUDA API		cudaD	eviceSynchronize				cudaDeviceSynchronize
		Profiler overhead							
		/ [61877] mps demo -							
								~	
	100.0%	Kernels	m dum dum dum.	. d'n	dum dum I d	lum dum	dum dun	n dı	um dum dum dum
	100.0761	Remeis du	m) aum) aum) aum	(aum)		aum) aum)	dum. dur	<u>n (</u> a	ium(aum(aum)
	Events Vi	iew 🔻							
						A			
	Literio i						1	Searc	-h
	#	Name	 Duration 	TID	GPU	Context	Start	Searc	ih
	#	Name dummy_k_hel	 Duration 69.856 μs 	TID -	GPU GPU 0	Context Stream 7	Start 1.10779s	Searc	h dummy_kernel Begins: 1.108s
	# 1 2	Name dummy_kr_nel dummy_mnel	 Duration 69.856 μs 68.767 μs 	TID - -	GPU GPU 0 GPU 0	Context Stream 7 Stream 7	Start 1.10779s 1.10786s	Searc	 dummy_kernel Begins: 1.108s Ends: 1.10807s (+69.215 μ s)
	# 1 2 3	Name dummy_kr_tel dummy_trnel dummy_kernel	 Duration 69.856 μs 68.767 μs 69.248 μs 	TID - -	GPU GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s	Searc	h dummy.kernel Begins: 1.108s Ends: 1.10807s (+69.215 μ s) gridt. <<<80, 1, 1>>> black <<<80, 1, 1>>>
	# 1 2 3 4	Name dummy_kr_nel dummy_rmel dummy_krmel dumm_krmel	 Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 	TID - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.10793s	Searc	h dummy_kernel Begins: 1.108s Ends: 1.10807s (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> Launch Type: Regular
	# 1 2 3 4 5	Name dummy_kr_nel dummy_rnel dummy_krnel dum_y_krnel dum_y_krnel	 Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 	TID - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.10793s 1.1080s	Searc	h dummy_kernel Begins: 1.1080s Ends: 1.10807s (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<80, 1>> block: <<<80, 1, 1>>> block: <<<80, 1>> block: <<80, 1>> block: <<80, 1>> block: <<80, 1>> block: <<80, 1>> block: <<80,
	# 1 2 3 4 5 6	Name dummy_kr.nel dummy_rnel dummy_krnel dum_y_krnel dum_y_krnel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.923 μs	TID - - - - - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.10793s 1.1080 1.10807s 1.10814s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<80, 1>>> block: <<<80, 1, 1>>> block: <<<80, 1>> block: < 80, 1 > block: <<<80, 1>> block: <<<80, 1>> block: <<<80, 1>> block: <<<80, 1>> block: <<<80, 1>> block: <<80, 1>> block: <<<80, 1>> block: <<80, 1>>
	# 1 2 3 4 5 6 7	Name dummy_kr.nel dummy_rmel dummy_krnel dum_y_krnel dum_y_krnel cmmy_krnel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.023 μs 69.023 μs 69.567 μs	TID - - - - - - - - - -	GPU GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.1080 1.10807s 1.10814s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<1024, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 b ytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 16
	# 1 2 3 4 5 6 7 7	Name dummy_kr.nel dummy_rmel dummy_krnel dum_y_krnel dum_y_krnel cmmy_krnel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.472 μs 69.023 μs 69.567 μs	TID - - - - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.10793s 1.10807s 1.10807s 1.10814s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<1024, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 b ytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 16
_	# 1 2 3 4 5 6 7 7 *	Name dummy_kr.nel dummy_kr.nel dummy_kr.nel dum_y_kr.nel dum_y_kr.nel f.mmy_kr.nel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.472 μs 69.023 μs 69.567 μs	TID - - - - - - - - - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10786s 1.10793s 1.108% 1.10807s 1.10814s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<1024, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 b ytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 16
_	# 1 2 3 4 5 6 7 7	Name dummy_kr.nel dummy_kr.nel dummy_kr.nel dum_y_kr.nel dum_y_kr.nel 	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.023 μs 69.023 μs 69.567 μs	TID - - - - - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.1088 1.10807s 1.10807s 1.10814s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block: <<<1024, 1, 1>>> block: <<<1024, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 b ytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 16
	# 1 2 3 4 5 6 7 7	Name dummy_krinel dummy_krinel dummy_krinel dum_ykrinel dum_ykrinel fimmy_krinel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.472 μs 69.023 μs 69.567 μs	TID - - - - - - - - - - - - -	GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.107935 1.10807s 1.10807s 1.10807s 1.10814s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: <<<80, 1, 1>>> block <<<1024, 1, 1>>> block <<1024, 1, 1>> block 1024, 1, 1 > block <<1024, 1, 1>> block 1024, 1 > block
	* 1 2 3 4 5 6 7 *	Name dummy_kriel dummy_rmel dummy_rmel dum_krenel dum_ykrenel immy_krenel immy_krenel	Duration 69.856 μs 68.767 μs 69.248 μs 69.215 μs 69.472 μs 69.472 μs 69.023 μs 69.567 μs		GPU GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Start 1.10779s 1.10786s 1.10793s 1.10807s 1.10814s 1.10821s 1.10821s	Searc	h dummy_kernel Begins: 1.10805 Ends: 1.108075 (+69.215 μ s) grid: «<<80, 1, 1>>> block <<<1024, 1, 1>>> block <<41024, 1, 1>>> block <<1024, 1, 1>>> block <<105 hared Memory: 0 ytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 16

PROCESSES SHARING GPU WITH MPS

No Context Switch Overhead

ect Explorer X	default.go	drep X mps.qdrep X											
default.qdrep mps.qdrep	E Time	eline View 🔹				<i>₽</i> 1x □	с с т	A 3 warnings, 22	messages				
	- T	hreads (10)	+107.9ms	+108ms	+108.1	ns		15	+108.06ms	+108.07ms	+108.08ms	+108.09ms	+108.1ms +1
		niedus (10)					8 threads hid	lden 💻 🕂					4
		✓ [61852] MPI Rank 0 -				- (UDA (MPI Ra	nk 0)					
		MPI				-	100.0% Kern	els 📌	dummy_kern	el		dummy_kernel	
		CUDA API			C	udaDe	▼ 100.0% du	mmy_kernel	dummy_kern	el		dummy_kernel	
		Profiler overhead					100.0% c	lummy_kernel(dummy_kern	el		dummy_kernel	
		✓ [61877] mps demo -				•0	[61853] ./mps	_demo					
				_		- 1	hreads (10)						
	100.0%	Kernels 📌 du	im dummy_ker dummy	kerdumr	ny_ker dummy	_ker		ST3157					
	100.0%	Kernels 🖉 du	um dummy_ker dummy	kerdum	my_ker dumm	100.0%	✓ [61853] N	1PI Rank 1 🔻				dimensional language	
	Events	lieu 🔻				100.07	o Kerneis	4	dummy_ke	nei		dummy_kernel	
	Lycits v	incw .					6						
	-	Name	* Duration	TID	CDU	Contract	Start						
	1	dummy kernel	69.856 µs	-	GPU 0	Stream 7	1.10779s	Begins: 1.108s					
	2	dummy_kernel	68.767 μs	-	GPU 0	Stream 7	1.10786s	Ends: 1.10807s (+69.2	215 μ				
	3	dummy_kernel	69.248 μs	-	GPU 0	Stream 7	1.10793s	grid: <<<80, 1, 1>>	>				
	4	dummy_kernel	69.215 μs		GPU 0	Stream 7	1.108s	block: <<<1024, 1, 1 Launch Type: Regular	>>> —				
	5	dummy_kernel	69.472 μs	-	GPU 0	Stream 7	1.10807s	Static Shared Memor	y: 0 b				
	6	dummy_kernel	69.023 μs	-	GPU 0	Stream 7	1.10814s	Dynamic Shared Mer	mory:				
	7	dummy_kernel	69.567 μs	-	GPU 0	Stream 7	1.10821s	0 bytes Registers Per Thread:	16 -				
							4 40000						

WITHOUT vs. WITH MPS, SIDE-BY-SIDE

MPS enables inter-process concurrency on the GPU



MPS makes processes concurrent on the GPU

CUDA 11 Multi-Instance GPU (MIG)

GPU ARCHITECTURE AND CUDA



INTRODUCING NVIDIA A100



Ampere World's Largest 7nm chip 54B XTORS, HBM2



3rd Gen Tensor Cores Faster, Flexible, Easier to use 20x Al Perf with TF32



New Sparsity Acceleration Harness Sparsity in Al Models 2x Al Performance





3rd Gen NVLINK and NVSWITCH Efficient Scaling to Enable Super GPU 2X More Bandwidth

NVIDIA A100: SPECS

	V100	A100
SMs	80	108
Tensor Core Precision	FP16	FP64, TF32, BF16, FP16, I8, I4, B1
Shared Memory per Block	96 kB	160 kB
L2 Cache Size	6144 kB	40960 kB
Memory Bandwidth	900 GB/sec	1555 GB/sec
NVLink Interconnect	300 GB/sec	600 GB/sec
FP64 Throughput		9.7 19.5 TFLOPS
TF32 Tensor Core	N/A	156 312 TFLOPS



MULTI-INSTANCE GPU (MIG)

Optimize GPU Utilization, Expand Access to More Users with Guaranteed Quality of Service



Up To 7 GPU Instances In a Single A100:

Dedicated SM, Memory, L2 cache, Bandwidth for hardware QoS & isolation

Simultaneous Workload Execution With Guaranteed Quality Of Service:

All MIG instances run in parallel with predictable throughput & latency

Right Sized GPU Allocation:

Different sized MIG instances based on target workloads

Diverse Deployment Environments:

Supported with Bare metal, Docker, Kubernetes, Virtualized Env.



MIG User Guide: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html

EXAMPLE: TWO LEVEL PARTITIONING

A100-40GB - Up to 7 GPUs



MIG ISOLATION

- Computational Isolation
 - Streaming Multiprocessors (SMs) are <u>not</u> shared between GPU Instances
 - This provides high Quality of Service (QoS) for each GPU Instance
- DRAM Bandwidth Isolation
 - Slices of the L2 Cache are physically associated with particular DRAM channels and memory
 - Isolating MIGs to non-overlapping sets of L2 Cache slices does two things:
 - Isolates BW
 - Allocates DRAM memory between the MIGs
- Configuration Isolation
 - Creating GPU Instances or Compute Instances do not disturb work running on existing instances
- Error Isolation
 - Resources within the chip are separately resettable

LOGICAL VS. PHYSICAL PARTITIONING



Multi-Process Service Dynamic contention for GPU resources Single tenant



Multi-Instance GPU

Hierarchy of instances with guaranteed resource allocation Multiple tenants

26 📀 NVIDIA.

CUDA BEHAVIOR AND DEVELOPER TOOLS

Can CUDA code continue to run on a separate part of the chip when we re-configure the chip?

- Yes, as long as you don't reconfigure the partition the code is running on
- All CUDA code will have to stop to enable/disable MIG mode

What is the difference between "compute instance" and "GPU instance"?

GPU instances can be further sub-divided into compute instances (one per slice or GPC) that share memory. It is a more advanced use-case.

What is difference between CUDA streams, MPS, and MIG?

MPS does not HW partition the processes

- MIG partitions jobs each with dedicated compute, memory, and memory bandwidth
- CUDA stream: Per process, no hardware isolation

Will MIG replace MPS

MPS is a feature of CUDA and will continue as a feature with roadmap, it is not meant to be a "replacement for" MPS or fixing any MPS bug.

FUTURE MODULES

Multi-Threading + CUDA Streams

MPI/MPS + CUDA

- Debugging Tools
 - Compute sanitizer & cuda-gdb



