An HPC Perspective for Femtoscale Imaging of Nuclei using Exascale Platforms

## About me

- I am a PhD student at Virginia Tech

   My primary field is HPC
   I have no background in Physics, much less QP, HEP, NP
- My work is primarily developing a performant theory module
  - ofitpack\_cpp Low latency, differentiable theory\* module, currently in production
  - $\odot \text{GPU}$  version of this theory
- I have also played around with • A PDF level experimental module • Using generative models for parameterization of distributions

## Disclaimer

- I have no background in Physics.
- I present a lot of "napkin math."

 Doing exhaustive runs for everything would empty our supercomputing allocation. An HPC Perspective for Femtoscale Imaging of Nuclei using Exascale Platforms

#### IDIS with 1-D QCFs in Mellin Space ("STATS-2")



**K Bootstraps** 

# VT-Argonne Computational Experiments ("Stats-2")

- PDF-Event Loss
- 5,000 experimental events for p and n targets
- Computational Run on Virginia Tech Computing Clusters: ONodes: 2
  - Tinkercliffs@VTCPU: AMD 7702x2 (128 Cores)
  - Infer@VTGPU: Nvidia V100
  - oBootstraps: 2,560
  - Theory: fitpack\_cpp Low latency, differentiable theory\* module, currently in production

#### Profiling & Projecting Performance of IDIS with 1-D QCFs in Mellin Space



#### Profiling & Projecting Performance of IDIS with 1-D QCFs in Mellin Space



Version	Speedup for cross-section calculation	Execution time (Hours)
Numpy	1.00	257.09
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13

Version	Speedup for cross-section calculation	Execution time (Hours)
Numpy	1.00	257.09
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13

Version	Speedup for cross-section calculation	Execution time (Hours)
Numpy	1.00	257.09
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13
C++ CPU (fitpack_cpp)		
Pytorch Vectorized (GPU)		

Version	Speedup for cross-section calculation	Execution time (Hours)	
Numpy	1.00	257.09	
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13	Pytorch GPU
C++ CPU (fitpack_cpp)	15.83		— 54% faster than optimized CPU
PyTorch Vectorized (GPU)	24.49		VEISION!!!

Version	Speedup for cross-section calculation	Execution time (Hours)	
Numpy	1.00	257.09	
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13	Pytorch GPU
C++ CPU (fitpack_cpp)	15.83	17.70	— 54% faster than optimized CPU
PyTorch Vectorized (GPU)	24.49	11.99	VEI SIUTI:::

	Version	Speedup for cross-section calculation	Execution time (Hours)	
	Numpy	1.00	257.09	
	Numpy vectorized*(~ Pvtorch CPU)	7.39	36.13	Pytorch GPU
1		15.83	17.70	— 54% faster than optimized CPU
		EDEE III	11.99	VCI 3IOII
7		TINEE LU		

Version	Speedup for cross-section calculation	Execution time (Hours)
Numpy	1.00	257.09
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13
C++ CPU (fitpack_cpp)	15.83	17.70
PyTorch Vectorized (GPU)	24.49	11.99

## **HPC Computational Resources**





CPU Node

**GPU** Node

## HPC Computational Resources

- We can execute 128 different ensembles per node for a CPU oriented implementation.
- If we assume we have 6 GPUs per node like DOE ORNL's Summit supercomputer, we can find the best count for number of ensembles per GPU.

#### Parallel Efficiency of Stats-2 (C++ CPU)



Number of Threads	Parallel Speedup
16	16
32	33.28
64	56.96
128	99.84

#### Parallel Efficiency of Stats-2 (Pytorch Vectorized GPU)



## **HPC Computational Resources**

- Even with overhead due to process-level parallelization, we still effectively perform 100 units of work on CPUs.
- Multi-GPU Nodes perform 6 units of work per GPU, but 4.16X faster

## **HPC Computational Resources**

- Even with overhead due to process-level parallelization, we still effectively perform 100 units of work on CPUs.
- Multi-GPU Nodes perform 6 units of work per GPU, but 4.16X faster

#### CPU version is 4 times faster than PyTorch Vectorized GPU

# The Memory Cost of Vectorization 10K evaluations

Library	Total Memory Used	Total Number of allocations
Numpy	122.559KB	53
Numpy Vectorized	748.081MB	82
Pytorch	1.323MB	78
Pytorch Vectorized	893.541MB	307

# The Memory Cost of Vectorization 10K evalutations

Library	Total Memory Used	Total Number of allocations
Numpy	122.559KB	53
Numpy Vectorized	748.081MB	82
Pytorch	1.323MB	78
Pytorch Vectorized	893.541MB	307
	6000 times	s more memory

Version	Speedup for cross-section calculation	Execution time (Hours)	
Numpy	1.00	257.09	
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13	
C++ CPU (fitpack_cpp)	15.83	17.70	
PyTorch Vectorized (GPU)	24.49	11.99	~ 66.12 Hours

Version	Speedup for cross-section calculation	Execution time (Hours)
Numpy	1.00	257.09
Numpy vectorized*(~ Pytorch CPU)	7.39	36.13
C++ CPU (fitpack_cpp)	15.83	17.70
PyTorch Vectorized (GPU)	24.49	11.99
CUDA GPU (fitpack_cpp)	Not as high as 97.50 expected speedup?	5.25 <b>~3.3x faster</b>

## Speedup of GPU Theory



## Speedup of GPU fitpack



We can either run multiple ranks per GPU like we did for Pytorch GPU

## Speedup of GPU fitpack



#### So what goes into the Salad?



## Memory Hierarchy





## Memory Hierarchy





#### Caches are everywhere!

## Compute Hierarchy (CPU)

Zenð	L2	32M L3	L2	Zen3	Zen3	12		L2	Zen3
Zen3	L2		12	Zen3	Zen3	12	32M L3	L2	Zen3
Zen3	L2		12	Zen3	Zen3	12		L2	Zen3
Zen3	L2		12	Zen3	Zen3	12	_	12	Zen3
Zen3	L2		12	Zen3	Zen3	L2	-	L2	Zeni
Zen3	L2	-	12	Zen3	Zen3	12	32M L3	12	Zen
Zen3	L2	13	L2	Zen3	Zen3	12		L2	Zen3
Zen3	L2		L2	Zen3	Zen3	12		12	Zen3
AMD: Proc	Secure essor		DDR4 Com	Memory trollers	Ser Control	ver ler Hub		PCIe SAT	3/4 A3
AMD: Proc	Secure essor		DDR4 Com	Memory trollers	Ser Control	ver ler Hub		PCIe SAT	3/4 A3
AMD Proc	Secure essor L2		DDR4 Com	Memory trollers Zen3	Ser Control Zen3	ver ler Hub L2		PCIe SAT	3/4 A3 Zen3
AMD Proc	Secure essor L2 L2		DDR4 Cont	Memory trollers Zen3 Zen3	Ser Control Zen3 Zen3	ver ler Hub L2 L2		PCIe SAT	3/4 A3 Zen3 Zen3
AMD Proc Zen3 Zen3 Zen3	Secure essor L2 L2 L2	32M L3	DDR4 Com	Memory trollers Zen3 Zen3 Zen3	Ser Control Zen3 Zen3 Zen3	ver ler Hub L2 L2 L2	32M L3	PClei SAT	3/4 A3 Zen3 Zen3 Zen3
AMD Proc Zen3 Zen3 Zen3 Zen3	Secure essor L2 L2 L2 L2	32M L3	DDR4 Com L2 L2 L2 L2 L2	Memory trollers Zen3 Zen3 Zen3 Zen3	Ser Control Zen3 Zen3 Zen3 Zen3	ver ler Hub L2 L2 L2 L2 L2	32M L3	PClei SAT	3/4 A3 Zen3 Zen3 Zen3 Zen3
AMD Proc Zen3 Zen3 Zen3 Zen3 Zen3	Escure essor 12 12 12 12	32M L3	DDR4 Com 12 12 12 12	Memory trollers Zen3 Zen3 Zen3 Zen3 Zen3	Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	ver ler Hub 12 12 12	32M L3	PClei SAT	3/4 A3 Zen3 Zen3 Zen3 Zen3
AMD Proc Zen3 Zen3 Zen3 Zen3 Zen3	Escure essor 12 12 12 12 12 12	32M L3	DDR4 Com 12 12 12 12	Memory trollers Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	Zen3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	ver ler Hub L2 L2 L2 L2 L2 L2 L2 L2 L2	32M L3	PClei SAT	3/4 A3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3
AMD Proc Zen3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	Secure essor 12 12 12 12 12 12 12 12	32M L3 32M L3	DDR4 Com 12 12 12 12 12 12	Memory trollers Zen3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	Ser Control Zen3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3	ver ler Hub	32M L3 32M L3	PClei SAT	3/4 A3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3 Zen3

## Compute Hierarchy (GPU)

SM													
L1 Instruction Cache													
	_	LO	Instructio	n Cache			זר			LO	Instructi	on C	ache
	Warn Scheduler (32 thread/clk)							Warn Scheduler (32 thread/clk)					
Dispatch Unit (32 thread/clk)								Dispatch Unit (32 thread/clk)					
	Dispatch Unit (32 thread/clk)							Dispatch Onit (52 threadreik)					
	Register File (16,384 x 32-bit) Register File (16,384 x 32-bit)												
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					IN132	FP32	FP32	FP64	-	
INT32	EP32	EP12	ED64					INT32	EP12	EP32	EP64	-	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	EP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64	TE	INSO	R CORE		INT32	FP32	FP32	FP64	1	TENSOR CORE
INT32	FP32	FP32	FP64	4 <sup>th</sup>	GEN	RATION		INT32	FP32	FP32	FP64	1	4 <sup>th</sup> GENERATION
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
IN132	FP32	FP32	FP64					IN132	FP32	FP32	FP64	_	
INT32	EP32	FP32	ED64					INT32	EP32	EP32	ED64		
1.01			101		1.01		11	1.01			101		
ST	ST S	ST ST	ST S	ST ST	ST	SFU		ST	ST	ST ST	ST	ST	ST ST SFU
	L0 Instruction Cache     L0 Instruction Cache       Warp Scheduler (32 thread/clk)     Warp Scheduler (32 thread/clk)       Dispatch Unit (32 thread/clk)     Dispatch Unit (32 thread/clk)       Register File (16.384 x 32-bit)     Register File (16.384 x 32-bit)								ache hread/clk) read/clk) 4 x 32-bit)				
INT32	EP32	EP32	EDRA					INT32	EP32	EP32	EP64		1
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64		TENSOR CORE 4 <sup>th</sup> GENERATION			INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		TENSOR CORE
INT32	EP32	FP32	EP64	th				INT32	EP32	EP32	FP64		Ath CENERATION
INT32	FP32	FP32	EP64	4				INT32	FP32	FP32	EP64		4 GENERATION
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	l.	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64		
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
INT32	FP32	FP32	FP64					INT32	FP32	FP32	FP64	1	
LD/ ST	LD/ L ST S	LD/ LD/ ST ST	LD/ L ST 1	.D/ LD/ ST ST	LD/ ST	SFU		LD/ ST	LD/ ST	LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ LD/ ST ST SFU
	Tensor Memory Accelerator												
	256 KB L1 Data Cache / Shared Memory												
	Tex Tex Tex Tex												

## Compute Hierarchy (GPU)

<u> </u>									
				L1 Instru	ction Cache				
		L0	Instruction (	Cache	LOI	Instruction	Cache		
		Warp Sc	heduler (32	thread/clk)	Warp Scheduler (32 thread/clk)				
		Dispate	ch Unit (32 tl	nread/clk)	Dispatch Unit (32 thread/clk)				
	F	Registe	r File (16,38	4 x 32-bit)	Register	r File (16,3	34 x 32-bit)		
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64	1		
INT32	FP32	FP32	FP64	1 1	INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64	1	INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64	-		
INT32	FP32	EP32	FP64	TENSOR CORE	IN132 FP32 FP32	FP64	TENSOR CORE		
INT32	FP32	FP32	FP64	Ath GENERATION	INT32 FP32 FP32	FP64	4 <sup>th</sup> GENERATION		
INT32	FP32	FP32	FP64	4 GENERATION	INT32 FP32 FP32	FP64	4 GENERATION		
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64	1		
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
INT32	FP32	FP32	FP64		INT32 FP32 FP32	FP64			
LD/ ST	LD/ L ST 1	LD/ LD/ ST ST	LD/ LD/ ST ST	LD/ LD/ SFU	LDV LDV LDV LDV ST ST ST ST	LD/ LD/ ST ST	LD/ LD/ ST ST SFU		
L0 Instruction Cache L0 Instruction Cache Warp Scheduler (32 thread/clk) Warp Scheduler (32 thread/clk) Dispatch Unit (32 thread/clk) Dispatch Unit (32 thread/clk)									
		L0 Warp Sc Dispate	Instruction ( heduler (32 ch Unit (32 tl	Cache thread/clk) nread/clk)	L0   Warp Sc Dispate	Instruction heduler (32 ch Unit (32 t	Cache thread/clk) hread/clk)		
	,	L0 Warp So Dispate Registe	Instruction ( heduler (32 ch Unit (32 th r File (16,38	ache thread/clk) nread/clk) 4 x 32-bit)	L0   Warp Sc Dispato Register	Instruction heduler (32 ch Unit (32 t r File (16,34	Cache thread/clk) hread/clk) 84 x 32-bit)		
INT32	FP32	L0 Warp So Dispate Registe	Instruction ( heduler (32 ch Unit (32 th r File (16,38 FP64	Fache thread/clk) aread/clk) 4 x 32-bit)	L0 Warp Sc Dispate Register	Instruction I heduler (32 ch Unit (32 t r File (16,31	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32	FP32 FP32	L0 Warp So Dispate Registe FP32 FP32	Instruction ( cheduler (32 ch Unit (32 th r File (16,38 FP64 FP64	iache thread/clk) aread/clk) 4 x 32-bit)	L0 Warp Sc Dispate Register INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32	Instruction I heduler (32 ch Unit (32 t r File (16,34 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32	Instruction ( sheduler (32 ch Unit (32 th r File (16,38 FP64 FP64 FP64 FP64	iache thread/clk) rread/clk) 4 x 32-bit)	L0 Warp Sc Dispato Register INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32	Instruction heduler (32 ch Unit (32 t r File (16,34 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 th r File (16,38 FP64 FP64 FP64 FP64 FP64	iache thread/clk) vread/clk) 4 x 32-bit)	L0 Warp Sc Dispate Register INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32	Instruction heduler (32 ch Unit (32 t r File (16,34 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 th r File (16,38 FP64 FP64 FP64 FP64 FP64 FP64	iache thread/clk) aread/clk) 4 x 32-bit)	L0 Warp Sc Dispatr Register INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32 INT32 FP32 FP32	Instruction heduler (32 ch Unit (32 t r File (16,33 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 th r File (16,38 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	iache thread/clk) aread/clk) 4 x 32-bit)	L0 Warp Sc Dispate Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction ( heduler (32 ch Unit (32 t r File (16,33 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit)		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 t) r File (16,38 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) rread/clk) 4 x 32-bit) TENSOR CORE	L0 Warp Sc Dispato Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction heduler (32 ch Unit (32 t r File (16,34 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( ch duler (32 th ch Unit (32 th r File (16,38 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) aread/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispate Register NT32 FP32 FP32 INT32 FP32 FP32	Instruction I heduler (32 ch Unit (32 t r File (16,33 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 1 cheduler (32 ch Unit (32 th FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) aread/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispatr Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction   heduler (32 ch Unit (32 t FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 1 sheduler (32 th r File (16,38 FP64 F	thread/clk) aread/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispate Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction   heduler (32 th Unit (32 t File (16,31 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 4 cheduler (32 ch Unit (32 th FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) wead/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispate Register NT32 FP32 FP32 INT32 FP32 FP32 INT33 FP32 FP32 INT33 FP32 FP32 INT33 FP32 FP32 INT33 FP32 FP32 INT33 FP32 FP32	Instruction   heduler (32 th Unit (32 t FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 4 cheduler (32 ch Unit (32 th FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) aread/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispatr Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction 1 heduler (32 ch Unit (32 t FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 4 cheduler (32 ch Unit (32 th FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) aread/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispatr Register INT32 FP32 FP32 INT32 FP32 FP32	Instruction 1 heduler (32 th Unit (32 t Ffile (16,34 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP6	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 4 cheduler (32 ch Unit (32 ti FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) wead/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispate Register NT32 FP32 FP32 INT32 FP32 FP32 INT33 FP32 FP32 INT33 FP32 FP32	Instruction   heduler (32 th Unit (32 t File (16,31 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 4 cheduler (32 ch Unit (32 ti FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/clk) wead/clk) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION	L0 Warp Sc Dispate Register NT32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP	Instruction 1 heduler (32 ch Unit (32 t FFile (16, 33 FF64 FF64 FF64 FF64 FF64 FF64 FF64 FF	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION		
INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction 1 cheduler (32 ch Unit (32 th PP64 PP64 PP64 PP64 PP64 PP64 PP64 PP6	thread/clk) thread/clk) tread/clk trensor Core the Generation trensor Core the Generation trensor Core the Generation trensor Core the Generation trensor Core	L0 Warp Sc Dispatr Register INT32 FP32 FP32 INT32 FP32 FP32 FP32 INT32 F732 F732 F732 F732 INT32 F732 F732 F732 F732 F732 INT32 F732 F732 F732 F732 F732 F732 F732 F7	Instruction 1 heduler (32 r File (16,34 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP6	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION ST ST SFU		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Warp Sc Dispate FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 tl r File (16, 38 FP64	thread/clk) tread/clk) 4 x 32-bit) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION UN LOY SFU TENSOR MEMOR	L0 Warp Sc Dispate Register INT32 FP32 FP32 INT32 FP32 F932 INT32 FP32 F932 INT32 F932 F932 INT32 F932 F932 INT32 F932 F932 INT32 F932 F932 F932 INT33 F932 F932 F932 INT34 F932 F932 F932 INT35 F932 F932 F932 F932 INT35 F932 F932 F932 F932 INT35 F932 F932 F932 F932 F932 F932 F932 F932	Instruction 1 heduler (32 ch Unit (32 t FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Cache thread/clk) hread/clk) 34 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION LD/ ST SFU		
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	L0 Dispate Registe FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Instruction ( cheduler (32 ch Unit (32 th FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	thread/cik) wead/cik) 4 x 32-bit) 4 x 32-bit) TENSOR CORE 4 <sup>th</sup> GENERATION 57 st SFU Tensor Memory KB L1 Data Cad	L0 Warp Sc Dispate Register NT32 FP32 FP32 INT32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 FP32 INT32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP	Instruction I heduler (32 ch Unit (32 t FFile (16,3) FF64 FF64 FF64 FF64 FF64 FF64 FF64 FF6	Cache thread/clk) hread/clk) 54 x 32-bit) 54 x 32-bit) 54 x 32-bit) 55 ENSOR CORE 4 <sup>th</sup> GENERATION LO/ ST SFU		

X 144

#### **Machine Balance**



#### Memory Hierarchy

• Everything is a cache for something else

	Access time	Capacity	Managed By
On the Registers	l cycle	I KB	Software/Compiler
Level I Cache	2-4 cycles	32 KB	Hardware
Level 2 Cache	10 cycles	256 KB	Hardware
On chip	40 cycles	I0 MB	Hardware
Other Main Memory	200 cycles	10 GB	Software/OS
chips Flash Drive	10-100us	100 GB	Software/OS
Mechanical Hard Disk devices	10ms	I TB	Software/OS

## Parallel and Distributed Computing

![](_page_36_Figure_1.jpeg)

## Overheads of Parallelism (Some...)

- Communication overhead
- Synchronization overhead
- Load balancing overhead
- Decomposition overhead

#### Overheads of Parallelism Communication Overhead

![](_page_38_Figure_1.jpeg)

![](_page_38_Figure_2.jpeg)

![](_page_38_Figure_3.jpeg)

#### Overheads of Parallelism Synchronization Overhead

![](_page_39_Picture_1.jpeg)

#### Overheads of Parallelism Load balancing Overhead

![](_page_40_Figure_1.jpeg)

Overheads of Parallelism Decomposition overhead

![](_page_41_Figure_1.jpeg)

## Conclusion

- Our goal should be to reduce the time to solution
  - Look at not just computational efficiency of 1 worker, but of all of them working together.
- We need to ensure that we are comparing APPLES to APPLES!
- Parallel computing is not free, we need to be cognizant of the many overheads of parallelization

Poor parallelization may even degrade performance