# Transparent Asynchronous Compute Made Easy With PETSc

Jacob Faibussowitsch

June 6, 2023

### Your GPU Code Is Slow



### GPUs are like a factory:

- ightarrow Big startup cost
- $\rightarrow~$  Needs steady supply of work
- ightarrow Works best in bulk

Difficult to keep saturated for small jobs

- $ightarrow\,$  Performance left on the table
- ightarrow You paid for the whole GPU, you should use the whole GPU

1 PetscReal norm

 $\mathbf{2}$ 

- 3 // Must copy result D2H and synchronize
- 4 VecNorm(x, NORM 2, &norm);
- 5 norm = 1.0 / norm
- 6 // Must copy norm H2D and synchronize after
- 7 VecScal e(x, norm);

This is a common scenario!

- Functions operate on, or produce scalar values
- Values piped to next GPU function after basic manipulation
- Results are immediate ightarrow must synchronize GPU after each call

1 PetscReal norm

2

- 3 // Must copy result D2H and synchronize
- 4 VecNorm(x, NORM 2, &norm);
- 5 norm = 1.0 / norm
- 6 // Must copy norm H2D and synchronize after
- 7 VecScal e(x, norm);

This is a common scenario!

- Functions operate on, or produce scalar values
- Values piped to next GPU function after basic manipulation
- Results are immediate ightarrow must synchronize GPU after each call

Ideally this is all done in a stream on the GPU...

typedef pthread\_t gpu\_stream\_t;

Essentially threads, both the good and the bad

- $\boldsymbol{\uparrow}\;$  Putting work "on" a stream  $\rightarrow$  launching a thread
- ↑ Efficient "communication" via recorded events/semaphores
- Non-linear execution path, hard to grok
- Race condition hazards
- Deadlock hazards

typedef pthread\_t gpu\_stream\_t;

Essentially threads, both the good and the bad

- $\boldsymbol{\uparrow}\;$  Putting work "on" a stream  $\rightarrow$  launching a thread
- ↑ Efficient "communication" via recorded events/semaphores
- Non-linear execution path, hard to grok
- Race condition hazards
- Deadlock hazards

Surely every vendor has agreed on a single implementation... right?





- 1. Abstract away vendor-specific data structures
- 2 "Do the right thing" for missing functionality
- 3 Give high-level control over streams and synchronization primitives



### **Bottom Line**

Dictate how, when, and where, a particular GPU op runs

### Petsc::ManagedMemory: The Solution To All Your Problems

- 1 Petsc:: ManagedReal norm
- <sup>2</sup> PetscDevi ceContext dctx;
- з
- 4 // Retrieve the current active device context (or create one)
- <sup>5</sup> Pet scDevi ceCont ext Get Current Cont ext (&dct x);
- 6 // Store result in device memory (asynchronously!)
- $\tau$  VecNormAsync(x, NCRM\_2, &norm, dctx);
- 8 // Evaluate the expression on device (asynchronously!)
- 9 norm = eval (dct x, 1.0 / norm);
- 10 // Use result (asynchronously!)
- 11 VecScal eAsync(x, norm, dct x);
- 12 // Wait for results to be ready
- 13 Pet scDevi ceCont ext Synchroni ze(dct x);

## Petsc::ManagedMemory: The Solution To All Your Problems

- 1 Petsc:: ManagedReal norm
- <sup>2</sup> PetscDevi ceContext dctx;
- з
- 4 // Retrieve the current active device context (or create one)
- <sup>5</sup> Pet scDevi ceCont ext Get Current Cont ext (&dct x);
- 6 // Store result in device memory (asynchronously!)
- $\tau$  VecNormAsync(x, NCRM\_2, &norm, dctx);
- 8 // Evaluate the expression on device (asynchronously!)
- 9 norm = eval (dct x, 1.0 / norm;
- 10 // Use result (asynchronously!)
- 11 VecScal eAsync(x, norm, dct x);
- 12 // Wait for results to be ready
- 13 Pet scDevi ceCont ext Synchroni ze(dct x);

✓ Values computed in device memory

- 1 Petsc:: ManagedReal norm
- <sup>2</sup> PetscDevi ceCont ext dct x;
- 3
- 4 // Retrieve the current active device context (or create one)
- <sup>5</sup> Pet scDevi ceCont ext Get Current Cont ext (&dct x);
- 6 // Store result in device memory (asynchronously!)
- 7 VecNormAsync(x, NORM\_2, &norm, dctx);
- ${\bf s}$  // Evaluate the expression on device (asynchronously!)
- 9 norm = eval (dct x, 1.0 / norm);
- 10 // Use result (asynchronously!)
- 11 VecScal eAsync(x, norm, dct x);
- 12 // Wait for results to be ready
- <sup>13</sup> Pet scDevi ceCont ext Synchroni ze( dct x);
  - ✓ Values computed in device memory
  - $\checkmark$  Support for arbitrary expressions  $\rightarrow$  values stay on device

- 1 Petsc:: ManagedReal norm
- <sup>2</sup> PetscDevi ceCont ext dct x;
- 3
- 4 // Retrieve the current active device context (or create one)
- <sup>5</sup> Pet scDevi ceCont ext Get Current Cont ext (&dct x);
- 6 // Store result in device memory (asynchronously!)
- 7 VecNormAsync(x, NORM\_2, &norm, dctx);
- 8 // Evaluate the expression on device (asynchronously!)
- 9 norm = eval (dct x, 1.0 / norm;
- 10 // Use result (asynchronously!)
- 11 VecScal eAsync(x, norm, dct x);
- 12 // Wait for results to be ready
- 13 Pet scDevi ceCont ext Synchroni ze(dct x);
  - ✓ Values computed in device memory
  - $\checkmark$  Support for arbitrary expressions ightarrow values  $\mathit{stay}$  on device
  - $\checkmark\,$  Ability to await results  $\rightarrow$  functions may be asynchronous

### The Rubber Hits The Road

- 1 Petsc:: ManagedReal norm
- <sup>2</sup> PetscScal ar \*cpu\_array;
- <sup>3</sup> Pet scDevi ceCont ext dct x\_a, dct x\_b, dct x\_c;
- 4
- 5 // These are all \*separate\* streams
- <sup>6</sup> Pet scDevi ceCont ext Get Current Cont ext ( &dct x\_a);
- 7 PetscDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_b);
- 8 Pet scDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_c);
- 9 // Store result in device memory (asynchronously!)
- 10 VecNormAsync(x, NORM\_2, &norm,  $dctx_a$ );
- 11 // Evaluate the expression on device (asynchronously!)
- 12 norm = eval (dct x\_b, 1.0 / norm);
- 13 // Use result (asynchronously!)
- 14 VecScal eAsync(x, norm,  $dct x_c$ );
- 15 // Get results (... synchronously?)
- 16 VecGetArray(x, &cpu\_array);

### To Go Even Further Beyond

- 1 Petsc:: ManagedReal norm
- 2 PetscScal ar \*cpu\_array;
- <sup>3</sup> Pet scDevi ceCont ext dct x\_a, dct x\_b, dct x\_c;
- 4
- 5 // These are all \*separate\* streams
- <sup>6</sup> Pet scDevi ceCont ext Get Current Cont ext ( &dct x\_a);
- $7 PetscDevi ceContextDupl i cate(dctx_a, &dctx_b);$
- 8 Pet scDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_c);
- 9 // Store result in device memory (asynchronously!)
- 10 VecNormAsync(x, NORM\_2, &norm,  $dctx_a$ );
- 11 // Evaluate the expression on device (asynchronously!)
- 12 norm = eval ( $dct x_b$ , 1.0 / norm);
- 13 // Use result (asynchronously!)
- 14 VecScal eAsync(x, norm, dct x\_c);
- 15 // Get results (synchronously!)
- 16 VecGetArray(x, &cpu\_array);
- 17 // PetscDevi ceCont ext Synchroni ze(dct x); ???

## To Go Even Further Beyond

- 1 Petsc:: ManagedReal norm
- 2 Pet scScal ar \*cpu\_array;
- <sup>3</sup> Pet scDevi ceCont ext dct x\_a, dct x\_b, dct x\_c;
- 4
- 5 // These are all \*separate\* streams
- <sup>6</sup> Pet scDevi ceCont ext Get Current Cont ext ( &dct x\_a);
- 7 PetscDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_b);
- 8 Pet scDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_c);
- 9 // Store result in device memory (asynchronously!)
- 10 VecNormAsync(x, NORM\_2, &norm,  $dctx_a$ );
- 11 // Evaluate the expression on device (asynchronously!)
- 12 norm = eval ( $dct x_b$ , 1.0 / norm);
- 13 // Use result (asynchronously!)
- 14 VecScal eAsync(x, norm, dct x\_c);
- 15 // Get results (synchronously!)
- 16 VecGetArray(x, &cpu\_array);
- 17 // PetscDevi ceCont ext Synchroni ze(dct x); ???

Automatically serializes stream dependencies for you

✓ Even across "regular" API

### To Go Even Further Beyond

- 1 Petsc:: ManagedReal norm
- 2 Pet scScal ar \*cpu\_array;
- <sup>3</sup> Pet scDevi ceCont ext dct x\_a, dct x\_b, dct x\_c;
- 4
- 5 // These are all \*separate\* streams
- <sup>6</sup> Pet scDevi ceCont ext Get Current Cont ext ( &dct x\_a);
- 7 PetscDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_b);
- 8 Pet scDevi ceCont ext Dupl i cat e( dct x\_a, &dct x\_c);
- 9 // Store result in device memory (asynchronously!)
- 10 VecNormAsync(x, NORM\_2, &norm,  $dctx_a$ );
- 11 // Evaluate the expression on device (asynchronously!)
- 12 norm = eval ( $dct x_b$ , 1.0 / norm);
- 13 // Use result (asynchronously!)
- 14 VecScal eAsync(x, norm, dct x\_c);
- 15 // Get results (synchronously!)
- 16 VecGetArray(x, &cpu\_array);
- 17 // PetscDevi ceCont ext Synchroni ze(dct x); ???

Automatically serializes stream dependencies for you

✓ Even across "regular" API !!!

CG Benchmark:

- Run on ANL Polaris:
  - GPU: NVIDIA A100
  - CPU: AMD EPYC "Milan"
- Solve Laplace equation of varying size and density<sup>1</sup>
- 20 KSP iterations (for simplicity of comparison)
- Jacobi preconditioning
- 1 MPI Rank

<sup>1</sup>Cartesian product of 2D, 3D, with, and without f nite difference stencil

## Time To Solution

TIME TO SOLUTION VS NNZ



#### cg\_nai n

- - ksp\_type cg on main branch
- Synchronous

#### cgasync:

- ksp\_type cgasync on exp. branch
- Fully Asynchronous

Flops

FLOPS VS NNZ



#### cg\_nai n

- -ksp\_type cg on main branch
- Synchronous

#### cgasync:

- ksp\_type cgasync on exp. branch
- Fully Asynchronous

### Time To Solution Ratio



RATIO OF TIME-TO-SOLUTION VS NNZ

### Time To Solution Ratio



RATIO OF TIME-TO-SOLUTION VS NNZ

### Mystery Solved?



# Mystery Solved?

+15ms +20														
						-			÷			÷		
1														
	-				KSPSolve	[45.046 m	5]							
V						D	octesyne [3	17.887 ms]						
di tratte that have														
ni i sunsegeringendennen	1													
Reconstruction of the second						cu	idaStreamS	ynchronize						
+15ms +20	ms	+251115	+3011				39.5			401115		+301113		100113
+15ms +20		+201115 Mat   MatM	+3011 Matt	at Mat	tM), (MatM.		- d Math	Mat.	MatM.	MatM.	() MatM   /load_bi	MatM	() MatM	MatM
+15ms +20		+20115 Nat. Nat. veid _	Watk M	at Mat	tM), (MatM.	IIS Mat.	- 4 (void	MatMult Begins: ( Ends: 0.1 grid: <<	K/cuspar 0.038678 0402129 <27219,	MatM., seSpMV, 3s s (+1.535 1, 1>>>	() () () () () () () () () () () () () (	MatM.	kernel	
+15ms +20	ms	+20115 Mat. MatM.	+30m	st / Mat	tM) (MatM.		- / Matt	MatMult Begins: ( Ends: 0.0 grid: << block: <	(/cuspar: 0.038678 0402129 <27219, <<512, 1,	MatM. seSpMV, 3s s (+1.535 1, 1>>> 1>>>	// MatM //oad_bi	MatM.	kernel	MatM.
+15ms +20	ms	+20115 Mat. MatMa	+3011	at Mat id Voi		- []] [Mat. - []] [Void	- 4 Mati	MatMult Begins: ( Ends: 0.1 grid: << block: <- Launch 1 Static St	t/cuspars 0.038678 0402129 <27219, <<512, 1, Type: Reg	seSpMV 3s 1, 1>>> 1>>> 1>>> 1>>> 1>>>	//oad_b; 5 ms)	Math.	_kernel	(void
+15ms +200	ms	+2011S	+3011	at ( Mat				MatMull Begins: ( Ends: 0.1 grid: << block: <- Launch T Static ST Static ST	K/cuspar 0.038676 0402129 <27219, <<512, 1, Type: Reg ared Me ashared Me	seSpMV, 3s (+1.535 1, 1>>> 1>>> gular mory: 16 Memory: 16	//oad_bi 5 ms) ,384 byt	Mam.	_kernel	(woid
+15ms +20	ms	+2011s	+3011	at Mat			(Matt	MatMult Begins: ( Ends: 0.4) block: << Launch 1 Static Sr Dynamic Register:	t/cuspars 0.038676 0402129 <27219, <<512, 1, Type: Res ared Me is Shared s Per Thr	seSpMV 3s (+1.535 1, 1>>> 1>>> gular mory: 16 Memory: 16 Memory: 19 ead: 40	(load_b) 5 ms) ,384 bytes	Mathematical Science S	_kernel	(wold)
+15ms +20	ms	+2011s Mat. Vat.		at Mat	KSPSolve		(Void	MatMull Begins: 0 grid: << Launch 1 Static Sh Dynamic Register: Local Me	K/cuspars 0.038676 0402129 <<512, 1, Type: Reg ared Me s Per Thr amory Pe	MetMa seeSpMV 3s s (+1.535 1, 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1	(load_b) 5 ms) ,384 byt 0 bytes : 0 bytes	alancing	kernel	(void
+15ms +20	ms	+20115	+3011 Watu   M Veid   1 Veid   1 Veid   1	at Mat	KSPSolve	- [] Mat. ] . [] woid ] . [45.046 mi	<ul> <li>-) (Matt</li> <li>-) (void</li> <li>-) (void&lt;</li></ul>	MatMull Begins: ( Ends: 0.1 grid: << block: <- Launch 1 Static Sh Dynamic Register: Local Me Local Me Shared b	t/cuspars 0.038676 0402129 <<512, 1, Type: Reg ared Me s Per Thr amory Pe amory To demory To	seSpMV 3s s (+1.535 1, 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1	//oad_bi 5 ms) ,384 by1 0 bytes 23,104 b 65 536	alancing ytes	kernel	(VatX_)
+15ms +20	ms	+23ins Mat. ( WatW wold _ ( wide )	+30/11   Vasu	at	KSPSolve	-     Mat.   -     void         	-) ( Matt -) ( void -) ( void -) ( void	MatMult Begins: ( Ends: 0.4 block: <- block: <- block: static Sh Dynamic Register: Local Me Shared N Shared N	Vanite t/cuspars 0.038677 0402129 <27219, <<512, 1, 1/ype: Reg ared Me shared is shared is share	MatMa seSpMV s (+1.535 1, 1>>> yular mory: 16 Memory: ead: 40 r Thread tal: 56,66 executed ank Size	//oad_b; 5 ms) ,384 by1 0 bytes : 0 bytes 23,104 b : 65,536 : 4 B	MattMa alancing res ytes i bytes	_kernel	( wax )
+15ms +20		+23(II)5		at	KSPSolve		s]	MatMult Begins: ( Ends: 0.) grid: << block: < block: < block:  block:  block:  block:  block:  block: static sh block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: block: b	t/cuspar 0.038675 0402129 <27219, <<512, 1, Type: Reg ared Me s Shared s Sh	Matter seSpMV 3s s (+1.53 1, 1>>> 1>>> yular mory: 16 Memory: 16 Memory: 16 Memory: 16 Memory: 10 Memory: 10	//oad_b; 5 ms) ,384 byt 0 bytes 23,104 b : 65,536 :: 4 B 0 %	MattM alancing ees ytes bytes	_kernel	(void
+15ms +20		*23(II)5	+30/11	nt Mat	tM MatM.		s] sda51reers	Mathematical Status Shared M Shared M Shared M Shared M Shared M	t/cuspar: 0.038676 0402129 <27219, <<512, 1, Type: Reg ared Me s Shared s S	seSpMV 3s s (+1.53 1, 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1>>> 1	//oad_b //oad_b 5 ms) ,384 byt 0 bytes 23,104 b : 65,536 : 4 B 00 % 92617	alancing es ytes bytes	kernel	
+15ms +200		Vatura Vatur	+SUII Manw.    M   Vanw.    M   Vanw.    M	at. ( Mar	KSPSolve	115 	e) CtraSyne (S rdaStreamS	Mathuli Begins: 0. grid: << block: <- Launch T Static Sh Dynamic Register: Local Me Local Me Shared N Theoreti Launcher Charet I Launcher Correleti Correleti Correleti	Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logarian Logari	Matter seSpMV, 3s (±1.53§ 1, 1>>> yular mory: 16 Memory: 16 Memory	/load_b; 5 ms) ,384 byt 0 bytes 23,104 b : 65,536 :: 4 B 00 % 92617	alancing ees ytes bytes	_kernel	void

Key Takeaways:

- $\checkmark~$  Faster across the board
- $\checkmark~$  Much faster (  $\sim 1.8\times$  ) if latency bound
- ✓ Speedup diminishes as solve becomes "work bound"...

Key Takeaways:

- $\checkmark\,$  Faster across the board
- $\checkmark~$  Much faster (  $\sim 1.8\times$  ) if latency bound
- ✓ Speedup diminishes as solve becomes "work bound"...
- ...but that assumes you cannot fill the time with other work!

### Future Work

- ✓ GMRES implementation (finishing touches)
- ✓ Automatic runtime kernel fusion
  - Implemented for Vec, but benefits still under investigation...
  - Needs tight integration with  $\mathbf{Mat}$  to be truly useful
- ✓ Ability to cancel GPU work in-flight ("unlaunch" a kernel)

### **Future Work**

- ✓ GMRES implementation (finishing touches)
- ✓ Automatic runtime kernel fusion
  - Implemented for Vec, but benefits still under investigation...
  - Needs tight integration with **Mat** to be truly useful
- ✓ Ability to cancel GPU work in-flight ("unlaunch" a kernel)

### Thanks For Listening!

Check out the branch

j acobf / 2022- 11- 28/ pet sc- nanaged- nenor  $y^a$ 

# Any questions?

<sup>a</sup>https://gitlab.com/petsc/petsc/-/merge\_requests/6178