OPTIMIZING HPC SIMULATION AND VISUALIZATION CODE USING NVIDIA NSIGHT SYSTEMS

Daniel Horowitz - Director of Platform Developer Tools, NVIDIA,
Robert (Bob) Knight - Principal System Software Engineer, NVIDIA
Mike Hallock - Research Programmer, U.Illinois at Urbana Champaign
John Stone - Senior Research Programmer, U.Illinos at Urbana Champaign
March 26, 2018



INTRODUCING NSIGHT SYSTEMS

System-wide Performance Analysis Tool

Focus on the application's algorithm - a unique perspective

Scale your application efficiently across any number of CPUs & GPUs

3.2x-4.1x Speedup Achieved on Visual Molecular Dynamics!

Stay tuned for the details



NSIGHT PRODUCT FAMILY

Standalone Performance Tools

Nsight Systems - System-wide application algorithm tuning

Nsight Compute - Debug/optimize specific CUDA kernel available @next major CUDA release - Use NVIDIA Visual Profiler today

Nsight Graphics - Debug/optimize specific graphics shader

Workflow Nsight Systems Start Here Nsight Compute

IDE Plugins

Nsight Visual Studio/Eclipse Edition - editor, debugger, some perf analysis

NSIGHT SYSTEMS USER



MAXIMIZE YOUR GPU INVESTMENT

Find the right optimization opportunities

Balance your workload across CPUs and GPUs

Achieve real-time performance requirements

Optimize for HPC environments - minimum time to solution



FEATURES

User Instrumentation

NVidia Tools eXtension - aka NVTX

API Tracing

CUDA, OpenGL,

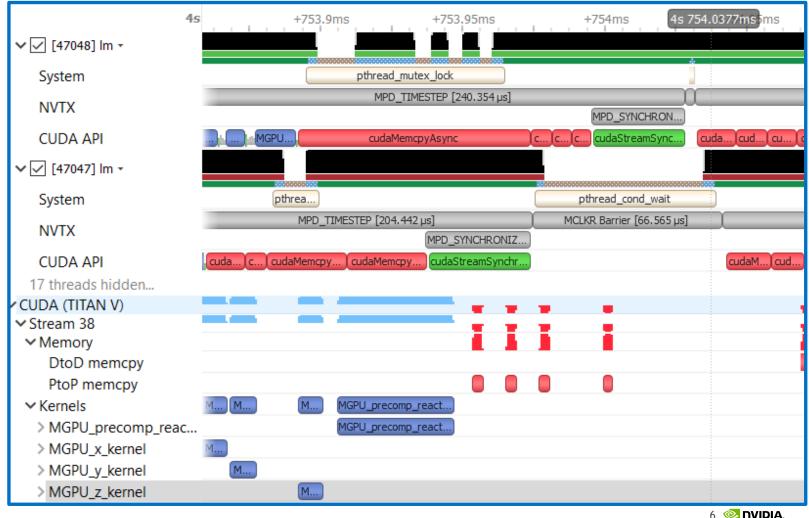
cuDNN, cuBLAS

System strace-lite

Backtrace Collection

Sampled IPs

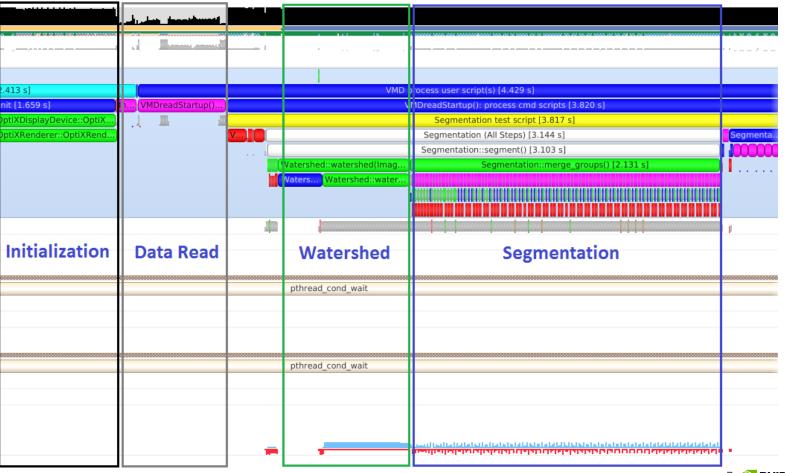
Blocked state



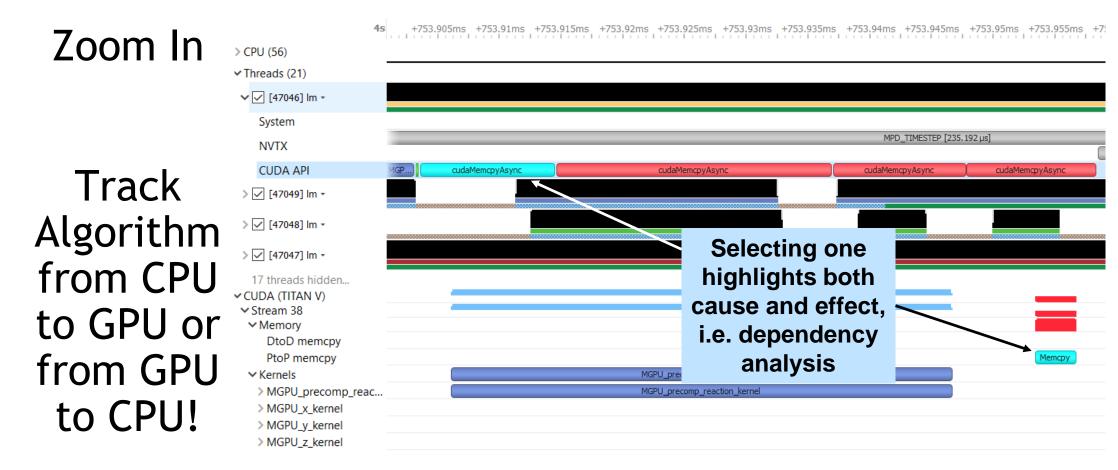
APPLICATION ALGORITHM

Zoom Out

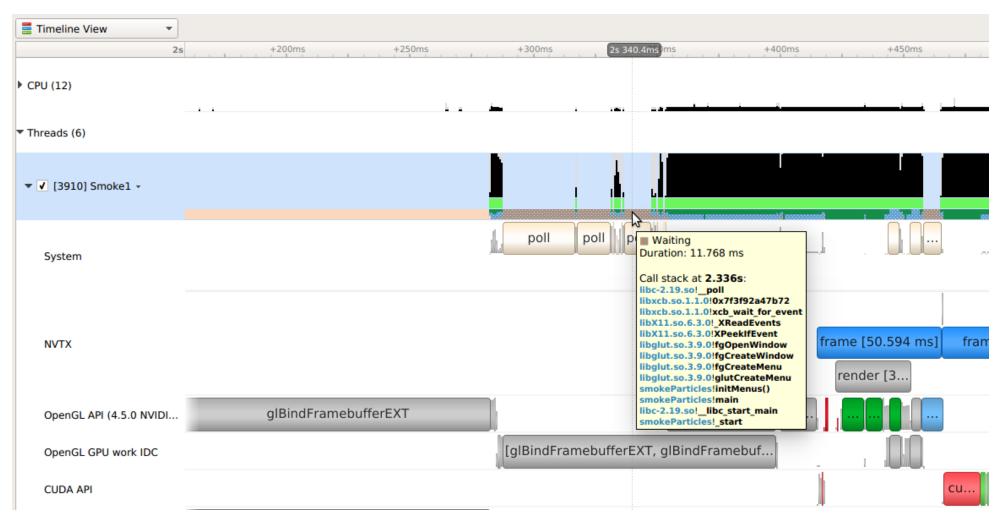
Four Distinct Phases of VMD Algorithm Become Visible



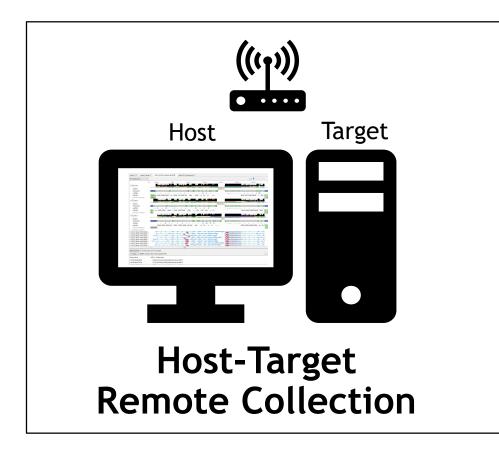
CORRELATION TIES API TO GPU BEHAVIOR

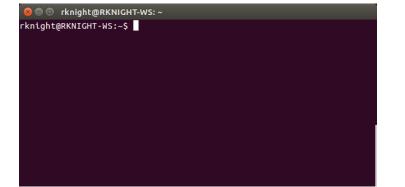


BLOCKED STATE BACKTRACE



DATA COLLECTION





Command Line Interface No connection! Import later

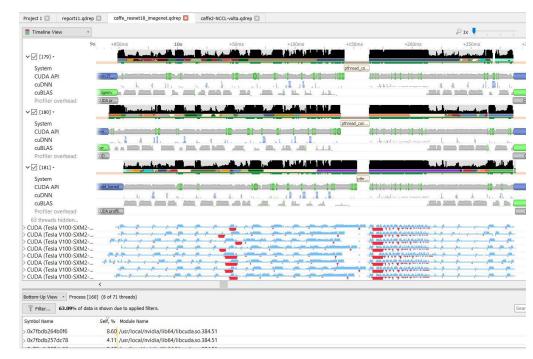
CLI enables easy collection on servers and in containers

REPORT NAVIGATION DEMO

New Tool - <u>Outstanding</u> Interactive Performance and Level of Detail Available

Core Areas

- Algorithm Overview Using NVTX Tags
- OS Thread Timeline including APIs Traced
- Correlation of OS Thread API Use with GPU Activity
- CPU Sampling Shows Hot OS Thread
 Code/Bottlenecks



COMMON OPTIMIZATION OPPORTUNITIES

► CPU

- Thread synchronization
- Algorithm bottlenecks starve the GPUs

Multi GPU

- Communication between GPUs
- Lack of stream overlap in memory management, kernel execution

Single GPU

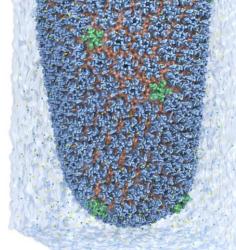
- Memory operations blocking, serial, unnecessary
- Excessive synchronization device, context, stream, default stream, implicit
- CPU/GPU overlap avoid excessive communication

VMD – "Visual Molecular Dynamics"

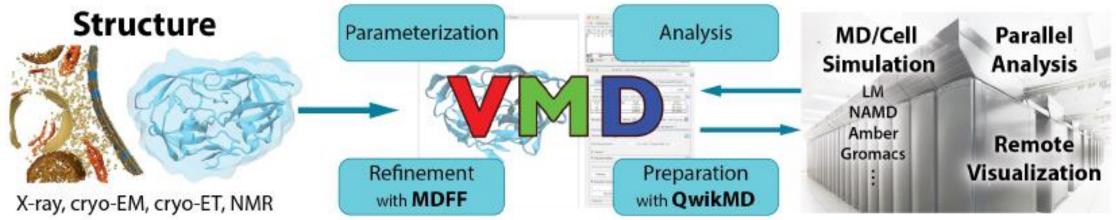
- Visualization and analysis of:
 - Molecular dynamics simulations
 - Lattice cell simulations
 - Quantum chemistry calculations
 - Sequence information
- User extensible scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



Cell-Scale Modeling



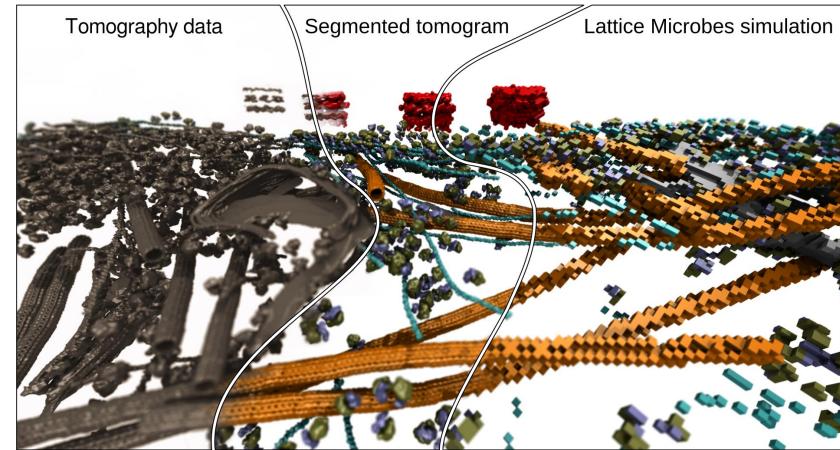




CRYO-EM / CRYO-ET IMAGE SEGMENTATION

Evaluate 3-D volumetric electron density maps and segment them, to identify key structural components

Index/label components so they can be referred to, colored, analyzed, and simulated...







CRYO-EM DENSITY MAP SEGMENTATION APPROACH, GOALS

Watershed segmentation:

- Smooth/denoise image (e.g. blur)
- Find local minima of image/gradients
- Connect minimum voxels with neighbors of similar intensity, marking them with the same "group" number
- "Grow" each group (merging groups where rules allow) until no more updates occur

Scale-space segmentation variant does further blurring and group merging

Goals:

- Reach interactive performance rates (under 1 second) for common density map sizes between 128³ to 256³ voxels
- Handle next-generation problem sizes (768³ to 2048³) smoothly with only a brief wait



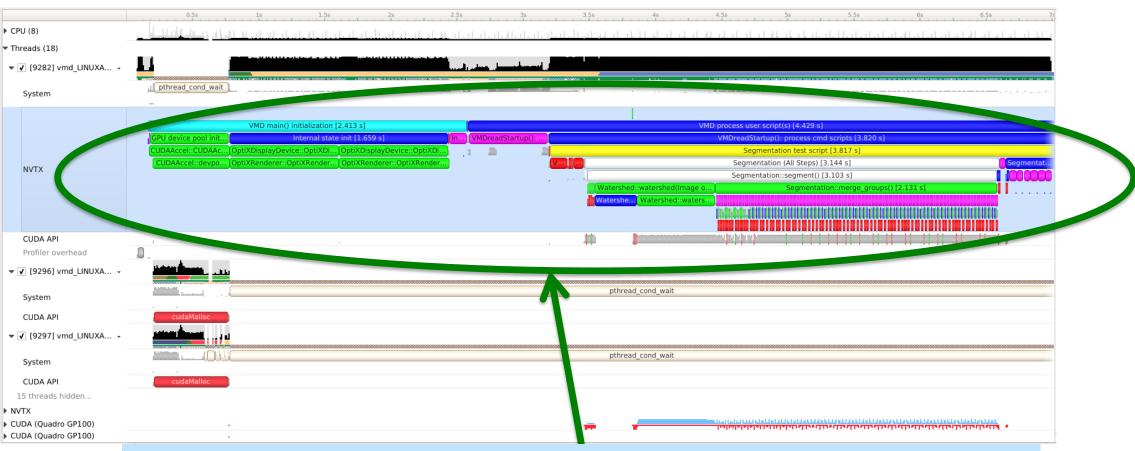


1: INITIAL VMD IMAGE SEGMENTATION TRACE

	3.6s	3.8s	4s	4.2s	4.4s	4.6s	4.8s	5s	5.2	s	5.6s	5.8s	(is	6.2s	6.4s	6.6s	6.8s	7s
CPU (8)		i ka ki					u u	Li u								1.11			<u></u>
Threads (18)																			
▼ ✔ [9234] vmd_LINUXA																			
System	, L						h												· · · · · ·
CUDA API Profiler overhead																			11 I.
▼ 🖌 [9249] vmd_LINUXA →																			
System									pt	hread_cond_wait				000000000000000000000000000000000000000			000000000000000000000000000000000000000	500000000000000000000000000000000000000	
CUDA API																			
🕶 🖌 [9248] vmd_LINUXA 🗸																			
System									pt	hread_cond_wait									
CUDA API																			
15 threads hidden CUDA (Quadro GP100) CUDA (Quadro GP100)				\leftarrow					*****	~~~~~	~~~~~	~~~~	~~~~	*****	~~~~	*****	*****	******	πτ
										1									
•	GP	U co	mpu	te act	ivity	show	wn in	BLU	IE.										

- Memory transfer activity shown in RED.
- Trace shows memory transfers taking a lot of the time in the second phase...
- What is the algorithm doing here? Why?

2: VMD PROFILE W/ NVTX TAGS



Added NVTX tags clearly show algorithm phases in the Nsight System timeline.

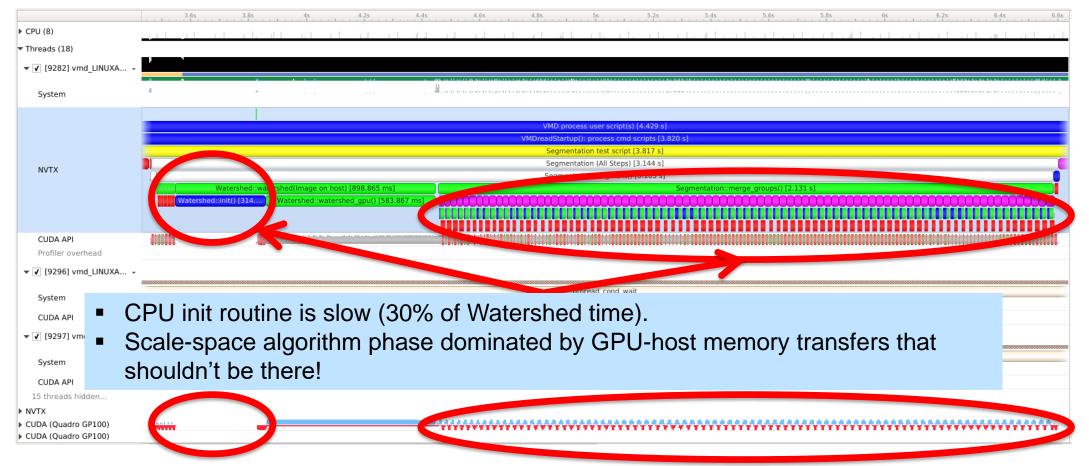




2: VMD IMAGE SEGMENTATION W/ NVTX

	3.5s	4s	4.5s	5s	5.5s	6s	6.5s 7s	
CPU (8)				al de de de de			ilala la	
 Threads (18) 								
▼ ✔ [9282] vmd_LINUXA •								
System	a	· · · · · · ·	, k					
				VMD process user script(s) [4.4	420 cl			
			V	/MDreadStartup(): process cmd sc				
				Segmentation test script [3				
NVTX				Segmentation (All Steps) [3.144 Segmentation::segment() [3.103			Segmentation split gr	
	Water	shed::watershed(Image on host) [89		S	egmentation::merge_groups() [2.131 s]			
	Watershee	d::init() [] Watershed::watershed_g	pu() [583.86]		*****			
CUDA API Profiler overhead								
▼ ▼ [9296] vmd LINUXA ·								
• • • [9290] • • • • • • •		***************************************						
	ncily zoom in c	n NIV/TY too	ie (doublo di	ick) so that	/ fill horizonal			
				· ·			Total time:	3.14s
Selected	ed yellow "Seg	amentation t	est script" vi	ew relevan	it work			
	•		-				Total speedup:	1. UX
Tags s	nown are algo	nunm phase	s, sub-phase	es, iteration	ns, kernels, an	u –		
CUDA	API calls							
•							Phases:	
 CUDA (Quadro GP100) CUDA (Quadro GP100) 		-		*****************	******	*****	Constructors: (10
							Watershed: 0).9s
							Soolo Space	120
							Scale-Space: 2	2.155
							Other: 0.	014s

2: IDENTIFIED BOGUS COPIES, SLOW CPU INIT



2: DETAIL: IDENTIFIED BOGUS GPU-HOST COPIES

4s +430ms	+440ms	+450ms	+460ms	+470ms	+480ms	+490ms	+500ms	+510ms	+520ms
A									
					10 K	5	e 19.00	,	4 66
			D I		al, a	h	i, lisat		a s.h
						20 s]			
Watershed::v	vatershed_gpu() [583.867 ms]								
			bearespacerine.						ianBlur::cop
			LoudaDeviceS	cudaMemcpy	cudaDevice	Sy	cpy CudaDev	viceSy cudat	Aemer [] [] [cudaDevice
A +									
000000000000000000000000000000000000000				pthrea	ad cond wait			******	
A +	Bogus	memory	conies						
000000000000000000000000000000000000000						000000000000000000000000000000000000000			
	Create	e daps in (GPU ex	ecution					
		9-1							
									A 11
		Watershed::watershed(Image on host) [898.865 Watershed::watershed_gpu() [583.867 ms]	Watershed::watershed(Image on host) [898.865 Watershed::watershed_gpu() [583.867 ms] Watershed::watershed_gpu() [583.867 ms] • Bogus memory • Create gaps in (Watershed:watershed(Image on host) [898.865) Watershed:watershed(Image on host) [898.865) Watershed:watershed(gput) [583.867 ms] ScalespeceFit Curre Curre Curre Bogus memory copies! Create gaps in GPU ex	Matershed:watershed[mage on host] [898.865 Watershed:watershed[mage on host] [898.865 Watershed:watershed] Watershed:watershed[mage on host] [898.865 Watershed:watershed] Watershed:watershed[mage on host] [898.865 Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watershed:watershed] Watersh	 More that the second sec	 Wetershed://watershed/image on host/ 1998.665 Watershed://watershed/image on host/ 1998.665 Watershed://watershed/image on host/ 1998.665 Watershed://watershed/image on host/ 1998.665 ScaleSpace/Hite://watershed/image on host/ 1998.665 Creater gaps in GPU execution 	<pre></pre>	 More process user script(s) [4.42 s) More addatus() process user script(s) [4.42 s) Segmentation (Mission) Sequestion (Mission) Sequestion) Sequestion) Sequestio





3: BOGUS COPIES ELIMINATED

	3.4s 3.6s 3.8s 4s 4.2s 4.4s 4.6s 4.8s 5s 5	.2s 5.4s 5.6s
CPU (8)	<u>1 </u>	
Threads (19)		
▼ ✔ [9341] vmd_LINUXA		
Custom	pthread_cond_wait	***************************************
System		
CUDA API		
▼ ✔ [9327] vmd_LINUXA →		***********
System	1 A. A. A. A. A. A. A. A. Managara, Manag Managara, Managara, Manag Managara, Managara, Manag	
	VMD process user script(s) [3.567 s]	
	VMDreadStartup(): process cmd scripts [3.089 s] Segmentation test script [3.082 s]	
NVTX	Segmentation (All Steps) [2:214 s]	
	Segmentation::errors 5] Watershed::watershed(Image on host) [893.945 ms] Segmentation::merge_groups() [1.248 s]	
	Watershed::watershed(Image on host) [893.945 ms] Segmentation::merge_groups() [1.248 s] Watershed::init() [310.236 ms] Watershed::watershed_gpu() [583.701 ms]	
CUDA API Profiler overhead		
▼ ▼ [9342] vmd LINUXA		Total time: 2.21s
• • • [9542] VIIId_LINOXA •	- Degue conice eliminated	Total an adverse 4 Av
System	 Bogus copies eliminated. 	Total speedup: 1.4x
CUDA API		
16 threads hidden		
 NVTX CUDA (Quadro GP100) 		Phases:
CUDA (Quadro GP100)		Constructors: 0.1a
		Constructors: 0.1s
		Watershed: 0.9s
		Scale-Space: 1.25s
		Öther: -

3: DETAIL: BOGUS COPIES ELIMINATED

4	4s +305ms +310ms +315ms +320ms +330ms +335ms +340ms +345ms +350ms +355ms
CPU (8)	
 Threads (19) 	
▶ ✔ [9341] vmd_LINUXA →	-
▼ ✔ [9327] vmd_LINUXA →	•
System	الله الله المراجع (iocti) الله (iocti) الله (iocti)
	VMD process user script(s) [3.567 s]
	VMD process data stript(a) (3,000 a) VMDreadStartup(): process cmd scripts [3,089 s]
	Segmentation test script (3.082 s)
NVTX	Segmentation (All Steps) [2.214 s] Segmentation::segment() [2.323
	Watershed://watershed/Image on host) [893.945 ms] Segmentation::segment(112) for
	Watershed::watershed_gpu() [583.701 ms] ScaleSpaceFilter::merge() [10.363 ms] ScaleSpaceFilter::merge() [11.08-n.
	ScaleSpaceFilter::find_grou, Scale, ScaleSpaceFilter::find_groups,
CUDA API	cudaM cudaM cudaM cudaM cudaM cudaM cuda cudaDeviceSynchronize cudaDeviceSynchronize cudaDeviceSynchronize cudaDeviceSynchronize cudaDeviceSynchronize
Profiler overhead	
▼ 🖌 [9342] vmd_LINUXA	
	pthread cond_wait
System	
CUDA API	
16 threads hidden	
NVTX	
CUDA (Quadro GP100) CUDA (Quadro GP100)	
(Quadro Gritoo)	
	Boaus conjes eliminated

- Bogus copies eliminated.
- Gaps between GPU kernels are now very short.
- Speedup for just the scale-space algorithm phase is 1.7x

22 📀 nvidia.

3: DETAIL: SLOW CPU INIT ROUTINE

	3.4s 3.5s	3.6s	3.7s	3.8s	3.9s	4.1s	4.2s	4.3s 4.4s
▶ CPU (8)								
▼ Threads (19)	-							
▶ ✔ [9341] vmd_LINUXA →			000000000000000000000000000000000000000	000000000000000000000000000000000000000			000000000000000000000000000000000000000	
▼ 🗸 [9327] vmd_LINUXA →			e 1	ĩ	8		1 X X X	
System	il .		1 .					10
					VMD process user script(s) [3.567 s]			
	-				eadStartup(): process cmd scripts [3.089	s]		
					Segmentation test script [3.082 s]			
NVTX					Segmentation (All Steps) [2.214 s]			
	Se				Segmentation::segment() [2.173 s]			
		Watershed::init() [310.236 ms]		Watershed::watershe	ed(Image on host) [893.945 ms]	had any () [502 701 ma]		Segmentation::merge_groups() [
		watershed::init() [310.236 ms]			watersned::waters	hed_gpu() [583.701 ms]		
CUDA API	.00.0			alalaaddadaa	a da		فالتعاريب فنفاذ فالمستطر واخذر فردافك	
Profiler overhead								
▼ ✔ [9342] vmd_LINUXA →			•	Water	shed algorith	m CPU init ta	akes 310r	ns.
	000000000000000000000000000000000000000				Natershed ru			
System				Total	valersneu ru	111111111111111111111111111111111111111	115.	
CUDA API			-		nit taking 31	% of the run	ntimo for	this phase!??!
16 threads hidden				CFUI	init taking 54	% of the ful	iume ior	
▶ NVTX								
CUDA (Quadro GP100)								
CUDA (Quadro GP100)								
								-





4: FAST GPU INIT ROUTINE

	3.4s 3.6s 3.8s 4s 4.2s 4.4s 4.6s 4.8s	5s 5.2s
▶ CPU (8)		
 Threads (19) 		
▼ ▼ [9373] vmd LINUXA		
System	h, ",h	
	VMD process user script(s) [3.330 s]	
	VMDreadStartup(): process cmd scripts (2.774 s)	
	Segmentation test script [2.768 s] Segmentation (All Steps) [1.914 s]	
NVTX	Segmentation: (kii Steps) (1:54 s)	
	Watershed::watershed(Image on host) [595.467 ms]	
	Watershed::watershed.gpu() [581.995 ms] 000000000000000000000000000000000000	
CUDA API		
Profiler overhead		
▼ 🖌 [9388] vmd_LINUXA		
	pthread_cond_wait	
System		
CUDA API		
🕶 🖌 [9387] vm	GPU-based Watershed init kernel 13.4ms.	Total time: 1.91s
System	Speedup over CPU-based initialization: 23x	Total speedup: 1.6x
CUDA API	opectup ever er e based initialization. Zex	
16 threads hidden		
CUDA (Quadro GP100)		
CUDA (Quadro GP100)		Phases:
		Constructors: 0.1s
		Watershed: 0.59s

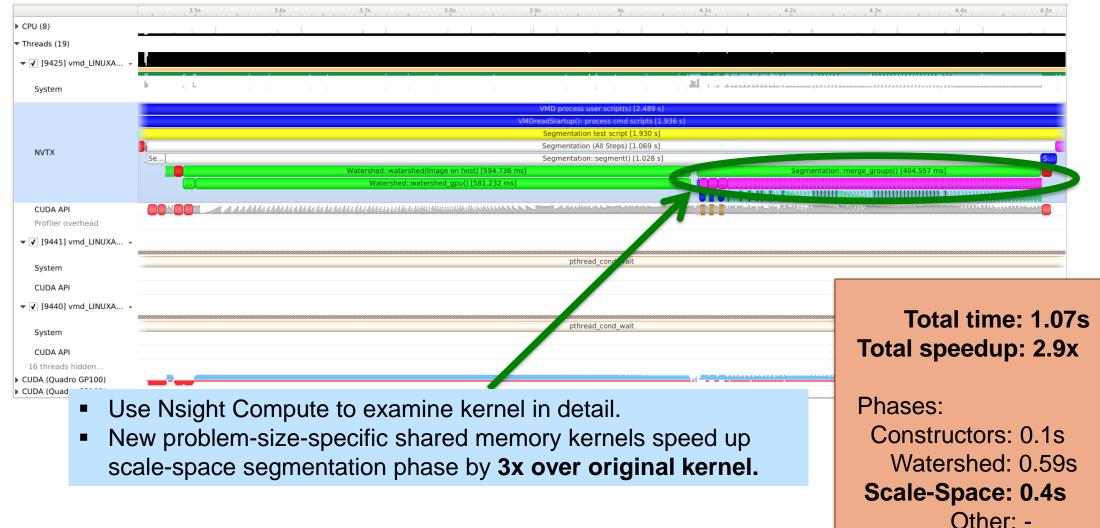
Scale-Space: 1.25s Other: -

4: DETAIL: SLOW SCALE-SPACE MERGE GROUPS KERNEL

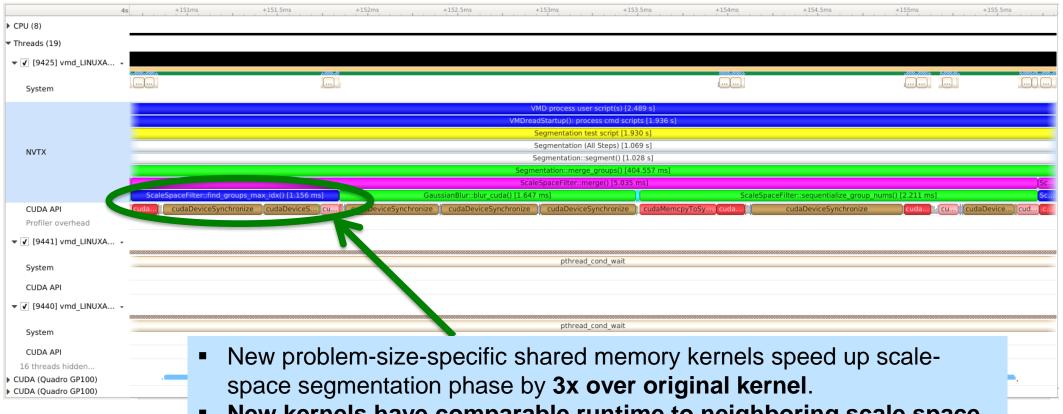
4s +249.5ms +250ms	+250.5ms +251ms +251.5ms +252ms +252.5ms +253ms	+253.5ms +254ms +254.5ms +255ms	+255.5ms +256ms +256.5ms +257n	ns +257.5ms +258ms
CPU (8)				
Threads (19)				
[9373] vmd_LINUXA +				
		-884	Rollo	i Mah Million
System			001	
		VMD process user script(s) [3.330 s]		
		VMDreadStartup(): process cmd scripts [2.774 s]		
		Segmentation test script [2.768 s]		
NVTX		Segmentation (All Steps) [1.914 s] Segmentation::segment() [1.873 s]		
		Segmentation::segment() [1:075 S]		
	564	35paceriteminarya.		ScaleSpaceFilte
	ScaleSpaceFilter::find_groups_max_idx() [11.018 ms]	GaussianBlur::blur_cuda	a() [1.643 ms] ScaleSpaceFilter::sequentialize_gro	
CUDA API		cuda cudaDeviceS cudaDeviceS	cudaDeviceS cudaMe c cudaDeviceSynch	ronize cuda c cudaDevi
Profiler overhead				
▼ √ [9388] vmd_LINUXA →				
		pthread_cond_wait		
System				
CUDA API				
✔ [9387] vmd_LINUXA +				
100000000000000000000000000000000000000		pthread_cond_wait	*****	
System				
CUDA API				
16 threads hidden				
UDA (Quadro GP100)				
	alforosolo opera arecum m	area anarationa ia ale		
 GPU keri 	nel for scale-space group me	erge operations is sid	ow compared to	
		•	•	
other ker	nels, opportunity!			

 Write new special-case scale-space merge kernels for problem sizes small enough to allow atomic ops in shared memory rather than global memory.

5: FASTER MERGE GROUPS KERNELS



5: DETAIL: FASTER MERGE GROUPS KERNELS



 New kernels have comparable runtime to neighboring scale space kernels, no longer an outstanding optimization opportunity.





5: DETAIL: EXCESSIVE ERROR CHECKING

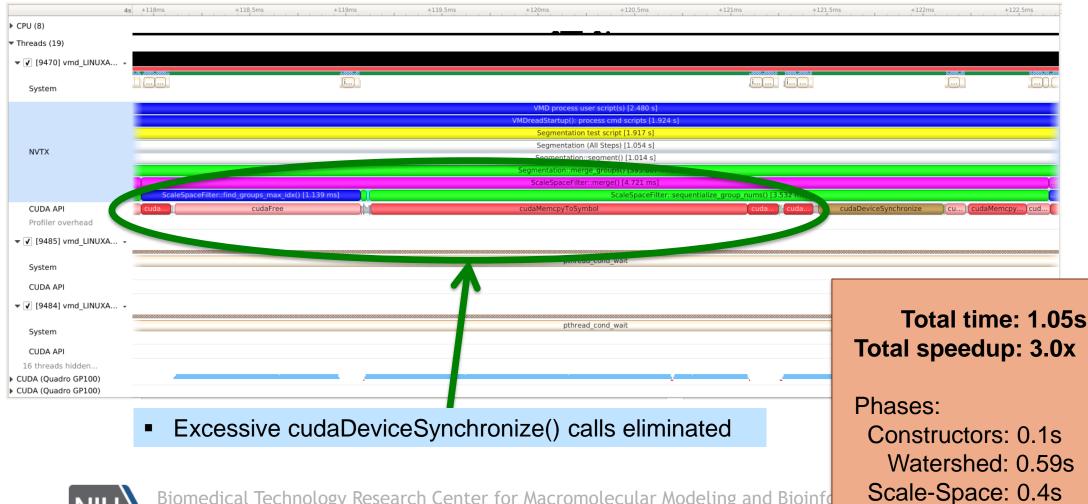
	4s +151ms	+151.5ms	+152ms	+152.5ms	+153ms	+153.5ms	+154ms	+154.5ms	+155ms	+155.5ms
▶ CPU (8)										
 Threads (19) 										
▼ ✔ [9425] vmd_LINUXA 、										
							()			· · · · · · · ·
System										
					VMD process user so					
					VMDreadStartup(): process					
					Segmentation test Segmentation (All S					
NVTX					Segmentation::segr					
					Segmentation::merge_gr					
					CooleCooleCliboration	/) [5 025 ma]				Sc
		ps_max_lax() [1.156 ms]		GaussianBlur::blur_cuda(Filter::sequentialize_group_nu		Sc
	cudaDeviceSynchror	nize cudaDeviceS cu	. cudaDeviceSynchron	nize cudaDeviceSynch	ronize cudaDeviceSyn	chronize cudaMemcpyT	oSy cuda	cudaDeviceSynchronize	cuda	udaDevice cud c
Profiler overhead										
▼ 🗸 [9441] vmd_LINUXA										
System					pthre 1_or	nd_wait				
System					pthre i_cai	nd_wait				
System CUDA API	800000000000000000000000000000000000000				pthre in set	nd_wait				
					pthre 1 cor	nd_wait				000000000000000000000000000000000000000
CUDA API ▼ ✔ [9440] vmd_LINUXA 、					pthread_cor	-				
CUDA API	-				000000000000000000000000000000000000000	-				
CUDA API ▼ ✔ [9440] vmd_LINUXA 、			000000000000000000000000000000000000000		000000000000000000000000000000000000000	-				
CUDA API V [9440] vmd_LINUXA V System CUDA API 16 threads hidden			000000000000000000000000000000000000000		000000000000000000000000000000000000000	-				
CUDA API V [9440] vmd_LINUXA V System CUDA API 16 threads hidden VUDA (Quadro GP100)					000000000000000000000000000000000000000	-				
CUDA API V [9440] vmd_LINUXA V System CUDA API 16 threads hidden		sive sync		, ,	pt read_con	nd_wait				· · ·

Many calls to cudaDeviceSynchronize(), checking error status, etc.





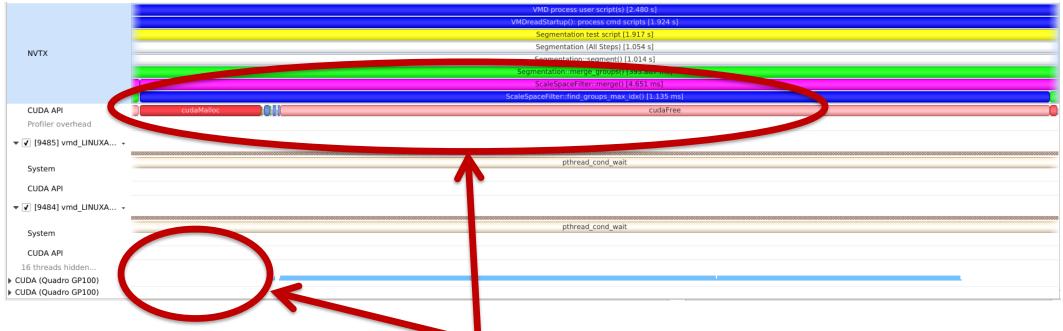
6: DETAIL: STREAMLINED ERROR CHECKING



Biomedical Technology Research Center for Macromolecular Modeling and Bioinfo Beckman Institute, University of Illinois at Urbana-Champaign - www.ks.uiuc

Other: -

6: DETAIL: ITERATED MALLOC/FREE

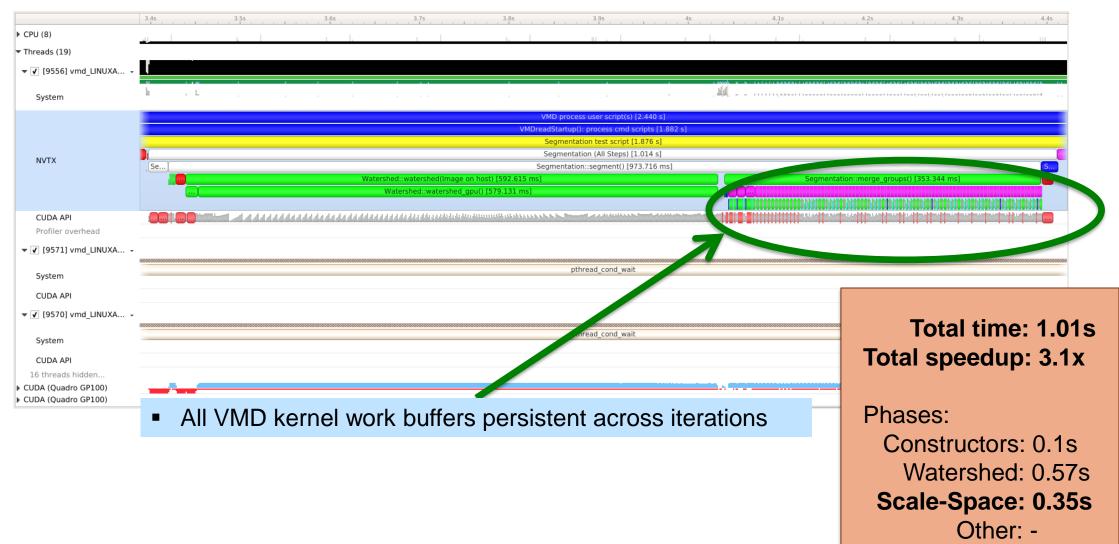


- Several areas of trace show CUDA malloc/free calls in iterative algorithm phase
- (Re)allocation APIs create gaps in GPU execution stream

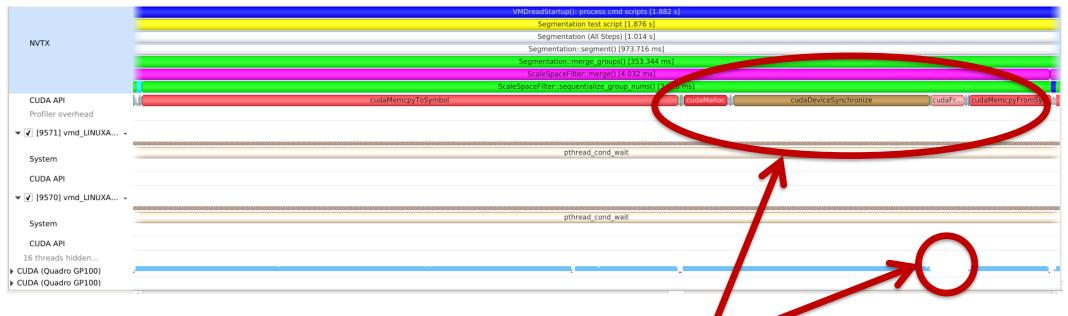




7: MADE WORK BUFFERS PERSISTENT



7: DETAIL: ... EXCEPT THRUST SCAN()

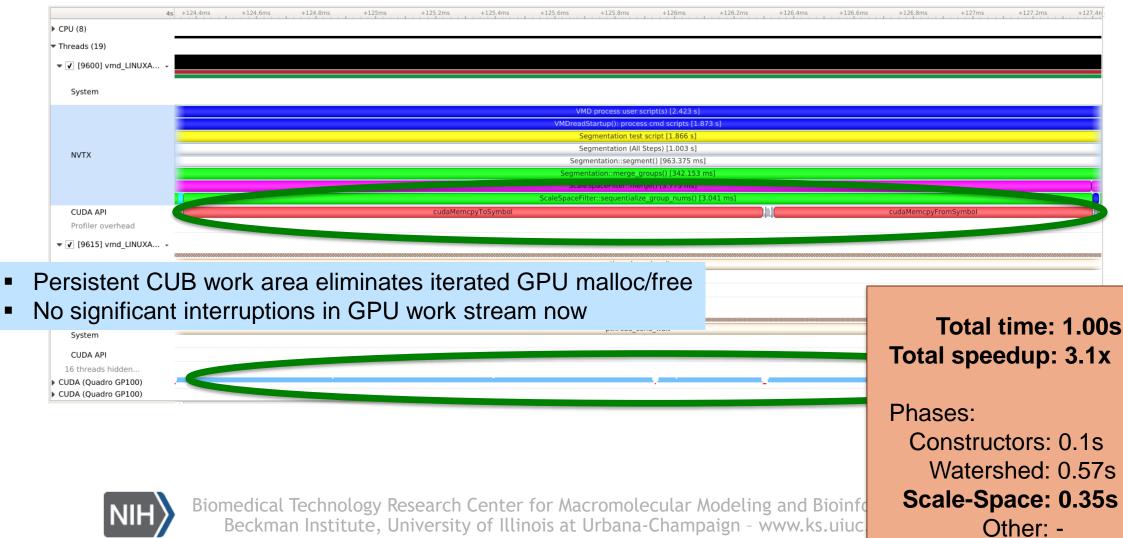


- Thrust scan performs GPU malloc/free
- Allocations disrupt GPU work stream slightly
- Can use special allocation scheme or use CUB instead





8: DETAIL: CUB SCAN PERSISTENT WORK BUFFERS



9: DETAIL: USE OF CUDA ASYNC APIS

	4s +73.391ms	+73.392ms	+73.393ms	+73.394ms	+73.395ms	+73.396ms	+73.397ms	+73.398ms	+73.399ms	+73.4ms	+73.401ms	+73.402ms	+73.403ms	+73.404ms	+73.405ms	+73.406ms
▶ CPU (8)																
 Threads (19) 																
▼ ✔ [9644] vmd_LINUXA	. •															
System																
									er script(s) [2.41							
	_						VMD	readStartup(): pr	cess cmd script test script [1.824							
	-								(All Steps) [1.014							
NVTX								Segmentation::se								
								hed::watershed(
							W	atershed::waters	hed_gpu() [571.4	10 ms)						
CUDA API		cudaMemcpyToS	wmbolAsync					unda	te_kernel				cuda	MemcpyFromSym	bol	
Profiler overhead			, ,					apao								
▼ 🗸 [9659] vmd_LINUXA																
System								pthrea	d_cond_wait							
CUDA API																
▼ ✔ [9658] vmd_LINUXA			000000000000000000000000000000000000000		****		000000000000000000000000000000000000000			005000000000000000000000000000000000000			000000000000000000000000000000000000000			000000000000000000000000000000000000000
System								pthrea	d_cond_wait							
CUDA API																
16 threads hidden																
 CUDA (Quadro GP100) CUDA (Quadro GP100) 									-					_		
F CODA (Quadro GF100)	• Us	se of	cudaN	/lemci	pyToS	Symb	olAsy	ync()	allows	S CPL	J to e	nque	ue su	bsequ	ient	

kernel and result copy-back efficiently while first copy is still running





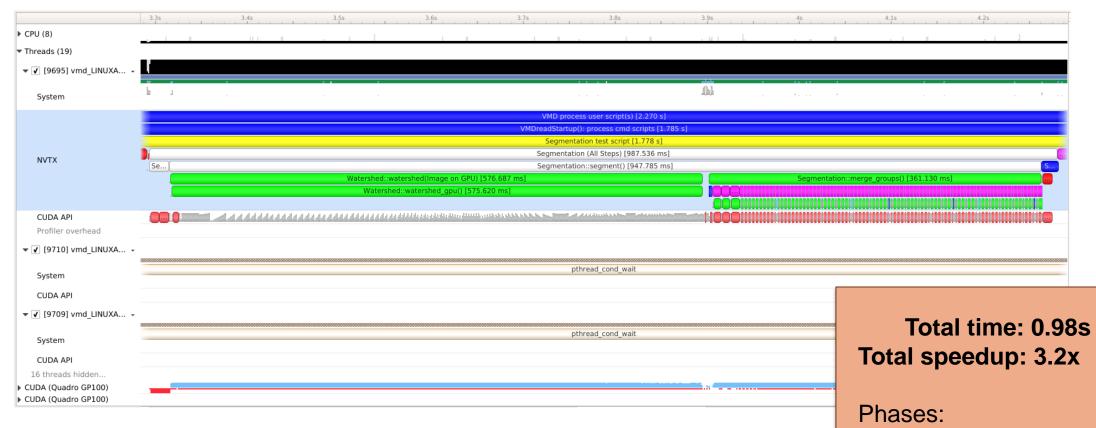
9: DETAIL: CONSTRUCTOR HOST-GPU COPY

	3s +455ms	+460ms	+465ms	+470ms	+475ms	+480ms	+485ms	+490ms	+495ms	+500ms +505ms	+510ms
▶ CPU (8)											
Threads (19)	-										
▼											
										100.	lance e
System	se)									Ш	JUMa J
						VMD proces	s user script(s) (2.413	sl			
							: process cmd scripts [
						Segmentat	ion test script [1,824 e	-1			
NVTX	V					Sec	(All Steps) [1.0)14 s]			
		Segm	entation::Segmentat	ion(MAP) [22.351 m	s]				Segmentation::segment()		
							CUDAGaussianBlur::c	copy_array_from_	gpu() [17.756 ms]	Watershed::watershed(Im	age on host) [584.570 ms] evice() [13.153 ms] [Watersh
										CODAwatersned::init_gpu_on_d	evice() [15.155 ms] watersh
CUDA API		cudaMemcpyA	sync	cudal	MemcpyAsync					cudaMemcpyAsync	c cu c
Profiler overhead											
▼ ▼ [9659] vmd LINUXA											
• • [9059] VIIId_LINUXA	•						000000000000000000000000000000000000000			*****	000000000000000000000000000000000000000
System						pth	read_cond_wait				
CUDA API				=	Extra co	pov: side	e effect	of a (C++ class	constructor	
▼ ✔ [9658] vmd_LINUXA	•										
	000000000000000000000000000000000000000				Eliminat	e via tir	nv refac	ctoring			
System							ly lolad				
CUDA API											
16 threads hidden											
CUDA (Quadro GP100)											
CUDA (Quadro GP100)											





10: FINAL RESULT



Constructors: 0.03s Watershed: 0.57s Scale-Space: 0.36s Other: -

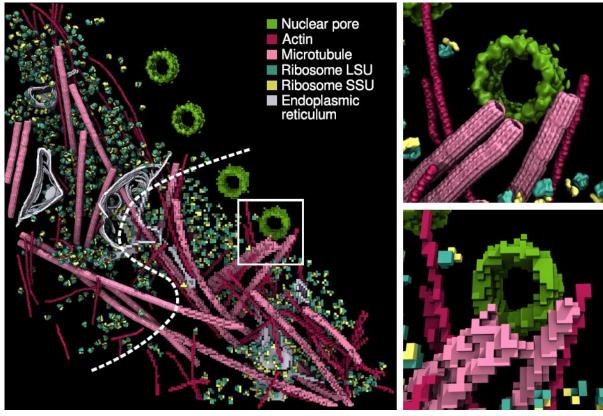
VMD CRYO-EM SEGMENTATION: LESSONS LEARNED

- Nsight Sytems helped identify unintentional copies caused by indirect side-effects of C++ class designs
- Demonstrates the value of applying profiling tool during ongoing algorithm development
- Final performance result on Quadro GP100 is 3.2x faster
- Speedup on Tesla V100 (Volta) is even more dramatic:
 - Initial runtime 2.66 seconds
 - Final optimized runtime: 0.64 seconds, 4.1x faster
- VMD GPU image segmentation is now 12x faster than competing tools



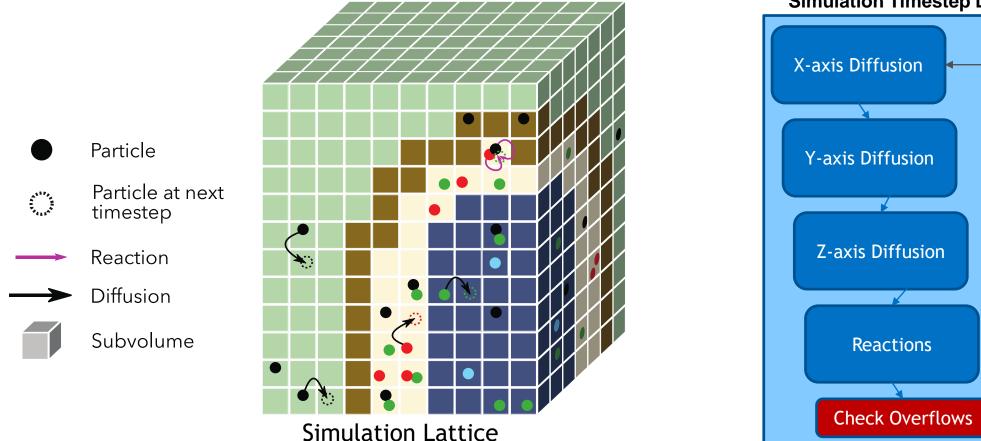
LATTICE MICROBES DESCRIPTION

Simulate cell dynamics on biologically relevant timescales using a lattice-based model



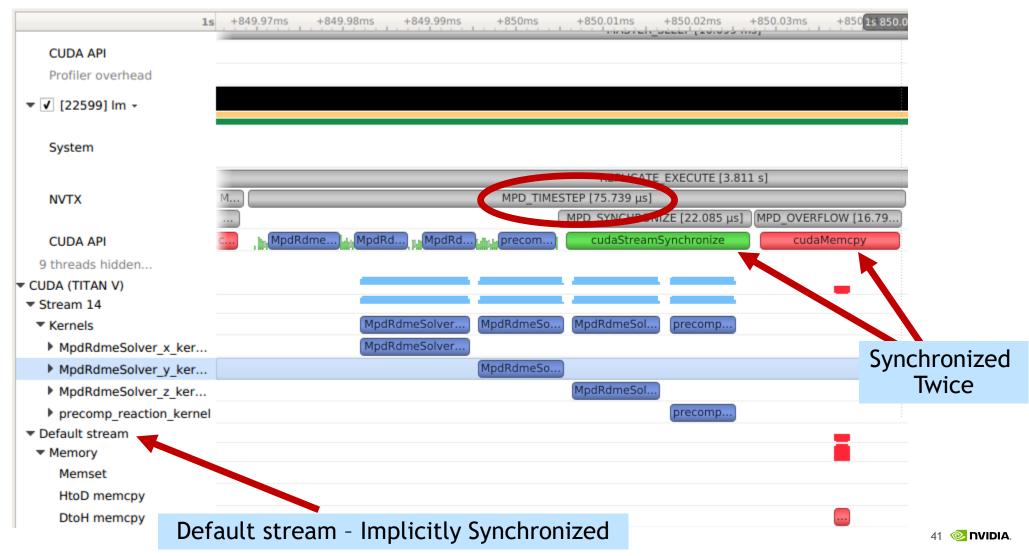
Earnest, et al. J. Physical Chemistry B, 121(15): 3871-3881, 2017.

LATTICE MICROBES SIMULATION

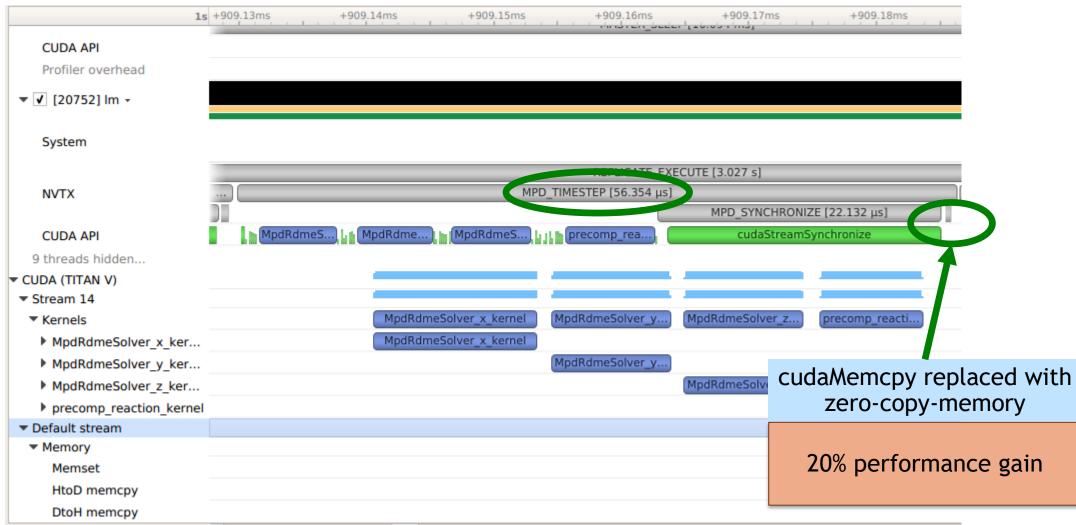


Simulation Timestep Loop

DEFAULT STREAM SYNCHRONIZATION

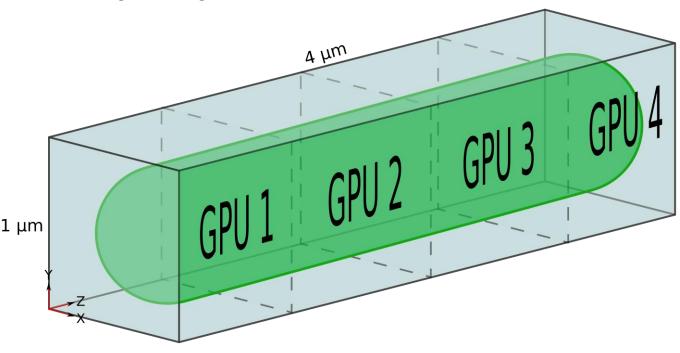


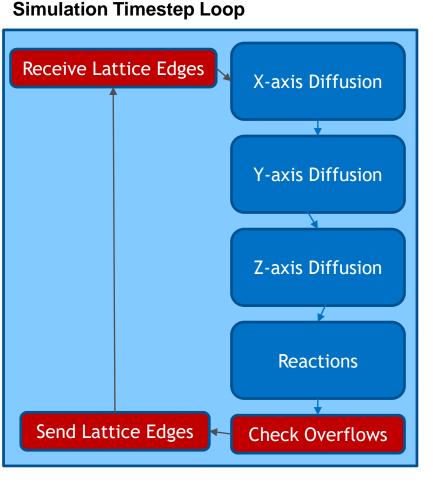
DEFAULT STREAM SYNCHRONIZATION 2



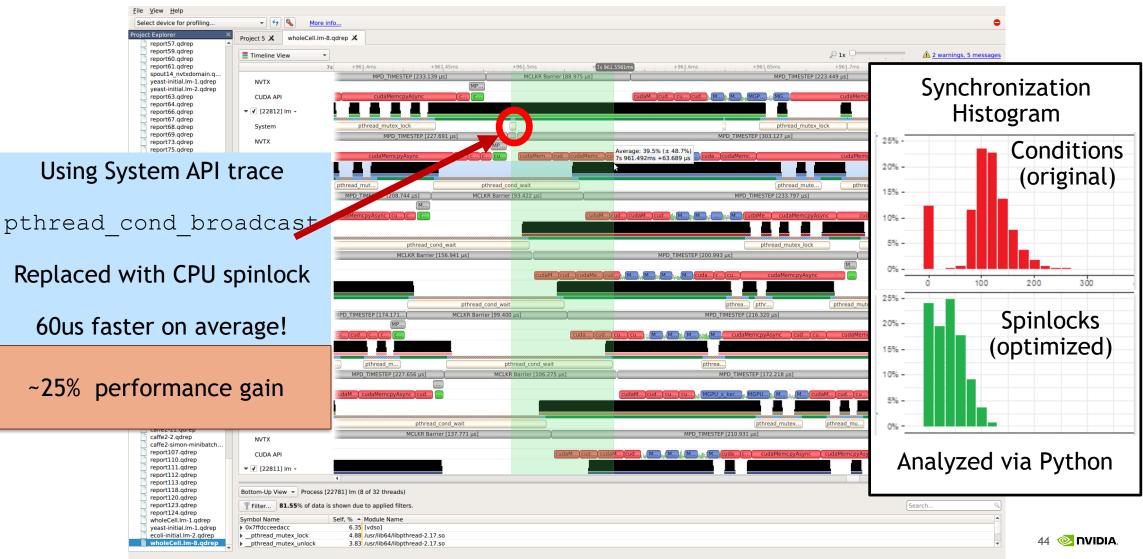
LATTICE MICROBES MULTI-GPU SIMULATION

- Divide the cell into chunks for each GPU to process
- Communicate particles on the edge of each volume to neighboring GPU



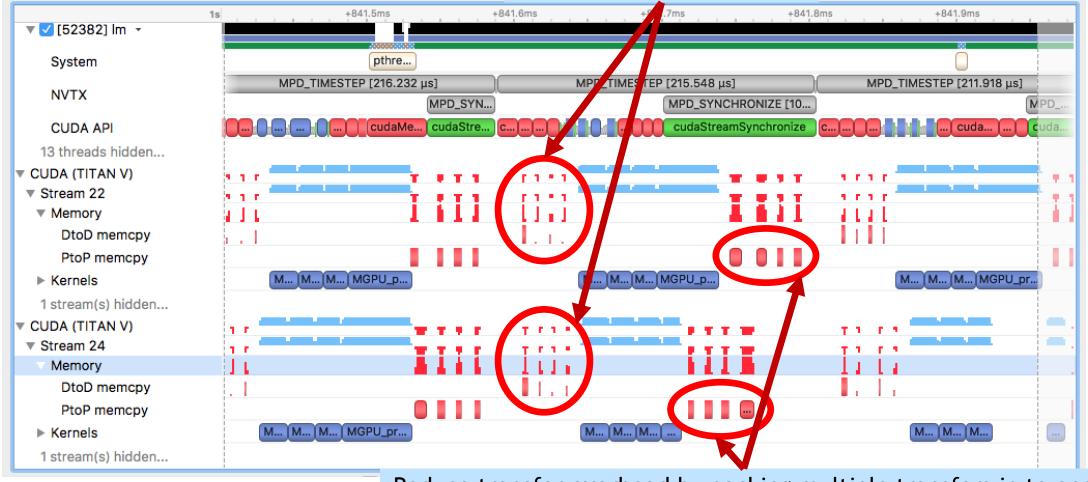


HOST THREAD PARALLEL OPTIMIZATION



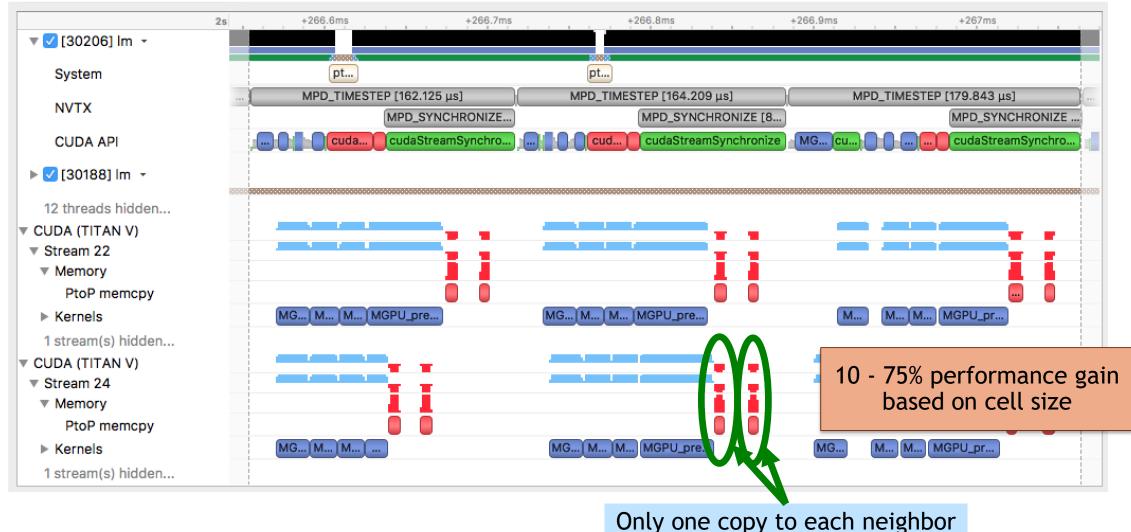
INTER-GPU TRANSFER IMPROVEMENTS

Eliminate small D2D Copies

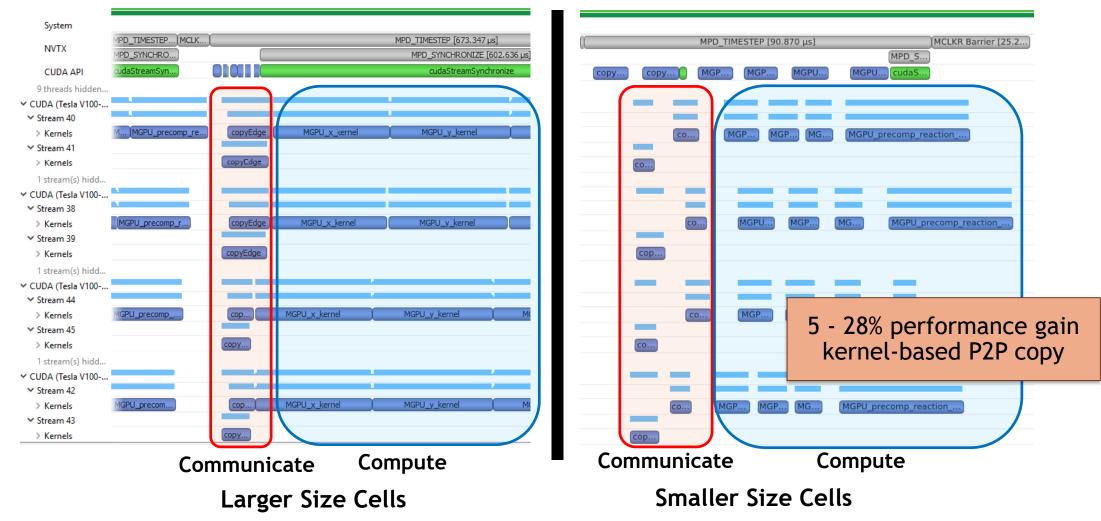


Reduce transfer overhead by packing multiple transfers in to one

INTER-GPU TRANSFER IMPROVEMENTS



4-WAY DGX INTER-GPU TRANSFERS WITH NVLINK



A kernel directly accessing remote lattice via P2P copies can achieve concurrent bidirectional transfers

COMMON OPTIMIZATION OPPORTUNITIES

► <u>CPU</u>

- Thread synchronization
- Algorithm bottlenecks starve the GPUs

Multi GPU

- Communication between GPUs
- Lack of Stream Overlap in memory management, kernel execution

Single GPU

- Memory operations blocking, serial, unnecessary
- Too much synchronization device, context, stream, default stream, implicit
- CPU GPU Overlap avoid excessive communication

TOOL COMPARISON

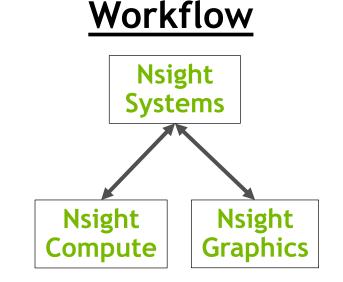
	NVIDIA © Nsight ™ Systems	NVIDIA© Nsight™ Compute	NVIDIA© Visual Profiler	Intel © VTune ™ Amplifier	Linux perf OProfile
Target OS	Linux	Linux, Windows	Linux, Mac, Windows	Linux, Windows	Linux
GPUs	Pascal, Volta, Future	Pascal, Volta, Future	Kepler, Maxwell, Pascal, Volta, Future	None	None
CPUs	x86_64	x86_64	x86, x86_64, Power	x86, x86_64	x86, x86_64, Power
Trace	NVTX, CUDA, OpenGL, CuDNN, CuBLAS, System	NVTX, CUDA	MPI, CUDA, OpenACC	MPI, ITT	Kernel
PC Sampling	High Speed	No	Yes	High Speed	High Speed
UVM, NVLINK, Power,Thermal	Future		Yes	No	No
Src Code View	Νο	Yes	Yes	Yes	No
Compare Sessions	Νο	Yes	Yes	Yes	No

NSIGHT SYSTEMS

Visit us at the NVIDIA booth in the exhibit hall for a live demo!

- When can you get it?
 - Soon. Fixing the last issues now.
- Where can you get it?
 - http://developer.nvidia.com/nsight-systems
- Questions/Requests/Comments?
 - <u>nsight-systems@nvidia.com</u>

For Tegra-based systems Codeworks JetPack DriveInstall



NSIGHT SYSTEMS

Upcoming features:

- NVIDIA GPU Cloud (near future)
- Future GPUs
- Future CUDA Releases
- Windows targets
- Many more HPC and cluster features

DON'T MISS THESE PRESENTATIONS

S8481: CUDA Kernel Profiling: Deep-Dive Into NVIDIA's Next-Gen Tools (Thursday 11:00AM)

- **S8337:** NVIDIA SDK Manager Simplify Your Development Environment Setup (Wednesday 3:30 PM)
- **S8275:** Introducing NVIDIA's New Graphics Debugger (Wednesday 4:00 PM)
- S8665: VMD: Biomolecular Visualization from Atoms to Cells Using Ray Tracing, Rasterization, and VR (Thursday 11:00AM)
- **S8709:** Accelerating Molecular Modeling Tasks on Desktop and Pre-Exascale Supercomputers (Monday 4:00PM)

Show floor *demos* available:

Tuesday 11-1 and 5:30-7:30; Wednesday 12-2 and 5-7; Thursday 12-2

Q&A



BACKUP

COMMAND LINE INTERFACE

usage: sp profile [<args>] [application] [<application args>]

args:

-y, --delay= Collection start delay in seconds. Default is 0.

-d, --duration=

Collection duration in seconds. Default is 10 seconds.

-e, --env-var=

Set environment variable(s) for application process to be launched. Environment variable(s) should be defined as 'A=B'. Multiple environment variables can be specified as 'A=B,C=D'

-h, --help

This help message.

-n, --inherit-environment=

Inherit environment variables. Possible values are 'true' or 'false'. Default is 'true'.

-o, --output=

Output QDSTRM filename. Default is report#.qdstrm.

-s, --sample=

Select the entity to sample. Possible values are 'cpu' or 'none'. Select 'none' to disable sampling. Default is 'cpu'.

-b, --backtrace=

Select the backtrace method to use while sampling. Possible values are 'lbr', 'dwarf', 'fp', or 'none'. Select 'none' to disable backtrace collection. Default is 'lbr'.

-w, --show-output=

If true, send target process' stdout and stderr streams to both the console and stdout/stderr files which are added to the QDSTRM file. If false, only send target process stdout and stderr streams to the stdout/stderr files which are added to the QDSTRM file. Possible values are 'true' or 'false'. Default is 'false'.

-t, --trace=

```
Select the API(s) to trace. Possible values are 'cublas', 'cuda', 'cudnn', 'nvtx', 'opengl', 'system', or 'none'.
Multiple APIs can be selected, separated by commas only (no spaces). If 'none' is selected, no APIs are traced.
Default is 'cuda,opengl,nvtx,system'.
```

