USING NSIGHT TOOLS TO OPTIMIZE THE NAMD MOLECULAR DYNAMICS SIMULATION PROGRAM

Robert (Bob) Knight – Software Engineer, NVIDIA

David Hardy – Senior Research Programmer, U. Illinois at Urbana Champaign

March 19, 2019

Fidelity INVESTMENTS

Financial Advisors | Why Register?

FIDELITY INSTITUTIO

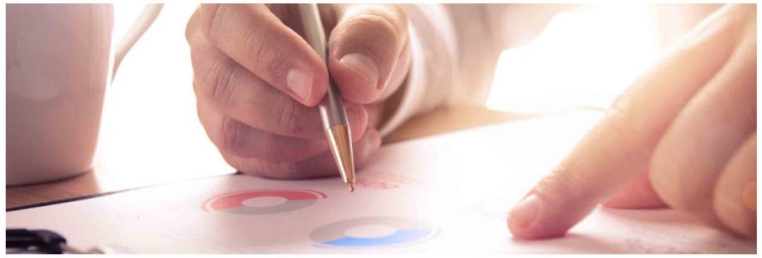
Home | Funds & Products | Defined Contribution | Investing

Markets & Economy > Viewpoints & White Papers

Rebalancing Your Investment Mix

Posted: 10/24/2018 by Fidelity Viewpoints

Even during a strong stock market, you should evaluate risks and returns.



MENU | SEARCH | Kiplinger

TIMELINE Kiplinger's Retirement Report Turns 25

SLIDE SHOW 8 Tax Deductions Eliminated Under New Tax Law

SLIDE SHOW The Kip ETF 20: The 20 Best Cheap ETFs You Can Buy

ADVERTISEMENT

FINANCIAL PLANNING

How to Rebalance Your Portfolio

A risk of this long-running bull market: You may have more money in stocks than you should.

Illustration by Nigel Buchanan

By TOM PETRUNO, Contributing Writer
October 3, 2018

TD Ameritrade

The TICKER TAPE

Search Ticker Tape

Market News | Ir

Home > Investing > Investment Strategies > Portfolio Strategy

Mid-Year Tune-Up: Reassess, Realign, Rebalance?

June 15, 2017 | 6 min read

Morgan Stanley

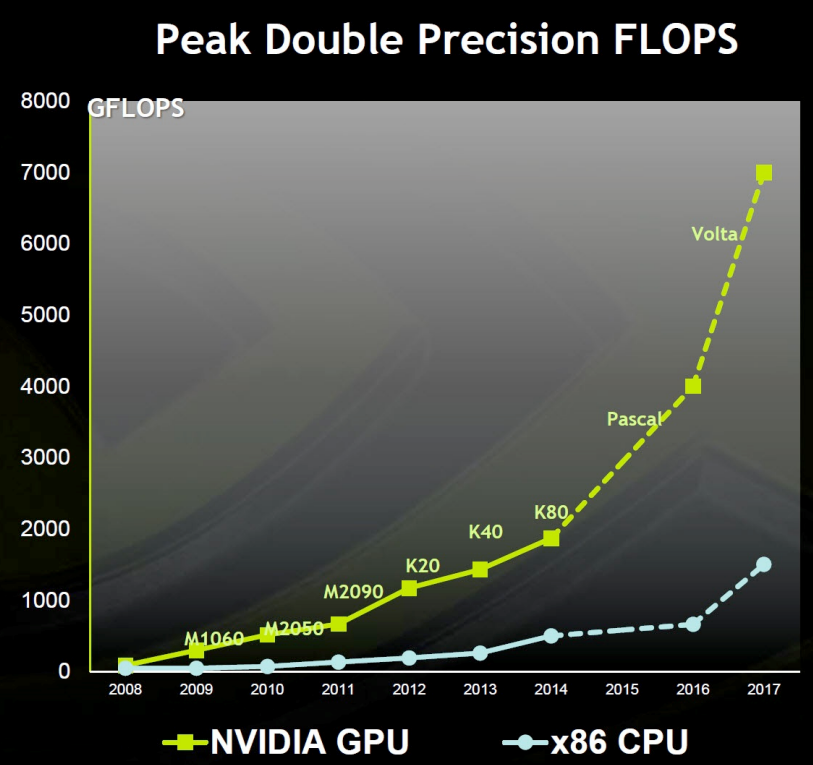
Ideas | People | What We Do | About U:

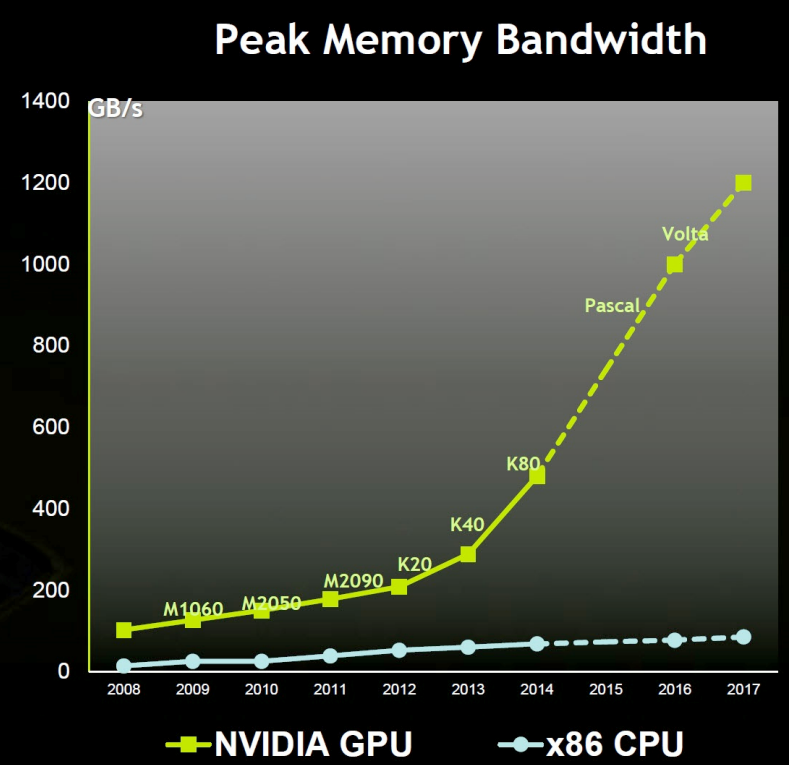
It's Time to Rebalance

NVIDIA

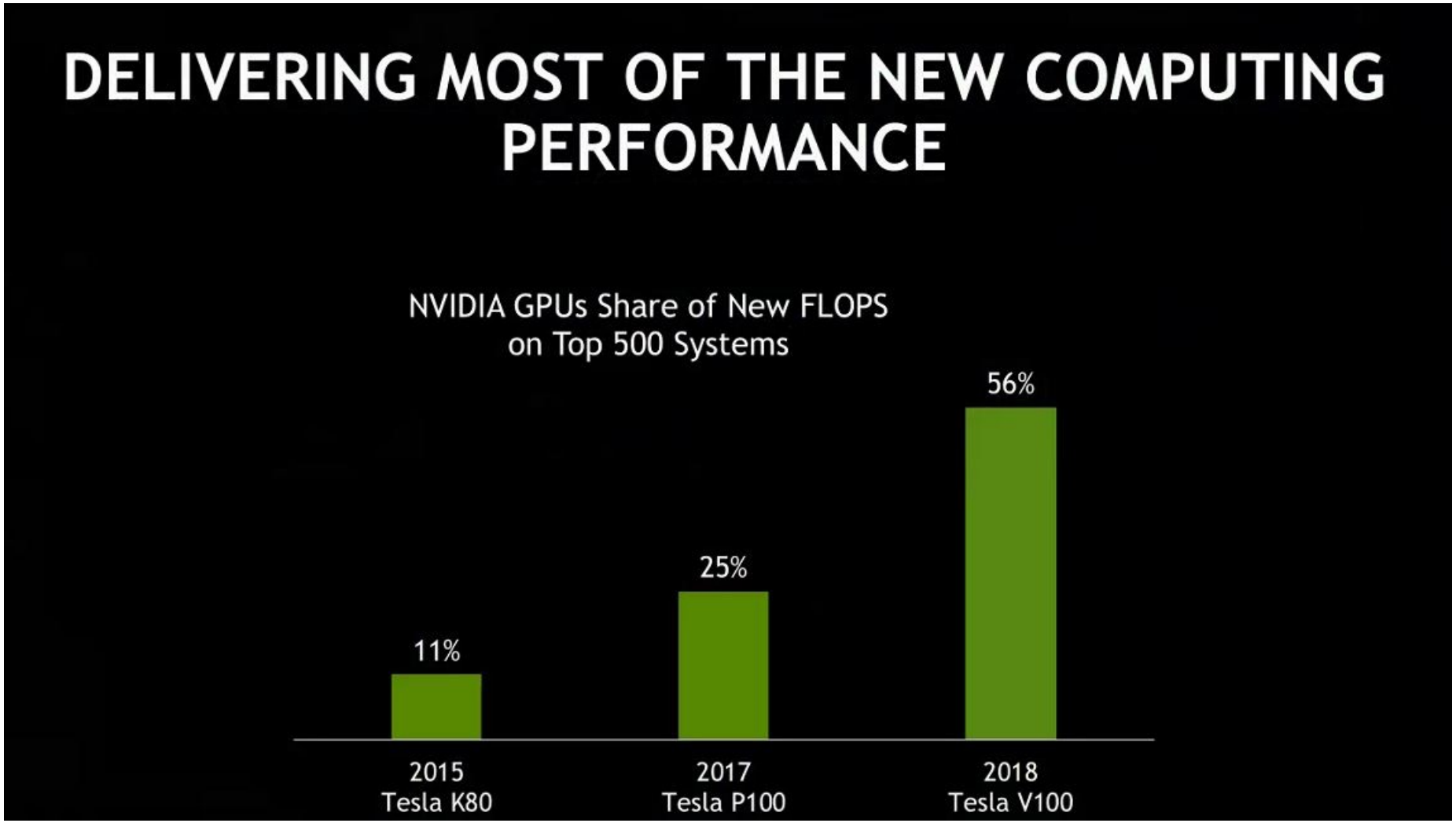
WHY REBALANCE?

GPU Motivation: Performance Trends





BECAUSE PERFORMANCE MATTERS

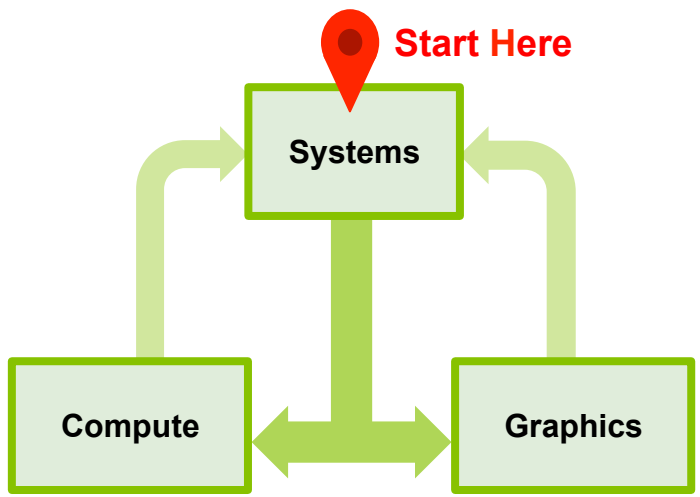


HOW?
NSIGHT SYSTEMS & NSIGHT COMPUTE

- Nsight Systems
  - Focus on the application's algorithm – a unique perspective
  - Rebalance your application's compute cycles across the system's CPUs & GPUs

- Nsight Compute
  - CUDA kernel profiling

Workflow



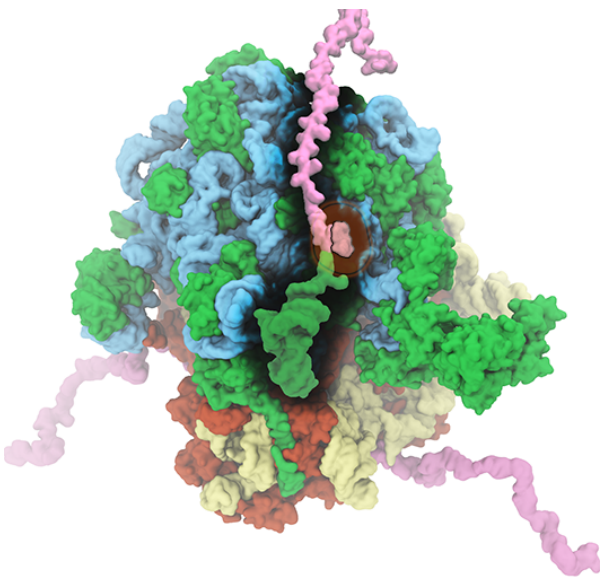
NAMD - NANOSCALE MOLECULAR DYNAMICS

- 25 years of NAMD
- 50,000+ Users
- Awards: 2002 Gordon Bell, 2012 Sidney Fernbach
- Solving Important Biophysics/Chemistry Problems
- Focused on scaling across GPUs – Biggest Bang for Their Compute $

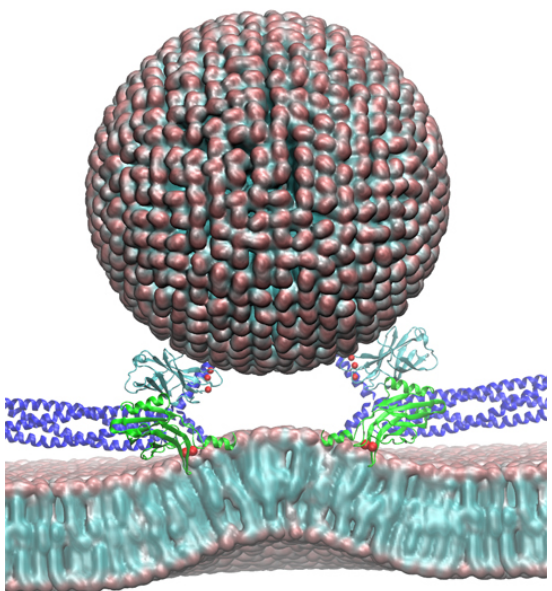
NAMD & VISUAL MOLECULAR DYNAMICS COMPUTATIONAL MICROSCOPE

Enable researchers to investigate systems at the atomic scale

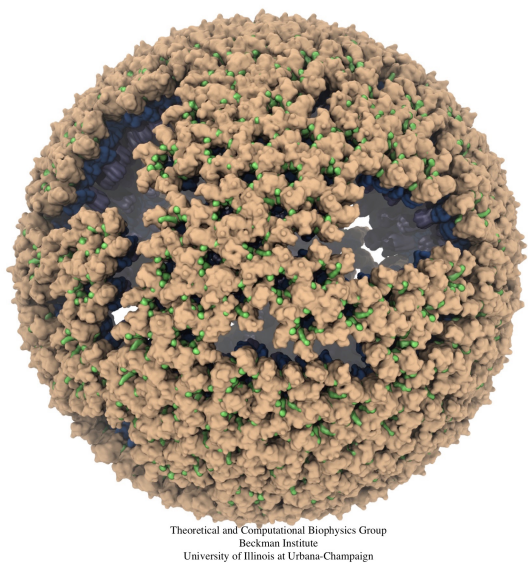
NAMD - molecular dynamics simulation
VMD - visualization, system preparation and analysis



Ribosome



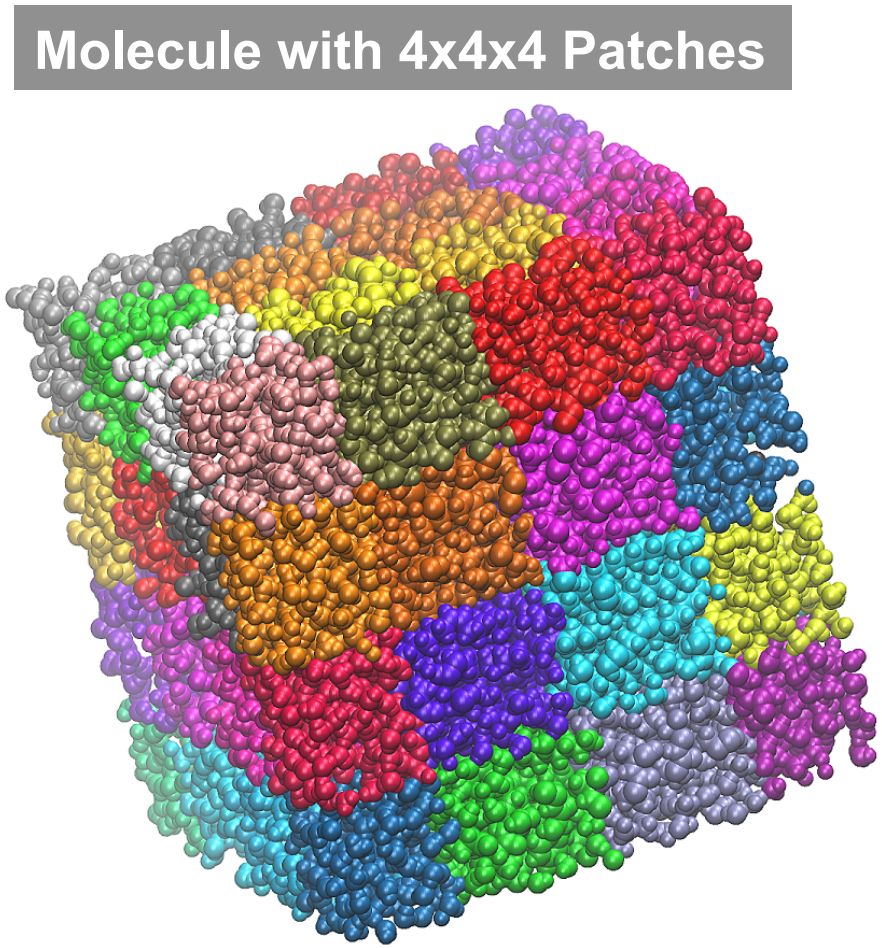
Neuron



Virus Capsid

NAMD OVERVIEW

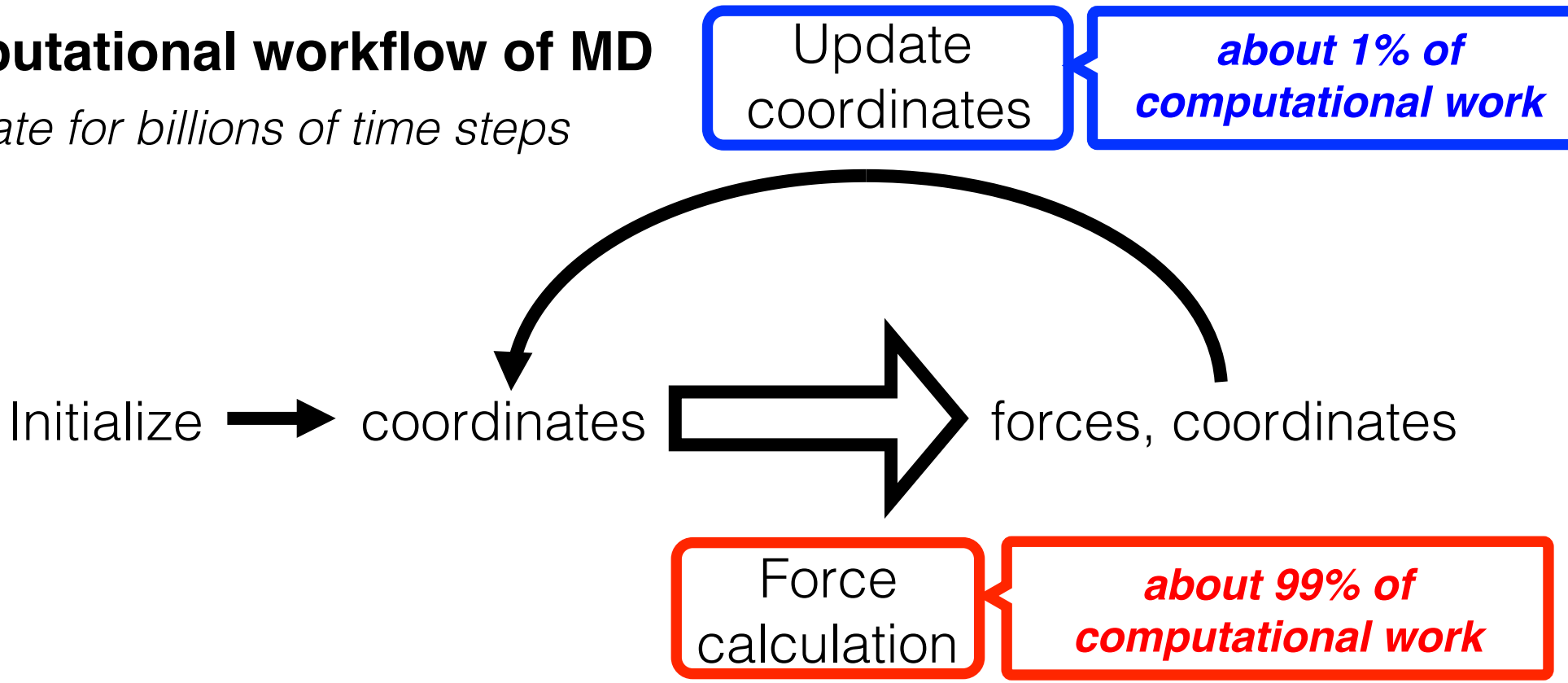
- Simulate the physical movement of atoms within a molecular system
- Atoms are organized in fixed volume *patches* within the system
- Forces that move atoms are calculated at each timestep
- After a *cycle* (e.g. 20 timesteps), atoms may migrate to an adjacent patch
- Performance measured as ns/day – the number of nanoseconds of simulation that could be calculated in one day of running the workload (higher is better)



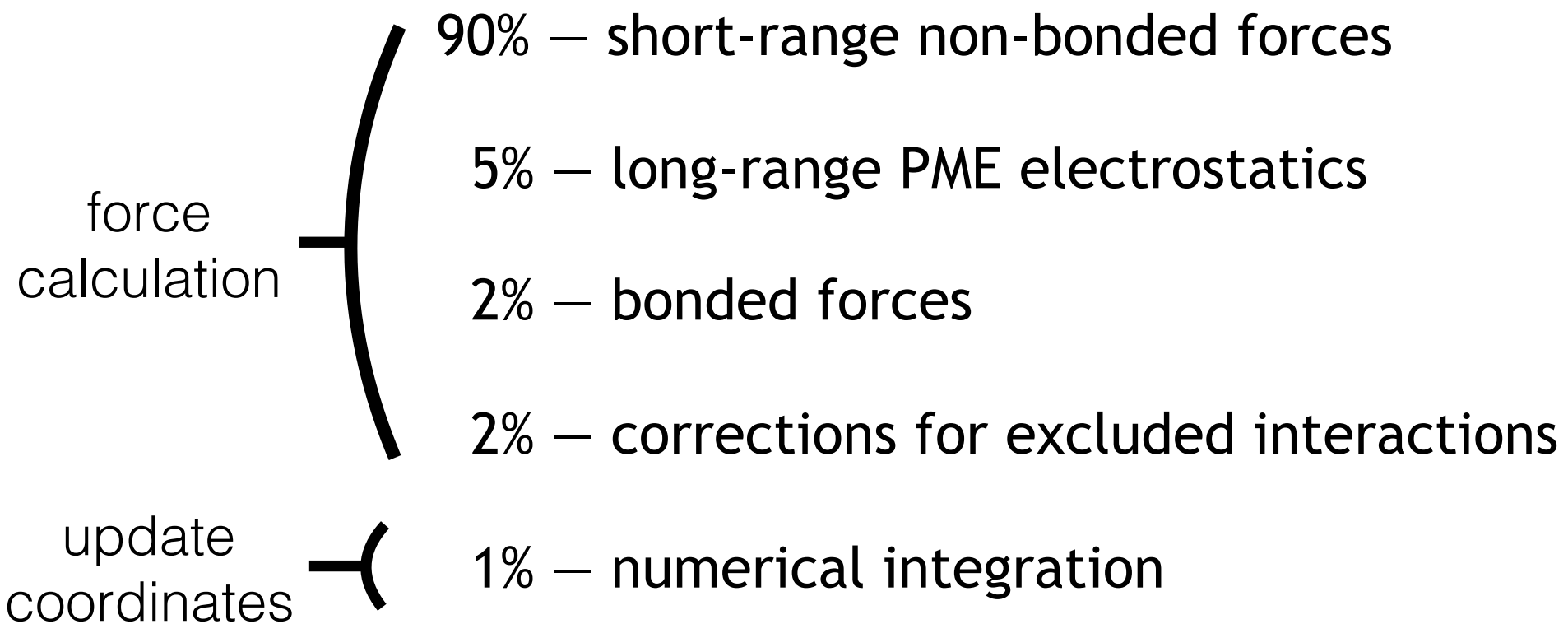
PARALLELISM IN MOLECULAR DYNAMICS LIMITED TO EACH TIMESTEP

Computational workflow of MD

Iterate for billions of time steps

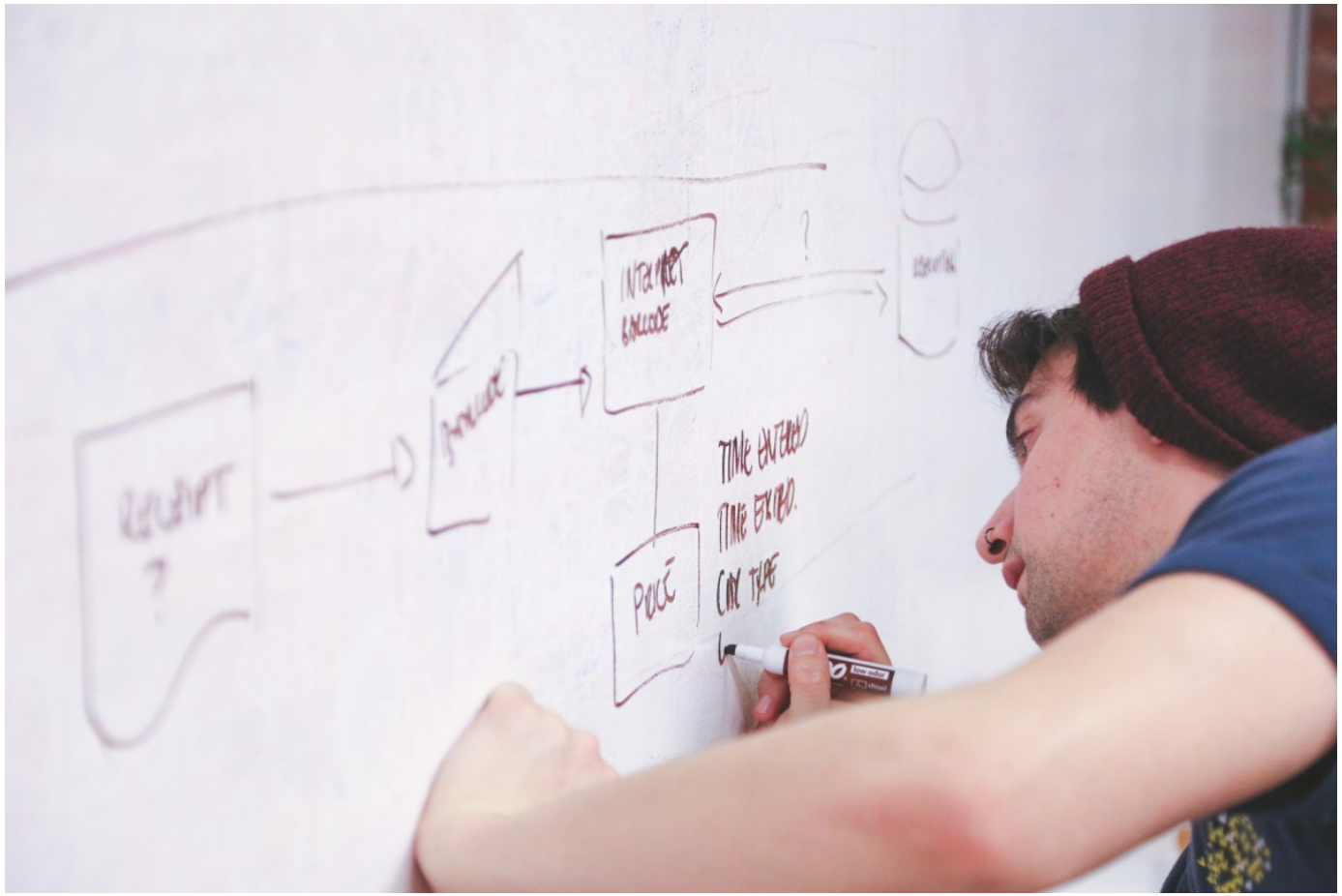


TIMESTEP COMPUTATIONAL FLOP COST

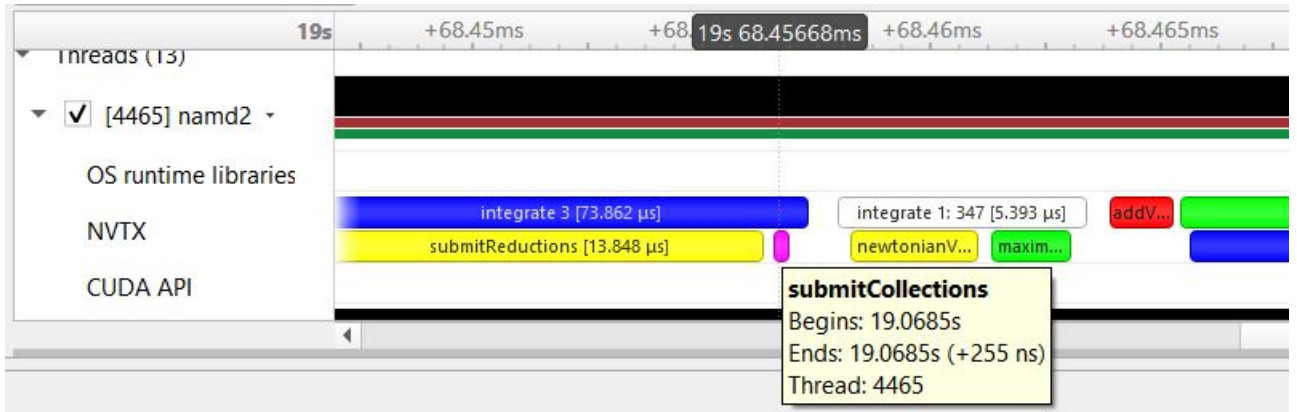


Start applying GPU acceleration to most expensive parts

BREAKDOWN A WORKLOAD



NVIDIA TOOLS EXTENSION (NTVX) API



- **Instrument application behavior**
  - Supported by all NVIDIA tools
- Insert markers, ranges
- Name resources
  - OS thread, CUDA runtime
- Define scope using *domains*

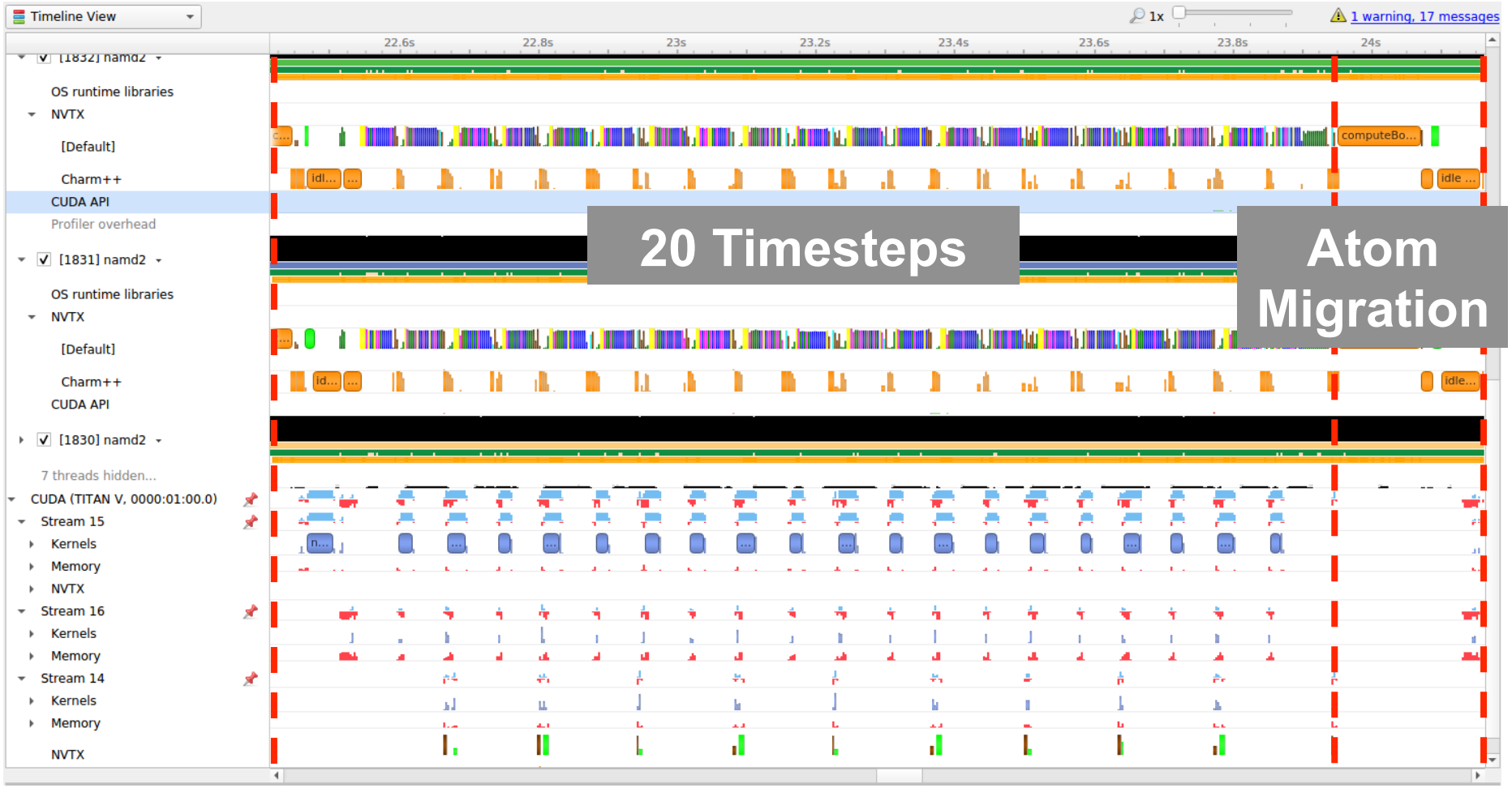
```
void Wait(int waitMilliseconds)
{
nvtxNameOsThread(“MAIN”);
nvtxRangePush(__FUNCTION__);
nvtxMark(>"Waiting...");
Sleep(waitMilliseconds);
nvtxRangePop();
}

int main(void)
{
nvtxNameOsThread("MAIN");
nvtxRangePush(__FUNCTION__);
Wait();
nvtxRangePop();
}
```

NAMD ALGORITHM SHOWN WITH NVTX

- Zoom Out
- Distinct Phases of NAMD Become Visible

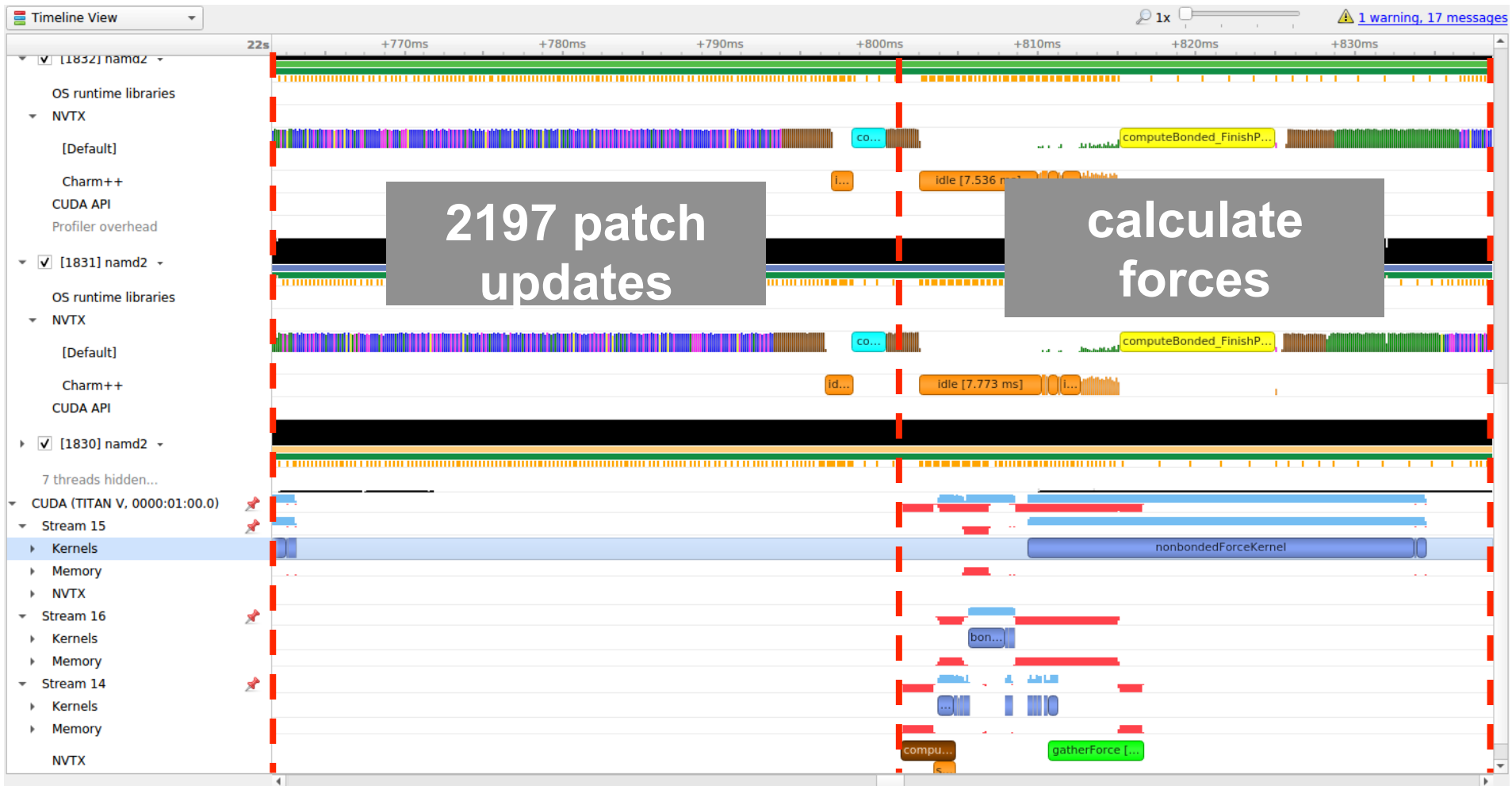
NAMD CYCLE

- Zoom In
- 20 Timesteps followed by Atom Migration

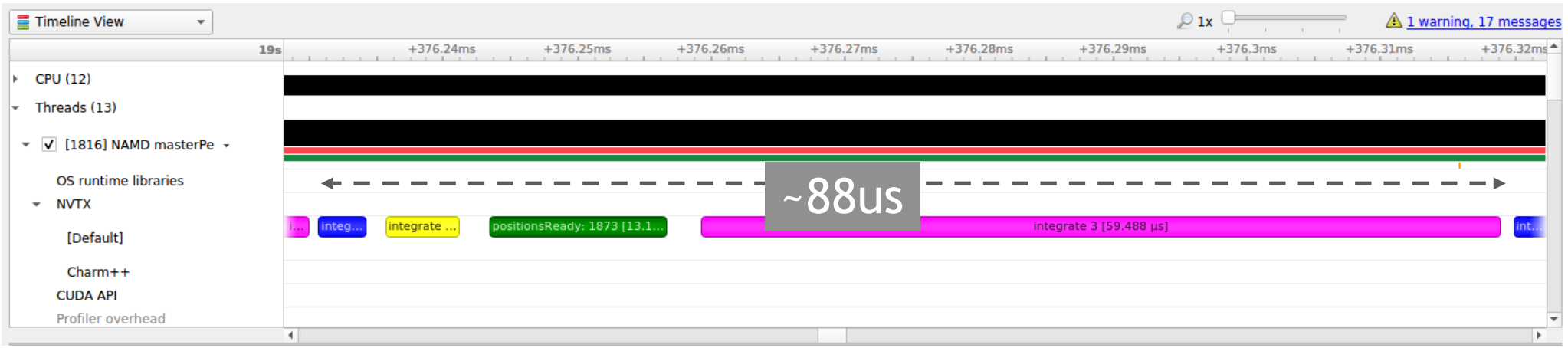


ONE NAMD TIMESTEP

- Zoom In



TIMESTEP - SINGLE PATCH



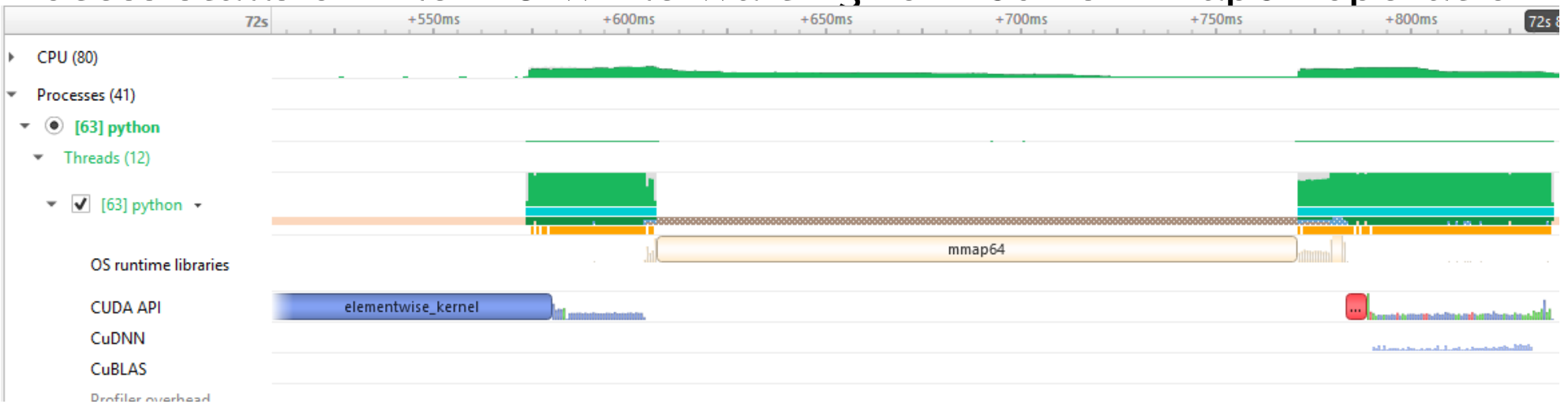
- Zoom In
- Patches are implemented as user-level threads

NSIGHT SYSTEMS

- User Instrumentation
- API Tracing
- Backtrace Collection
- Custom Data Mining
- Nsight Compute Integration

API TRACING

- Process stalls on file I/O while waiting for 160ms mmap64 operation



- Thread communicates over socket

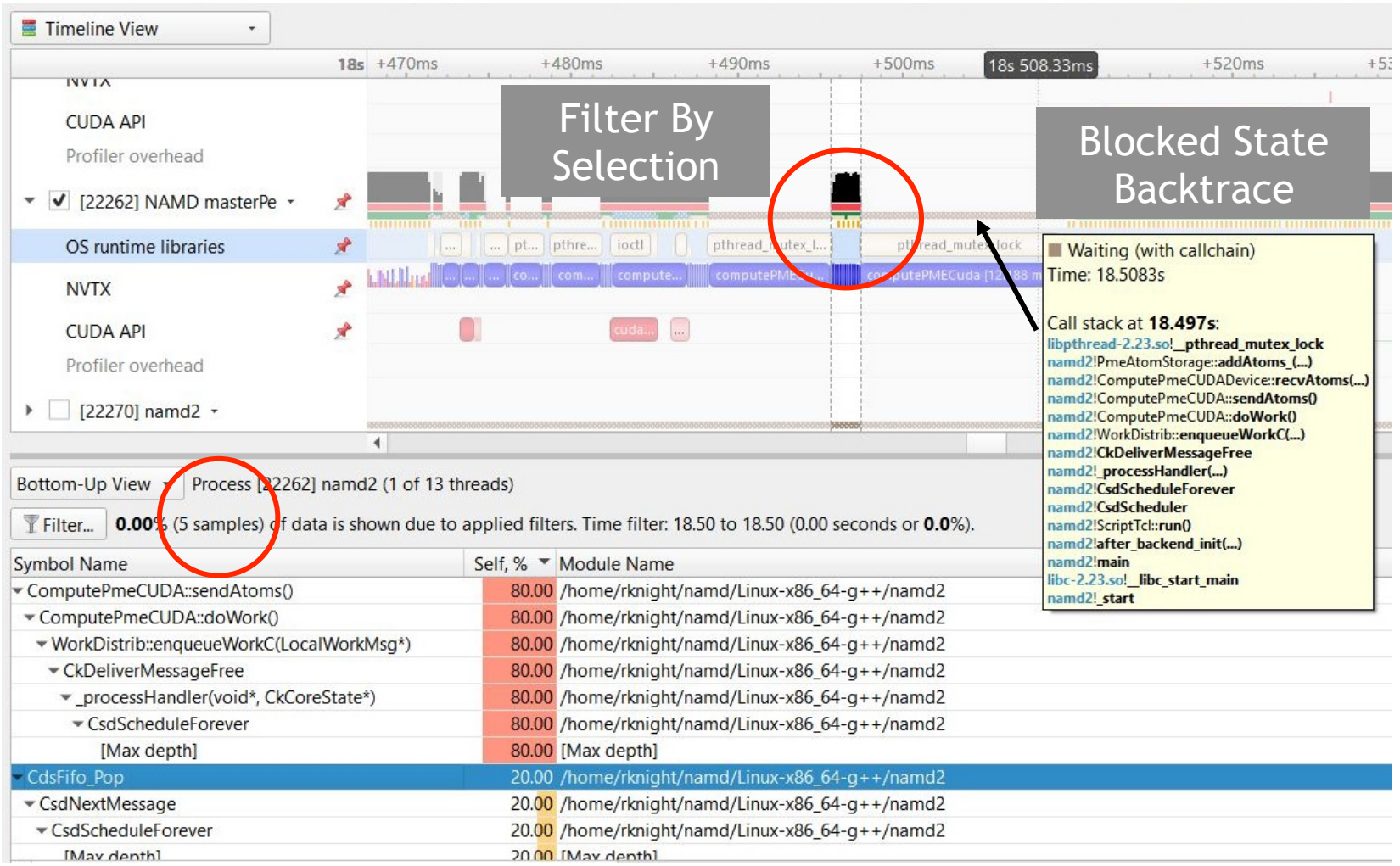
- APIs: CUDA, cuDNN, cuBLAS, OSRT (OS RunTime), OpenGL, *OpenACC, DirectX 12, Vulkan**

* Available in next release

SAMPLING DATA UNCOVERS CPU ACTIVITY

Filter By Selection
shows a specific thread's activity

Blocked State Backtrace
shows the path leading to an OS runtime library call



REPORT NAVIGATION DEMO

CUSTOM DATA MINING

Find Needles in Haystacks of Data

- nsys-exporter*

  QDREP->SQLite

- Use Cases
  - Outlier Discovery
  - Regression Analysis
  - Scripted Custom Report Generation

```
Kernel Statistics - all times in nanoseconds

minimum     maximum     average     kernel
----------  ----------  ----------  ----------------------------

1729557     5347138     2403882.7   nonbondedForceKernel
561821      631674      581409.6    batchTranspose_xyz_yzx_kernel
474173      574618      489148.1    batchTranspose_xyz_zxy_kernel
454621      593402      465637.6    spread_charge_kernel
393470      676060      420914.9    gather_force
52288       183455      116258.2    bondedForcesKernel
…

nonbondedForceKernel

duration    start         stream      context     GPU
----------  ----------  ----------  ----------  ----------

5347138     35453528745   133         7           1

5245934     39527523457   132         8           0

5076271     41048810842   132         8           0
```

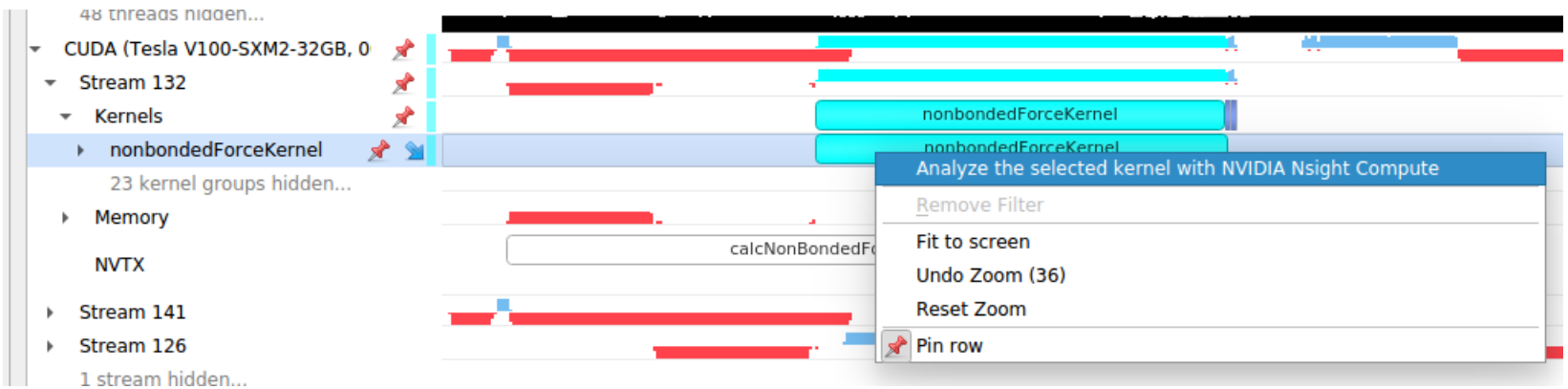
The longest nonBondedForceKernel is at 35.453s on GPU1, stream 133

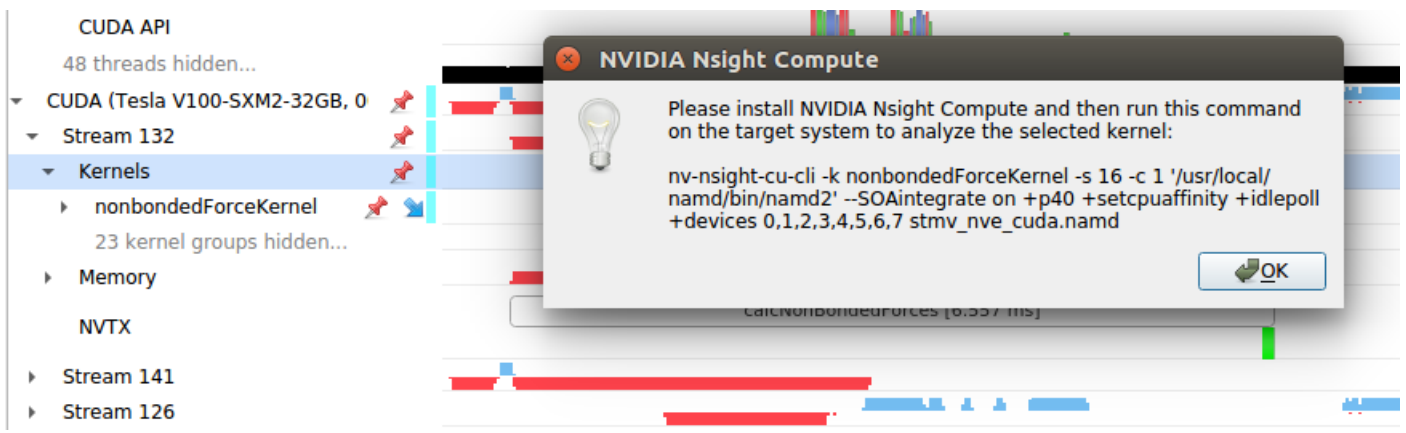
* Available in next release

KERNELSTATS SCRIPT

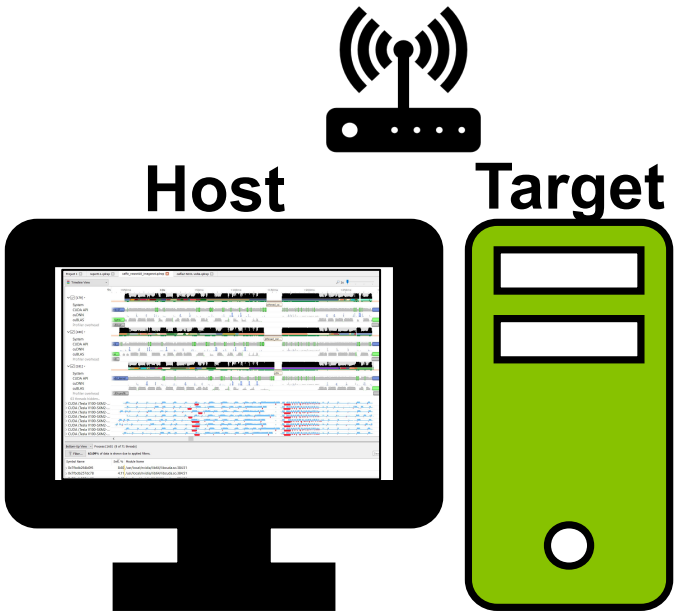

```
#!/bin/bash

sqlite3 $DB "ALTER TABLE CUPTI_ACTIVITY_KIND_KERNEL ADD COLUMN duration INT;"

sqlite3 $DB "UPDATE CUPTI_ACTIVITY_KIND_KERNEL SET duration = end-start;"

// add duration column, set duration column's value

sqlite3 $DB "CREATE TABLE kernelStats (shortName INTEGER, min INTEGER, max INTEGER, avg
INTEGER);"

sqlite3 $DB "INSERT INTO kernelStats SELECT shortName, min(duration), max(duration),
avg(duration) FROM CUPTI_ACTIVITY_KIND_KERNEL GROUP BY shortName;"

// create new table, insert kernel name IDs, min, max, avg into it

sqlite3 -column -header $DB SELECT min as minimum, max as maximum, round(avg,1) as
average, value as kernel FROM kernelStats INNER JOIN StringIds ON StringIds.id =
kernelStats.shortName ORDER BY avg DESC;"

// print formatted min, max, avg, and kernel name values, order by descending avg
```

NSIGHT COMPUTE INTEGRATION





DATA COLLECTION



Host – Target Remote Collection

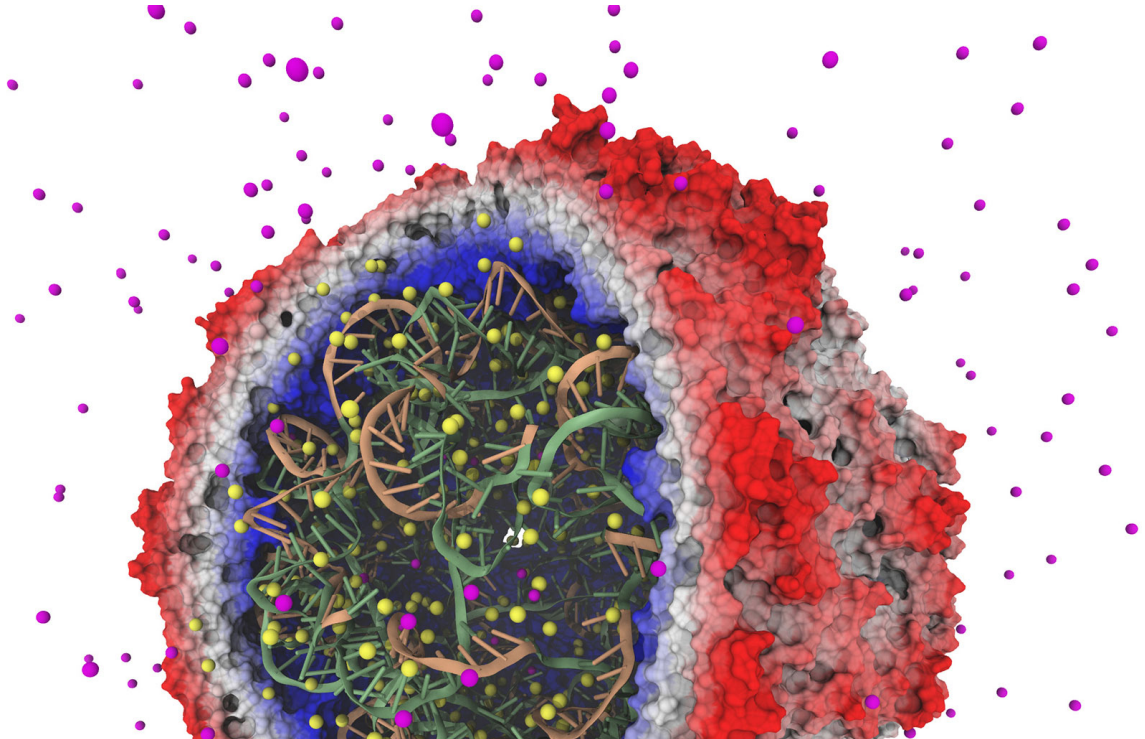
```
rknight@RKNIGHT-WS: ~
rknight@RKNIGHT-WS:~$ nsys launch -t cuda,nvtx,osrt --osrt-threshold 100000 -w true
/usr/local/namd/bin/namd2 --beginEventStep 1010 --endEventStep 1070 +p40 +setcpuaffi
nity +idlepoll +devices 0,1,2,3 ./stmv_nve_cuda.namd &
```

Command Line Interface

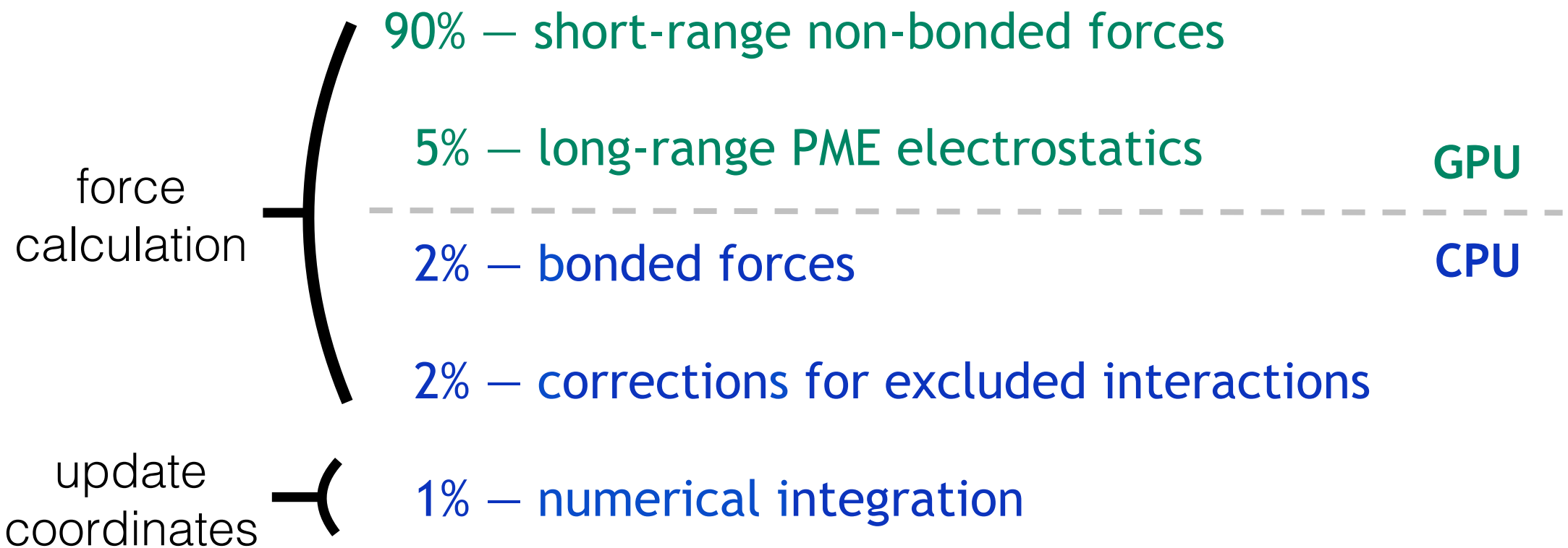
- No root access required
- Works in Docker containers
- Interactive Mode
- ***Supports cudaProfilerStart/Stop APIs***

What about NAMD...

Profiling Simulations of
Satellite Tobacco Mosaic Virus (STMV)
~ 1 Million Atoms

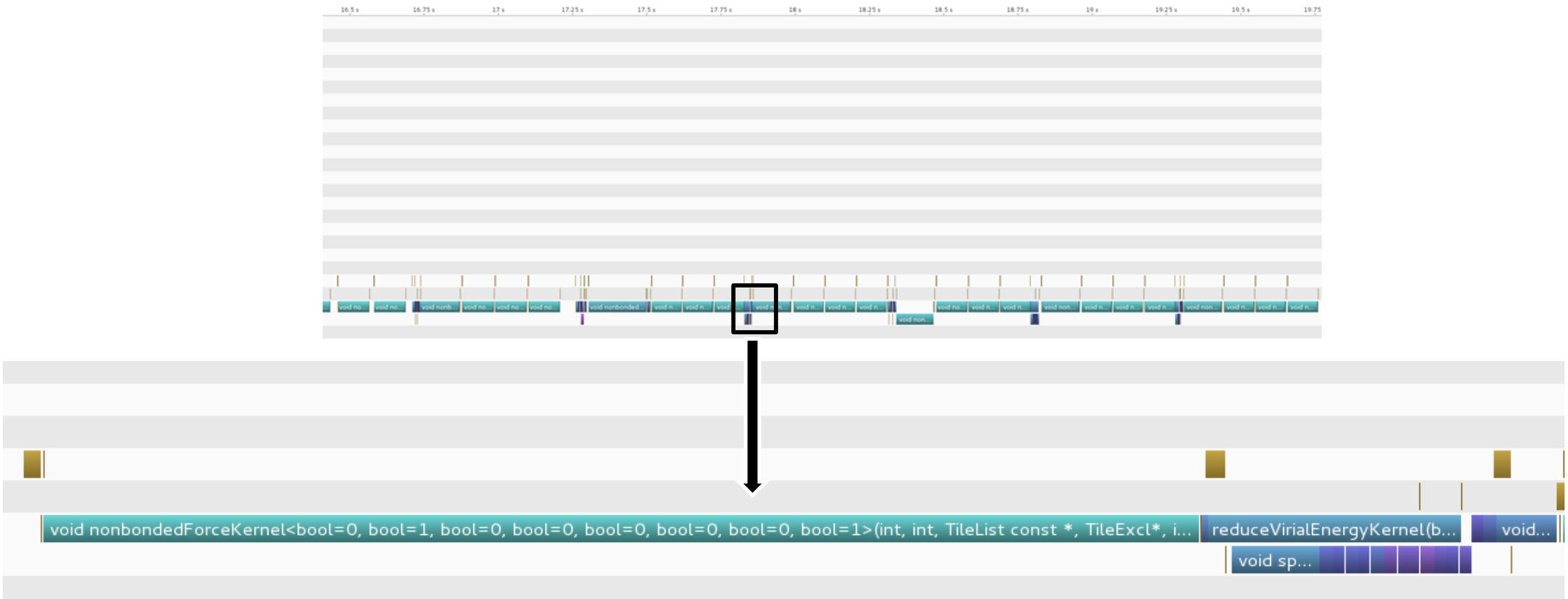


V2.12 TIMESTEP COMPUTATIONAL FLOP COST



NAMD V2.12

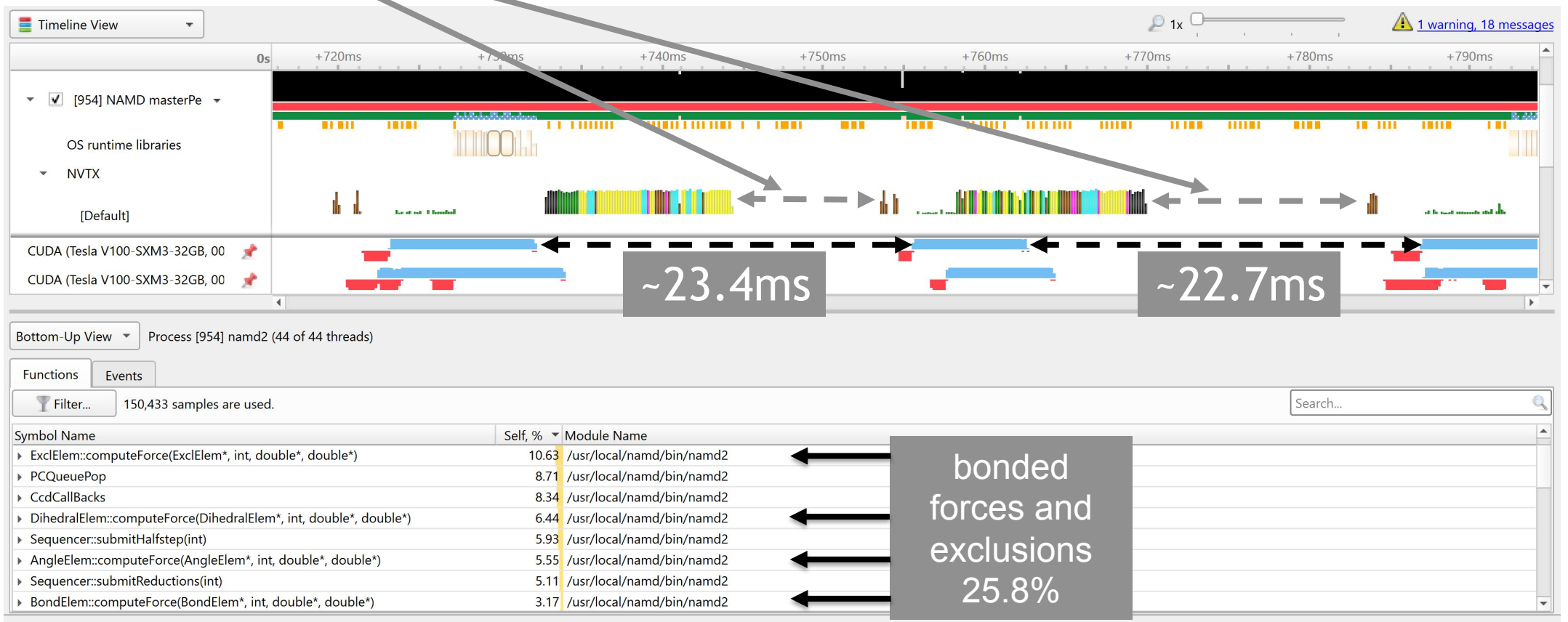
Profiling STMV with nvprof - Maxwell GPU Fully Loaded



NAMD V2.12

Volta GPU Severely Underloaded

bonded force and exclusion calculations



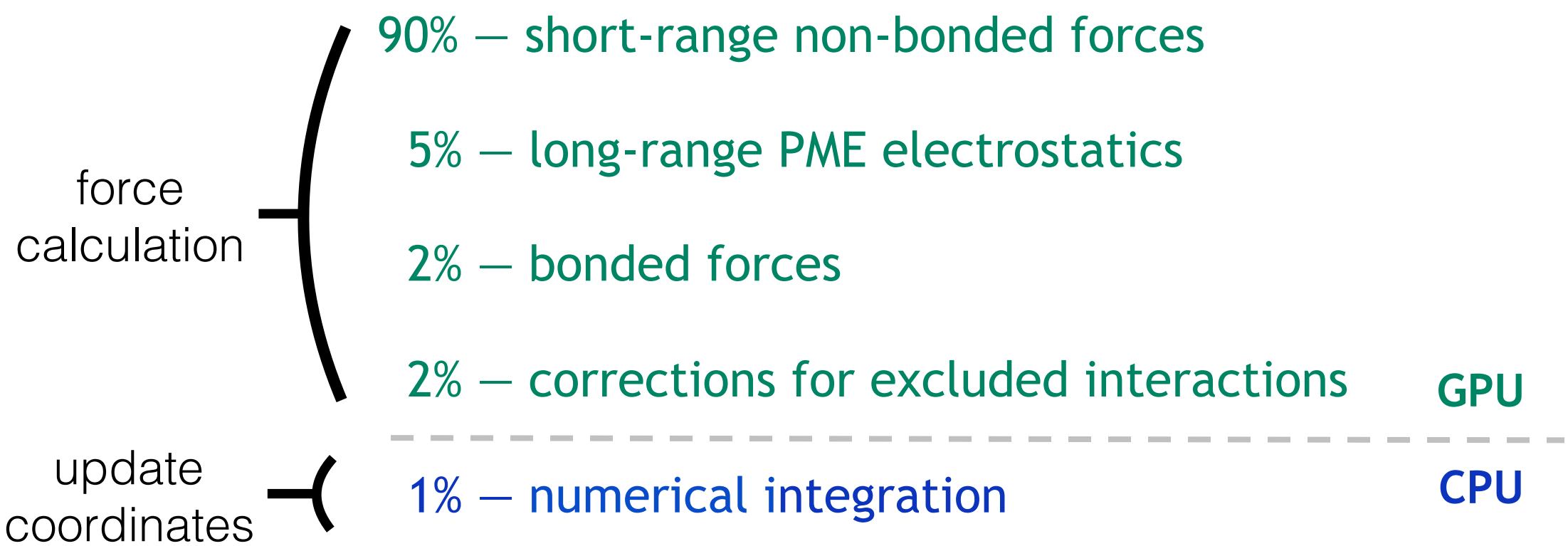
NAMD PERFORMANCE

GPUs	Architecture	V2.12 Nanoseconds/Day
1	Maxwell	0.65
1	Volta	5.34619
2	Volta	5.45701
4	Volta	5.35999
8	Volta	5.31339

Volta (2018) delivers ~10x performance boost relative to Maxwell (2014)

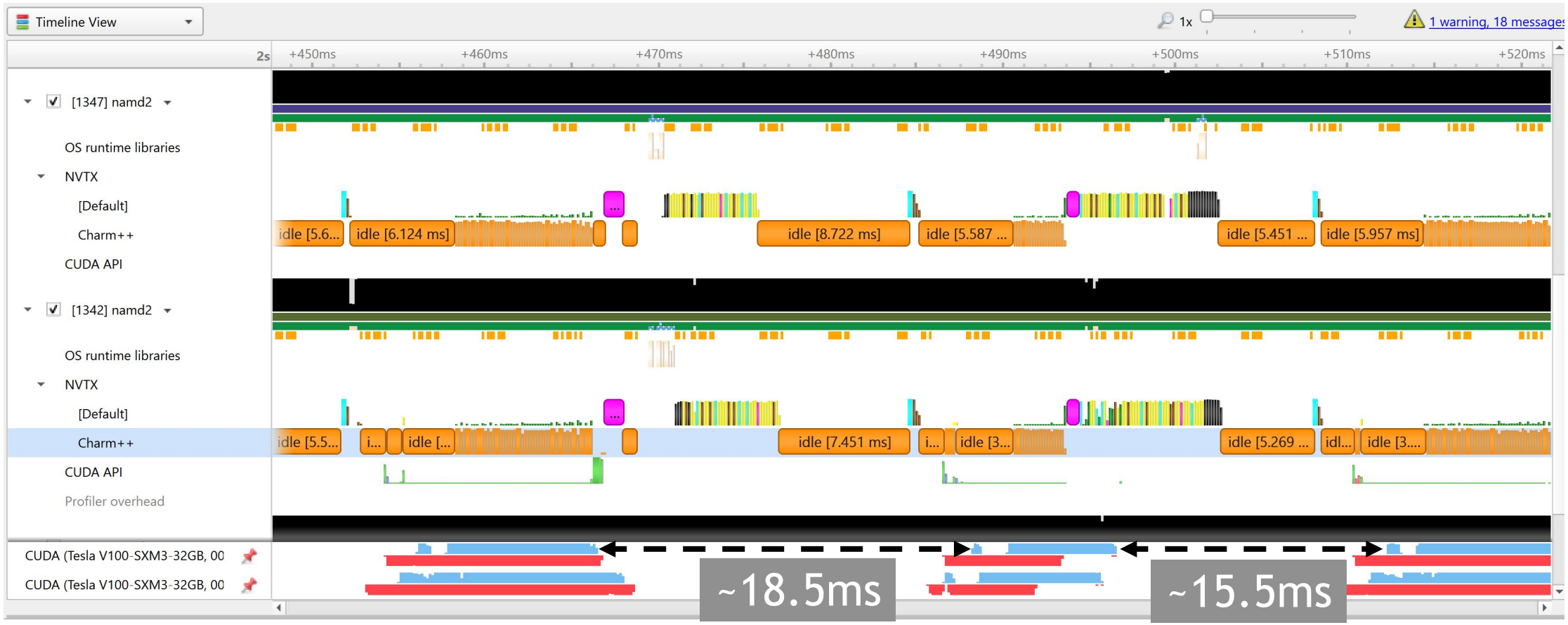
Failure to scale is caused by unbalanced resource utilization

V2.13 TIMESTEP COMPUTATIONAL FLOP COST



NAMD V2.13

Moving all force calculations to GPU shrinks timeline gap

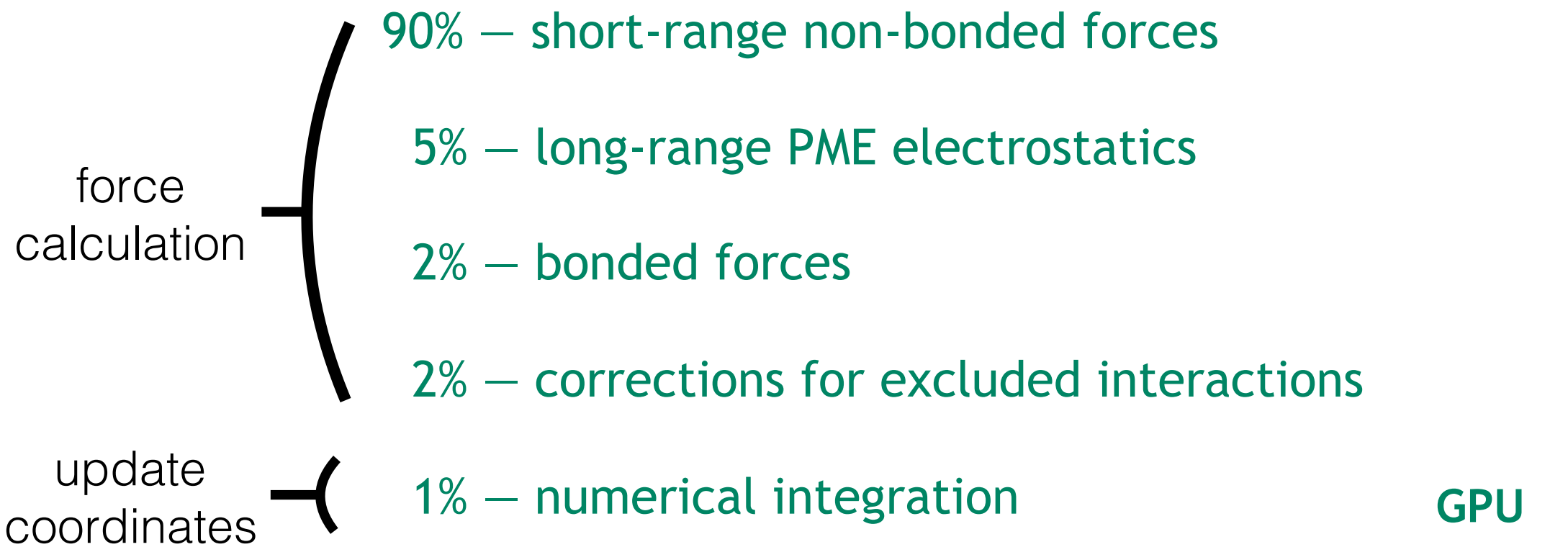


NVIDIA

NAMD PERFORMANCE

GPUs (Volta)	V2.12 Nanoseconds/Day	V2.13 Nanoseconds/Day (% Gain vs 2.12)
1	5.34619	5.4454 (1.2%)
2	5.45701	5.97838 (9.5%)
4	5.35999	7.49265 (39.8%)
8	5.31339	7.55954 (42.3%)

NEXT TIMESTEP COMPUTATIONAL FLOP COST

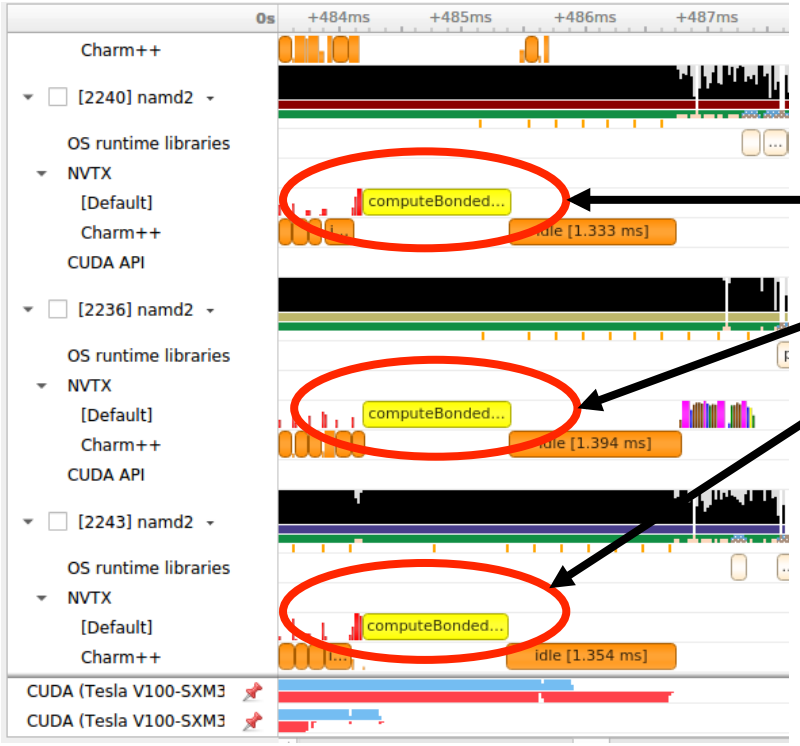


NAMD NEXT

Bonded Kernel Optimization

Symbol Name	Self, %	Module Name
Sequencer::submitReductions_SOA()	19.01	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
Sequencer::submitHalfstep_SOA()	16.59	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
Sequencer::addForceToMomentum_SOA(double, double, double, double, int)	6.40	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
CmiWallTimer	5.55	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
HomePatch::doMarginCheck_SOA()	4.18	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
0x7ffe51ffee2f	3.86	[vdso]
ComputeBondedCUDA::finishPatchesOnPe()	3.53	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
CmiGetNonLocal	3.52	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
ComputePmeCUDA::sendAtoms()	2.93	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2
CcdCallBacks	2.42	/Projects/dhardy/namd/Linux-x86_64-icc-smp-CUDA_devel/namd2

Host-side post-processing of bonded forces still a significant bottleneck

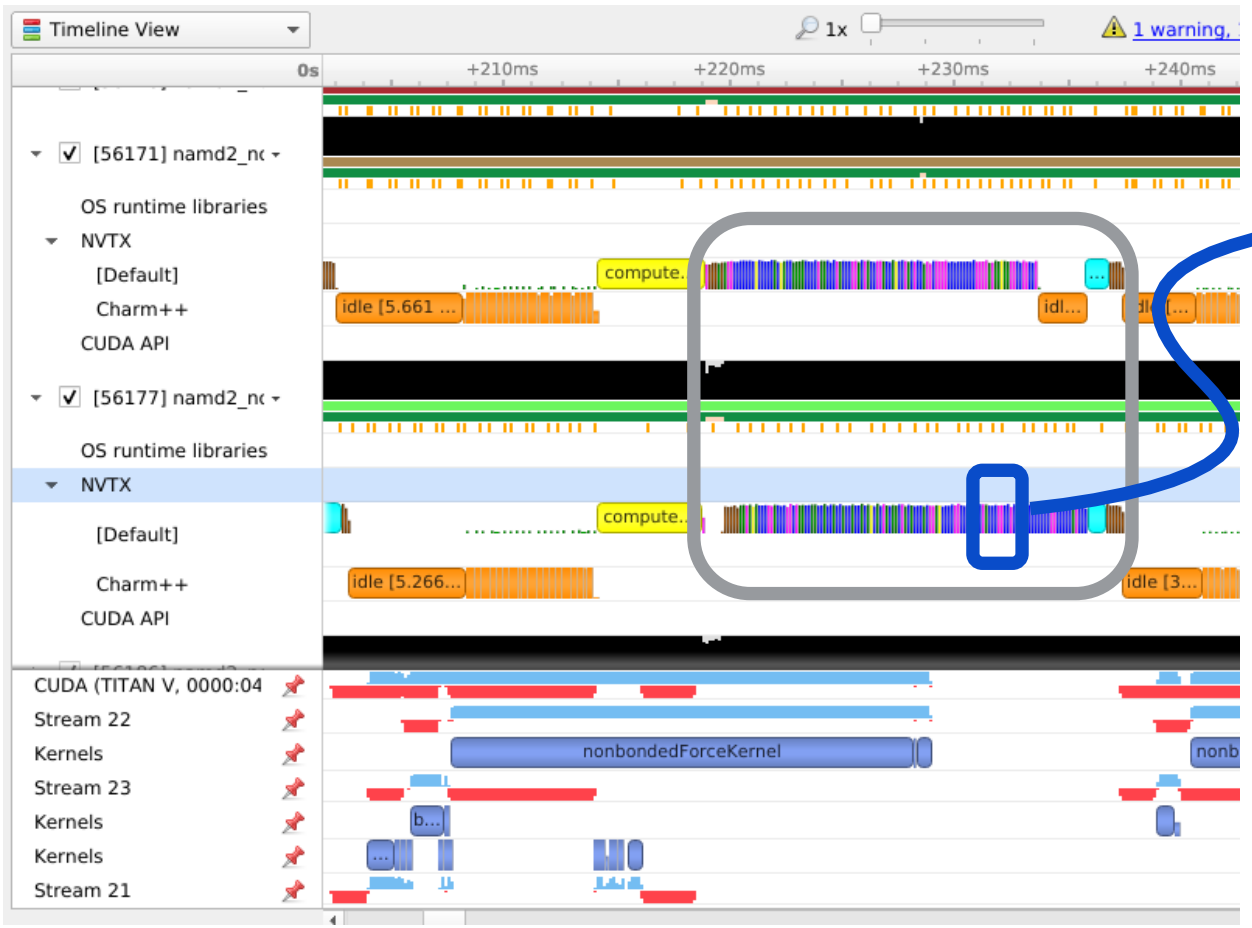


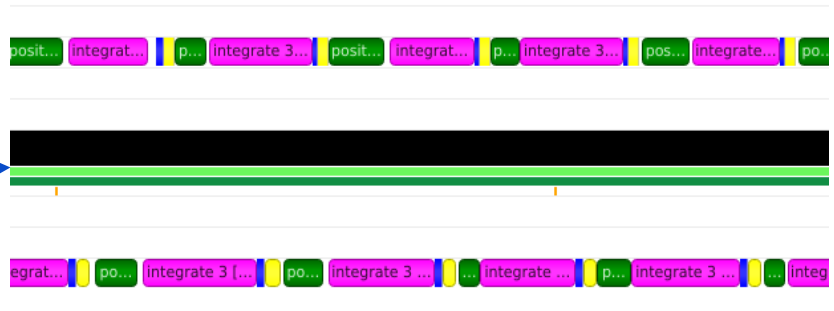
Cache locality optimized – type conversion on GPU, loop rearranged

ns/day gain 0%
Not on critical path.
Will be future benefit in multi-GPU environment.

NAMD NEXT

CPU integrator causing bottleneck



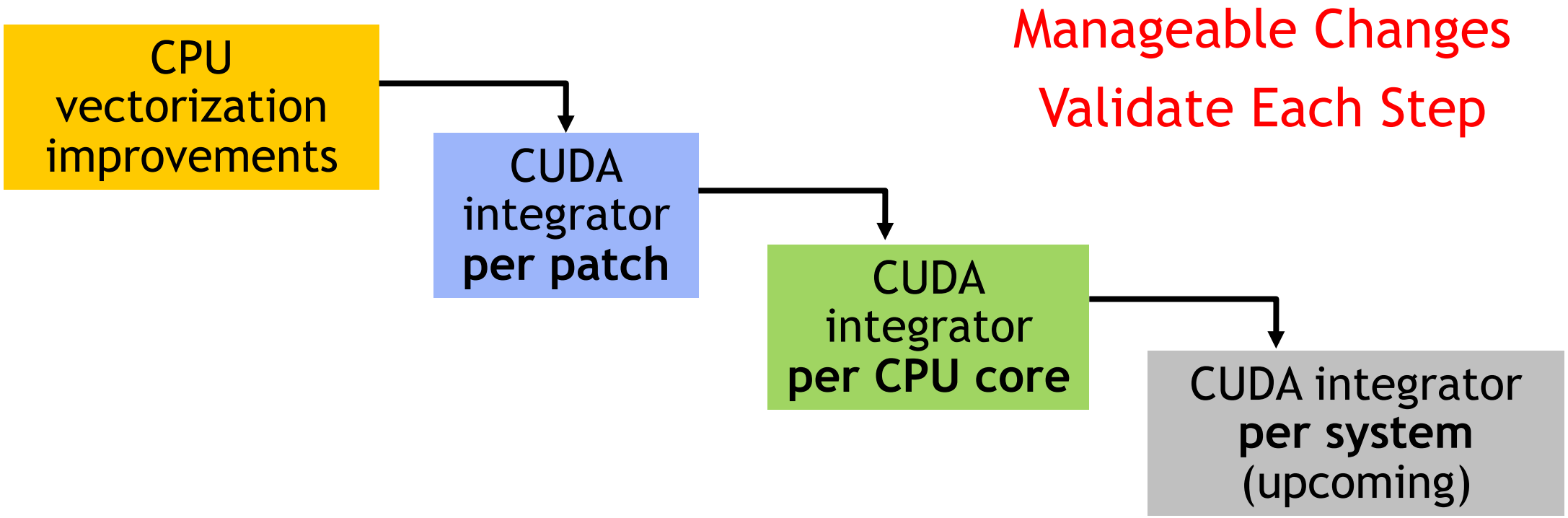


1% of computation is now
~50% of timestep work.
Amdahl's Law strikes again.

Data parallel calculation for GPU!

NAMD NEXT

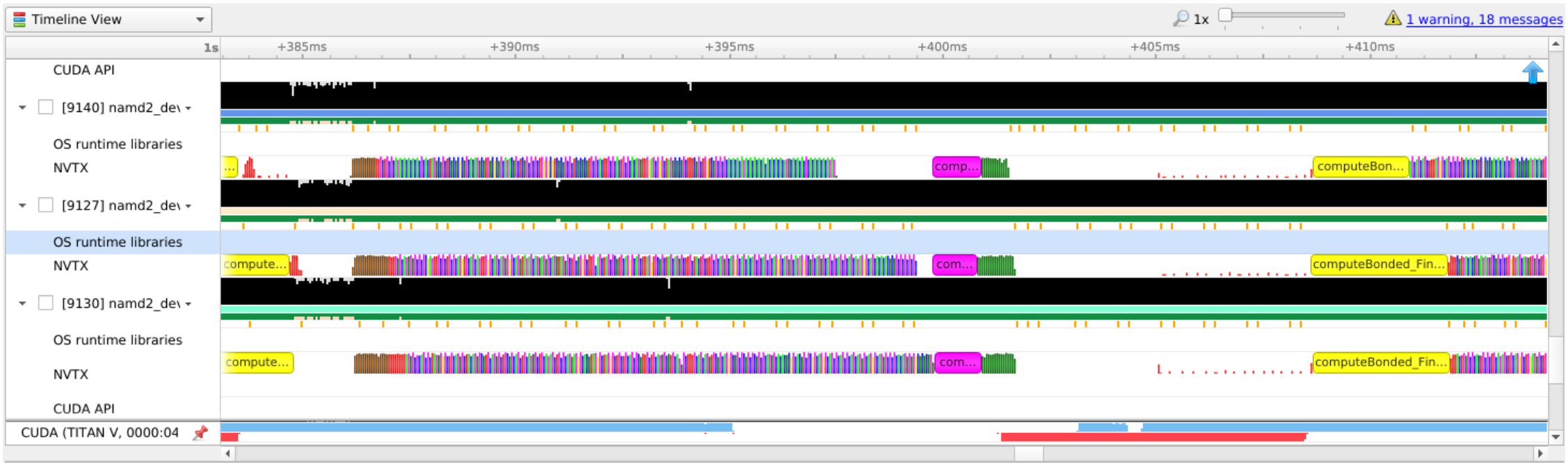
Integrator Development Phases



NAMD NEXT

Integrator - Phase 1

CPU vectorization – arrange data into SOA (structure of arrays) form



Speedup calculated via custom SQL-based script, NVTX ranges

Speedups: 26.5% for `integrate_SOA_2`, 52% for `integrate_SOA_3`

NAMD NEXT

Integrator – Phase 2

Per **Patch** Integrator Offload – Zoom In

Memory Transfer Hell
GPU Underutilized
Each Kernel Handles
~500 atoms
STMV includes 1M atoms

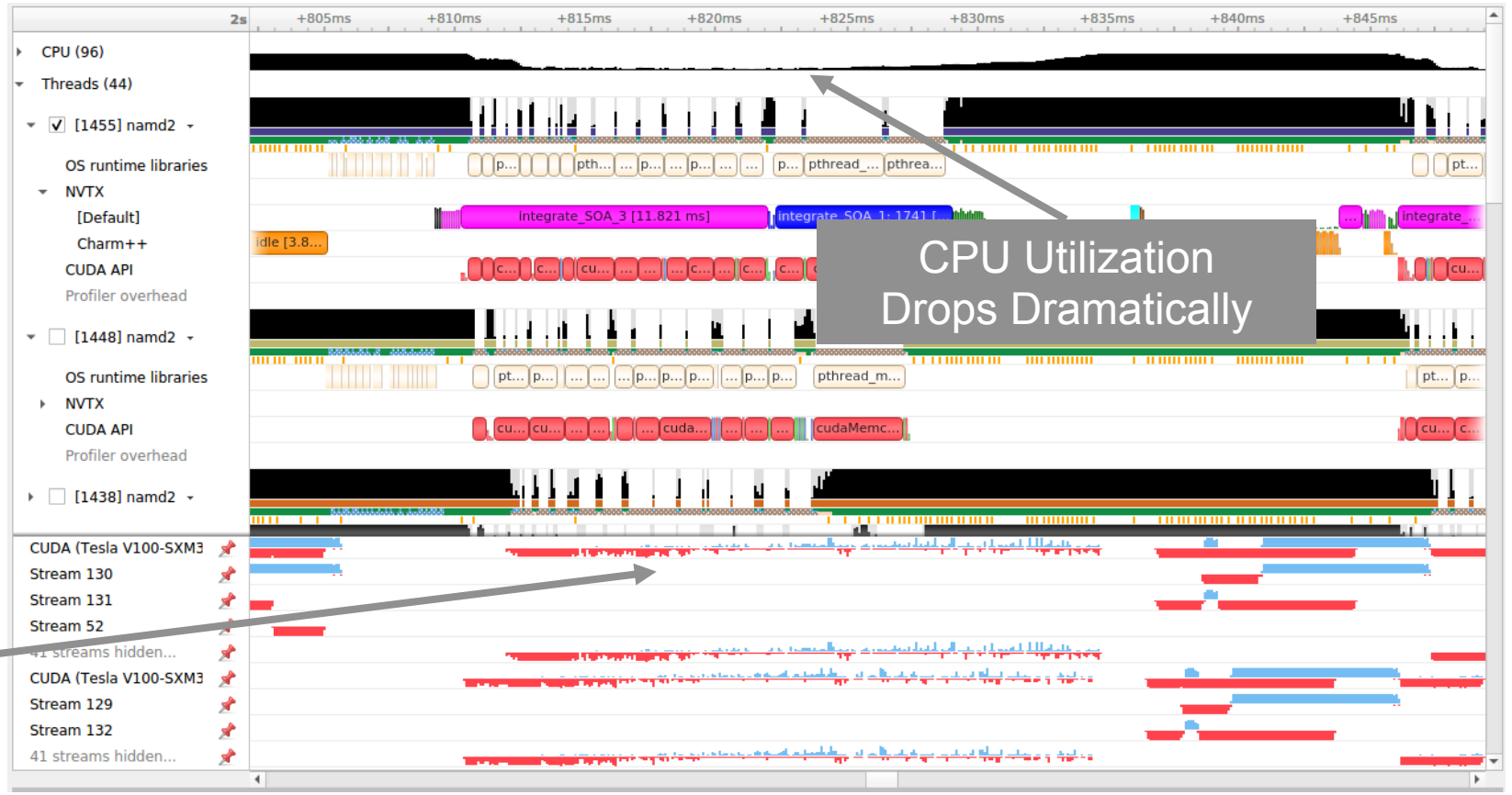
2200 *more* streams

NAMD NEXT

Integrator - Phase 3

Per **CPU Core** Integrator Offload - Zoom In

Improved GPU Utilization
Each Kernel Handles
~33K atoms
STMV includes 1M atoms



GPU timeline
filling with integrator work

NAMD NEXT

Integrator - Phase 3

Small Memory Copy Penalties

Small memory copy operations should be avoided by grouping data together

Improve memory access performance by using `Pinned` memory

NAMD PERFORMANCE

GPUs	V2.12 Nanoseconds/Day	V2.13 Nanoseconds/Day (% Gain vs 2.12)	NEXT Nanoseconds/Day SOA + Incomplete Integrator Phase 3 (% Gain vs 2.12)
1	5.34619	5.4454 (1.2%)	4.1451 (-22.5%)
2	5.45701	5.97838 (9.5%)	5.76149 (5.6%)
4	5.35999	7.49265 (39.8%)	7.11889 (32.8%)
8	5.31339	7.55954 (42.3%)	8.10406 (52.5%)

Integrator Phase 3 Optimization Development In Progress

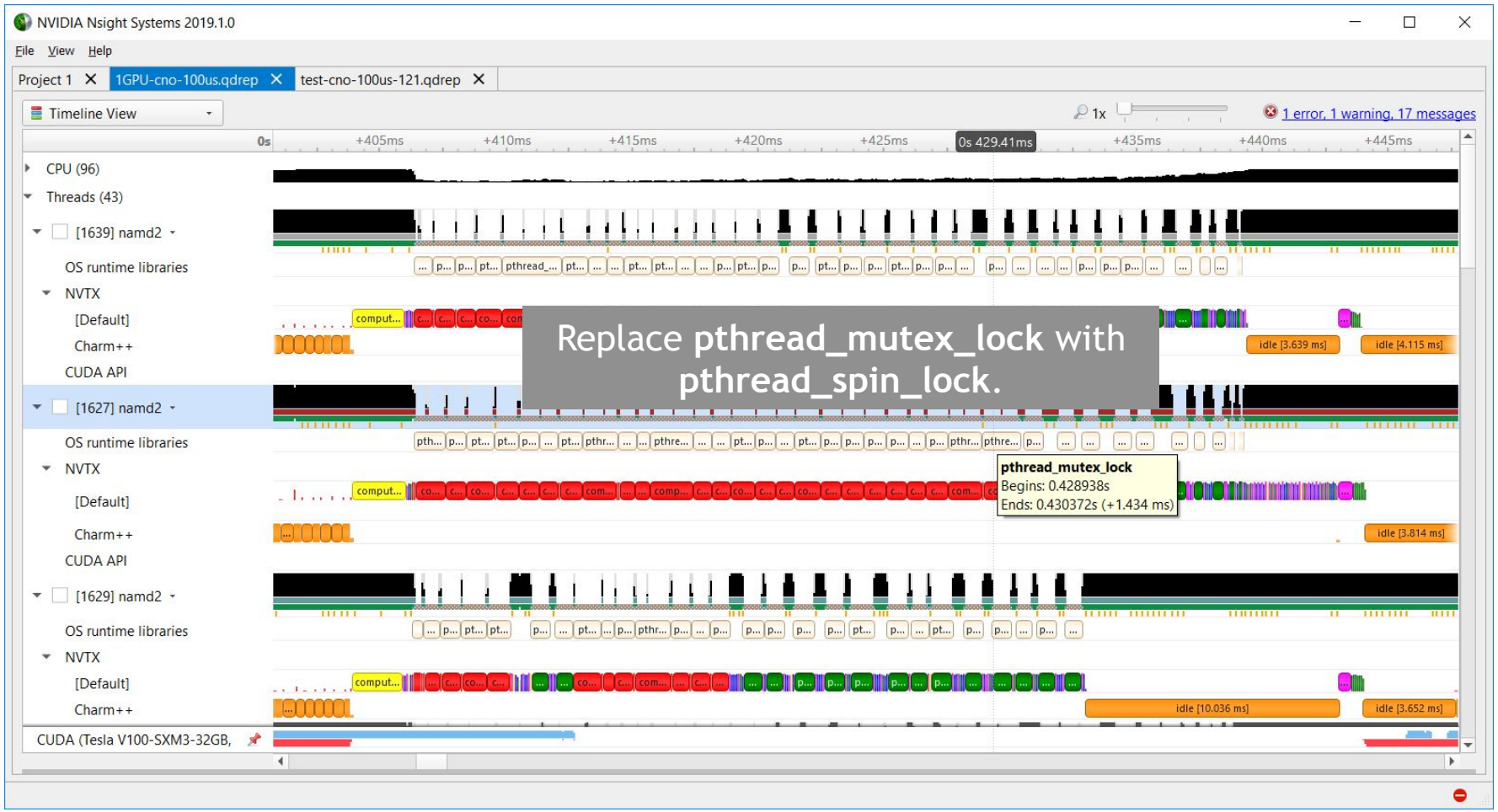
NAMD NEXT

What? Charm++ Runtime using 21.3% on `gettimeofday()`!

Replace with x86_64 `RDTSC` instruction, avoid unnecessary calculations.

NAMD NEXT

Charm++ Runtime Lock Optimization



NVIDIA

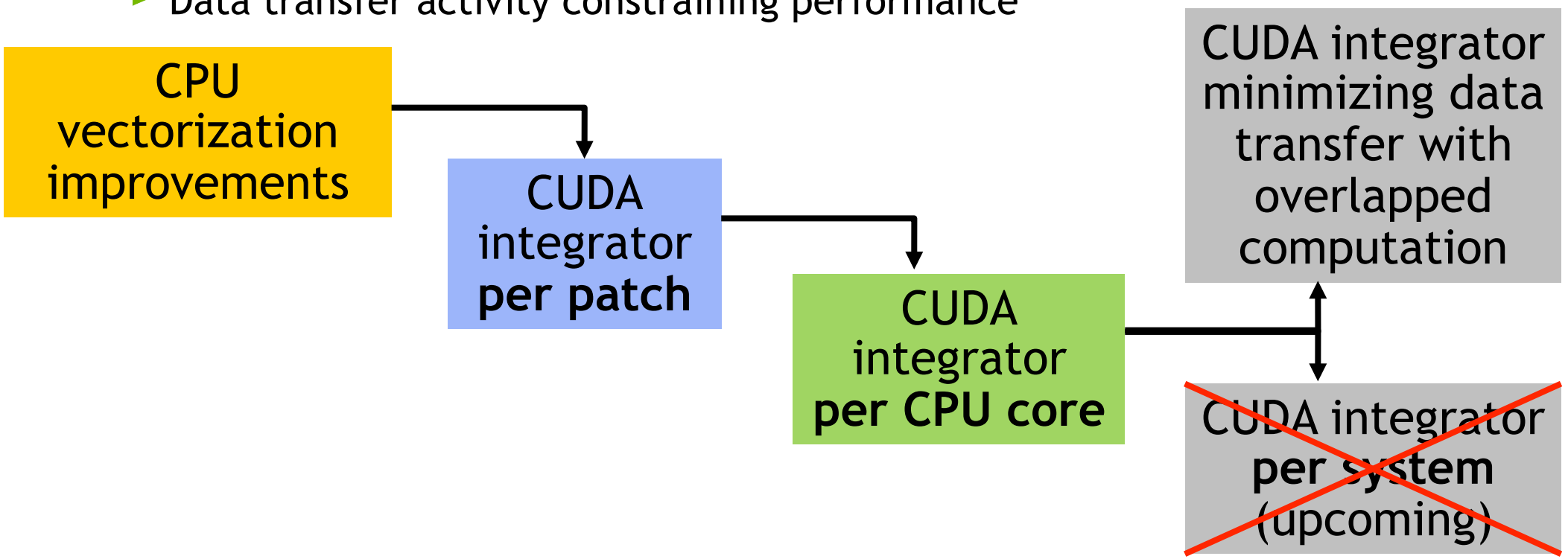
NAMD PERFORMANCE

GPUs	V2.12 Nanoseconds/Day	V2.13 Nanoseconds/Day (% Gain vs 2.12)	NEXT Nanoseconds/Day SOA+Runtime (% Gain vs 2.12)
1	5.34619	5.4454 (1.2%)	6.7032 (25.4%)
2	5.45701	5.97838 (9.5%)	7.24473 (32.8%)
4	5.35999	7.49265 (39.8%)	8.95371 (67.0%)
8	5.31339	7.55954 (42.3%)	9.26881 (74.4%)

Integrator Phase 3 Optimization Development In Progress

NAMD SUMMARY

- Nsight Systems guides development process
  - Estimated best case performance: 10 to 12 nanoseconds/day (single GPU)
  - Data transfer activity constraining performance

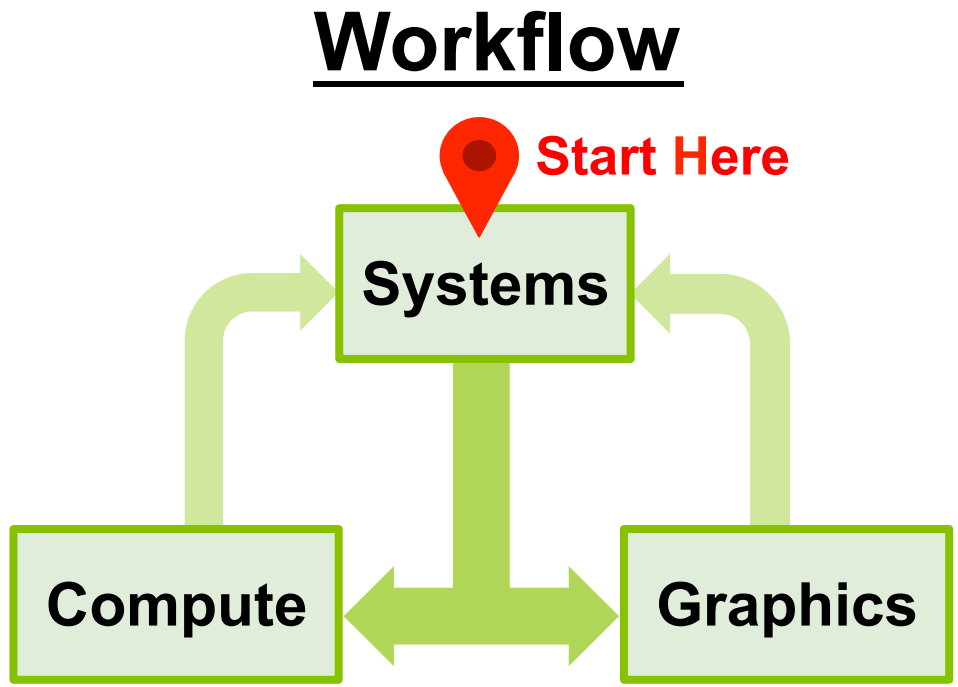


COMMON OPTIMIZATION OPPORTUNITIES

- CPU
  - Thread Synchronization
  - Algorithm bottlenecks starve the GPU(s)
- 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

NSIGHT PRODUCT FAMILY

- **Nsight Systems** - Analyze application algorithm system-wide
- **Nsight Compute** - Debug/optimize CUDA kernel
- **Nsight Graphics** - Debug/optimize graphics workloads



ACKNOWLEDGMENTS

- U. of Illinois
  - Julio Maia, Ronak Buch, John Stone, Jim Phillips
- NVIDIA
  - Daniel Horowitz, Antoine Froger, Sneha Kottapalli, Peng Wang

THANK YOU!

Visit us at the NVIDIA booth for a live demo!

Download latest public version
https://developer.nvidia.com/nsight-systems

Also available in CUDA Toolkit (v10.1 and later)

Forums: https://devtalk.nvidia.com

Email: nsight-systems@nvidia.com

DEVELOPER TOOLS AT GTC19

Talks

- S9345: CUDA Kernel Profiling using NVIDIA Nsight Compute
- S9661: Nsight Graphics - DXR/Vulkan Profiling/Vulkan Raytracing
- S9751: Accelerate Your CUDA Development with Latest Debugging and Code Analysis Developer Tools
- S9866: Optimizing Facebook AI Workloads for NVIDIA GPUs
- S9339: Profiling Deep Learning Networks

Demos of DevTools products on Linux, DRIVE AGX, & Jetson AGX at the showfloor

- Wednesday @12-7
- Thursday @11-1


NVIDIA

DEMO BACKUP

CORRELATE ACTIVITY

FINDING A CORRELATION