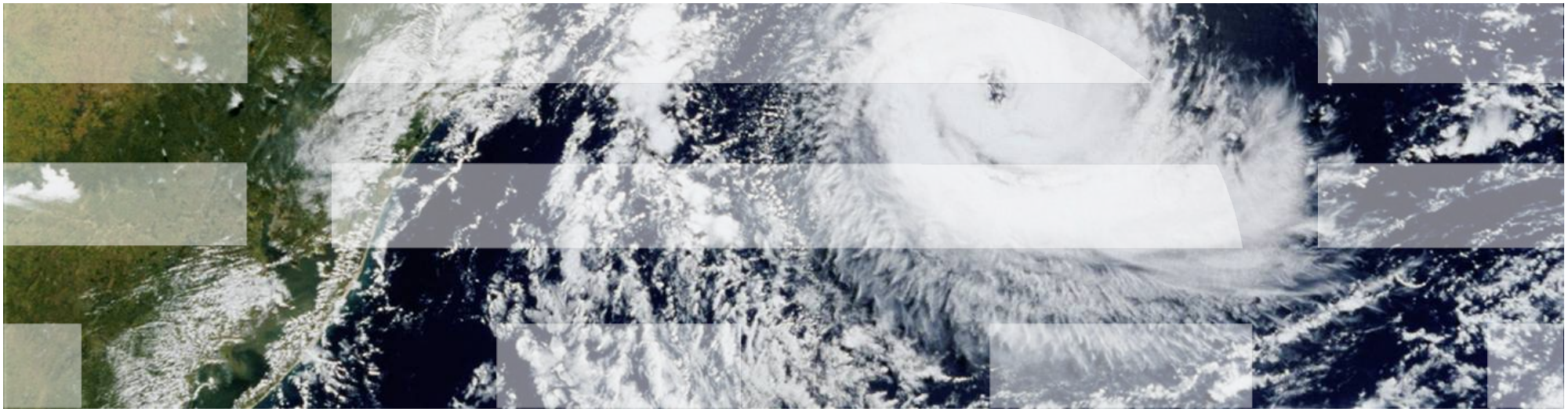
Programming Techniques for Supercomputers

(Heterogeneous and Distributed systems)

– Aditya Nitsure (IBM India)



Content

- Performance Engineering
- HPC Benchmarks
- Performance Monitoring Tools
- Gromacs case study

Performance Engineering

Performance Engineering (1)

- Not (only) performance testing !
- Enables -
  - End to end optimization
  - Cross functional team collaboration among all stakeholders
- Performance planning
  - Define performance goal/matrix starting from design phase
- Run performance benchmarks

Performance Engineering (2)

- Performance tuning – System, Environment, Application
- Performance monitoring and analysis – Tracing, sampling, profiling
- Publish guideline for performance tuning
- Publish performance paper, competitive performance etc.
- Support customer to solve performance issues

HPC Benchmarks

HPC Benchmarks

Micro-benchmarks

- Compute
  - CPU utilization, Frequency variation, CPI
  - Example : Sysbench
- Memory
  - Memory bandwidth, Utilization, Usage, Latency
  - Example : lmbench, unixbench
- IO
  - Throughput (bandwidth), IOPS
  - Example : fio
- Network
  - Throughput (bandwidth), latency
  - Example : iperf

HPC Benchmarks

- **HP LINPACK**
  - Used for Top500 supercomputer ranking
  - Solves linear equations (dense matrix operations)
- **Stream**
  - Measures CPU memory bandwidth
  - Copy, Scale, Add, Triad operations on large arrays
  - GPU stream available to measure GPU memory bandwidth
- **HPC Challenge**
  - Combination of existing benchmarks – DGEMM,STREAM,FFT
- **NPB (NAS Parallel benchmark)**
  - Derived from some of the NASA applications
  - Hybrid programs (MPI + OpenMP)
  - Integer sort, Multigrid, 3D FFT

HPC Benchmarks (CORAL)

Application benchmarks (mini-app)

- **Scalable**
  - HACC (Hardware accelerated cosmology code)
    - Simulate the formation of structure in collision less fluids under the influence of gravity in an expanding universe.
    - Compute intensity, random memory access, all-to-all communication
  - Nekbone
    - Incompressible Navier- Stokes CFD solver
    - Compute intensity, small messages, allreduce
- **Data centric**
  - Graph500
    - Scalable breadth-first search of a large undirected graph.
  - SpecInt
    - CPU integer processor benchmark

HPC Benchmarks (CORAL)

Application benchmarks (mini-app)

- **Throughput**
  - AMG (Algebraic Multi-Grid)
    - Linear system solver for unstructured mesh physics packages
    - memory-access bound, generates many small messages, stresses memory and network latency
  - NAMD
    - Molecular dynamics
    - Compute intensity, random memory access, small messages, all-to-all communications
- **Data Science and Deep learning**
  - BDAS – Big data analytic suit
  - DLS – Deep Learning suit

Reference:

- https://asc.llnl.gov/coral-benchmarks
- https://asc.llnl.gov/coral-2-benchmarks

Monitoring Tools

top

- Display Linux processes and threads managed by Linux kernel
- User can monitor CPU/memory utilization per process or thread, individual core utilization

```
top - 07:38:06 up 400 days, 21:49,  2 users,  load average: 0.41, 0.53, 0.94
Tasks: 1528 total,   3 running, 768 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.4 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem : 53194912+total, 49547968+free, 33711168 used,  2758272 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 49519660+avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
71810 root      20   0  980160   9088   3008 R  72.0  0.0   0:07.85 fio
 1747 root      20   0       0      0      0 S  11.8  0.0 117:32.58 kdmwork-253:4
  179 root      20   0       0      0      0 S   3.9  0.0   1:38.79 ksoftirqd/28
  155 root      20   0       0      0      0 S   3.0  0.0   1:43.74 ksoftirqd/24
10690 root       0 -20       0      0      0 I   2.3  0.0   0:56.88 kworker/28:1H
 9968 root       0 -20       0      0      0 I   2.0  0.0   0:57.89 kworker/24:1H
71121 aditya    20   0  121472  10240   4544 R   1.3  0.0   0:02.80 top
123642 root     20   0    8704   6464   3072 S   1.3  0.0 416:55.49 irqbalance
  161 root      20   0       0      0      0 S   1.0  0.0   0:33.61 ksoftirqd/25
  185 root      20   0       0      0      0 S   1.0  0.0   0:34.65 ksoftirqd/29
 7472 root       0 -20       0      0      0 I   0.7  0.0   0:14.98 kworker/25:1H
22672 root      20   0       0      0      0 R   0.7  0.0   0:09.01 kworker/24:2
24216 root      20   0       0      0      0 I   0.7  0.0   0:08.85 kworker/28:0
130653 root     20   0 3106176 141696  84096 S   0.7  0.0 762:46.72 openshift
     9 root     20   0       0      0      0 I   0.3  0.0 452:55.24 rcu_sched
  167 root      20   0       0      0      0 S   0.3  0.0   0:17.81 ksoftirqd/26
  191 root      20   0       0      0      0 S   0.3  0.0   0:17.33 ksoftirqd/30
 7363 root      20   0 1271936   9536   4416 S   0.3  0.0   1807:11 pmsensors
 9381 root       0 -20       0      0      0 I   0.3  0.0   0:08.09 kworker/30:1H
 9906 root       0 -20       0      0      0 I   0.3  0.0   0:15.07 kworker/29:1H
22547 root      20   0       0      0      0 I   0.3  0.0   0:00.56 kworker/30:2
25684 root      20   0       0      0      0 I   0.3  0.0   0:01.16 kworker/25:1
71807 root      20   0 1044928 423680 418112 S   0.3  0.1   0:00.14 fio
158156 root     20   0       0      0      0 I   0.3  0.0   0:00.86 kworker/29:2
     1 root     20   0  161856  13440   6720 S   0.0  0.0   1039:19 systemd
```

mpstat

- Reports processor related statistics
  - Interrupts received per second by CPU(s)
  - User and kernel CPU utilization
  - Percentage of time spent by CPU in IO request or idle
- Fetches system, core and node level details

mpstat -n -P 0,1,2,3 -N 0,8 1

08:56:10 AM CDT	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
08:56:11 AM CDT	0	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
08:56:11 AM CDT	1	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
08:56:11 AM CDT	2	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
08:56:11 AM CDT	3	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
08:56:10 AM CDT	NODE	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
08:56:11 AM CDT	0	5.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	94.99
08:56:11 AM CDT	8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

free / numastat

- ***free***
  - Displays the total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel.

```
[aditya@localhost ~]$ free -h
              total        used        free      shared  buff/cache   available
Mem:           1.0Ti        33Gi       708Gi       2.8Gi       280Gi       981Gi
Swap:          4.0Gi          0B       4.0Gi
```

- ***numastat***
  - Shows numa node memory statistics for processes and OS
  - numa_hit, numa_miss, numa_foreign, interleave_hit

numastat

```
[aditya@hpcnw4 ompeval]$ numastat -m | grep "Node\|MemUsed"
                          Node 0          Node 8        Node 252        Node 253        Node 254        Node 255           Total
MemUsed                104872.31        57992.06           11.19           11.19           11.19         1455.19       164353.12
```

CPU memory

GPU memory

vmstat

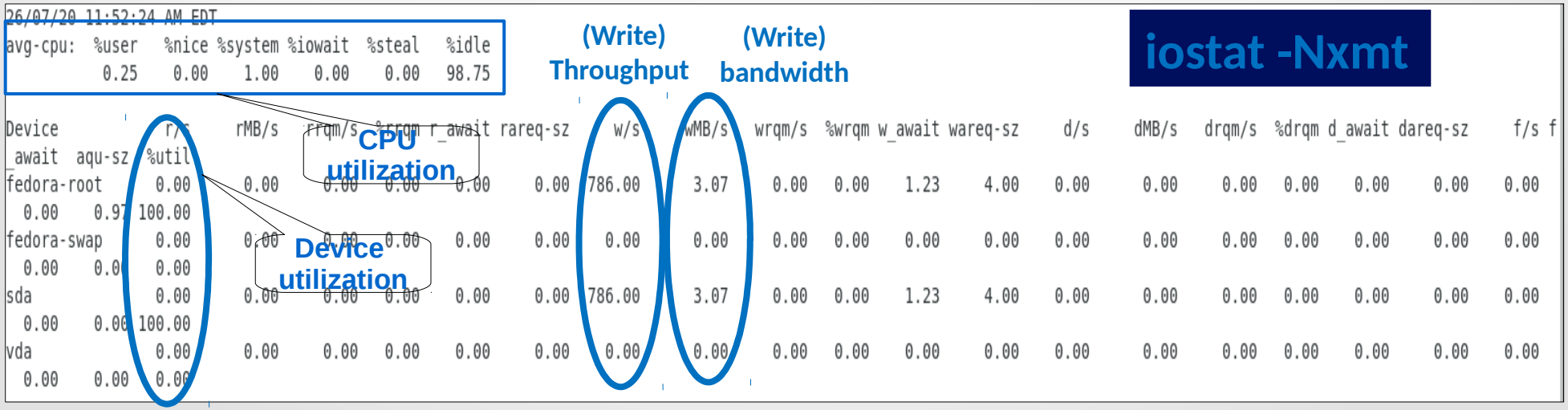
- Reports virtual memory statistics
  - Amount of virtual and free memory
  - Amount of memory used as buffer and cache
- Display CPU statistics
  - Time spent in kernel, user, IO wait, idle etc

vmstat -wtS M

```
[aditya@perfbos22 fio_output]$ vmstat -wtS M
procs -----------------------memory---------------------- ---swap-- -----io---- -system-- --------cpu-------- -----timestamp-----
 r  b         swpd         free         buff        cache   si   so    bi    bo   in   cs  us  sy  id  wa  st                 CDT
 2  0            0       483858            0         2719    0    0     0    20    0    0   0   0 100   0   0 2020-10-14 08:00:36
```

iostat

- Provides insight on input/output load on physical disks
- Gives CPU and IO statistics
  - CPU and device utilization
  - Throughput, Read/write bandwidth



perf

- Perf stat
  - Gathers performance counter statistics
  - Examples
    - perf stat -C 0 -e cycles sleep 1
    - perf stat -a -e cycles,instructions < executable >
- Perf record
  - Records profile data in *perf.data* file
  - Example : perf record -a -g -e cycles -o perf.raw < executable >
- Perf report
  - Reads *perf.data* and display the profile
  - Example : perf report -n --no-children --sort=symbol -i perf.raw

nvidia-smi

```
[aditya@hpcnw4 ompeval]$ nvidia-smi
Sun Apr 21 03:03:12 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.64                 Driver Version: 396.64                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   42C    P0   153W / 300W |   1539MiB / 15360MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000004:05:00.0 Off |                    0 |
| N/A   38C    P0    37W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   35C    P0    36W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000035:04:00.0 Off |                    0 |
| N/A   41C    P0    38W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     54063      C   ./matmul_gpuoffload_cl                      1528MiB |
+-----------------------------------------------------------------------------+
```

Also check “nvidia-smi –query-gpu” more monitoring options

nvprof

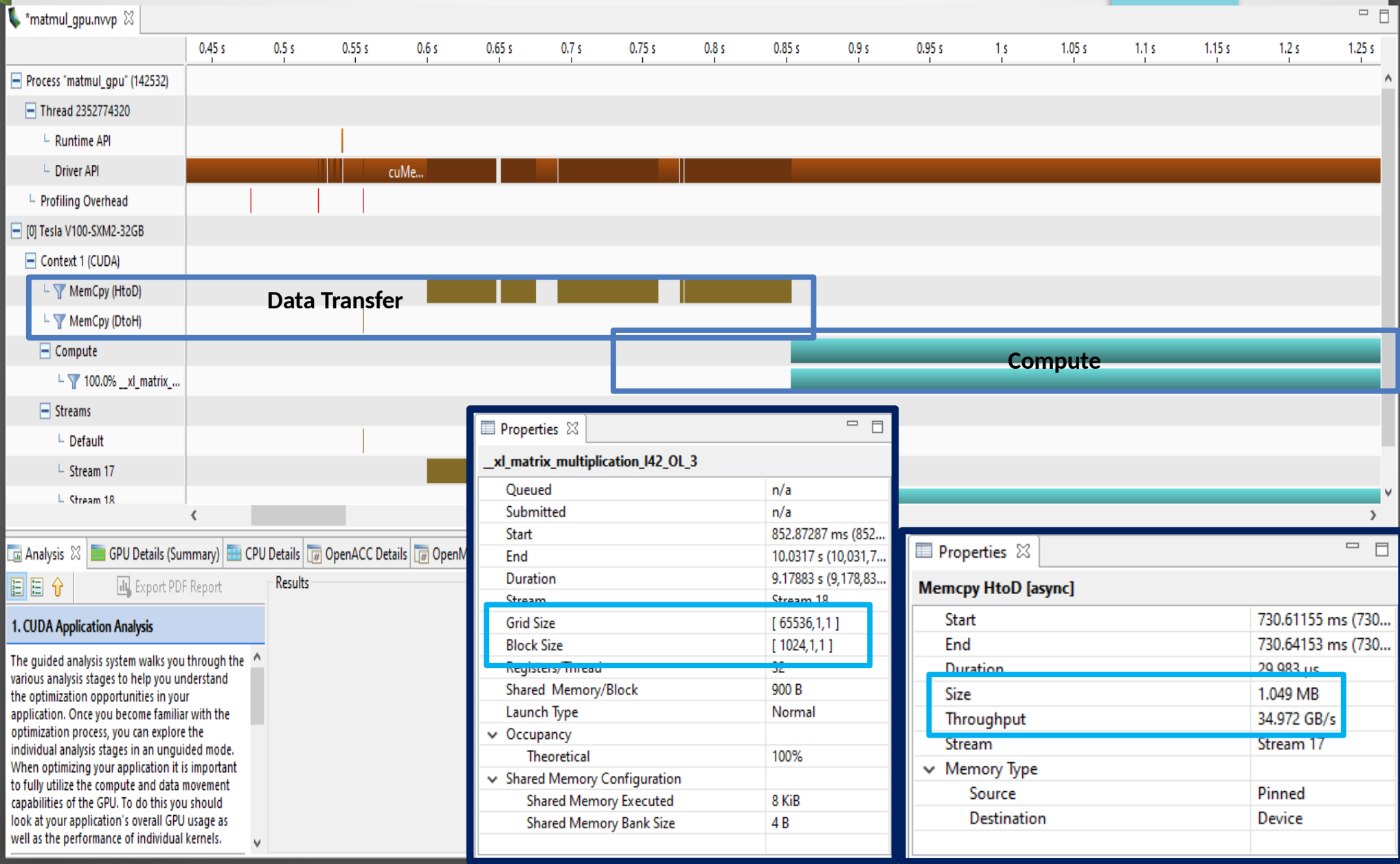
- The *nvprof* is command-line profiling tool which enables you to collect and view profiling data
- Using *nvprof* one can collect –
  - kernel execution time
  - Memory transfers
  - Memory set and CUDA API calls
  - Events or metrics for CUDA kernels
- *nvprof --print-gpu-trace -u col -o* <nvprof-output.nvvp> <Application binary> <Application parameters>

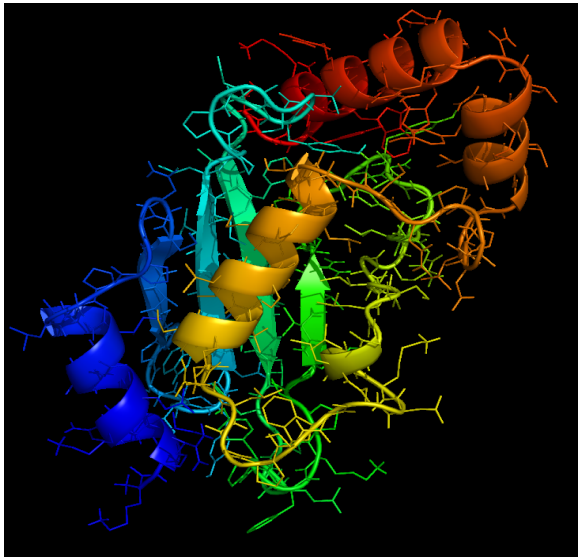
NVIDIA Visual Profiler (NVVP)

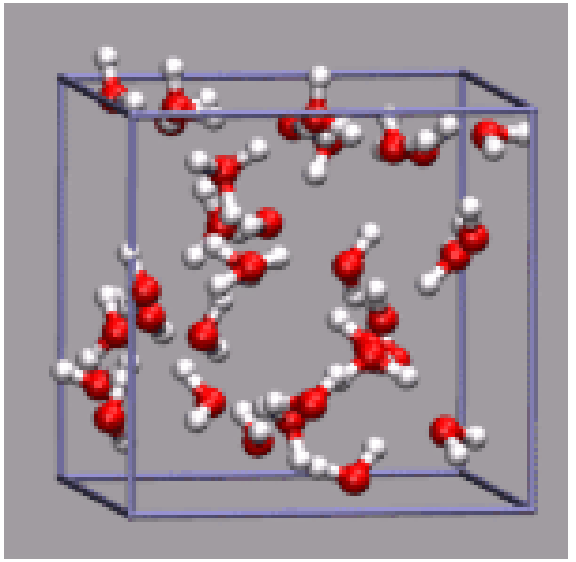
- Visualize profile data collected from *nvprof*
- The visual profiler displays a timeline of your application activity on both CPU and GPU so that one can identify opportunities for performance improvement

More documentation can be found @
https://docs.nvidia.com/cuda/profiler-users-guide/index.html

NVIDIA Visual Profiler (NVVP)







HPC Case Studies

HPC Workload and Performance

HPC performance usually driven by

- #cores / CPUs /Nodes
- Core frequency
- #GPUs
- CPU/GPU Memory bandwidth
- NVLink bandwidth
- Interconnect bandwidth
- Energy consumption

HPC workload characterization

- Problem size
- #of tasks per Node/GPU
- Amount of data handled by each Node and GPU
- Compute – communication ratio
- Amount of data transferred by each task
- Amount of work done by CPU & GPU separately
- Amount of collaboration done by CPU & GPU

GROMACS

GROningen MAchine for Chemical Simulations

- GROMACS is a versatile package to perform molecular dynamics simulation with the Newtonian equations of motion for systems with hundreds to millions of particles.
- GPU acceleration is core part of GROMACS that works in combination with GROMACS' domain decomposition and load balancing code.
- Gromacs team claims –
  - GPU accelerated code performances up to 5x better compared to CPU-only processing.

`	Gromacs 5.1.4 (8/09/16)	Gromacs 2016.3 (14/03/2017)
C++ standard	C++98	C++11
GPU compute capability	2.0 (Fermi or Kepler)	2.0 (Fermi, Kepler, Maxwell or Pascal)
CUDA version	4.0 onwards	5.0 onwards
FFTW		3.0 or later
cmake		2.8.8 or later
XL compiler		13.1.5 or later

Compiling Gromacs

CC=<mpicc|xlc|gcc> CXX=mpicxx cmake ..
-DGMX_OPENMP=<ON/OFF>
-DGMX_GPU=<ON/OFF>
-DGMX_MPI=<ON /OFF>
-DGMX_BUILD_OWN_FFTW=ON
-DCMAKE_BUILD_TYPE=Release
-DGMX_SIMD=IBM_VSX
-DGMX_CYCLE_SUBCOUNTERS=ON
-DGMX_OPENMP_MAX_THREADS=256
-DCMAKE_INSTALL_PREFIX="/home/aditya/GROMACS/install/GNU/MPI/CPU"

Simulation Examples

1	ADH cubic RF (Reaction Field) (alcohol dehydrogenase protein)	Atoms : 134,177
2	ADH cubic PME (Particle mesh Ewald) (alcohol dehydrogenase protein)	Atoms : 134,177
3	**Water_GMX50 RF (Reaction Field)**	**1.5 million atoms**
4	Water_GMX50 PME (Particle mesh Ewald)	1.5 million atoms

Running Gromacs

Running example : WATER GMX50

Configurable Parameters

#MPI tasks

#OpenMP threads

#GPUs

#Time steps

Generate *mdout.mdp* file

gmx grompp -f rf.mdp

CPU

gmx mdrun -noconfout -g Water50_rf_c20 -v -ntmpi 10 -ntomp 4 -pin on

GPU

gmx mdrun -noconfout -nsteps 5000 -resethway -g Water50_rf_c20 -v -ntmpi 8 -ntomp 10 -pin on -gpu_id "0123"

Running using MPI enabled Gromacs

mpirun -np 20 gmx_mpi mdrun -s lignocellulose-rf.tpr -v -noconfout -resethway -nsteps 5000 -g cellulose_mpi -pin on -ntomp 2

Gromacs functions distribution (CPU+GPU)

M E G A - F L O P S A C C O U N T I N G

Wall time: 15.71 sec
Performance : 27.49 ns/day

Computing:	M-Number	M-Flops	% Flops
Pair Search distance check	28679.216704	258112.950	0.2
NxN RF Elec. + LJ [F]	**3733618.418624**	**141877499.908**	**98.0**
NxN RF Elec. + LJ [V&F]	39223.903488	2118090.788	1.5
Reset In Box	193.536000	580.608	0.0
CG-CoM	193.536000	580.608	0.0
Virial	39.945360	719.016	0.0
Stop-CM	39.936000	399.360	0.0
Calc-Ekin	769.536000	20777.472	0.0
Constraint-V	3925.578069	31404.625	0.0
Constraint-Vir	40.810326	979.448	0.0
Settle	1308.526023	422653.905	0.3
Total		144731798.689	100.0

D O M A I N D E C O M P O S I T I O N S T A T I S T I C S

av. #atoms communicated per step for force: 2 x 455240.7
av. #atoms communicated per step for LINCS: 2 x 33675.7

Average load imbalance: 2.1 %
Part of the total run time spent waiting due to load imbalance: 0.3 %

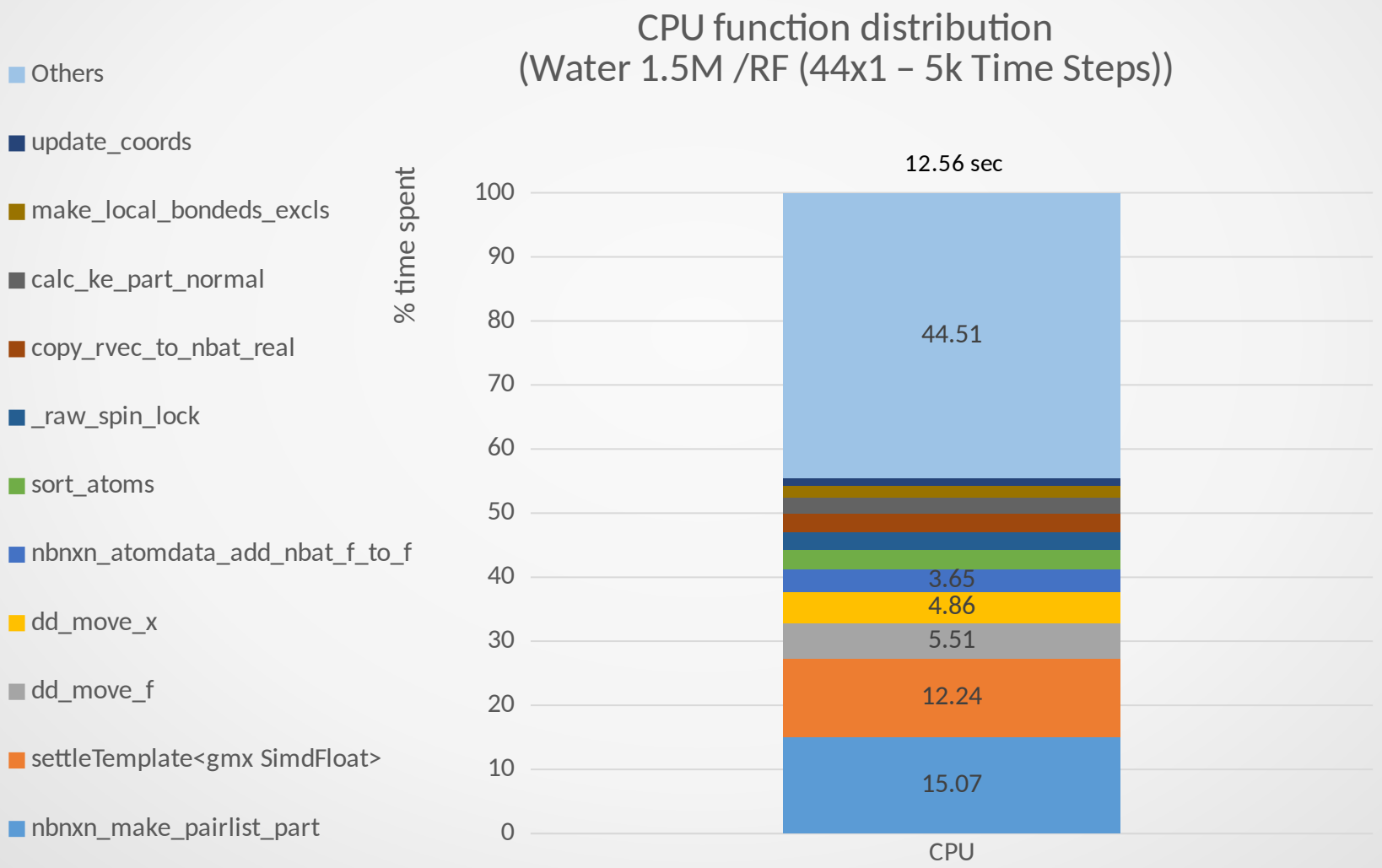
R E A L C Y C L E A N D T I M E A C C O U N T I N G

On 8 MPI ranks, each using 10 OpenMP threads

Computing:	Num Ranks	Num Threads	Call Count	Wall time (s)	Giga-Cycles total sum	%
Domain decomp.	8	10	126	3.193	130.772	20.3
DD comm. load	8	10	26	0.001	0.047	0.0
Neighbor search	8	10	126	3.208	131.390	20.4
Launch GPU ops.	8	10	5002	0.524	21.469	3.3
Comm. coord.	8	10	2375	1.705	69.856	10.9
Force	8	10	2501	0.180	7.361	1.1
Wait + Comm. F	8	10	2501	2.059	84.324	13.1
Wait GPU nonlocal	8	10	2501	0.354	14.485	2.2
Wait GPU local	8	10	2501	0.020	0.805	0.1
NB X/F buffer ops.	8	10	9752	0.919	37.636	5.8
Update	8	10	2501	1.032	42.263	6.6
Constraints	8	10	2501	2.231	91.383	14.2
Comm. energies	8	10	251	0.012	0.476	0.1
Rest				0.281	11.520	1.8
Total				**15.717**	**643.787**	**100.0**

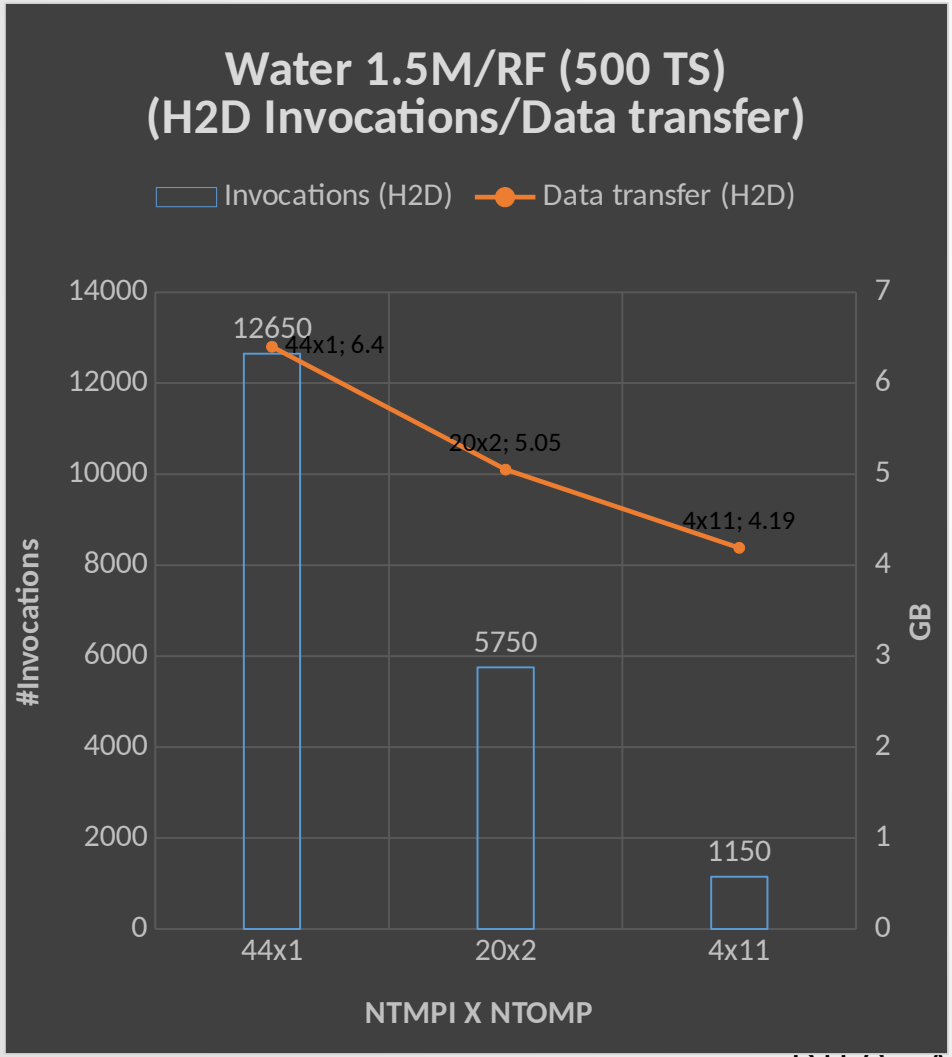
NOTE: 20 % of the run time was spent in domain decomposition,
20 % of the run time was spent in pair search

perf report (CPU)



(MPI tasks Vs Invocations & data)

Note: The data represented in below graph is from single GPU (0)



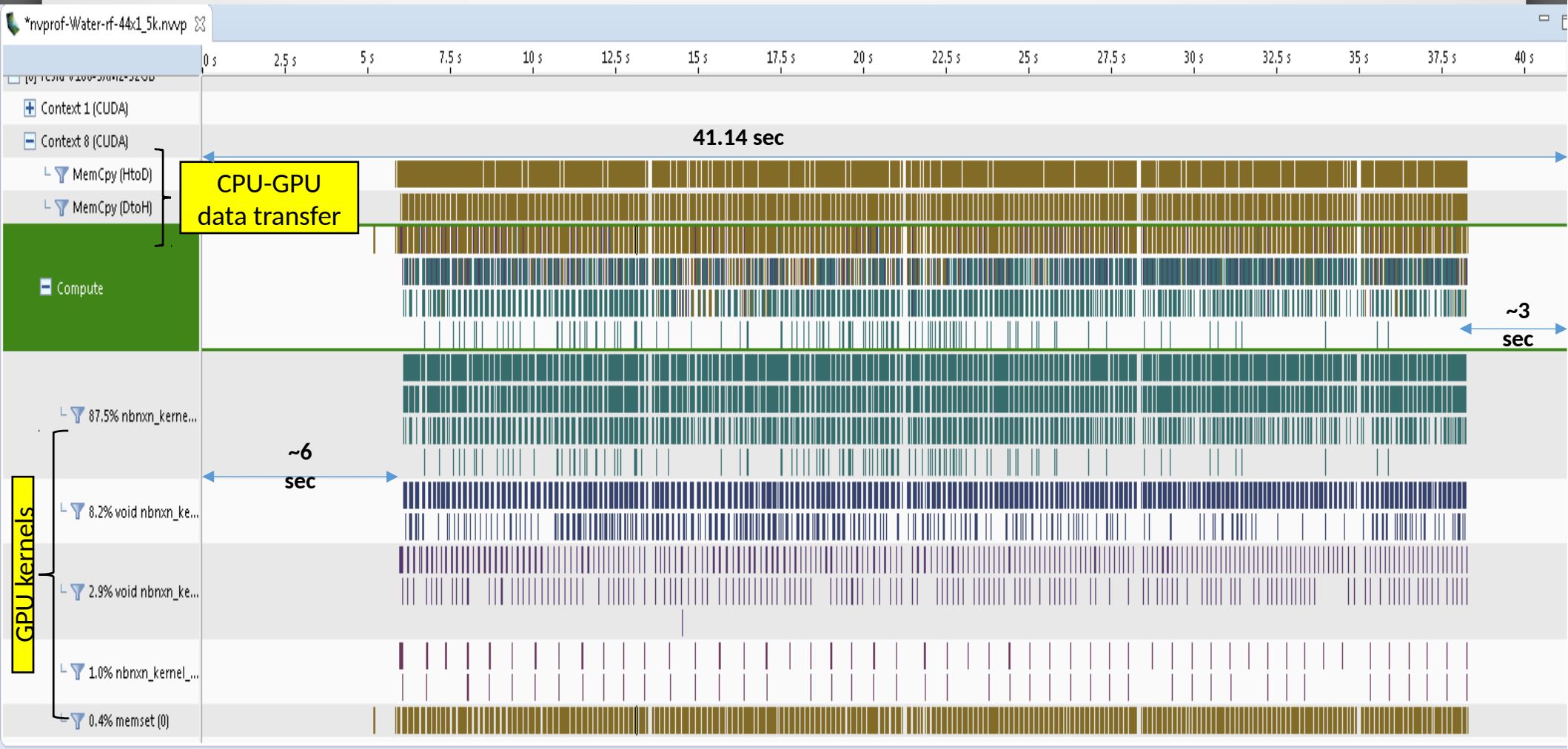
From observations

- Overhead due to MPI tasks
- *Assumption* : 5% overhead for 1 MPI task run
- Original H2D data transfer size – **4GB**

Tasks	% overhead	Calculated transfer size (GB)
1	5%	4.2
5	25%	5
11	55%	6.2

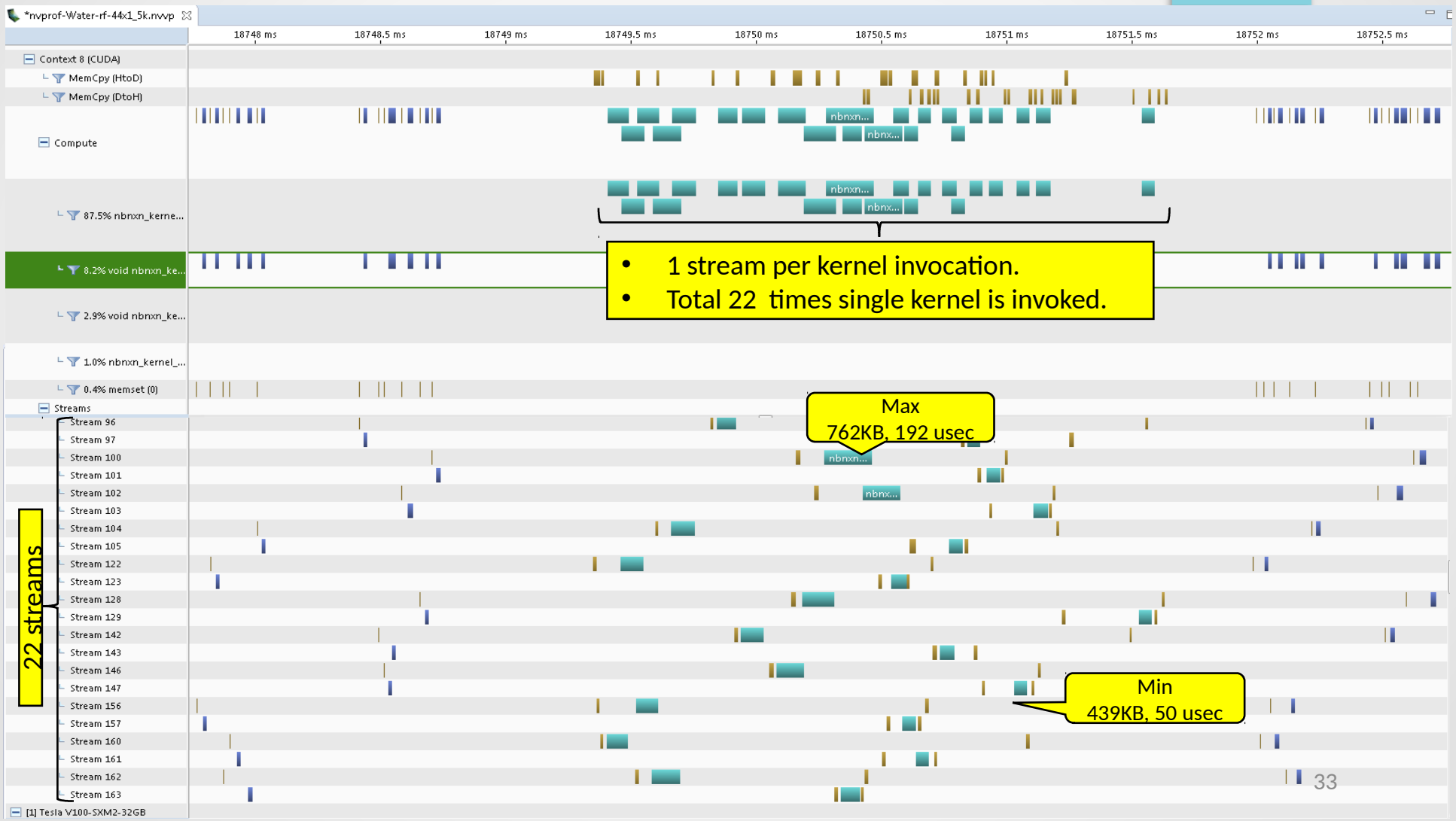
The number of invocations are proportional to the overhead or rate of increase in data size.

NVIDIA Visual Profiler (Water 1.5/RF)



NVIDIA Visual Profiler (Water 1.5/RF)

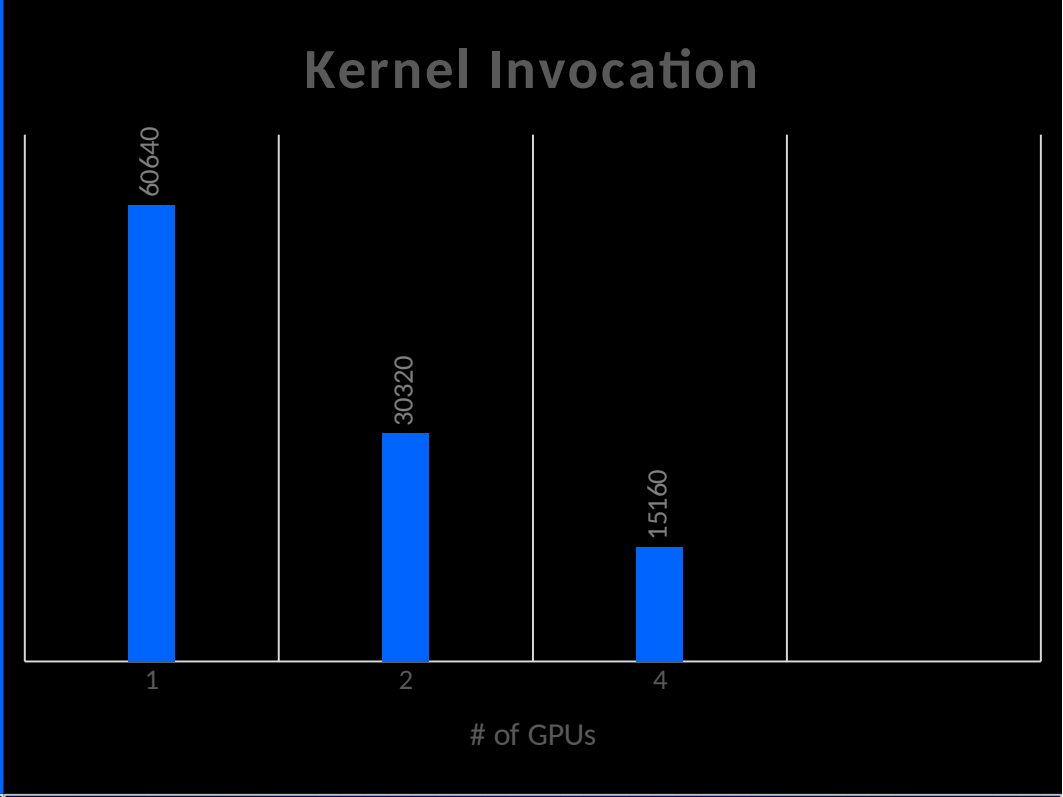
NVIDIA Visual Profiler (Water 1.5/RF)

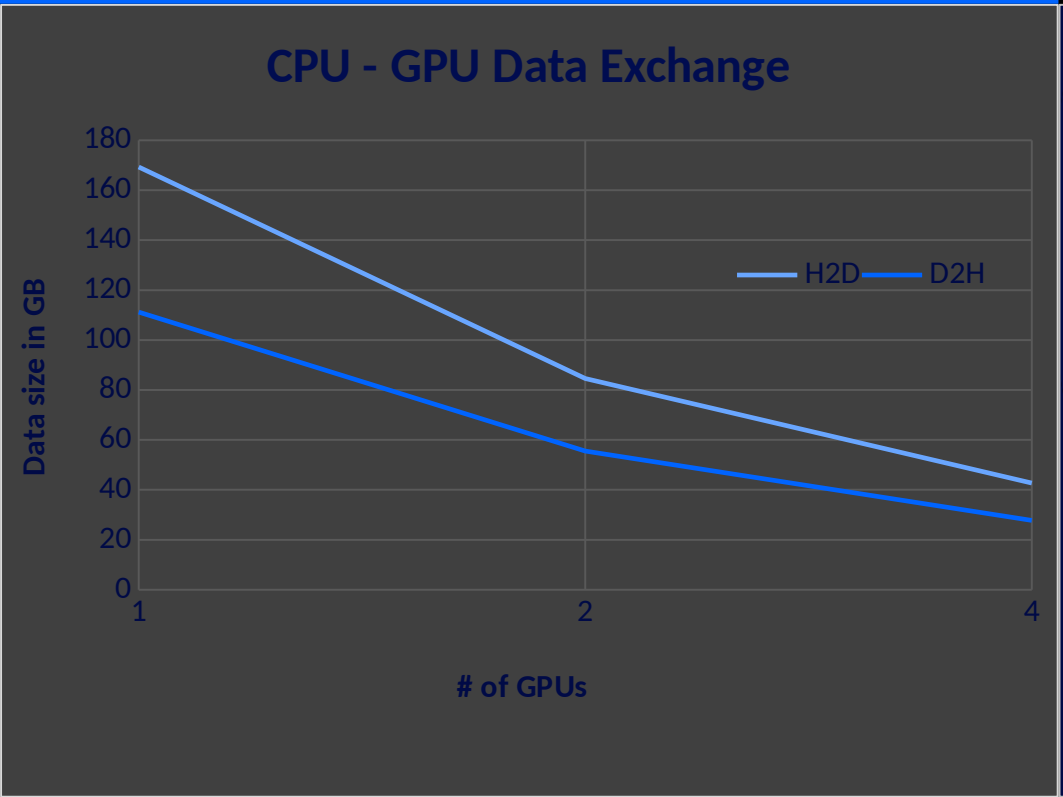


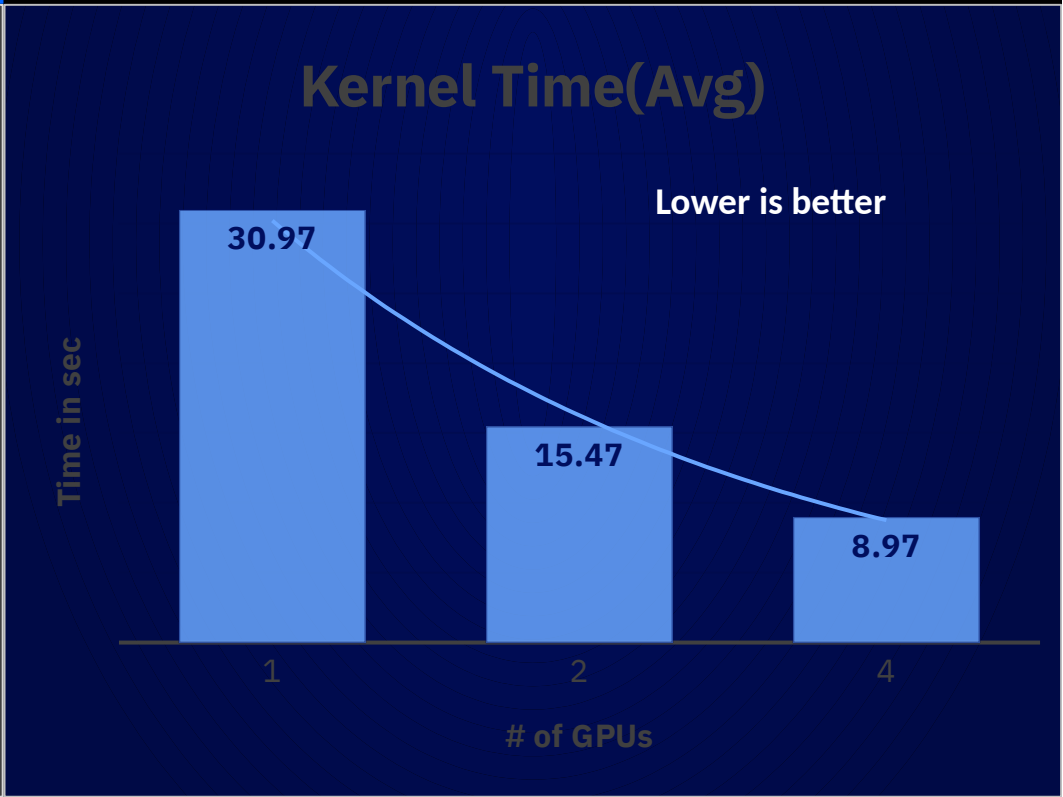
Gromacs – 1.5M Water Observations

Less amount of data exchange with increased number of GPUs

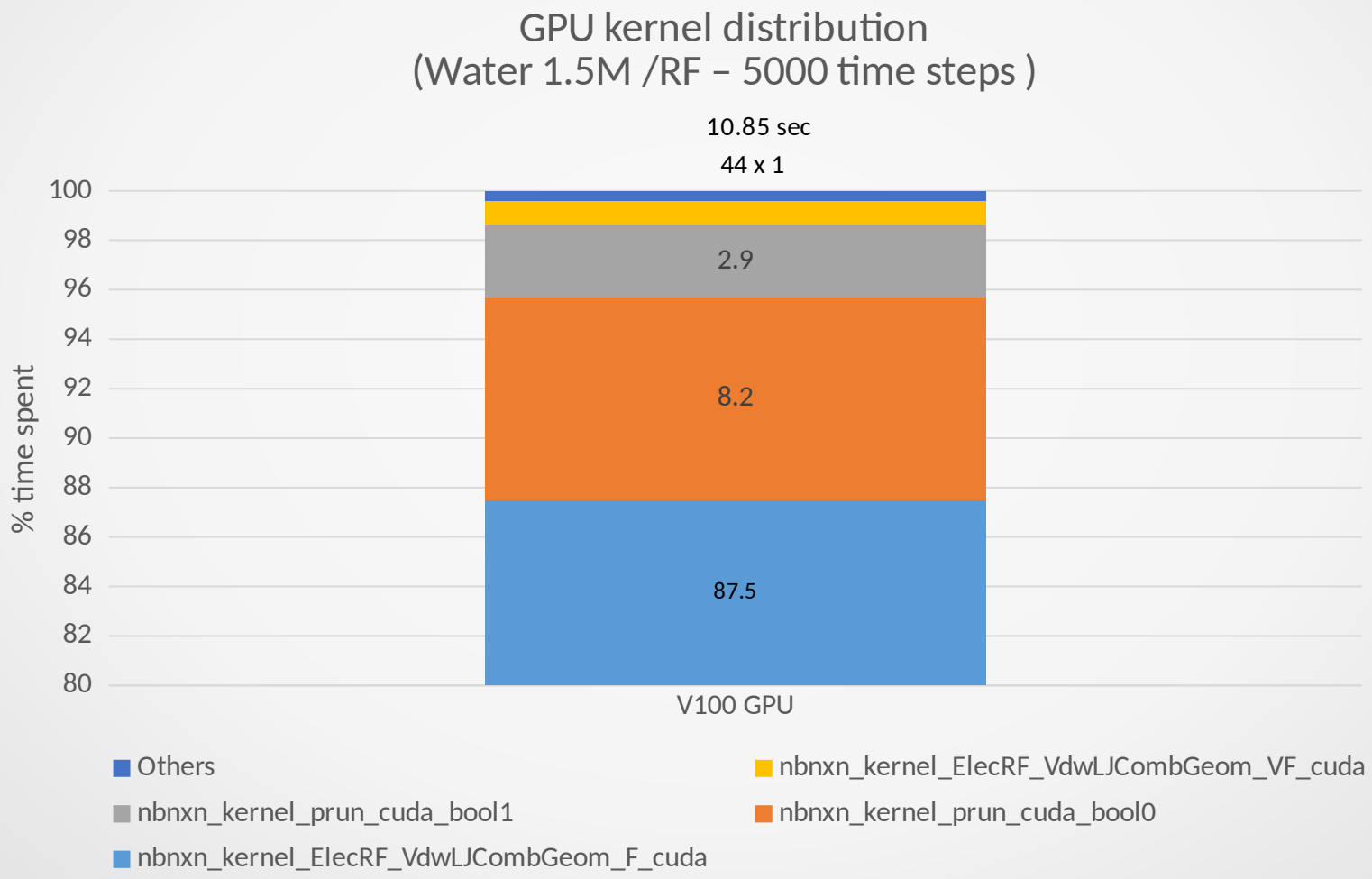
Large amount of data transfer – Potential NVLink benefit







GPU kernel distribution



Thank You !!!