# Performance Tools



NERSC New User Training Feb 16, 2024 Justin Cook PEM

#### Performance tools available on Perlmutter

- Profiling tools are used to measure and analyze code performance
- Metrics available help identify memory and compute bottlenecks
- List of profiling tools available on Perlmutter

CrayPat Perftools, -lite, Reveal	CPU, GPU, MPI	cc, CC, ftn, omp
NVIDIA Nsight Systems, Compute	GPU	nvc++, nvcc, omp, oacc, python
Linaro MAP, Performance Reports	CPU and GPU	Diverse
Darshan I/O profiler	Data I/O	Diverse
Timemory	Profiling kit	Diverse

A lot more! Full list at <a href="https://docs.nersc.gov/tools/performance/">https://docs.nersc.gov/tools/performance/</a>







# Nsight Systems









## **NVIDIA Nsight Systems**

- Nsight systems is a low overhead profiler
- Provides general broad description of GPU based applications
- Modules
  - o cuda
    - required to load Nsight systems
- Application must be compiled by linking cuda libraries
- Supports:
  - cuda (nvcc, nvc++)
  - Kokkos (cuda, OpenMP)
  - OpenMP
  - OpenACC





- Profiling steps for an OpenMP Offload based application compiled using clang++
- Code run in \$SCRATCH
- To visualize results, it is recommended to transfer profile files to local machine and use local install of Nsight Systems (available for free)
- Run steps are as follows:

```
$ module unload darshan
$ module load PrgEnv-<required>
$ ... compile your code ...
$ salloc --nodes 1 --qos interactive --time 01:00:00 --C gpu --account=mxxxx
$ srun -n 1 nsys profile --stats=true ./test_snap.exe -ns 100
```





#### Generates text of profiling results

CUDA API Statistics:

Time(%)	Total Time (ns)	Num Calls	Average	Minimum	Maximum	Name
83.9	6563435467	805	8153336.0	2413	44859498	cuStreamSynchronize
14.1	1100682085	33	33354002.6	2335	656900656	cuMemcpyHtoDAsync_v2
1.3	99718751	1	99718751.0	99718751	99718751	cuDevicePrimaryCtxRelease_v2
0.5	40948782	200	204743.9	26326	1158675	cuMemcpyDtoHAsync_v2
0.1	11082944	700	15832.8	6302	64274	cuLaunchKernel
0.1	6082361	17	357785.9	2668	2189898	cuMemAlloc_v2
0.0	858283	1	858283.0	858283	858283	cuModuleLoadDataEx
0.0	668654	32	20895.4	1345	353717	cuStreamCreate
0.0	654039	1	654039.0	654039	654039	cuModuleUnload
0.0	192187	32	6005.8	3933	21337	cuStreamDestroy_v2
0.0	159833	7	22833.3	14950	58247	cuMemcpyDtoH_v2
0.0	2250	1	2250.0	2250	2250	cuDevicePrimaryCtxSetFlags_v2







• Kernel names may be mangled (eq:

\_omp\_offloading\_70e68f56\_5d00401c\_\_ZN3SNA10compute\_yiEPd\_1471)

CUDA Ker	mel Statistics:					
Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Name
61.4	3946535611	100	39465356.1	39233639	44223639	ZN3SNA10compute yiEPd 1471
17.5	1125861668	100	11258616.7	11178964	11771631	ZN3SNA14compute duidrjEv 1652
16.3	1048090801	100	10480908.0	10082240	11828974	
3.8	243394690	100	2433946.9	2403145	2521384	ZN3SNA14compute deidrjEv 1577
0.8	53957727	100	539577.3	535003	635610	ZN3SNA14zero_uarraytotEv_1687
0.1	8023377	100	80233.8	79007	93024	ZN3SNA10compute_yiEPd_1460
0.0	2334567	100	23345.7	22720	29376	
ZN3SNA	17addself_uarrayt	otEd 1704				





CUDA	Memory	Operation	Statistics	(by	time):	
------	--------	-----------	------------	-----	--------	--

Time(%)	Total Time (ns)	Operations	Average	Minimum	Maximum	Operation
99.1	1098309937	33	33282119.3	1408	656553887	[CUDA memcpy HtoD]
0.9	10013795	207	48375.8	1696	99007	[CUDA memcpy DtoH]

CUDA Memory Operation Statistics (by size in KiB):

Total	Operations	Average	Minimum	Maximum	Operation
2660577.207	33	80623.552	0.008	1589250.000	[CUDA memcpy HtoD]
121875.788	207	588.772	0.001	1218.750	[CUDA memcpy DtoH]





<u></u>	NVIDIA Nsight Systems 2022.2.1	
<u>File View Tools H</u> elp		
Project Explorer ×	report2.qdrep 🗙	
Project 1	E Timeline View	📾 Q 1x 🖓
i profile_rank1	• 0s       1s       2s       3s       4s       5s         • CPU (80)       • CUDA HW (0000:88:00.0 - Test)       • Mem., (Memcpy Ht., 1)       • Mem., (Memcpy Ht.	6s 75 8s *







6		NVID	IA Nsight Systems	2022.2.1				$\lor$ $\diamond$ $\times$
<u>File View Tools H</u> elp								
Project Explorer ×	report2.qdrep ×							
Project 1 report2.qdrep	■ Timeline View ▼					1	📾 Q 1x 🖂	
profile_rank1 report2.gdrep	CPU (80) CUDA HW (0000:88:00.0 - Tesl. [All Streams] >99.9% Stream 13 < 0.1% Default stream 7 Threads (7) ✓ [7391] test_snap.exe  OS runtime libraries CUDA API Profiler overhead ✓ [7408] cuda-EvtHandlr  ØS runtime libraries Sthroadc biddon Events View	0.65 0.85 Memcpy HtoD (Pageable) Memcpy HtoD (Pageable) CuMemcpyHtoDAsync		15	1.25 Memcpy HtoD (Pageable Memcpy HtoD (Pageable cuMemcpyHtoDAsync	, 1,45	1.65	
	-	Right-click a timeline row and sele	ect "Show in Events	View" to see even	its here			







6		N	/IDIA Nsight Systems 2022	2.1			~
<u>File View Tools H</u> elp							
Project Explorer ×	report2.qdrep 🗙						
Project 1 report2.qdrep	Timeline View					📾 Q 1x 🖂	
profile_rank1	1s •	+650ms	+700	ms	+750ms	+800ms	· · · · · ·
report2.qdrep	<ul> <li>CPU (80)</li> </ul>						
	<ul> <li>CUDA HW (0000:88:00.0 - Tesl</li> </ul>						
	[All Streams]	omp_offloading_70e68f56	_5ompom	omp_offloading_70e68f56_5d.	om	omp_offloading_70e68f56	om )o
	▶ >99.9% Stream 13	omomp_offloading_70e68f56	_5]_omp]om]_	omp_offloading_70e68f56_5d	om	omp_offloading_70e68f56	]_om]
	<ol> <li>&lt;0.1% Default stream 7</li> </ol>						
	<ul> <li>Threads (7)</li> </ul>						
	▼ 🗸 [7391] test_snap.exe 🔹						
	OS runtime libraries	sem sem_wait	sem	sem_wait	sem	sem_wait	
	CUDA API	cuStre cuStreamSynchronize	cuStre	cuStreamSynchronize	cuStr	cuStreamSynchronize	cuStr
	Profiler overhead						
	▼ 🗹 [7408] cuda-EvtHandlr 🔹			000000000000000000000000000000000000000			
	OS runtime libraries	poll poll	poll poll	poll	poll poll	poll poll	poll poll
	5.throads.biddon 🗕 📥				7K7K31.11K		
	E contra Minus						
	Events view				ſ		
						Name 🔻	<u> </u>
						Description:	
		Right-click a timeline row and s	elect "Show in Events View	" to see events here			
							`
							Q)







<b>(</b>		NVIDIA Nsight Sy	stems 2022.2.1	1			$\sim$ $\diamond$ $\times$			
<u>File View Tools H</u> elp										
Project Explorer ×	report2.qdrep 🗙									
Project 1 report2.qdrep	■ Timeline View ▼									
report2.qdrep	0s • 1s +100ms +200ms +300ms +400ms +500ms +600ms +700ms +800ms CPU (80) CUDA HW (0000:88:00.0 - Tesl									
	Events View	•				Name	•			
	# *	Name	Start	Duration	GPU	Context	Description:			
	48	Memcpy DtoH (Pageable)	1.68302s	3.392 µs	GPU 0	Stream 13	omp_offloading_70e68f56_5d004			
	49	Memcpy DtoH (Pageable)	1.68304s	99.007 µs	GPU 0	Stream 13	01c_ZN3SNA10compute_uiEv_l425			
	50	omp_offloading_70e68f56_5d00401cZN3SNA14zero_uarr	1.68589s	626.170 µs	GPU 0	Stream 13	Ends: 1.69859s (+11.829 ms)			
	51	_omp_offloading_70e68f56_5d00401c_ZN3SNA17addself_u	1.68666s	26.527 µs	GPU 0	Stream 13	grid: <<<407, 1, 1>>>			
	52	omp_offloading_70e68f56_5d00401cZN35NA10compute	1.68677s	11.829 ms	GPU 0	Stream 13	block: <<<128, 1, 1>>> Launch Type: Regular			
	53	omp_offloading_70e68f56_5d00401cZN3SNA10compute	1.69883s	91.935 µs	GPU 0	Stream 13	Static Shared Memory: 2,336 bytes			
	54	omp_offloading_70e68f56_5d00401cZN3SNA10compute	1.69901s	44.188 ms	GPU 0	Stream 13	Dynamic Shared Memory: 0 bytes Registers Per Thread: 110			
	55	omp_offloading_70e68f56_5d00401cZN3SNA14compute	1.74355s	11.704 ms	GPU 0	Stream 13	Local Memory Per Thread: 0 bytes			
	56	omp_offloading_70e68f56_5d00401cZN3SNA14compute	1.75554s	2.510 ms	GPU 0	Stream 13	Local Memory Total: 112,721,920 byt			
	57	Memcpy DtoH (Pageable)	1.75833s	2.367 µs	GPU 0	Stream 13	Shared Memory executed: 16,384 by			
	58	Memcpy DtoH (Pageable)	1.75835s	98.463 µs	GPU 0	Stream 13	tes			
	59	omp_offloading_70e68f56_5d00401cZN3SNA14zero_uarr	1.7604s	629.498 µs	GPU 0	Stream 13	Shared Memory Bank Size: 4 B Theoretical occupancy: 25 %			
	60	omp_offloading_70e68f56_5d00401cZN3SNA17addself_u	1.76117s	26.847 µs	GPU 0	Stream 13	Launched from thread: 7391			
	61	omp_offloading_70e68f56_5d00401cZN3SNA10compute	1.76128s	11.335 ms	GPU 0	Stream 13	Latency: ← 8.603 µs			
	62	omp_offloading_70e68f56_5d00401c_ZN3SNA10compute	1.78036s	84.703 µs	GPU 0	Stream 13	Stream: Stream 13			







Office of Science

QI

# Nsight Compute









#### **NVIDIA Nsight Compute steps**

Case 1: baseline

- Exploit the ability to collapse nested for loops
- Case 2: collapse

```
#pragma omp target teams distribute parallel for
    for(int natom = 0; natom < num_atoms; ++natom)</pre>
         for(int nbor = 0; nbor < num_nbor; ++nbor)</pre>
             for(int j = 0; j < idxu_max; ++j)</pre>
                compute();
#pragma omp target teams distribute parallel for collapse(2)
    for(int natom = 0; natom < num_atoms; ++natom)</pre>
        for(int nbor = 0; nbor < num_nbor; ++nbor)</pre>
             for(int j = 0; j < idxu_max; ++j)
                compute();
             }
```





## **NVIDIA Nsight Compute**

- Nsight compute allows a deeper dive for profiling GPU based applications
- Modules
  - o cuda
    - required to load Nsight compute
- To visualize results, it is recommended to transfer profile files to local machine and use local install of Nsight Compute (available for free) or use NX (<u>https://docs.nersc.gov/connect/nx/</u>)

```
$ module unload darshan
$ module load PrgEnv-<required>
$ ... compile your code ...
$ salloc --nodes 1 --qos interactive --time 01:00:00 --C gpu --account=mxxxx
$ dcgmi profile --pause
$ srun -n 1 ncu -o profile_snap --set full ./test_snap.exe -ns 100
```





no: Détails		casez.neuriep A	.ncu-rep 🗙 🔬 case4.n	cu-rep * 🗙							
ye. Decano		✓ <u>R</u> esult: 4 - 642 - nvke	nel_ZN3SNA10comp 👻	😽 👻 Add Baseline	- Apply <u>R</u> ules	🖬 Occupancy	Calculator		Co	oy as In	nage
	Report	Result		Time	Cycles	Regs GPU	SM Frequenc	y CC Proces	s 🕀	Θ	B
Current	case2	642 - nvkernel_ZN3SNA100	ompute_yiEPd_F1L467_8	(395 16.20 msecon	d 17,731,183	96 0 - A100-S	KM4-40GB 1.09 cycle/ns	econd 8.0 [77358	] test_snap.exe		
Baseline 1	case1	642 - nvkernel_ZN3SNA100	ompute_yiEPd_F1L467_8	(16, 1 626.22 mseco	nd 688,052,413	96 0 - A100-S	KM4-40GB 1.10 cycle/ns	econd 8.0 [15917	] test_snap.exe		
GPU Speed	d Of Liaht	Throughput						All		-	0
mpute (SM	1) Through	1001 [%]		40,43 (+6,102,7	70%) Duration In	second			16.20	(-97.41	%)
mory Thro	ughput [%	]		96.11 (+662.8	(9%) Elapsed Cy	cles (cycle)			17,731,183	(-97.42	%)
TEX Cache	e Through	- iput [%]		96.22 (+10.2	3%) SM Active	cycles [cycle]			17,706,775.81	(-82.17	%)
Cache Thre	oughput [	%]		9.32 (+142.9	6%) SM Freque	ncy [cycle/nsecond	1		1.09	(-0.39	%)
AM Throug	ghput [%]			0.30 (+439.2	2%) DRAM Freq	uency [cycle/nseco	ond]		1.21	(-0.37	%)
⚠ FP64/3	32 Utiliza	tion performance. If Comp Kernel Profiling Guide	ne Workload Analysis de for mode details on roofli	termines that this kernel ne analysis. GPI	is fp64 bound, co	nsider using 32-bit	precision floating point of	perations to improve	its performance. Se	e the	
					onnoughput						
	т				1	1					1
ompute (Si	SM) [%]										
ompute (S	M) [%]									1	
Compute (S Memo	SM) [%]									)	
Compute (S Memo	SM) [%]	10.0	20.0 3	0.0 40.0	50.0 Speed Of Light	60.0 (SOL) [%]	70.0	80.0	90.0	 10	0.0



• Improvement in AI and Performance due to atom and neighbor loop fusing





			۵					
Executed Ipc Elapsed [inst/cycle]	1.62 (+6,197.40%)	SM Busy [%]	40.48 (+810.22%)					
Executed Ipc Active [inst/cycle]	1.62 (+809.91%)	Issue Slots Busy [%]	40.48 (+810.22%)					
Issued Ipc Active [inst/cycle]	1.62 (+810.22%)							
Balanced No pipeline is over-utilized.								
Pipe Utilization								
0,0	25.0	50.0	5.0 100.0					
FP64								
LSU								
FMA								
ALU								

✓ Memory Workload Analysis			All 🔹 🔎
Memory Throughput [Gbyte/second]	4.65 (+437.23%)	Mem Busy [%]	96.11 (+662.89%)
L1/TEX Hit Rate [%]	95.73 (+22.12%)	Max Bandwidth [%]	67.65 (+1,374.25%)
L2 Hit Rate [%]	98.82 (-0.93%)	Mem Pipes Busy [%]	36.19 (+5,451.48%)
L2 Compression Success Rate [%]	0 (+0.00%)	L2 Compression Ratio	0 (+0.00%)







19/20























## CrayPAT (Perftools)

- CrayPat specifically for use on Cray machines
- GUI and TUI tools
- Modules
  - perftools-base
    - required to load perftools or perftools-lite
  - perftools (full suite)
  - perftools-lite (fast, specific)
    - Additional perftools-lite modules (events, gpu, loops, hbm)
    - perftools-lite-gpu (GPU kernel and data movement with an even profile)





- Profiling steps for a Jacobi solver written in Fortran with MPI and OpenMP
- Code must be run in \$SCRATCH (or set an env to run it from elsewhere)
- Object (\*.o) files must be created in a separate step and remain present
- To visualize, use App2, launch terminal with X forwarding or use NX (<u>https://docs.nersc.gov/connect/nx/</u>)
- Run steps are as follows:

```
$ module load perftools-lite
$ module load PrgEnv-cray
$ ftn -c -fopenmp jacobi_mpiomp.F90
$ ftn -fopenmp -o jacobi_mpiomp jacobi_mpiomp.o
$ salloc --nodes 4 --qos interactive --time 01:00:00 --C cpu --account=mxxxx
$ srun -n 32 --cpu-bind=cores ./jacobi_mpiomp
```





############	##########	+#######	#######	####	##########	#########	*#########
#							#
#	CrayPat	-lite P	erforma	nce	Statistic	S	#
#							#
############	#########	+#######	#######	####	#########	#########	##########
CrayPat/X:	Version	22.06.0	Revisi	on 4	b5ab6256	05/21/22	02:03:49
Experiment:			lite	lit	e-samples		

Experiment:litelite-samplesNumber of PEs (MPI ranks):32Numbers of PEs per Node:8PEs on each of 4NodesNumbers of Threads per PE:2Number of Cores per Socket:64Execution start time:Wed Sep 21 11:00:59 2022System name and speed:nid0056662.334 GHz (nominal)AMDMilanCore Performance Boost:64 PEs have CPB capability

Avg Process Time:7.13 secsHigh Memory:6,850.6 MiBytes214.1 MiBytes per PEI/O Read Rate:14.231151 MiBytes/secI/O Write Rate:11.174613 MiBytes/sec





Notes for table 1: This table shows functions that have significant exclusive sample hits, averaged across ranks. For further explanation, use: pat\_report -v -0 samp\_profile ... Table 1: Profile by Function Samp% | Samp Imb. Imb. | Group Samp | Samp% | Function=[MAX10] PE=HIDE Thread=HIDE 100.0% | 687.2 | -- | -- | Total -----70.3% | 483.0 | -- | -- | USER || 53.3% | 366.3 | 16.7 | 4.5% | jacobi mpiomp .LOOP@li.61 || 16.8% | 115.7 | 34.3 | 23.6% | compute\_diff\_.LOOP@li.261 22.3% | 153.4 | -- | -- | ETC || 20.9% | 143.3 | 32.7 | 19.2% | cray memcpy ROME 6.3% | 43.4 | -- | -- | MPI 11------4.6% | 31.3 | 13.7 | 31.3% | MPI\_ALLREDUCE 1.7% | 11.5 | 16.5 | 60.8% | MPI SENDRECV 





Notes for table 2:

This table shows functions, and line numbers within functions, that have significant exclusive sample hits, averaged across ranks. For further explanation, use: pat_report -v -0 samp_profile+src
Table 2: Profile by Group, Function, and Line
Samp%       Samp       Imb.       Imb.       Group                 Samp       Samp%       Function=[MAX10]                                 Source                                 Line                                 PE=HIDE                                 Thread=HIDE
100.0%   687.2       Total
70.3%   483.0       USER
53.3%   366.3       jacobi_mpiompLOOP@li.61 3        n/namehta4/Tutorial/jacobi_mpiomp.F90
 4    33.5%   230.3   25.7   10.4%   line.63 4    19.4%   133.0   13.0   9.2%   line.66
==================================
 4    5.8%   39.9   22.1   36.8%   line.263 4    11.0%   75.4   23.6   24.6%   line.264
-    20.9%   143.3   32.7   19.2%  cray_memcpy_ROME 
4.6%   31.3   13.7   31.3%   MPI_ALLREDUCE    1.7%   11.5   16.5   60.8%   MPI_SENDRECV







Notes for table 3:

This table shows functions that have the most significant exclusive time, taking for each thread the average time across ranks. The imbalance percentage is relative to the team observed to participate in execution. Use -s th=ALL to see individual thread values. For further explanation, use: pat\_report -v -0 profile\_th\_pe ...

Table 3: Profile by Function Group and Function

Samp%   Samp   Im     Sa     	b.   Imb.   Te mp   Samp%   S: 	eam   Group ize   Function=[MAX10]   Thread=HIDE   PE=HIDE
100.0%   687.2		Total
70.3%   483.0	[ [	USER
53.3%   366.3      16.8%   115.7	1.3   0.7%   1.8   3.0%	2   jacobi_mpiompLOOP@li.61 2   compute_diffLOOP@li.261
22.3%   153.4		ETC
20.9%   143.3	4.5   6.3%	2  cray_memcpy_ROME
   6.3%   43.4	[ [	MPI
4.6%   31.3      1.7%   11.5   	 	1   MPI_ALLREDUCE 1   MPI_SENDRECV

\_\_\_\_\_\_

4/20







Notes for table 4:

This table shows energy and power usage for the nodes with the maximum, mean, and minimum usage, as well as the sum of usage over all nodes.

Energy and power for accelerators is also shown, if applicable. For further explanation, use: pat\_report -v -O program\_energy ...

Table 4: Program energy and power usage (from Cray PM)

Node		Node	Process		Node Id
Energy	I	Power (W)	Time	I	PE=HIDE
(L)	I			1	Thread=HIDE

\_





Notes for table 6:

This table show the average time and number of bytes written to each output file, taking the average over the number of ranks that wrote to the file. It also shows the number of write operations, and average rates.

\_\_\_\_\_\_

For further explanation, use: pat\_report -v -0 write\_stats ...

#### Table 6: File Output Stats by Filename

Avg	L	Avg	L	Write Rate	Ľ	Number	I	Avg	В	ytes/	I	<pre>File Name=!x/^/(proc sys)</pre>
Write	Ì	Write	Ĺ	MiBytes/sec	Î.	of	İ	Writes		Call	Ì	PE=HIDE
ime per	L	MiBytes	Ľ		Î	Writer		per			Ľ	
Writer	1	per	Ľ		Î	Ranks		Writer			L	
Rank	Î.	Writer	Ľ		Î.	ĵ		Rank			Ľ	
	l	Rank	I		I		I	L			I	
0.000091		0.002563		28.099071		1	1	101.0	1	26.60	1	stdout
0.000062	Ì.	0.000439	l	7.044082	1	6	Ì	7.5	Î.	61.33	1	_UnknownFile_

4/20







```
Notes for table 1:
```

This table shows functions that have significant exclusive time, averaged across ranks. For further explanation, use: pat\_report -v -0 profile ...

Table 1: Profile by Function Group and Function









## CrayPAT (Perftools) steps

- Similar to steps used for Perftools-lite
- Code must be run in \$SCRATCH

\$ pat\_build -u -g mpi <name of exec>

- Object (\*.o) files must be created in a separate step and maintained
- To visualize results using App2, launch terminal with X forwarding
- Run steps are as follows:

```
$ module unload darshan xalt
$ module load perftools
$ export PAT_RT_EXPDIR_NAME=./data_dir
$ ftn -c -fopenmp jacobi_mpiomp.F90
$ ftn -fopenmp -o jacobi_mpiomp jacobi_mpiomp.o
$ salloc --nodes 32 --qos interactive --time 01:00:00 --C cpu --account=mxxxx
$ pat_build -O apa jacobi_mpiomp
$ srun -n 32 ./jacobi_mpiomp+pat
```

#### https://www.nersc.gov/assets/Uploads/Perftools-Reveal-Levesque-Sept2022.pdf





## CrayPAT (Perftools) steps

Running the +pat compiled binary will generate .xf files in the data\_dir folder

6/20

- These .xf2 files have to be converted to ap2 format to visualize using App2
- Run steps are as follows:

```
$ cd ./data dir
$ pat report -f ap2 *.xf2
$ app2 result.ap2
```











7/20













Thank You and Welcome to NERSC!

