

Overview of HPCToolkit

Performance Tools for GPU-accelerated Supercomputing

John Mellor-Crummey, Rice University ALCF Hands-on HPC Workshop Oct. 11, 2023

HPCToolkit Project Team

- Rice University
 - HPCToolkit PI: Prof. John Mellor-Crummey
 - Research staff: Laksono Adhianto, Mark Krentel, Wil Phan, Matt Barnett
 - Contractor: Marty Itzkowitz
 - Students: Jonathon Anderson, Dragana Grbic, Vladimir Indjic, Yumeng Liu
- University of Wisconsin Madison
 - Dyninst PI: Prof. Barton Miller



Outline

- HPCToolkit performance tools for CPU and GPU-accelerated applications
 - Overview of HPCToolkit components and their workflow
 - HPCToolkit's graphical user interfaces
- Analyzing the performance of GPU-accelerated codes with HPCToolkit
 - GAMESS
 - Quicksilver
 - LAMMPS at Exascale
- Status
- Resources
- Hands-on directions



Rice University's HPCToolkit Performance Tools

Measure and analyze performance of CPU and GPU-accelerated applications

- Easy: profile unmodified application binaries
- Fast: low-overhead measurement
- · Informative: understand where an application spends its time and why
 - call path profiles associate metrics with application source code contexts
 - optional hierarchical traces to understand execution dynamics

Broad audience

- application developers
- framework developers
- runtime and tool developers
- Supported platforms
 - CPU: x86_64, Power, ARM
 - GPU: NVIDIA, AMD, Intel















Measurement of CPU and GPU-accelerated Applications

- Sampling using Linux timers and hardware counter overflows on the CPU
- Callbacks when GPU operations are launched and (sometimes) completed
- Event stream for GPU operations
- Instruction-level measurements
 - PC Samples (NVIDIA)
 - Binary instrumentation of GPU kernels (Intel)



Call Stack Unwinding to Attribute Costs in Context

- Unwind when timer or hardware counter overflows
 - measurement overhead proportional to sampling frequency rather than call frequency
- Unwind to capture context for events such as GPU kernel launches







Calling context tree

hpcrun: Measure CPU and/or GPU activity

- GPU profiling
 - hpcrun -e gpu=xxx <app>
- GPU PC sampling (NVIDIA GPU only)
 - hpcrun -e gpu=nvidia,pc <app>
- CPU and GPU Tracing (in addition to profiling)
 - hpcrun -e CPUTIME -e gpu=xxx -t <app>
- Use hpcrun with job launchers
 - jsrun -n 32 -g 1 -a 1 hpcrun -e gpu=xxx <app>

xxx ∈ {nvidia,amd,opencl,level0}

Profiles

- a calling context tree per thread
- instruction level measurements

CPU traces

- trace of call stack samples

GPU traces

- trace of call stacks that initiate GPU operations







hpcstruct: Analyze CPU and GPU Binaries Using Multiple Threads

• Usage

hpcstruct [--gpucfg yes] <measurement-directory>

- What it does
 - Recover program structure information
 - Files, functions, inlined templates or functions, loops, source lines
 - Analyze all CPU and GPU binaries that were measured by HPCToolkit using multithreading
 - Cache binary analysis results for reuse when analyzing other executions



Step 4:





hpcprof/hpcprof-mpi: Associate Measurements with Program Structure

Analyze data from modest executions with multithreading

hpcprof <measurement-directory>

• Analyze data from large executions with distributed-memory parallelism + multithreading

jsrun -n 2 -a 1 -c 18 -b packed hpcprof-mpi <measurement-directory>



Step 4:





Code-centric Analysis with hpcviewer





Understanding Temporal Behavior

- Profiling compresses out the temporal dimension
 - Temporal patterns, e.g. serial sections and dynamic load imbalance are invisible in profiles
- What can we do? Trace call path samples

17

- N times per second, take a call path sample of each thread
- Organize the samples for each thread along a time line
- View how the execution evolves left to right
- What do we view? assign each procedure a color; view a depth slice of an execution





Time-centric Analysis with hpcviewer



Argonne Leadership A multi-level call stack based view of execution over time

execution trace shown

Case Studies

- GAMESS: MPI, Fortran, OpenMP offloading
- Quicksilver: MPI, C++, CUDA
- LAMMPS: MPI, C++, Kokkos at exascale



rofile: gamess.00.x 🗮 Trace: gamess.00.x 🖼 Profile: gamess.00.x 🗮 Trace: gamess.00.x	- ++++	· (2 - H	-
ime Range: [0s, 53s] Cross Hair: (29s, RANK 0 GPUCONTEXT 1 GPUSTREAM 16)	Jeptn: 40	1.5	
	Call stack Statistics (GPU Idleness (Blame
nvkernel_gpu_rhf_j05_pppsF1L575_18_ [305a4090489511077c8b6ad43909d31f.gpubin]	<pre>cprogram root> main gamess brnchx_ energx_ wfn_</pre>		
	<pre>rhfcl_ twoei_ ompmod_ompmod_twoei_jk_ gpu_ompmod_twoei_jk_ gpu_rhf_j05_ppps_ nvomp_target [libnvor</pre>	mp.so]	
	launchTarget (libnvomp, launchHKTarget (libnvom hxLaunch (libnvomp,so) launchInternal (libnvom [I] targetLaunch launchInternal (libnvor	.so] mp.so] mp.so]	
	<pre><gpu kernel=""> nvkernel_gpu_rhf_j05_pp</gpu></pre>	ppsF1L575_18_	[305a40
s 2's 3's 4's 5's 6's 7's 8's 9's 10's 12's 14's 16's 18's 20's 22's 24's 26's 28's 30's 32's 34's 36's 38's 40's 42's 44's 46's 50's 52's			
th view Summary view	Mini map		

20 Argonne Leadership Computing Facility GAMESS original

All CPU threads and GPU streams ne 🔥

e View Filter Help		and the set of the set
rofile: gamess.00.x 🗮 Trace: gamess.00.x 🖻 Profile: gamess.00.x 🗮 T	ace: gamess.00.x 편 Profile: gamess.00.x 🗮 Trace: gamess.00.x 💌 Profile: games	s.01.x 💐 Trace: gamess.01.x 🔄 Profile: gamess.00.x 🗮 Trace: gamess.00.x 🧮
	Select rank to display	
n utau	Check all Uncheck all Regular expression	
Time Range: [0s 53s] (ross Hair: (42s RANK A THREAD 10)		Depth: 40 - +
	Filter: GPU	Call stack Statistics GDH Idlaness Blame
	Paulo or threads	Catt Statk Statistics ore intelless Stalle
	1 RANK 0 GPUCONTEXT 1 GPUSTREAM 16	<thread root=""></thread>
	2 RANK 1 GPUCONTEXT 2 GPUSTREAM 26	threadpoolEntryPoint [Libnvomp.so]
	3 RANK 2 GPUCONTEXT 2 GPUSTREAM 26	hxltxecuteHostTreeBarrierwithTasks [(ibnvomp.
	4 RANK 3 GPUCONTEXT 2 GPUSTREAM 26	[I] executeHostTreeBarrier
		[I] waitForNeighborThreads
		hxAddressWalt [[lbnvomp.so]
		syscall [libc-2.31.so]
	And the second se	L LOUI DE LEURA LEUR DE LEUR LEUR LEUR
and the second state of the se		
	Cancel	
s 2s 3s 4s 5s 6s 7s 8s 9s 10s 12s 14s 16s 18s 2	s 22s 24s 26s 28s 30s 32s 34s 36s 38s 40s 42s	44s 46s 48s 50s 52s
th view Summary view		Mini map

21 Argonne Leadership Computing Facility GAMESS original

All CPU threads and GPU streams ne



22 Argonne Leadership Computing Facility GAMESS original

All GPU streams, whole execution



23 Argonne Leadership Computing Facility GAMESS original

GPU streams: 1 iteration Argonne



1 + 6 fx 😰 🕅 🗮 At 🖉 11 - 📽

Scope	REALTINE (sec): Sum (I)	REALTIME (sec): Sum (E)	GPUOP (sec): Sum (I)	GPUOP (sec); Sum (E)	GNER (sec): Sum (I)	GRER (sec): Sun (E)	GXCOPY (sec): Sum (I)	GOCOPY (sec): Sum (E)
A Experiment Aggregate Metrics	3.48e+03 100.	0% 3.48e+03 100.0	1.66e+02 100.	0% 1.66e+02 100.0%	1.66e+02 100.01	1.66e+02 100.0%	2.64e-01 100.0%	2.64e-01 10
⊿ <program root=""></program>	2.04e+02 5.	9%	1.66e+02 100.	6%	1.66e+02 100.01		2.64e-01 100.0%	
▲ » main [gamess.00.x]	2.04e+02 5.	9%	1.66e+02 100.	0%	1.66e+02 100.01		2.64e-01 100.0%	
🖌 » gamess	2.04e+02 5.	9%	1.66e+02 100.	0%	1.66e+02 100.0%		2.64e-01 100.0%	
▲ 772 > brnchx_	2.03e+02 5.	8%	1.66e+02 100.	0%	1.66e+02 100.04		2.64e-01 100.0%	
▲ 1095 energx	2.03e+02 5.	8%	1.66e+02 100.	.0%	1.66e+02 100.04		2.64e-01 100.0%	
▲ loop at gamess.F: 1316	2.03e+02 5.	8%	1.66e+02 100.	on.	1.66e+02 100.01		2.64e-01 100.0%	
loop at gamess.F: 1436	1.97e+02 5.	7%	1.66e+02 100.	e%	1.66e+02 100.01		2.64e-01 100.0%	
▲ loop at gamess.F: 1436	1.97e+02 5.	7%	1.66e+02 100.	0%	1.66e+02 100.01		2.64e-01 100.0%	
▲ 1440	1.97e+02 5.	7%	1.66e+02 100.	on	1.66e+02 100.01		2.64e-01 100.0%	
Loop at gamess.F: 2645	1.97e+02 5.	7%	1.66e+02 100.	.en	1.66e+02 100.0%		2.64e-01 100.0%	
▲ 2568 w rhfcl_	1.97e+02 5.	7% 1.04e-02 0.0	1.66e+02 100.	0%	1.66e+02 100.0%		2.64e-01 100.0%	
Loop at rhfuhf.f: 2678	1.96e+02 5.	7% 5.25e-03 0.0	1.66e+02 100.	0%	1.66e+02 100.0%		2.64e-01 100.0%	
▲ loop at rhfuhf.f: 2723	1.96e+02 5.	6%	1.66e+02 100.	0%	1.66e+02 100.05		2.64e-01 100.0%	
▲ loop at rhfuhf.f: 2723	1.96e+02 5.	6%	1.66e+02 100.	.0%	1.66e+02 100.0%		2.64e-01 100.0%	
∠ 2859 * twoei_	1.69e+02 4.	9%	1.66e+02 100.	0%	1.66e+02 100.01		2.64e-01 100.0%	
▲ 3994 >> ompmod_ompmod_twoei_jk_	1.69e+02 4.	9%	1.66e+02 100.	0%	1.66e+02 100.05		2.64e-01 100.0%	
∠ 246 >> gpu_onpmod_twoei_jk_	1.69e+02 4.	9%	1.66e+02 100.	0N	1.66e+02 100.01		2.64e-01 100.0%	
▶ 656 >> gpu_rhf_j16_dppp_	2.98e+01 0.	9% 1.00e-02 0.0	2.97e+01 17.	91	2.97e+01 17.91		1.07e-02 4.1%	
\$89gpurhf196_pppp	2.74e+01 0.	85 5.00e-03 0.0	2.74e+01 16.	51	2.74e+01 15.5t		8.56e-03 3.2%	
Kana Kana	a. 11. a.a. a.	*** * ** **		ani.			a ta .aa.m	

24 Argonne Leadership Computing Facility

GAMESS original



Time-centric Analysis: GAMES 5 nodes, 40 ranks, 20 GPUs on Perlmutter



25 Argonne Leadership Computing Facility

GAMESS improved

CPU Threads and GPU Streams

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



26 Argonne Leadership Computing Facility GAMESS improved



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter





Quicksilver: Detailed analysis within a Kernel using PC Sampling

hpcviewer

File View Filter Help

- 0

🖷 Pro	ile: qs	
main.	CollisionEvent.cc X	
69	int uniqueNumber = monteCarlo->_materialDatabase->_mat[globalMatIndex]jso[isoIndex]gid;	
70	<pre>int numReacts = monteCarlo->_nuclearData->getNumberReactions(uniqueNumber);</pre>	
71	for (int reactIndex = 0; reactIndex < numReacts; reactIndex++)	
72	{	
73	currentCrossSection -= macroscopicCrossSection(monteCarlo, reactIndex, mc_particle.domain, mc_particle.cell,	
74	isoIndex, mc_particle.energy_group);	
75	if (currentCrossSection < 0)	
76	{	
77	selectedIso = isoIndex;	
78	selectedUniqueNumber = uniqueNumber;	
79	selectedReact = reactIndex;	
80	break;	
81		
82		
83		
84	ssert(selectedIso != -1);	
85		

Top-down view Bottom-up view Flat view

🕆 🖶 🍝 £ 🐖 🕅 🚍 🗛 🖌 🗤 📽

Scope	GINS: Sun	n (I) 🔻	GINS: Sum	(E)	GINS:STL_ANY: Sum (:	I) GINS:STL_ANY:	Sum (E)	GINS:STL_IFET: Sum (I)	GINS:STL_IFET: Sum (E)	GINS:STL_IDEP:
<pre>4 14 » [1] cudaLaunchKernel<char></char></pre>	1.30e+11	100.0%			1.19e+11 100	0.0%		5.2/e+09 100.0	*6	9.34e+
▲ 211 » cudaLaunchKernel [qs]	1.30e+11	100.0%			1.19e+11 100).0%		5.27e+09 100.0	%	9.34e+
∡ » <gpu kernel=""></gpu>	1.30e+11	100.0%			1.19e+11 100).0%		5.27e+09 100.0	%	9.34e+
A » CycleTrackingKernel(MonteCarlo*, int, ParticleVault*, ParticleVau	1.30e+11	100.0%	4.08e+07	0.0%	1.19e+11 100).0% 3.62e+	07 0.0	5.27e+09 100.0	% 2.11e+07 0.49	9.34e+
I32 » CycleTrackingGuts(MonteCarlo*, int, ParticleVault*, Particle	1.30e+11	100.0%	9.03e+09	7.0%	1.19e+11 100).0% 9.01e+	09 7.6	5.24e+09 99.5	% 8.98e+06 0.2%	9.32e+
▲ 26 » [I] CycleTrackingFunction(MonteCarlo*, MC_Particle&, int, P	8.36e+10	64.4%	4.12e+08	0.3%	7.25e+10 61	1.1% 3.65e+	08 0.3	5.21e+09 98.9	% 1.02e+08 1.9	9.25e+
loop at CycleTracking.cc: 118	8.35e+10	64.3%	3.76e+08	0.3%	7.25e+10 61	.1% 3.34e+	08 0.3	5.21e+09 98.8	% 9.90e+07 1.9%	9.24e+
▲ 63 » CollisionEvent(MonteCarlo*, MC_Particle&, unsigned int) [5.20e+10	40.1%	4.99e+09	3.8%	4.44e+10 37	7.4% 4.02e+	09 3.4	3.85e+09 73.1	% 4.89e+08 9.3	6.37e+
loop at CollisionEvent.cc: 67	4.09e+10	31.5%	8.15e+08	0.6%	3.42e+10 28	3.8% 6.54e+	08 0.6	3.54e+09 67.1	% 1.27e+08 2.4%	5.67e+
loop at CollisionEvent.cc: 71	3.85e+10	29.6%	2.70e+09	2.1%	3.22e+10 27	7.1% 2.06e+	09 1.7	3.27e+09 62.0	% 2.28e+08 4.39	5.33e+
▲ 73 ≫ macroscopicCrossSection(MonteCarlo*, int, int, i	3.58e+10	27.5%	1.22e+10	9.4%	3.01e+10 25	5.4% 9.85e+	09 8.3	3.04e+09 57.7	% 1.79e+09 33.9%	4.60e+
▲ 41 » NuclearData::getReactionCrossSection(unsigned int, u	2.09e+10	16.1%	1.09e+10	8.4%	1.79e+10 15	j.1% 9.42e+	09 7.9	i 1.26e+09 23.8	% 6.68e+08 12.7°	s 2.19e⊣
253 » [I] NuclearDataReaction::getCrossSection(unsigned	6.89e+09	5.3%	3.77e+09	2.9%	5.86e+09 4	1.9% 3.32e+	09 2.8	6 2.25e+08 4.3	% 8.24e+07 1.6%	8.86e+
NuclearData.cc: 253	6.28e+09	4.8%	6.28e+09	4.8%	5.66e+09 4	1.8% 5.66e+	09 4.8	4.76e+08 9.0	% 4.76e+08 9.0%	6.lle+
NuclearData.cc: 251	1.85e+09	1.4%	1.85e+09	1.4%	1.64e+09 1	1.64e+	09 1.4	8.12e+07 1.5	% 8.12e+07 1.5%	s 2.47e+
NuclearData.cc: 248	1.61e+09	1.2%	1.61e+09	1.2%	1.18e+09 1	1.18e+	09 1.0	1.10e+08 2.1	% 1.10e+08 2.19	s 3.62e+
<pre>> 252 » [I] qs_vector<nucleardataspecies>::operator[](int)</nucleardataspecies></pre>	1.29e+09	1.0%	1.29e+09	1.0%	1.14e+09 1	1.14e+	09 1.0	5 7.37e+04 0.0	% 7.37e+04 0.0%	1.24e+
NuclearData.cc: 252	1.12e+09	0.9%	1.12e+09	0.9%	9.48e+08 0).8% 9.48e+	08 0.8	s 3.44e+05 0.0	% 3.44e+05 0.09	s 2.50e+
<pre>> 252 » [I] qs_vector<nucleardatareaction>::size() const</nucleardatareaction></pre>	9.41e+08	0.7%	9.41e+08	0.7%	8.17e+08 0).7% 8.17e+	08 0.7	5		4.63e+
•	2 20-100	0.00	2 20-100	0.00	0 4100		<u> </u>	1 41-100 0 7	1 41-100 0 70	7 . 77



Quicksilver: Detailed analysis within a Kernel using PC Sampling

Scope
▲ 14 » [1] cudaLaunchKernel <char></char>
✓ 211 » cudaLaunchKernel [qs]
⊿ » <gpu kernel=""></gpu>
A >> CycleTrackingKernel(MonteCarlo*, int, ParticleVault*, ParticleVau
I32 >> CycleTrackingGuts(MonteCarlo*, int, ParticleVault*, Particle
26 » [I] CycleTrackingFunction(MonteCarlo*, MC_Particle&, int, P
loop at CycleTracking.cc: 118
▲ 63 » CollisionEvent(MonteCarlo*, MC_Particle&, unsigned int) [
loop at CollisionEvent.cc: 67
▲ loop at CollisionEvent.cc: 71
73 » macroscopicCrossSection(MonteCarlo*, int, int, i
41 >> NuclearData::getReactionCrossSection(unsigned int, u
253 » [I] NuclearDataReaction::getCrossSection(unsigned
NuclearData.cc: 253
NuclearData.cc: 251
NuclearData.cc: 248
252 » [I] qs_vector <nucleardataspecies>::operator[](int)</nucleardataspecies>
NuclearData.cc: 252
<pre>> 252 » [I] qs_vector<nucleardatareaction>::size() const</nucleardatareaction></pre>















31 Argonne Le





32

Argonne



1 00-07 0.00

33 Argonne L

LAMMPS_NS:CommKokkos:borders(



8.37e+00

0.10-100

0.1

1.10e+06

0.07--00

4.8%



🕆 🗄 🌜 🕼 🗊 🕅 🖬 🔺 🗚 🖬 🗮 🕷

Scope	(E)	GMSET:COUNT: Sum (I)	GMSET COUNT. Sum (E)	GXCOPY:COUNT: Sum	i (I) GXI	COPY:COUNT: Sum (E)	REALTIME (sec): S	um (I) -	REALTIME (sec) S	sum (E)
 Experiment Aggregate Metrics 	0.0%	3.46e+06 100.0%	3.46e+06 100.0%	1.45e+09 10	0.0%	1.45e+09 100.0%	2.30e+07	100.0%	2.30e+07	100.
<pre>▲ <program root=""></program></pre>		3.46e+06 100.0%		1.45e+09 100	0.0%		2.30e+07	100.0%	6	
🔺 » main [Imp]		3.46e+06 100.0%		1.45e+09 100	0.0%		2.30e+07	100.0%	6 1.00e-02	0.1
▲ 98 » LAMMPS_NS:Input:file() [liblammps.so.0]		2.15e+06 62.1%		1.45e+09 99	9.9%		2.19e+07	95.2%	5.66e-02	0.1
A » LAMMPS_NS:Input:execute_command() [li		2.15e+06 62.1%	·	1.45e+09 99	9.9%		2.18e+07	94.8%	6 2.02e-01	0.1
▲ » LAMMPS_NS:Run::command(int, char**)		1.01e+06 29.3%		1.44e+09 99	9.4%		2.14e+07	92.9%	6.02e-02	0.0
▲ » LAMMPS_NS::VerletKokkos::run(int) [libl		5.33e+05 15.4%	(1.44e+09 99	9.2%		2.13e+07	92.6%	6 2.66e+02	0.1
A » LAMMPS_NS CommBrick forward_c							5.03e+06	21.9%	3.49e+02	0.1
MPI_Send [libmpi_gnu_91 so 12.0.0]							2.45e+06	10.6%	4.75e+02	θ.)
MPI_Wait[libmpi_gnu_91.so.12.0.0]							1.81e+06	7.9%	6 3.60e+01	0.1
» LAMMPS_NS AtomVec pack_comm							7.11e+05	3.1%	6 7.11e+05	3.
MPI_Irecv [libmpi_gnu_91.so.12.0.0]							6.16e+04	0.3%	6 1.91e+02	0.1
liblammps.so.0@0x425b2c3							2.61e-02	0.0%	6 2.61e-02	0.1
LAMMPS_NS_ModifyKokkos_final_inte_				2.85e+08 1	9.6%		4.72e+06	20.5%	6 1.13e+02	0.1
LAMMPS_NS:AtomVecAtomicKokkos				2.46e+08 1	7.0%		3.44e+06	14.9%	6 7.25e+01	0.1
» LAMMPS_NS:CommKokkos: forward				2.46e+08 1	7.0%		3.42e+06	14.8%	6.48e+01	0.4
» LAMMPS_NS:NeighborKokkos:build		5.01e+05 14.5%		1.04e+08	7.2%		1.59e+06	6.9%	6 2.29e+01	0.1
» LAMMPS_NS: CommKokkos: borders(1.10e+06	4.8%	6 8.37e+00	0.1
K I MARO NO. O				1 20-107	0.00		0.0705	0.00	0.1000	-

Argonne 🕰

34 Argonne Le

LAMMPS on Frontier: 8K nodes, 64K MPI ranks + GPU times

Kernel duration of microseconds



35 Argonne Leadership Computing Facility



HPCToolkit Status on AMD, Intel, NVIDIA GPUs

- Heterogeneous profiles
- GPU operation traces
- Hardware counters to measure GPU kernels
- Instruction-level measurement within GPU kernels
 - NVIDIA: PC sampling
 - Intel: binary instrumentation
 - AMD: PC sampling coming soon!



HPCToolkit Documentation and Training

User Manual

- http://www.hpctoolkit.org/manual/HPCToolkit-users-manual.pdf
- Installing HPCToolkit's hpcviewer on your Laptop
 - http://www.hpctoolkit.org/download.html
- Training slides and videos: http://www.hpctoolkit.org/training.html
 - Introduction to HPCToolkit [Youtube (13:22)] [Slides]
 - Sampling-based Performance Analysis with HPCToolkit [Youtube (23:09)] [Slides]
 - Identifying Scalability Bottlenecks with HPCToolkit [Youtube (19:27)] [Slides]
 - Analyzing GPU-accelerated Applications with HPCToolkit [Youtube (23:59)] [Slides]
 - Using HPCToolkit to Analyze the Performance of GPU-accelerated Applications [Youtube (1:27:35)] [Slides]
 - Analyzing GPU-accelerated Applications Using HPCToolkit [Youtube (34:02)] [Slides]
 - HPCToolkit Graphical User Interface [Youtube (35:03)] [Slides]
 - Analyzing CPU Applications with HPCToolkit [Youtube (2:24:30)] [Slides]
- Downloading and Installing HPCToolkit
 - http://www.hpctoolkit.org/software-instructions.html



Want Some Help?

Join our hpctoolkit-ECP Slack workspace

- https://join.slack.com/t/hpctoolkit-ecp/shared_invite/ zt-24rtkvwma-4HNYe~TiwFwEiJpH~RqUuw
- Send email to our mailing list
 - hpctoolkit-forum@rice.edu



Hands-on Directions

- Log into Polaris with X11 forwarding for hpcviewer
 - ssh -Y <username>@polaris.alcf.anl.gov
- Download example programs to measure and analyze
 - git clone https://github.com/hpctoolkit-tutorial-examples
- Load HPCToolkit into your environment
 - module use /soft/perftools/hpctoolkit/polaris/modulefiles
 - module load hpctoolkit
- Set up environment variables for running HPCToolkit examples at the workshop
 - export HPCTOOLKIT_TUTORIAL_PROJECTID=fallwkshp23
 - export HPCTOOLKIT_TUTORIAL_RESERVATION=fallws23single
 - export HPCTOOLKIT_HPCSTRUCT_CACHE=\$HOME/.hpctoolkit/hpcstruct-cache



Working with HPCToolkit's Tutorial Examples

- **Examples in** hpctoolkit-tutorial-examples/examples/gpu
 - Quicksilver (highly recommended)
 - LAGHOS
 - MiniQMC
 - PeleC
 - LAMMPS
- Working with an example
 - cd <example-name>
 - source setup-env/polaris.sh
 - make build
 - make # enumerates commands for running jobs and inspecting their results
- Pro tip
 - watch ls -1 # wait until a done file appears, e.g. log.run-pc.done to indicate your data is ready
 - also look at the log.*.stderr and log*.out logs
- 40 Argonne Leadership Computing Facility

