

Human-Oriented Discussions on Performance Analysis

Scalable Tools Workshop '23
Connor Scully-Allison

Participants

- Connor Scully-Allison (The University of Utah)
- Kate Isaacs (The University of Utah)
- Wileam Phan (Rice University)
- Sameer Shende (University of Oregon)
- Jan Laukemann (I'm University of Erlangen-Nuremberg / NHR@FAU)
- Shadmaan Hye (University of Utah)
- David Boehme (LLNL)
- William Jalby (UVSQ/UPSaclay)
- Stephanie Brink (LLNL)
- Kathleen Shoga (LLNL)
- Rahat Zaman (University of Utah)
- David Montoya (Trenza)
- Mahesh Rajan (Trenza)

Working Group Goals

- What groups of users exist in the performance tools space?
- Who are they?
 - What do we know about them now?
 - What do we still need to know?
 - How do they differ from us?
- What can we do to understand them better?

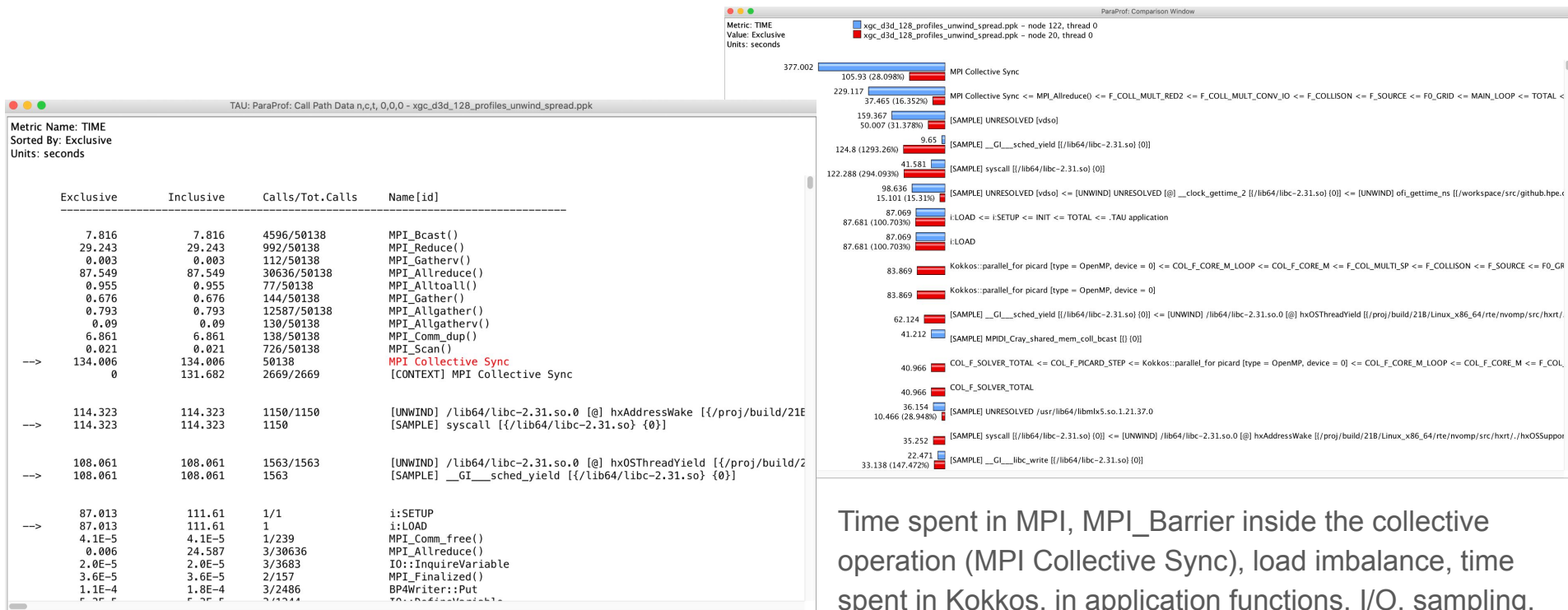
Exercise: Who is your user?

- Code developers are the people we expect to use the tools. Code runners are the people who generate the configurations and performance problems. There isn't always a good channel for the configurations/data to pass back to the developers.
- Code developers are encouraged to add features over performance.
- Code developer teams need a performance champion — or tool developer champions need to come in and set them up and create internal experts.
- People don't want to work with performance tools unless they absolutely need them but then there's a learning curve. Currently community looking into encouraging them to look at performance continuously.
- There are probably 5-6 kinds of users. Which should we prioritize.
- Users in private sectors can be very secretive about source code access and could benefit from tools that perform analysis without any annotation
- Not all users are code developers. Many are domain experts and need to relate performance data back to the domain execution

Exercise: How do you differ?

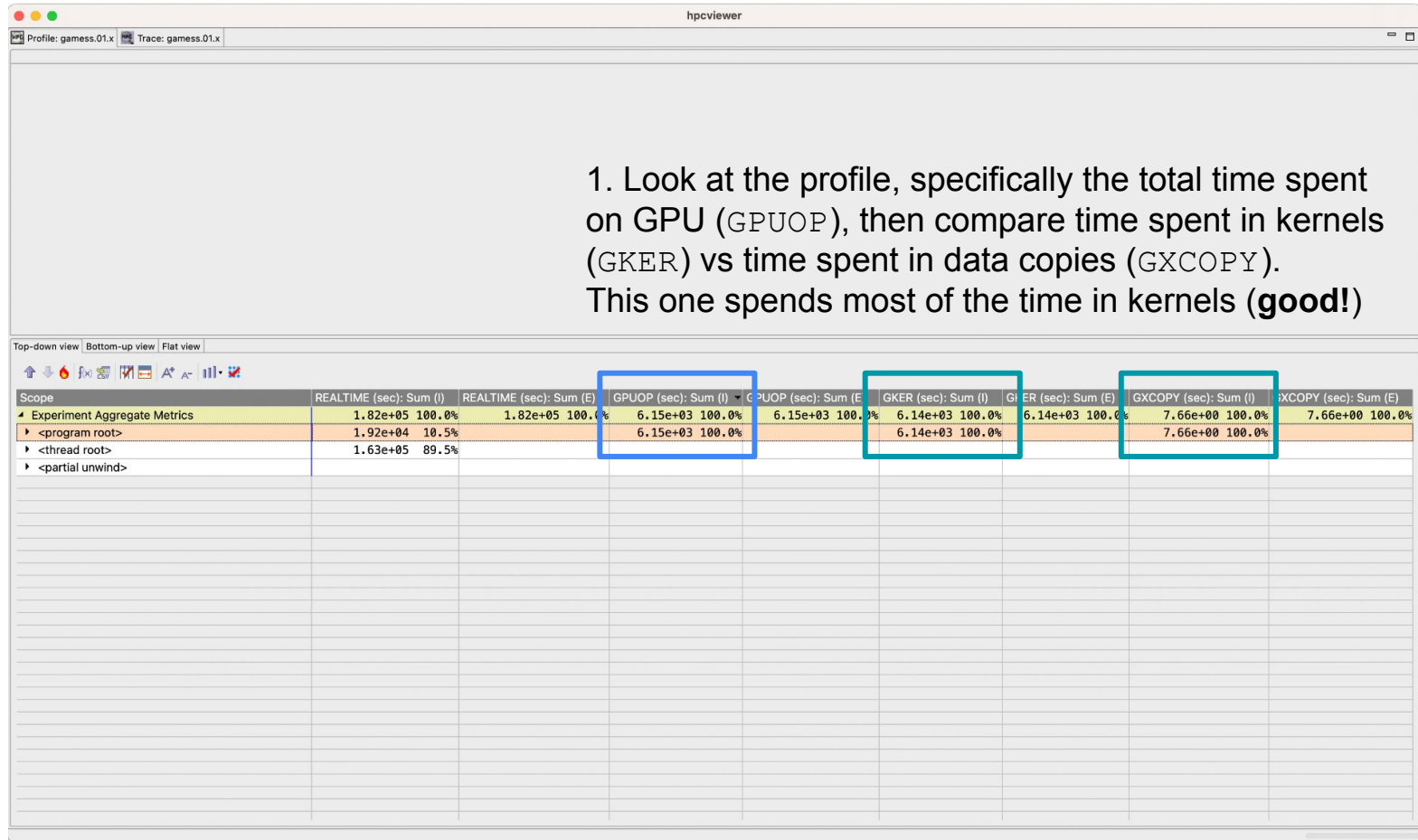
- Tool developers have comparatively structured workflows for performance analysis compared to their users
 - With significant foundation understanding of where performance issues commonly lie
- Tool developers are frustrated by the performance and usability of visualization in tools
 - “Zooming on a trace should be like google maps”
- Legacy data formats hold us back from performant GUI development

TAU performance data explains why MPI collectives take as long as they do!



Time spent in MPI, MPI_Barrier inside the collective operation (MPI Collective Sync), load imbalance, time spent in Kokkos, in application functions, I/O, sampling, system calls.

HPCToolkit analysis of GAMESS 5-node run on Perlmutter



HPCToolkit analysis of GAMESS 5-node run on Perlmutter

hpcviewer

Profile: gamess.01.x Trace: gamess.01.x

```
tgpu_ompmmod_tdhf_apb_sp.F90 X
649 !$omp target teams distribute parallel do simd &
650 !$omp num_teams(nteams) thread_limit(limitthread) &
651 !$omp private(i_tmp, iijjkkll, iijj_tmp, kkll_tmp) &
652 !$omp private(iijj, kkll, ii_TMP, jj_TMP, kk_TMP, ll_TMP) &
653 !$omp private(ii, jj, kk, ll) &
654 !$omp shared(N_BRA) &
655 !$omp shared(XINT_BRA) private(IQUART) &
656 !$omp shared(IA, DA) &
657 !$omp shared(CO, KTYPE, KLOC, KMIN, KMAX, KSTART, KNG, EX, CS, CP) &
658 !$omp shared(RFINC, FGRID, XGRID, RMR, CMAX) &
659 !$omp shared(FA) &
660 !$omp private(gpople) &
661 !$omp reduction(+:nintn)
662 DO IQUART=nshell_start_, nshell_end_
663   iijjkkll = IQUART-1
664   iijj_tmp = (1 + sqrt(one+eight*iijjkkll))/2
665   kkll_tmp = iijjkkll - iijj_tmp*(iijj_tmp-1)/2+1
666   if(XINT_BRA(iijj_tmp)*XINT_BRA(kkll_tmp) .lt. cutoff_denmax) cycle
667
```

2. Sort by GKER and click the “flame” button to find the most expensive GPU kernel
(This looks the same as their 1-node runs)

Top-down view Bottom-up view Flat view

Scope	REALTIME (sec): Sum (I)	REALTIME (sec): Sum (E)	GPUOP (sec): Sum (I)	GPUOP (sec): Sum (E)	GKER (sec): Sum (I)	GKER (sec): Sum (E)	GX
3329 » gpcubg_dyn_							
loop at cphf_tddnlf: 96	5.09e+03 2.8%		4.22e+03 68.6%		4.22e+03 68.6%		
loop at cphf_tddnlf: 192	4.40e+03 2.4%		3.65e+03 59.3%		3.64e+03 59.3%		
184 » aocptd_dyn_	4.40e+03 2.4%	5.00e-01 0.0%	3.65e+03 59.3%		3.64e+03 59.3%		
610 » cpdyn_apbx_	2.36e+03 1.3%	1.76e+00 0.0%	1.97e+03 32.0%		1.97e+03 32.0%		
536 » twoei_cphfdyn_apb_	2.33e+03 1.3%	5.00e-03 0.0%	1.97e+03 32.0%		1.97e+03 32.0%		
1224 » twoei_cphfdyn_omp_twoei_cphfdyn_apb_omp_	2.02e+03 1.1%		1.97e+03 32.0%		1.97e+03 32.0%		
174 » gpu_ompmmod_tdhf_apb_gpu_twoei_cphfdyn_apb_omp_	2.02e+03 1.1%		1.97e+03 32.0%		1.97e+03 32.0%		
352 » gpu_tdhf_apb_i06_pppp_	1.50e+03 0.8%	8.00e-02 0.0%	1.49e+03 24.2%		1.49e+03 24.3%		
689 » __nvomp_target [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
1294 » launchTarget [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
1189 » launchHXTarget [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
437 » hxLaunch [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
163 » launchInternal [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
573 » [] targetLaunch	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
525 » launchInternal [libnvomp.so]	1.49e+03 0.8%		1.49e+03 24.2%		1.49e+03 24.3%		
3062 » <gpu kernel>			1.49e+03 24.2%		1.49e+03 24.3%		
» nvkernel_gpu_tdhf_apb_i06_pppp_F1649_22_ [50...			1.49e+03 24.2%	1.49e+03 24.2%	1.49e+03 24.3%	1.49e+03 24.3%	
» hxCuda.c: 3708	1.49e+03 0.8%						
» hxCuda.c: 3682	2.50e-02 0.0%						
3708 » <gpu sync>							
3682 » <gpu kernel>							

HPCToolkit analysis of GAMESS 5-node run on Perlmutter



What else should we be asking? What do we want to know about them?

- How often is analysis being done?
- Granularity of what they are looking for/ their interest.
 - “See everything at absolutely no overhead”
- Where does a user lie on the interactive non interactive analysis spectrum
- Users are interested in seeing data related to abstractions that they are familiar with
 - Annotations that they have inserted into the code
 - “Kokkos parallel for picard”
 - They want to relate the performance to the domain application function

What actionable steps can we take to understand our users better?

- Provide clear use cases/samples alongside API Documentation
 - **Quickstart for <class of user>**
 - This makes it transferable between different users on the same team
 - Provide a template github repo alongside documentation
 - Keep tutorials short
- Acquaint new users with the diversity of commonly used tools
 - There is few opportunities to try new tools and they are often driven by problems
 - **A reference document of some variety that guides tool developers**
 - **Provide pros and cons of the tools**
- We need to measure user workflows and integrate them better.
 - **HCI experts can help here**