Human-Oriented Discussions on Performance Analysis - Outbrief

Scalable Tools Workshop ‘23
Connor Scully-Allison

https://bit.ly/stw-humans

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)

https://bit.ly/stw-humans

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?

- User Distinction
  - Code developers are the people we expect to use the tools.
  - Code runners are the people who generate the configurations and performance problems.
    - Domain expert code runners need to relate performance data back to the domain execution
- 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.

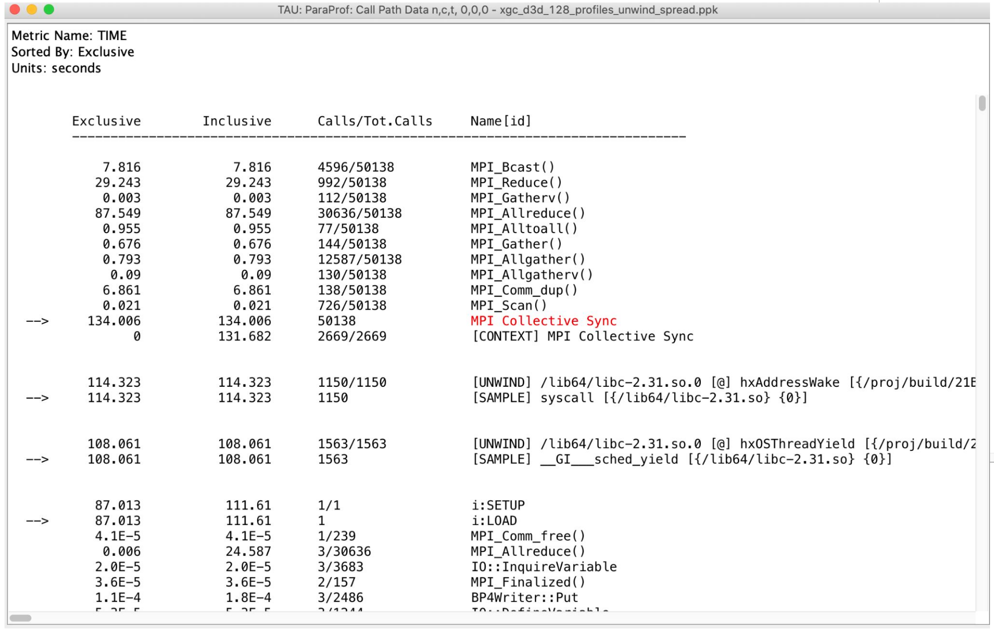
Exercise: Who is your user? - Part 2

- People don’t want to work with performance tools unless they absolutely need them
  - Then there’s a learning curve.
  - 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.

Exercise: How do you differ?

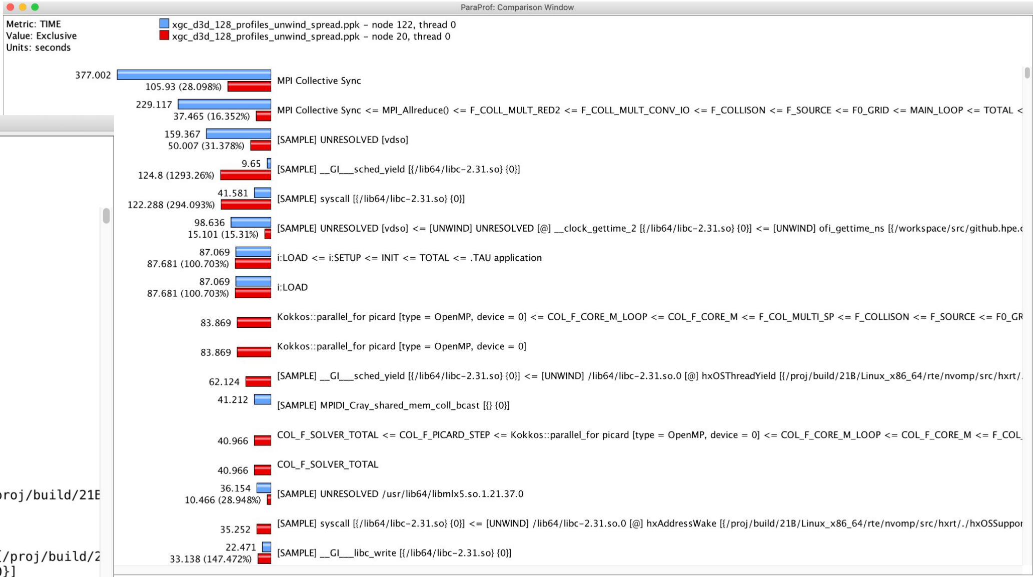
- Tool developers have comparatively structured workflows for performance analysis compared to their users
    - With significant foundation understanding of where performance issues commonly lie
- Tool developers are frustrated by the performance and usability of visualization in tools
    - “Zooming 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!



TAU: ParaProf: Call Path Data n,c,t, 0,0,0 - xgc_d3d_128_profiles_unwind_spread.ppk

```
Metric Name: TIME
Sorted By: Exclusive
Units: seconds


       Exclusive        Inclusive      Calls/Tot.Calls     Name[id]
       --------------------------------------------------------------------------------

           7.816            7.816      4596/50138          MPI_Bcast()
          29.243           29.243      992/50138           MPI_Reduce()
           0.003            0.003      112/50138           MPI_Gatherv()
          87.549           87.549      30636/50138         MPI_Allreduce()
           0.955            0.955      77/50138            MPI_Alltoall()
           0.676            0.676      144/50138           MPI_Gather()
           0.793            0.793      12587/50138         MPI_Allgather()
            0.09             0.09      130/50138           MPI_Allgatherv()
           6.861            6.861      138/50138           MPI_Comm_dup()
           0.021            0.021      726/50138           MPI_Scan()
-->      134.006          134.006      50138               MPI Collective Sync
               0          131.682      2669/2669           [CONTEXT] MPI Collective Sync


         114.323          114.323      1150/1150           [UNWIND] /lib64/libc-2.31.so.0 [@] hxAddressWake [{/proj/build/21E
-->      114.323          114.323      1150                [SAMPLE] syscall [{/lib64/libc-2.31.so} {0}]


         108.061          108.061      1563/1563           [UNWIND] /lib64/libc-2.31.so.0 [@] hxOSThreadYield [{/proj/build/2
-->      108.061          108.061      1563                [SAMPLE] __GI___sched_yield [{/lib64/libc-2.31.so} {0}]


          87.013           111.61      1/1                 i:SETUP
-->       87.013           111.61      1                   i:LOAD
          4.1E-5           4.1E-5      1/239               MPI_Comm_free()
           0.006           24.587      3/30636             MPI_Allreduce()
          2.0E-5           2.0E-5      3/3683              IO::InquireVariable
          3.6E-5           3.6E-5      2/157               MPI_Finalized()
          1.1E-4           1.8E-4      3/2486              BP4Writer::Put
```



ParaProf: Comparison Window

Metric: TIME
Value: Exclusive
Units: seconds

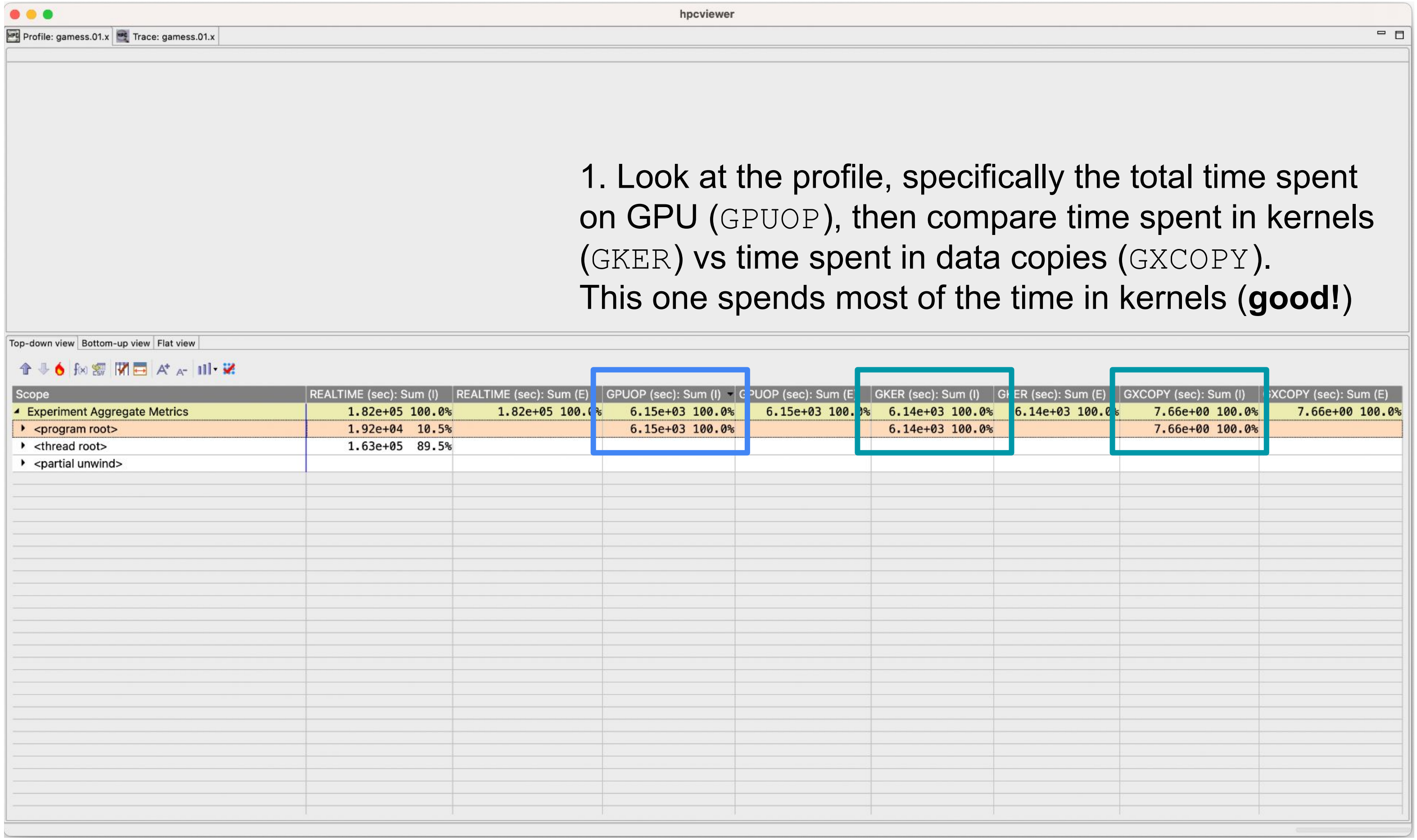
🟦 xgc_d3d_128_profiles_unwind_spread.ppk - node 122, thread 0
🟥 xgc_d3d_128_profiles_unwind_spread.ppk - node 20, thread 0

```
377.002
105.93 (28.098%)       MPI Collective Sync
229.117
37.465 (16.352%)       MPI Collective Sync <= MPI_Allreduce() <= F_COLL_MULT_RED2 <= F_COLL_MULT_CONV_IO <= F_COLLISON <= F_SOURCE <= F0_GRID <= MAIN_LOOP <= TOTAL <
159.367
50.007 (31.378%)       [SAMPLE] UNRESOLVED [vdso]
9.65
124.8 (1293.26%)       [SAMPLE] __GI___sched_yield [{/lib64/libc-2.31.so} {0}]
41.581
122.288 (294.093%)     [SAMPLE] syscall [{/lib64/libc-2.31.so} {0}]
98.636
15.101 (15.31%)        [SAMPLE] UNRESOLVED [vdso] <= [UNWIND] UNRESOLVED [@] __clock_gettime_2 [{/lib64/libc-2.31.so} {0}] <= [UNWIND] ofi_gettime_ns [{/workspace/src/github.hpe.
87.069
87.681 (100.703%)      i:LOAD <= i:SETUP <= INIT <= TOTAL <= .TAU application
87.069
87.681 (100.703%)      i:LOAD
83.869                 Kokkos::parallel_for picard [type = OpenMP, device = 0] <= COL_F_CORE_M_LOOP <= COL_F_CORE_M <= F_COL_MULTI_SP <= F_COLLISON <= F_SOURCE <= F0_GR
83.869                 Kokkos::parallel_for picard [type = OpenMP, device = 0]
62.124                 [SAMPLE] __GI___sched_yield [{/lib64/libc-2.31.so} {0}] <= [UNWIND] /lib64/libc-2.31.so.0 [@] hxOSThreadYield [{/proj/build/21B/Linux_x86_64/rte/nvomp/src/hxrt/.
41.212                 [SAMPLE] MPIDI_Cray_shared_mem_coll_bcast [{} {0}]
40.966                 COL_F_SOLVER_TOTAL <= COL_F_PICARD_STEP <= Kokkos::parallel_for picard [type = OpenMP, device = 0] <= COL_F_CORE_M_LOOP <= COL_F_CORE_M <= F_COL_
40.966                 COL_F_SOLVER_TOTAL
36.154
10.466 (28.948%)       [SAMPLE] UNRESOLVED /usr/lib64/libmlx5.so.1.21.37.0
35.252                 [SAMPLE] syscall [{/lib64/libc-2.31.so} {0}] <= [UNWIND] /lib64/libc-2.31.so.0 [@] hxAddressWake [{/proj/build/21B/Linux_x86_64/rte/nvomp/src/hxrt/./hxOSSuppor
22.471
33.138 (147.472%)      [SAMPLE] __GI___libc_write [{/lib64/libc-2.31.so} {0}]
```

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.

https://bit.ly/stw-humans

HPCToolkit analysis of GAMESS 5-node run on Perlmutter

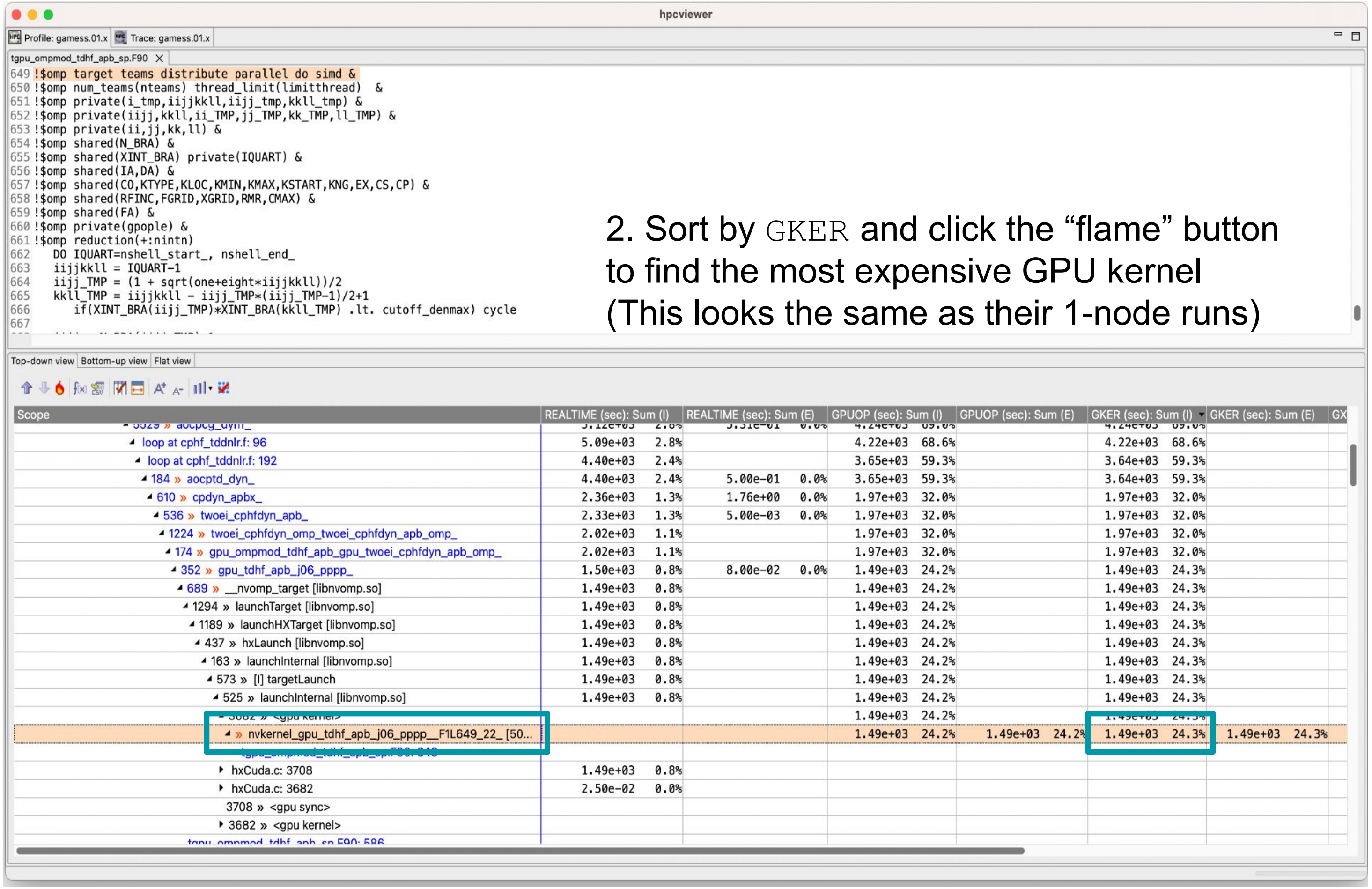


1. Look at the profile, specifically the total time spent on GPU (`GPUOP`), then compare time spent in kernels (`GKER`) vs time spent in data copies (`GXCOPY`).
This one spends most of the time in kernels (**good!**)

Scope	REALTIME (sec): Sum (I)	REALTIME (sec): Sum (E)	GPUOP (sec): Sum (I)	G PUOP (sec): Sum (E	GKER (sec): Sum (I)	GI ER (sec): Sum (E)	GXCOPY (sec): Sum (I)	XCOPY (sec): Sum (E)
Experiment Aggregate Metrics	1.82e+05 100.0%	1.82e+05 100.(%	6.15e+03 100.0%	6.15e+03 100.)%	6.14e+03 100.0%	6.14e+03 100.(%	7.66e+00 100.0%	7.66e+00 100.0%
<program root>	1.92e+04 10.5%		6.15e+03 100.0%		6.14e+03 100.0%		7.66e+00 100.0%	
<thread root>	1.63e+05 89.5%							
<partial unwind>								

https://bit.ly/stw-humans

HPCToolkit analysis of GAMESS 5-node run on Perlmutter



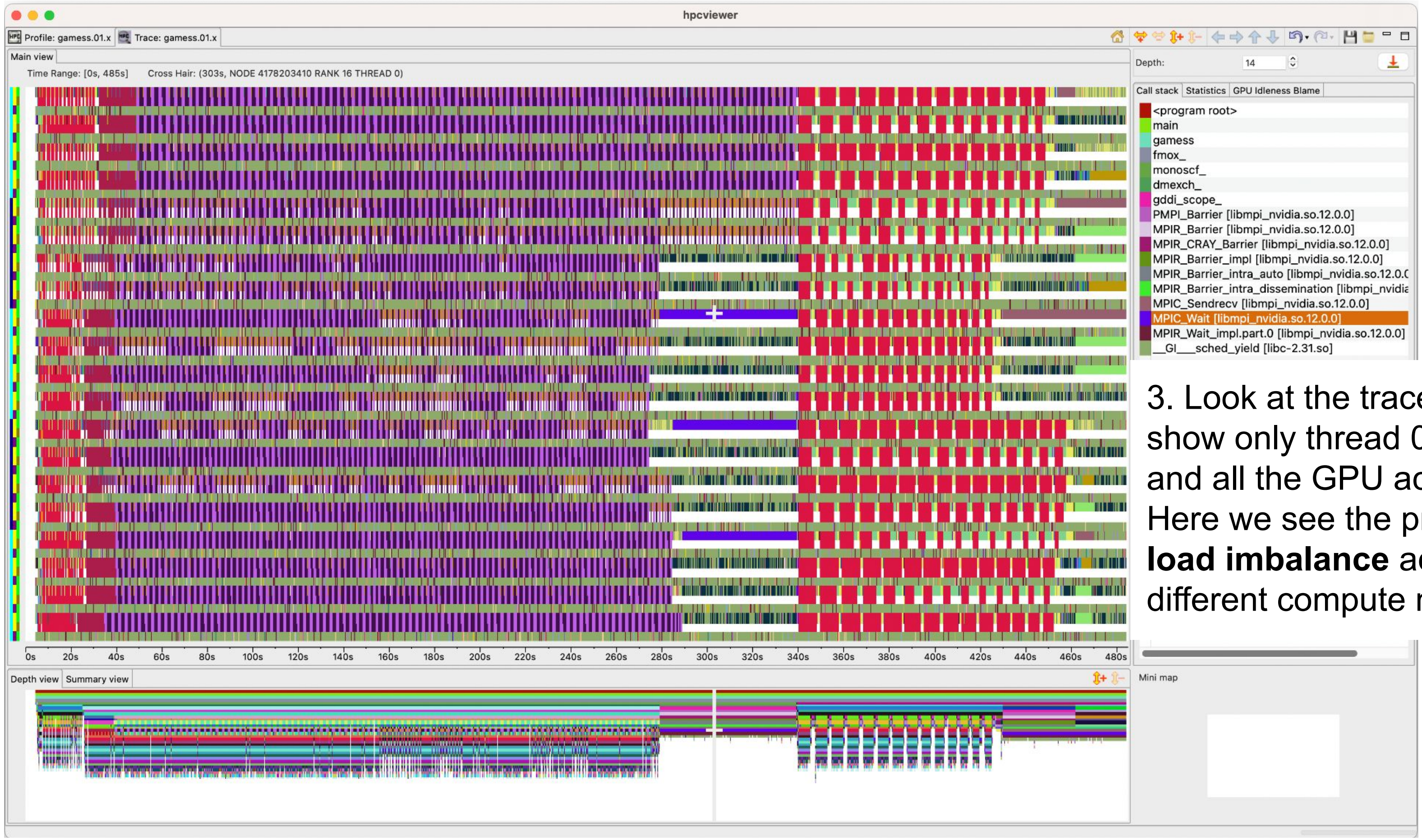
```fortran
!$omp target teams distribute parallel do simd &
!$omp num_teams(nteams) thread_limit(limitthread)  &
!$omp private(i_tmp,iijjkkll,iijj_tmp,kkll_tmp) &
!$omp private(iijj,kkll,ii_TMP,jj_TMP,kk_TMP,ll_TMP) &
!$omp private(ii,jj,kk,ll) &
!$omp shared(N_BRA) &
!$omp shared(XINT_BRA) private(IQUART) &
!$omp shared(IA,DA) &
!$omp shared(CO,KTYPE,KLOC,KMIN,KMAX,KSTART,KNG,EX,CS,CP) &
!$omp shared(RFINC,FGRID,XGRID,RMR,CMAX) &
!$omp shared(FA) &
!$omp private(gpople) &
!$omp reduction(+:nintn)
   DO IQUART=nshell_start_, nshell_end_
   iijjkkll = IQUART-1
   iijj_TMP = (1 + sqrt(one+eight*iijjkkll))/2
   kkll_TMP = iijjkkll - iijj_TMP*(iijj_TMP-1)/2+1
      if(XINT_BRA(iijj_TMP)*XINT_BRA(kkll_TMP) .lt. cutoff_denmax) cycle
```

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
loop at cphf_tddnlr.f: 96	5.09e+03 2.8%		4.22e+03 68.6%		4.22e+03 68.6%		
loop at cphf_tddnlr.f: 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_ompmod_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_j06_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 » [I] 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%		
3682 » <gpu kernel>			1.49e+03 24.2%		[illegible]		
» nvkernel_gpu_tdhf_apb_j06_pppp__F1L649_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>							

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)

https://bit.ly/stw-humans

HPCToolkit analysis of GAMESS 5-node run on Perlmutter



Time Range: [0s, 485s]   Cross Hair: (303s, NODE 4178203410 RANK 16 THREAD 0)

- <program root>
- main
- gamess
- fmox_
- monoscf_
- dmexch_
- gddi_scope_
- PMPI_Barrier [libmpi_nvidia.so.12.0.0]
- MPIR_Barrier [libmpi_nvidia.so.12.0.0]
- MPIR_CRAY_Barrier [libmpi_nvidia.so.12.0.0]
- MPIR_Barrier_impl [libmpi_nvidia.so.12.0.0]
- MPIR_Barrier_intra_auto [libmpi_nvidia.so.12.0.0
- MPIR_Barrier_intra_dissemination [libmpi_nvidia
- MPIC_Sendrecv [libmpi_nvidia.so.12.0.0]
- MPIC_Wait [libmpi_nvidia.so.12.0.0]
- MPIR_Wait_impl.part.0 [libmpi_nvidia.so.12.0.0]
- __GI___sched_yield [libc-2.31.so]

3. Look at the trace, filter to show only thread 0 on CPU and all the GPU activity. Here we see the problem is **load imbalance** across different compute nodes

https://bit.ly/stw-humans

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/HCC experts can help here**

https://bit.ly/stw-humans

Raw Notes Slides

Exercise: Who is **your** “user”? - Stage Right Group (High Level)

- 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.

What we hope to answer:

- 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"? - Stage Left

What is their expected performance analysis workflow (with or without your tool)?

- Scientific simulations application builder
- Two types (domain scientists and coders)
- PThread based performance analysis
  - Async io programming
  - With python
  - Financial Users are very closed off
- Straightforward/traditional performance analysts
- Slurm Based-Job Launchers/MPI
- TAU traces visualized in Vampir/Perfetto
- Nvidia Nsight -> only for NVIDIA GPUs
- Proprietary software does not scale on a desktop

What is their biggest frustration?

- Scalability of scaling tools
- AMD is in early stages
- They cannot even try the big runs
- HPCToolkit -> Learning curve is difficult
  - Needs to be handheld with how to analyze the data
  - It's not evident how the data relates to runtime performance issues
  - They do not have the time or resources to learn it in depth
- They don't want to change the binary
- Users don't want to be overwhelmed with unnecessary information
- Not enough visibility into the main problems
- They need to understand why their collective operations are taking so long
- Performance data needs to be interpretable in an abstractions they understand

How much time do you think they spend analyzing performance in their job? How long per session?

- Only when needed

How often do they try new tools?

- Not often at all
- There is a focus on desktop-class systems with a GPU

https://bit.ly/stw-humans

Exercise: Who is **your** “user”? - Stage Right Group (Raw)

What is their expected performance analysis workflow (with or without your tool)?

- They are scientists, less experienced in CS, physicists, chemists, etc. Others have those more used to coding.
- Many students use tools for class in CS and computational engineering.
- They would like to use tools but they don’t have time to use tools. They know it’s important but as long as things are working they don’t do things. (Guilty?) ← Might be a bit of a disconnect
- Teams have their own cultures and it’s hard to get into it unless someone from the inside wants to do something (and adds money). Some will create their own and some will go find tool developers. So there’s some Not-Invented-Here. You need humans in the teams to help adoption.
- We tell them run a profile, find hotspot, focus on them, go a level deeper with markers and annotations to actually see what’s going on. Use LIKWID or PAPI to see memory bound or compute bound… OR compare to the expected, is there space for optimization if you’re at 90% of peak… but in practice people don’t want to spend that much time. They want to run a command and get an answer.
- This leads to teaching them to do automated performance regression on the daily with Caliper or other tools. Some teams are using it.
- Tools like TotalView are for when you absoultely need it. Performance is the same thing but then there’s a learning curve. You don’t know how to use it and you don’t understand the data. So how do we build performance as a continuous process. If they see something every day that alerts them there’s a level of performance in different categories, then they can see the expected. Teams are in different shape with preparedness.
- User group is so vast – the application USERS and the application DEVELOPERS. The developers don’t necessarily run at scale but the users/scientists do. Developers are doing features. At Sandia there used to be a separate performance team.This is sometimes an ‘application readiness’ team.
    - There is some moving tools from Developers to Code Runners problem. Developers in theory can read a profile and optimize their code.The Runner is the one who runs into the performance bottlenecks.
    - It’s hard to run multiple performance runs because of the bank limitations
- We should still develop for the code developers and not as much the code users… the users could generate data but it’s hard to get them to use it. It would be nice to have their input decks for things like application parameters.

How much time do you think they spend analyzing performance in their job? How long per session?

How often do they try new tools?

- Tools are not super-intuitive. Need learning curve so it scares away
- Users may not have time for that
- Tools that are too simple and easy to use but that takes away some of their potential power. So it’s a trade off of simple results versus more power.
- Each tool has a different interfaces, and they remember having struggles and then there’s a wall to get into things. The benefit has to be really obvious.
- Having human champions to do it for them has been great. So people from the Caliper team come in and put things in.
- Social dynamics may get in the way of valuing performance expertise versus science expertise. The champions teach the scientists until they have expertise.
- Tools have help visualize in an ensemble are beautiful and so nice

What are barriers/frustrations?

- Documentation limitations – why is it hard to find info on Caliper? Is there no quickstart guide? Again a learning curve.
- What’s the Apple design philosophy? How can you make this so easy? First comparmentalize, what are our 5-6 categories of users.
    - Funding/focus tends to be on a few major applications and there’s some rickling down over that.
    - We are identifying the groups, figuring out what data they need, figuring out how to get the data, and then finding ways to build up from there. However, we have tech issues at making sue we can help people do things at scale
- We tend to focus on individual parts of the system (compilers, I/O) but there’s a need to look at the whole workflow. This can be ensembles or it could be heterogeneous. Workflow managers are starting to collect data. Monitoring is involved. Application-specific not enough.
- We have difficulty getting enough traction to have someone look at it.
- Users want to be able to turn on and off instrumentation. Once the instrumentation is there,then we can show them how to use the data.Meet with them occasionally has been somewhat successful.
- Silo’d institutions make it difficult for application users to talk to the developers. Institutional problem between application users not having the data going directly back to the developers

https://bit.ly/stw-humans

Exercise 2: How do **you** differ? - Stage Left

What is your perf analysis workflow?

- Sameer -> Run application with `tau_exec` -> look at hardware performance counters -> look at traces through vampir or other
  - This workflow is more structured than a users workflow, maybe
- Wil -> Collect CPU+GPU profile+trace w/ HPCToolkit, open database in HPCViewer
  - HPCViewer profile view is a tree+table paradigm
  - Can sort by inclusive vs exclusive metrics
  - Looking at GPU kernels before anything else
  - Looking at GPU kernels vs. GPU copies
    - To understand execution vs. transportation overhead
  - A case with 30% time on the GPU is a very very good case
    - Some apps can be as low as 1%
  - Select a metric and use the flame button to highlight a hotpath immediately
  - Look at the trace view
  - Load imbalance is a commonly identified issue

How much time do you spend analyzing performance in your job? How long per session?

What is your biggest frustration?

- Would like to have visualization that scales with levels of the runtime.
  - Google maps as a metaphor
  - No overview to zoom
  - Formats the support hierarchical views
  - There is no performance visualization with different scales
- Our data formats are legacy and do not adhere to hierarchical views “like a microscope”

How often do you try new tools?

https://bit.ly/stw-humans

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

- Is there merit in socializing users to multiple tools?
  - There is few opportunities to try new tools
  - **A reference document of some variety that guides tool developers**
    - **Provide pros and cons of the tools**
- Did we all learn performance and debugging under duress of an issue?
  - time() as the "ultimate performance tool"
    - Why is this the case: is it a usability issue, discoverability of tools issue, learability issue?
- We need to measure the workflow and integrate it better.
  - We start from assumptions that the space of performance problems are pretty limited
- Complexity of tool installation is very difficult
  - Containers, Different Configurations
  - A user space package manager system
    - Spack fills this role to a certain extent

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

- Provide clear use cases/samples alongside API Documentation
  - Sometimes these tools are hard to move out of their silos
  - There are often very clear and simple “5 steps” we do when we use a tool
  - Should documentation be more task based first?
    - Is that an easier step or too constraining
    - **Quickstart for <class of user>**
      - This makes it transferable between different users on the same team
      - Will users get too impatient for tutorials
    - Provide a template github repo alongside documentation
    - Are we different from our users in this learning workflow?
  - Are there academic users? Are they any different? Should we do anything for them?
    - Do we need to focus down our documentation to avoid struggling with meeting too many diverse needs?
-

https://bit.ly/stw-humans