ARGONNE
ATPESC2025
EXTREME-SCALE COMPUTING

Introduction to Darshan

Understanding the I/O behavior of your application

Phil Carns

carns@mcs.anl.gov

Mathematics and Computer Science Division

Argonne National Laboratory

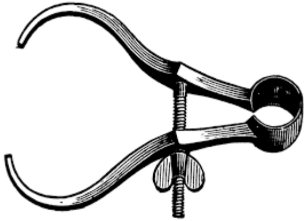
August 7, 2025

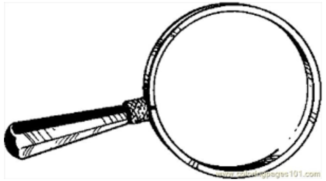
Understanding I/O problems in your application

Example questions:

- ❑ How much of your run time is spent reading and writing files?
- ❑ Does it get better, worse, or is it the same as you scale up?
- ❑ Does it get better, worse, or is it the same across platforms?
- ❑ How should you prioritize I/O tuning to get the most bang for your buck?

We recommend using a tool called **Darshan** as a starting point to answer these kinds of questions.





What is Darshan?

Darshan is a scalable HPC I/O characterization tool. It captures a concise picture of application I/O behavior with minimal overhead.

★Widely available

- Deployed at most large supercomputing sites
- Including ALCF, OLCF, and NERSC systems

★Easy to use

- No changes to code or development process
- Negligible performance impact: just “leave it on”

★Produces a *summary* of I/O activity for every job

- This is a great starting point for understanding your application’s data usage
- Includes counters, timers, histograms, etc.

How does Darshan work?

Two primary components:

1. Darshan runtime library

- Instrumentation modules: lightweight wrappers intercept application I/O calls and record per-file statistics.
  - File records are stored in bounded, compact memory on each process.
- Core library: aggregate statistics when the application exits.
  - Collect, filter, and compress records and write a single summary file.

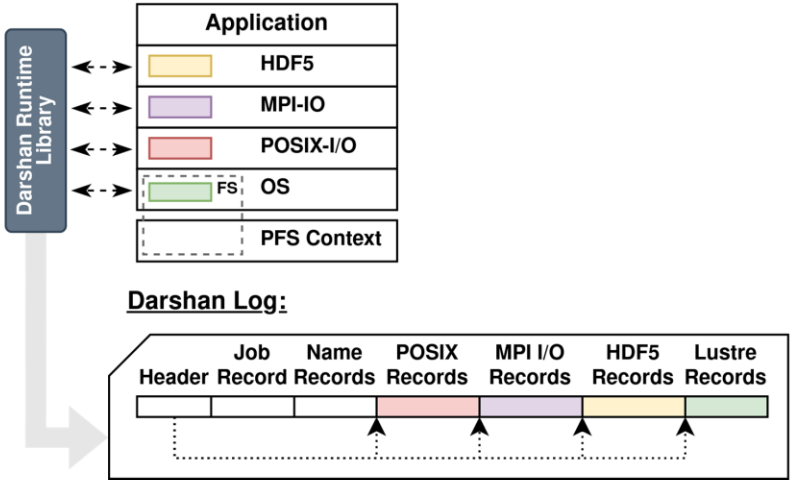


Figure courtesy Jakob Luettgau (Inria)

How does Darshan work?

Two primary components:

1. **Darshan runtime library**

NOTE: Darshan was originally designed to instrument MPI applications, but it can now be used for non-MPI applications as well.

You'll see some examples in the last session today.

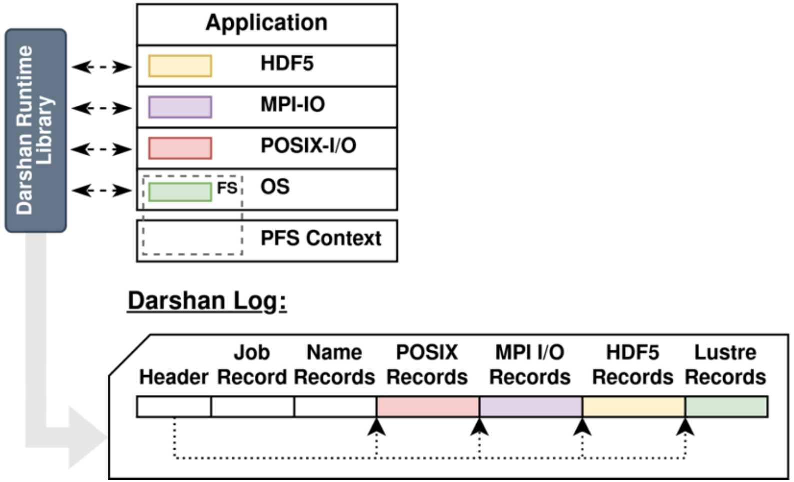


Figure courtesy Jakob Luettgau (Inria)

How does Darshan work?

Two primary components:

2. **Darshan log analysis tools**
   - Tools and interfaces to inspect and interpret log data
     - PyDarshan command line utilities
     - Python APIs for usage in custom tools, Jupyter notebooks, etc.
     - Legacy C-based tools/library

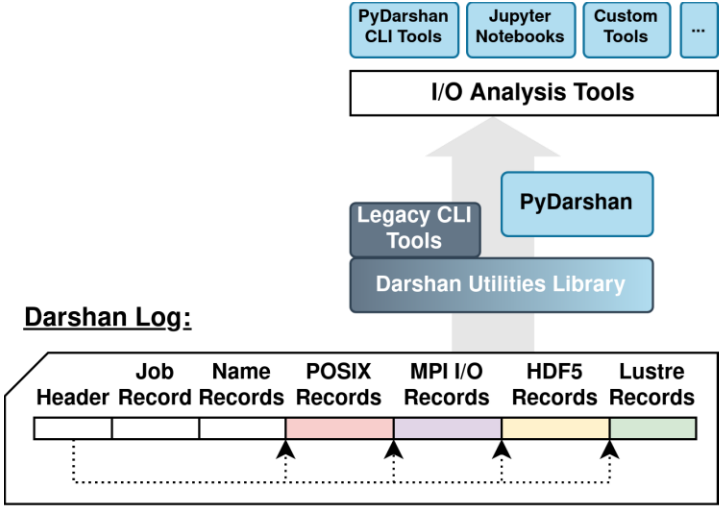


Figure courtesy Jakob Luettgau (Inria)

Darshan hands on exercise

- We’ll start by collectively working through a hands on exercise demonstrating how to use the Darshan toolchain on Aurora.
- Usage on other systems is very similar, though. The most likely differences are:
  - Location of log files (where to find data after your job completes)
  - Analysis utility availability (usually easiest to just copy logs to your workstation to analyze)
  - Loading the Darshan module (if it’s not already there by default)
- We’ll briefly cover differences on other DOE systems after this Aurora example.

One caveat: we will not be using the default Darshan installation on Aurora today. We are using a special configuration for ATPESC, but the default installation will be updated to match it soon!

Darshan hands on exercise: build/instrument the helloworld example

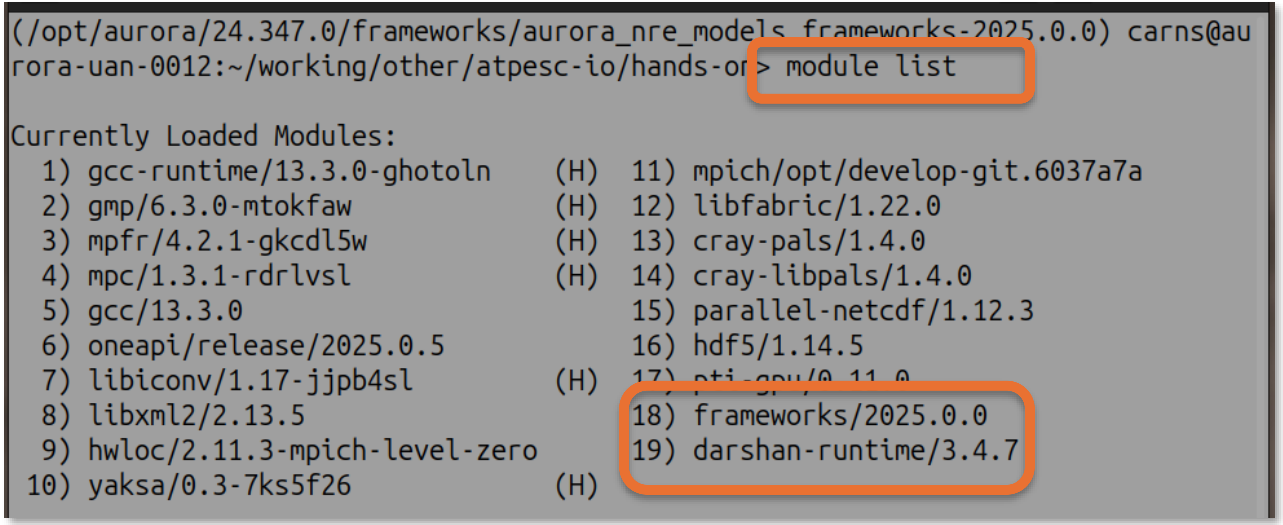
- Log onto **aurora.alcf.anl.gov**
- Setup a working directory and checkout the **hands-on** repo.

```
mkdir atpesc-io
cd atpesc-io
git clone https://github.com/radix-io/hands-on.git
cd hands-on
```

- Load the Darshan configuration for ATPESC 2025.

```
source ./aurora-setup-env.sh
```

Darshan hands on exercise:
make sure the software is loaded



```
(/opt/aurora/24.347.0/frameworks/aurora_nre_models_frameworks-2025.0.0) carns@au
rora-uan-0012:~/working/other/atpesc-io/hands-on> module list

Currently Loaded Modules:
  1) gcc-runtime/13.3.0-ghotoln     (H)  11) mpich/opt/develop-git.6037a7a
  2) gmp/6.3.0-mtokfaw              (H)  12) libfabric/1.22.0
  3) mpfr/4.2.1-gkcdl5w             (H)  13) cray-pals/1.4.0
  4) mpc/1.3.1-rdrlvsl              (H)  14) cray-libpals/1.4.0
  5) gcc/13.3.0                          15) parallel-netcdf/1.12.3
  6) oneapi/release/2025.0.5             16) hdf5/1.14.5
  7) libiconv/1.17-jjpb4sl          (H)  17) pti-gpu/0.11.0
  8) libxml2/2.13.5                      18) frameworks/2025.0.0
  9) hwloc/2.11.3-mpich-level-zero       19) darshan-runtime/3.4.7
 10) yaksa/0.3-7ks5f26              (H)
```

Use "**`module list`**" to view your environment.

We want to confirm that two modules are loaded:

- Darshan 3.4.7 (the latest version of Darshan)
- Frameworks (an Intel Python environment that we will use in our analysis tools).

Darshan hands on exercise: build/instrument the helloworld example

- Move to the **darshan/helloworld** example directory and build the example code. (TIP: note that Aurora uses “mpicc” rather than “cc” to build MPI programs)

```
cd darshan/helloworld
mpicc -o helloworld helloworld.c
```

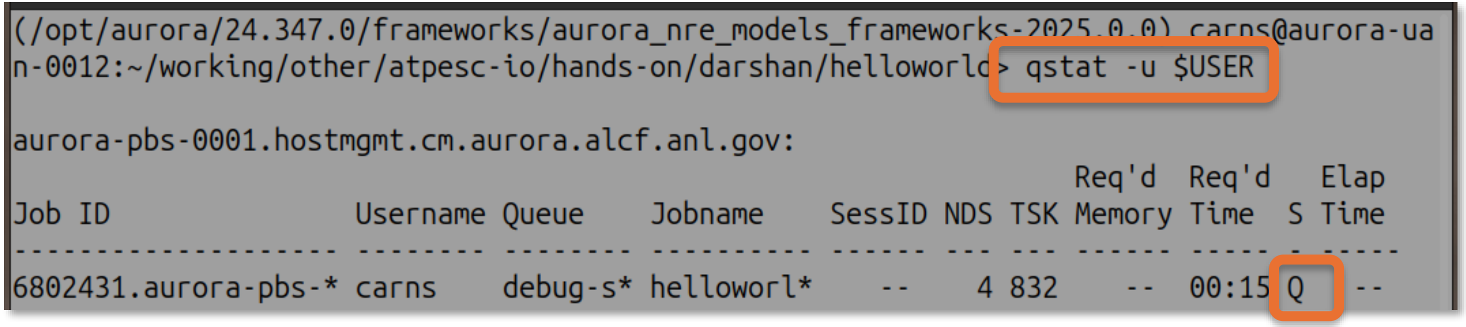
- No other steps are needed to get Darshan instrumentation
- If you have the darshan-runtime module loaded, then it will automatically instrument any MPI applications written in C, C++, or Fortran.
- Environment variables are needed for Python (usually) and for non-MPI applications; we’ll cover that later.

Darshan hands on exercise: run the job

○ Submit the **helloworld** job script to the scheduler.

```
qsub helloworld-aurora.qsub
```

○ Use the **`qstat -u <username>`** tool to help determine when your job is complete. (If there is no qstat output, then there are no queued/running jobs).



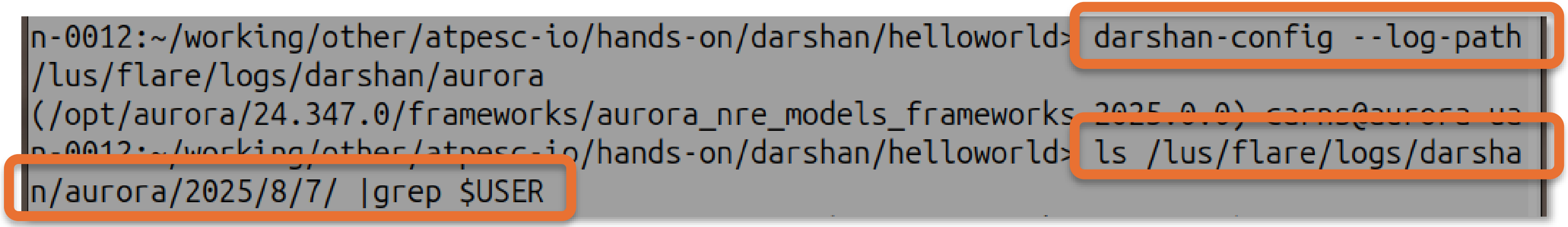
```
(/opt/aurora/24.347.0/frameworks/aurora_nre_models_frameworks-2025.0.0) carns@aurora-ua
n-0012:~/working/other/atpesc-io/hands-on/darshan/helloworld> qstat -u $USER

aurora-pbs-0001.hostmgmt.cm.aurora.alcf.anl.gov:
                                                                 Req'd  Req'd    Elap
Job ID               Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
-------------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
6802431.aurora-pbs-* carns    debug-s* helloworl*     --   4 832     -- 00:15 Q   --
```

Darshan hands on exercise: find your log file

Once your job has completed (it is no longer listed in qstat, and your directory has helloworld-aurora.qsub.e* and helloworld-aurora.qsub.o* files), we need to find the Darshan log.

All Darshan logs are placed in a central location. The '**darshan-config --log-path**' command will show the log directory location.



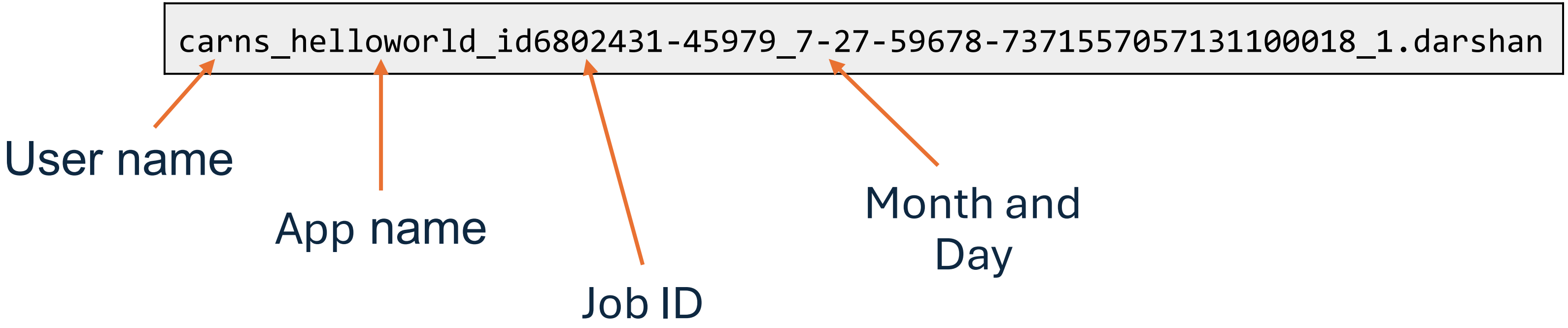
```
n-0012:~/working/other/atpesc-io/hands-on/darshan/helloworld> darshan-config --log-path
/lus/flare/logs/darshan/aurora
(/opt/aurora/24.347.0/frameworks/aurora_nre_models_frameworks 2025.0.0) carns@aurora-ua
n-0012:~/working/other/atpesc-io/hands-on/darshan/helloworld> ls /lus/flare/logs/darsha
n/aurora/2025/8/7/ |grep $USER
```

Check the subdirectory for the **year/month/day** your job executed. During this presentation, your logs should be landing in **/lus/flare/logs/darshan/aurora/2025/8/7/**.

Be aware of the time zone (or just check adjacent days)!
Aurora uses the UTC time zone and will roll over to the next day at 7pm local time.

Darshan hands on exercise:
Anatomy of a Darshan log file name

You should have found a file that looks like this in the Darshan log directory:



```
carns_helloworld_id6802431-45979_7-27-59678-737155705713110001 8_1.darshan
```

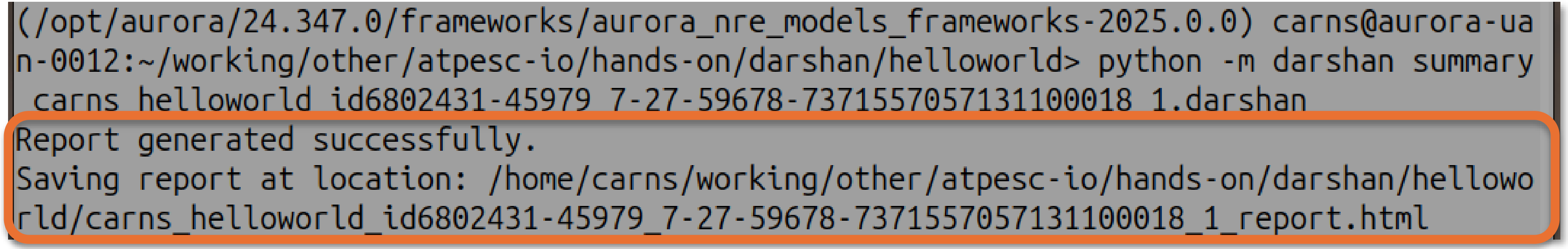
Darshan files are in a binary format, but a variety of tools are available to analyze them. Copy the file to your hands-on/darshan/helloworld directory:

```
cp /lus/flare/logs/darshan/aurora/2025/8/7/${USER}*.darshan .
```

Darshan hands on exercise: Generate a summary report

The environment setup script that we executed earlier gives you access to PyDarshan analysis tools.

Generate an HTML summary report with PyDarshan using the following command: ‘**`python -m darshan summary <log_file>`**’.



```
(/opt/aurora/24.347.0/frameworks/aurora_nre_models_frameworks-2025.0.0) carns@aurora-ua
n-0012:~/working/other/atpesc-io/hands-on/darshan/helloworld> python -m darshan summary
 carns helloworld id6802431-45979 7-27-59678-7371557057131100018 1.darshan
Report generated successfully.
Saving report at location: /home/carns/working/other/atpesc-io/hands-on/darshan/hellowo
rld/carns_helloworld_id6802431-45979_7-27-59678-7371557057131100018_1_report.html
```

It will generate an HTML report matching the input log file name.

Darshan hands on exercise: analyze job summary report in a browser

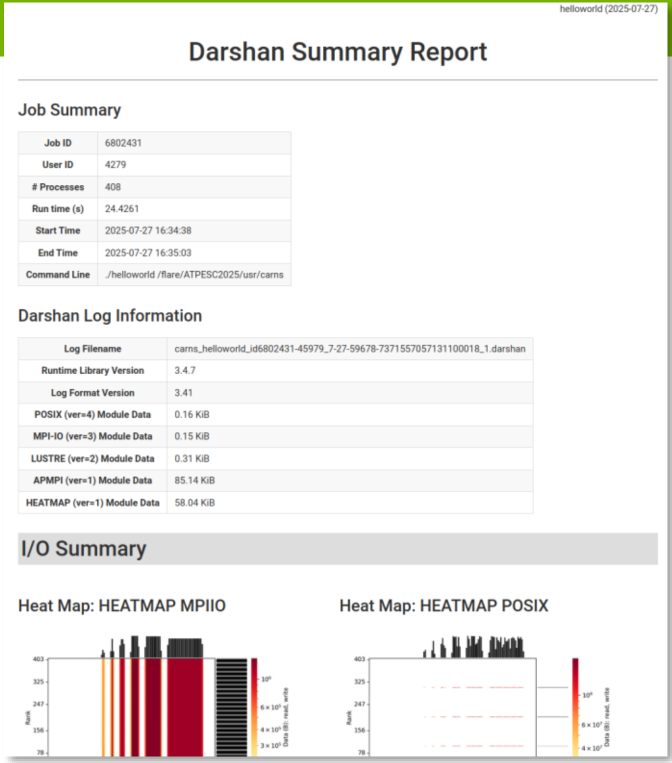
- ○ First, use **scp** to copy the log to your own personal system.

```
scp aurora.alcf.anl.gov:/path/to/report.html \
    /local/path/to/report.html
```

- ○ Next, open up the HTML report with your browser of choice and scan through the information it provides.

We will pause here to give everyone some time to catch up before moving onto interpreting the job summary report results. **Reach out to an instructor if you need help!**

Job summary report example



helloworld (2025-07-27)

Darshan Summary Report

Job Summary

Job ID	6802431
User ID	4279
# Processes	408
Run time (s)	24.4261
Start Time	2025-07-27 16:34:38
End Time	2025-07-27 16:35:03
Command Line	./helloworld /flare/ATPESC2025/usr/carns

Darshan Log Information

Log Filename	carns_helloworld_id6802431-45979_7-27-59678-7371557057131100018_1.darshan
Runtime Library Version	3.4.7
Log Format Version	3.41
POSIX (ver=4) Module Data	0.16 KiB
MPI-IO (ver=3) Module Data	0.15 KiB
LUSTRE (ver=2) Module Data	0.31 KiB
APMPI (ver=1) Module Data	85.14 KiB
HEATMAP (ver=1) Module Data	58.04 KiB

I/O Summary

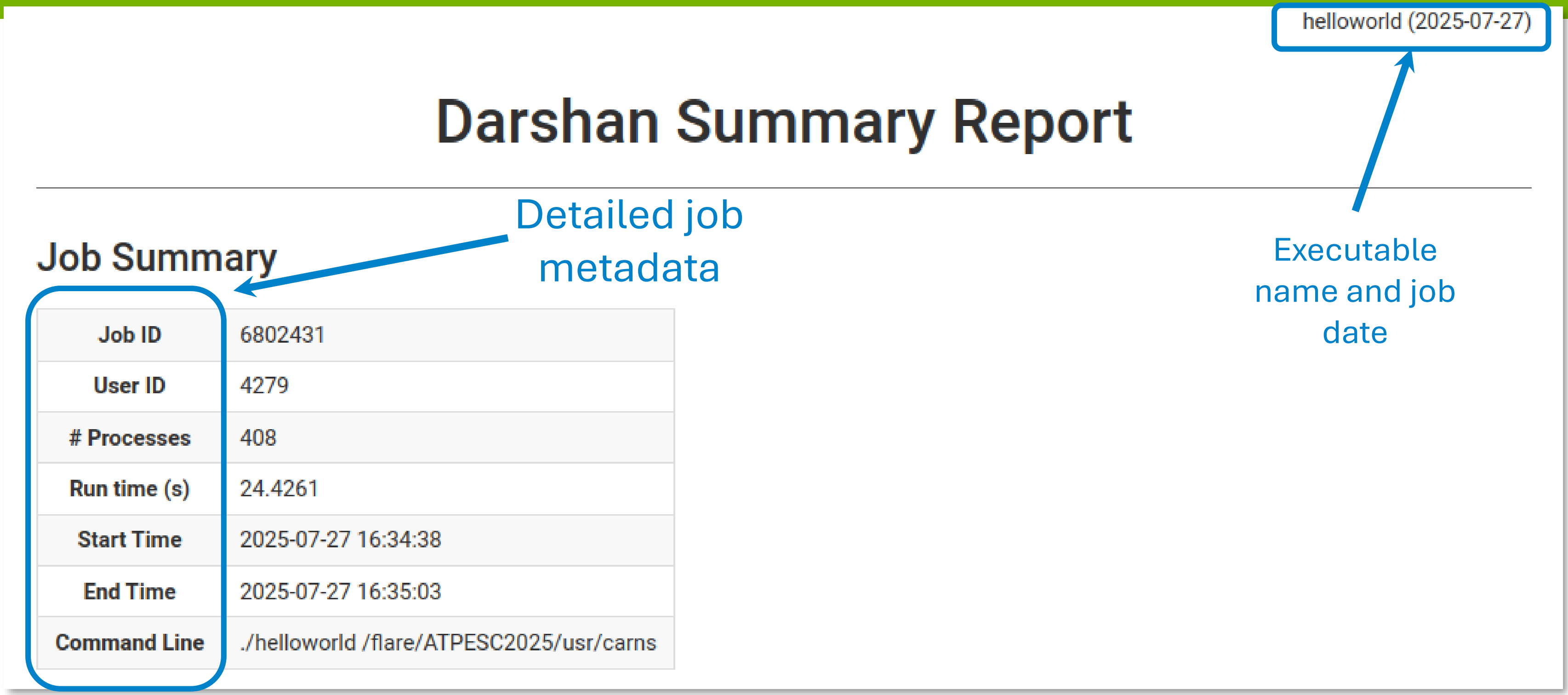
Heat Map: HEATMAP MPIIO

Heat Map: HEATMAP POSIX

The PyDarshan job summary tool generates an HTML report with various graphs and tables the I/O workload of the application.

We will walk through some highlights from the helloworld example in the following slides.

Job summary report: high-level job info



helloworld (2025-07-27)

Darshan Summary Report

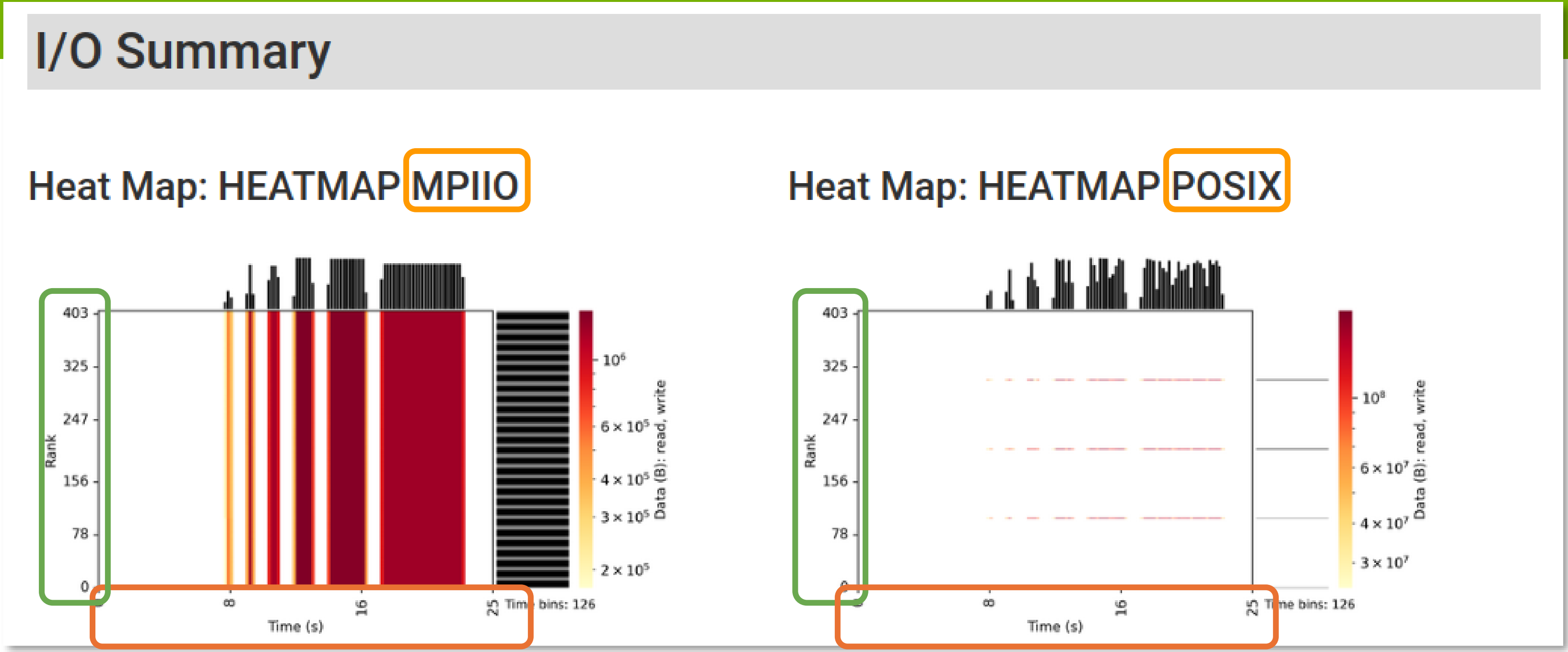
Job Summary

Job ID	6802431
User ID	4279
# Processes	408
Run time (s)	24.4261
Start Time	2025-07-27 16:34:38
End Time	2025-07-27 16:35:03
Command Line	./helloworld /flare/ATPESC2025/usr/carns

Detailed job metadata

Executable name and job date

Job summary report: I/O heatmaps



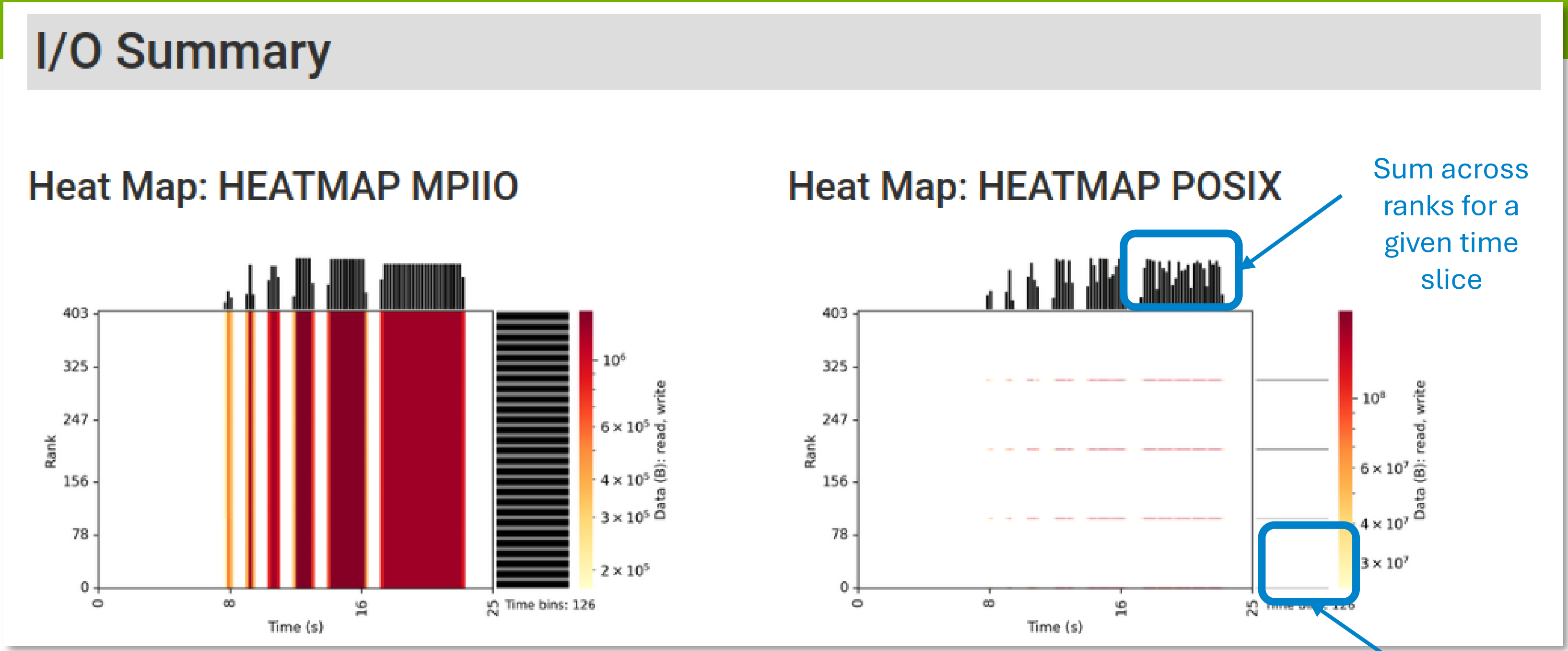
I/O Summary

Heat Map: HEATMAP MPIIO

Heat Map: HEATMAP POSIX

Heatmaps showcase application I/O intensity (read+write volume) across **time**, **ranks**, and **interfaces** – helpful for identifying hot spots, I/O and compute phases, etc.

Job summary report: I/O heatmaps

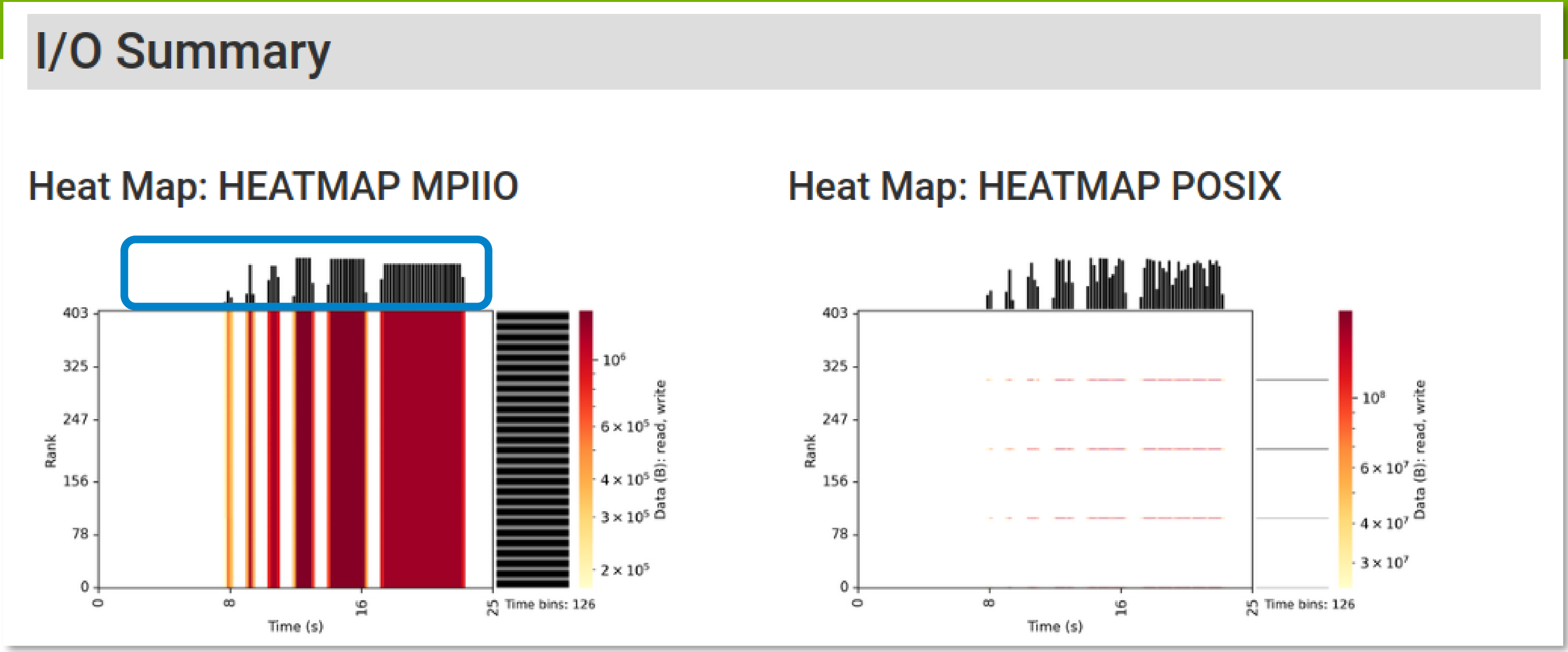


Sum across ranks for a given time slice

Sum across time for a given rank

Heatmaps showcase application I/O intensity (read+write volume) across time, ranks, and interfaces – helpful for identifying hot spots, I/O and compute phases, etc.

Job summary report: I/O heatmaps



I/O Summary

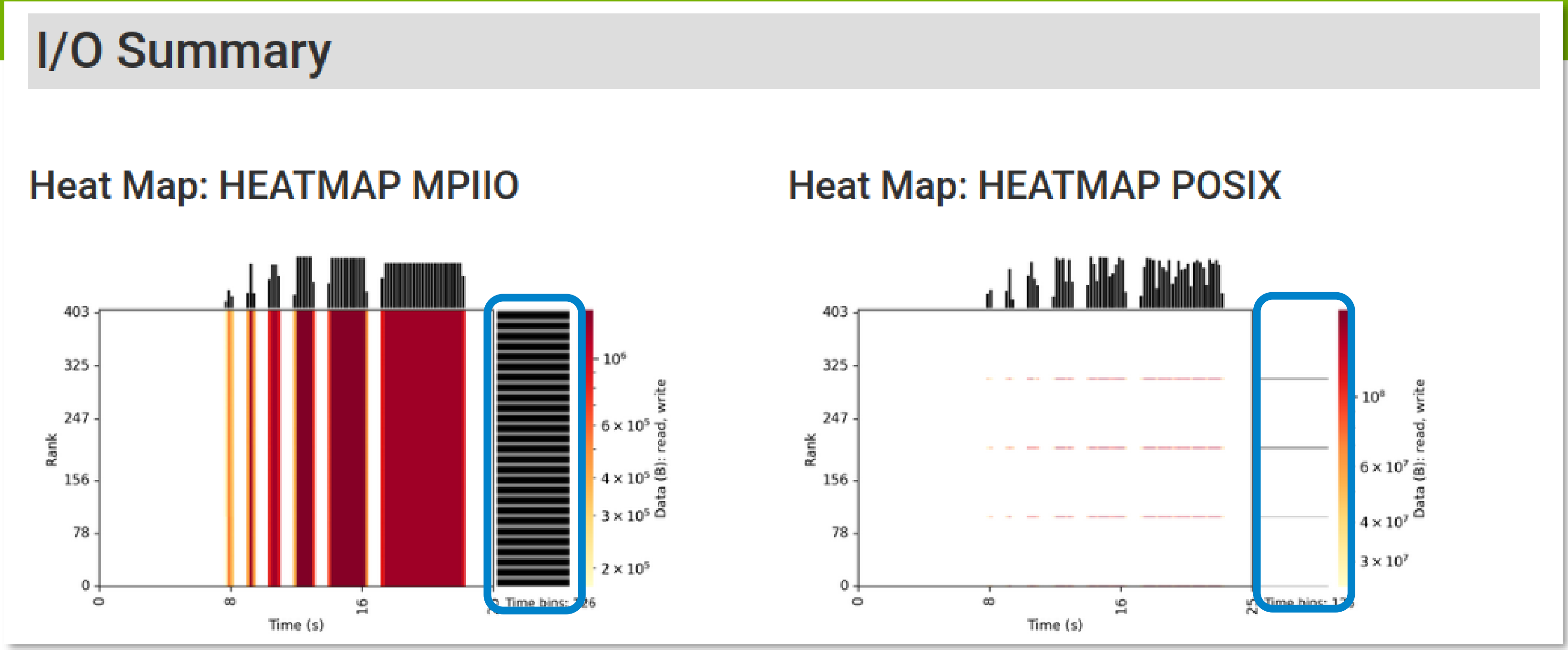
Heat Map: HEATMAP MPIIO

Heat Map: HEATMAP POSIX

This application exhibits some notable I/O characteristics:

Application I/O phases increase in I/O intensity over time

Job summary report: I/O heatmaps



I/O Summary

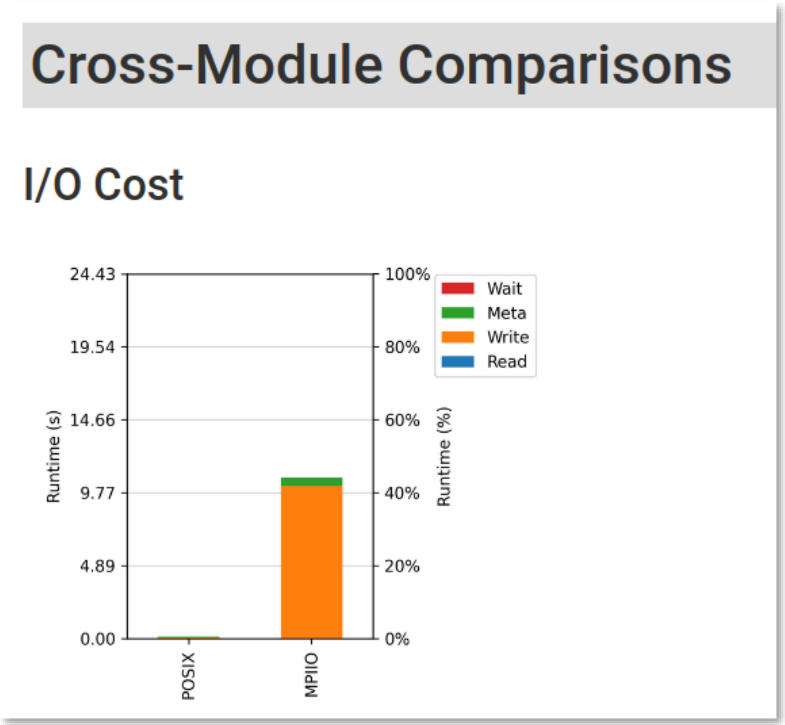
Heat Map: HEATMAP MPIIO

Heat Map: HEATMAP POSIX

This application exhibits some notable I/O characteristics:

Balanced, collective MPI-IO accesses transformed to a subset of POSIX “aggregators”

Job summary report: I/O cost



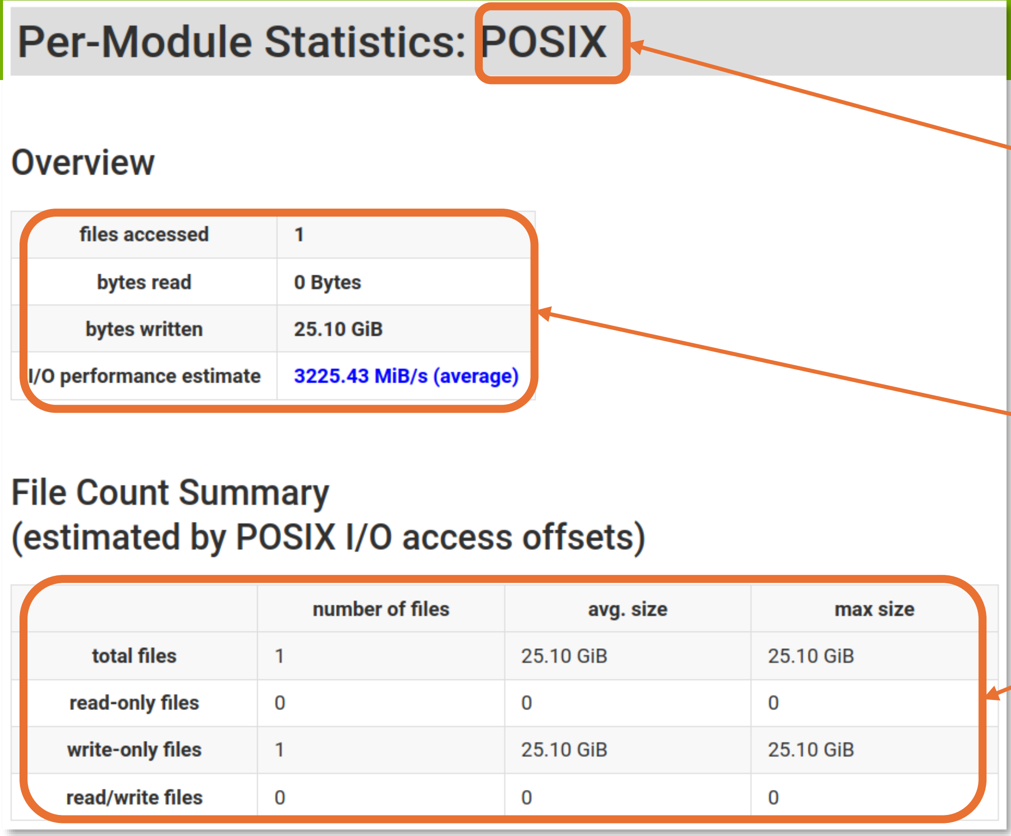
Cross-Module Comparisons

I/O Cost

I/O cost indicates how much time on average was spent in reading, writing, and metadata across different I/O interfaces.

If I/O cost is a small portion of application runtime, then tuning I/O isn’t likely to have a big impact.

Job summary report: Per-interface statistics



Per-Module Statistics: POSIX

Overview

files accessed	1
bytes read	0 Bytes
bytes written	25.10 GiB
I/O performance estimate	3225.43 MiB/s (average)

File Count Summary (estimated by POSIX I/O access offsets)

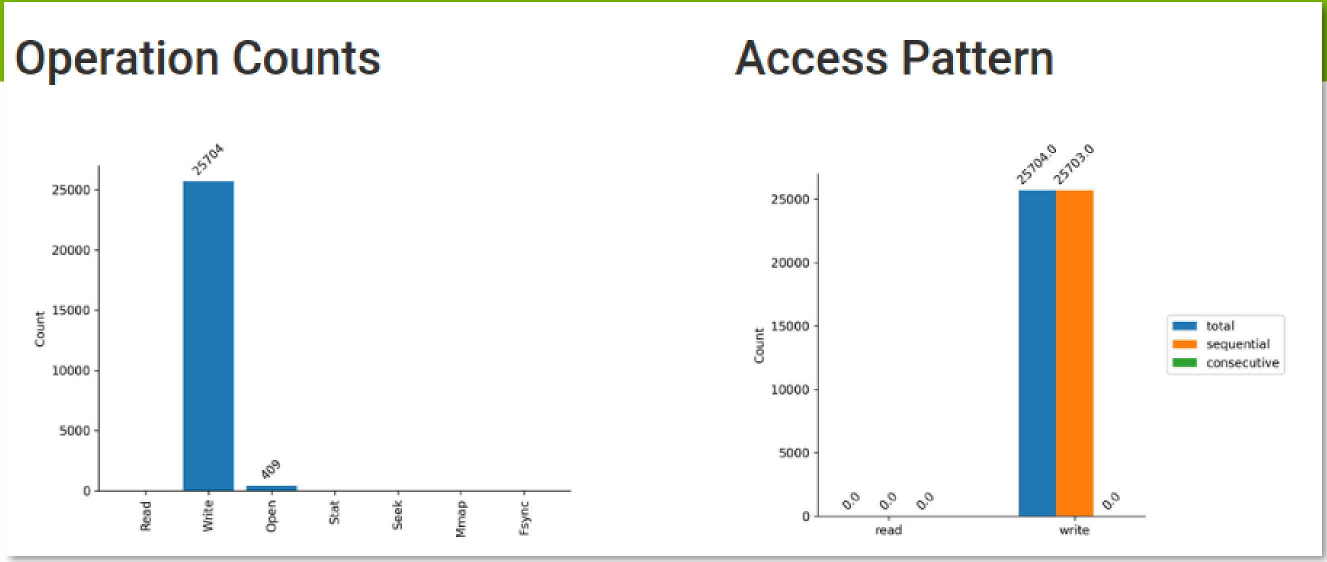
	number of files	avg. size	max size
total files	1	25.10 GiB	25.10 GiB
read-only files	0	0	0
write-only files	1	25.10 GiB	25.10 GiB
read/write files	0	0	0

Stats available for various I/O APIs: POSIX, MPI-IO, STDIO, HDF5, PnetCDF

Aggregate stats for the interface, as well as a **performance estimate**

Number of files of different types (total, read-only, read/write, etc.) that were accessed

Job summary report: Per-interface statistics



Operation Counts

Access Pattern

sequential: greater than previous offset
consecutive: immediately following previous offset

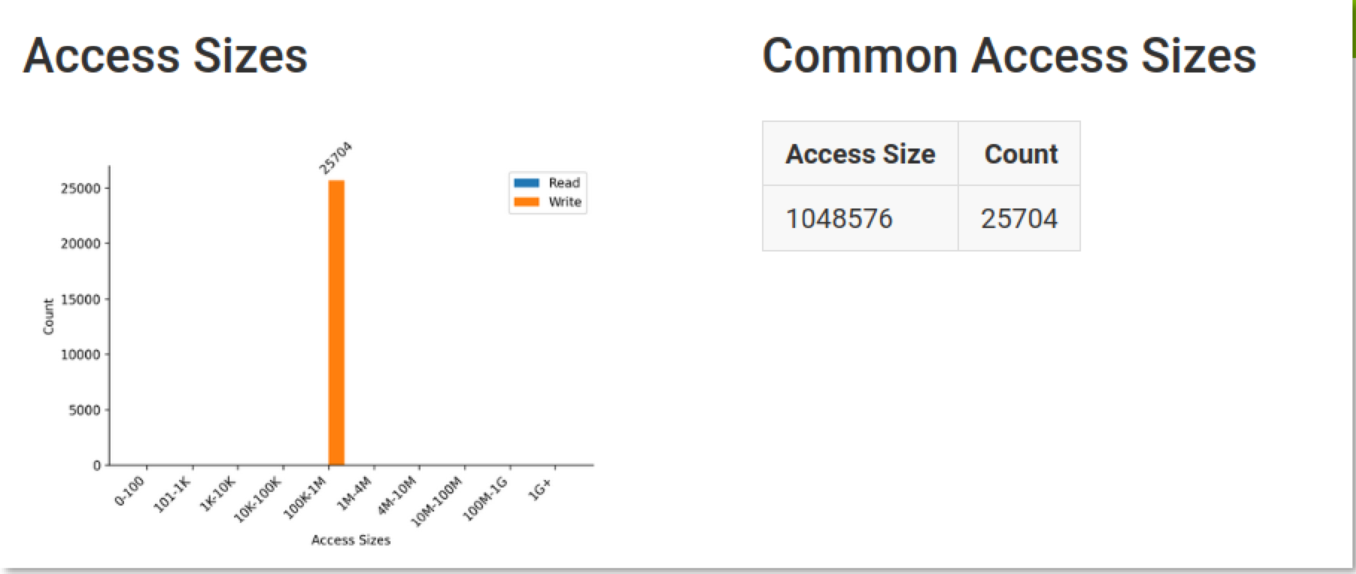
Operation counts provide the relative totals of different types of I/O operations.

Lots of metadata operations (open, stat, seek, etc.) could be a sign of poorly performing I/O.

Access pattern indicates if I/O was sequential or consecutive.

More random access patterns can be expensive for some types of storage.

Job summary report: Per-interface statistics



Access Sizes

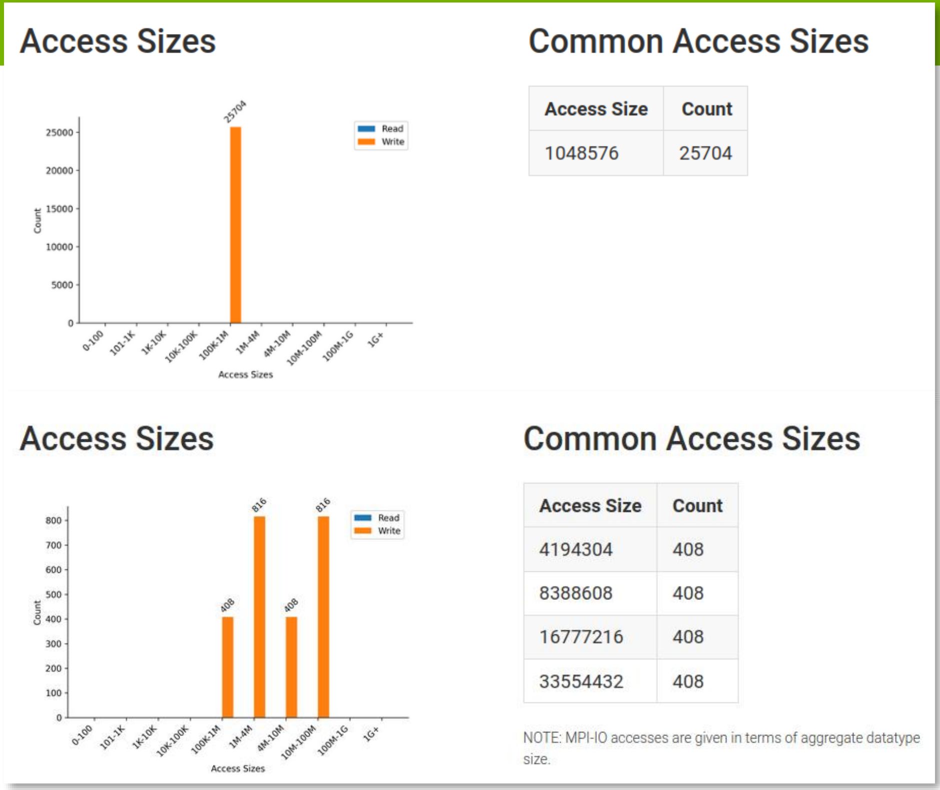
Common Access Sizes

Access Size	Count
1048576	25704

Details on access sizes are provided to better understand granularity of application read/write accesses.

In general, larger access sizes (e.g., MiBs) perform better with most storage systems.

Job summary report: Per-interface statistics



POSIX

Access Sizes

Common Access Sizes

Access Size	Count
1048576	25704

MPI-IO

Access Sizes

Common Access Sizes

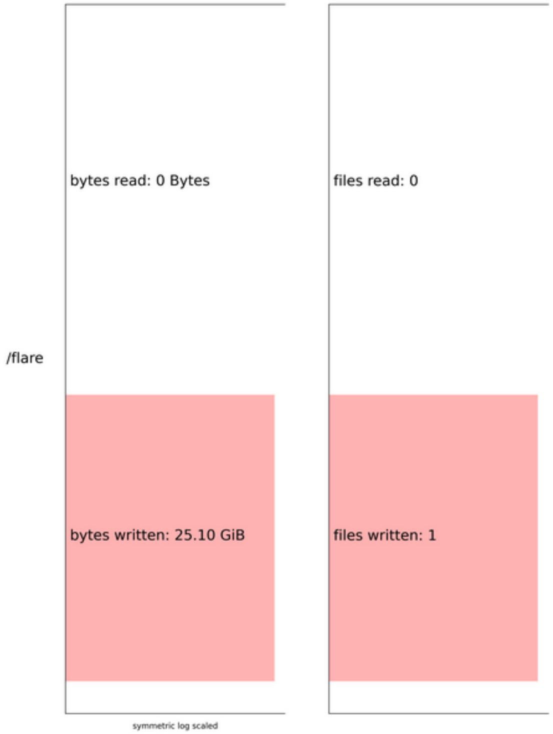
Access Size	Count
4194304	408
8388608	408
16777216	408
33554432	408

NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

Note that the file access pattern issued by the application (i.e., using MPI-IO) can vary drastically from what is ultimately issued to the file system (i.e., using POSIX).

This application increases its access size each I/O phase, yet only 1 MiB operations are ever issued to the file system via POSIX calls.

Job summary report: Data access by category

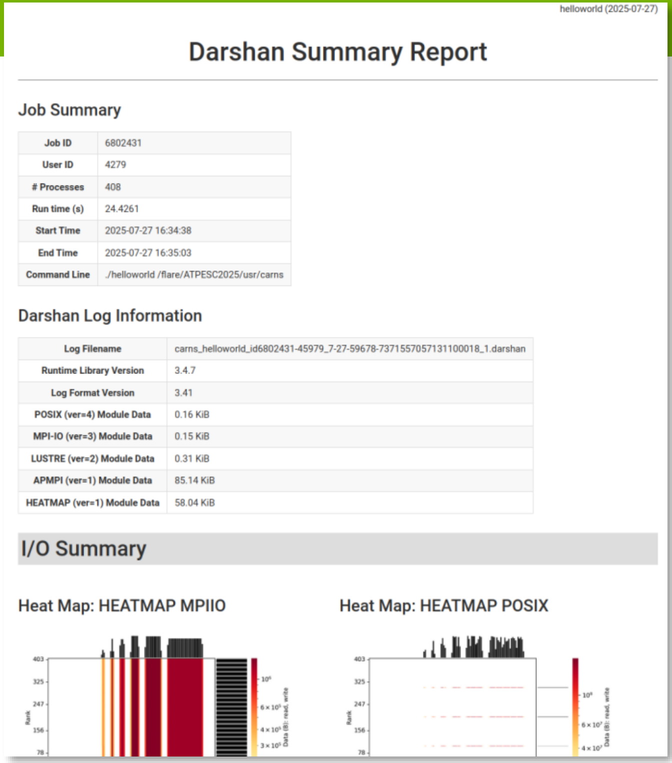


Data accesses, in terms of total files read/written and total bytes read/written, are binned into categories:

- mount points (e.g., `/home`, `/scratch`)
- standard streams (e.g., STDOUT)
- object storage pools
- etc.

Did you use the best storage system for your use case?

Job summary report: additional help



helloworld (2025-07-27)

Darshan Summary Report

Job Summary

Job ID	6802431
User ID	4279
# Processes	408
Run time (s)	24.4261
Start Time	2025-07-27 16:34:38
End Time	2025-07-27 16:35:03
Command Line	./helloworld /flare/ATPESC2025/usr/carns

Darshan Log Information

Log Filename	carns_helloworld_id6802431-45979_7-27-59678-7371557057131100018_1.darshan
Runtime Library Version	3.4.7
Log Format Version	3.41
POSIX (ver=4) Module Data	0.16 KiB
MPI-IO (ver=3) Module Data	0.15 KiB
LUSTRE (ver=2) Module Data	0.31 KiB
APMPI (ver=1) Module Data	85.14 KiB
HEATMAP (ver=1) Module Data	58.04 KiB

I/O Summary

Heat Map: HEATMAP MPIIO

Heat Map: HEATMAP POSIX

Remember to contact your site's support team for help! The Darshan job summary can be a good discussion starter if you aren't sure how to proceed with performance tuning or problem solving.

What about using Darshan on other DOE systems?

Polaris (ALCF), Perlmutter (NERSC), Frontier (OLCF):

- Automatically enabled when you log in
- Use ` darshan-config –log-path` to find the log directory.

Aurora (ALCF):

- Darshan will eventually be enabled automatically on Aurora as well, but for now use the scripts provided in ATPESC and check the ALCF documenation for udpates.

On most systems, the easiest way to analyze logs is to copy them to your own workstation and install PyDarshan.

If not available on a system, Darshan can either be installed via Spack or from source. It is provided as two separate packages in Spack:

- **darshan-runtime** - library for instrumenting apps
- **darshan-util** - tools for analyzing Darshan log files

Note that admin privileges are **not** required for installing/using Darshan tools on a system.

PyDarshan is available on PyPI (e.g., ‘**`pip install darshan`**’) and also in Spack.

See our website for more details: https://www.mcs.anl.gov/research/projects/darshan/

What about Python and non-MPI applications?

The short story: set the following two environment variables at runtime and then run your application. Limit the scope of when you enable these environment variables so that you don’t inadvertently instrument random command line tools like “ls”.

```
export LD_PRELOAD="$DARSHAN_RUNTIME_ROOT/lib/libdarshan.so:$LD_PRELOAD"
export DARSHAN_ENABLE_NONMPI=1
```

The long story: For some Python frameworks, you may also need to look for (and recover) intermediate Darshan logs from /tmp. You may also need to set Darshan options to constrain the types of files/directories that it instruments. More information can be found here:

https://www.mcs.anl.gov/research/projects/darshan/2025/03/11/using-darshan-with-non-mpi-applications-e-g-ai-ml-frameworks/

Darshan: a recap

- These slides covered some basic Darshan usage and tips.
- Refer to facility documentation, support channels, or these slides when you need to.
- Key takeaways:
  - Tools are available to help you understand how your application accesses data.
  - The simplest starting point is Darshan.
  - It’s likely already instrumenting your application, or can quickly be made to do so.
  - You will probably start with an HTML report generated using PyDarshan.
- We’ll learn about more advanced tools and use cases this afternoon.

More Darshan hands on exercises

○ The hands-on repo contains additional Darshan examples (**warpdrive** and **fidgetspinner**) that you can try as time permits:

  – Each example has A and B versions that you can compare to spot the performance differences (and their cause).

  – These examples will be easier to understand after seeing this afternoon’s MPI-IO presentation.

○ We encourage you to try these exercises out and to check with the instructors to share what you find!

○ **We also encourage you to try Darshan with your own applications to see what sorts of insights it can provide!**

Thank you!

ARGONNE
ATPESC2025
EXTREME-SCALE COMPUTING

ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.

The U.S. Government retains for itself and others acting on its behalf a nonexclusive, royalty-free license in this video, with the rights to reproduce, to prepare derivative works, and to display publicly.