7 AUGUST 2026

ATPESC 2026
DATA AND I/O

U.S. DEPARTMENT of ENERGY

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY

MEET YOUR LECTURERS

Rob Latham is a research software developer at ANL who strives to make applications use I/O more efficiently. He has played a prominent role in the ROMIO MPI-IO implementation, the PVFS file system, and the PnetCDF high level I/O library.

Amal Gueroudji is an Assistant Computer Scientist at Argonne National Laboratory, specializing in data management, distributed HPC workflows, and scientific provenance. She is the lead developer of DEISA and has played a prominent role in extending Flowcept with provenance support for agentic workflows built with Academy, LangGraph, CrewAI, and AutoGen.

Scot Breitenfeld is the director of engineering at the HDF Group. His technical specialty is HPC application use of HDF5. He has implemented and tuned HDF5 for HPC applications and third-party libraries across a wide range of platforms.

Hands on exercises:
https://github.com/radix-io/hands-on

PLAN FOR THE HALF DAY

- HPC storage
- Tools and libraries to effectively use that storage
  - Darshan
  - MPI-IO
  - I/O libraries (Parallel-NetCDF, HDF5)
- I/O and Workflows

U.S. DEPARTMENT of ENERGY
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

Hands on exercises:
https://github.com/radix-io/hands-on

Argonne NATIONAL LABORATORY

LOGISTICS FOR ATPESC-IO

Agenda:

https://extremecomputingtraining.anl.gov/2026-agenda

Discussion and questions:

- Please ask questions as we go!
- At least one of us will monitor the #atpesc-2026-track-9-io slack channel at all times.
- We can provide one-on-one help and relay questions to lecturers if needed.
- We will be available on Slack for the remainder of the ATPESC program.

Hands on exercises:
https://github.com/radix-io/hands-on

HANDS-ON EXERCISES

Hands-on exercises and machine reservations:

See https://github.com/radix-io/hands-on

How to use the hands-on exercises:

- Some exercises will be covered in real time during today's lectures.
- Additional exercises are available for you to explore at your own pace according to your interests.
- Please work on the ones that are the most important to you and ask questions as you go!

We are also happy for you to experiment with I/O strategies in your own codes and "ask the experts" for help.

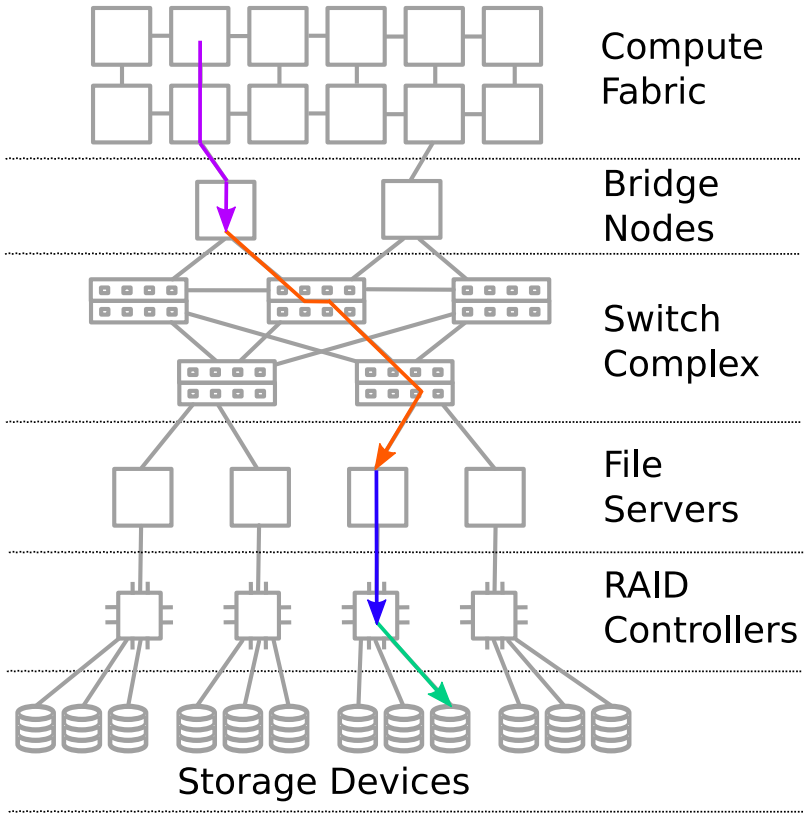
ATPESC attendees have dedicated reservations on Aurora (ALCF) today for experiments and exercises, but you are welcome to use and ask questions about any of the ATPESC systems.

HPC STORAGE

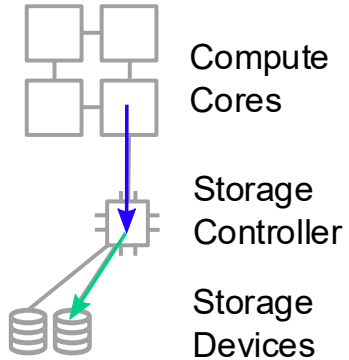
U.S. DEPARTMENT of ENERGY
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.
Argonne NATIONAL LABORATORY

A LOOK UNDER THE HOOD

HPC system storage path



Workstation (laptop) storage path



- A typical workstation/laptop has only one storage device.
- The path between applications and storage is *short*.
- Properties:
  - Low latency
  - Low bandwidth

- In contrast, an HPC storage system manages many (e.g., **thousands** of) disaggregated devices.
- Paths between applications and storage devices are quite long, but numerous.
- Properties:
  - High latency
  - High bandwidth

Hands on exercises:
https://github.com/radix-io/hands-on

https://docs.olcf.ornl.gov/data/index.html#orion-lustre-hpe-clusterstor-filesystem
https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#nvme

HPC STORAGE SYSTEMS

Oak Ridge (OLCF) Lawrence Berkeley (NERSC) Argonne (ALCF)

Facility documentation is your friend! Some of the systems that we discuss will change over time.

- Orion parallel file system (Lustre)
  - Multiple storage tiers are transparently combined:
    - Metadata tier: ~10 PiB capacity
    - Performance tier: ~11 PiB capacity, ~10 TiB/s performance
    - Capacity tier: ~680 PiB capacity, ~5 TiB/s performance
  - File layouts: Orion uses a sophisticated progressive file layout. Unlike some Lustre file systems, it is **not** recommended for you to manually change file stripe settings on Orion!

- Node-local SSDs on Frontier
  - NVMe drives provide 5 GiB/s read and 2 GiB/s write per node.
  - Excellent for latency, but they are not shared between nodes or accessible outside of your job.
  - You must explicitly request them in your job script.
  - You must explicitly stage data on and off of them.

Hands on exercises:
https://github.com/radix-io/hands-on

https://docs.nersc.gov/filesystems/
https://docs.nersc.gov/filesystems/perlmutter-scratch/

HPC STORAGE SYSTEMS

Oak Ridge (OLCF) **Lawrence Berkeley (NERSC)** Argonne (ALCF)

- Perlmutter Scratch parallel file system (Lustre)
  - Unlike the Orion file system at OLCF, the Perlmutter scratch file system consists of a single all-flash (high-performance) tier of storage.
  - 35 PiB capacity, 5 TiB/s throughput
  - File layouts: 1 server per file by default
    - This layout is great for small files accessed by single processes, but you'll want to increase the striping if you need to access large files in parallel.

https://docs.alcf.anl.gov/data-management/filesystem-and-storage/
https://docs.alcf.anl.gov/aurora/data-management/daos/daos-overview/

HPC STORAGE SYSTEMS

Oak Ridge (OLCF) Lawrence Berkeley (NERSC) **Argonne (ALCF)**

- Flare and Eagle parallel file systems (Lustre)
  - Similar caveats as Polaris Scratch: you may need to tune the Lustre settings for your use case; the default settings are only optimal for small, non-shared files.
- Polaris also has local NVMe drives
  - Similar caveats as Frontier local disks: you need to think about how to stage data.
- Aurora also has a DAOS storage system
  - 230 PiB capacity, 30 TiB/s throughput.
  - Under the covers it is a distributed object store.
    - You can access it like a conventional file system or use libraries like HDF5 or PyTorch that have been adapted to access DAOS natively.
  - DAOS is remarkably fast but has a little bit more of a learning curve.

Hands on exercises:
https://github.com/radix-io/hands-on

DARSHAN: UNDERSTANDING I/O BEHAVIOR

U.S. DEPARTMENT of ENERGY

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

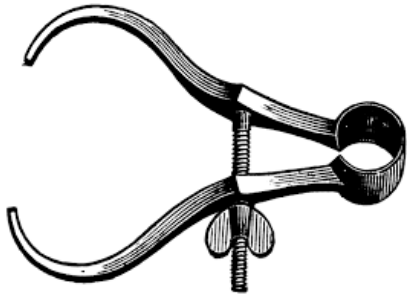
Argonne NATIONAL LABORATORY

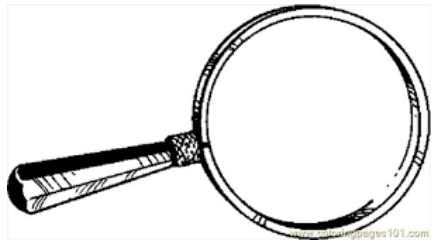
UNDERSTANDING I/O PROBLEMS IN YOUR APPLICATION

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.

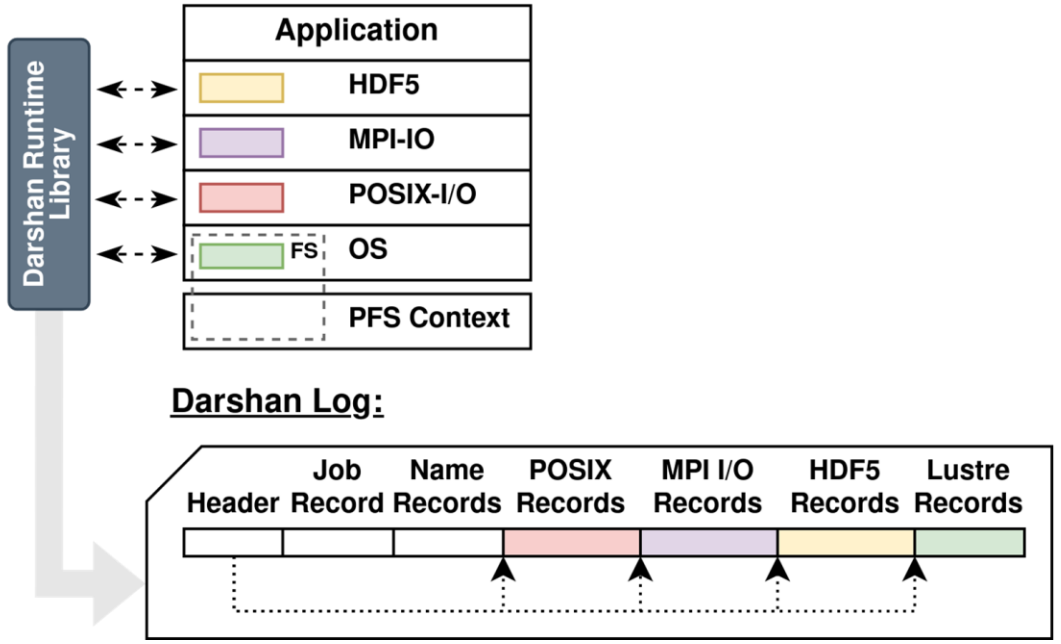


Figure courtesy Jakob Luettgau (Inria)

Hands on exercises:
https://github.com/radix-io/hands-on

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

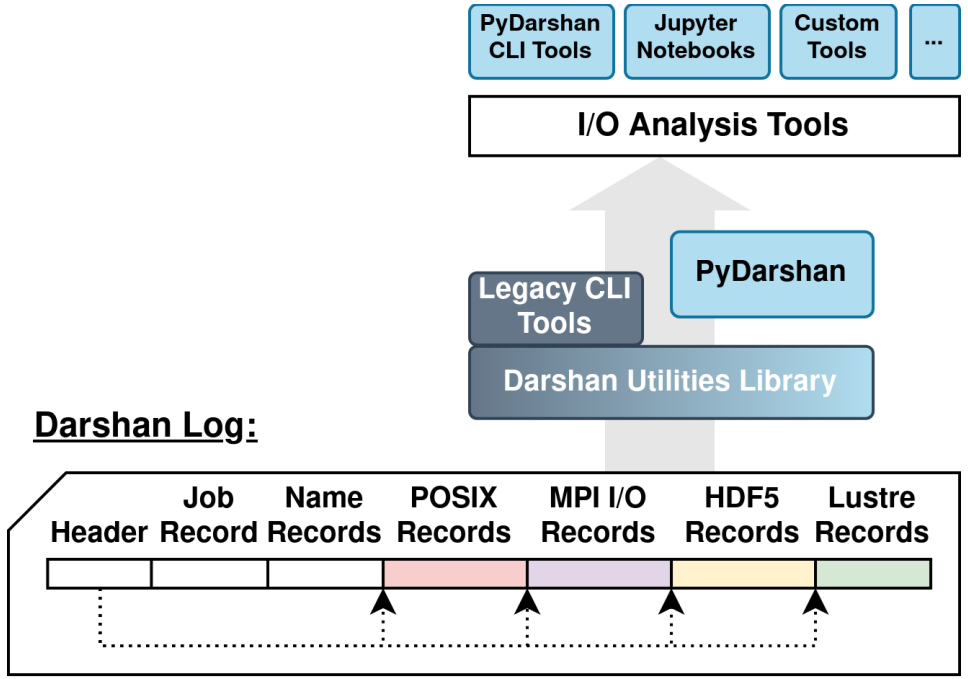


Figure courtesy Jakob Luettgau (Inria)

Hands on exercises:
https://github.com/radix-io/hands-on

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 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: I'll describe how Darshan works today on Aurora but as software and environment updates roll out things likely to change in the future.

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 2026.

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

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).

```
[robl@aurora-uan-0010 helloworld (main *+=)]$ qstat -u robl

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
---------------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
8736043.aurora-pbs-* robl debug helloworl* -- 2 416 -- 00:10 Q --
```

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>`**'.

```
[robl@aurora-uan-0010 helloworld (main *+=)]$ python -m darshan summary
robl_helloworld_id8735646-32903_8-5-61041-11693286648730049O5_1.darshan
Report generated successfully.
Saving report at location: /home/robl/src/hands-on/darshan-
demos/helloworld/robl_helloworld_id8735646-32903_8-5-61041-
11693286648730049O5_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

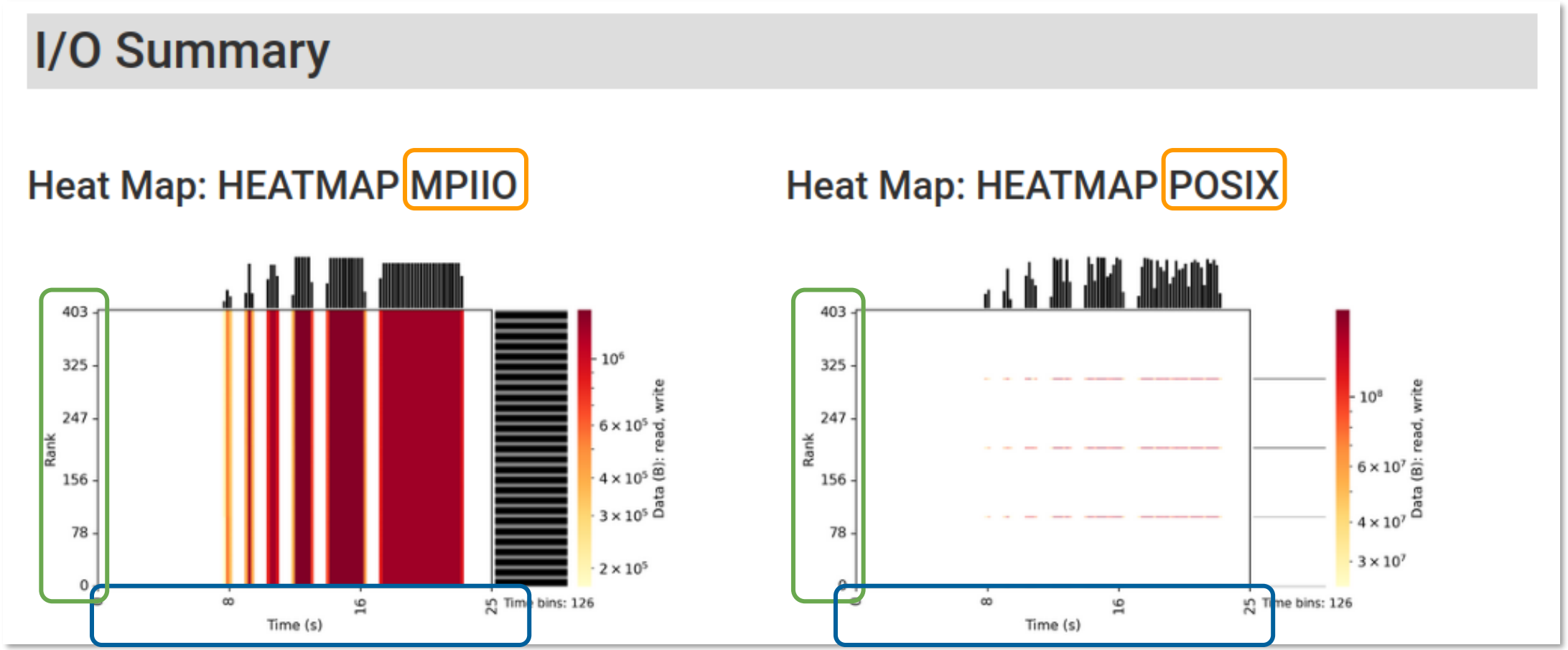
- First, use **`scp`** to copy the log to your own personal system.

```
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: I/O HEATMAPS



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.

Hands on exercises:
https://github.com/radix-io/hands-on

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/

Hands on exercises:
https://github.com/radix-io/hands-on

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.

Hands on exercises:
https://github.com/radix-io/hands-on