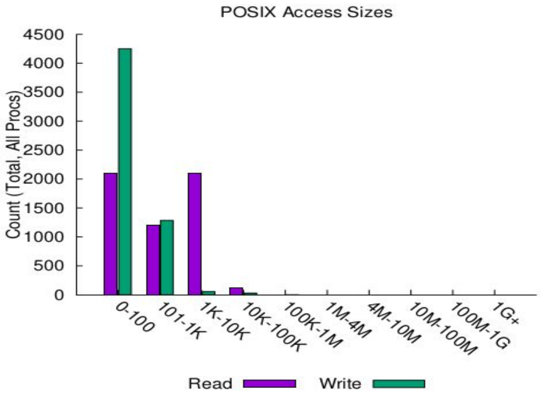
November 20, 2019

Analyzing Parallel I/O BoF



Shane Snyder
Argonne National Laboratory

Julian Kunkel
University of Reading

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

SC’19, Denver, Co

What’s our motivation for being here?

- Understanding of parallel I/O performance is critical to maximizing efficiency of our applications and datacenters
  - Deepening storage hierarchies, emerging hardware, and innovative runtimes are all exciting to consider, but no silver bullet to historical I/O woes
  - I/O monitoring/analysis more important than ever for extracting the most possible performance from increasingly complex/diverse systems

- This BoF provides a sampling of state-of-the-art I/O research from a range of data-intensive computing contexts, with the following goals in mind:
  - Inform community of recent advances in tools/techniques for I/O monitoring
  - Discuss experience/limitations of current approaches
  - Derive a roadmap for future I/O tools to capture, analyze, predict, and tune I/O

Introducing our speakers

- **What’s New with Darshan?**
  - Shane Snyder, ANL
- **HPC Storage as a Blank Canvas in Google Cloud**
  - Dean Hildebrand, Google
- **Timeline-based I/O Behavior Assessment of Parallel Jobs**
  - Eugen Betke, DKRZ
- **Measuring I/O with Tau**
  - Kevin Huck, University of Oregon
- **State of I/O profiling in Forge**
  - Florent Lebeau, ARM
- **Research Community I/O patterns**
  - Gordon Gibb, EPCC
- **Tracking User-Perceived I/O Slowdown via Probing**
  - Julian Kunkel, University of Reading

Introducing our speakers

- ❖ **What’s New with Darshan?**
  - ➢ Shane Snyder, ANL
- ❖ **HPC Storage a Blank s in G le Cloud**
  - ➢ Dean Hild
- ❖ **Timeli llel Jobs**
  - ➢ Eug
- ❖ **Meas**
  - ➢ K
- ❖ **State of**
  - ➢ E
- ❖ **Research Com**
  - ➢ Gordon G b, EPCC
- ❖ **Tracking User-Perceived I/O Slowdown via Probing**
  - ➢ Julian Kunkel, University of Reading

Following these talks, Julian will lead a panel of our speakers.

We’d be happy to hear any questions/comments from the audience!

Notable SC’19 events related to parallel I/O

Plenty of opportunities to learn more!

- Technical program:
  - “End-to-End I/O Portfolio for the Summit Supercomputing Ecosystem” (**Thursday, 10:30 AM**), Oral et al.
  - “Revisiting I/O Behavior in Large-Scale Storage Systems: The Expected and the Unexpected” (**Thursday, 11:30 AM**), Patel et al.
- PDSW Workshop (**Monday**):
  - “Profiling Platform Storage Using IO500 and Mistral”, Monnier et al.
  - “Understanding Data Motion in the Modern HPC Data Center”, Lockwood et al.
  - “Active Learning-based Automatic Tuning and Prediction of Parallel I/O Performance”, Agarwal et al.
  - “Applying Machine Learning to Understand Write Performance of Large-scale Parallel Filesystems”, Xie et al.
- BoFs:
  - “The IO-500 and the Virtual Institute of I/O”, Markomanolis et al. (**Tuesday, 12:15PM**)
- Tutorials:
  - “Parallel I/O in Practice”, Latham et al. (**Sunday**)

What’s new with Darshan?

Recent/ongoing Darshan developments

- Instrumentation of non-MPI workloads (WIP, Darshan 3.2.0)
  - Breaks Darshan's dependence on MPI, greatly expanding instrumentation coverage
  - Developed with Glenn Lockwood (NERSC)
- Python bindings for the darshan-util library (WIP, Darshan 3.2.0)
  - Allows development of Python-based Darshan log analysis tools that can interact with native log format directly (i.e., no expensive conversion to text format)
  - Developed with Jakob Luettgau (DKRZ)
- Darshan eXtended Tracing (DXT) trace triggering (complete, Darshan 3.1.8)
  - Allows user-specified triggers to control which files are traced at runtime
    - *File-* or *rank-based* triggers using regexes (e.g., files ending in '.h5' or accessed by rank 0)
    - Access characteristics triggers, including frequent *small* or *unaligned* I/O accesses

Why do we need a non-MPI version of Darshan?

- MPI is traditionally a logical interposition point for HPC instrumentation tools

- But, focusing exclusively on MPI ignores entire classes of relevant workloads:
  - non-MPI runtimes (OpenMP, Legion, Charm++, HPX)
  - non-MPI distributed frameworks and data services (Spark, Dask, TensorFlow, Horovod)
  - file transfer utilities (cp, scp, bbcp, rsync, Globus)
  - other serial applications (bioinformatics applications like HMMER, usearch)

- An ability to instrument these *non-traditional* HPC I/O workloads is invaluable
  - Users can gain insights into and tune previously inaccessible workloads
  - System administrators can greatly improve instrumentation coverage to better understand system workload characteristics

Breaking Darshan’s dependence on MPI

- Required a significant refactor of Darshan’s codebase, but now there is logic for detecting MPI at Darshan build time and application runtime

- Non-MPI support is only available via Darshan’s shared library which applications must LD_PRELOAD

- This effort is still a WIP (i.e., **not** suitable for production deployment), but an experimental pre-release of this version is coming soon:

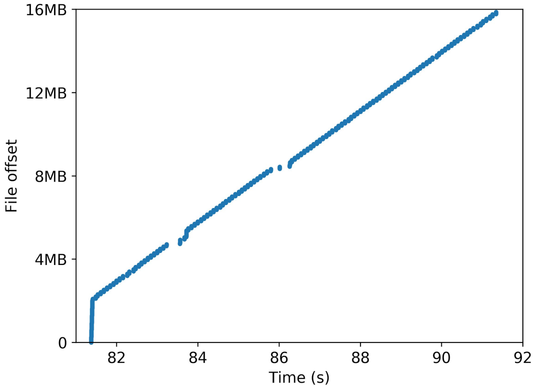
  https://lists.mcs.anl.gov/mailman/listinfo/darshan-users

  https://www.mcs.anl.gov/research/projects/darshan/download/

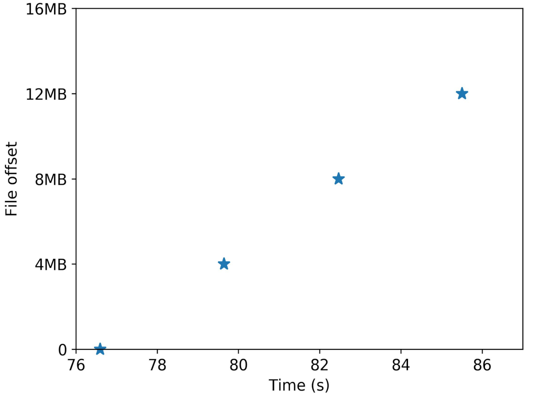
Darshan non-MPI use case

File transfer utilities

- File transfer utilities frequently used at HPC facilities to transfer data between systems or storage tiers
- Limited tuning opportunities, but we can gain a more complete picture of storage system usage
- Illustrated is the first 10 seconds of a painfully slow transfer from my laptop to Theta
  - Note drastic difference in access sizes (16 KiB vs 4 MiB)



source:
my laptop



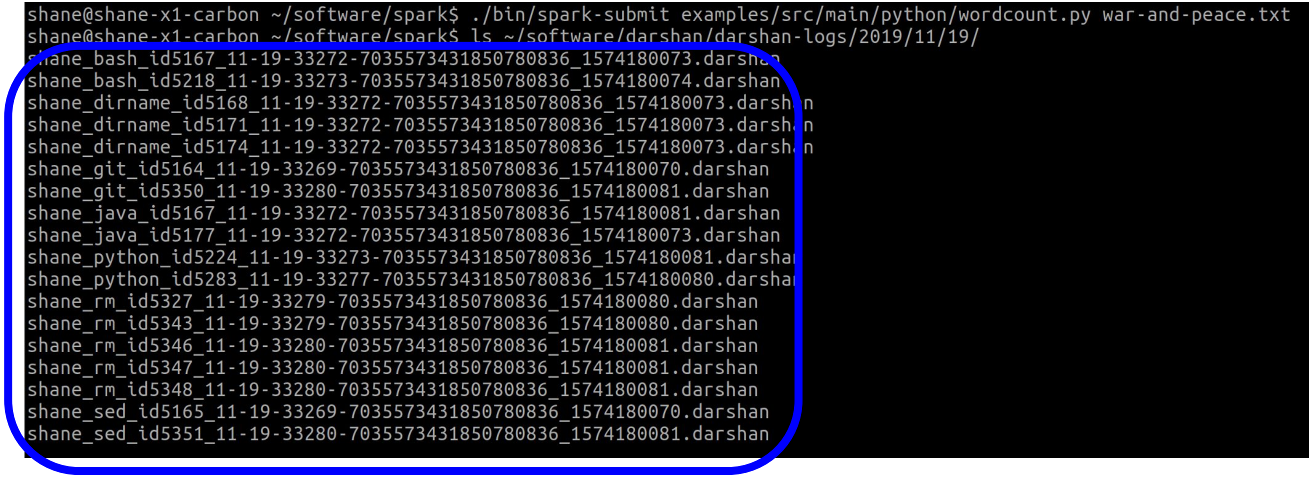
target:
Theta (ALCF)

scp

Darshan non-MPI use case

Instrumenting the Spark framework

- Non-MPI distributed compute/data services like Spark are becoming increasingly popular in HPC
- Illustrated are Darshan results from a simple Spark (Python) word count example



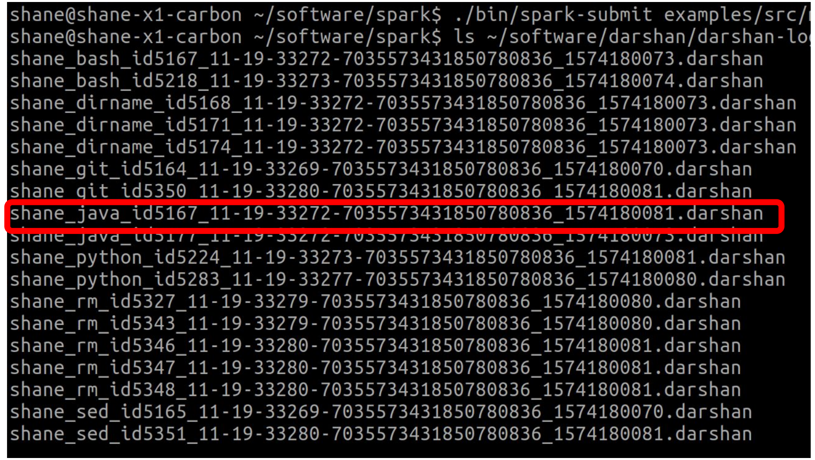
```
shane@shane-x1-carbon ~/software/spark$ ./bin/spark-submit examples/src/main/python/wordcount.py war-and-peace.txt
shane@shane-x1-carbon ~/software/spark$ ls ~/software/darshan/darshan-logs/2019/11/19/
shane_bash_id5167_11-19-33272-7035573431850780836_1574180073.darshan
shane_bash_id5218_11-19-33273-7035573431850780836_1574180074.darshan
shane_dirname_id5168_11-19-33272-7035573431850780836_1574180073.darshan
shane_dirname_id5171_11-19-33272-7035573431850780836_1574180073.darshan
shane_dirname_id5174_11-19-33272-7035573431850780836_1574180073.darshan
shane_git_id5164_11-19-33269-7035573431850780836_1574180070.darshan
shane_git_id5350_11-19-33280-7035573431850780836_1574180081.darshan
shane_java_id5167_11-19-33272-7035573431850780836_1574180081.darshan
shane_java_id5177_11-19-33272-7035573431850780836_1574180073.darshan
shane_python_id5224_11-19-33273-7035573431850780836_1574180081.darshan
shane_python_id5283_11-19-33277-7035573431850780836_1574180080.darshan
shane_rm_id5327_11-19-33279-7035573431850780836_1574180080.darshan
shane_rm_id5343_11-19-33279-7035573431850780836_1574180080.darshan
shane_rm_id5346_11-19-33280-7035573431850780836_1574180081.darshan
shane_rm_id5347_11-19-33280-7035573431850780836_1574180081.darshan
shane_rm_id5348_11-19-33280-7035573431850780836_1574180081.darshan
shane_sed_id5165_11-19-33269-7035573431850780836_1574180070.darshan
shane_sed_id5351_11-19-33280-7035573431850780836_1574180081.darshan
```

Many logs to filter through for a single Spark invocation -- 18 logs for this example

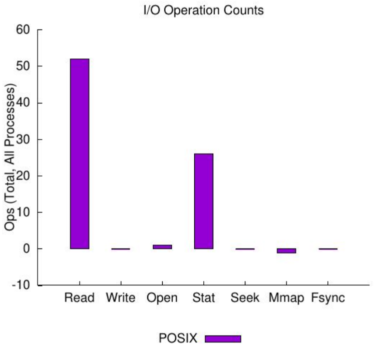
Darshan non-MPI use case

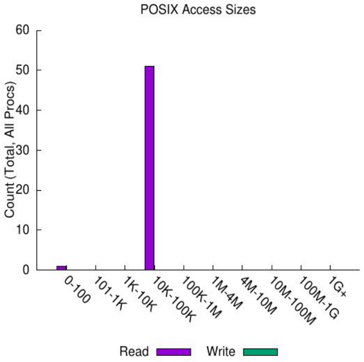
Instrumenting the Spark framework

- Non-MPI distributed compute/data services like Spark are becoming increasingly popular in HPC
- Illustrated are Darshan results from a simple Spark (Python) word count example



```
shane@shane-x1-carbon ~/software/spark$ ./bin/spark-submit examples/src/
shane@shane-x1-carbon ~/software/spark$ ls ~/software/darshan/darshan-lo
shane_bash_id5167_11-19-33272-7035573431850780836_1574180073.darshan
shane_bash_id5218_11-19-33273-7035573431850780836_1574180074.darshan
shane_dirname_id5168_11-19-33272-7035573431850780836_1574180073.darshan
shane_dirname_id5171_11-19-33272-7035573431850780836_1574180073.darshan
shane_dirname_id5174_11-19-33272-7035573431850780836_1574180073.darshan
shane_git_id5164_11-19-33269-7035573431850780836_1574180070.darshan
shane_git_id5350_11-19-33280-7035573431850780836_1574180081.darshan
shane_java_id5167_11-19-33272-7035573431850780836_1574180081.darshan
shane_java_id5177_11-19-33272-7035573431850780836_1574180073.darshan
shane_python_id5224_11-19-33273-7035573431850780836_1574180081.darshan
shane_python_id5283_11-19-33277-7035573431850780836_1574180080.darshan
shane_rm_id5327_11-19-33279-7035573431850780836_1574180080.darshan
shane_rm_id5343_11-19-33279-7035573431850780836_1574180080.darshan
shane_rm_id5346_11-19-33280-7035573431850780836_1574180081.darshan
shane_rm_id5347_11-19-33280-7035573431850780836_1574180081.darshan
shane_rm_id5348_11-19-33280-7035573431850780836_1574180081.darshan
shane_sed_id5165_11-19-33269-7035573431850780836_1574180070.darshan
shane_sed_id5351_11-19-33280-7035573431850780836_1574180081.darshan
```





Summarizing input text file

Thanks to all for attending!

BoF website will include slides, notes, etc.:
https://hps.vi4io.org/events/2019/sc-analyzing-io