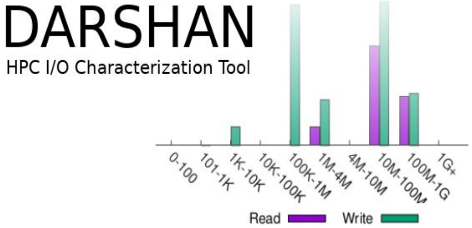
HEP-CCE IOS: Analysis of I/O Behavior in HEP Workflows with Darshan

Shane Snyder (ANL), Doug Benjamin (ANL), Patrick Gartung (FNAL), Ken Herner (FNAL), Rui Wang (ANL)

10/11/22

Darshan background

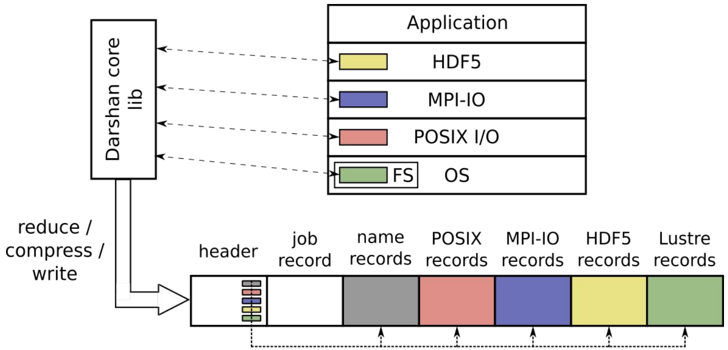


Darshan is a lightweight tool that can capture details about the I/O behavior of applications

- ➢ Inform tuning decisions of app scientists
- ➢ Gain insight into I/O trends on large-scale computing platforms

Darshan's design geared towards full-time deployment on HPC systems (currently on by default at ALCF, NERSC, OLCF, etc.)

- ➢ Transparent – no app changes required
- ➢ Low overhead – minimal perturbations to app runtime
- ➢ Modular – instrumentation can be extended to account for new I/O technologies



Default mode: capture bounded statistical records of I/O activity for each file accessed by the app

DXT (Darshan eXtended Tracing) mode: high-fidelity tracing of read/write operations

Darshan as a utility for HEP-CCE

Motivation: An ability to instrument the I/O behavior of HEP workflows is critical to characterizing and improving their usage of HPC storage

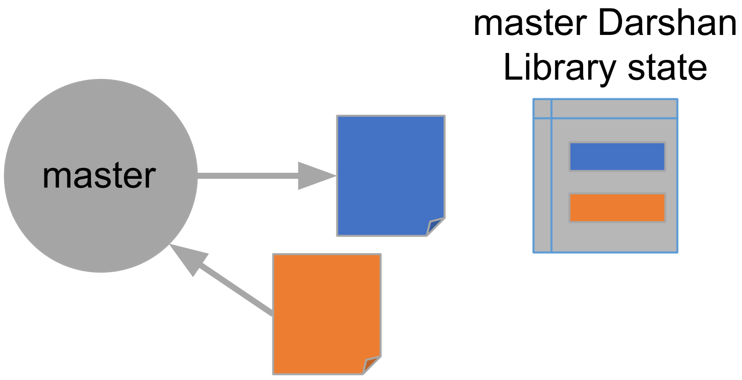
- ➢ The ongoing shift of HEP workflows to HPC facilities points to potential untapped I/O tuning opportunities here

Plan: Leverage Darshan in non-MPI mode to better understand HEP workflow I/O access patterns and performance characteristics

Darshan enhancements driven by HEP workflows

Darshan handling of `fork()`

ATLAS AthenaMP framework leverages `fork()` to spawn worker processes that perform event processing and I/O independently



Darshan library not originally designed to handle `fork()` gracefully

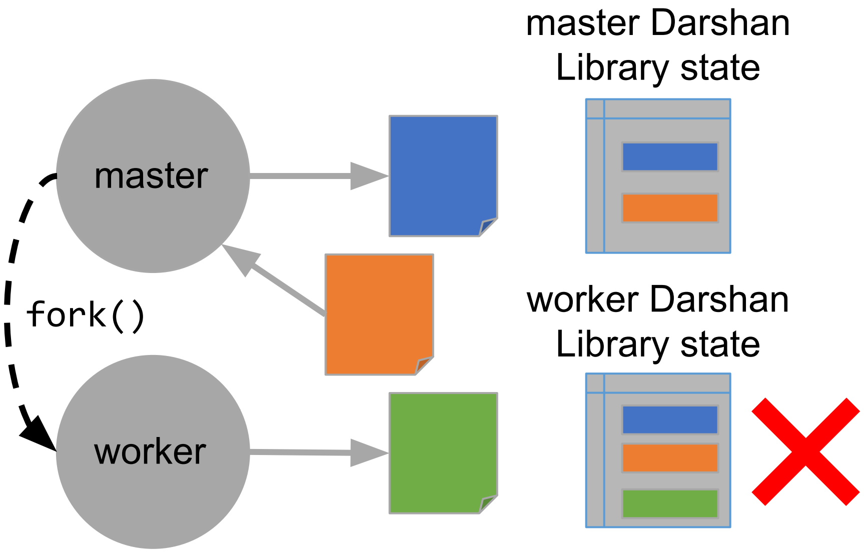
- Child process inherits copy of the parent library state due to copy-on-write semantics – child instrumentation state reflects access patterns of parent pre-`fork()` and child process post-`fork()`

Darshan handling of `fork()`

ATLAS AthenaMP framework leverages `fork()` to spawn worker processes that perform event processing and I/O independently

Darshan library not originally designed to handle `fork()` gracefully

- Child process inherits copy of the parent library state due to copy-on-write semantics – child instrumentation state reflects access patterns of parent pre-`fork()` and child process post-`fork()`

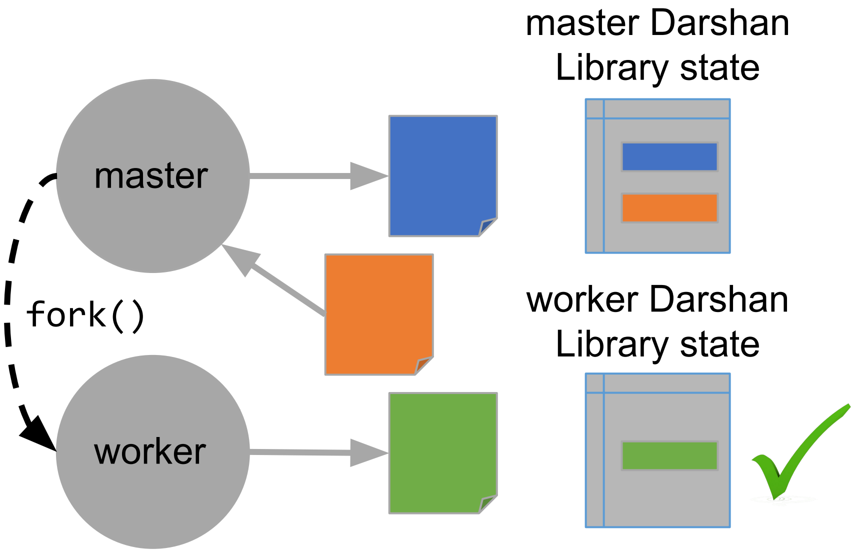


master input and master output files incorrectly accounted for in worker instrumentation state

Darshan handling of `fork()`

To address this Darshan was modified to properly handle apps that call `fork()`

- Use `pthread_atfork()` to register a callback that is executed before passing control to child process
- Darshan atfork callback re-initializes the library to clear all parent process state
- Child processes maintain mapping to corresponding parent process ID, allowing Darshan logs to capture process relationships



Darshan library runtime configuration

To bound memory overheads, Darshan imposes several internal memory limits

- Total memory for all module records
- Total memory for all record names
- Per-module limits on number of instrumented records

However, Darshan has traditionally offered insufficient mechanisms for fine-tuning library memory usage and instrumentation scope

Darshan library runtime configuration

To bound memory overheads, Darshan imposes several internal memory limits

- Total memory for all module records
- Total memory for all record names
- Per-module limits on number of instrumented records

Build and runtime configurable

hardcoded

However, Darshan has traditionally offered insufficient mechanisms for fine-tuning library memory usage and instrumentation scope

- No method to increase module record limits

Darshan library runtime configuration

To bound memory overheads, Darshan imposes several internal memory limits

- Total memory for all module records
- Total memory for all record names
- Per-module limits on number of instrumented records

Build and runtime configurable

hardcoded

However, Darshan has traditionally offered insufficient mechanisms for fine-tuning library memory usage and instrumentation scope

- No method to increase module record limits
- Limited methods for restricting which record names to instrument

```
export DARSHAN_EXCLUDE_DIRS=”/home,/tmp”
```

Users can only exclude record names using directory prefixes

Darshan library runtime configuration

This lack of user control can complicate full instrumentation of apps, particularly the Python frameworks used in HEP projects – often ROOT I/O is completely missed!

WARNING: This Darshan log contains incomplete data. This happens when a module runs out of memory to store new record data. Please run darshan-parser on the log file for more information.

To address this problem, we added a comprehensive runtime configuration system to Darshan, allowing users to control specific instrumentation settings:

- Active/inactive instrumentation modules
- Global and per-module memory limits
- Record name exclusions
- etc.

```
# enable DXT module (off by default)
MOD_ENABLE      DXT_POSIX

# allocate 2000 file records for POSIX module
# (darshan only allocates 1024 per-module by default)
MAX_RECORDS     2000      POSIX

# ignore record names prefixed with "/home", with an
# overriding inclusion for files with a ".out" suffix)
NAME_EXCLUDE    ^/home         POSIX
NAME_INCLUDE    .out$          POSIX
```

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

filename_glob	glob_count
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.h$	3111
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hpp$	798
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.py$	616
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.so.*	530
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4PARTICLEXS/.*	328
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4EMLOW/.*	283
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hxx$	255
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/PhotonEvaporation/.*	235
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pyc$	204
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pcm$	159
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*\.h$	146
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*include.*	140
/lib64/.*\.so.*	56

* File name regex code borrowed from Tyler Reddy (LANL)

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

filename glob	glob count
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.h$	3111
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hpp$	798
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.py$	616
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.so.*	530
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4PARTICLEXS/.*	328
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4EMLOW/.*	283
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hxx$	255
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/PhotonEvaporation/.*	235
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pyc$	204
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pcm$	159
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*\.h$	146
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*include.*	140
/lib64/.*\.so.*	56

3500+ header files

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

filename_glob	glob_count
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.h$	3111
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hpp$	798
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.py$	616
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.so.*	530
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4PARTICLEXS/.*	328
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4EMLOW/.*	283
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hxx$	255
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/PhotonEvaporation/.*	235
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pyc$	204
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pcm$	159
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*\.h$	146
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*include.*	140
/lib64/.*\.so.*	56

800+ Python source & compiled code

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

filename_glob	glob_count
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.h$	3111
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hpp$	798
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.py$	616
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.so.*	530
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4PARTICLEXS/.*	328
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4EMLOW/.*	283
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hxx$	255
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/PhotonEvaporation/.*	235
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pyc$	204
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pcm$	159
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*\.h$	146
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*include.*	140
/lib64/.*\.so.*	56

500+ shared libraries

U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
BROOKHAVEN NATIONAL LABORATORY
Fermilab
BERKELEY LAB Bringing Science Solutions to the World

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

filename_glob	glob_count
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.h$	3111
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hpp$	798
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.py$	616
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.so.*	530
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4PARTICLEXS/.*	328
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/G4EMLOW/.*	283
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.hxx$	255
/cvmfs/atlas.cern.ch/repo/sw/software/22.0/Geant4/.*/data/PhotonEvaporation/.*	235
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pyc$	204
/cvmfs/atlas.cern.ch/repo/sw/software/.*\.pcm$	159
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*\.h$	146
/cvmfs/sft.cern.ch/lcg/releases/gcc/.*include.*	140
/lib64/.*\.so.*	56

and more…

Darshan library runtime configuration

Full instrumentation of ATLAS AthenaMP requires 7000+ file records:

/cvmfs/atlas.cern.ch/repo/sw/.*\.root.*	8
/cvmfs/atlas-condb.cern.ch/repo/conditions/.*\.root.*	5
/global/cscratch1/sd/ssnyder/.*\.root.*	3

and finally, a few ROOT files

Darshan record name exclusion/inclusion properties can help ensure we get the instrumentation data we want without exorbitant memory costs

```
# ignore /cvmfs directory, except ROOT files
NAME_EXCLUDE     ^/cvmfs          POSIX
NAME_INCLUDE     .*root.*         POSIX
```

Potential next steps with Darshan in IOS

Utilize new Darshan instrumentation modules to better understand I/O behavior of other IOS activities

- *HDF5 module*: insights into DUNE HDF5 usage, ROOT→HDF5 serialization efforts
- *DAOS module*: insights into ROOT's RNTuple DAOS backend

Utilize PyDarshan log analysis tools and extend them to help analyze I/O characteristics of HEP workflows