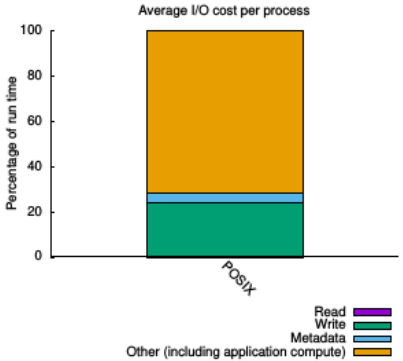
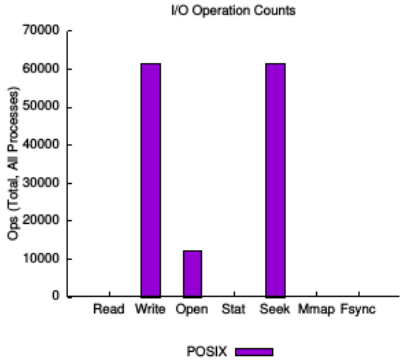
Argonne NATIONAL LABORATORY

WHAT'S NEW WITH DARSHAN?



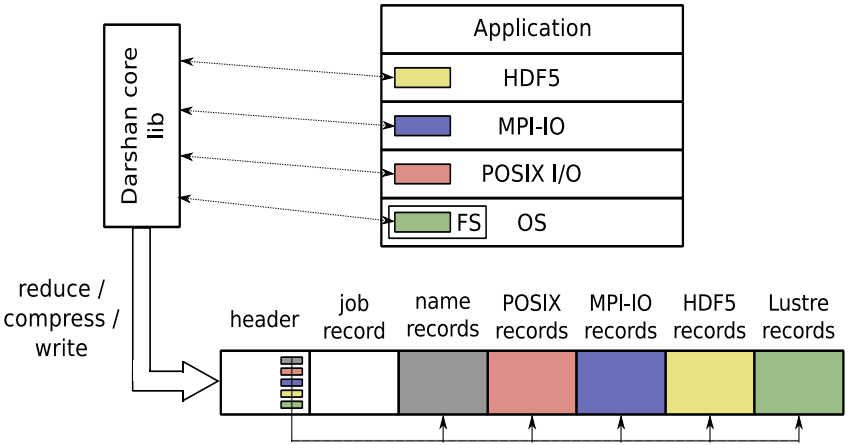


SHANE SNYDER

Nov. 16, 2016

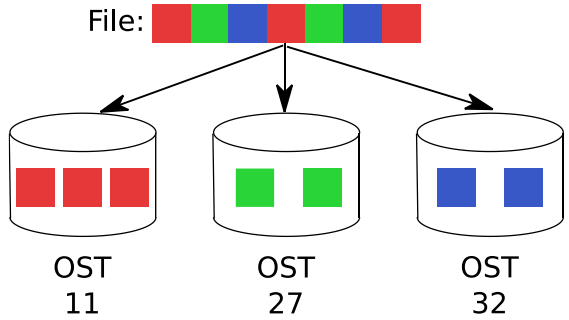
MODULARIZED INSTRUMENTATION

- Modularized architecture
  - Instrumentation modules:
    - Instruments arbitrary source of I/O data (I/O interface, file system, etc.)
    - Creates/registers/updates data records characterizing application I/O workload
  - Darshan core library:
    - Exposes interface for modules to coordinate with Darshan
    - Compresses and writes module data to log
- Self-describing log file format to index each module's data



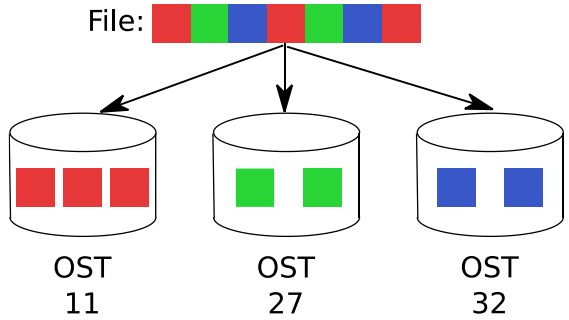
NEW INSTRUMENTATION MODULES

- Lustre
  - Instruments Lustre FS details and stripe parameters using `ioctls`:
    - Stripe width & stripe size
    - Number of Lustre OSTs & MDTs
    - Enumeration of OSTs allocated to a file
  - Provides view of how application workloads interact with FS



NEW INSTRUMENTATION MODULES

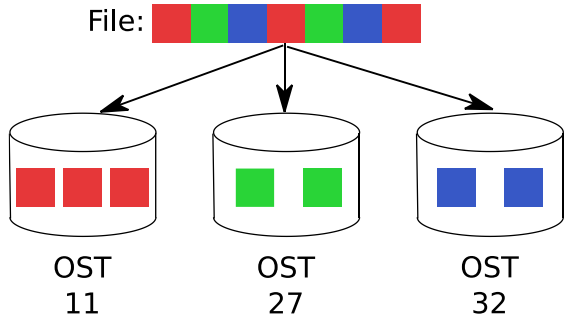
- Lustre
  - Instruments Lustre FS details and stripe parameters using `ioctls`:
    - Stripe width & stripe size
    - Number of Lustre OSTs & MDTs
    - Enumeration of OSTs allocated to a file
  - Provides view of how application workloads interact with FS



- `stdio` interface
  - Instruments `stdio.h` functions (e.g., `fscanf()`, `fprintf()`, etc.)
  - Extends coverage to applications using text-based I/O
    - Genomics and bioinformatics apps

NEW INSTRUMENTATION MODULES

- Lustre
  - Instruments Lustre FS details and stripe parameters using `ioctls`:
    - Stripe width & stripe size
    - Number of Lustre OSTs & MDTs
    - Enumeration of OSTs allocated to a file
  - Provides view of how application workloads interact with FS

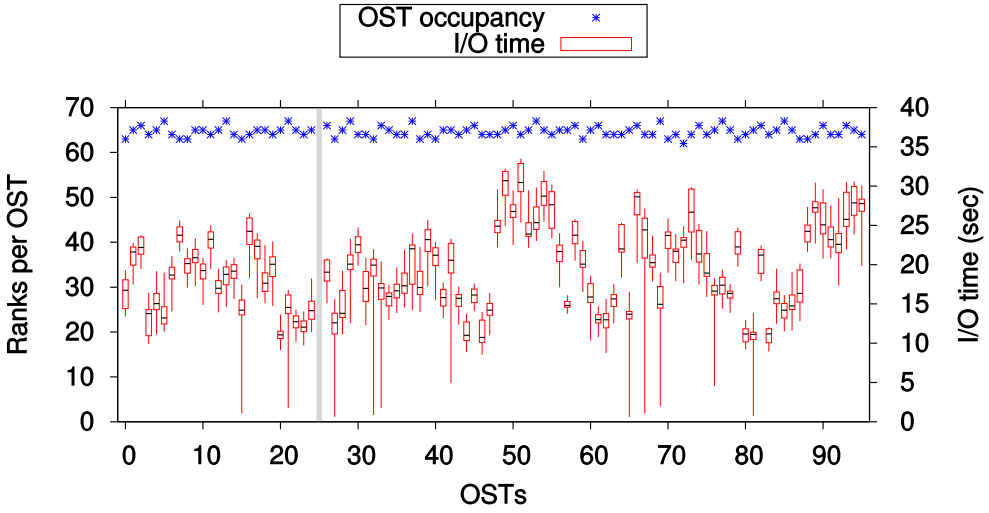


- `stdio` interface
  - Instruments `stdio.h` functions (e.g., `fscanf()`, `fprintf()`, etc.)
  - Extends coverage to applications using text-based I/O
    - Genomics and bioinformatics apps

- BG/Q
  - Provides details on how jobs interact with BG/Q platform:
    - Compute node & I/O node count
    - Processes per compute node
    - Active torus dimensions

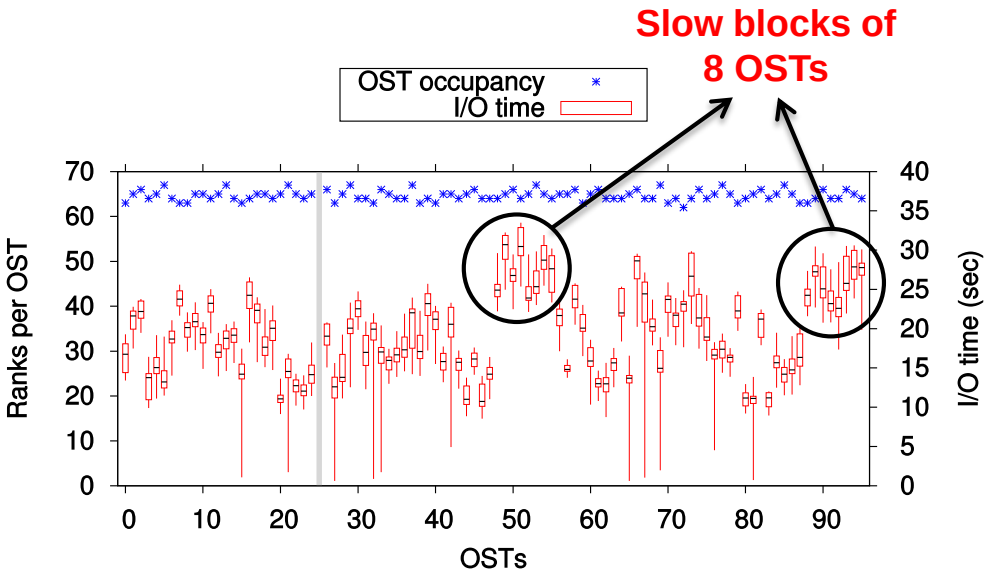
CASE STUDY: HACC-IO

- Workload details:
  - 6,144 processes, file-per-process
  - Checkpoint only
    - ~100 MiB/process, 600 GiB total write volume
  - Output to Lustre scratch volume on Edison system at NERSC
    - Stripe width of 1 (each process writes to exactly one OST)
- What can we learn from cross-correlating data from Darshan's POSIX & Lustre modules?



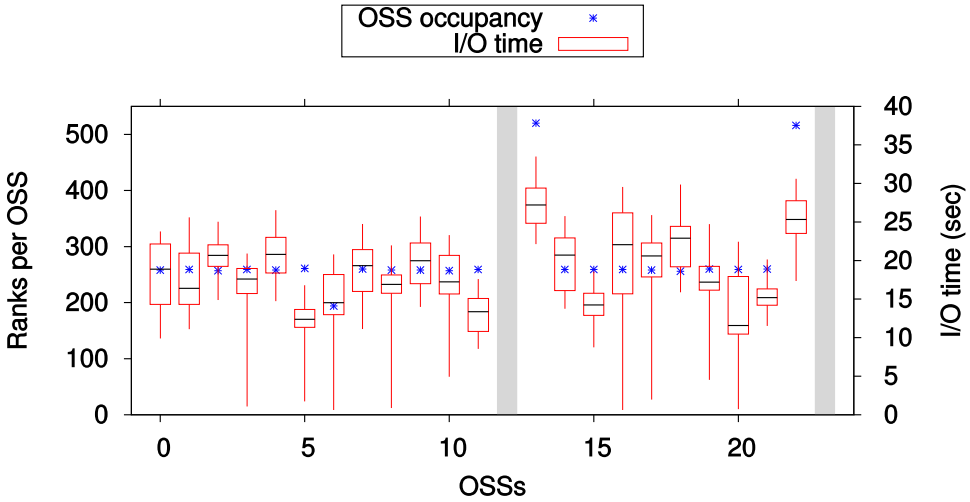
CASE STUDY: HACC-IO

- Workload details:
  - 6,144 processes, file-per-process
  - Checkpoint only
    - ~100 MiB/process, 600 GiB total write volume
  - Output to Lustre scratch volume on Edison system at NERSC
    - Stripe width of 1 (each process writes to exactly one OST)
- What can we learn from cross-correlating data from Darshan's POSIX & Lustre modules?



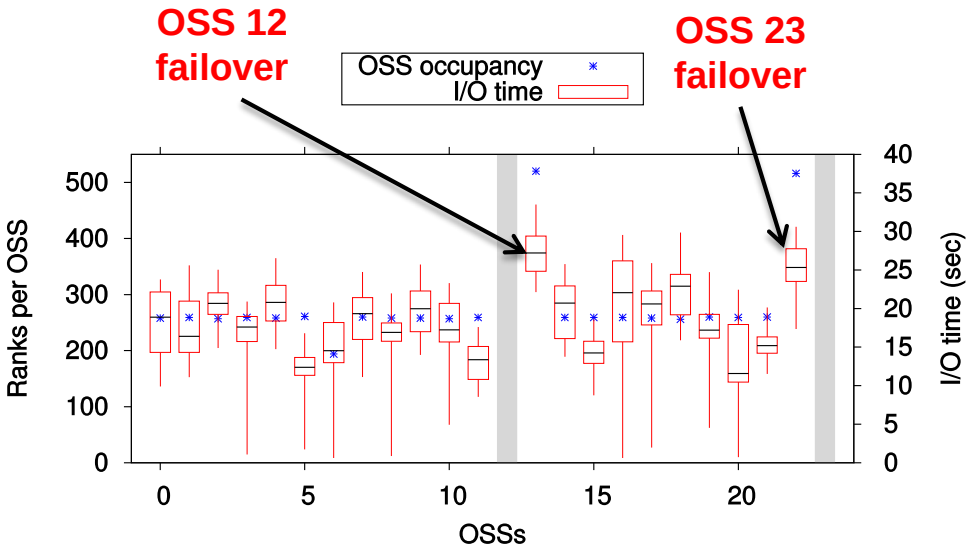
CASE STUDY: HACC-IO

- To investigate further, we mapped OSTs to corresponding OSSes
  - Each block of 4 OSTs should map to a distinct OSS
  - However, the blocks of 8 slow OSTs we observed each map to a single OSS rather than 2?
- We were able to confirm failures in the underlying storage appliance with NERSC systems staff
  - Pairs of OSSes provide active-active failover capability



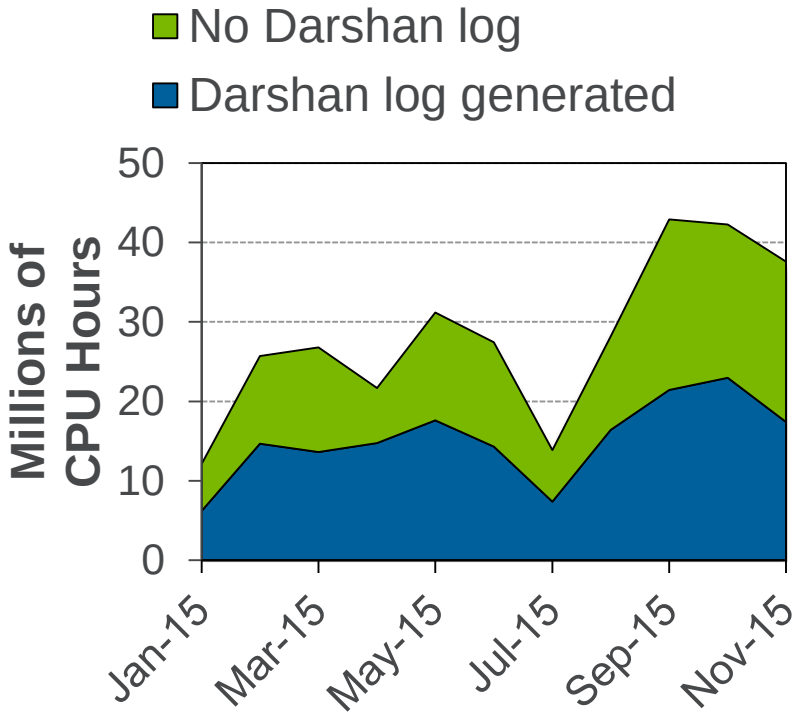
CASE STUDY: HACC-IO

- To investigate further, we mapped OSTs to corresponding OSSes
  - Each block of 4 OSTs should map to a distinct OSS
  - However, the blocks of 8 slow OSTs we observed each map to a single OSS rather than 2?
- We were able to confirm failures in the underlying storage appliance with NERSC systems staff
  - Pairs of OSSes provide active-active failover capability



INSTRUMENTING APPS THAT TERMINATE ABNORMALLY

- Perhaps due to app hitting wall-time limit or because of a general crash
- New robust logging mechanism to combat this problem:
  - Darshan memory allocations replaced with calls to `mmap`
    - `MAP_SHARED` flag forces propagation of updates to temporary log file
    - Recommended backing store is a node-local RAMdisk
- Merge tool used to combine uncompressed, per-process logs into traditional per-job log files



COMING SOON: DARSHAN EXTENDED TRACING (DXT)

- Detailed I/O tracing of POSIX & MPI-IO interfaces
  - Functionality can be enabled at runtime using environment variable
  - Individual I/O ops can be mapped to specific Lustre OSTs
- Tentatively planning to include DXT in the next Darshan release (3.1.3)
- Contributed by Cong Xu and Intel's High Performance Data Division (HPDD)

```
# DXT, file_id: 11616430107429575973, file_name: /home/shane/software/benchmarks/ior/testFile
# DXT, rank: 0, write_count: 4, read_count: 4
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)
 X_POSIX       0  write        0               0       262144      0.0067      0.0072
 X_POSIX       0  write        1          262144       262144      0.0072      0.0083
 X_POSIX       0  write        2          524288       262144      0.0083      0.0089
 X_POSIX       0  write        3          786432       262144      0.0089      0.0096
 X_POSIX       0   read        0               0       262144      0.0121      0.0122
 X_POSIX       0   read        1          262144       262144      0.0122      0.0123
 X_POSIX       0   read        2          524288       262144      0.0123      0.0124
 X_POSIX       0   read        3          786432       262144      0.0124      0.0125
```

QUESTIONS?