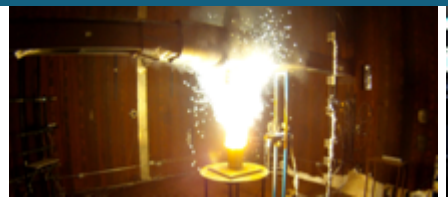
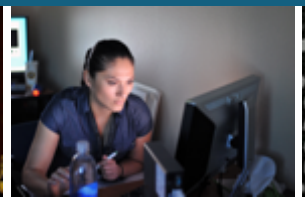
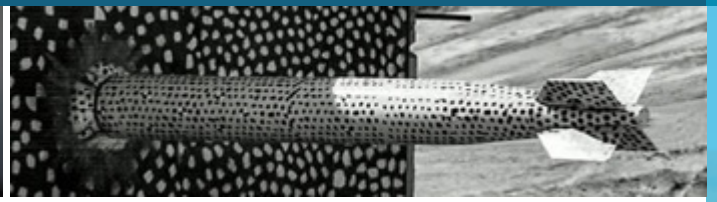
SAND2022-16879C

Sandia National Laboratories

LDMS Darshan Connector: For Run Time Diagnosis of HPC Application I/O Performance







Sara Walton, SNL
Devesh Tiwari, Northeastern University
Ana Luisa V. Solórzano, Northeastern University
Ben Schwaller, SNL
Jim M. Brands, SNL

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Motivation

- I/O performance degradation is one of the main culprits of application behavior variation
- Darshan currently collects detailed I/O information about an application run, however it aggregates data over the course of the run and reports out at the end
- Darshan Extended Trace Plugin can acquire run time data but it is deliberately constrained in buffer/memory, and therefore data, size, and also reports at the end
- Timestamped time-series of I/O data available at run time is needed to find errors, correlate with other events, and take actionable response

Benefits

- Captures I/O events at runtime to generate time-series histories of I/O behavior
- Provides absolute time-series timestamps that can be used to evaluate I/O performance in many levels of the I/O subsystem and correlate to system monitoring data / logs
- Captures read/write/close/open/flushes
- Captures POSIX, MPI-IO and STDIO
- Distinguishes STDIN, STDOUT, and STDERR for POSIX opens

Overview

A framework that integrates Darshan and LDMS to provide low-latency monitoring of I/O event data during runtime.

Darshan: a lightweight I/O characterization tool that transparently captures application I/O behavior from HPC applications with minimal overhead.

Lightweight Distributed Metric Service (LDMS): a low-overhead production monitoring system that can run on HPC machines. Used to collect, aggregate and store timeseries I/O event data during runtime.

I/O Event Data Collection

darshanConnector

A Darshan-LDMS Integration functionality that utilizes **LDMS Streams** capability to collect Darshan's original I/O tracing, Darshan's eXtended tracing (DXT) and absolute timestamp during runtime. It can optionally *publish* a message in **JSON** format to the **LDMS Streams daemon.**

I/O Event Data Storage

Distributed Scalable Object Store (DSOS)

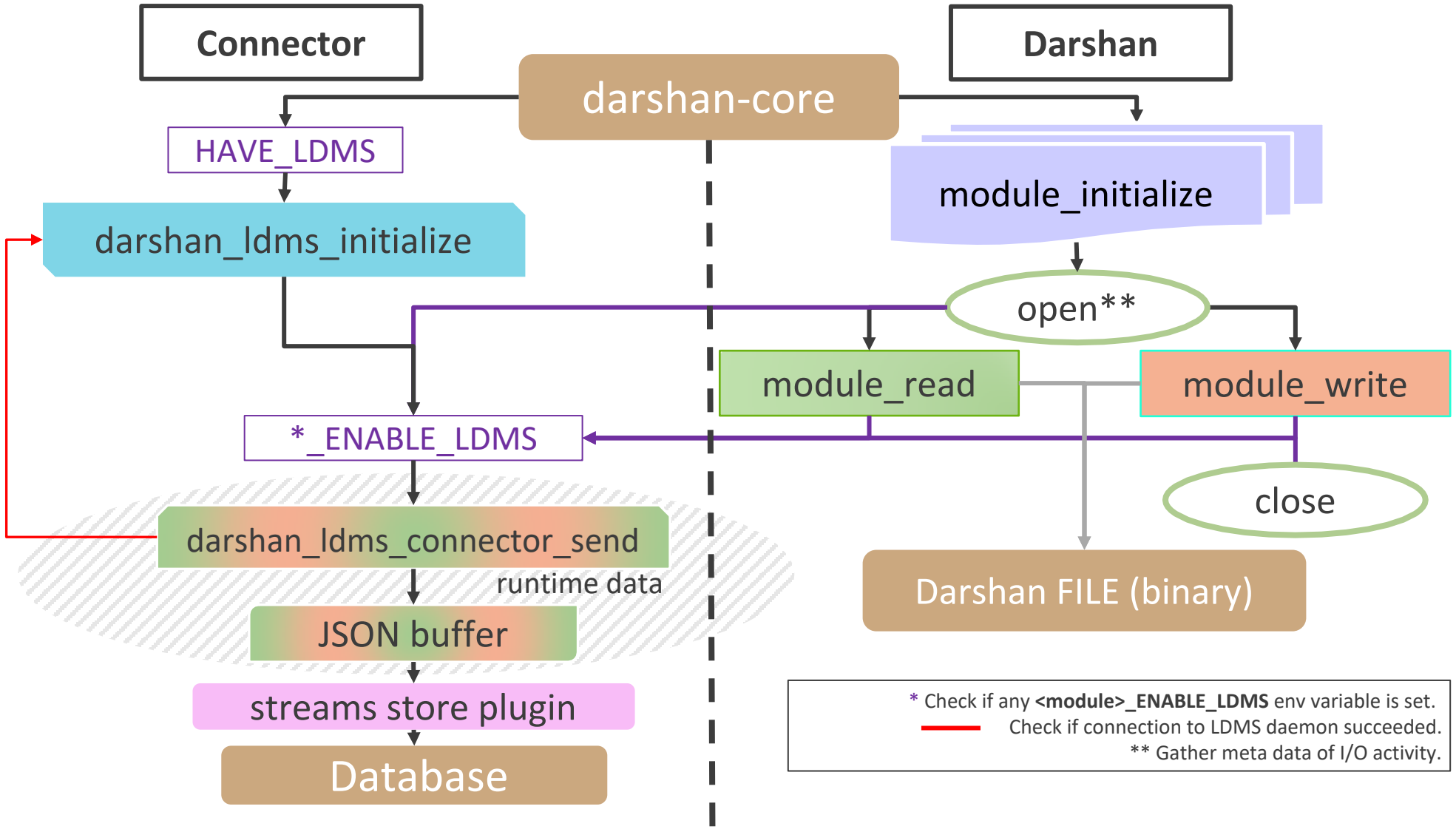
A storage database designed to manage large volumes of HPC data efficiently. Used to store timeseries I/O event data for analysis and visualization.

I/O Event Data Analysis and Visualization

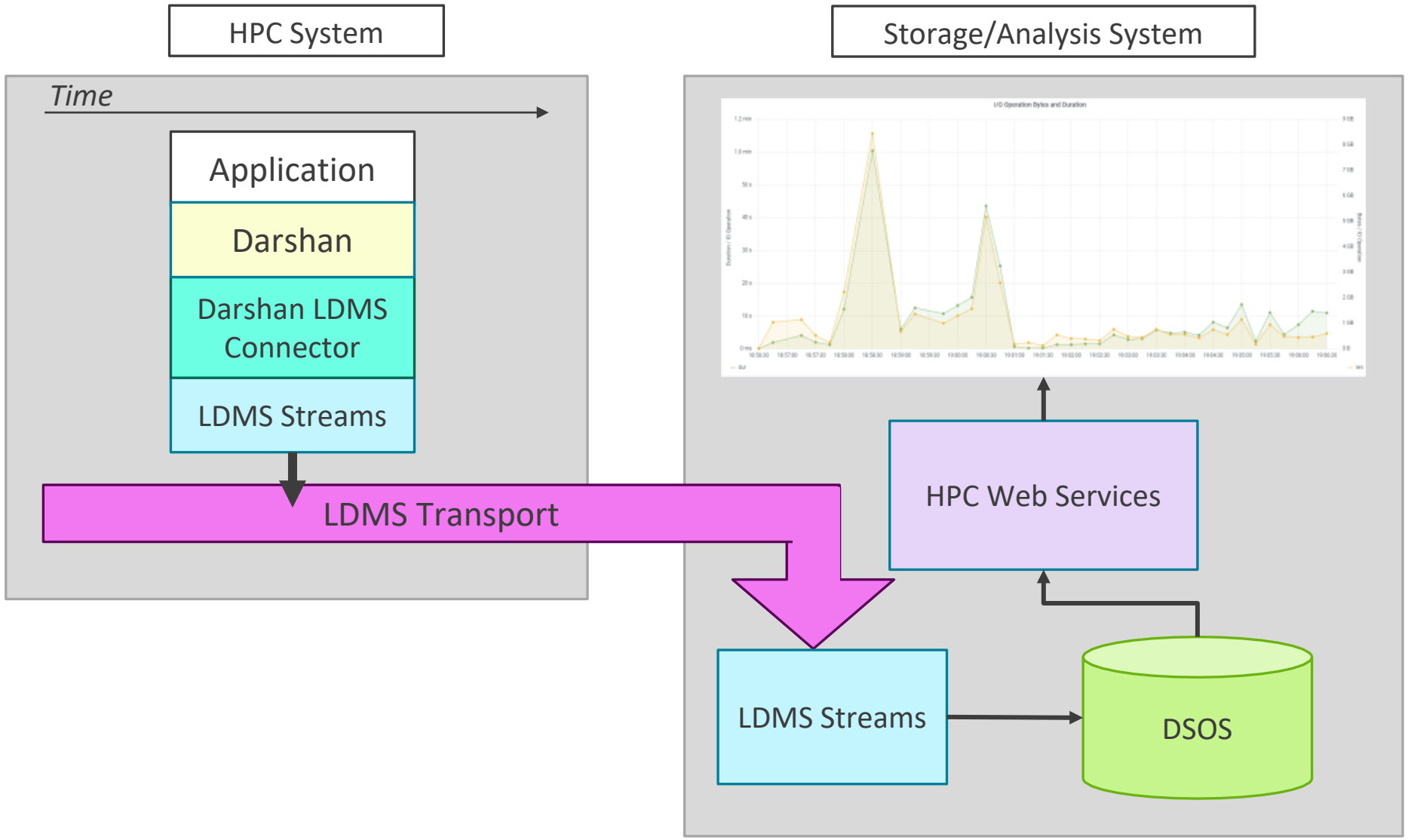
HPC Web Services

An analysis and visualization infrastructure for LDMS that integrates an open-source web application, Grafana. Used to generate new and meaningful analyses about I/O behavior.

darshanConnector



Overview


HPC System
Storage/Analysis System
Time
Application
Darshan
Darshan LDMS Connector
LDMS Streams
LDMS Transport
HPC Web Services
LDMS Streams
DSOS

Darshan LDMS Integration: Defined metrics

uuid	User ID of the job run
exe	Absolute directory of the application executable
module	Name of the Darshan module data being collected
ProducerName	Name of the compute node the application is running on
switches	Number of times access alternated between read and write
file	Absolute directory of the filename where the operations are performed
rank	Rank of the processes at I/O
flushes	Number of "flush" operations. It is the HDF5 file flush operations for H5F, and the dataset flush operations for H5
record_id	Darshan file record ID of the file the dataset belongs to
max_byte	Highest offset byte read and written per operation
type	The type of JSON data being published: MOD for gathering module data or MET for gathering static meta data
job_id	The Job ID of the application run
op	Type of operation being performed (i.e. read, write, open, close)
cnt	The count of the operations performed per module per rank. Resets to 0 after each "close" operation
seg	A list containing metrics names per operation per rank
seg:pt_sel	HDF5 number of different access selections
seg:dur	Duration of each operation performed for the given rank (i.e. a rank takes "X" time to perform a r/w/o/c operation)
seg:len	Number of bytes read/written per operation per rank
seg:ndims	HDF5 number of dimensions in dataset's dataspace
seg:reg_hslab	HDF5 number of regular hyperslabs
seg:irreg_hslab	HDF5 number of irregular hyperslabs
seg:data_set	HDF5 dataset name
seg:npoints	HDF5 number of points in dataset's dataspace
seg:timestamp	End time of given operation per rank (in epoch time)

TABLE I: Metrics defined in the JSON file published to the *Darshan LDMS Integration*.

Darshan-LDMS Integration: Experiments & Use Cases

Experiments

1. **Machine:** Stria - 2 Ghz Arm Cavium ThunderX2 cores, 266 Nodes
2. **Experiments:** 16 nodes with 32 ranks each (512 total), 5 repetitions each
3. **Applications**:
    1. HACC-IO 5 million particles | HACC-IO 10 million particles
        1. HACC-IO: An IO-performance benchmark that uses n-body simulation for collision-less fluids in space.
    2. MPI-IO with collective operations | MPI-IO without collective operations
        1. MPI-IO: Darshan utility to test MPI I/O performance on HPC machines.
4. **Filesystem**: Lustre | NFS
5. **Logs**: Darshan-LDMS Integration | Darshan | DXT

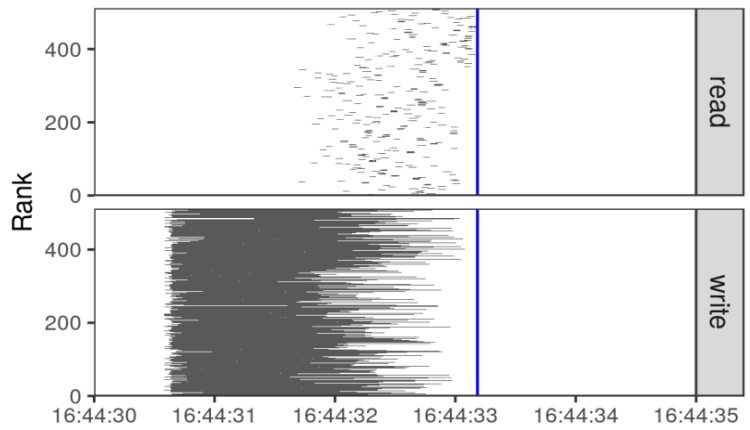
Use Cases

1. Detect I/O-related runtime bottlenecks (utilizing time-series data)
2. Correlate I/O and non-I/O bottlenecks (joint analysis with system health logs)

Use Case: Detect I/O-related runtime bottlenecks

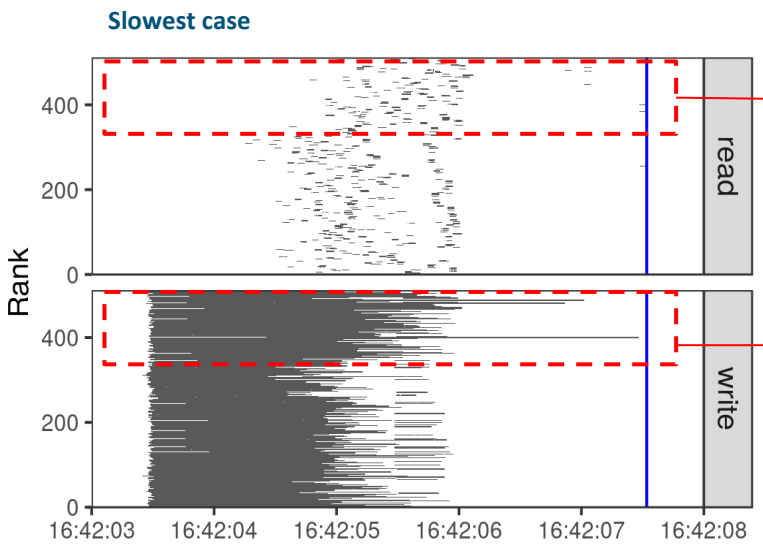
HACC-IO - Lustre

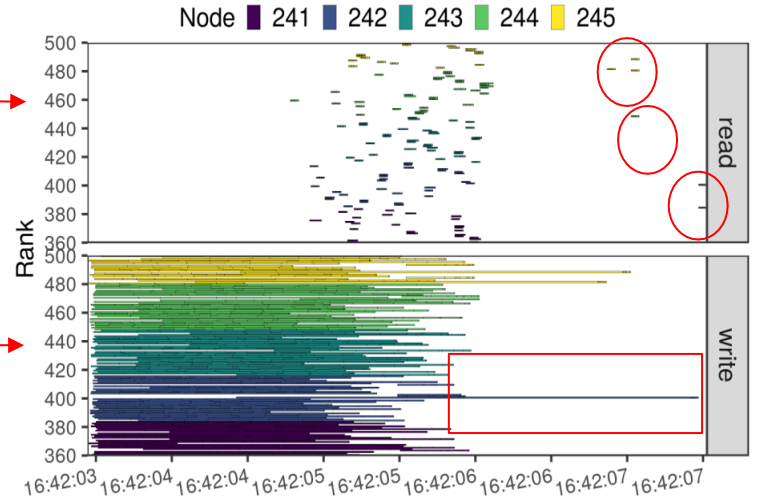
Fastest case



- Difference in the execution time of I/O operations in the same machine for the same experiment
- Previous investigations show no spatial imbalance between ranks (number of operations and size of requests).
- Is the difference from temporal imbalance? With the absolute timestamps we can investigate and correlate with the system health logs.

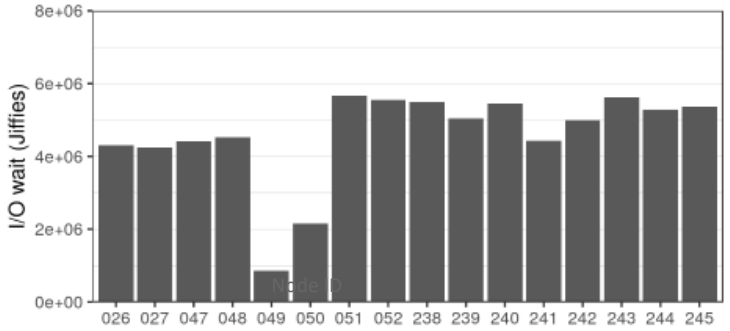
Slowest case

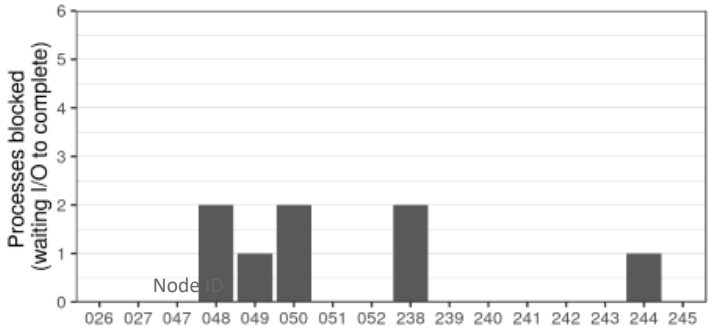




Use Case 2: Correlate with System Data

Fastest HACC-I/O run

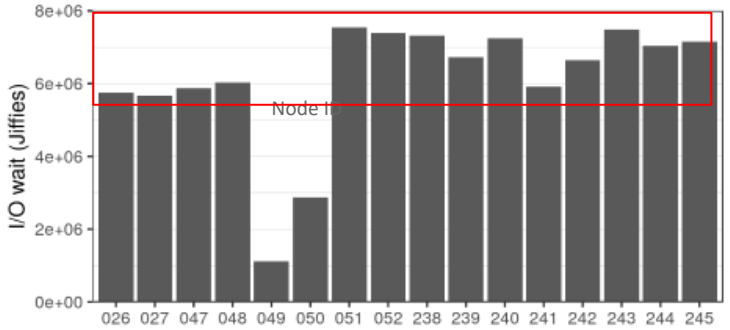


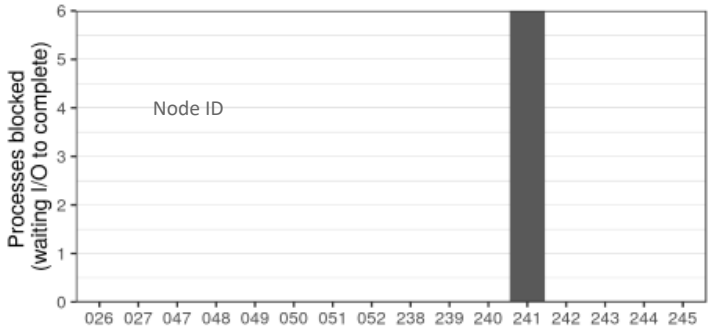


- Slowest case had 6 processes blocked in the same node (241).
- This is one of the slowest nodes.
- And 2e+06 Jiffes more in I/O wait than the fastest case.

/proc/stat iowait data

Slowest HACC-I/Orun





Darshan-LDMS Integration: Next Steps

- Overhead of LDMS-Darshan integration
  - Experiments collecting Darshan information only + LDMS-Integration
- Validation against a varied set of applications:
  - Benchmarks
    - HACC-IO
    - MPI-IO: Bind-to node in different repetitions and report-bindings
    - Scientific I/O kernels: FLASH, OpenPMD, E2E benchmarks, and block-cyclic I/O
  - “Real world” applications
    - Quantum Espresso: https://www.quantum-espresso.org/
    - Distributed Deep Learning: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9188225
    - STDIO applications
- Collect LDMS system monitoring information in the background
  - To show the benefits of correlating absolute timestamp with system health
  - Collect per core for all cores
- Investigate difference between NFS and Lustre filesystems

Questions?