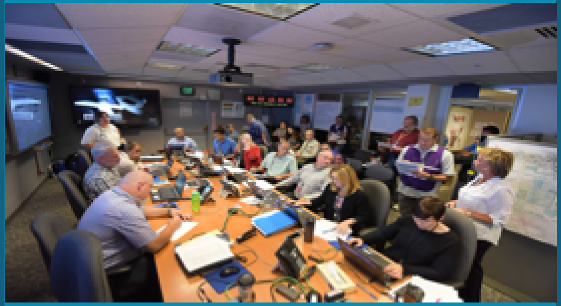
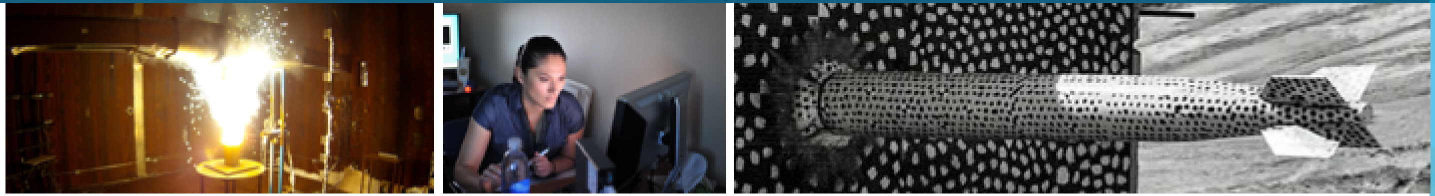
SAND2022-11736C

Sandia National Laboratories

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





Sara Walton, SNL

Omar Aaziz, SNL

Ana Luisa V. Solórzano, Northeastern University

Ben Schwaller, SNL

Introduction & Motivation

Many efforts to identify the origin of application I/O performance variation have been performed in post-run analyses.

- I/O Characterization tools (e.g. Darshan, Datadog, etc.)
- Linux tools for disk I/O monitoring (e.g. iotop, dstat, etc.)

However, I/O Performance continues to show high variations on large-scale production systems. This can be caused by a variety of system related components (i.e. system usage, file system, network congestion, etc.)

- Difficult to determine root cause of I/O related problems
- Difficult to have a thorough understanding of throughput for system-specific behaviors and I/O performance in similar applications across a system

Limitations with existing I/O characterization tools

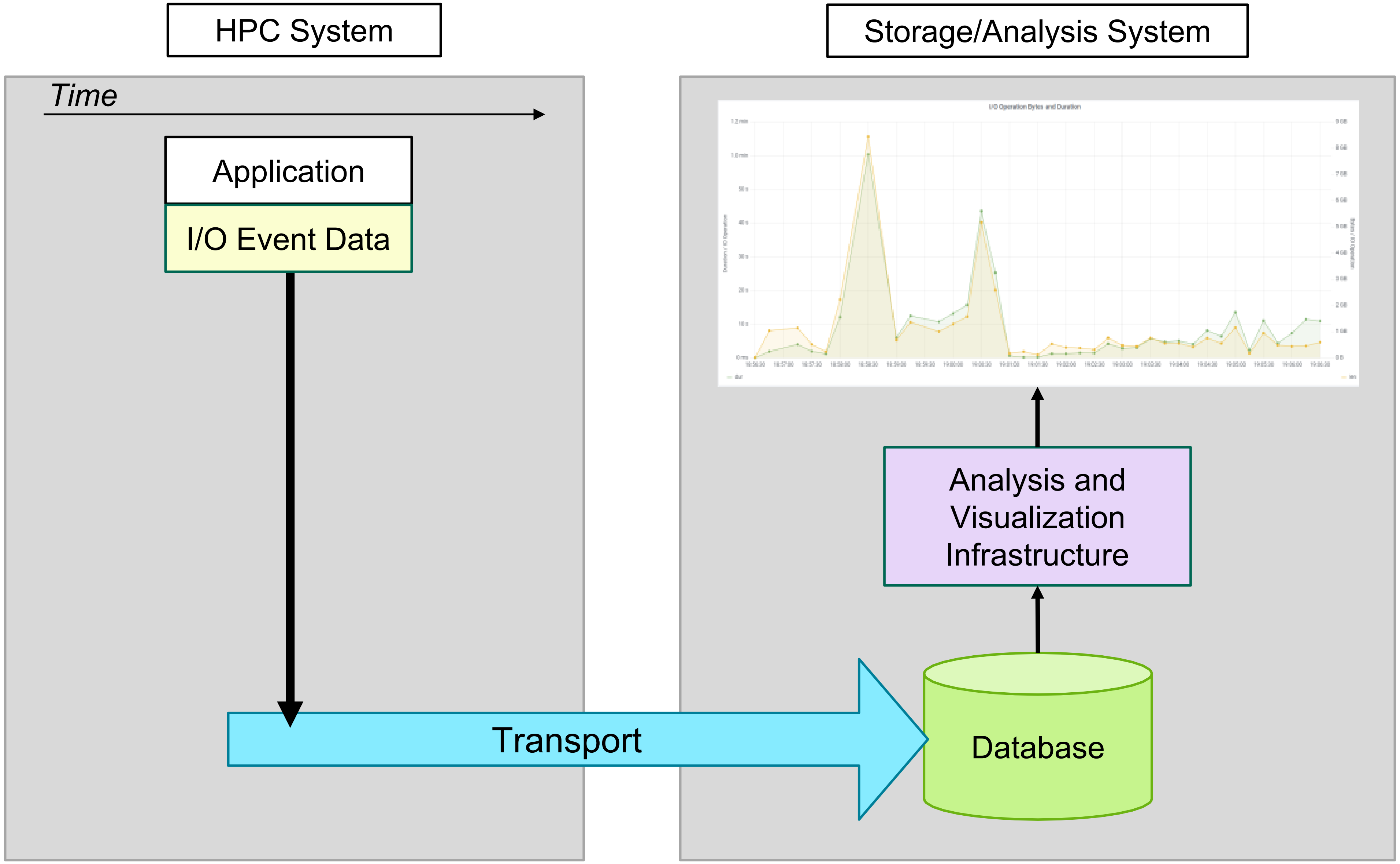
- The I/O application data is used for *post-run* analysis.
- Possible to get a set timeseries during an application run.
  - Unable to see what happens before or after this set of data.

Other collaborators (e.g. Northeastern University) have found a full set of timeseries data will benefit their studies.

Developed a tool that collects and provides runtime access to I/O event timeseries data.

Results show the success of capturing runtime I/O behavior and patterns.

Approach


HPC System
Storage/Analysis System
Time
Application
I/O Event Data
I/O Operation Bytes and Duration
Analysis and Visualization Infrastructure
Transport
Database

Approach - Continued

This tool collects runtime data via ***absolute timestamps.*** The runtime timeseries data will the allow further insights into the I/O behavior:

- Identify the **root cause(s) of I/O performance variability**.
- Detect **when an I/O performance variability occurs** during application runtime.
- Able to **quickly detect and address** any operational issues.
- Reveal **correlations between I/O performance variability and system behavior**.
- Investigate any **irregular or inconsistent** timeseries related analyses.
- Produce **new and meaningful** analyses and visualizations.

I/O event data will be collected, stored, analyzed and visualized with the **Darshan LDMS Integration -** A framework that integrates multiple tools to provide low-latency monitoring of I/O event data during runtime:

1. Darshan – A lightweight I/O characterization tool that transparently captures application I/O behavior from HPC applications with minimal overhead.
   - Generates a binary file of I/O traces post-run.
2. **L**ightweight **D**istributed **M**etric **S**ervice (**LDMS**) – A low-overhead production monitoring system that can run on HPC machines.
   - Capability to collect, transport, aggregate and store timeseries data during runtime.
3. An Analysis and Visualization Infrastructure for LDMS
   - Demonstrates further insights into I/O behavior through analysis modules and a web interface.

Integration: Darshan

Darshan (lightweight I/O characterization tool) data is used to tune I/O behavior of HPC applications for increased scientific productivity or to gain insight into trends in large-scale computing systems.

Provides detailed statistics about file accesses from MPI and non-MPI applications with the following modules:

- POSIX, MPI-IO, STDIO, and many more.
- **Darshan eXtended Tracing (DXT):** Provides a more detailed profiling of I/O software stacks such as kernel I/O traces and systematic analysis on the I/O behavior of applications.
  - Ex: Start and end time of read and write operations for MPIIO and POSIX.

Two main parts:

- **darshan-runtime:** The instrumentation portion of the Darshan characterization tool. Intent includes the collection of I/O characterization information of MPI & non-MPI applications.
  - Produces a binary file containing the I/O traces.
- **darshan-util:** A collection of tools for parsing and summarizing log files produced by Darshan instrumentation.
  - Darshan log files are platform-independent.

Darshan uses absolute timestamps for reporting time related I/O event data

- Does not report the raw absolute timestamps due to memory limitations on host node.
- ***Modifications were made to expose this.***

This framework utilizes this tool to continuously collect I/O event data and absolute timestamp during application runtime. This data is then published and transported via LDMS.

Integration: LDMS

The **LDMS** monitoring system collects data via samplers and plugins:

- Sampler: Type of daemon that collects the data.
- Plugin: Determines the kind of data collected/sampled, aggregated or stored.

Provides ***absolute timestamp*** view of system conditions through multi-hop aggregation.

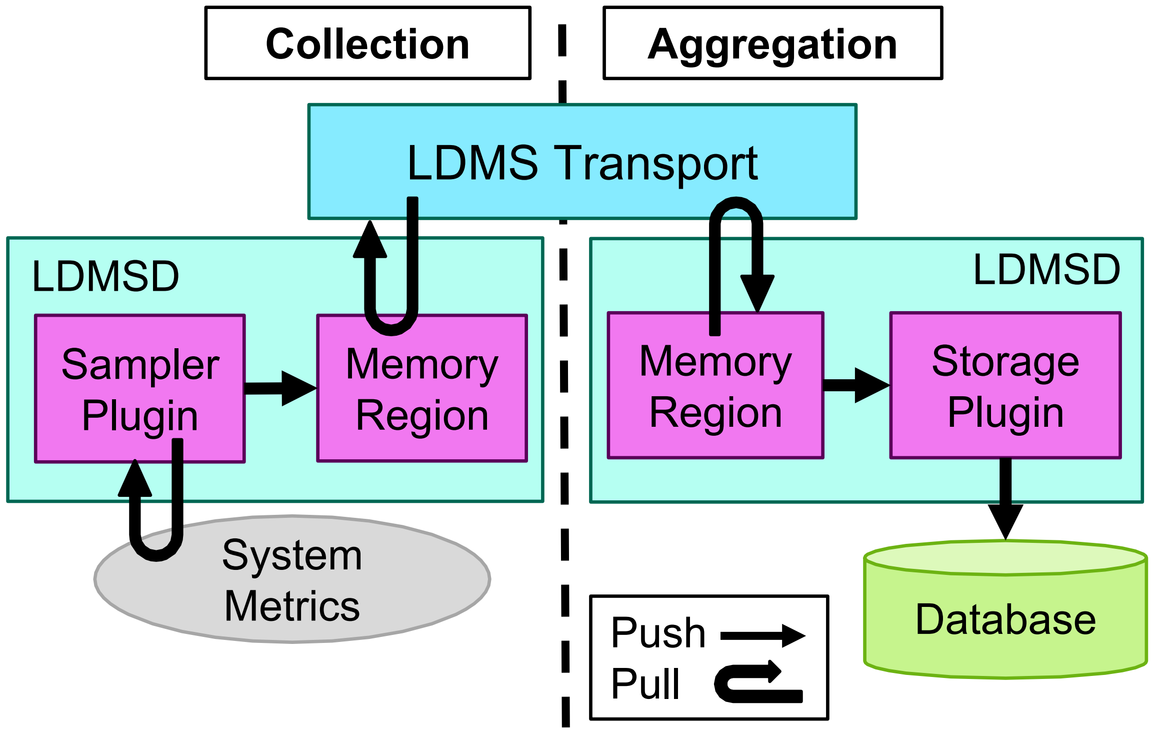
- Intermediate aggregators (transport) and head node aggregators (storage).
- Does not take up host memory when collecting timestamped data.

Darshan LDMS Integration ***leverages*** LDMS transport functionality for I/O data injection.

*Darshan LDMS Integration utilizes an LDMS functionality consisting of a **push-based** method.*

- Requires a *push-based* method to reduce memory consumed and data loss on the node.

Collection and transport process for system data.



Integration: LDMS Streams

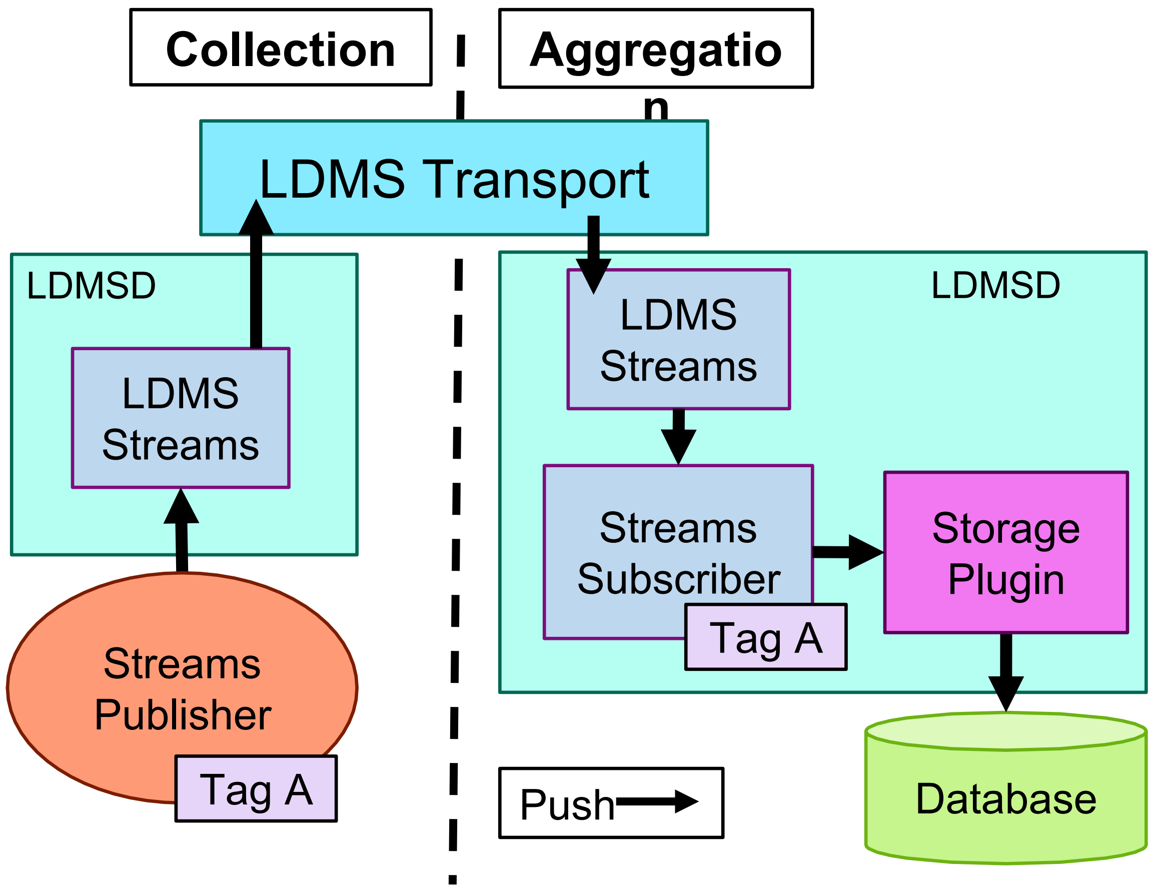
The *push-based* method for I/O event data injection is achieved through the LDMS Streams API.

LDMS Streams API - LDMS Functionality that allows for the aggregation of event-based application data.

- **LDMS Streams:** A publish-subscribe bus capability.
- Intended for publishing and subscribing to an LDMS streams tag (e.g. Tag A) via publish API call.
- Tag needs to be specified in LDMS daemons and plugins to publish and receive LDMS Streams data with matching tags.
- ***Enhanced to support the collection of I/O event data.***

I/O event data is stored to a database designed to hold large volumes of data.

Collection and transport process for event data.



Integration: Storage Database

The **D**istributed **S**calable **O**bject **S**tore (**DSOS**) - A storage database designed to manage large volumes of HPC data efficiently.

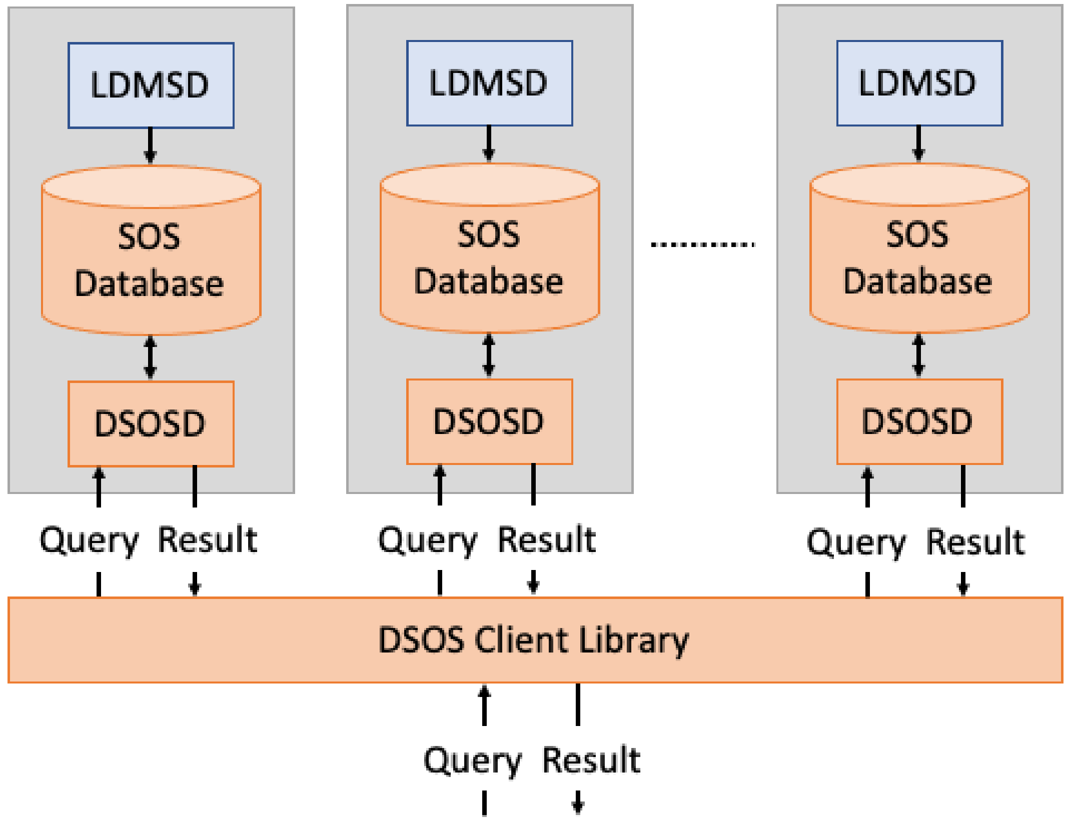
- Allows for interaction via a command line interface.
  - Fast query testing and data examination.
- Provides **scalable data ingest** and ability to **query and store large volumes of data.**

DSOS cluster - Consists of multiple instances of DSOS daemons, *dsosd*, that run on multiple storage servers on a single cluster.

Queried results are sorted based on joint indices selected by user to provide different query performances

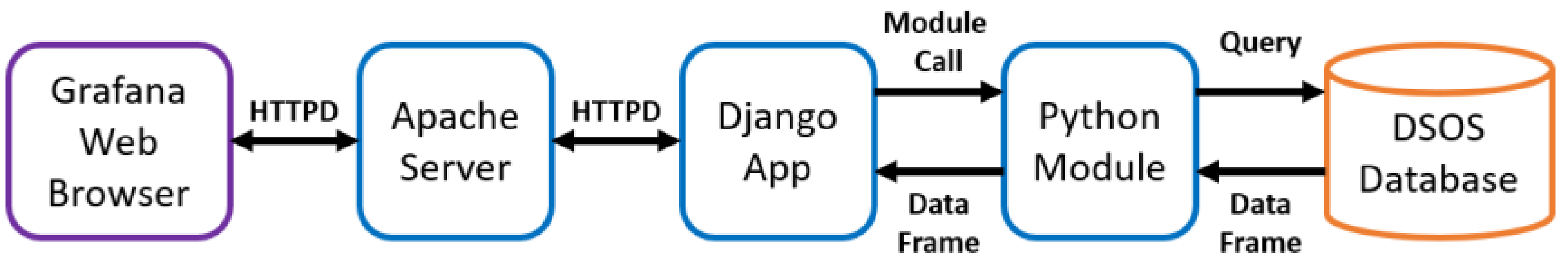
- For published LDMS Streams I/O event data, the indices consist of combinations of job ID, rank and timestamp.

This framework transports I/O event data, via LDMS Transport, to the DSOS database. It is then queried with the analysis and visualization infrastructure.



Sort by joint indices example: "*job_rank_time***"** where data will by ordered by job, rank then timestamp and search the data by specific rank within a specific job over time.

Integration: Analysis and Visualization Infrastructure



HPC Web Services – An analysis and visualization infrastructure for LDMS that integrates an open-source web application, Grafana, with a custom back-end web framework (Django).

- Calls python modules for analysis and visualization of HPC data.
- *Used to query, analyze (Python analysis modules) and view (Grafana) sorted I/O event data in real time.*

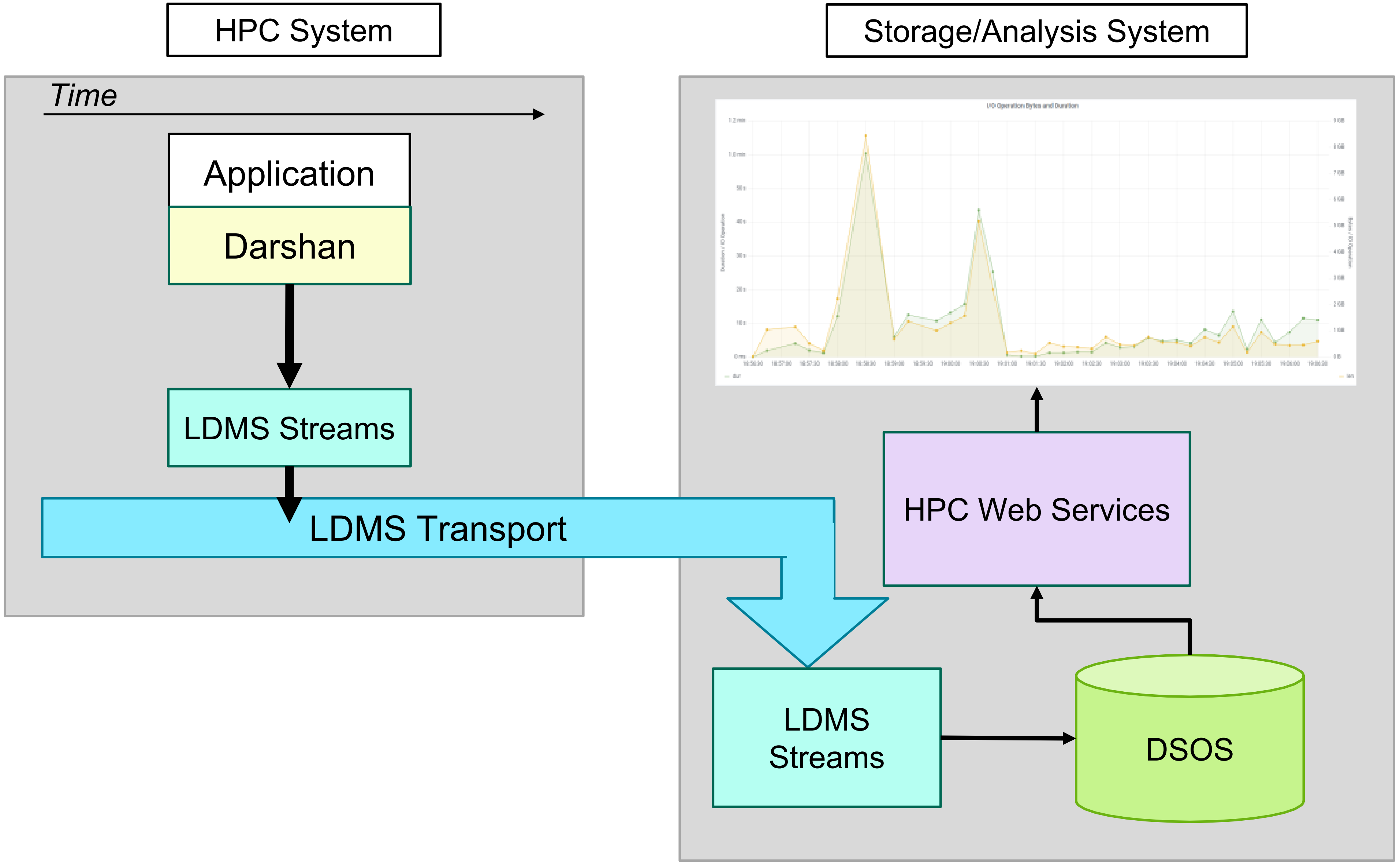
Grafana – An opensource visualization tool tailored towards time-series data from various database sources.

- Provides charts, graphs, tables, etc. for viewing and analyzing queried data in real time.
- Enables wide variety of visualization options for the data and allows users to save and share those visualizations to others.

Python Analysis Module – Where the queried data from DSOS to the Grafana dashboard is transformed, calculated, analyzed, etc. and then returned back to Grafana.

- Uses a custom DSOS-Grafana API specified in a Grafana query.
- Many analysis modules already exist for other types of system and application data.
- *Allow for complex calculations, transformation and aggregations of I/O event data*.

Integration: Overview



Integration: Darshan-LDMS Connector

Q: How is data sent from Darshan to an LDMS daemon?

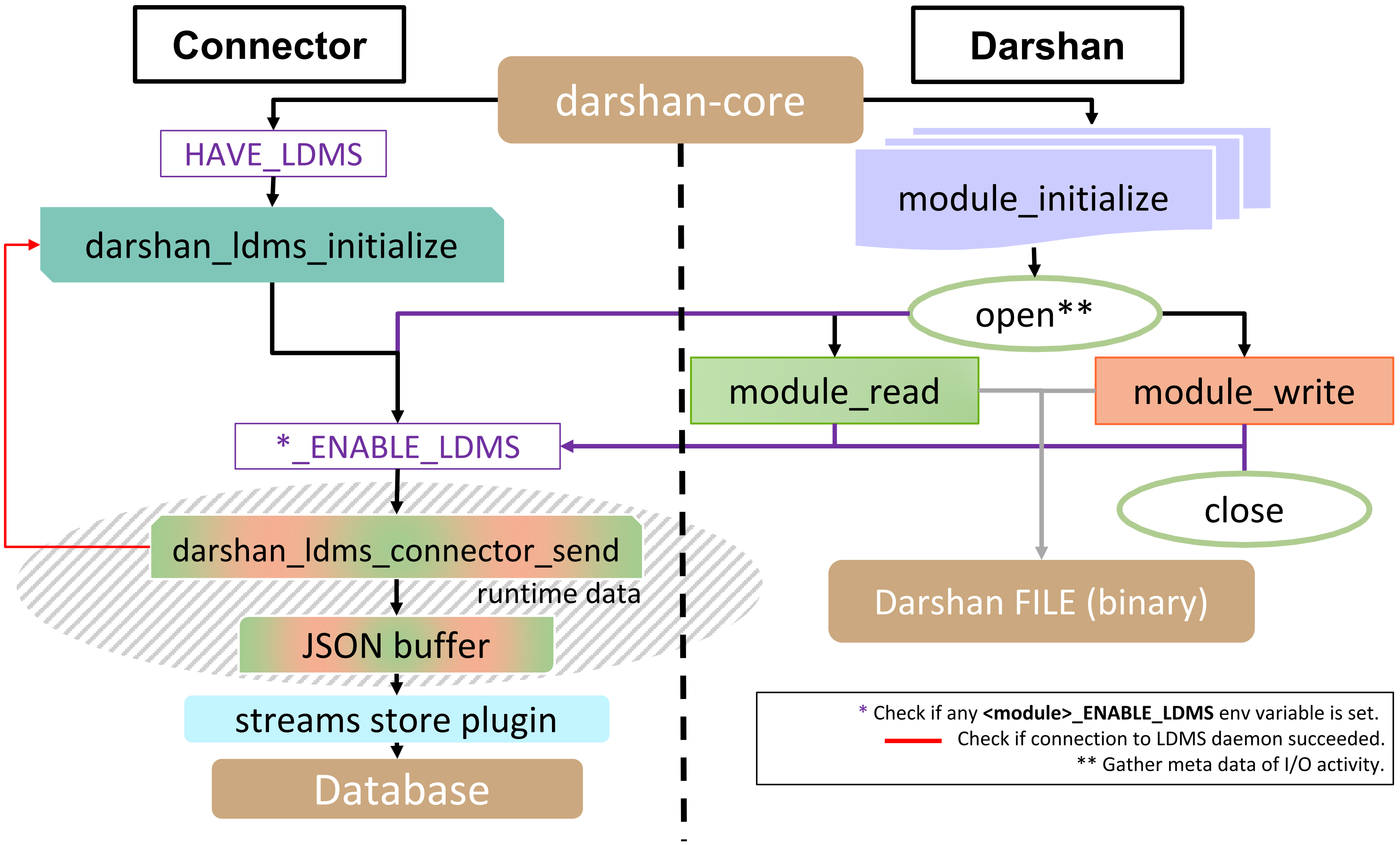
A: Darshan-LDMS Connector

A Darshan-LDMS Integration functionality that collects both DXT and Darshan's original I/O tracing and optionally *publishes* a message in **JSON** format to the **LDMS Streams interface**.

- Current stages of connector only collect a subset of this data due to the large volume of metrics Darshan uses for I/O tracing and post-processing calculations.
- Uses a single unique LDMS stream tag for the Darshan I/O data.
- Uses environment variables to establish a connection to an LDMS daemon.
- Little to no interference with Darshan's original program.
- Absolute timestamp is included in JSON message with the name "timestamp".

Overview of the Darshan-LDMS connector, configurations and defined metrics will be covered in the following slides.

Integration: Darshan-LDMS Connector



* Check if any **<module>_ENABLE_LDMS** env variable is set.

🟥 Check if connection to LDMS daemon succeeded.

** Gather meta data of I/O activity.

Integration: Configuration

Darshan – Easy to use and initialize

- Only need to set the LD_PRELOAD environment variable to the full path of the Darshan shared library before executing an application.
- Ex: ***srun -n 4 --export=LD_PRELOAD=<path-to-shared-library> <application>***

LDMS – Simply set the following list of environment variables to connect to an LDMS streams daemon and published I/O event data:

- *MODULE*_ENABLE_LDMS -> Set to publish *MODULE* module data to LDMS daemon.
- DARSHAN_LDMS_PORT -> Port number that the LDMS daemon is listening on.
- DARSHAN_LDMS_HOSTNAME -> Hostname that the LDMS daemon is running on.
- DARSHAN_LDMS_XPRT -> Type of transport the LDMS daemon is listening on.
- DARSHAN_LDMS_STREAM -> *Name tag (identifier)* of the stream.

Darshan-LDMS Integration – Easy to build and install Darshan against LDMS library:

- Only need to include ***--with-ldms=<path-to-LDMS-install>*** to the configuration line.

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)

Integration: Summary

Data Collection

- **Darshan** to collect timeseries I/O event data during application runtime.
- **Darshan-LDMS Connector** to collect Darshan I/O data and publish to LDMS Streams interface.
- Configuration and setup for Darshan and LDMS is quick and easy.
- Various I/O event metrics are collected from Darshan’s I/O tracing.

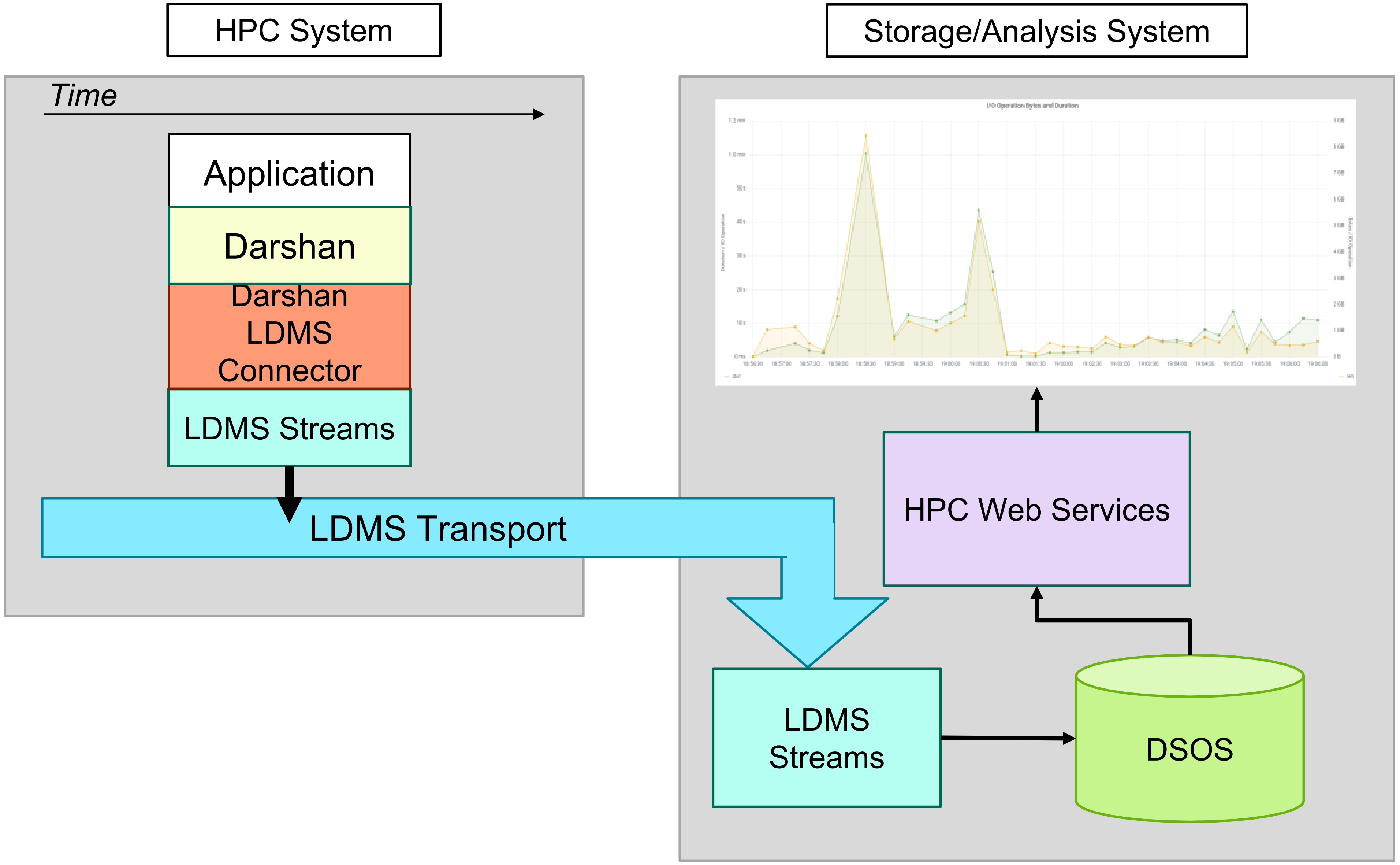
Transport and Storage

- **LDMS** to provide and transport live run time data feed about application I/O events.
- **DSOS** to store and query large volumes of I/O data generated on a production HPC system.

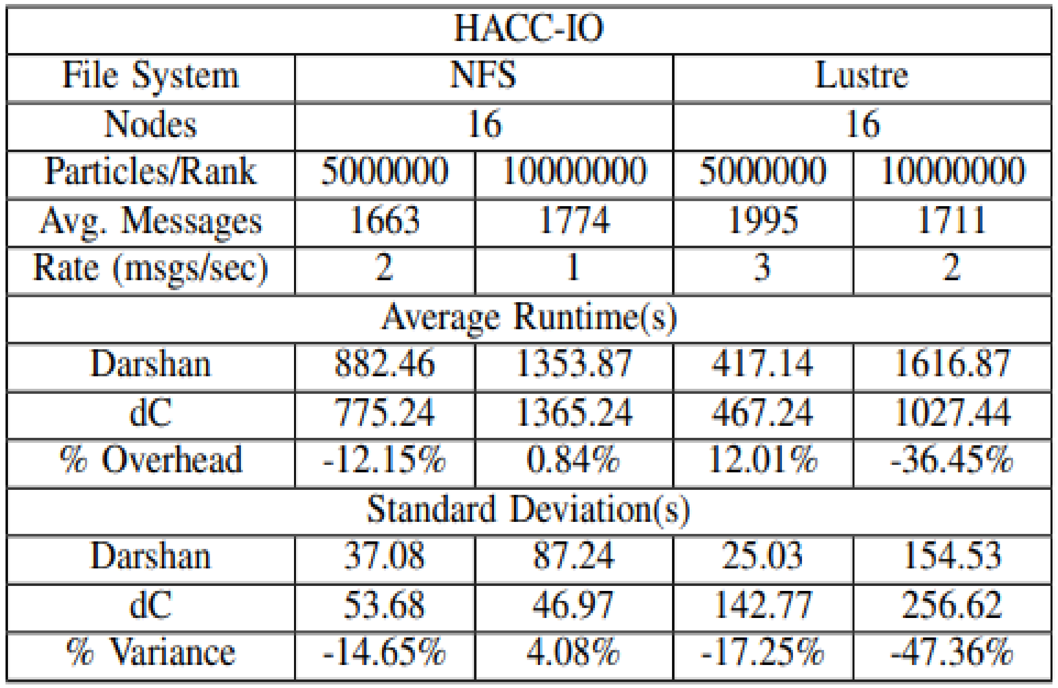
Analysis and Visualization

- **HPC Web Services** to analyze and present run time I/O data.
  - Able to demonstrate further insights into the application I/O behavior, patterns, etc.
  - Provides the ability to identify correlations between I/O performance and system behavior.

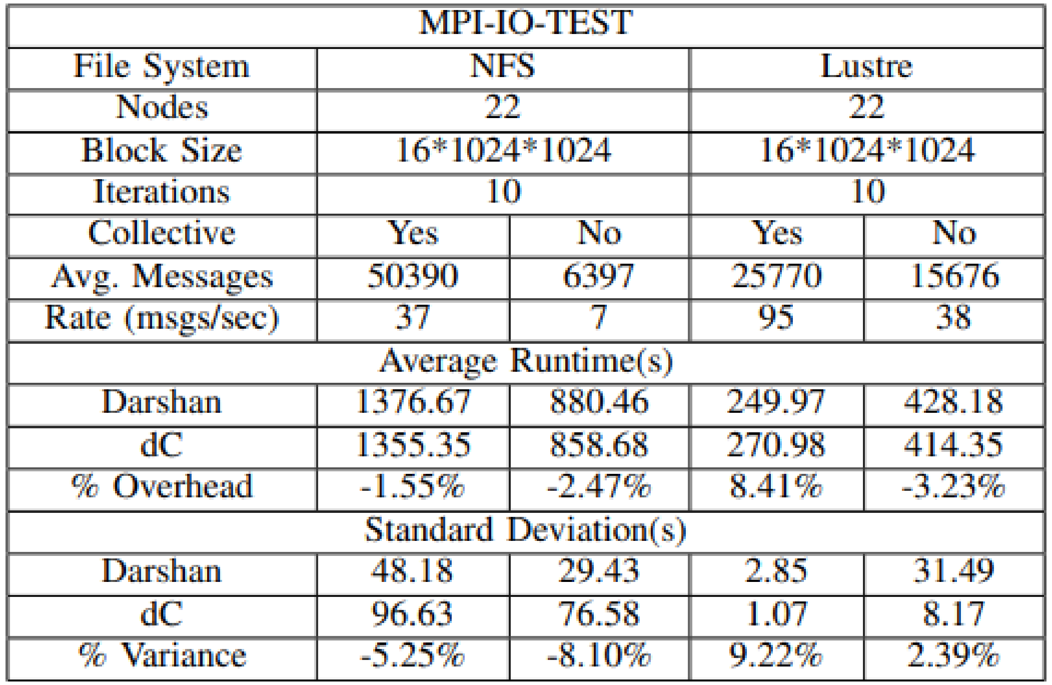
Integration: Summary



Results: LDMS Overhead



HACC-IO				
File System	NFS		Lustre	
Nodes	16		16	
Particles/Rank	5000000	10000000	5000000	10000000
Avg. Messages	1663	1774	1995	1711
Rate (msgs/sec)	2	1	3	2
Average Runtime(s)				
Darshan	882.46	1353.87	417.14	1616.87
dC	775.24	1365.24	467.24	1027.44
% Overhead	-12.15%	0.84%	12.01%	-36.45%
Standard Deviation(s)				
Darshan	37.08	87.24	25.03	154.53
dC	53.68	46.97	142.77	256.62
% Variance	-14.65%	4.08%	-17.25%	-47.36%



MPI-IO-TEST				
File System	NFS		Lustre	
Nodes	22		22	
Block Size	16*1024*1024		16*1024*1024	
Iterations	10		10	
Collective	Yes	No	Yes	No
Avg. Messages	50390	6397	25770	15676
Rate (msgs/sec)	37	7	95	38
Average Runtime(s)				
Darshan	1376.67	880.46	249.97	428.18
dC	1355.35	858.68	270.98	414.35
% Overhead	-1.55%	-2.47%	8.41%	-3.23%
Standard Deviation(s)				
Darshan	48.18	29.43	2.85	31.49
dC	96.63	76.58	1.07	8.17
% Variance	-5.25%	-8.10%	9.22%	2.39%

Overview:

- Framework is evaluated by analyzing sampled I/O data captured from two HPC applications
  - MPI-IO-TEST: Darshan utility to test MPI I/O performance on HPC machines.
  - HACC-IO: I/O Proxy for the Hardware Accelerated Cosmology Code.
- Experiments tested on Lustre and NFS file systems with various configurations.
- 5 runs for each configuration for a total of 40 job submissions.

Key Takeaways:

- Decrease in overall runtime time in 3 out of 4 experiments for both applications
  - Most likely due to file system performance issues as tests were ran 1-2 weeks apart
- Variance statistically significant to determine the overhead calculations are inconclusive.

Results: Grafana Visualization

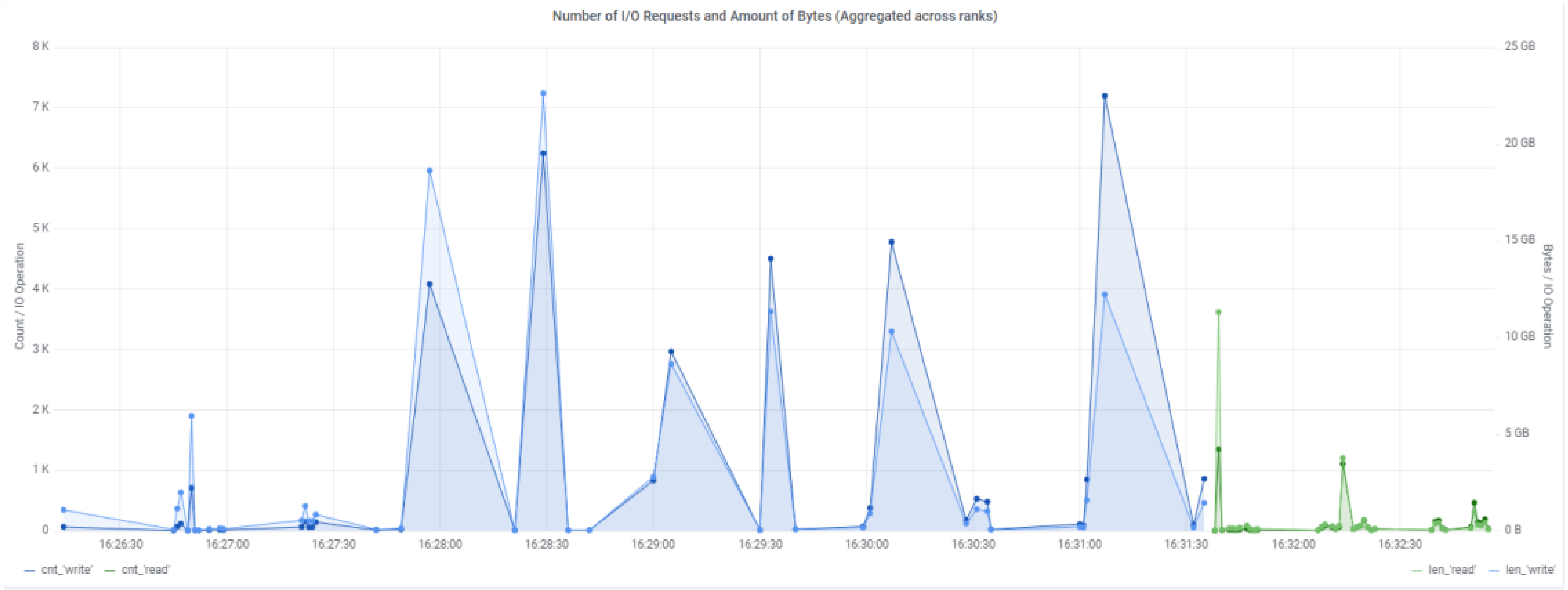
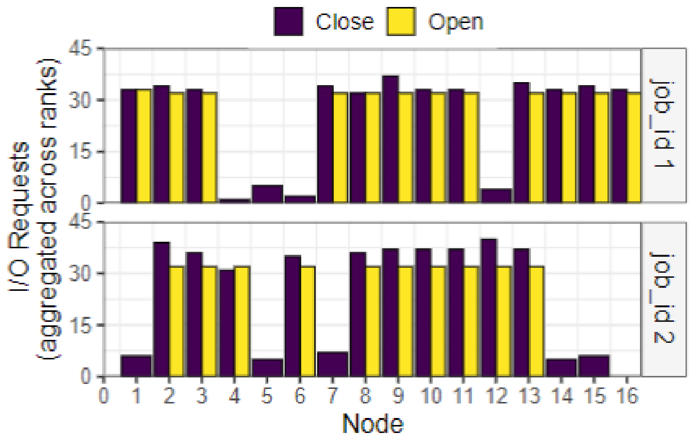


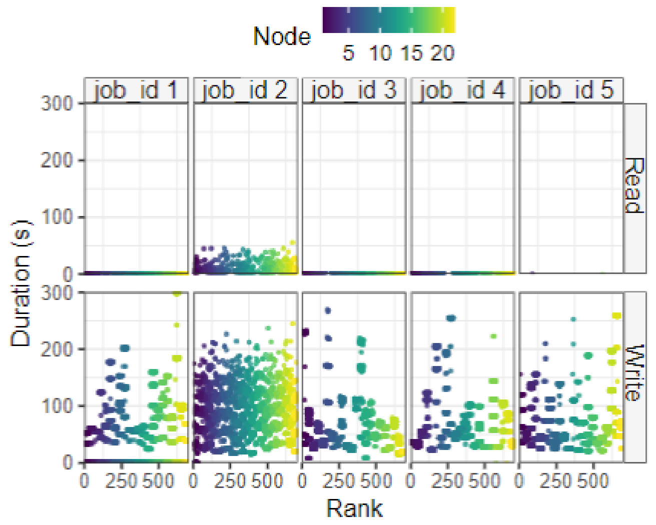
Figure above presents a Grafana visualization of MPI-IO job_id 2 writes (blue) and reads (green) operations and number of bytes read/written aggregated across all ranks.

- Uses the absolute timestamp metric collected with Darshan LDMS Integration.
- Grafana offers the interactive front-end view where users can easily filter though specific times and metric intervals.
- Representation using absolute timestamps facilitates the correlation of I/O performance congestion with system behavior monitoring.

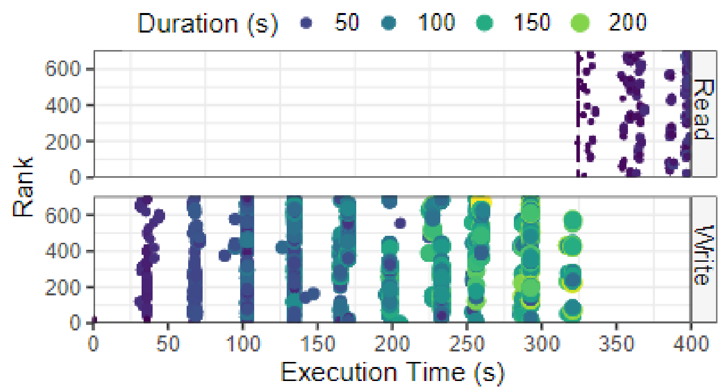
Results: Analysis



1. Post-Run Analysis of I/O event data: Number of I/O requests per node for close and open operations of 2 HACC-IO jobs. Can be produced with Darshan and framework.



2. Runtime Analyses of I/O event timeseries data: Duration of reads and writes per rank of 5 MPI-IO jobs (job_id 1-5) without collective operations. Only produced with framework.



3. Runtime Analyses of I/O event timeseries data:
Distribution of job_id 2 to display application I/O pattern and behavior over time. Only produced with framework.

Key Takeaways

- Variation is seen between two runs of the same experiment in the far left (1).
  - Possible data loss from connector and is currently under investigation.
- Darshan LDMS Integration makes is possible to create the meaningful analyses and visualizations (2 and 3).
  - Ability to see when an operation occurs, it's duration and variability between runs.
  - Allows for further insights into I/O behavior and patterns during application execution.
- Experiment on the far left shows the aggregate I/O behavior which can be created with Darshan alone (e.g. post run data).
  - Unable to analyze when an operation occurs and duration of each operation.
  - Prevents further insights into I/O behavior and patterns during application execution.

Related Work

PASSION Runtime Library for parallel I/O proposed by Syracuse University

- Optimizes I/O intensive applications through Data Prefetching and Data Sieving.

IOPin: Runtime Profiling of Parallel I/O in HPC Systems

- Proposes dynamic instrumentation to show the interactions from a parallel I/O application to the file system.

Design and Implementation of a Parallel I/O Runtime System for Irregular Applications

- Proposes two different collective I/O techniques for improving I/O performance.

Other Open-Source Tools

- iostat – Linux command that collects and reports I/O device statistics and loads between physical disks.
- ioprof – Provides insights into I/O workloads.

How Is Darshan LDMS Integration Different?

- This work ***leverages and enhances*** existing applications and tools
  - Integrates LDMS’s timestamped data collection and storage capabilities with Darshan.
  - Integrates python analysis modules and open-source web application for the analyses and visualizations.
- Provides extensive I/O tracing (i.e. statistics of individual I/O operations) of ***applications***.
- Implemented a database to allow for efficient queries of large volumes of data.

Conclusions & Future Work

Overview:

- This framework utilized Darshan’s I/O event data tracing and LDMS Streams interface to capture timeseries data of I/O events during application runtime.
- Provided the capability to query, analyze and view the timeseries data via DSOS and HPC web services.
- Demonstrated further insights into application I/O behaviors and patterns with visualizations and analyses.

Completed work:

- Ability to enable or disable the LDMS Streams functionality for specific Darshan modules.
- Establish a simple connection from Darshan to LDMS via the LDMS Streams API to publish runtime data generated by Darshan.
- Completed multiple I/O application runs with the Darshan LDMS Integration and successfully collected runtime timeseries data of the I/O events.
  - Full timeseries data can be collected throughout entire application execution.

Future work:

- Characterize the Darshan LDMS connector overhead with a variety of I/O intensive applications.
- Further validate any data loss by comparing Darshan’s output log files to the timeseries data.
- Utilize this framework to investigate impact of file system performance on I/O intensive applications.
- Incorporate existing analyses and create new analyses to present in a Grafana dashboard.

QUESTIONS?