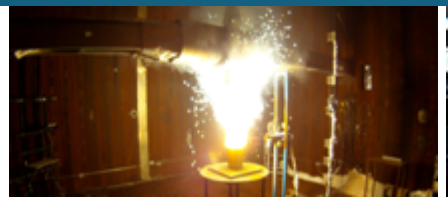
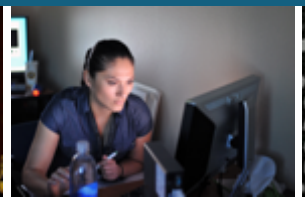
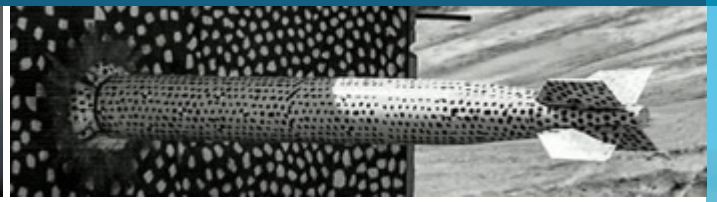
SAND2022-16974C

Sandia National Laboratories

# LDMS Streams Integration: For Run Time Diagnosis of HPC Applications







Sara Walton, SNL
Devesh Tiwari, Northeastern University
Ana Luisa V. Solórzano, Northeastern University
Omar Aaziz, SNL
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.

# LDMS Streams Integration: Motivation

Application 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 performance related problems
- Difficult to have a thorough understanding of throughput for system-specific behaviors and application performance of similar applications across a system

Timestamped time-series of performance profiling and I/O data at run time is needed to find errors, correlate with other events, and take actionable response.

Enable HPC users, system administrators, application developers, system architects and researches to gain further insights into application performance and interaction with system resources.

Developed tools for Darshan, Caliper and Kokkos that provides performance profiling and I/O event timeseries data to be analyzed in real time.

# LDMS Streams Integration: Benefits

## Darshan

- Uses Darshan's I/O tracing events to collect 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
- Captures read/write/close/open/flushes events for POSIX, MPI-IO, STDIO and HDF5 data

## Caliper

- Take advantage of Caliper framework to easily instrument application codes or keep the current instrumentations.
- Allow users to collect Caliper data parallel to other system metrics currently being collected by LDMS on production machines.

## Kokkos

- Leverage performance and progress instrumentation already implemented within Kokkos. User does not need specify what data to collect.
- No application changes or recompilation required.

Provide the ability to correlate application performance and I/O behavior to system monitoring data.

# LDMS Streams Integration: Overview

**LDMS Streams:** An LDMS message bus (publish/subscribe) capability that enables injection of application progress and performance information, during runtime, into its data stream.

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

**Caliper:** A program instrumentation and performance measurement framework that allows others to implement analysis capabilities (e.g. performance profiling, tracing, monitoring, and auto-tuning) into applications using Caliper's annotation API.

**Kokkos:** A C++ parallel programming ecosystem for performance portability across multi-core, many-core, and GPU node architectures. Provides abstractions of parallel execution of code and data management.

## Kokkos Data Collection

**kokkosConnector**

A Kokkos-LDMS functionality that utilizes **LDMS Streams** to collect Kokkos related data during runtime. *Kokkos sampler* controls the sampling rate and provides the option to sample data using a count-based push. *Publishes* **JSON** formatted message to **LDMS Streams interface**.

## Darshan Data Collection

**darshanConnector**

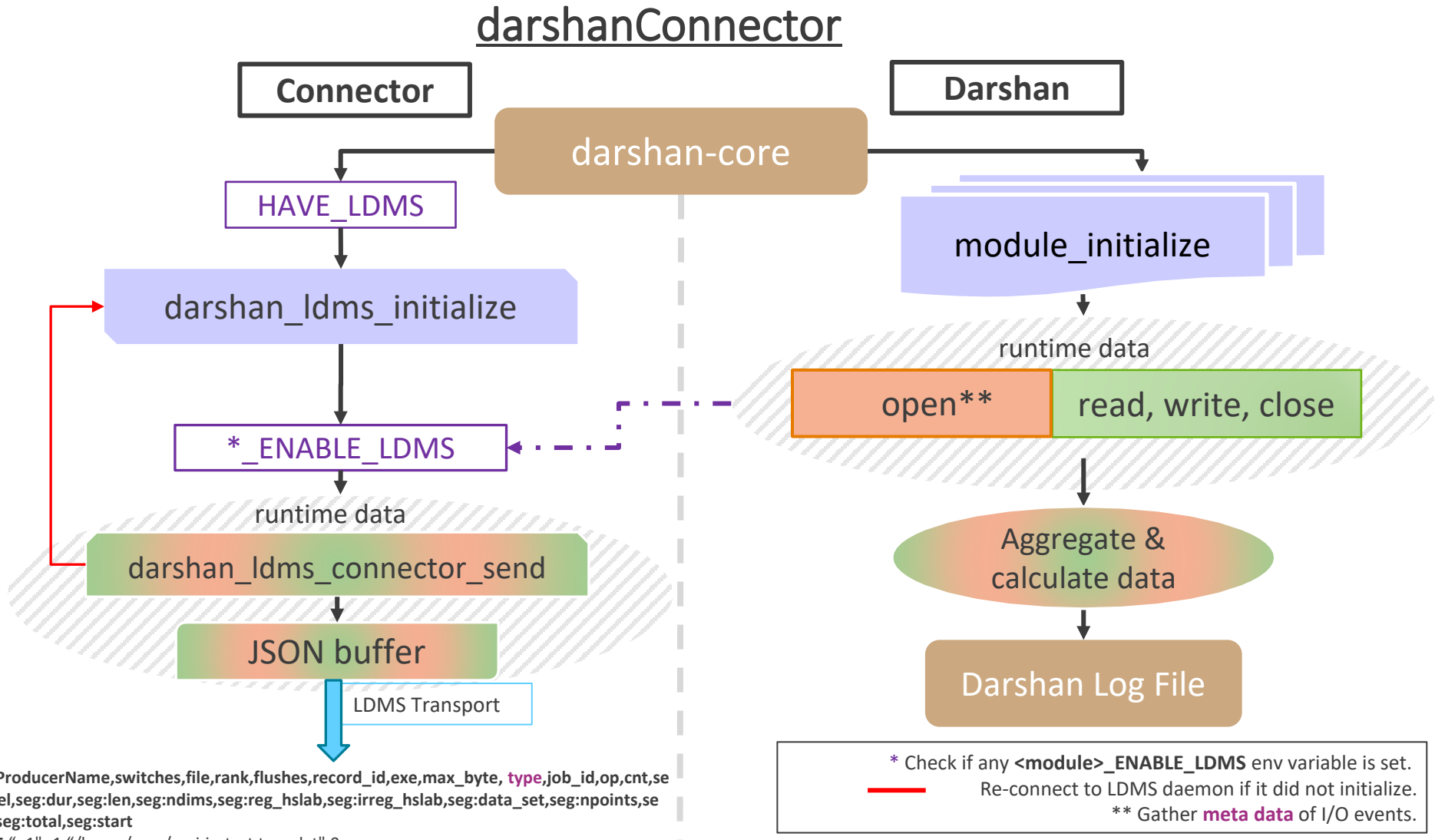
A Darshan-LDMS functionality that utilizes **LDMS Streams** to collect Darshan's original I/O tracing, Darshan's eXtended tracing (DXT) and absolute timestamp during runtime. *Publishes* **JSON** formatted message to the **LDMS Streams interface.**

## Caliper Data Collection

**caliperConnector**

A Caliper-LDMS functionality that utilizes **LDMS Streams** to collect Caliper related data and absolute timestamp during runtime. *Publishes* **JSON** formatted message to the **LDMS Streams interface.**

# LDMS Streams Integration: Darshan Data Injection




**#module,uid,ProducerName,switches,file,rank,flushes,record_id,exe,max_byte, type,job_id,op,cnt,seg:off,seg:pt_sel,seg:dur,seg:len,seg:ndims,seg:reg_hslab,seg:irreg_hslab,seg:data_set,seg:npoints,seg:timestamp,seg:total,seg:start**

"POSIX",**12345**,“n1",-1,“/home/user/mpi-io-test.tmp.dat",0,-1,7268045394536033567,“/home/user/mpi-io-test",-1,"**MET**",**11801265**,"open",1,-1,-1,0.000602,-1,-1,-1,-1,"N/A",-1,**1669666470.827069**,0.000602,0.329433

"POSIX",**12345**,“n1",-1,"N/A",0,-1,7268045394536033567,"N/A",-1,"**MOD**",**11801265**,"close",1,-1,-1,0.000177,-1,-1,-1,-1,"N/A",-1,**1669666470.827495**,0.000778,0.330284

# LDMS Streams Integration: Darshan Configuration

**Darshan** – Build and install Darshan-LDMS integrated code and set a single env variable

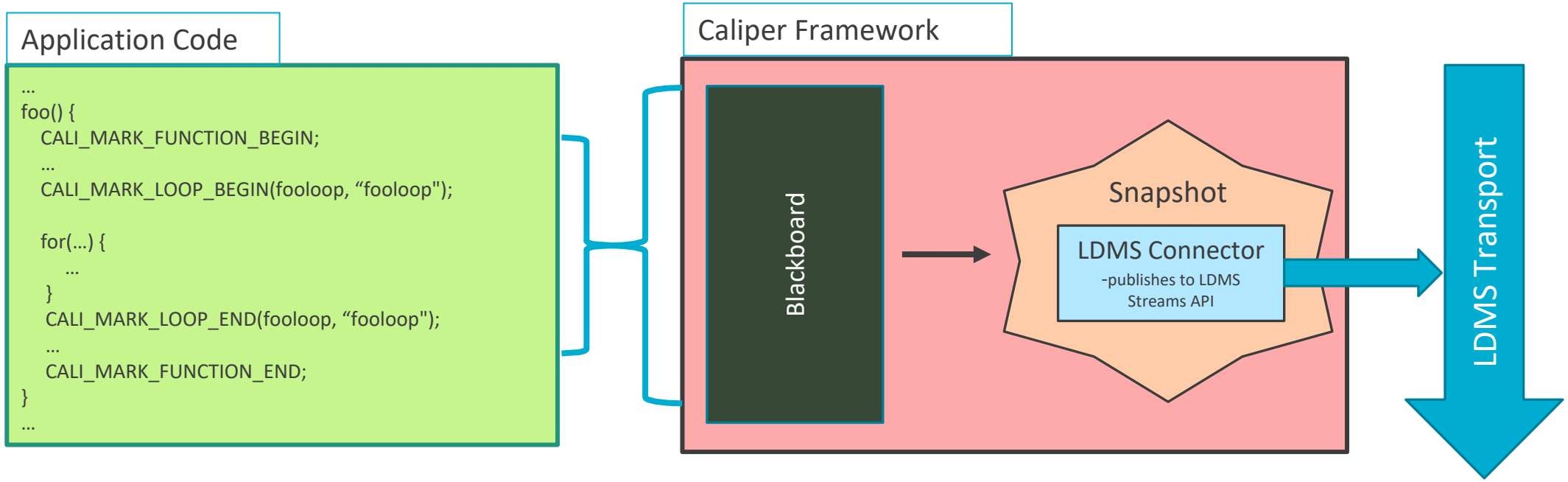
- Build against the ldms library by adding ***--with-ldms=<path-to-LDMS-install>*** to the configuration line.
- Set the **LD_PRELOAD** environment variable to the full path of the Darshan shared library before executing an application.
  - **LD_PRELOAD** = *<absolute-path>/**libdarshan.so*** **OR** *srun -n 4 --export=LD_PRELOAD=<absolute-path>**/libdarshan.so** <application>*

**LDMS & Darshan** – 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_HOST** → 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.

# LDMS Streams Integration: Caliper Application Data Injection

## caliperConnector




- Caliper runtime measurement configuration controls the sampling rate in the snapshot triggers
- Target collection: user preference

{"job_id":11878171,"ProducerName":“n1","rank":0,"timestamp":1670373198.056455,"region":"init","time":33.172237 }
{"job_id":11878171,"ProducerName":"n1","rank":0,"timestamp":1670373198.056455,"region":"initialization","time":33.211929 }
{"job_id":11878171,"ProducerName":“n1","rank":0,"timestamp":1670373198.056455,"region":"main","time":44.147736 }
{"job_id":11878171,"ProducerName":“n1","rank":0,"timestamp":1670373203.556555,"region":"main","time":0.049086 }
{"job_id":11878171,"ProducerName":“n1","rank":0,"timestamp":1670373203.556555,"region":"run","time":0.049086 }

# LDMS Streams Integration: Caliper Configuration

**Caliper** – Build original Caliper program with the application and export a single env variable

- No modifications to Caliper's code or instrumentation were required to integrate LDMS.
- Point the LD_LIBRARY_PATH to Caliper's library
  - **LD_LIBRARY_PATH**= *$LD_LIBRARY_PATH:<path-to-caliper-installation>/**lib64***

**LDMS & Caliper** – Set the following list of caliper variables when executing a program:

- **CALI_LOOP_MONITOR_ITERATION_INTERVAL** → Collect measurements every n loop iterations
- **CALI_SERVICES_ENABLE** → Define which services will be combined to collect data
  - Enable *LDMS service* to collect data through LDMS by adding "ldms" to **CALI_SERVICES_ENABLE**
  - Enable *MPI service* to associate the MPI rank to the LDMS data by adding "mpi" to **CALI_SERVICES_ENABLE**

**EXAMPLE:**

**CALI_SERVICES_ENABLE**=*loop_monitor,mpi,ldms* **CALI_LOOP_MONITOR_ITERATION_INTERVAL**=*10* ./caliper_example.o 400

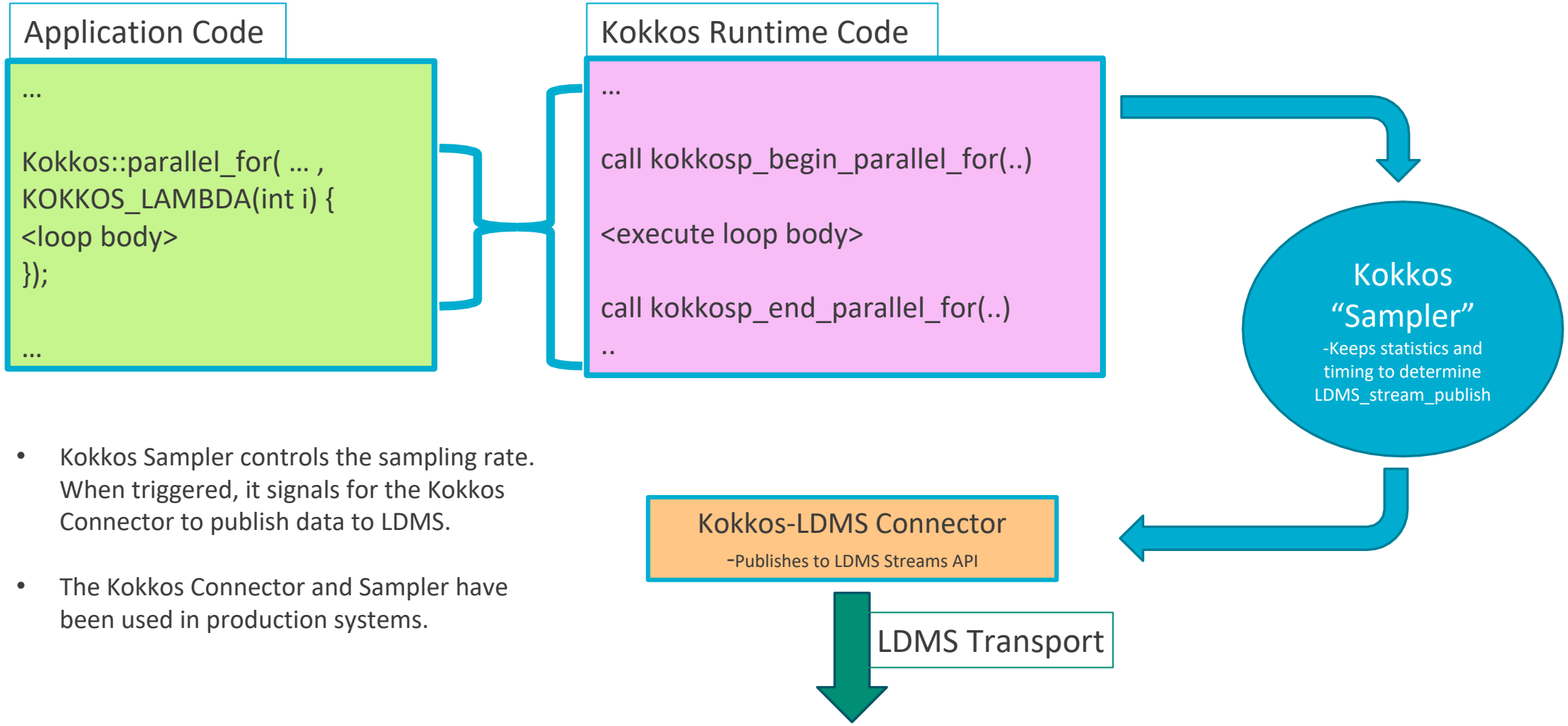
**OR**

export **CALI_SERVICES_ENABLE**=*loop_monitor,mpi,ldms*

export **CALI_LOOP_MONITOR_ITERATION_INTERVAL**=*10*

# LDMS Streams Integration: Kokkos to LDMS Publish

## kokkosConnector




- Kokkos Sampler controls the sampling rate. When triggered, it signals for the Kokkos Connector to publish data to LDMS.
- The Kokkos Connector and Sampler have been used in production systems.

```
#timestamp,job_id,rank,name,type,current_kernel_count,total_kernel_count,level,current_kernel_time,total_kernel_time
1627835612.086679,10195735,1,Kokkos::View::initialization [diagnostic:Solver Field:B_Field:temp],0,1218,57972687,0,0.000005,182.693422
1627835613.709526,10195735,1,TimeAverage::Continuous,0,24758,57972788,0,0.000006,182.693428
1627835616.787472,10195735,1,MigrateParticles::count,1,3540,57972889,0,0.000001,182.693430
1627835620.448333,10195735,1,SolverInterface::Apply Trivial BC,0,7512,57972990,0,0.000002,182.693432
```

# LDMS Integration: Kokkos Configuration

**Kokkos** – Build and install Kokkos-LDMS integrated code and export a single env variable

- Set the **KOKKOS_PROFILE_LIBRARY** environment variable to the full path of the *kokkos kernel* and *kokkos-ldms* sampler shared library files:
  - **KOKKOS_PROFILE_LIBRARY =** *<absolute-path>/**kp_sampler.sdh.so**; <absolute-path>/**kp_kernel_ldms.so**;*

**LDMS** – Export the following env variables for the LDMS library

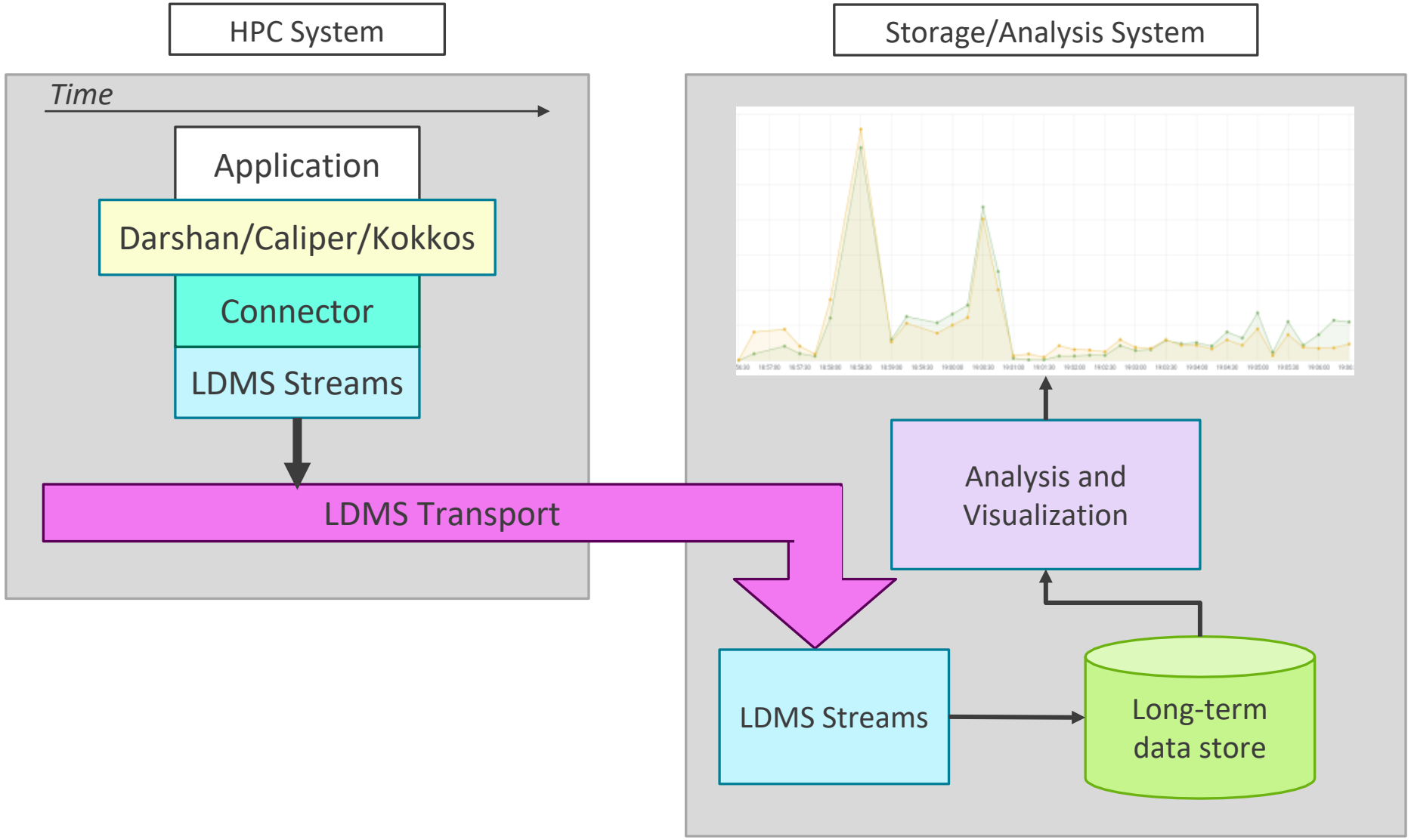
- **LD_LIBRARY_PATH**= *$LD_LIBRARY_PATH:<path-to-ldms-library>/**lib64***
- **PATH**= *$PATH:<path-to-ldms-bin>/**bin**:<path-to-ldms-sbin>/**sbin***

**LDMS & Kokkos** - Export the following list of environment variables to connect to an LDMS streams daemon and set the kokkos sampler rate:

- **KOKKOS_LDMS_PORT** → Port number that the LDMS daemon is listening on.
- **KOKKOS_LDMS_HOST** → Hostname that the LDMS daemon is running on.
- **KOKKOS_LDMS_XPRT** → Type of transport the LDMS daemon is listening on.
- **KOKKOS_LDMS_STREAM** → *Name tag (identifier)* of the stream.
- **KOKKOS_LDMS_SAMPLER_RATE**→ Optionally sample Kokkos data. Default is 101 (1%).
- **KOKKOS_LDMS_VERBOSE*** → Optionally display extra information. Default is disabled.

* Contains information about the kokkos sampler so users are aware of the rate at which they are collecting data (i.e. 1% vs 100%)

# LDMS Integration: Data Flow Diagram



# Next Steps

- Overhead Testing (Darshan/Caliper)
  - Experiments collecting Darshan/Caliper information only + LDMS-Integration
- Collect LDMS system monitoring information in the background (Darshan/Caliper)
  - Ability to correlate absolute timestamp with system health
- Stream Injection notification with throttling

## Darshan

- Validation against a varied set of applications:
  - Benchmarks - HACC-IO, MPI-IO, Scientific I/O kernels (e.g. FLASH, OpenPMD, E2E benchmarks, etc.)
  - “Real world” - Quantum Espresso, STDIO, etc.
- Investigate difference between NFS and Lustre filesystems

## Caliper

- Validation against various services with Caliper’s annotation API
  - Context annotations, measurement services and data processing services
- Preserve hierarchy of function executions

## Kokkos

- Utilize Kokkos Sampler feature on other applications of interest
- Investigate run time telemetry collection on non-Kokkos applications

# Questions?