Sandia National Laboratories

Exceptional service in the national interest

LDMS New Features for Deployment in Advanced Environments and Feedback for Operations

Jim Brandt (SNL)

CUG 2025

05/07/2025

SAND2025-05708C

ABSTRACT

Title: LDMS New Features for Deployment in Advanced Environments and Feedback for Operations

Abstract: The Lightweight Distributed Metric Service (LDMS) monitoring, transport, and analysis framework has been deployed on large-scale Cray and HPE systems for over a decade. Over that time its capabilities have improved dramatically. In this talk we provide updates on capabilities including deployment and management methods in bare metal, containerized, and cloud (including hybrid on+off prem) environments. We describe how LDMS is being used to collect application data concurrent with system data and how the low-latency availability of this data for analysis can be used for real-time data analysis and feedback in order to support efficient, resilient, and reliable system operations. Finally, we will describe current related research areas including **1) use of machine learning for modeling application and system behavioral characteristics and 2) use of new features in the bi-directional communication capability of LDMS to provide low-latency communication and feedback from a distributed analysis system to user, system, and application processes on disparate clusters and to inform data center orchestration decisions.**

On the product development, management, and distribution fronts we will present our planned improvements over the next year, release cadence, and package distribution methods including how we plan to stay in sync on HPE's CSM and HPCM releases.

MONITORING CHALLENGES IN EXTREME SCALE AND CLOUD ENVIRONMENTS

- Components always coming and going
  - Host names/addresses may change or be temporarily unavailable
- Many components
  - Need simple configuration
  - Snapshots enable full system analysis and better attribution
  - Gather important data without adversely impacting system performance
    - **Ensure or rate-limit on a per-data-source-basis**
    - **Might change over time (e.g., rate, application, processor type)**
- Large-scale bulk-synchronous applications sensitivity to OS noise
  - Minimize noise and jitter introduced by monitoring
- Many users/applications/problem domains/parameter space
  - Want to utilize ML to enable ~best resource mapping, anomaly detection, mitigation of performance-degrading conditions
  - Security – dynamic information separation (system and application)
  - Dynamic enabling of event sources and feedback loops

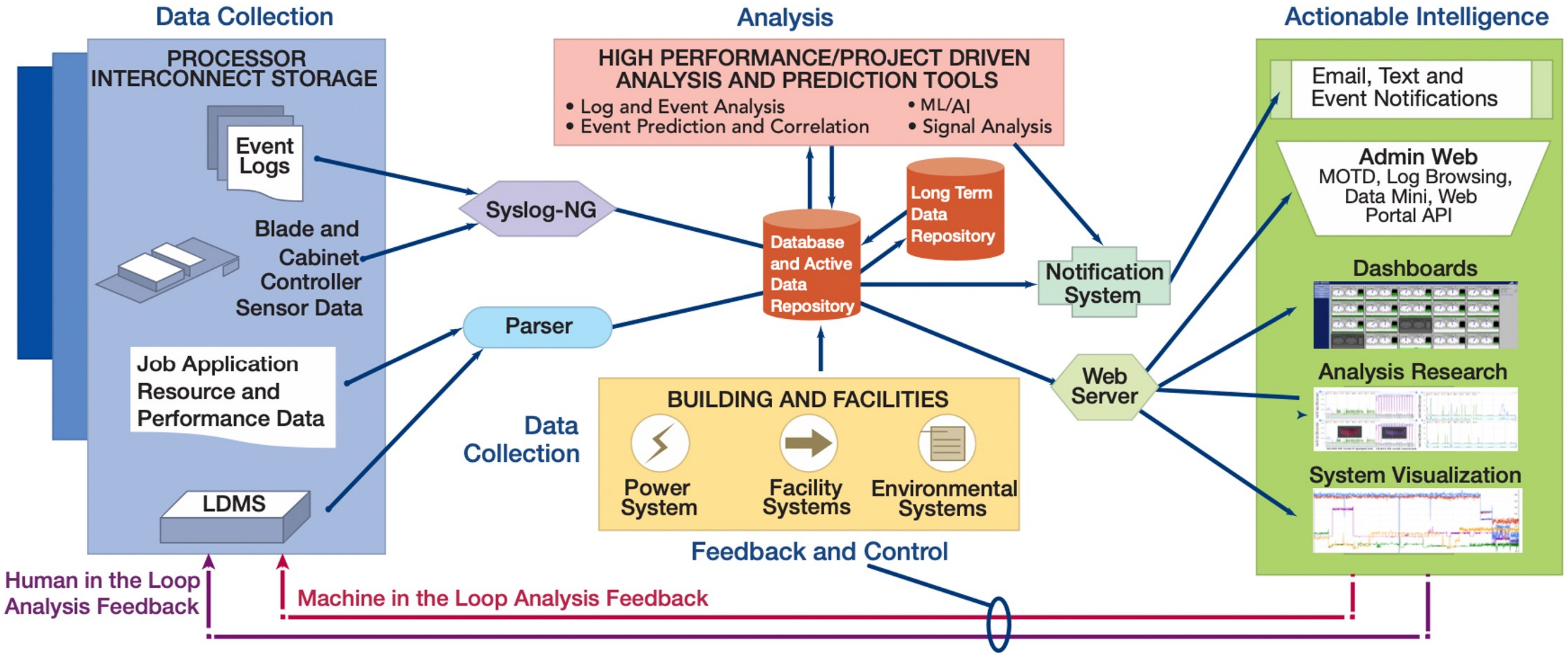
LDMS Background

LDMS IN A NUTSHELL

- Lightweight Distributed Metric Service (LDMS) provides a low-overhead data collection, transport, and storage capability designed for continuous monitoring supporting run time analyses and feedback
  - **Provide synchronized whole-system "snapshots"** of system metrics
  - Daemon-based data collection
    - Plugin architecture
      - Sample numeric data gathered from any source
    - Support for injection of event and application data
  - Transport and aggregate data
  - Store data
    - CSV, Avro Kafka, InfluxDB, Scalable Object Store
- Typical use cases for information "stored" by LDMS
  - Identify application execution behaviors
  - Run time discovery of abnormal application execution behaviors
  - Identify applications' memory (and other resource) utilization behaviors
  - Right-size job allocations based on resource utilization profiles
  - Identify network congestion
  - Identify heavy file system (e.g., Lustre) users
  - Identify baseline resource behaviors

LIGHTWEIGHT DISTRIBUTED METRIC SERVICE (LDMS)

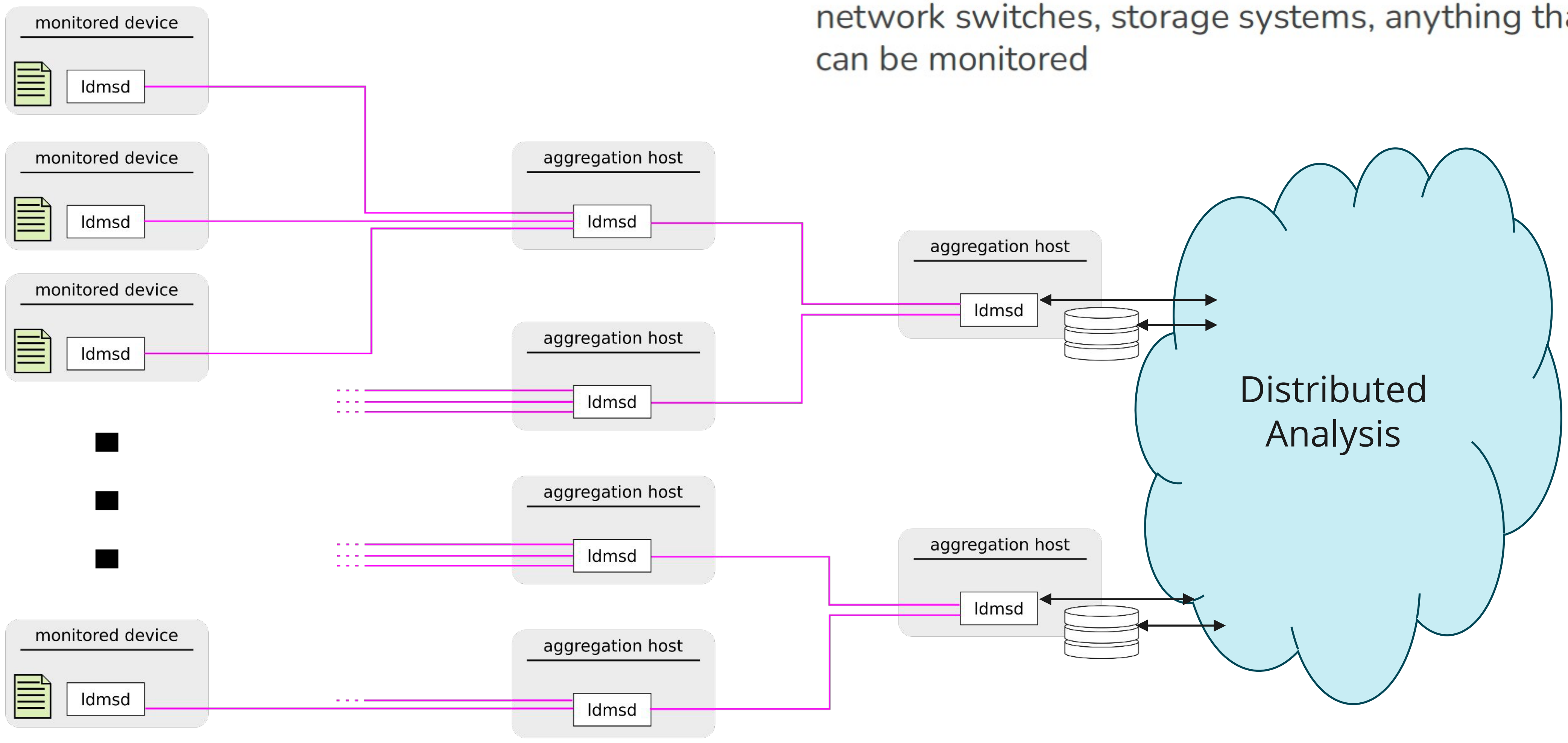
DESIGNED FOR GLOBAL COLLECTION OF HIGH-FIDELITY DATA TO SUPPORT RUN TIME ANALYSIS, AND FEEDBACK



Data Collection
PROCESSOR INTERCONNECT STORAGE
Event Logs
Blade and Cabinet Controller Sensor Data
Job Application Resource and Performance Data
LDMS
Syslog-NG
Parser
Analysis
HIGH PERFORMANCE/PROJECT DRIVEN ANALYSIS AND PREDICTION TOOLS
• Log and Event Analysis
• Event Prediction and Correlation
• ML/AI
• Signal Analysis
Database and Active Data Repository
Long Term Data Repository
Notification System
Web Server
Data Collection
BUILDING AND FACILITIES
Power System
Facility Systems
Environmental Systems
Feedback and Control
Actionable Intelligence
Email, Text and Event Notifications
Admin Web
MOTD, Log Browsing, Data Mini, Web Portal API
Dashboards
Analysis Research
System Visualization
Human in the Loop Analysis Feedback
Machine in the Loop Analysis Feedback

LDMS Deployment Overview

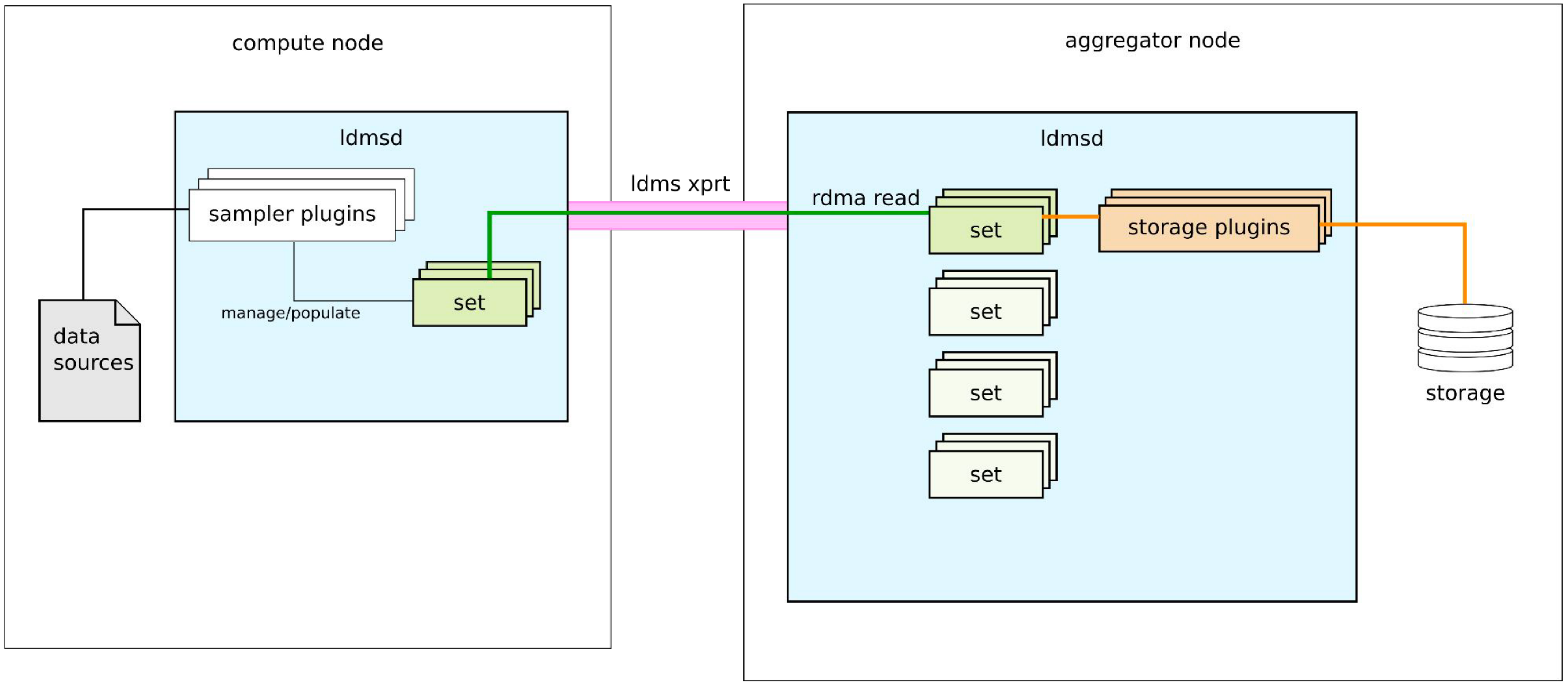
Monitored devices: compute nodes, non-compute nodes, network switches, storage systems, anything that can be monitored



~2000:1 fan-in depending on load

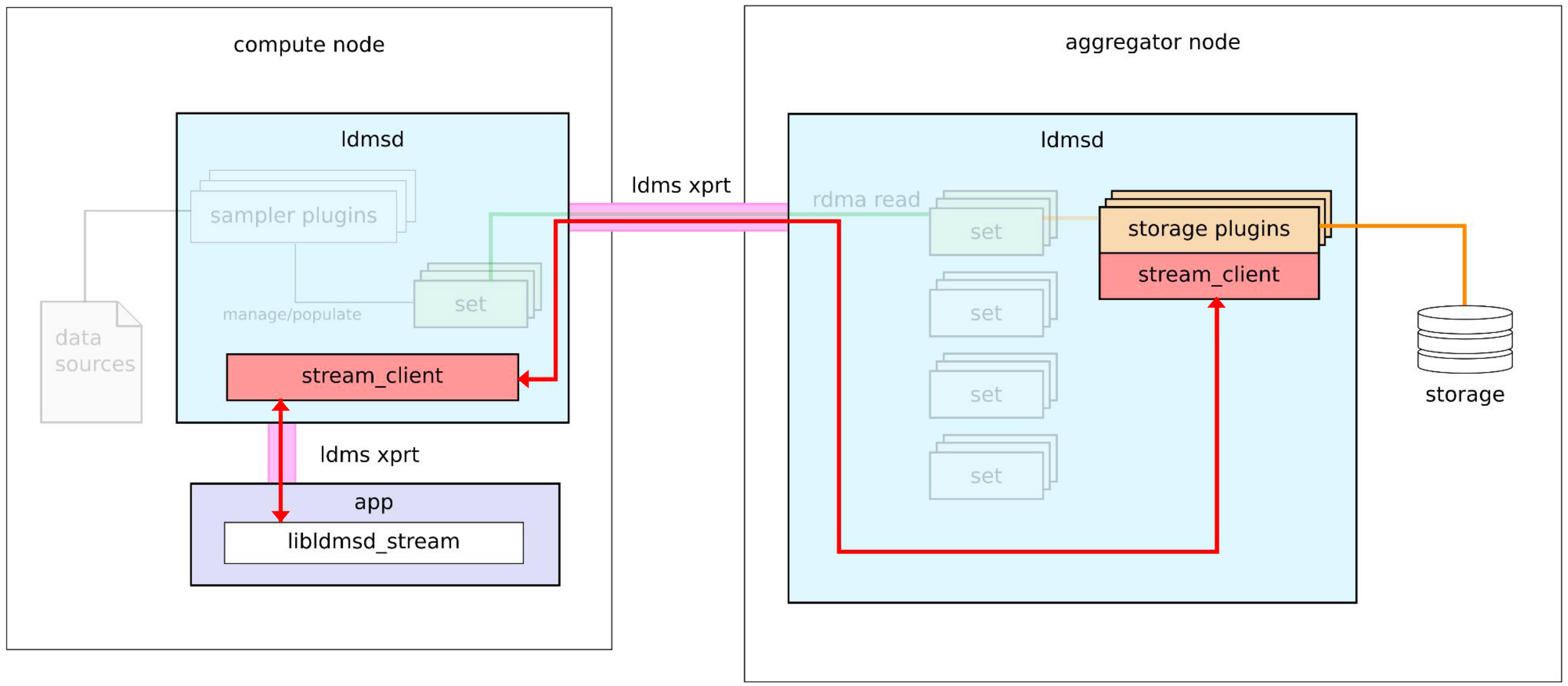
fan-out or fan-in depending on storage load

Transport Modes: Lightweight Regular and Synchronous Collection and Pull of System Metric Data

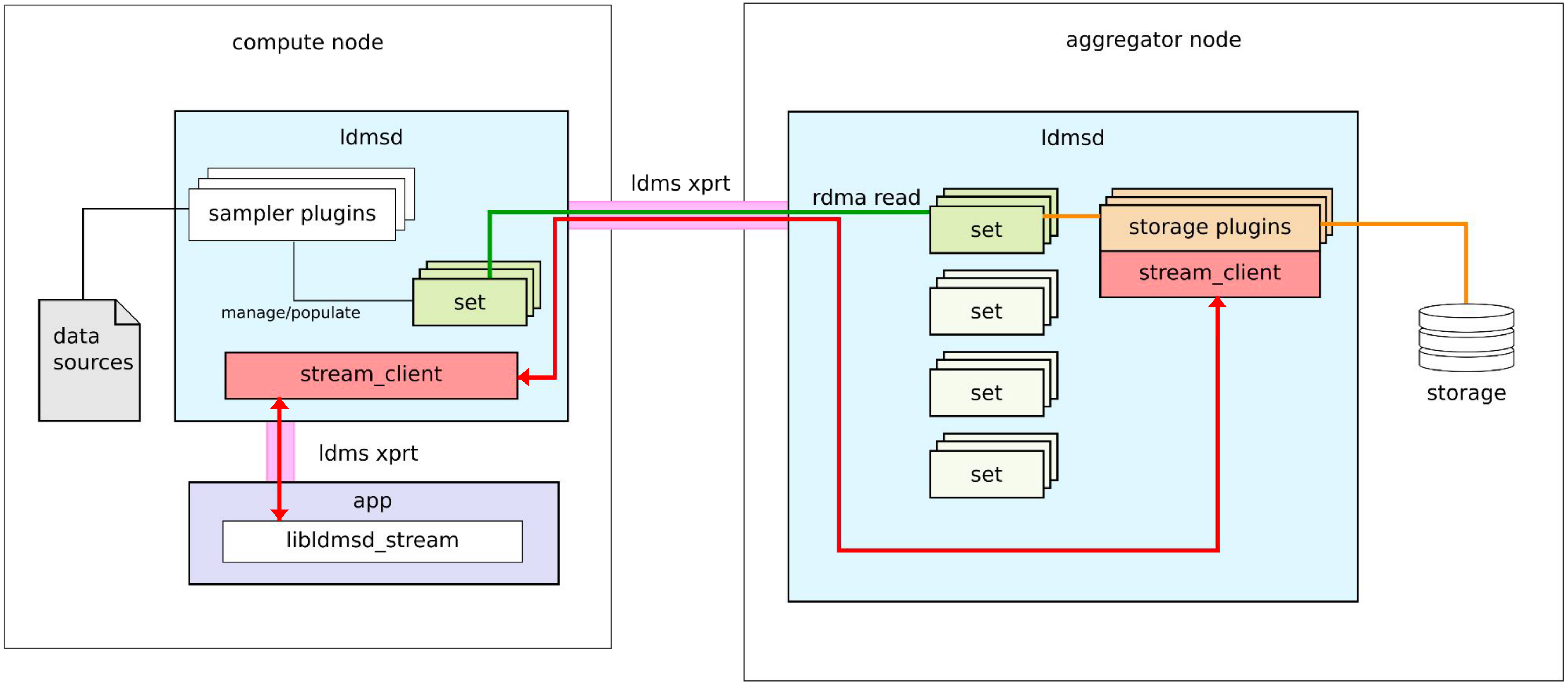


Transport Modes: Event-driven Push of JSON/String Data

Note: This can be bi-directional



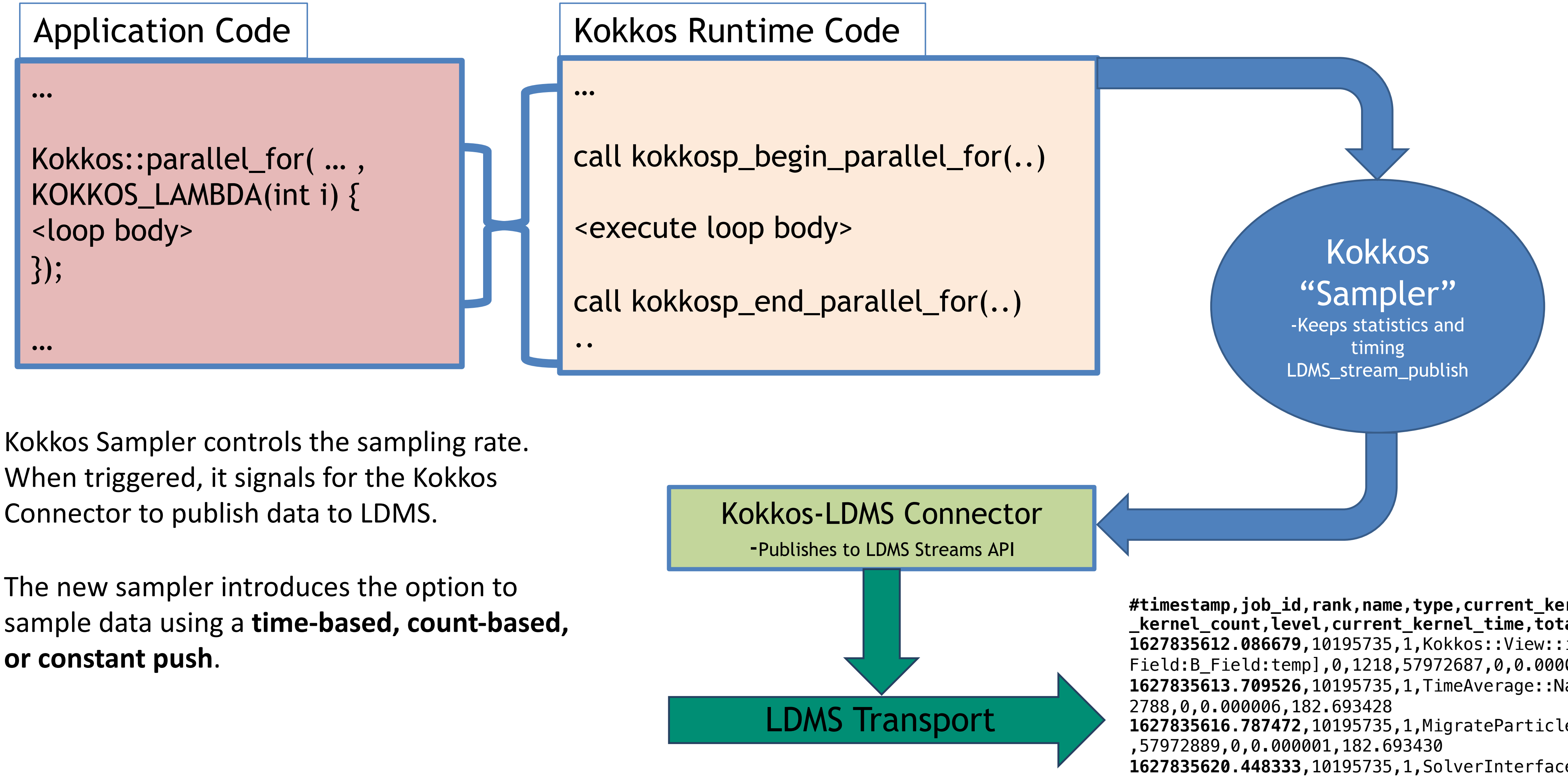
Application+System Data Collection



Application Progress & Performance Data Acquisition & Use

PUBLISH KOKKOS PERFORMANCE DATA TO LDMS

(CALIPER & DARSHAN DATA LIKEWISE)



Application Code

```
…

Kokkos::parallel_for( … ,
KOKKOS_LAMBDA(int i) {
<loop body>
});

…
```

Kokkos Runtime Code

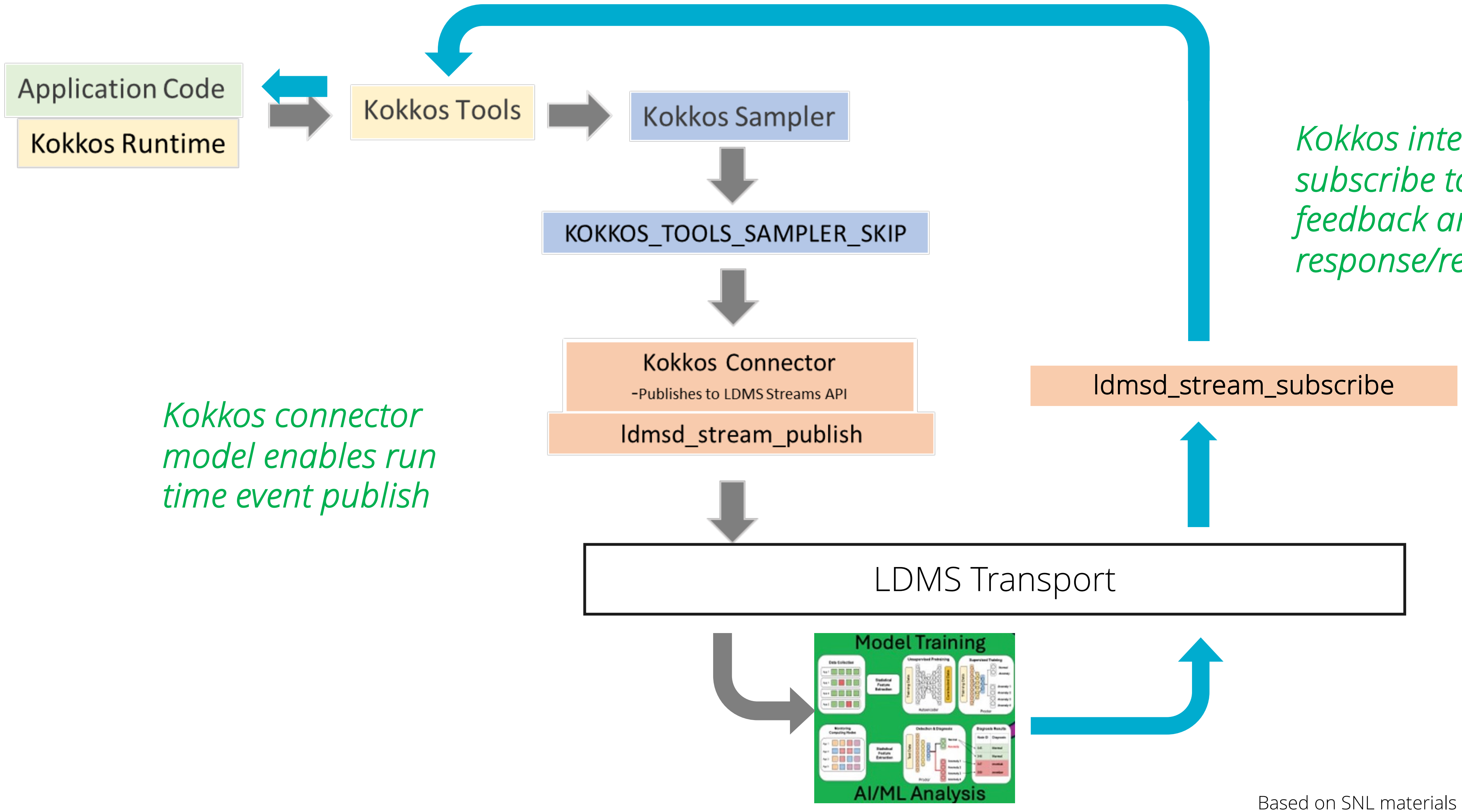
```
…

call kokkosp_begin_parallel_for(..)

<execute loop body>

call kokkosp_end_parallel_for(..)
..
```

Kokkos Sampler controls the sampling rate. When triggered, it signals for the Kokkos Connector to publish data to LDMS.

The new sampler introduces the option to sample data using a **time-based, count-based, or constant push**.

```
#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:
Field:B_Field:temp],0,1218,57972687,0,0.000005,182.693422
1627835613.709526,10195735,1,TimeAverage::Name,0,24758,5797
2788,0,0.000006,182.693428
1627835616.787472,10195735,1,MigrateParticles::count,1,3540
,57972889,0,0.000001,182.693430
1627835620.448333,10195735,1,SolverInterface::BC,0,7512,579
72990,0,0.000002,182.693432
```

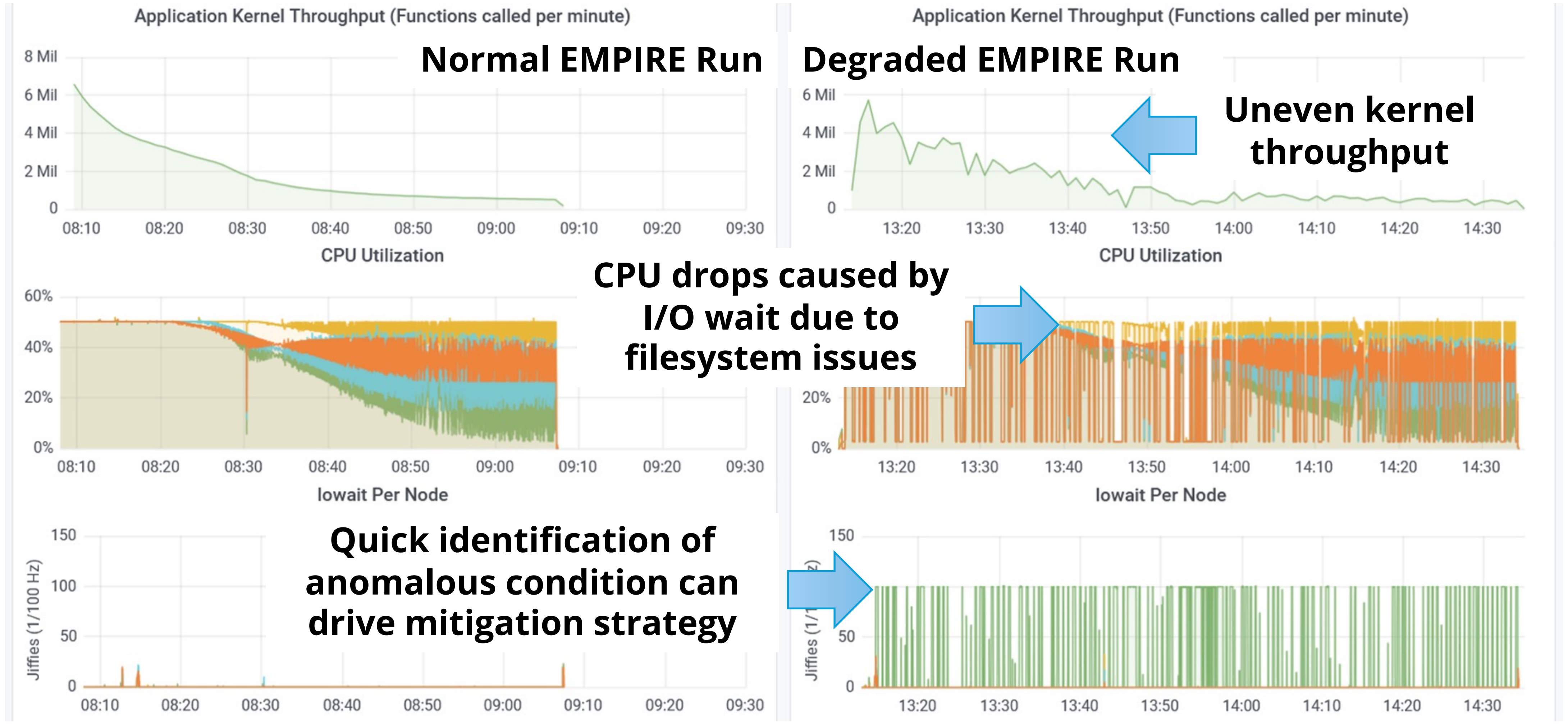
APPLICATION RESPONSE/RECONFIGURATION VIA LDMS-KOKKOS INTERACTION



Kokkos interface can also subscribe to analysis feedback and use to drive response/reconfiguration

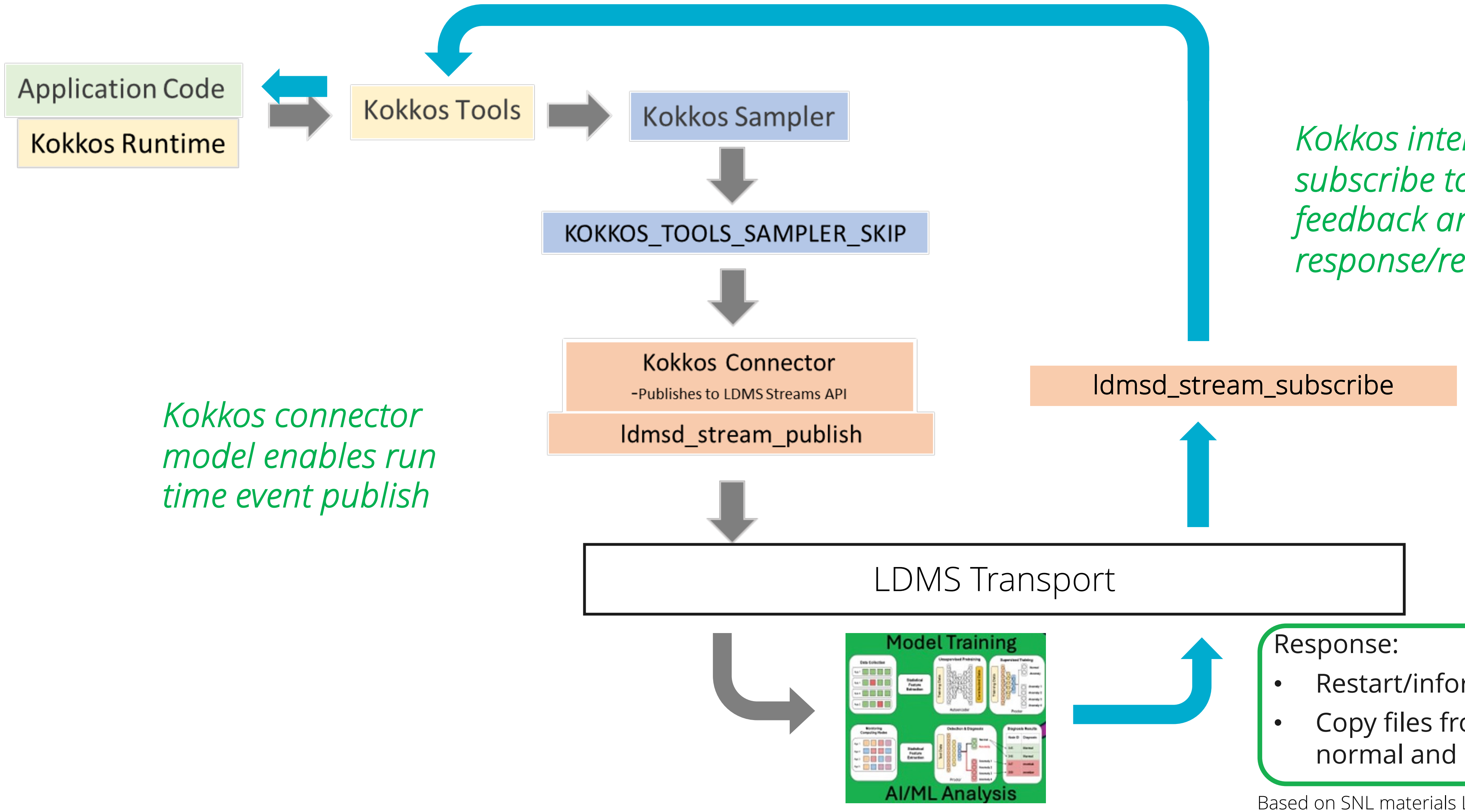
Kokkos connector model enables run time event publish

APPLICATION RESPONSE/RECONFIGURATION VIA LDMS-KOKKOS INTERACTION



EMPIRE Performance Degradation Investigation

APPLICATION RESPONSE/RECONFIGURATION VIA LDMS-KOKKOS INTERACTION



Kokkos interface can also subscribe to analysis feedback and use to drive response/reconfiguration

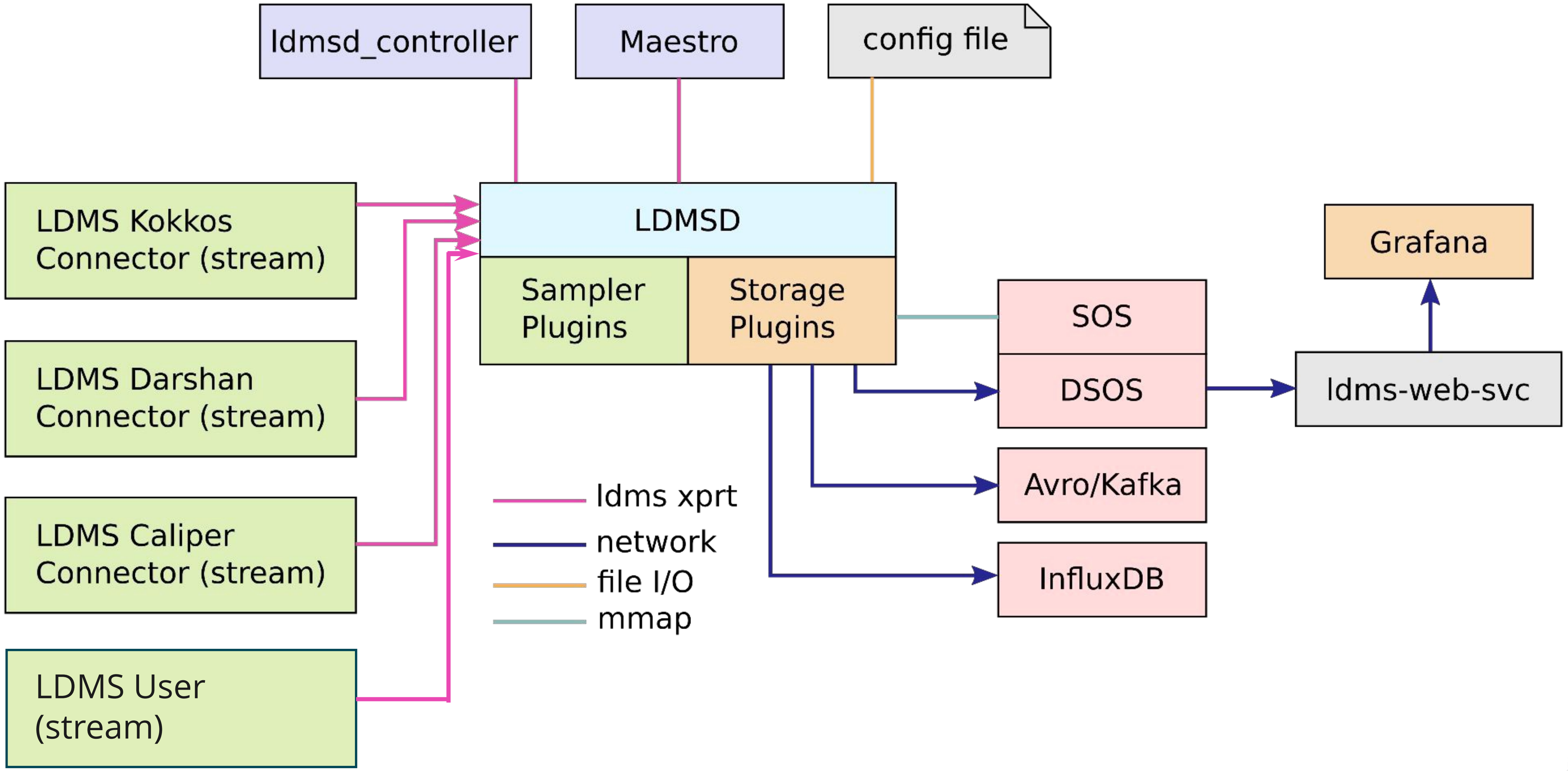
Kokkos connector model enables run time event publish

Response:

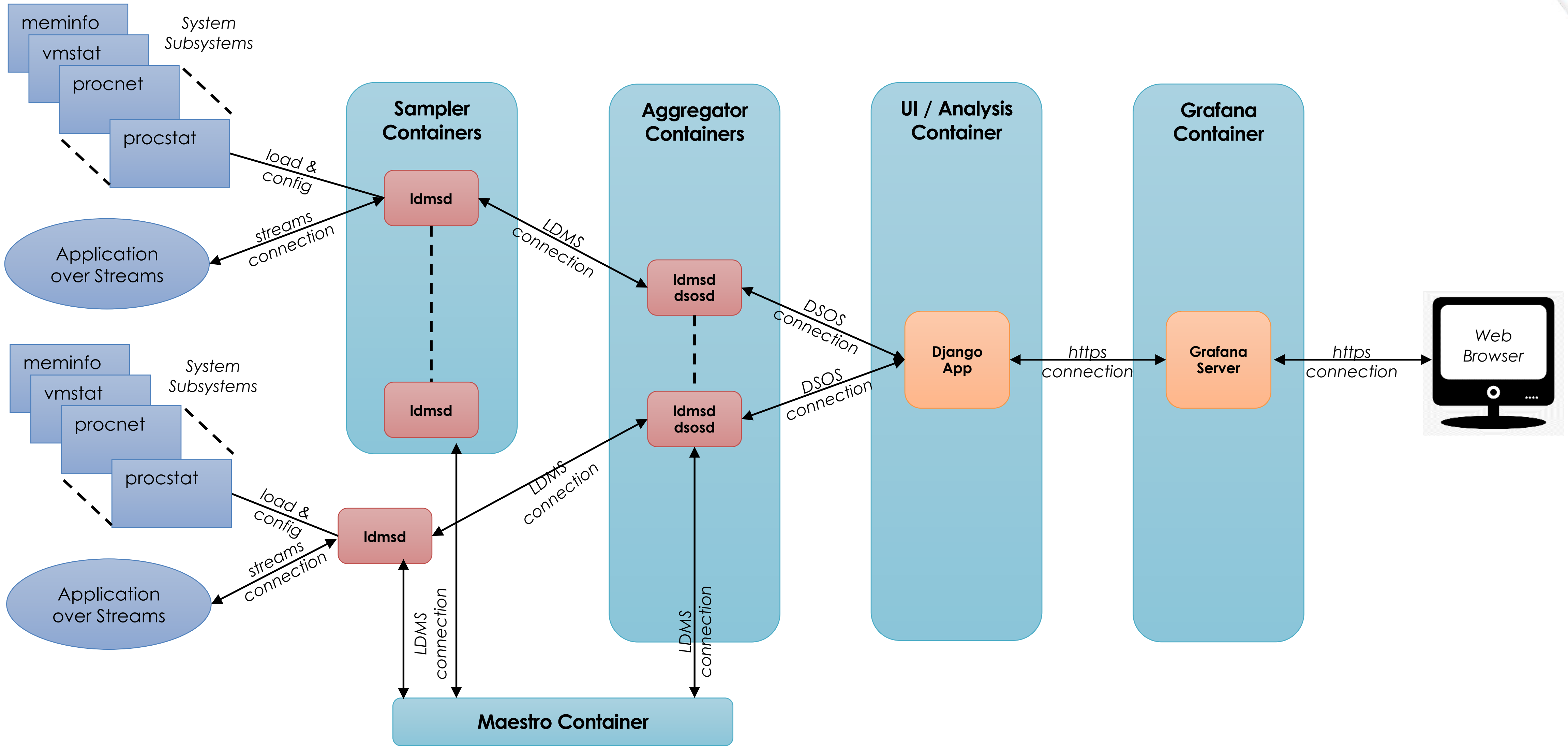
- Restart/inform application
- Copy files from slow OST to normal and continue

Based on SNL materials LDMSCON 2024

LDMS Ecosystem



CONTAINERIZED ELEMENTS EASE DEPLOYMENT



LOCATIONS FOR DOCKER LDMS RELATED CONTAINERS

LDMS Sampler: https://hub.docker.com/r/ovishpc/ldms-samp/

LDMS Aggregator: https://hub.docker.com/r/ovishpc/ldms-agg/

Web services: https://hub.docker.com/r/ovishpc/ldms-web-svc/

Grafana: (with DSOS plugin): https://hub.docker.com/r/ovishpc/ldms-grafana/

Mastro: (ldmsd orchestration): https://hub.docker.com/r/ovishpc/ldms-maestro/

LDMS Feature Update

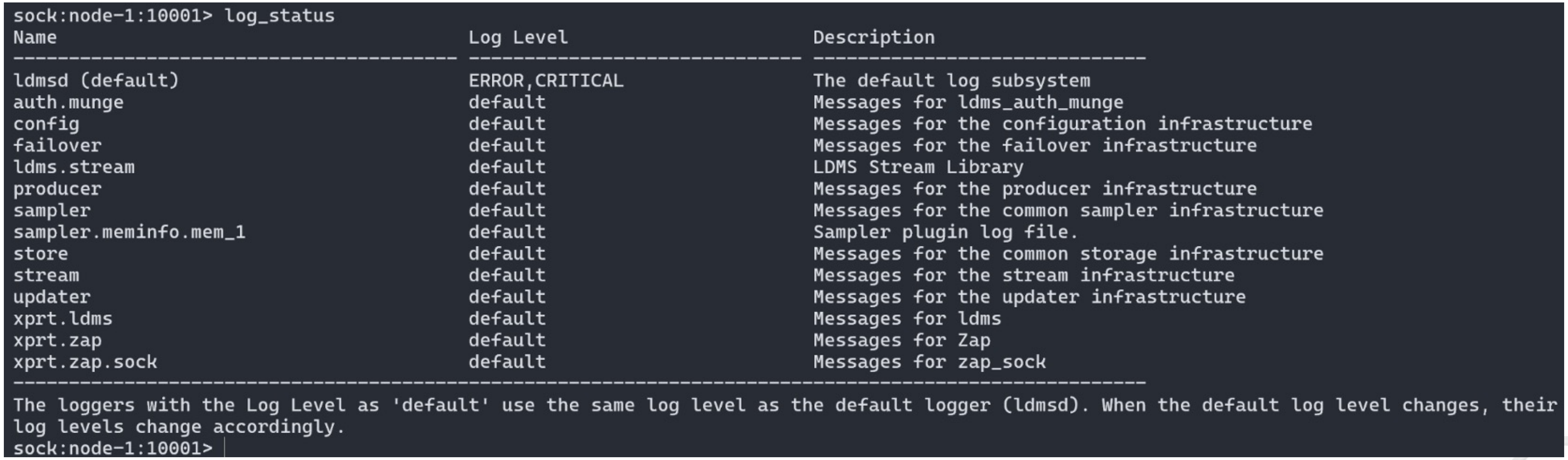
LDMS Ecosystem Main Features (recap)

Transports	Sampler Plugins	Store Plugins
• Support for multiple transports: • Ethernet, IB, iWarp, Omnipath, RoCE, Aries, Slingshot • RDMA: on supported transports, there is no CPU intervention/overhead on RDMA read • Authentication: • Munge, shared secret, none	• System Metrics: • CPU utilization • GPU utilization • Memory usage • Network bytes/packets read/written etc • File system bytes read/written • Hardware performance counters • Facility resources • and more • Application Information • Job information • Kokkos • Darshan • Caliper • And more	• CSV • Avro/Kafka • InfluxDB • SOS • Victoria Metrics (under development)

NEW LDMS FEATURES AND IMPROVEMENTS

- **Per-subsystem logging definitions:**
  - **Enables high fidelity insight into defined subsystem** for troubleshooting/debug without incurring the overhead of logging at that level for ALL subsystems
    - **4.4.x and prior:** ldmsd log levels were ordered from low to high (DEBUG, INFO, WARNING, ERROR, CRITICAL, QUIET) where any level would provide logs from that level and everything higher e.g., ERROR would provide logs at both ERROR and CRITICAL levels
    - **4.5.x:** logs are on a per-component basis (e.g., samplers, stores, transports) and are the union of levels specified e.g., "log_level name=sampler.meminfo level=INFO,CRITICAL" would provide logs at both INFO and CRITICAL levels but only for meminfo sampler plugins, "log_level name=config level=DEBUG, would provide DEBUG only logging of the configuration subsystem (NOTE: for backwards compatibility a trailing comma must be present if only a single log level is specified)

Example list of subsystems and log levels



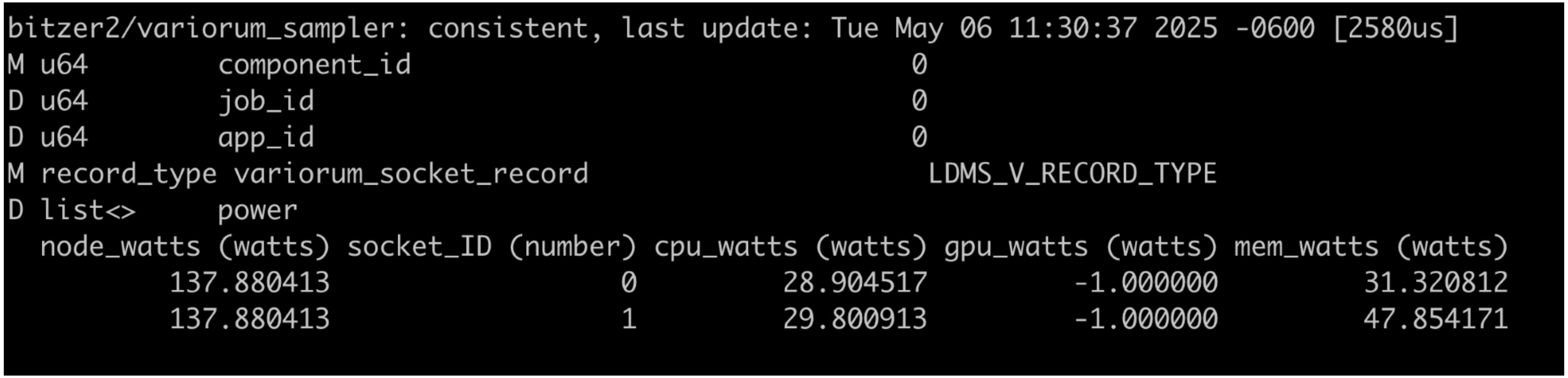
```
sock:node-1:10001> log_status
Name                                     Log Level                      Description
---------------------------------------- ------------------------------ ------------------------------
ldmsd (default)                          ERROR,CRITICAL                 The default log subsystem
auth.munge                               default                        Messages for ldms_auth_munge
config                                   default                        Messages for the configuration infrastructure
failover                                 default                        Messages for the failover infrastructure
ldms.stream                              default                        LDMS Stream Library
producer                                 default                        Messages for the producer infrastructure
sampler                                  default                        Messages for the common sampler infrastructure
sampler.meminfo.mem_1                    default                        Sampler plugin log file.
store                                    default                        Messages for the common storage infrastructure
stream                                   default                        Messages for the stream infrastructure
updater                                  default                        Messages for the updater infrastructure
xprt.ldms                                default                        Messages for ldms
xprt.zap                                 default                        Messages for Zap
xprt.zap.sock                            default                        Messages for zap_sock
----------------------------------------------------------------------------------------------------------
The loggers with the Log Level as 'default' use the same log level as the default logger (ldmsd). When the default log level changes, their
log levels change accordingly.
sock:node-1:10001>
```

NEW LDMS FEATURES AND IMPROVEMENTS

- **Operator Decomposition:** enables users to store derived metrics in the database via configuration
- **`json_stream_sampler`:** converts Stream (message) data to LDMS metric sets
- **Same YAML file for Maestro orchestration & direct ldmsd configuration:**
  - Using the -y command-line option with a YAML file that defines configuration for all, or a subset of, ldmsds in a system
- **LDMS in the cloud:** Enable dynamic producer hostname resolution
- **LDMS Rails:** bundle LDMS endpoints together to reduce bottleneck / increase aggregate bandwidth
- **Quota Group (qgroup):** limits stream/message data flowing through a user-defined group of LDMS processes
- **IPv6 support in LDMS transport**

NEW LDMS FEATURES AND IMPROVEMENTS CONT.

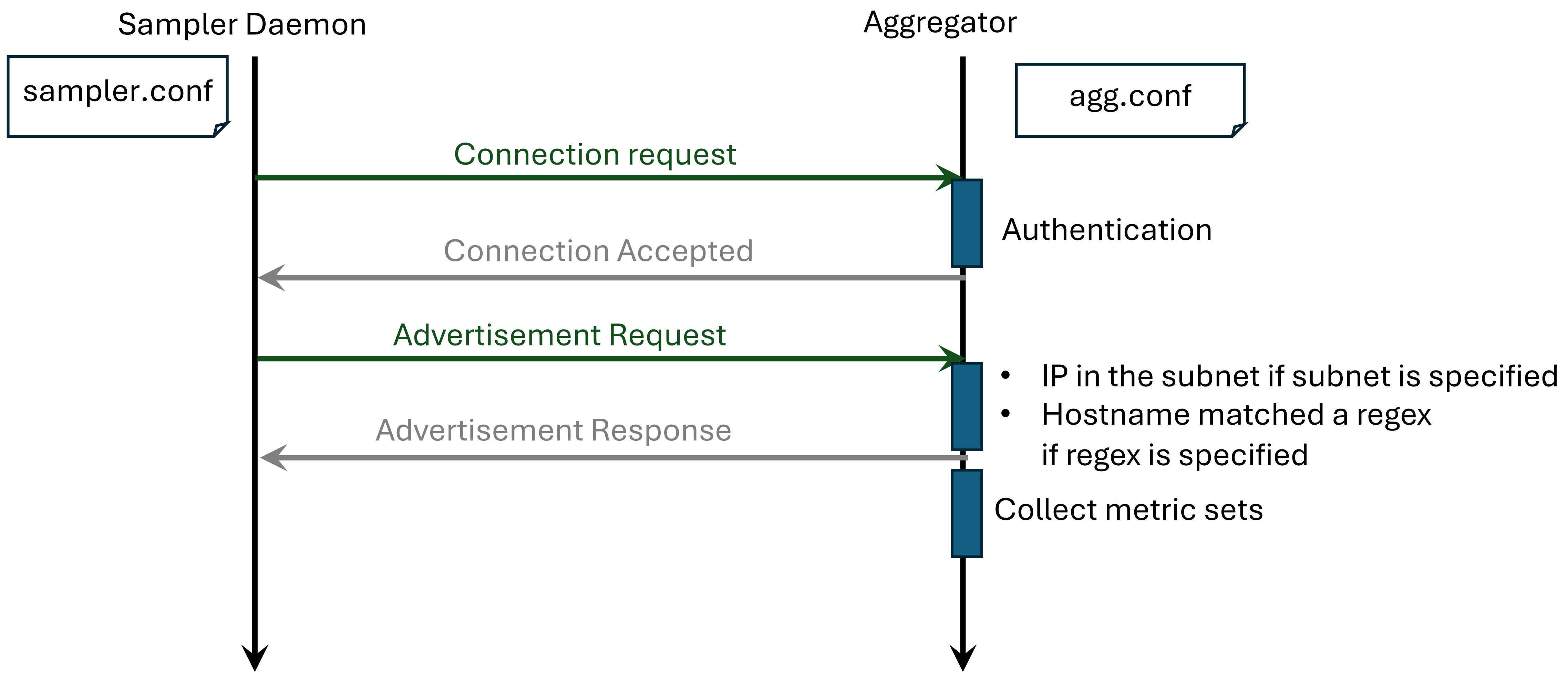
- **Increased performance/load information:**
  - **Thread statistics** in various levels (I/O, ldmsd workers)
  - **Stream (message) statistics** (e.g. dropped messages, byte counts, message counts per stream)
- **Exclusive thread for sampler plugin:** prevent high-overhead samples from blocking others
- Documentation improvements
  - **Read-the-doc support:** documents accessible on-line
- Automatic docker container generation & publication upon LDMS release
- LLNL's **Variorum LDMS sampler**: power monitoring



```
bitzer2/variorum_sampler: consistent, last update: Tue May 06 11:30:37 2025 -0600 [2580us]
M u64        component_id                                 0
D u64        job_id                                       0
D u64        app_id                                       0
M record_type variorum_socket_record                       LDMS_V_RECORD_TYPE
D list<>      power
  node_watts (watts) socket_ID (number) cpu_watts (watts) gpu_watts (watts) mem_watts (watts)
          137.880413                  0         28.904517         -1.000000         31.320812
          137.880413                  1         29.800913         -1.000000         47.854171
```

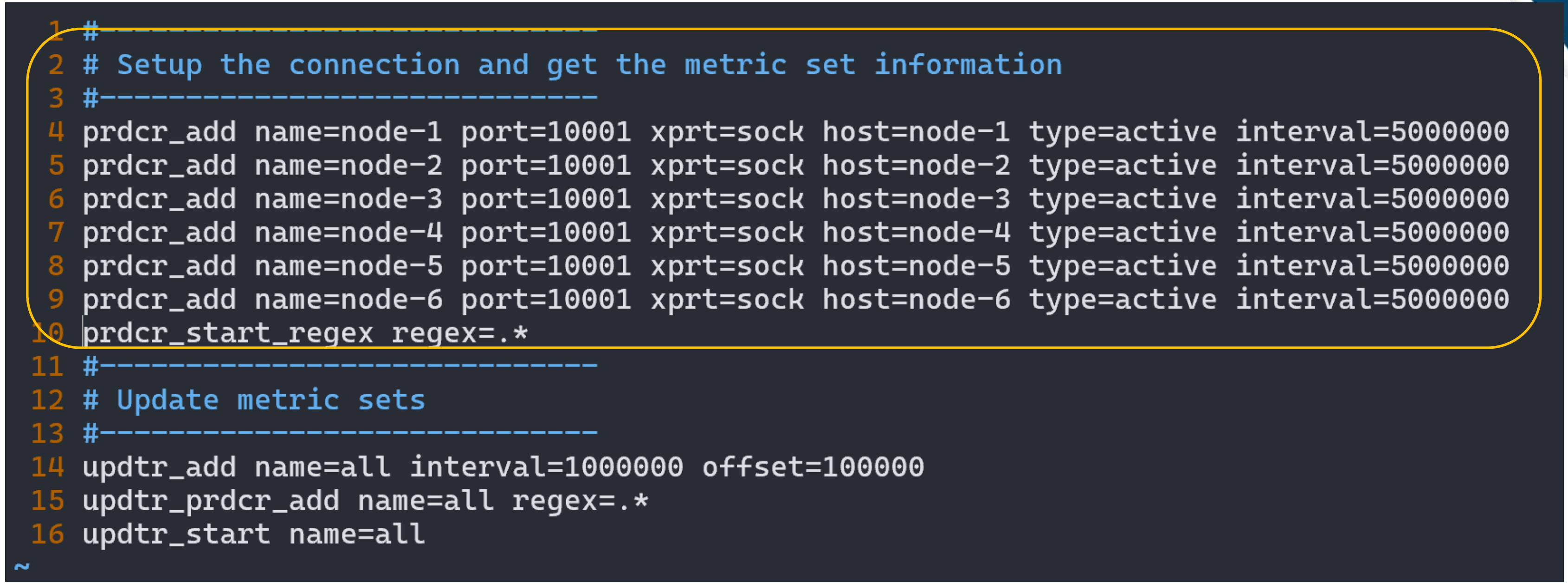
Peer Daemon Advertisement: Simplifying Deployment

Enables an ldmsd to add producers for advertisers whose hostnames match a regular expression or whose IP address is in a specified range



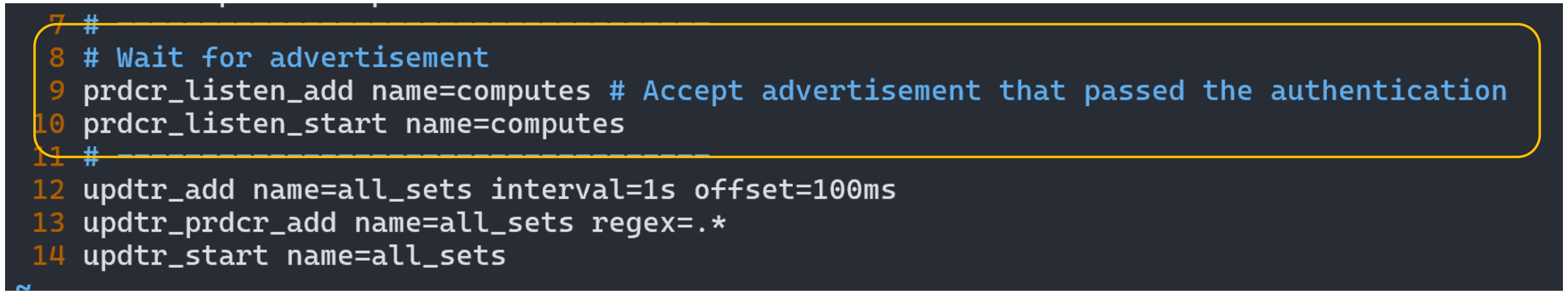
COMPARISON OF AGG.CONF V4.4.X VS. V4.5.X

v4.4.x



```
#------------------------------
# Setup the connection and get the metric set information
#------------------------------
prdcr_add name=node-1 port=10001 xprt=sock host=node-1 type=active interval=5000000
prdcr_add name=node-2 port=10001 xprt=sock host=node-2 type=active interval=5000000
prdcr_add name=node-3 port=10001 xprt=sock host=node-3 type=active interval=5000000
prdcr_add name=node-4 port=10001 xprt=sock host=node-4 type=active interval=5000000
prdcr_add name=node-5 port=10001 xprt=sock host=node-5 type=active interval=5000000
prdcr_add name=node-6 port=10001 xprt=sock host=node-6 type=active interval=5000000
prdcr_start_regex regex=.*
#------------------------------
# Update metric sets
#------------------------------
updtr_add name=all interval=1000000 offset=100000
updtr_prdcr_add name=all regex=.*
updtr_start name=all
~
```

v4.5.x using
Sampler Advertisement



```
# ----------------------------------
# Wait for advertisement
prdcr_listen_add name=computes # Accept advertisement that passed the authentication
prdcr_listen_start name=computes
# ----------------------------------
updtr_add name=all_sets interval=1s offset=100ms
updtr_prdcr_add name=all_sets regex=.*
updtr_start name=all_sets
~
```

LDMS Avro/Kafka Store

AVRO-KAFKA STORAGE PLUGIN

- LDMSD storage plugin
- Publishes LDMS metric set data to the Kafka bus
- Uses AVRO C-library to encode metric set data on the Kafka bus
- Store publishes metric set schema to AVRO Schema Registry
- Serdes encodes the Avro values on the wire given the schema, the schema-id is included in the header
- Kafka clients use AVRO schema service to lookup schema in order to deserialize LDMS metric set data read from the Kafka bus

JSON ENCODING MODE

- Metric set data is encoded in the Kafka message as a JSON dictionary
- Dictionary attributes are constructed from the metric set attribute names and types

```
{ “schema” : “meminfo”, “instance” : “nid00063/meminfo”, “producer” : “nid00063”,
  “attrs” : [
       { “timestamp” : { “type” : “timestamp”, “value” : 16134353, ”indexed” :  true }},
      { “component_id” : { “type” : “uint64”, “value” : 1, “indexed” : true }},
      . . .
      ]
}
```

AVRO ENCODING MODE

- Metric set data is encoded and optionally compressed using an AVRO serdes
  - Significantly reduces message size
- Kafka message contains schema UID obtained from the AVRO Schema Service
- Clients must use the AVRO Schema Service to obtain schema and decode messages read from the Kafka bus

AVRO-KAFKA STORE CONFIGURATION

- **Encoding** attribute - optional
  - One of “avro”, or “json”, default is “avro”
- Kafka **topic** attribute - required
  - Value is a format specifier
- Avro **serdes_conf** attribute - optional
  - Default schema registry is "http://localhost:8081"
- Kafka **kafka_conf** attribute - required
  - Specifies the location of the broker

```
# decomposition storage policy
strgp_add name=decomp regex=.* plugin=store_sos container=ldms_data
decomposition=/opt/ovis/etc/flex_decomp.json
strgp_start name=decomp

load name=store_avro_kafka
config name=store_avro_kafka kafka_conf=/opt/ovis/etc/kafka.conf
serdes_conf=/opt/ovis/etc/serdes.conf topic="ldms/%F/%I"

# Storage policy
strgp_add name=aks regex=.* plugin=store_avro_kafka decomposition=/opt/ovis/etc/flex_decomp.json
container=<kafka broker hostname>:<port> (NOTE: This is optional)
strgp_start name=aks
```

Future Plans

PLANNED DIRECTIONS OVER NEXT YEAR

- Simpler deployment
  - Default configurations that just work
    - Curated metric sets along with Operator Decomposition recipes to provide base-level insights
    - Remove burden of specifying set memory size
  - Configurations tailored to initial system standup phase
    - Analyses aimed at identification of anomalous behaviors and components
  - More flexible, efficient, and secure Stream configuration
    - e.g., on-the-fly application-initiated setup and teardown of bi-directional communication paths specifically for solicitation of run time feedback from analysis systems
- Multi-configuration plugins
  - Enables multiple instances of same sampler with different metrics and rates
- Better daemon stats for tuning aggregator loading and load-balancing
  - Providing histogram instead of just min, max, and average
- Additional samplers
  - Additional GPU, network, and storage
- Multi-tenancy
  - Support for attribution of per-tenant resource utilization

COLLABORATION OPPORTUNITIES

Application code developers and analysts:

- Engaging with users and code developers to include hooks for application insight and provide actionable intelligence

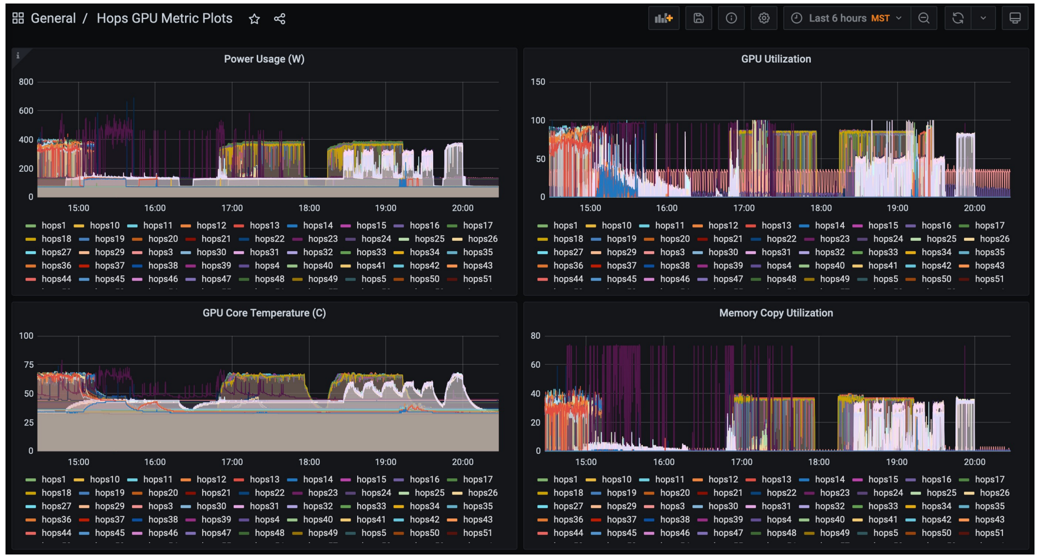
Analysis and visualization:

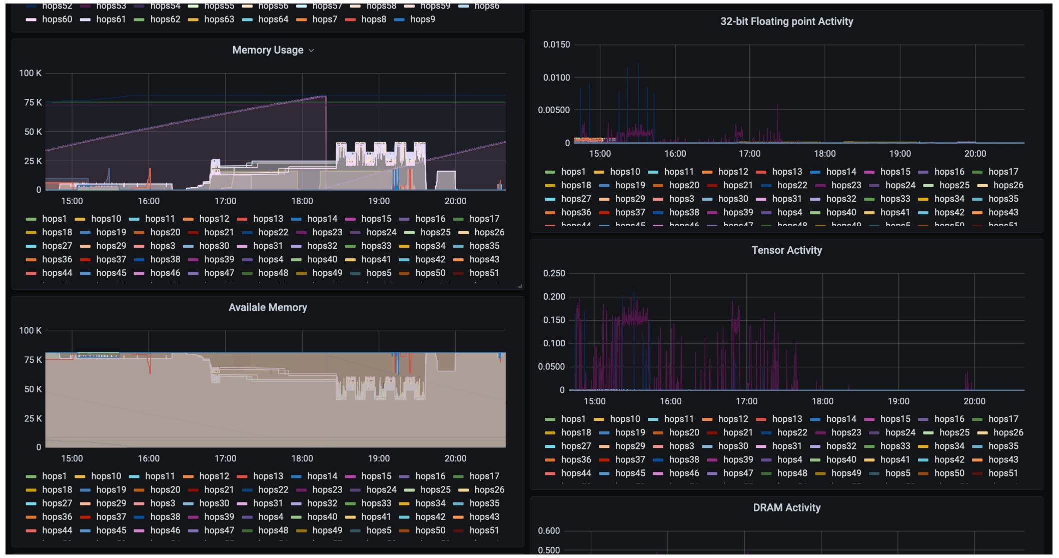
- Analysis and dashboarding on production system & application data
- AI/ML-driven low-latency inference and multi-objective cost function-based resource allocation and scheduling
- Grafana & Jupyter notebook

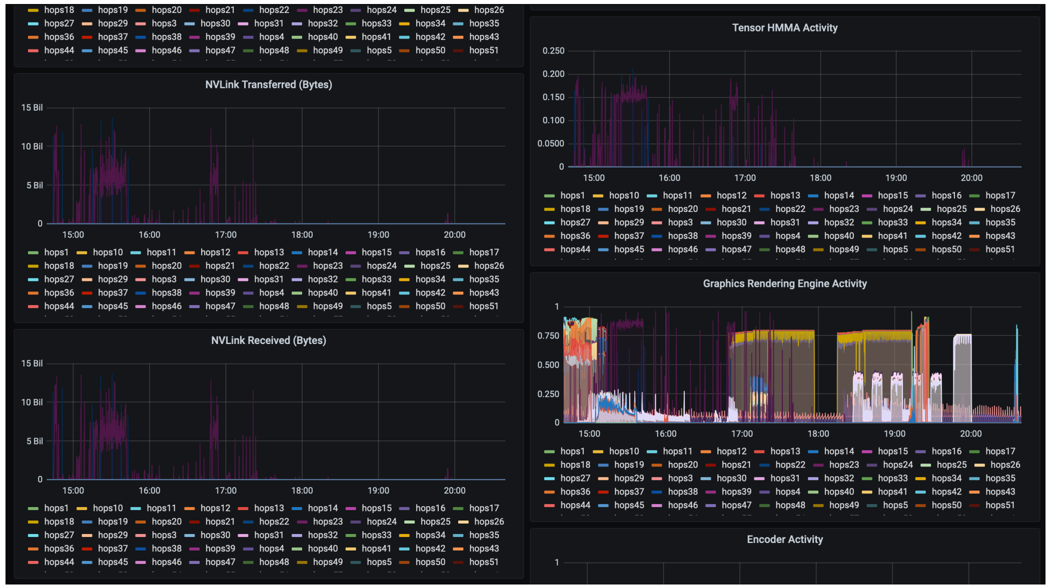
Contribute code to LDMS ecosystem:

- Resources and information at: github.com/ovis-hpc/ldms

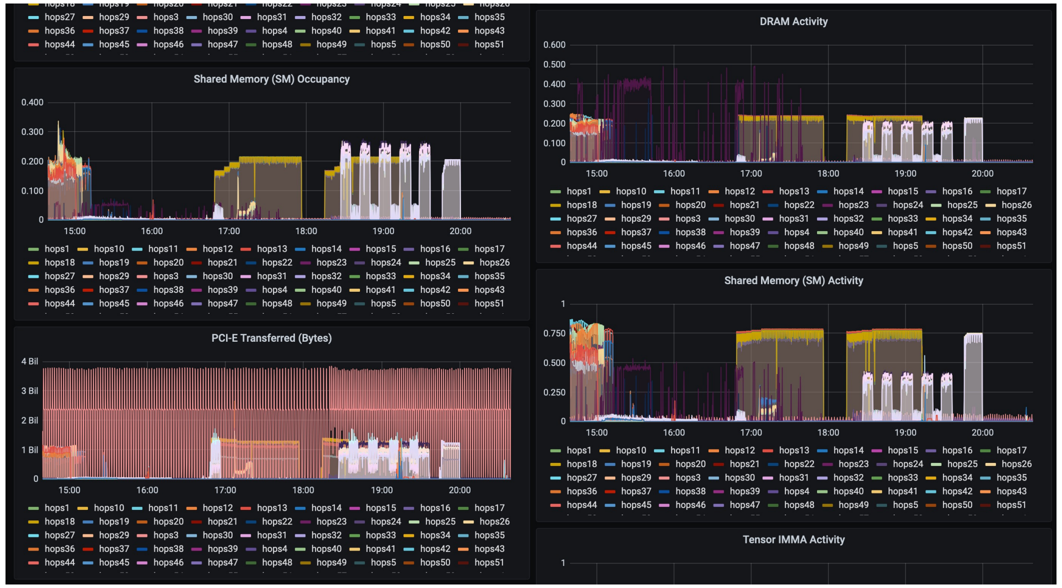
GPU DASHBOARDS FROM NVIDIA DCGM SAMPLER DATA







NVLink Transferred (Bytes)
NVLink Received (Bytes)
Tensor HMMA Activity
Graphics Rendering Engine Activity
Encoder Activity



Shared Memory (SM) Occupancy
PCI-E Transferred (Bytes)
DRAM Activity
Shared Memory (SM) Activity
Tensor IMMA Activity

GET ENGAGED

- We have regular LDMS user group meetings to discuss current work, plan future work, and to discuss our community driven project. For a low traffic announcement and meeting info list, mail listserv@listserv.llnl.gov with the body:
  `subscribe ldms-announce $FirstName $LastName`
- LDMS on GitHub lives at https://github.com/ovis-hpc/ldms
- LDMS readthedocs - https://ovis-hpc.readthedocs.io/en/latest/
- **Contact ldms@sandia.gov for any additional questions or info**

LDMSCON2025 PLUG

2025 LDMS Conference

- **Theme:** How to deliver monitoring information derived knowledge to end users in easily consumable form
- **Dates:** June 24th - 26th (Chicago, IL USA)
- **Activities:** Tutorials, user and developer presentations, Lightning talks, discussions on future directions, networking
- Currently soliciting contributions for presentations

Regular registration ends May 15, 2025 @ 11:59 pm CDT

NOTE: You do not have to be a LDMS user to attend!!!

- Come and share monitoring experiences and needs
- Network with the community
- Influence project directions

THANKS TO COLLABORATORS, CONTRIBUTORS, AND USERS

BERKELEY LAB

UNIVERSITY OF CENTRAL FLORIDA 1963

aws

Caliper

Los Alamos NATIONAL LABORATORY

NM STATE UNIVERSITY

ILLINOIS

AWE

Hewlett Packard Enterprise

NORTHEASTERN UNIVERSITY 18 98 LVX VERITAS VIRTVS

DARSHAN HPC I/O Characterization Tool

kokkos

BOSTON UNIVERSITY

LVX VERITAS VIRTVS

Questions?