SAND2023-03680C

EVALUATING AND INFLUENCING EXTREME-SCALE MONITORING IMPLEMENTATIONS

Andrew Barry (HPE), Jim Brandt (SNL), Ann Gentile (SNL)
Christopher Morrone (LLNL), Eric Roman (NERSC)
Alec Scott (LLNL), Kathleen Shoga (LLNL), Tom Tucker (OGC)

CUG 2023 - May 10, 2023

MOTIVATION

- Compare HPE's CSM with a site-specific non-CSM approach to HPC monitoring with a **Focus on:**
  - Lightweight Distributed Metric Service (LDMS) implementation for node-level monitoring
  - Integration with sites' data analysis capabilities
  - Encourage community discussion on these points and how these differences should evolve / be supported
- Update on new LDMS features
- Get feedback from CUG community on desired directions for LDMS development
  - Deployment
  - Functionality
  - External interactions
- Generate closer collaboration between HPE and the HPC monitoring community

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

CSM BACKGROUND

- System Monitoring Application (SMA) is a collection of technologies which collect, aggregate, store, and display System Monitoring Data.
- All SMA services are run as Kubernetes *pods* with the exception of LDMS samplers, which are run as native services within a node's operating system.
- The Metrics pipeline has Kafka at its center.
- Node metrics are collected and transported off-node and into the Kafka bus using the LDMS infrastructure.
- SMA PostgreSQL persisters are instances of a scalable, parallel tool for taking metrics (including LDMS metrics) from the Kafka bus and storing them in the HPE SMA PostgreSQL database.

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

LDMS-SPECIFIC CSM OPERATION

- LDMS configuration is accomplished by an *sma-ldms- config* pod taking the LDMS configuration from a Kubernetes ConfigMap, and writing it to a local Simple Storage Service (S3) volume, which is then read by nodes during the node personalization process.
    - Credential distribution to LDMS configuration at boot time is slow and unreliable
        - SMA starts a pod on a fixed worker node (*ncn-w001*) with a Kubernetes Persistent Volume Claim (PVC) mount holding the LDMS secrets.
        - SMA next copies the secrets to a node and then terminates the pod on *ncn-w001*.
        - Using this approach on each node running a sampler is not only slow but unreliable
- LDMS sampler daemons publish data to a scalable set of LDMS aggregator pods; one set for compute nodes and one set for NCNs.
- The LDMS aggregators then publish the data to the same Kafka bus that is used for transmitting log and hardware metric data.
- The *telemetry-api* acts as a Kafka consumer and http Security Service Edge (SSE) subscription service, feeding Kafka topic data to an http client
    - Problems with efficiency and reliability
    - Stopped feeding data for no apparent reason
    - Frequently triggered Kafka rebalancing

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

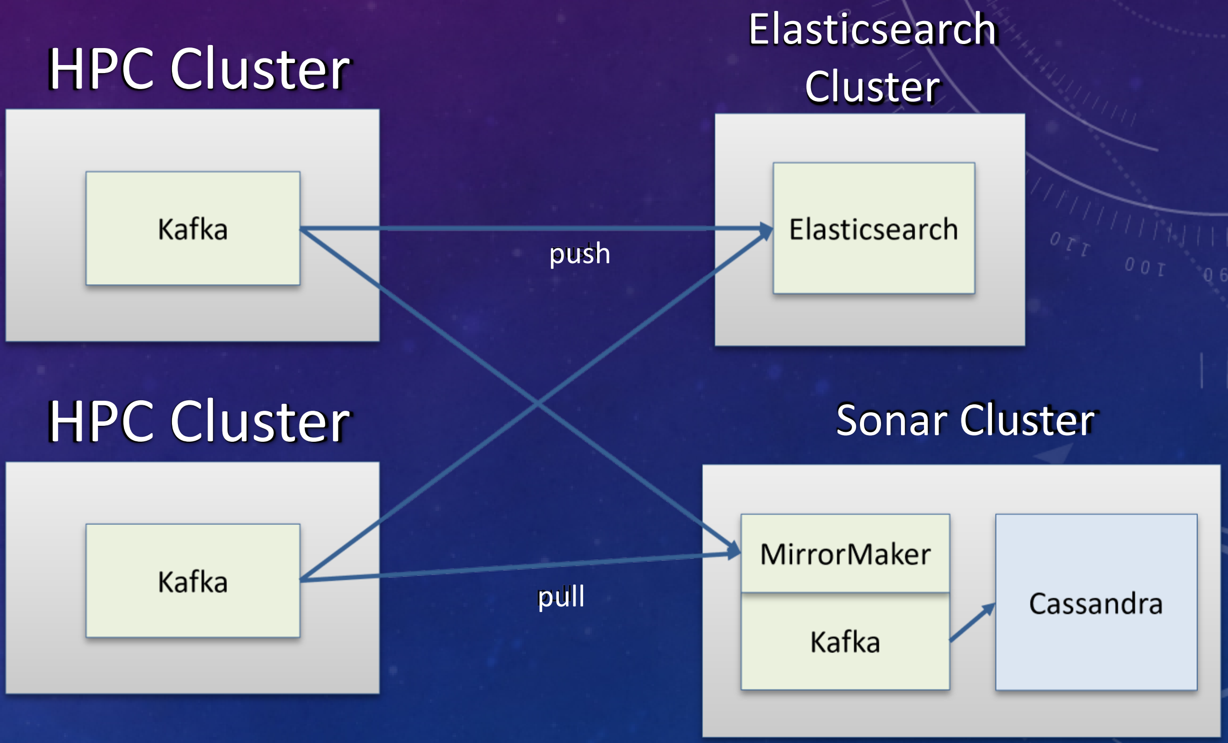
NERSC CUSTOMIZATIONS

- Run separate community release version of LDMS infrastructure for collecting additional node metrics
  - CSM provided Ansible script not very customizable
  - CSM supported sampler plugins are limited compared with those provided by the open-source community
  - Site specific CSM customizations would have to be re-implemented with each CSM update
- Manage separate community release version of LDMS aggregators with CSM Kubernetes cluster
  - Connect to SMA-started LDMS samplers
  - Feed NERSC-wide shared data store called OMNI using custom LDMS VictoriaMetrics storage plugin
- Improved credential distribution to LDMS node configurations by moving the *sma-ldms-compute* Ansible role from *node personalization* to *image customization*
- Demonstrated that a more efficient confluent-python based version of *telemetry-api* could handle full data rate
- Left HPE LDMS aggregator with Cray-provided Kafka store feeding Kafka
  - Read data directly from Kafka and write to OMNI

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

LLNL TOSS 4 DEPLOYMENT

- LLNL's goal is uniformity (scale number of clusters without corresponding scaling in number of staff)
    - Use of same OS (TOSS) and tooling across all systems
    - Use of common infrastructure for ingestion and analysis of data across the data center



TOSS 4 DEPLOYMENT OF LDMS

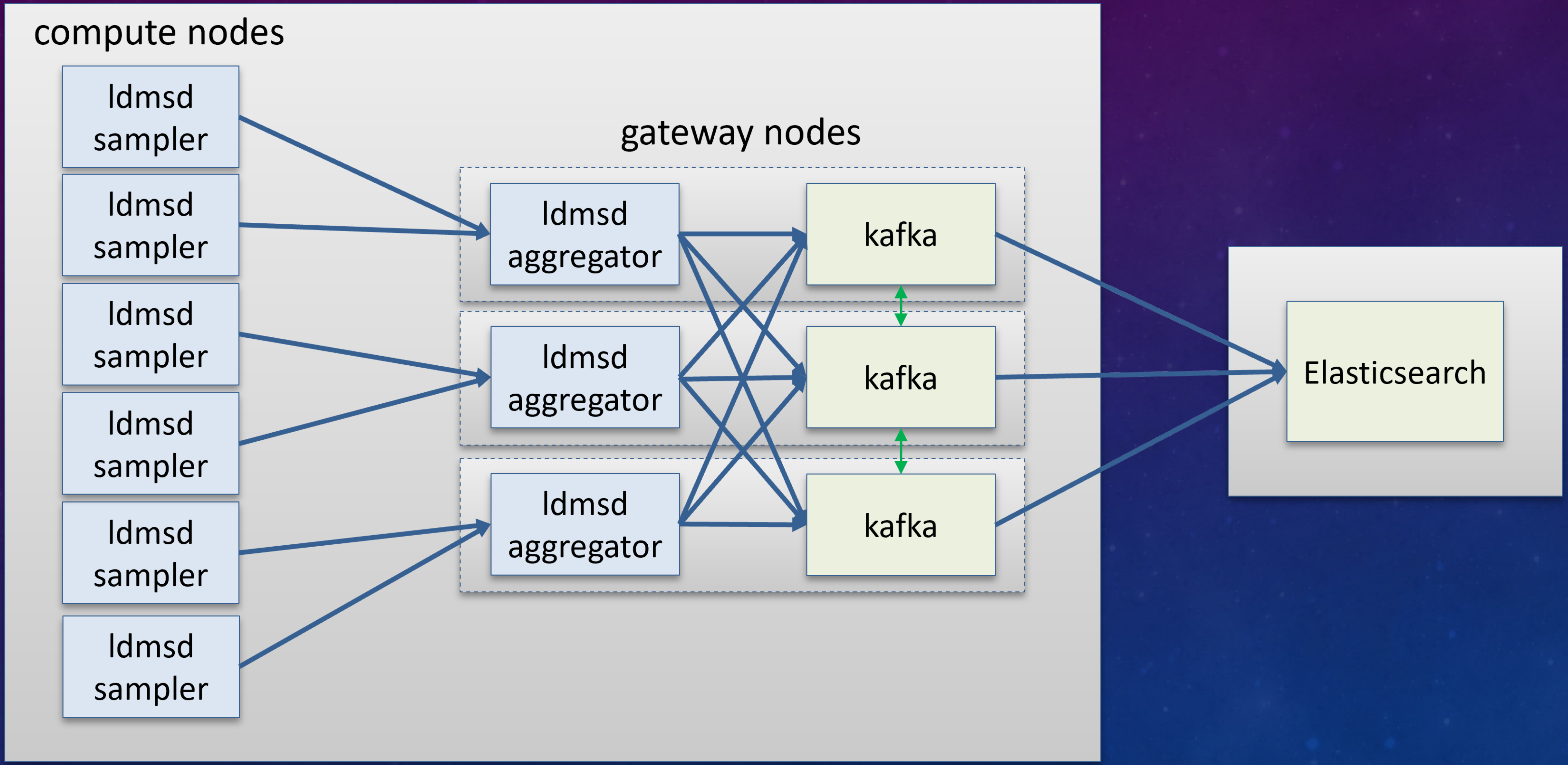
- LNL's goal is uniformity
  - One build of LDMS should run ALL LLNL HPC systems (including El Capitan)
- Toss 4 based systems configured from one central Ansible repository
- In the Ansible repository, there are two main *ldmsd* configuration templates:
  - Samplers
  - Aggregators
- All decisions about which samplers are to run on a particular node are made through logic in Ansible.
  - Example: LLNL custom "facts" script parses the output of lspci and sets various facts about types of GPUs and network devices that are found.
  - Example: Ansible inventory settings
- Each node of a cluster runs a sampler *ldmsd* while only a select subset of cluster nodes run an aggregator
- Aggregator ldmsds have a subset of *ldmsd* samplers statically assigned to them in their configurations via the Ansible template.

LLNL ENHANCEMENTS TO LDMS INFRASTRUCTURE

- Two samplers, slingshot_metrics and slingshot_info, were custom developed by LLNL for El Capitan.
  - Utilizes the new LDMS metric types LIST and RECORD available in versions 4.3.10 and later
  - Samplers use a single list of records in which each entry in the list contains a record of metrics for a single Slingshot interface.
  - Samplers resize the list as interfaces appear and disappear.
- LLNL and Open Grid Computing collaborated to develop a new LDMS store named store_avro_kafka.
- An internal Kafka instance per cluster provides all clusters with an independent and reliable location in which to publish all monitoring information.
- The Kafka data bus acts as a common point of data sharing between multiple producers and consumers.
- Initial default sampling intervals are 5 seconds

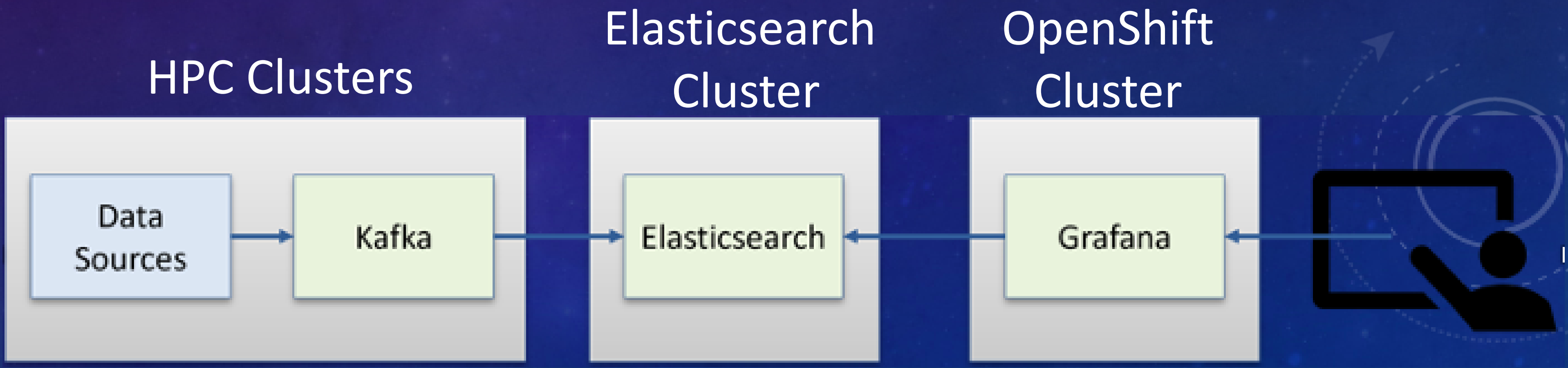
Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

HPC Cluster


compute nodes
ldmsd sampler
ldmsd sampler
ldmsd sampler
ldmsd sampler
ldmsd sampler
ldmsd sampler
gateway nodes
ldmsd aggregator
ldmsd aggregator
ldmsd aggregator
kafka
kafka
kafka
Elasticsearch

LLNL DATA FLOW DIAGRAM

- Data from Kafka topics is streamed to Elasticsearch using the Kafka Connect Elasticsearch Service Sink connector.
- As messages appear in one or more topics, the Sink consumes those messages, uses Avro to deserialize them (looking up the matching schemas in the Schema Registry), and writes the data into the correct indices in Elasticsearch.
- Using a simple regular expression, LLNL configures the Sink to watch many topics and write the data from each into their own respective index in Elasticsearch.



SIMILARITIES AND DIFFERENCES

- NERSC runs two separate versions of LDMS simultaneously
  - CSM version provisioned by SMA
    - CSM LDMS Kafka store plugin only used for persisting data to CSM's PostgreSQL database
  - Community version provisioned using NERSC boss/worker model, deployed in a Kubernetes pod
    - The boss finds all the nodes to monitor (compute, application, and management) from the Hardware State Manager (HSM) and generates a configuration file for each type
    - Boss starts a sub-process worker for each configuration, with a message queue from the boss to each worker.
  - Community version uses VictoriaMetrics store plugin to write to NERSC-wide shared data store "OMNI"
    - This includes metrics from CSM deployed samplers
- LLNL utilizes ONLY the community version of LDMS provisioned by LLNL Ansible
  - LLNL developed Kafka store plugin

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

SIMILARITIES AND DIFFERENCES (CONT.)

- NERSC and LLNL both currently using LDMS over Socket between Samplers and Aggregators but would like to use RDMA
  - LDMS RDMA transport, “fabric”, using HPE’s libfabric over Slingshot
    - HAS been tested for functionality on Perlmutter
    - HAS NOT yet been tested at scale for resiliency
  - HPE’s libfabric implementation is different than the upstream OFA implementation
    - LLNL use of the LDMS “fabric” transport is dependent on HPE upstreaming their implementation

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

DEPLOYMENT CONSIDERATIONS

- Some site-specific choices / multiple options are intended to be supported (e.g., data stores, sampling rates, metrics sampled)
- Implementation details may impede ability to stay current with open-source LDMS release (and take advantage of new features)
  - Divergence in core functionality creates community burden to support multiple paths
- NERSC and LLNL are regular contributors to open community discussion and development which helps to drive common paths forward

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

RELEVANT NEW FEATURES OF LDMS

LIST AND RECORD METRIC TYPES

- Motivation
  - Previously schema were static
  - The number of metrics and length of arrays could not be changed once the set was created
  - Many system elements are dynamic, e.g., disk drives and network interfaces may come and go while the system is running
  - This required either deleting the set and recreating it, or creating a separate set for each system resource
- Architecture
  - Add two new metric types: LDMS_V_RECORD, and LDMS_V_LIST
  - LDMS_V_RECORD is a collection of LDMS metrics
  - An LDMS_V_LIST is a variable length list of LDMS_V_RECORD
  - Distributed heap used for the allocation of new data in an existing schema

LIST AND RECORD METRIC TYPES CONT.

- Sampler Usage
  - A system "resource" sampler (e.g. network interface, disk) can create a single set for all system resources
  - When a resource is added to the system, a record with the suitable attributes is added to a list
  - When a system resource is removed, the associated record is removed from the list
- Storage Usage - Decomposition
  - Key attributes are copied to each row in order to discriminate between different resources (e.g., disk name, network interface name)

FLEXSTORE STORAGE DECOMPOSITION

- Motivation
  - Storage plugins bind metric set handling and storage technology
  - Some metric sets require special processing to convert metric set data into rows in a database, e.g., PAPI, Slurm, …
  - This results in many more plugins than would otherwise be necessary, e.g., a separate PAPI storage plugin for Avro-Kafka, JSON-Kafka, SOS, CSV, InfluxDB, etc…
- Architecture
  - Create a metric set parsing capability in the storage policy core that “decomposes” metric set data into one or more rows
  - These rows are then presented to a storage plugin for storing
  - Significantly reduces the number of plugins required to support each schema across all storage technologies
- Configuration
  - Storage policy is configured with a JSON file that specifies how a metric set’s contents are converted into rows in a database
  - A metric set is bound to a decomposition configuration by its schema

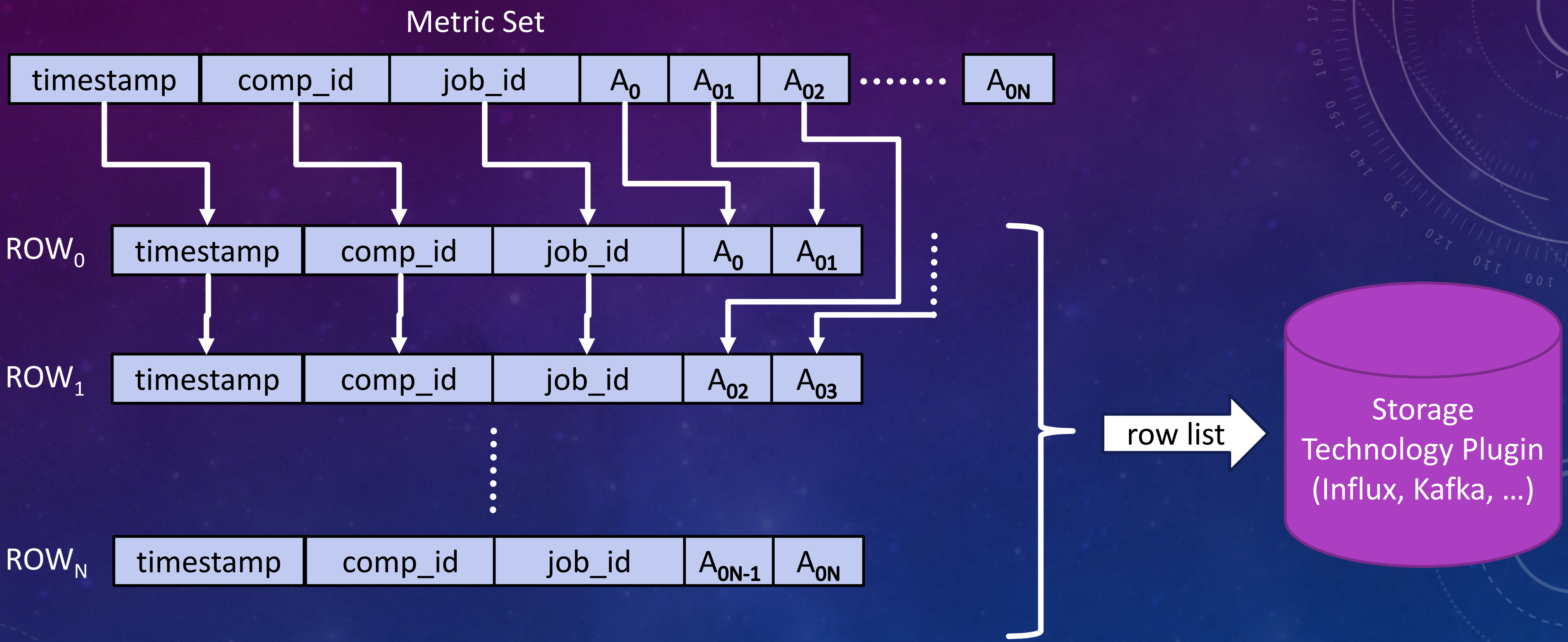
Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

FLEXSTORE STORAGE DECOMPOSITION - CONT

- Benefits
  - A single storage policy can now store multiple schema
  - Selected features from a metric may be stored independently of other features in the set
  - Records in lists and arrays can be stored as multiple rows
  - Arrays of different sizes do not result in new schema
  - No longer necessary to create “custom” plugins that parse metrics set into rows suitable for storage
  - Decouples metric set decomposition from storage plugins
    - Now need for PAPI for CSV, PAPI for Influx, etc…

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

FLEXSTORE STORAGE DECOMPOSITION

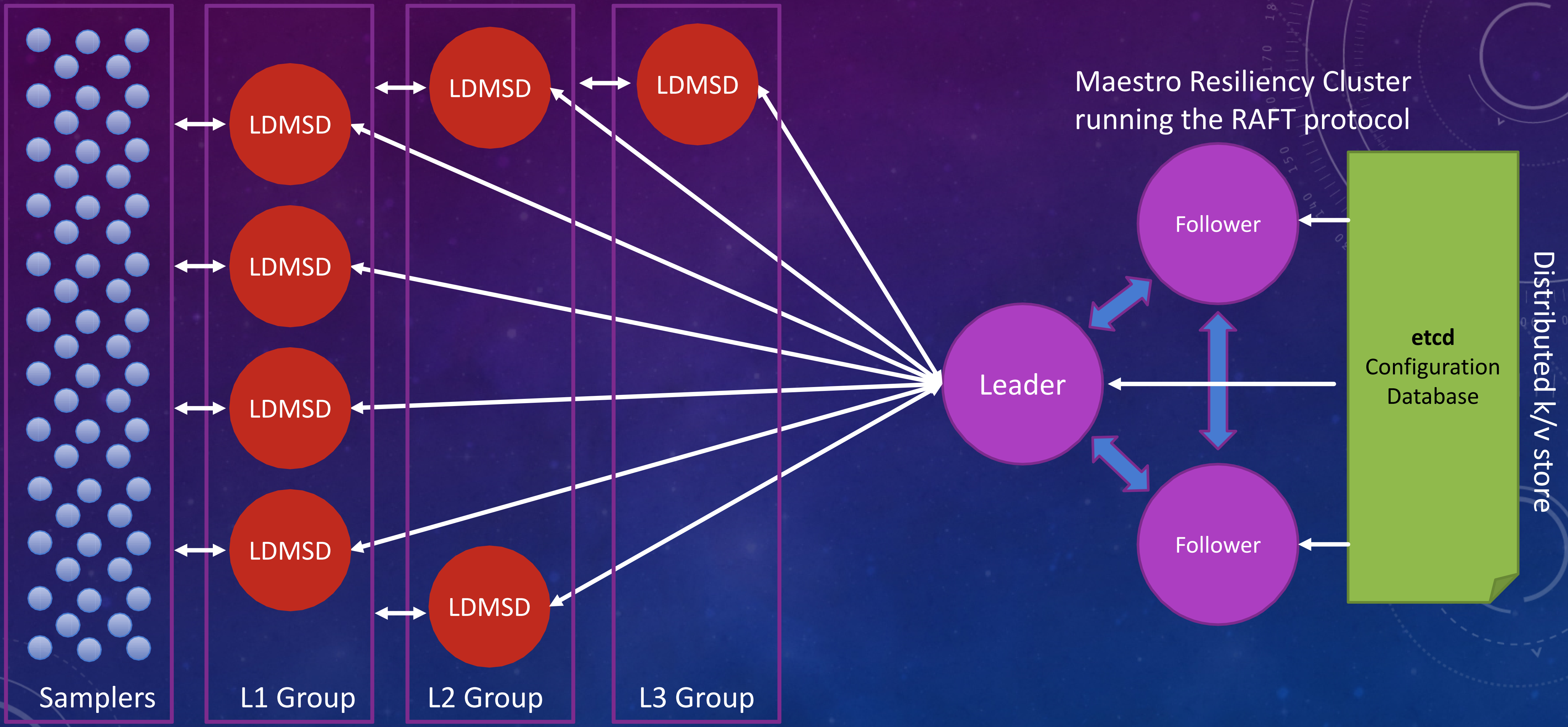


Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

LDMS MAESTRO

- Distributed configuration management
- Load balance configuration across aggregators
  - By producer or by set
- Continuous cluster monitoring
  - If aggregator fails, its configuration is re-distributed across remaining aggregators
- Maestro and database are both resilient

MAESTRO DISTRIBUTED CONFIGURATION MANAGEMENT



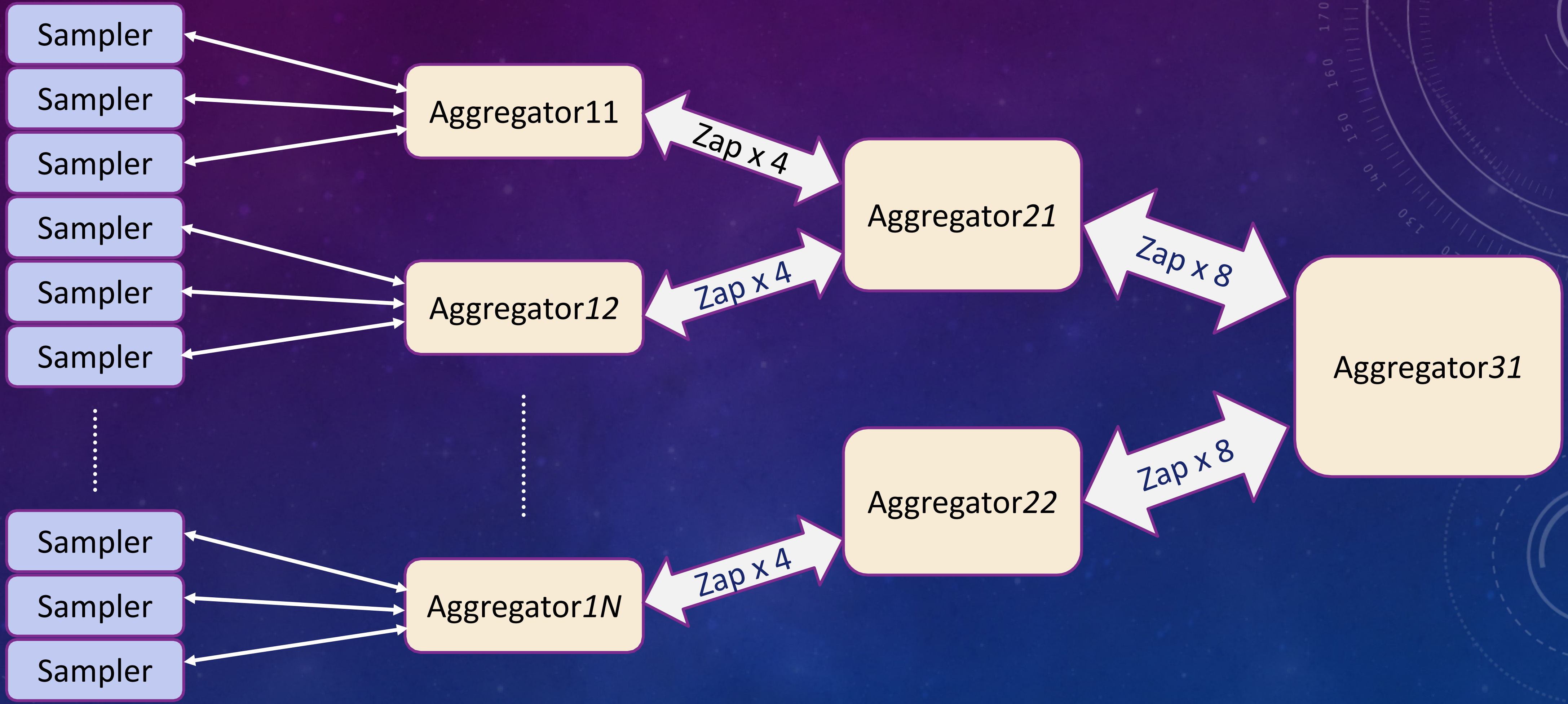
Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

MULTI-CONNECTION ENDPOINTS (RAILS)

- Motivation
  - An endpoint is assigned to a single I/O thread
  - A 1st level aggregator has an endpoint for each sampler
  - A 2nd level aggregator has an endpoint for each 1st level aggregator
  - The number of sets handled at the highest levels is far higher than the 1st level, however, there are fewer I/O threads to which the work is assigned
- Architecture
  - Support multiple connections over a single LDMS transport
  - Bandwidth and message rate flow control at the transport level
- Benefits
  - I/O bandwidth and compute are now scalable at the transport level
  - Allows administrators to flow control data injection by clients (e.g., applications using streams to publish event data)

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

RAILS


Sampler
Sampler
Sampler
Sampler
Sampler
Sampler
Sampler
Sampler
Sampler
Aggregator11
Aggregator12
Aggregator1N
Zap x 4
Zap x 4
Zap x 4
Aggregator21
Aggregator22
Zap x 8
Zap x 8
Aggregator31

KAFKA INTEGRATION

- Avro-Kafka LDMSD storage plugin
- Publishes LDMS metric set data to the Kafka bus
- Uses decomposition to normalize many *similar* schema to a single Avro schema definition, e.g.
  - meminfo.* → meminfo
  - Only figures of interest can be selected from metric sets for storage
  - Missing columns in a metric set are "filled" with default values as necessary
  - Differences in array sizes are handled by Avro "lists"
- Uses AVRO C-library to encode metric set data on the Kafka bus
- Metric set schema is stored in the 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
- Topic and schema naming are handled with format specifiers, e.g.,
  - topic = "/ldms/%F/%I/…", where %S == schema name, %F == encoding (JSON/AVRO), …

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

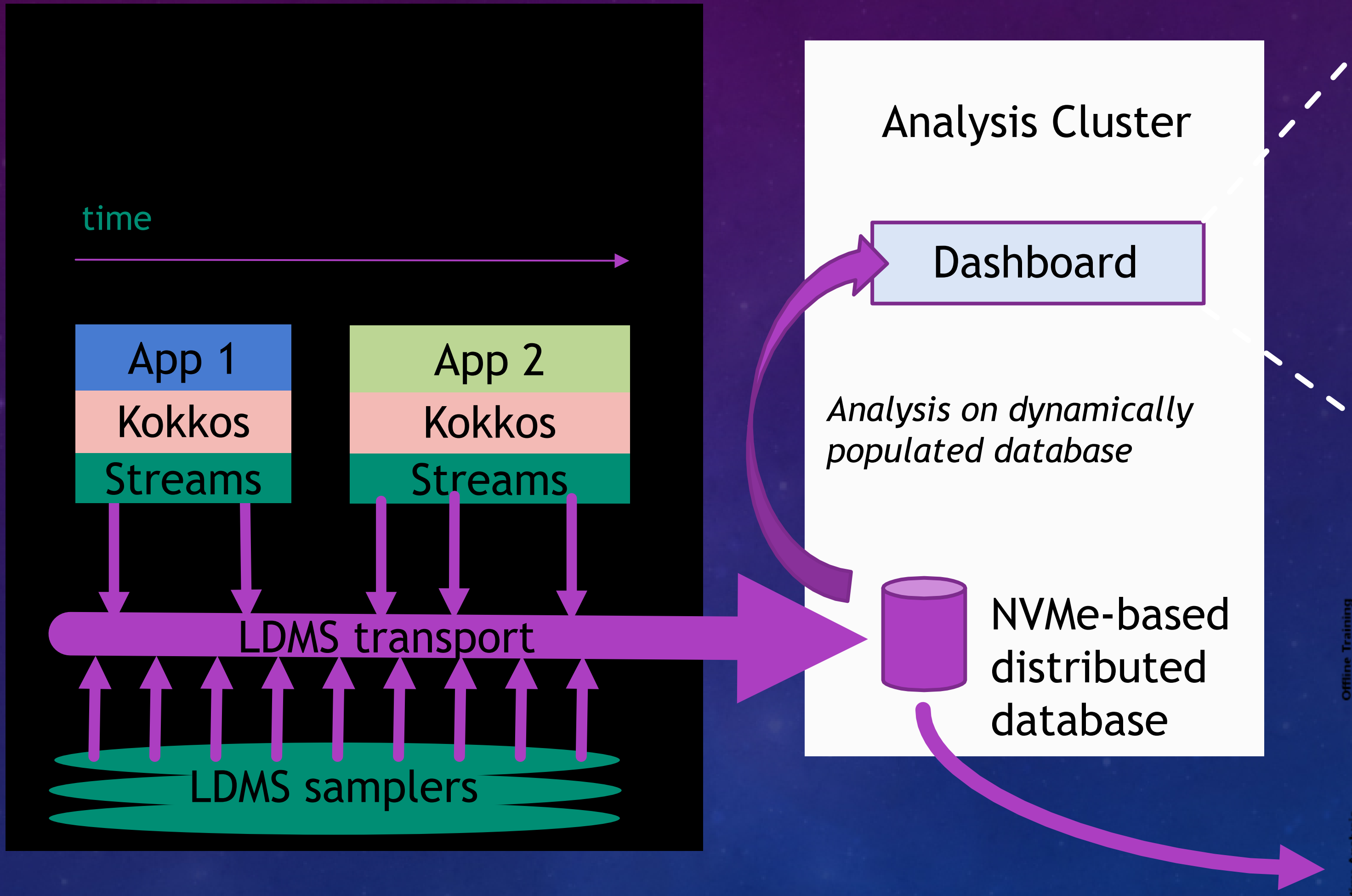
LDMS SLINGSHOT SWITCH SAMPLER

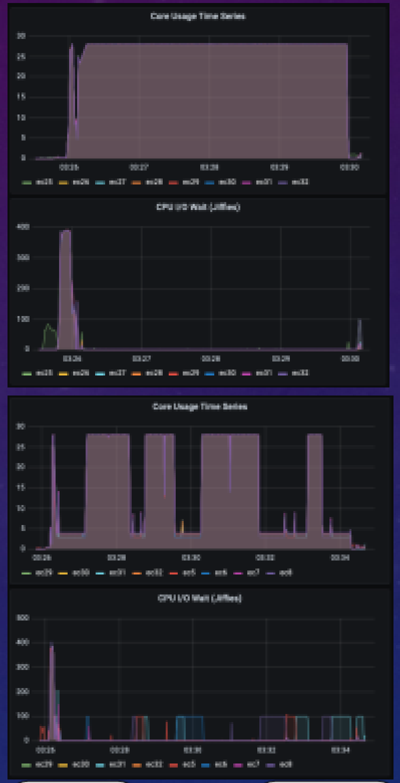
- LDMS build that runs directly on the Slingshot Switch
  - In testing put binaries in /rwfs/OVIS-4.3.11 (~7M)
- Sampler metrics configured through configuration file
  - Use the “dump_counters” binary to fetch counter values
  - Runs over socket transport on management port
  - Have not yet set up Munge authentication on switch (use none or shared secret)
  - Over 1000 metrics per port available so choose small subset

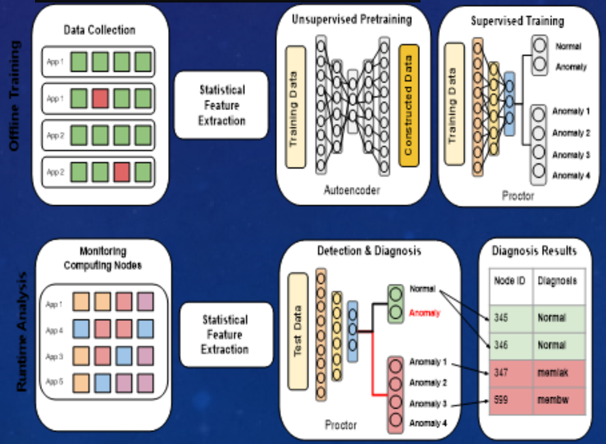
DATA-DRIVEN RESPONSE IN THE APPSYSFUSION FRAMEWORK

Applications dynamically and irregularly inject data into the LDMS transport

LDMS continuously and regularly collects and transports full system data







Sandia National Laboratories

OGC

NERSC

Hewlett Packard Enterprise

CONCLUSIONS

- Looking forward to HPE’s adoption of community developed LDMS
- HPE should upstream their libfabric over Slingshot implementation so that LLNL can utilize the LDMS “fabric” transport
- The authors will be working together to explore possible convergence on some of the identified directions and questions.
- Perlmutter is currently undergoing work for acceptance and while there are El Capitan Early Access Systems, El Capitan is not yet delivered.
  - The authors will be working together to explore best practices to scalably support the potential data flow from these extreme- scale systems.

Sandia National Laboratories OGC NERSC Hewlett Packard Enterprise

RESOURCES

- Download code and contribute through issues interaction and code contributions :
  - https://github.com/ovis-hpc/ovis
- Join the LDMS User group bi-weekly meeting:
  - https://github.com/ovis-hpc/ovis-wiki/wiki/Mailing-Lists
- LDMSCON2023 being held June 13 – 15, 2023 in Boston:
  - https://sites.google.com/view/ldmscon2023

QUESTIONS?