Sandia National Laboratories

Exceptional service in the national interest

HPC Application Performance Monitoring and Feedback with LDMS

Presented by Ann Gentile (SNL)

Representing the LDMS developer community,
SNL's LDMS and AppSysFusion teams.
Including material from LDMSCON and other

SAND2024-10658PE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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.

Outline

- LDMS – Lightweight Distributed Metric Service
- Feedback for Improved Computing Efficiency
- Enabling Feedback: LDMS Scalable Event Transport

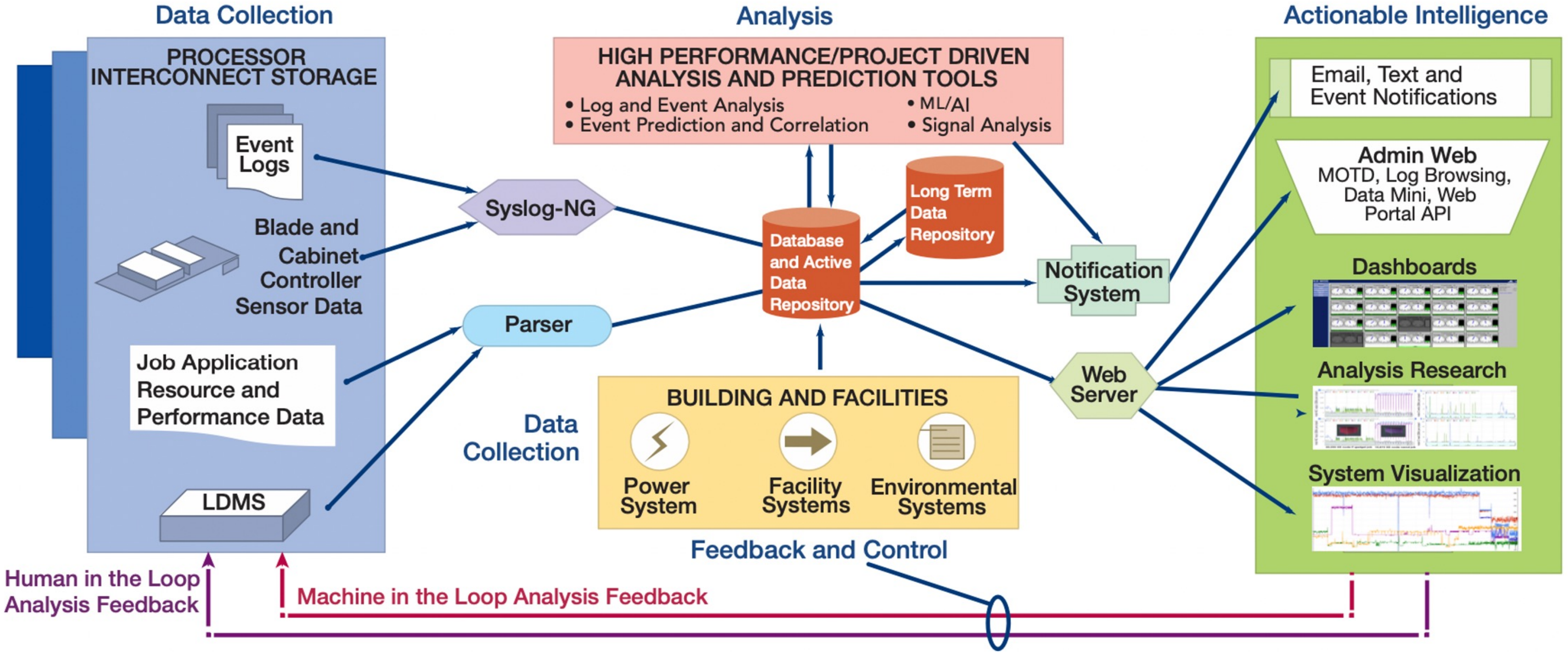
You shouldn't operate a system like a black box!

LDMS

BLUF

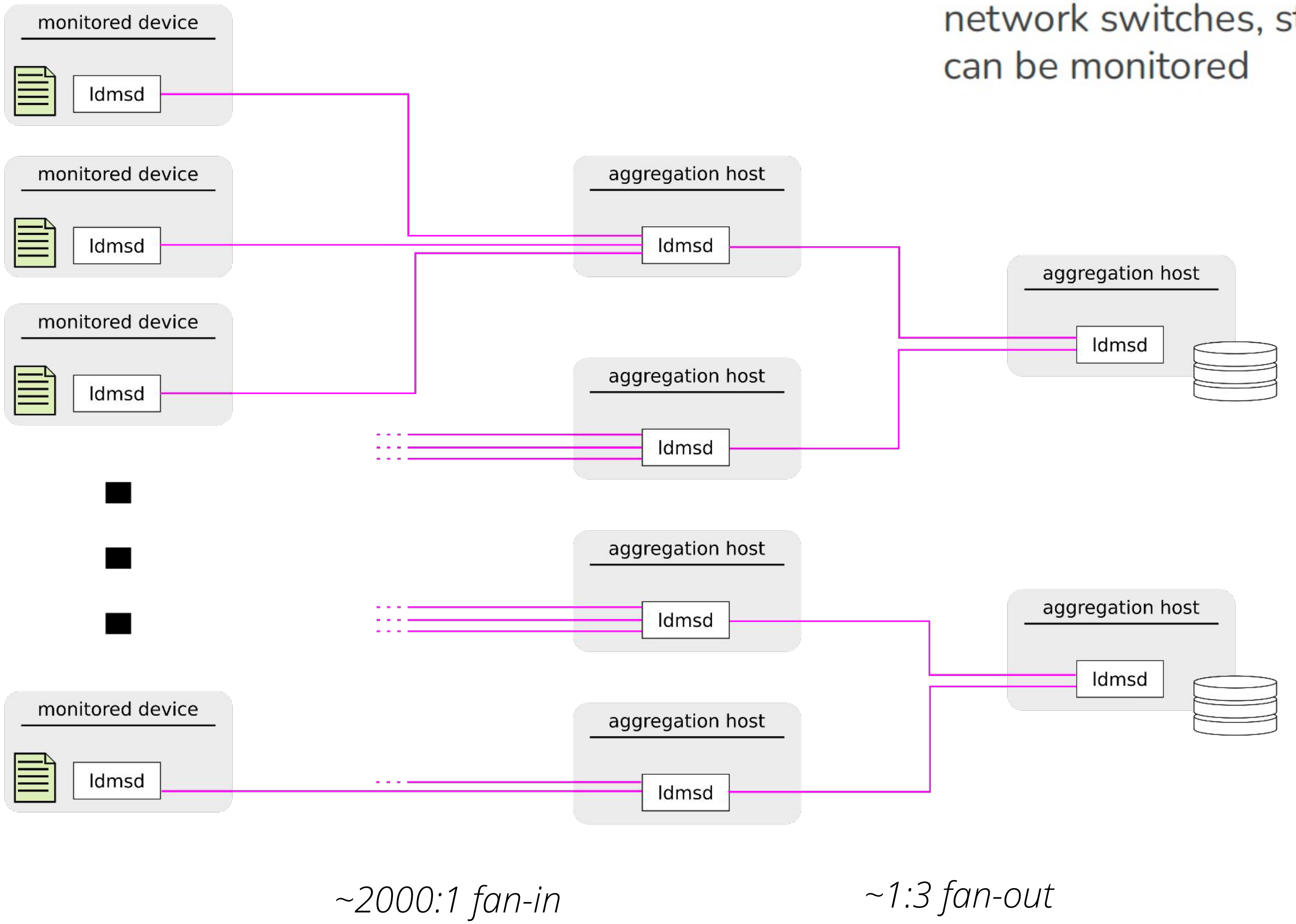
LDMS: designed for global collection of high-fidelity data and run time analysis, feedback, and response

- **Lightweight:** LDMS enables lightweight data collection and transport with no statistically significant negative impact on application performance
- **Resolve features of interest:** LDMS uniquely designed for collecting and transporting a lot of data, often
- **Respond:** Global, multi-directional transport enables analysis feedback to applications and system software



LDMS Deployment Overview

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

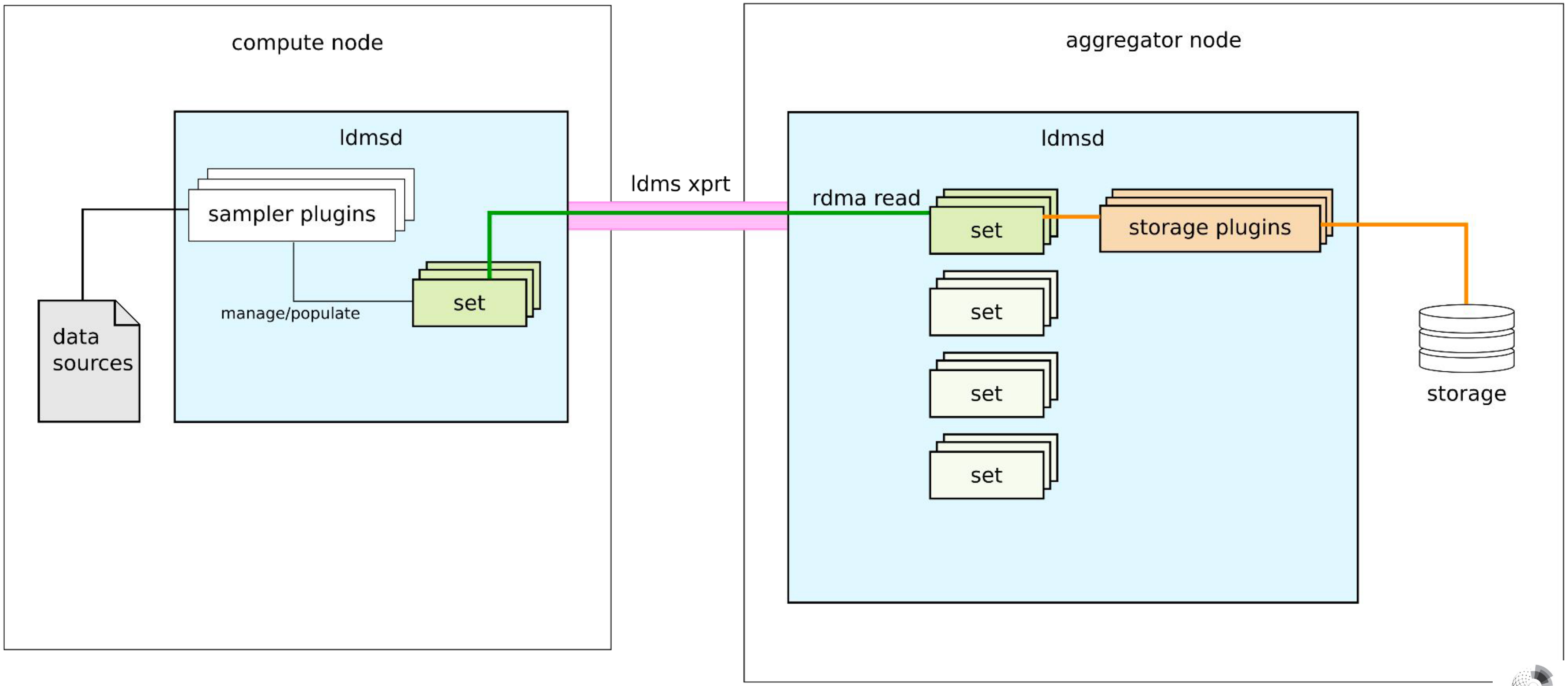


Per monitored device 2-3K metrics/second. For 10K node system, ~2Gb/sec aggregate across whole network

Images OGC LDMSCON2024

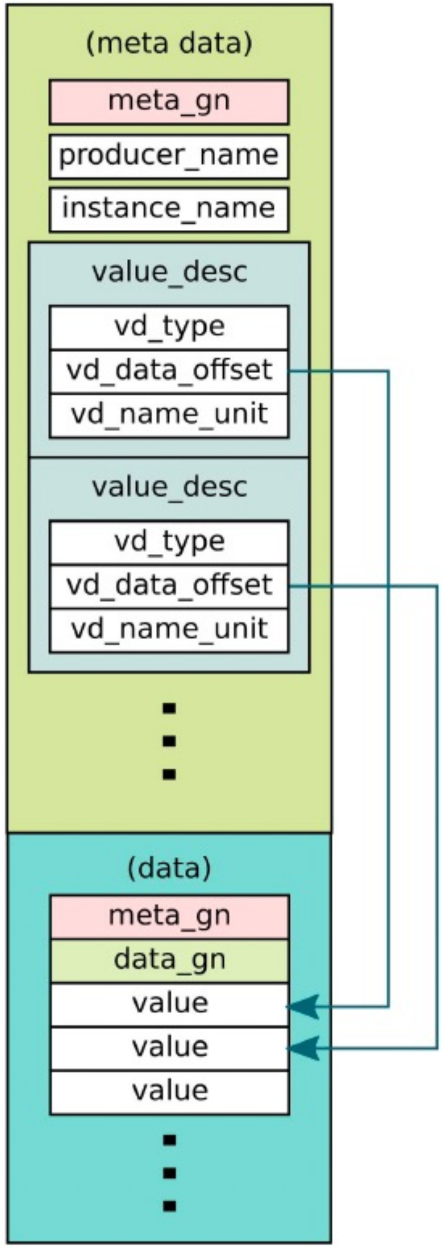
Transport Modes: Lightweight regular *pull* of metric data

- Optimized memory organization for metric sets: only transport data, not metadata, each time
- No CPU intervention/overhead on RDMA read



Images OGC
LDMSCON2024

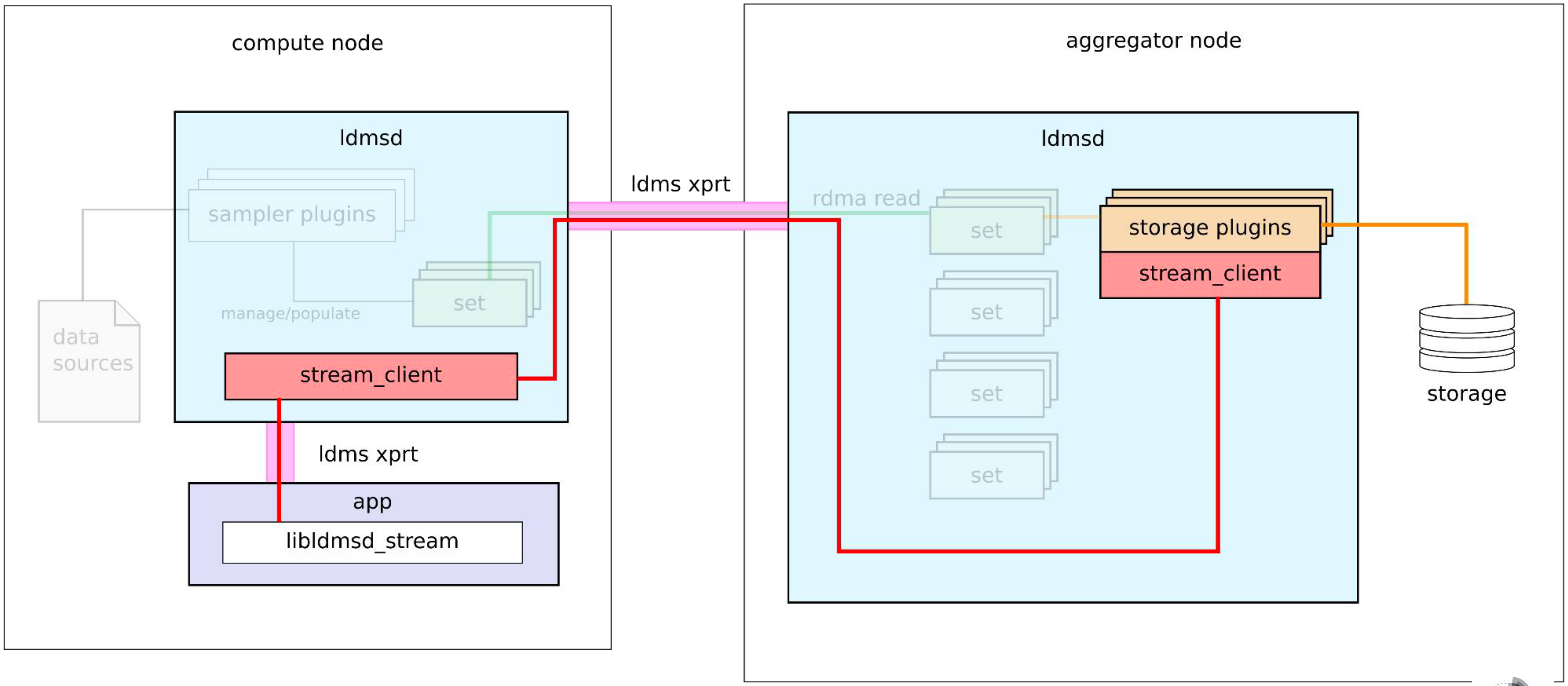
Design for Lightweight Transport



- Metric sets are self-describing
- Metric set memory organization (fixed footprint)
  - MetaData – largely stable values
    - Generation number
    - Set Schema: metric names, value types, units
  - Data: metric values. Updated at sample intervals
- Limit access by UID, GID, and permission bits
- Transfer protocol:
  - Only the data section
  - MetaData only upon change
  - RDMA read and memory map for transport
    - No CPU intervention/overhead on RDMA read
    - Pull-based reduces the on-node requirements

Transport Modes: Event-driven *push* of json/string data

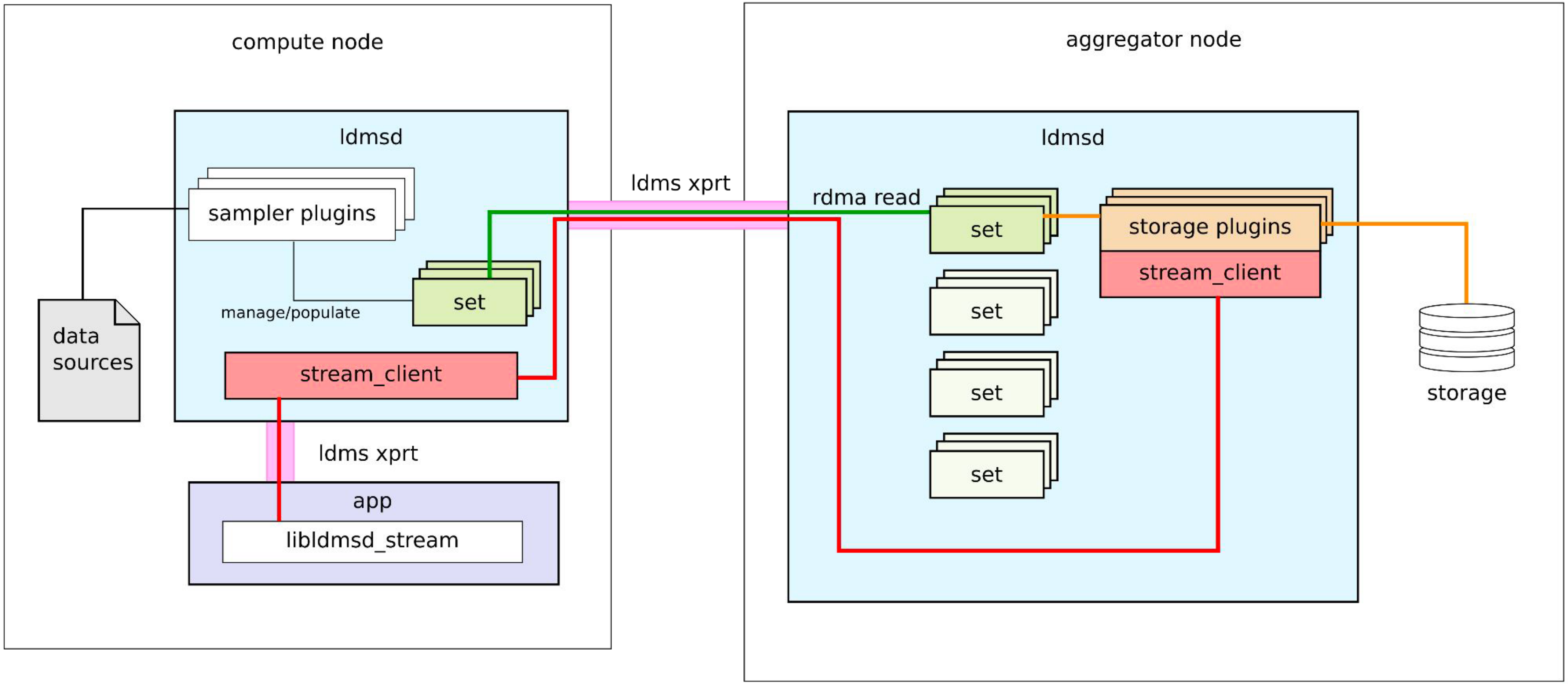
- Application/connectors select and pack event data



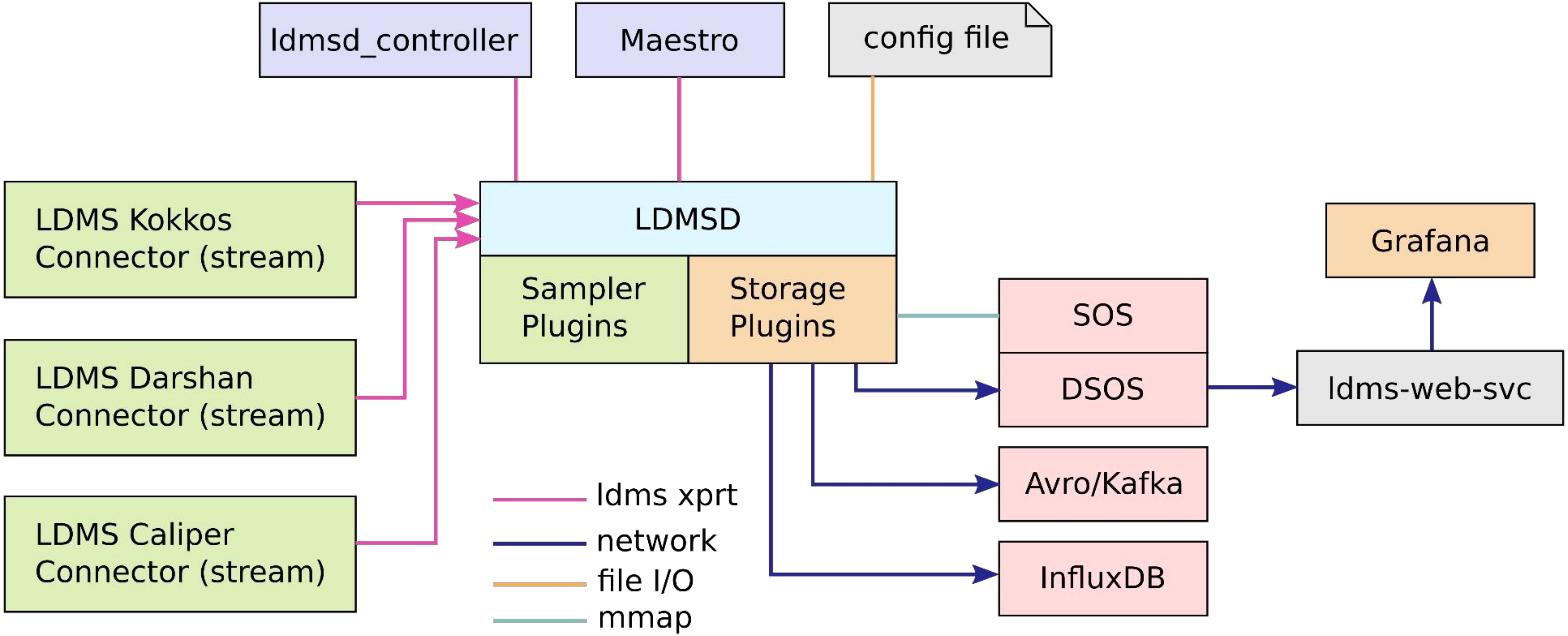
Images OGC
LDMSCON2024

Always-on Application+System Collection, *Feedback, and Response*

- **Always-on:** Build profiles of *at-scale, in the wild* behaviors
- **Run time data availability:** Insights and responses enabled when/if problems occur.
- Transport also functions as a bidirectional pub-sub bus – ***can also push back to applications!***
  - Easy to publish back into the cluster off-cluster analysis results via the existing monitoring plumbing



LDMS Ecosystem



Provides run time transport for interoperable tools as well

Images OGC
LDMSCON2024

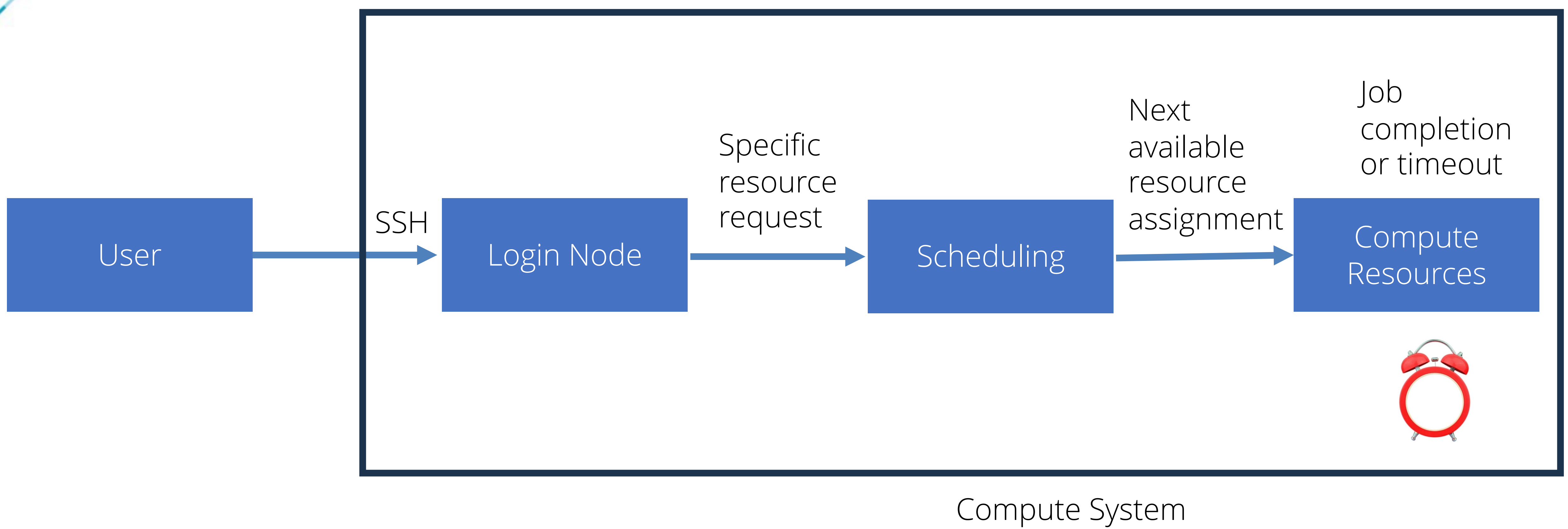
Transports	Sampler Plugins	Store Plugins

• 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
	  • Memory usage
	  • Network bytes/packets read/written etc
	  • File system bytes read/written
	  • PAPI counters
	  • Facility resources
	  • and more
	• Application Information
	  • Job information
	  • Kokkos
	  • Darshan
	  • Caliper
	  • And more
		• CSV
		• Avro/Kafka
		• InfluxDB
		• SOS
		• Victoria Metrics (underdevelopment)

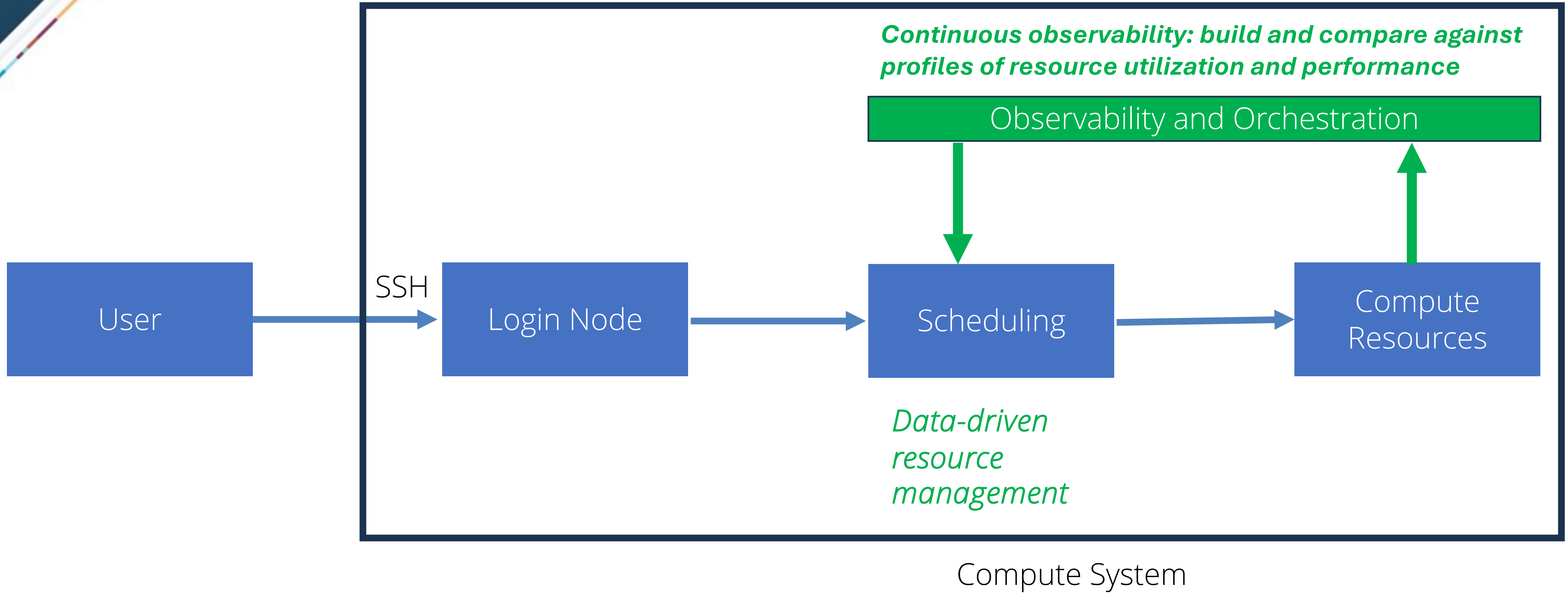
Feedback for Computing Efficiency

Nobody collects data just to collect data!

Current HPC Operations

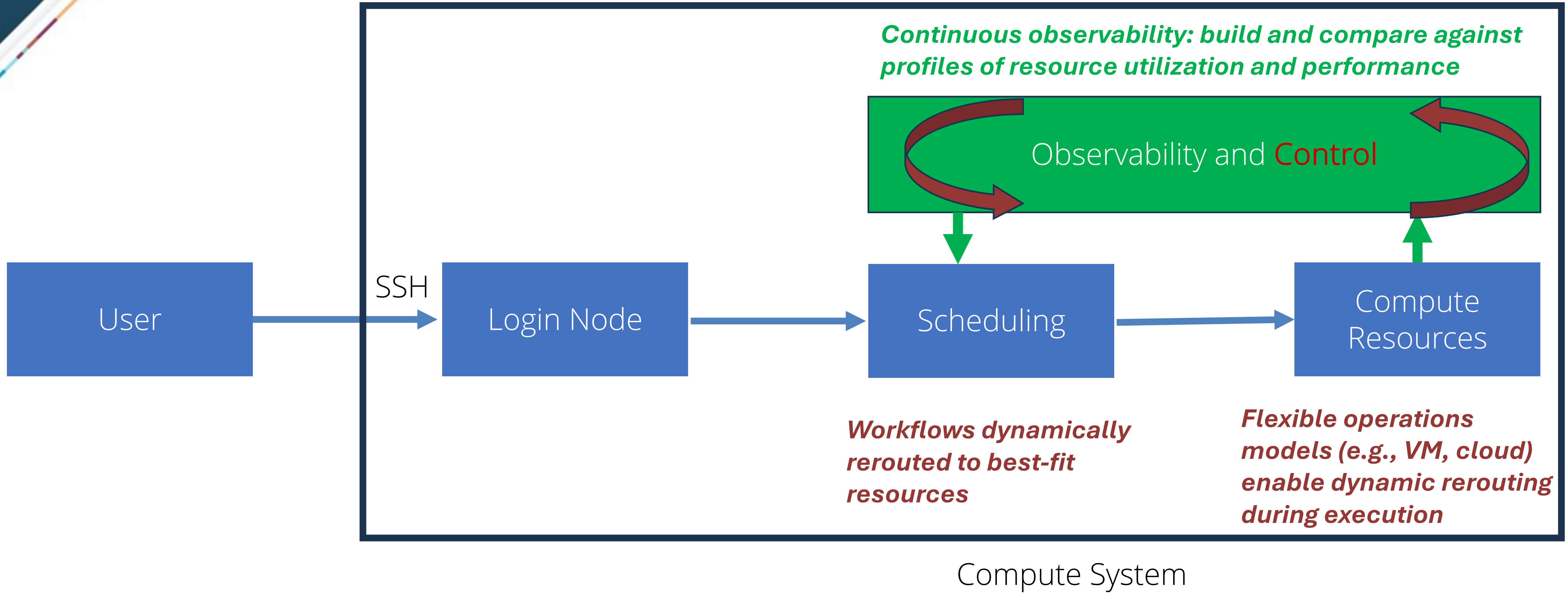


Data-driven Operations: Static Best-fit



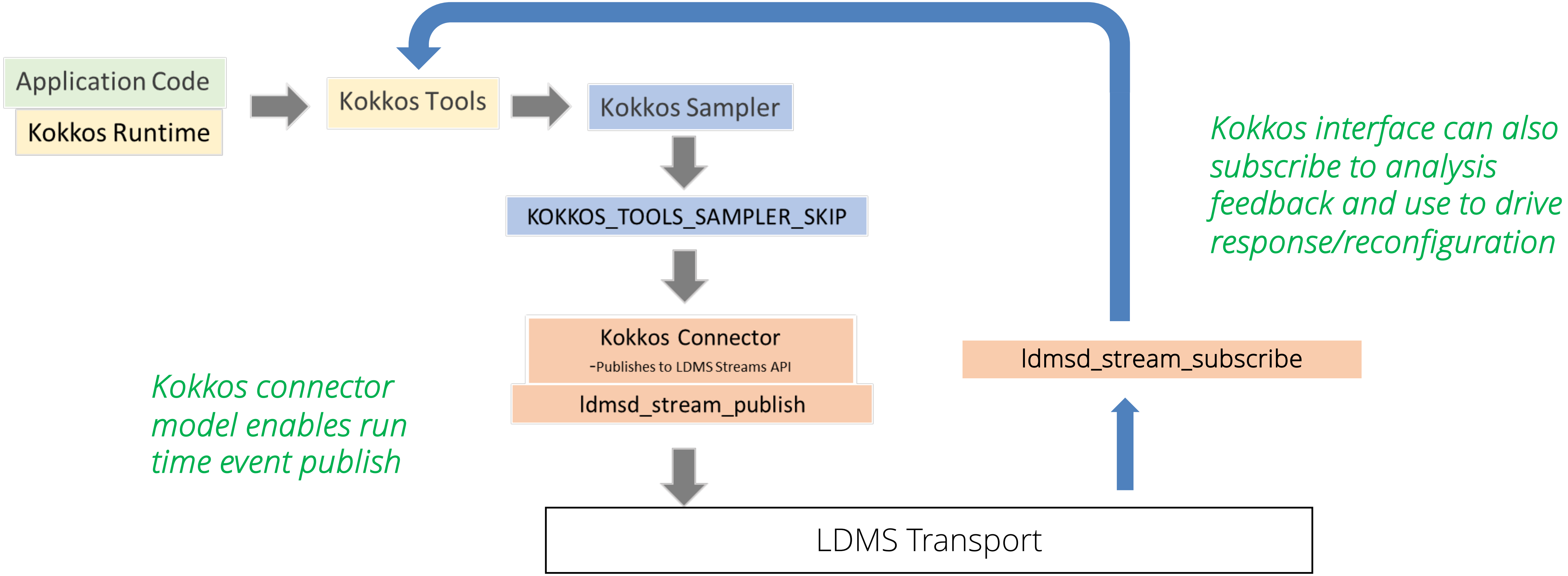
Feedback to system software

Data-driven Operations: Dynamic Best-fit



Feedback to system software

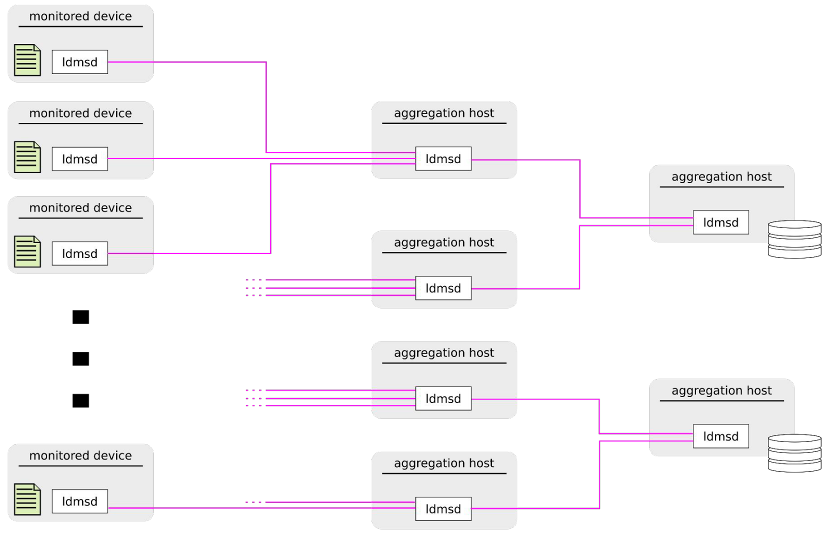
Application Response/Reconfiguration via LDMS-Kokkos interaction




```
#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
```

New Design for Scalable Event Transport

Scalable Event Transmission: Direction Matters



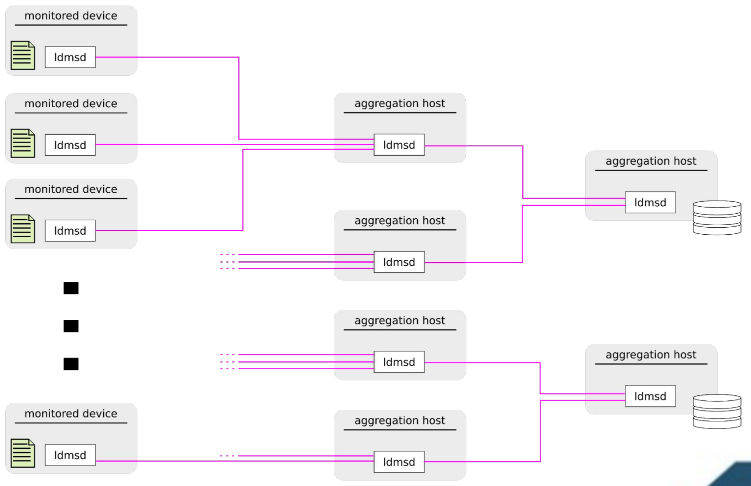
- **Not the common use mode of a pub-sub bus:** Using the transport bi-directionally with very dynamic and finite-lived applications as publishers and subscribers
- Applications publishing progress/performance data to local LDMS daemon **scales as the number of nodes allocated to an application** and incurs the overheads:
  - Formatting and publishing (low per-message cost per compute node but potentially high in aggregate for large frequent messages)
  - Network bandwidth (<<< available HSN BW)
  - Unpacking and storing (high but can scale out on monitoring cluster)
- **Analysis cluster sending feedback**/control messages to application processes **currently not scalable** because ***current static* Streams subscription model implies all feedback goes to all subscribing *processes:***
  - Current design driven by collection of a few well-known event sources (initially slurm), not feedback: Static subscription model
  - Filtering of messages would be on a per-process granularity (potentially high overhead because of large numbers of interrupts)
  - Potential for blocking at compute node LDMS daemon if processes don't handle interrupts fast enough
  - Though all interrupts have to be handled most will be ignored by most processes

Low-Overhead Event Transmission: Event Frequency & Encoding Matters

- Currently published **JSON encoded events** include full metadata in every message
  - **Message size can be substantial** for events with long associated names (can be KBs)
  - Network bandwidth can become substantial for frequent events with large meta-data
  - **Possibility for separation of MetaData and Data, similar to the design of the Metric Set**
- Events can be frequent (sub millisecond) depending on how applications are instrumented and what methods are used for selection of events to publish
  - **Encoding overhead can become substantial for high frequency events**
    - Pass-through nature of non-storing LDMS daemons means that publishing should not be a bottleneck even for ~100s core processors
  - New LDMS **Streams credit-based flow control** can render much of this encoding overhead a waste as the messages may not have credits for publication
    - If too many messages for available credits, publisher decides if hold in queue, best-effort etc
    - Root user has no constraints

Event and Feedback Message Latency and Throughput Considerations

- **Event message latencies** (generation to arrival at an analysis cluster) only matter in the context of the window of opportunity for modifying the behavior they might reveal
- **Event message throughput** dictates the maximum amount of event data available for analysis and hence the fidelity of the data and results
  - This will be bounded by acceptable event processing overhead and network bandwidth available to a particular process given all other processes concurrently competing for LDMS Streams network credits
- **Feedback message latencies** also matter in the context of the window of opportunity for modifying behavior that run time analyses have identified as needing to be changed
  - No credit-based restrictions on this
- **Feedback message throughput** is not expected to be an issue
  - Expected to be very infrequent and small
  - These messages are not flow controlled



New Design for Scalable Event Transmission

New design plans:

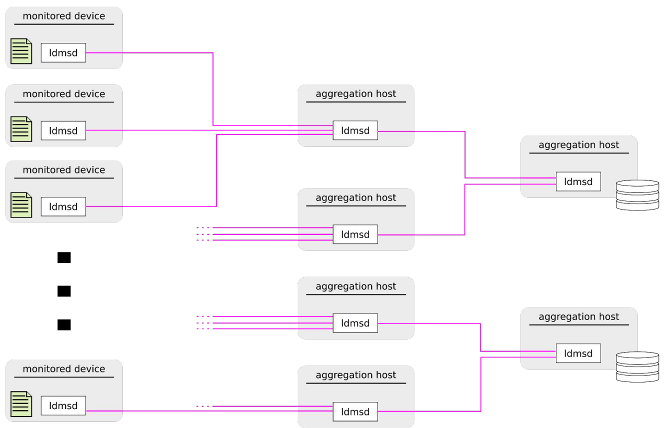
- Enable an authenticated user to **dynamically push subscription to a new stream name** all of the way from publisher to storage consumer
- Enable an authenticated user process to **subscribe all of the way from the analysis endpoint** to the subscribing consumer
- Enable authenticated user processes to **tear down all subscriptions** established on their behalf
- Utilize **AVRO binary encoding of LDMS Streams data** to reduce network impact
- **Shim layer that facilitates** and enables setting bounds on how often a particular event can be published and enables user defined representation of data collected over an interval for a given event
  - (e.g., first event info. and timestamp, last event info., number of events since prior published event, and last event info.)

Key features:

- Freedom of users/applications to create and publish new event types. Reduced administrator intervention
- Simpler LDMS deployment configuration
- Reduced network overhead
- Reduced compute node processing overhead

Enabling run time, analysis-driven feedback:

- Feedback channels can be defined on-the-fly for analyses being performed during run time (data processed on arrival to analysis cluster)



Conclusion

- LDMS: designed for global collection of high-fidelity data and run time analysis, feedback, and response
- Feedback provides opportunities for improved computing efficiency
- New Design for Scalable Event Transport overcomes challenges enabling feedback to application processes

LDMS Open Source: https://github.com/ovis-hpc/ovis

For more info: https://ovis-hpc.readthedocs.io/en/latest/

LDMS Users Group Conference: https://sites.google.com/view/ldmscon2024