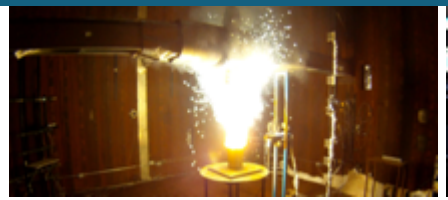
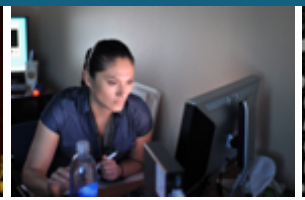
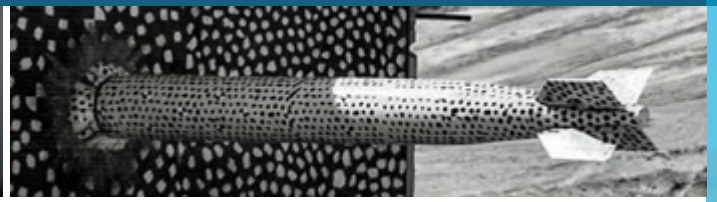
Sandia National Laboratories

SAND2021-5819PE

Integrating Systems Management into CoDesign







Ann Gentile (SNL) presenting

ASC PI Meeting 2021

SAND

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.

Incorporating Systems Management into CoDesign

CoDesign has been about the **HW and SW** being designed together to enable efficient performance through leveraging features of each.

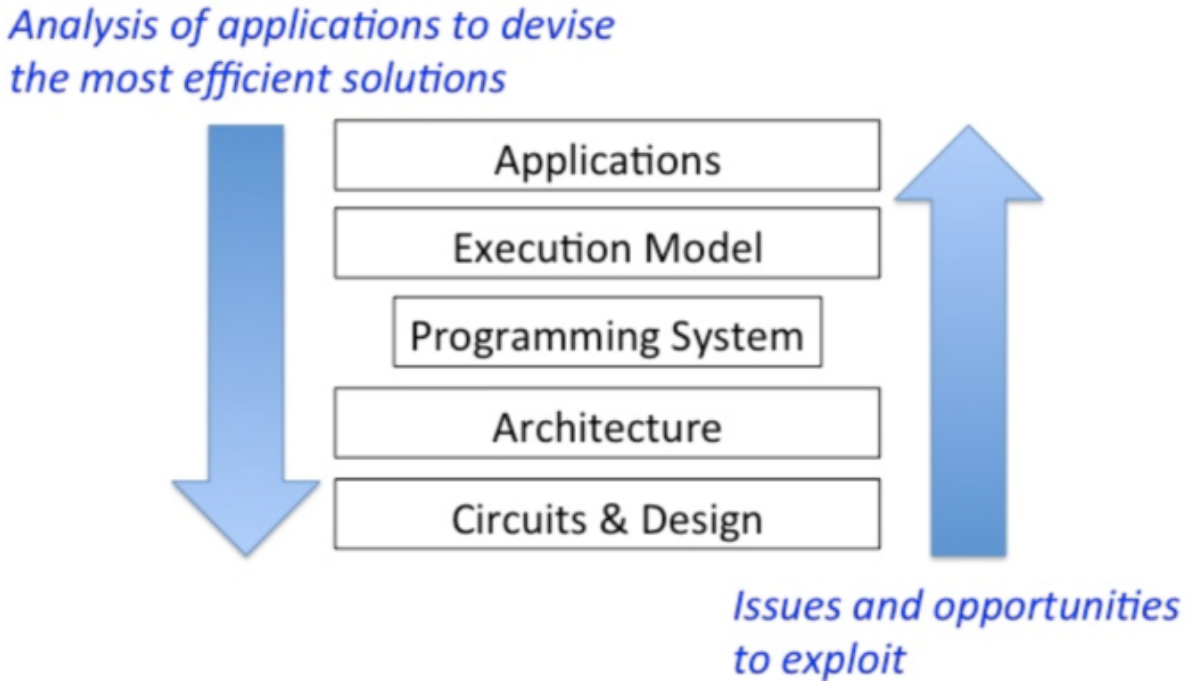
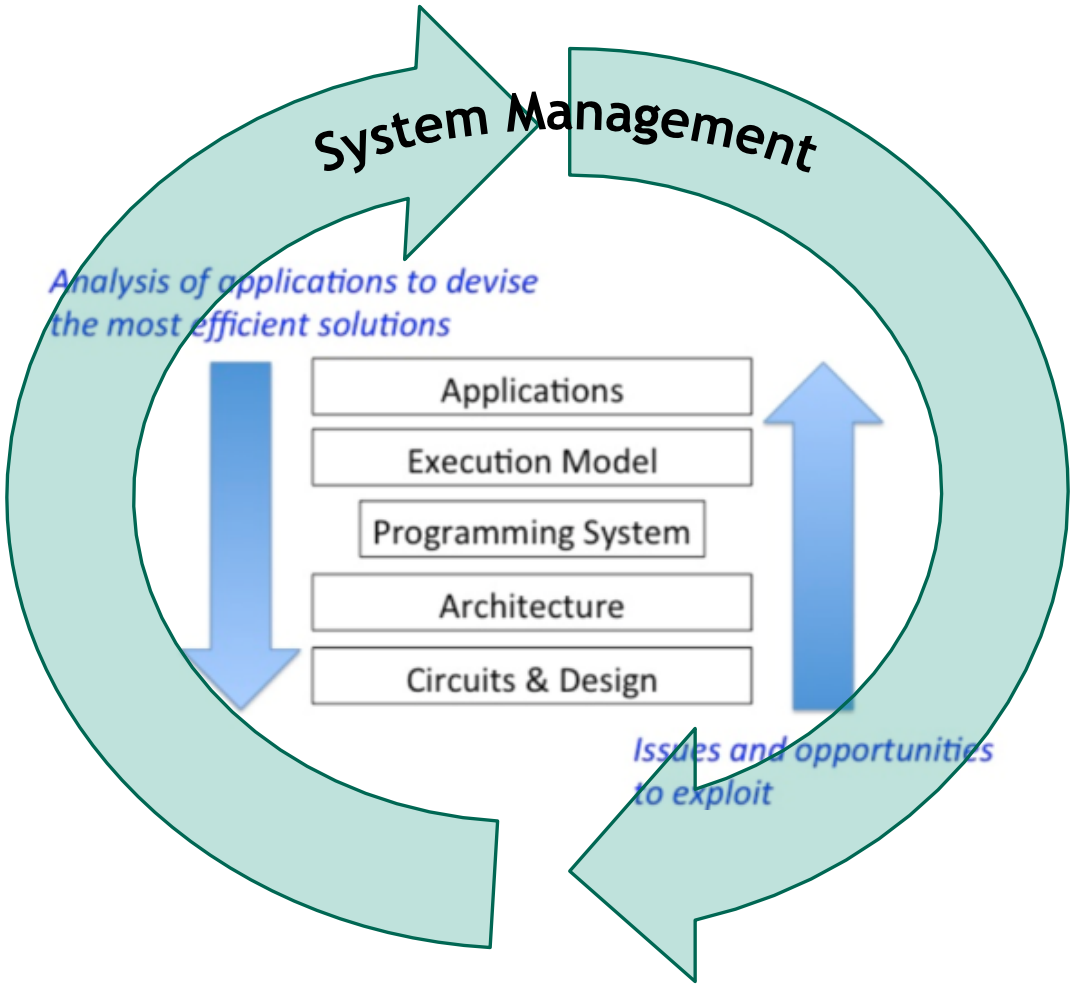


Figure from "On the Role of Co-design in High Performance Computing", Barrett et al, 2013

…but even the best CoDesigned system+application will perform badly if the system is not managed with consideration of applications and architecture!

Incorporating Systems Management into CoDesign



Dynamic System Management based on architecture, workload, application demands, resource state. Examples:

- Co-scheduling and allocations based on application demands and architectural support
- Application remapping due to changing requirements or contention

HPC System Management Ideals:

- **Data-driven** -- We shouldn't operate systems like a black box
- **Continuous** -- Systems management does not end at job launch
- **Holistic** – System/application doesn't exist in a vacuum
- **Autonomous** -- Human in the loop is inefficient

Revolutionizing Systems Management

Needed capabilities:

- Continuous holistic data collection
- Large-scale low-latency analytics
- Feedback channels for and response mechanisms to conditions of interest

These must be first class citizens, peer to the rest of the codesigned capabilities, and built in from the ground up

Approach - Develop and deploy capabilities on current systems:

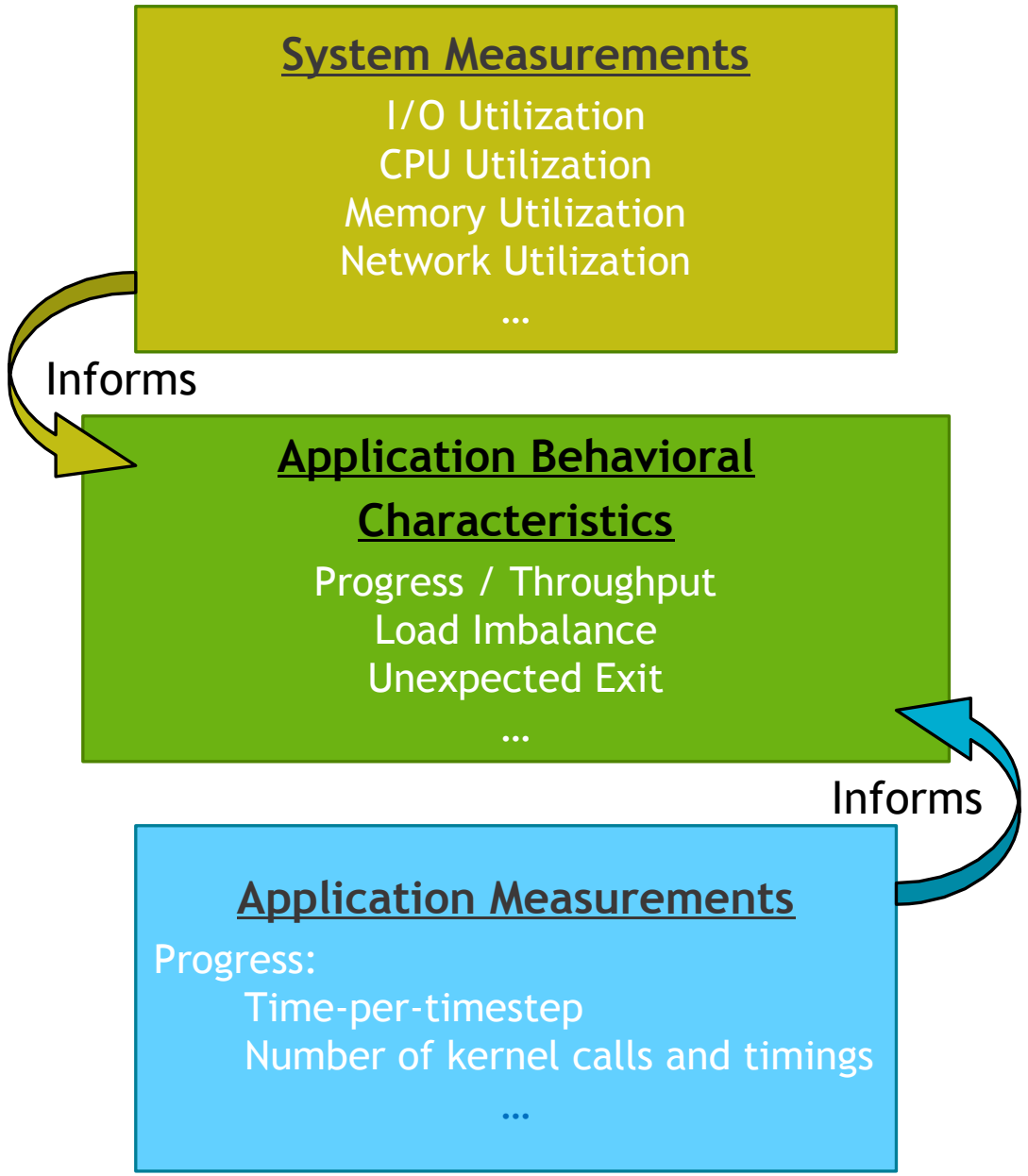
- Maximize performance/throughput of current systems and applications
- Enable data-driven next-generation design

Understanding Performance

The **dynamic nature** of system state over the time of an applications execution **makes effects** on application performance **difficult to quantify**

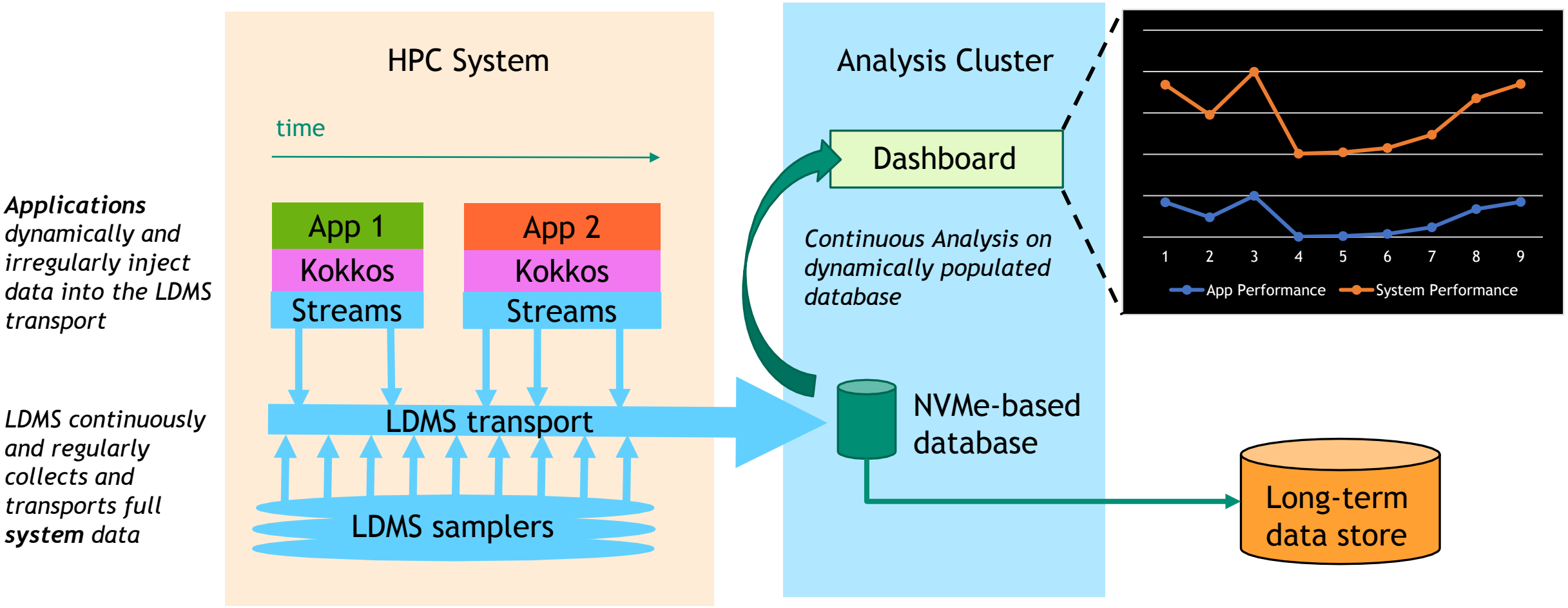
Fusion of system and application state and performance metrics can provide insights into application behaviors:

- **Temporal association** of application progress with changes in system resource state
- **Location** (e.g., spatial, temporal) of **behavior of interest** can be **expedited** through **examination** of an application's **run time progress**
- **Quantify relationships** between application performance and **degree** of system resource **contention**



FY21 ASC L2 Milestone:
Unified Framework for **Continuous, Run Time, Fused** System and Application Data and Performance Analysis

Data Flow Diagram



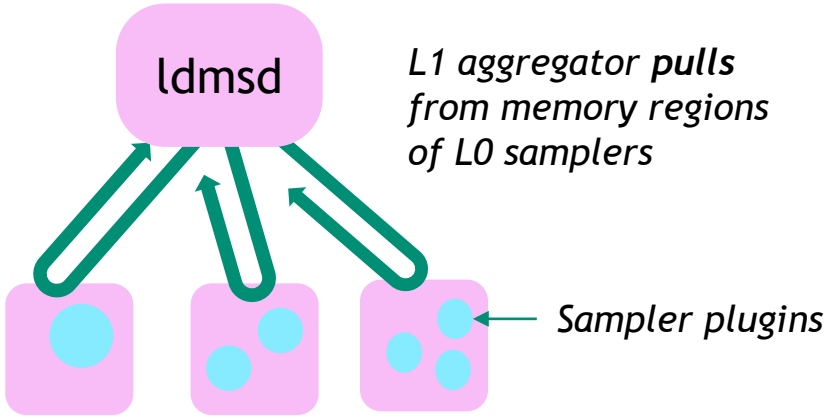
Applications *dynamically and irregularly inject data into the LDMS transport*

LDMS continuously and regularly collects and transports full ***system*** *data*

Enabling Application Data Injection via LDMS

LDMS - **low-overhead** data collection, transport, and storage capability designed for **continuous monitoring** supporting **runtime analytics** and feedback.

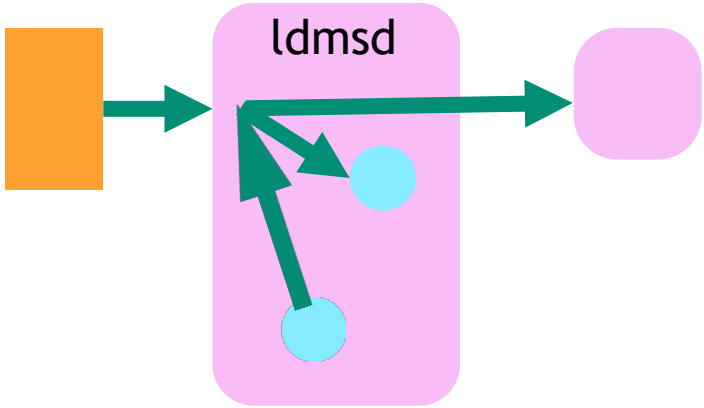
- System data collection is typically **synchronous** at regular (e.g., second or less) intervals
- **Structured** data format (i.e., metric set) designed to minimize data movement
- Transport is typically **pull** based to minimize CPU interference
- Transport to multiple arbitrary consumers over both RDMA and socket



GOAL: Leverage the efficient and secure LDMS transport to support Application Data Injection

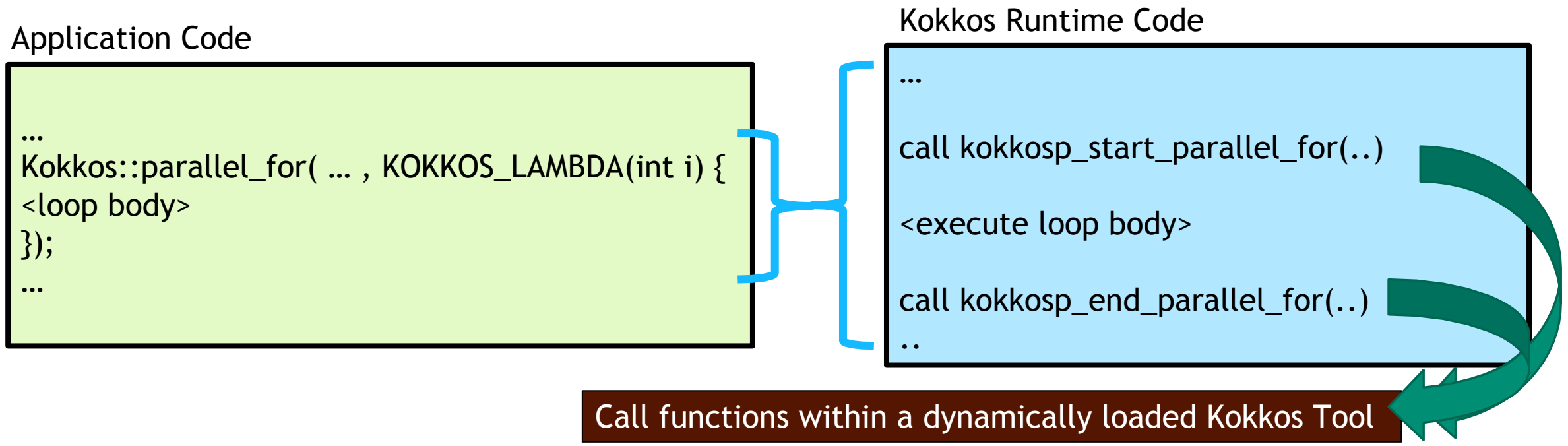
LDMS Streams – **on demand publication** of loosely formatted information to subscribers

- Transport is **push** based and supports **asynchronous** event data (e.g. scheduler and log data)
- **Unstructured** data



*Daemon publish API called from externally or by a plugin **pushes** to ldmsd which pushes to all subscribing plugins and aggregators*

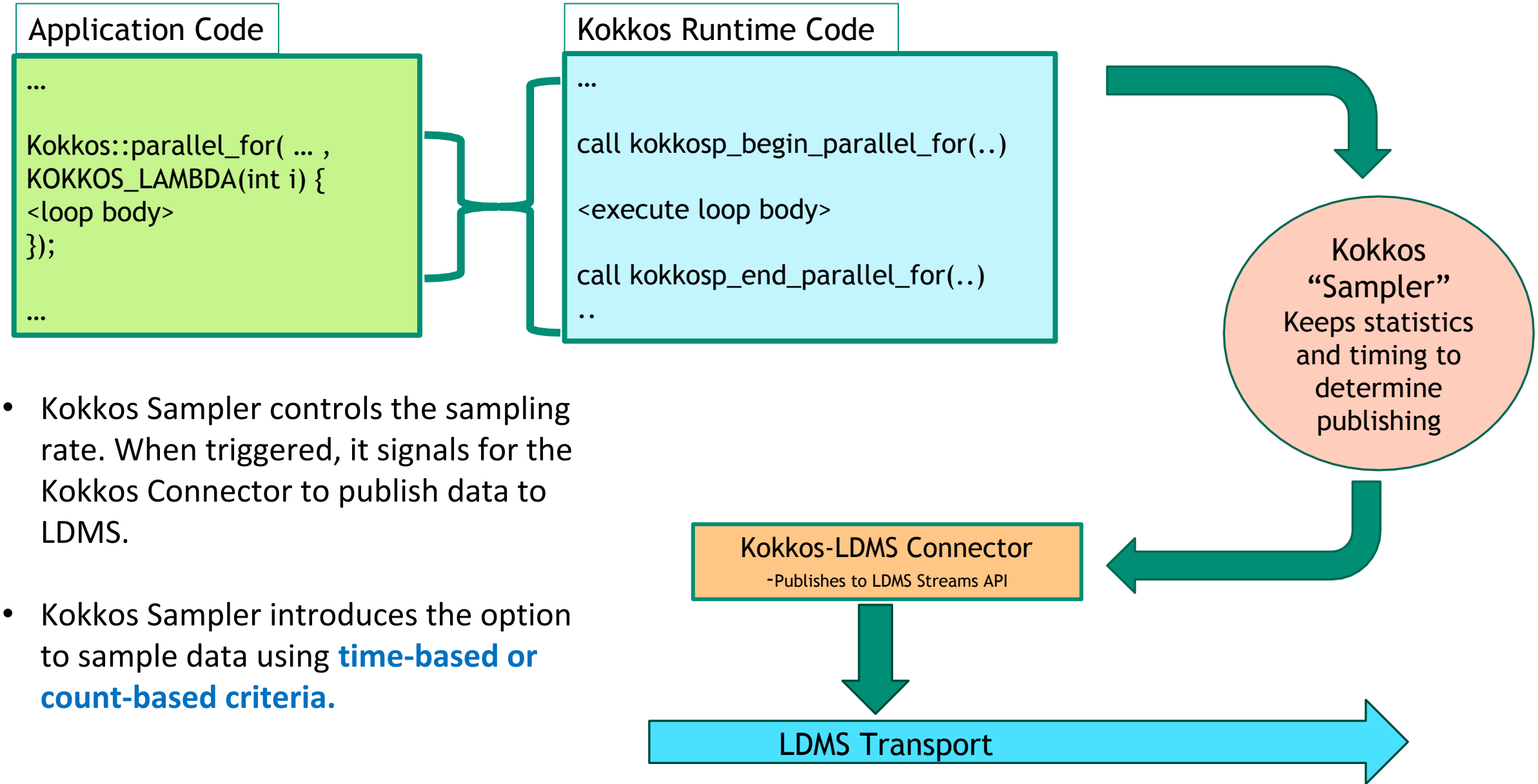
Kokkos Performance Portability Layer: Background



Exposing application data via **Performance Portability Layer** enables general data availability without application modifications

- Kernels and Teuchos timers within Trilinos are configured to **dynamically load a Kokkos supplied "connector"**. This requires **no recompilation** for profile enabled code and can be used for any Kokkos application (not just Trilinos, EMPIRE, etc.)
- Hook points already exist for **kernels** (parallel-for, reduce, scan), **"regions"** (arbitrary points in code which can stack) and **"sections"** (arbitrary points in code which may overlap)
- Already have a good idea of what the **valuable profiling information** would be (**doesn't require user input**)

Run time Injection of Application Data into the LDMS Transport



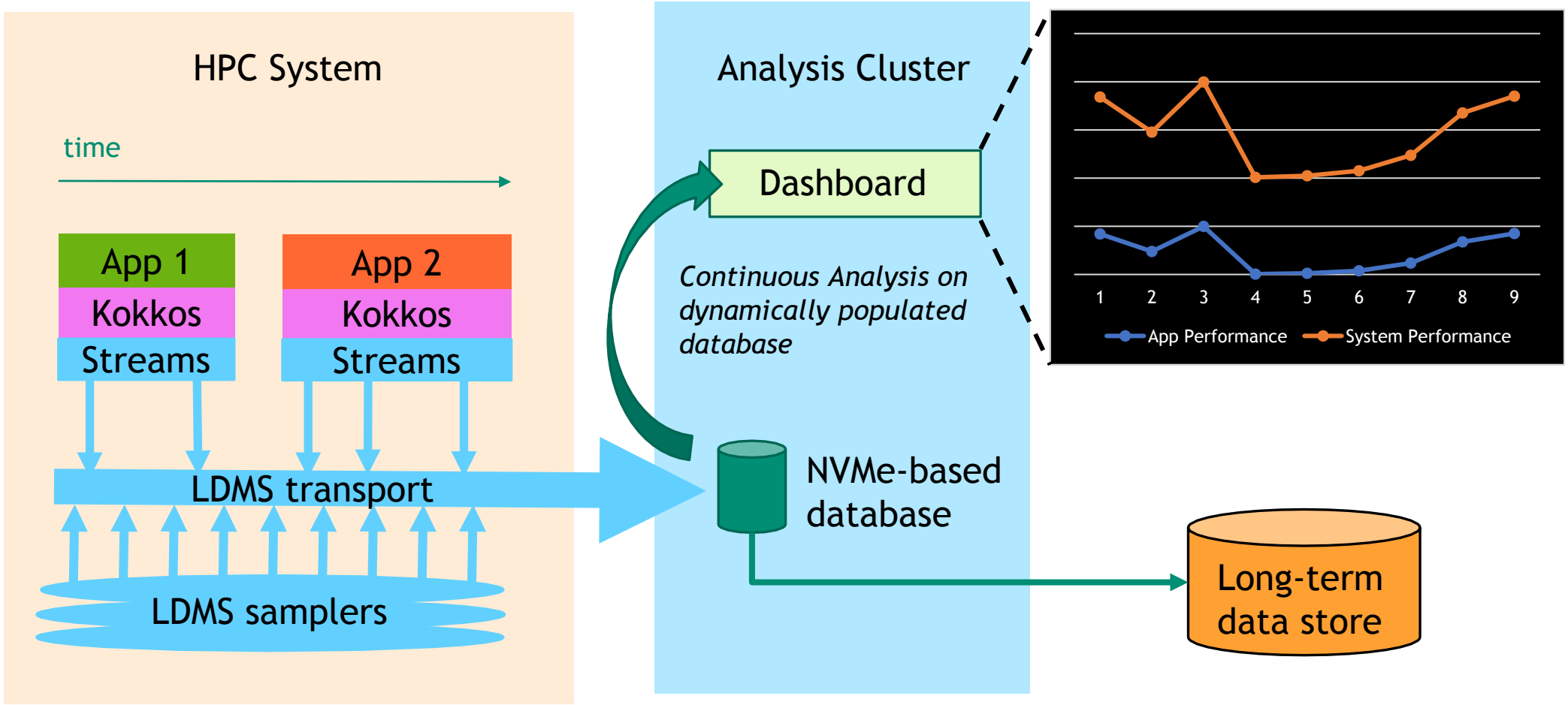
- Kokkos Sampler controls the sampling rate. When triggered, it signals for the Kokkos Connector to publish data to LDMS.
- Kokkos Sampler introduces the option to sample data using **time-based or count-based criteria.**

Unified Framework for **Continuous, Run Time, Fused** System and Application Performance Monitoring and Analysis

Data Flow Diagram

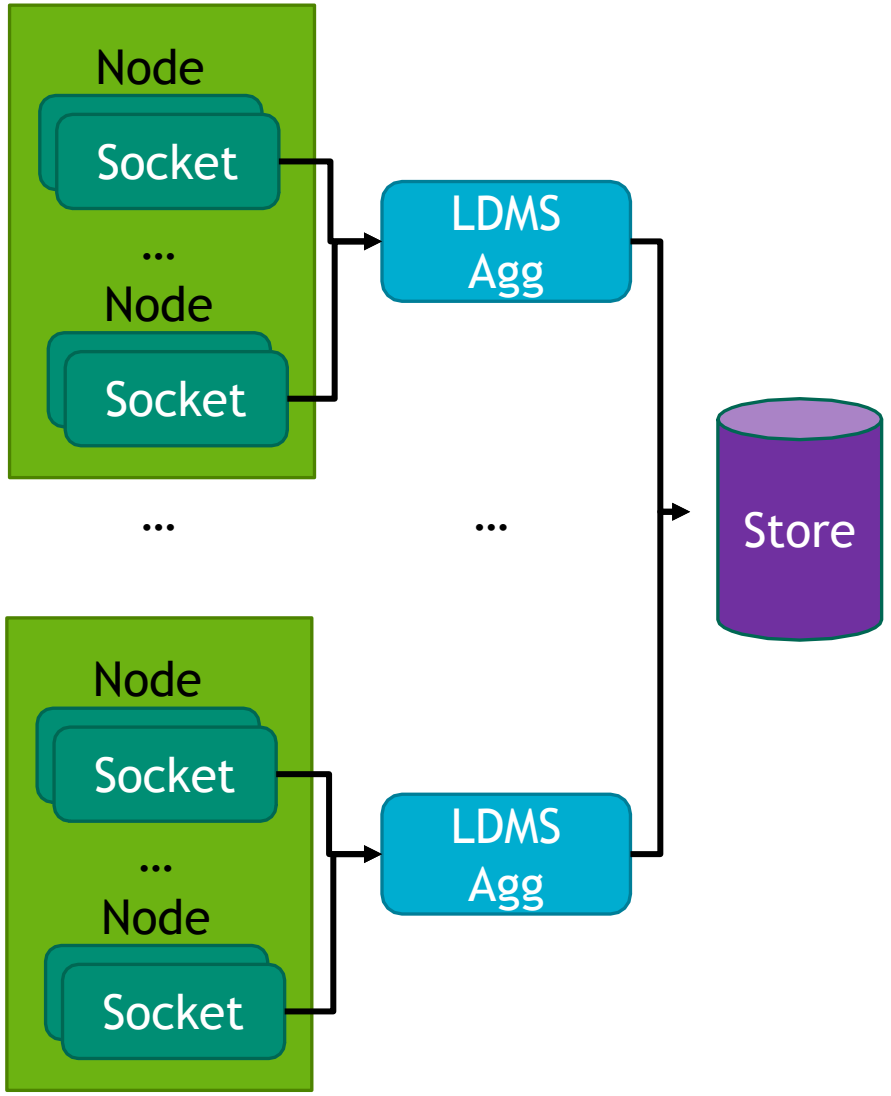
Applications dynamically and irregularly inject data into the LDMS transport

LDMS continuously and regularly collects and transports full system data



Application and LDMS Configuration

- Eclipse (CTS1) ~1500 nodes
- Target recording **~1% of kernel execution events** (e.g., one or more instances of {kernel name, kernel executions count, time})
- Provide **reasonable representation of execution behavior** while having **little instrumentation overhead** (can dial in whatever desired)
  - **Resolving application features at ~10 ms to a few seconds resolution**
- **System data collection at 1 sec** intervals



Recorded information per message:

```
rank,timestamp,job-id,kokkos-perf-data:time,kokkos-perf-data:type,kokkos-perf-data:name,kokkos-perf-data:count

0,100907.012310,8290750,0.000003,0,"Kokkos::View::initialization [Kokkos::Random_XorShift64::state]",2
0,100907.012360,8290750,0.000008,0,"Kokkos::View::initialization [DualView::modified_flags]",5
0,100907.012400,8290750,0.000014,0,"Kokkos::View::initialization [SurfCollide:nsingle]",4
```

Large-scale low-latency analytics

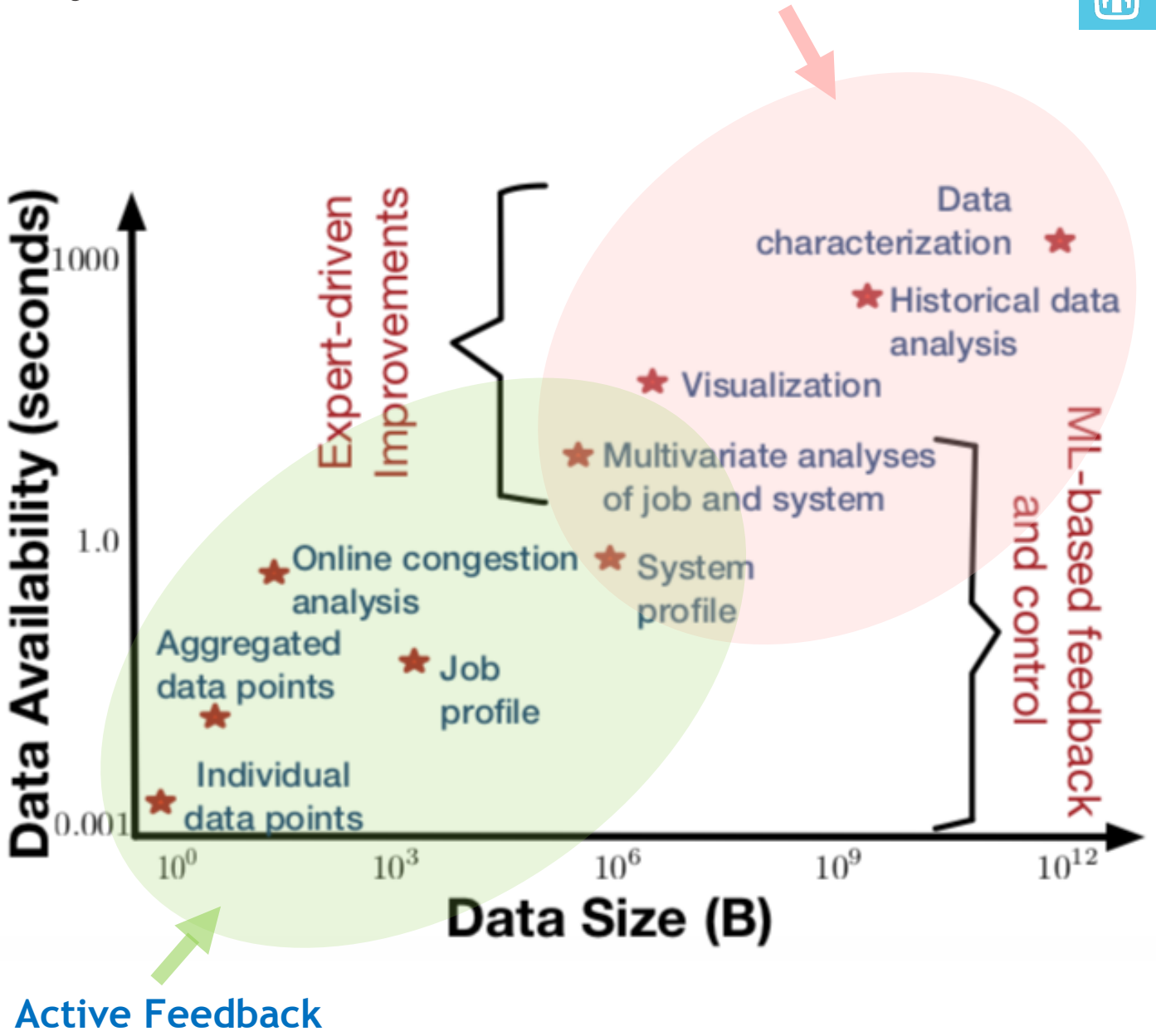
Challenge: Acquiring **tangible and realistic** run time **representations of resources' state** and of **application performance** that can be utilized to quantify performance effects for any given application and input deck.

Data Features

- High volume (10s of TB/day)
- High dimensionality (100s to 1000s of discrete metrics)
- Timeseries data from complex application science in variable workloads

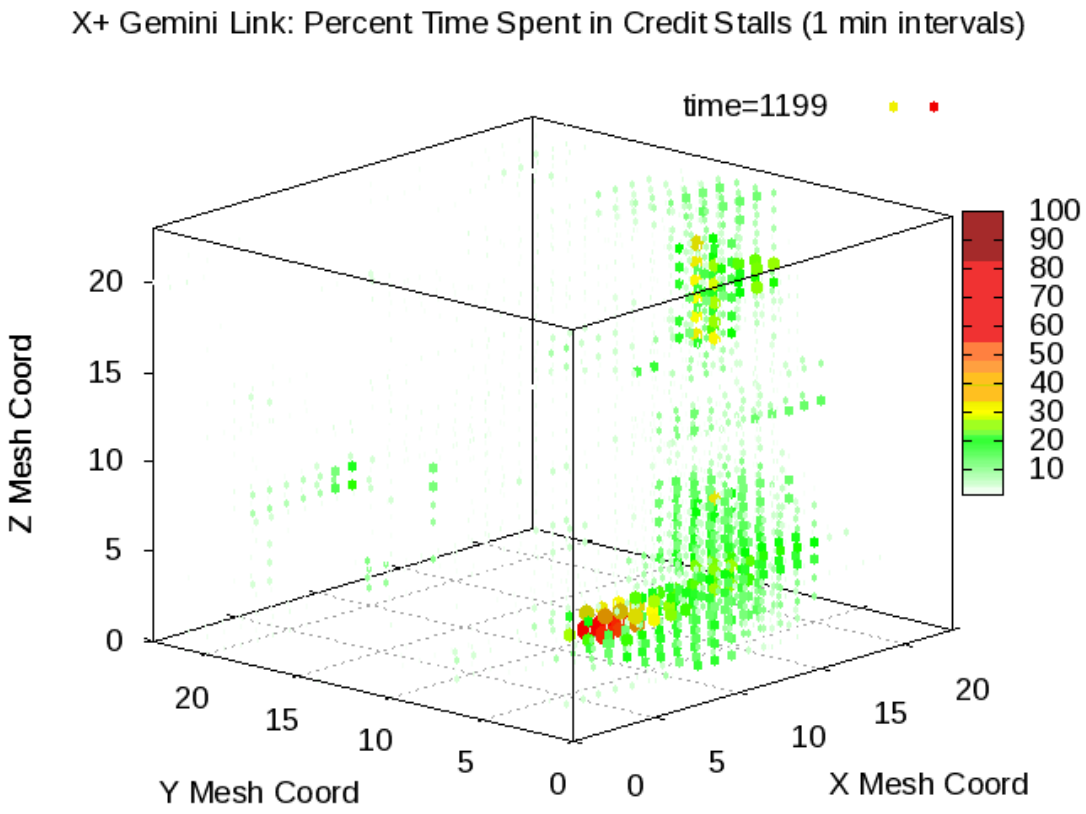
Machine Learning (ML) focus areas:

- Fundamental techniques for time-series, rare events
- Physics-constrained ML -> Architectural constraints
- Validated and explainable ML for automated response



Base Source: Holistic Measurement Driven System Assessment - *ECP Annual Meeting 2019*

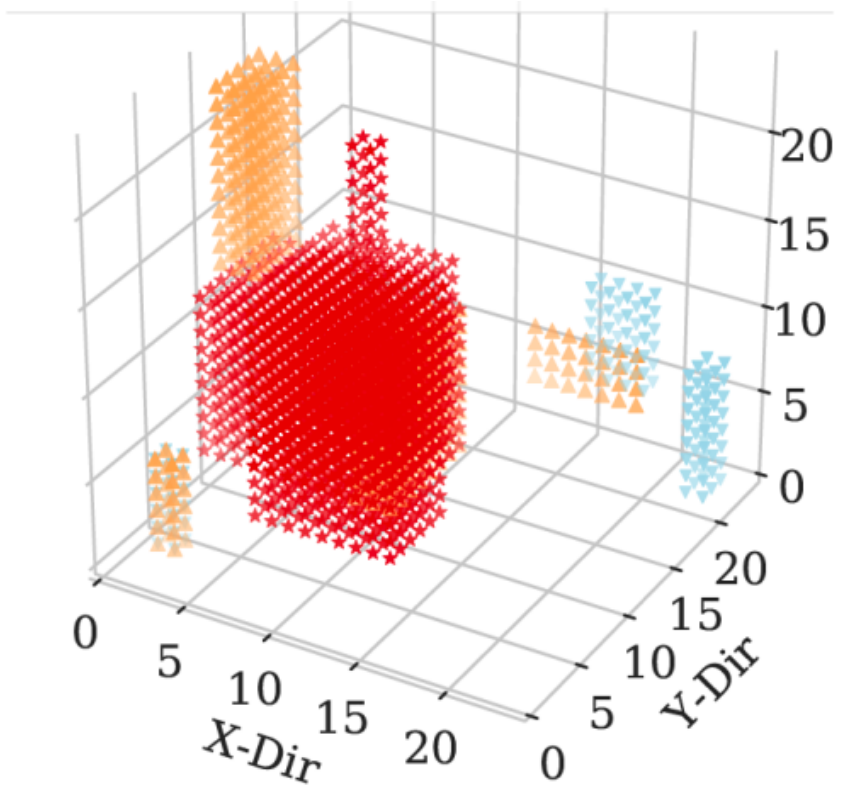
Example: Characterizing Network Congestion



Plays at 10 real minutes per second

Analysis from NCSA's Blue Waters, *Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications SC14*

Automated Characterization of "Congestion Clouds" In Cray Gemini Networks

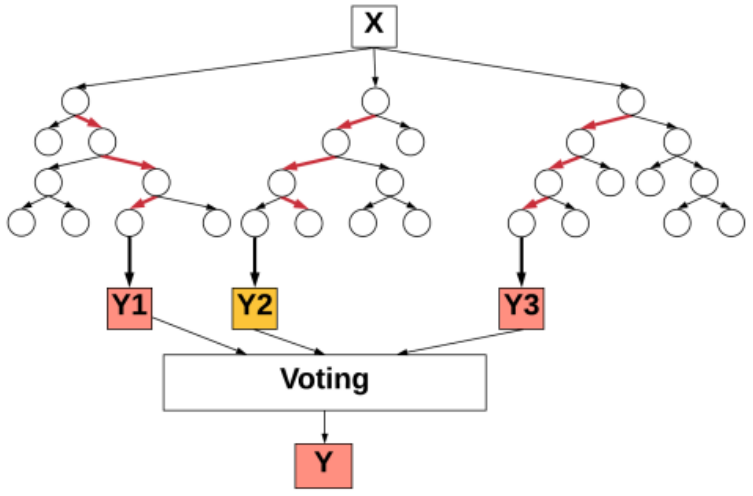


Long duration congestion clouds (direction independent) (NCSA Blue Waters)

From *Measuring Congestion in High Performance Data Center Interconnects NDSI20*

Determining performance-affecting data features using ML

- **Build models to characterize behavior** (resource utilization and performance) and adaptively allocate / control resources
- **Architectural and workload complexity** can make it difficult to determine the association of resource counters with performance effects
- Example: Use ensemble models to **identify features most indicative of application performance** and associated performance sensitivity



Ensemble Models

k-Fold

Cross Validation

Feature Type	Feature List (Cray XC)	Avg R2*
Application and Scheduler Specific	*app,hugepages, placement, balance, avg hops*	0.64
Network Specific	*nic2proc,proc2nic, hl2hl* PTS & SPF	0.86
All features	All the above	0.91

R2 is the coefficient of determination (higher is better)

From A Study of Network Congestion in Two Supercomputing High-Speed Interconnects HOTI26 2019

Conclusions

Systems are inefficient when components cannot know or respond to applications needs or resource conditions:

- Limits the effective use of **heterogeneous resources**
- Integrating Systems management into codesign would enable full interplay of management + applications + architecture
  Example: **Maximize total system throughput** in jobs/time and **ensure fairness** of access to network resources*
  - **Continuous assessment** of network performance impact drives **targeted throttling** of application injection based on injection rate and sensitivity to network latency
  - Enables low-bandwidth latency sensitive applications to maintain performance
  - ML-based characterization of applications to degree of latency or bandwidth sensitivity
  - Currently throttling at the MPI-layer, but **would like to directly control the fabric**

Need new CoDesign capabilities to support **on-platform** resource characterization analytics and response**:

- Pervasive low-latency **communication** channels among all components
- **Analytics sited closer to the data and responders**
- **Response hooks and capabilities**
- **Resource-aware** components with respect to capabilities and headroom
- **Negotiation** between resource components and application components in order to meet application needs within the capabilities of the resources

* From Delay Sensitivity-Driven Congestion Mitigation for HPC Systems ICS21
From **Including Operations, Analytics, & Communication In Next Generation CoDesign: It Just Makes Sense! ASCR Reimagining CoDesign Workshop 2021