# High-Performance Observability Systems for Dynamic HPC Environments

Working Group Outbrief

Scalable Tools Workshop 2025

## Motivation

Monitoring and analysis of performance information of HPC applications

- Need to gather information in a collective way (aggregate) across application
- Intention to process the information online for some purpose

Applications can be of any type

- SPMD modsim, task-graph, many-task, workflow, AI-coupled HPC, …
- Maybe some aspect of being dynamic in some way, but not necessary

HPC environment itself can present dynamics that impact HPC application execution

- Resources usage and management
- Inter-application dynamics
- …

## Need

Monitoring systems and integration with applications and HPC resources

Every layer of execution should be measurable

- Gather data from application, runtimes, schedulers, system, ...
- Each layer should be able to use the data to adapt execution
- Performance aware programming models and/or runtimes to adapt to monitoring data

## Existing approaches

- LDMS pushes data out multiple times during a job
  - (Characterizing HPC Codes and Associated Resource Consumption- https://sites.google.com/view/ldmscon2025/program?authuser=0)
  - Observability applications can request data from LDMS
- ADC - Application data collection
  - Combine data from various sources- Application, system metrics
  - Online and offline, enabling use with schedulers and AI
  - This comes out of Sandia and includes work from Ben Allan (baallan@sandia.gov), Ben Greigo (bfgrieg@sandia.gov), Ben Schwaller (bschwal@sandia.gov)
  - Pairs with Caliper, Kokkos, etc. for application data
- Power-monitoring with powercap control
- Analyze OpenMP regions and test different OpenMP settings to optimize execution
- Grafana: Different perspective but impressive
- Kubernetes: Provides generic monitoring of the services
- Weights&Biases for AI/ML workloads: https://wandb.ai/site

## Challenges

- Data from different sources are hard to integrate to make it easier to work with. Meta information is required. Correlation of data at runtime.
- Dedicated monitoring system next to new HPC systems
- In-band monitoring causes overhead
  - Could we get the vendors to open up co-processors for out-of-band monitoring?
  - Dedicated cores for management?
- Next-gen applications require built in monitoring and alternative execution paths to adapt based on the data

Notes

# High-Performance Observability Systems for Dynamic HPC Environments

Working Group Outbrief

Scalable Tools Workshop 2025

## Motivation

Monitoring and analysis of performance information of HPC applications

- Need to gather information in a collective way (aggregate) across application
- Intention to process the information online for some purpose

Applications can be of any type

- SPMD modsim, task-graph, many-task, workflow, AI-coupled HPC, …
- Maybe some aspect of being dynamic in some way, but not necessary

HPC environment itself can present dynamics that impact HPC application execution

- Resources usage and management
- Inter-application dynamics
- …

## Need

Monitoring systems and integration with applications and HPC resources

What are the types of problems and services are need

- Application-level
- System-level

What needs re-invention and what needs implementation

Is there anything that can be leveraged from the cloud observability technology?

# notes

- Focus on online processing or also offline?
  - Enable access to online measurements and (partly) the system
  - Process online and maybe move it somewhere else
  - Issue with batch systems because you have to wait until the job starts
  - Instrumentation is always on but why cannot we plug it in on the fly
    - Some are quite regular, so you get the profile fast and can turn instrumentation off
    - Or add more instrumentation based on the first profile
- Two ideas:
  - Observe or stear application from the frontend
  - Give information out one-way on-the-fly
- LDMS pushes data out multiple times during a job
  - (Characterizing HPC Codes and Associated Resource Consumption- https://sites.google.com/view/ldmscon2025/program?authuser=0)
  - Observability applications can request data from LDMS
- Interest in adaptive application that look at global state of execution and behave based on the findings.
- In-Situ visualization already a thing but different (and commonly cannot be stored)
- Power-monitoring (steer powercapping to meet requirements)
  - Monitor power per node every second
  - Update power distribution among workload
- Analyze OpenMP regions and test different OpenMP settings to optimize execution

## notes

- Every layer of execution should be measurable?
  - Gather data from application, runtimes, schedulers, ….
  - Each layer can use the data to adapt execution
- For next-gen application built in monitoring is required
- On mainframes you needed a smaller mainframe for instrumentation next to the big system. Dedicated monitoring hardware in new HPC systems next to it.
  - Could we get the vendors to open up co-processors for out-of-band monitoring?
- Most HPC applications are not using the resources well
  - Underutilization of CPU and GPU resources
  - Co-location of applications on the same resources
  - With constant monitoring, they could run side-by-side better without disturbing interaction
  - Cloud observability try to understand the effects of running applications close to each other and what it does to the system

## notes

- Train ML based on application data for a surrogate model
  - Compare correctness during further executions
  - Learn data movement to better tune online
- RADICAL Pilot system
- Cloud: Monitoring services
  - Orchestrator controls services
  - This orchestrator needs service/application data for resource assignment
- Workflows are a good example for observability
- ML model training:
  - Data staging/splitting
  - GPU amount/location
  - Accuracy
  - Who implemented the PyTorch profiler? They use somehow the same data but less granularity, quite simplistic (execution time but no HWC data) and not good at multi-node
  - But mostly post-mortem analysis
  - Use the context manager of PyTorch to hook in and to get information about the current code location

- Performance aware programming model or runtime to adapt to monitoring data
- Async runtime systems split work into small units that can be scheduled separately and efficiently
    - Keeping counts like queue fill state, communication tasks, … could control the creation of new work units
- Data from different sources are hard to integrate to make it easier to work with. But do we need all data at any time or are the interesting ones enough? Meta information is required. Correlation of data at runtime.
- Function-as-a-service: Scheduling the execution to resources
- Grafana: Different perspective but impressive
- Kubernetes: Provides generic monitoring
- Missing connection between application monitoring and system monitoring tools

- ADC - Application data collection
  - Combine data from various sources- Application, system metrics
  - Online and offline, enabling use with schedulers and AI
  - This comes out of Sandia and includes work from Ben Allan (baallan@sandia.gov), Ben Greigo (bfgrieg@sandia.gov), Ben Schwaller (bschwal@sandia.gov)
  - Pairs with Caliper, Kokkos, etc. for application data
- Streaming time-series databases like InfluxDB, TimescaleDB, VictoriaMetrics
- Weights & Biases by CoreWeave for ML training: https://wandb.ai/site