SIMON: *A Simple Monitoring Framework for Heterogeneous Application Observability*

Michael Dushkoff, Allen D. Malony, Grace McLewee, Kevin Huck
University of Oregon

Scalable Tools Workshop 2025
Granlibakken Resort, Lake Tahoe

HPC Heterogeneity

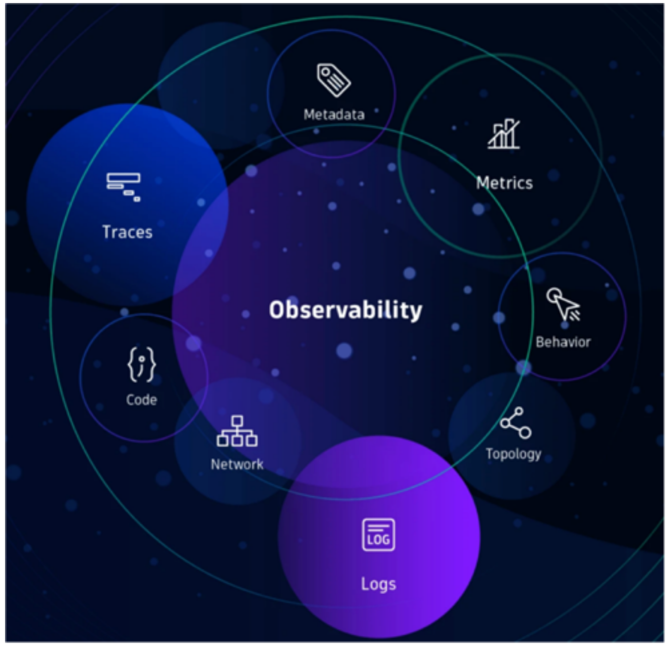
- Rapidly changing HPC landscape
- Heterogeneous architectures and system hardware
  - Processor types / diversity: CPU, GPU, NPU, … from multiple companies
  - Memory: cache hierarchy, NVRAM, CXL, unified memory, …
  - Network: interconnection, smarNICs, DPUs, …
- Drives software technology for dealing with heterogeneity
  - Higher-level programming languages and frameworks
  - Support for cross-platform performance portability
- Heterogeneity in HPC applications (parallelism)
  - Traditional “faster is better” orthodoxy of (*hierarchical (vertical) parallelism*)
  - New paradigms of concurrent heterogeneous tasks (*horizontal parallelism*)
  - Broader computing continuum and variety in types of users/applications
  - Different execution models (SPMD, workflow, streaming, AI-coupled, …)

Observability

- In general, *observability* of a system is the ability to make measurements of its operation in order to understand how and how well the system performs when used by application
  - Ability to understand the internal state of a system by analyzing its external outputs to gain insights into its health and performance.
- Concept of observability has gained considerable attention in scaled-out distributed enterprise system environments (cloud)
  - Challenges in developing/deploying applications in diverse ecosystems
  - Complexity of workflows of many heterogeneous tasks
  - Various runtime stacks in use and scaling out performance challenges

Cloud-native Observability

- Visibility into how applications behave in cloud-based systems
  - Identify execution anomalies and evaluate performance issues
- Observation telemetry data produced from measurements
  - *Event logs*: record details of an event
  - *Metrics*: data to quantify performance and health
  - *Traces*: track end-to-end services
  - Collectively use the term *monitoring*
  - Emphasize online telemetry accessibility
- Observability is more about making sense of it
  - Reasoning and improving in real time
- Commercial importance of cloud-based applications
  - Motivated development of sophisticated observability frameworks



HPC Performance Observability

- Older history of observability in HPC (see my Ph.D. thesis 🙂)
- How to use HPC resources effectively to achieve optimal performance
- Observability focus on parallel performance measurement and analysis
  - Identify performance inefficiencies and limitations
  - Detailed understanding of processing, memory, and communications hardware
  - Inform application tuning decisions
  - Observability methods should minimally impact the system
- HPC research community has developed robust performance tools
  - Address HPC application observability challenges
  - Different in design and operation from cloud-based environments
- Heterogeneity in HPC raises issues about tools for observability
- Reimagine observability in the context of HPC application monitoring

My Obsession with Application-level Monitoring

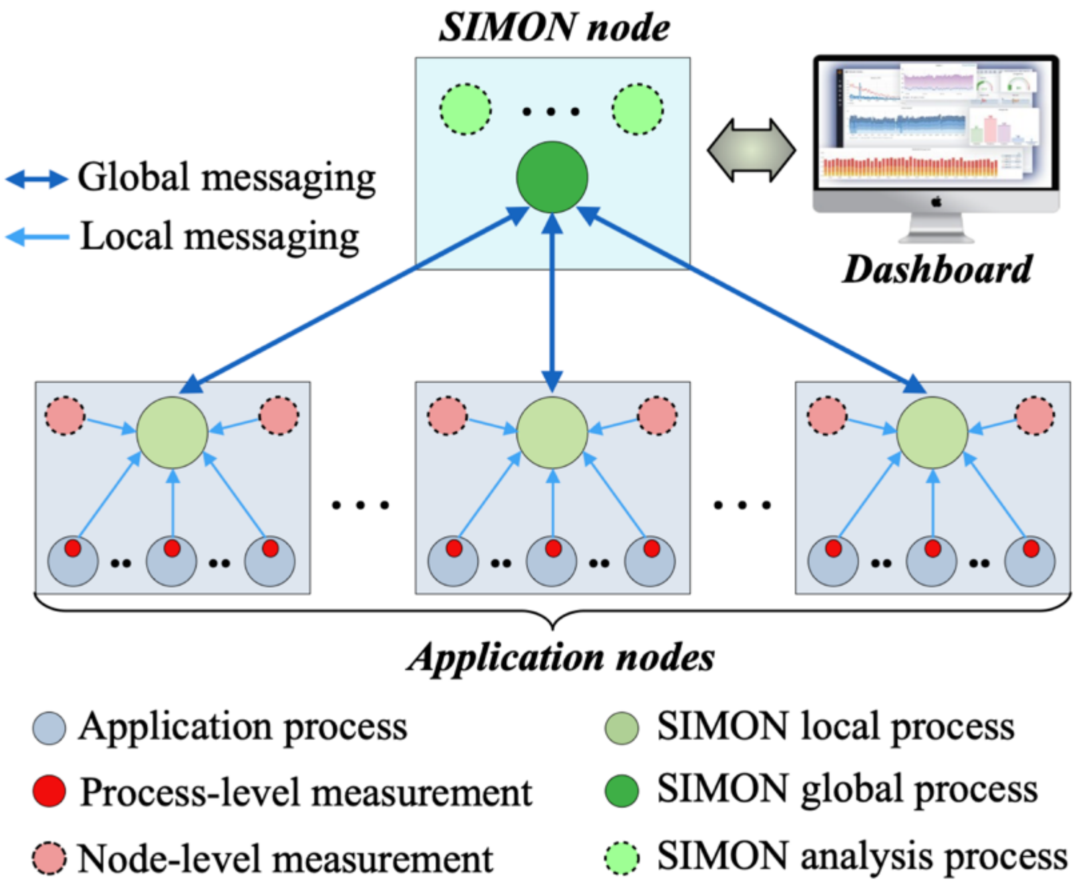
- Interested HPC application-level monitoring for a long time:
  - Hypermon (1990)
  - VNG (2003)
  - TAUg (2006)
  - TAUoverSupermon (2007)
  - TAU snapshots (2007)
  - TAUoverMRNet (2008)
  - Monitoring sweetspots (2008)
  - TAUmon (2010)
  - POWMon (2015)
  - APEX (2015)
  - WOWMON (2016)
  - SOS (2016)
  - Artemis (2021)
  - Symbiomon (2021)
  - ZeroSum (2023)
  - SOMA (2024)
- Motivated by belief of value in online access to performance data

SIMON Objectives

- Support heterogeneous HPC application observability
  - Simple monitoring architecture and framework
  - Based on an asynchronous time-series observational model
  - Enable in situ analytics and visualization
- Offer (simple) functionality and development
  - Broadly available and easy to use
  - Straightforward to integrate with HPC applications and tools
  - Can operate out-of-the-box in standard HPC environments
  - Programmable, extensible, and configurable
  - Addresses modern observability requirements
  - Utilize modern observability technology
- Easily construct new monitoring solutions (however simple)

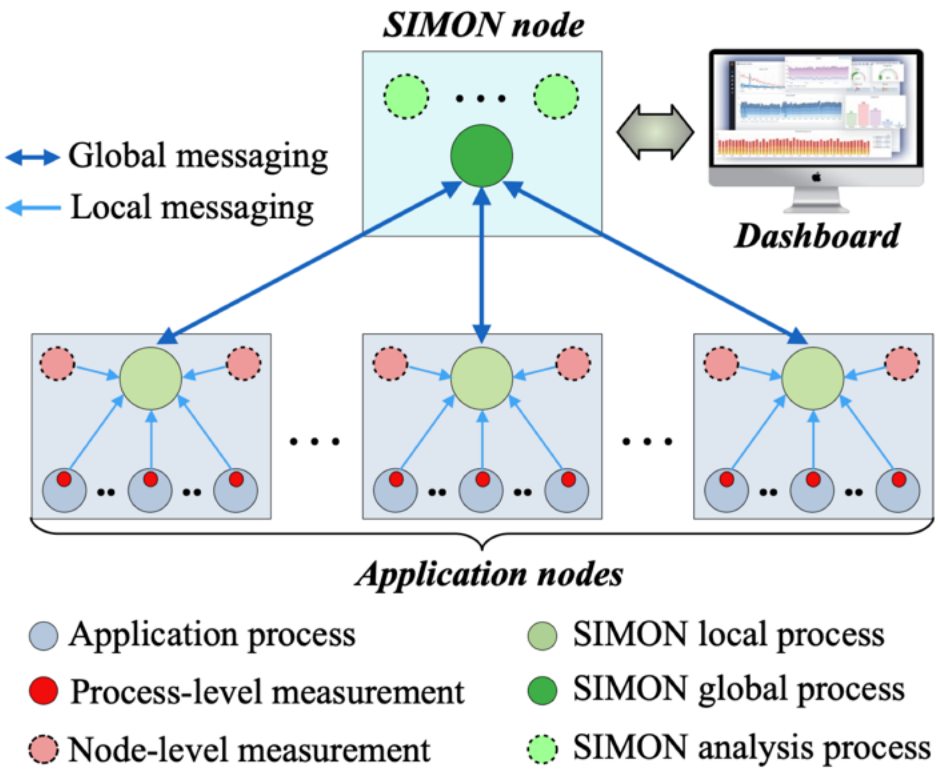
SIMON Design Architecture for Application Monitoring

- Minimalist approach
  - Separation of concerns
  - Focus on telemetry data transport
- Components
  - ***Providers*** (measurement, telemetry)
    - Process-level (embedded in process)
    - Application node-level (daemon process)
  - ***Collectors*** (local)
    - Application node-level (SIMON process)
  - ***Aggregator*** (global)
    - SIMON resource (SIMON process)
    - Analytics helpers
  - ***Messaging*** (local, global)
  - Dashboard
- SIMON launches / terminates with the application



SIMON Prototype

- SIMON started as a graduate course project
  - *Advanced Parallel Computing*, Winter 2025
  - Two 1st-year graduate students
  - 2-3 months worth of effort thus far
- Development decisions
  - Python for all SIMON processes
  - ZeroMQ for messaging
- Leverage existing measurement providers
  - HPC performance: TAU, APEX, ZeroSum, …
  - System: /proc, Linux `top` program, ..
  - Other profilers: PyTorch Profiler, …
- Work with Slurm to launch
- Python-based visualization
  - Seaborn, Matplotlib



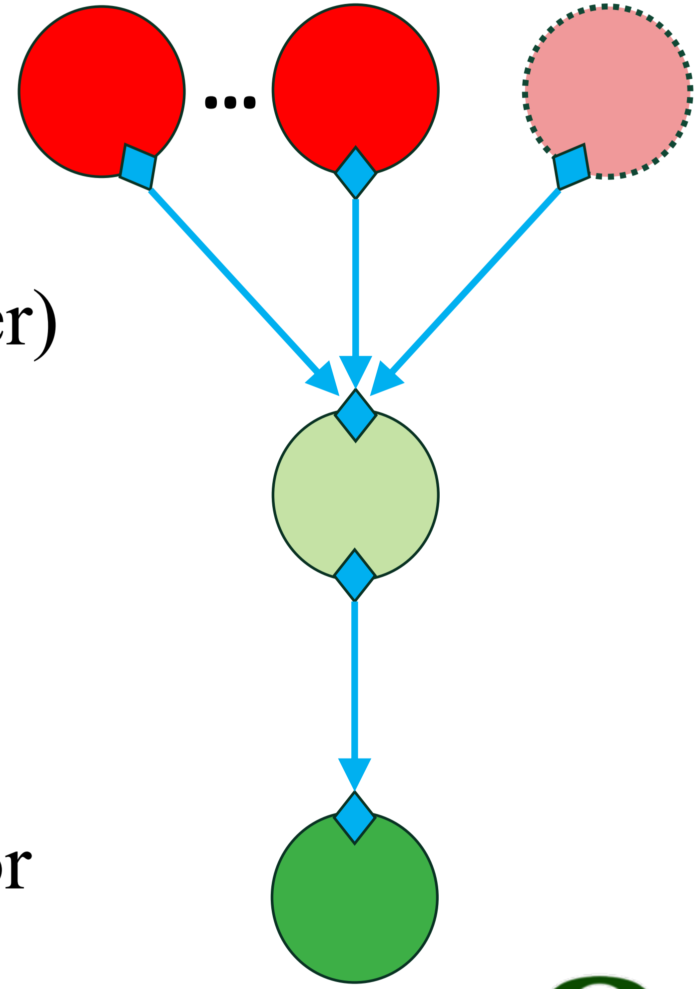
Why Python and ZeroMQ?

- Python
  - Ubiquitous and portable
  - Highly functional
  - Many libraries and tools
  - Used in other observability tools and frameworks
  - Used in other application environments of interest
  - Easy to use for prototyping
- Why not?

- ZeroMQ
  - Open source messaging library
  - Robust and universally available
  - Asynchronous message processing
  - Mulitple communication patterns
  - Fast enough for clustered systems
  - Several language APIs and OS
  - Inter-process communication transport (`zmq_ipc`, `zmq_tcp`)
  - NOT MPI

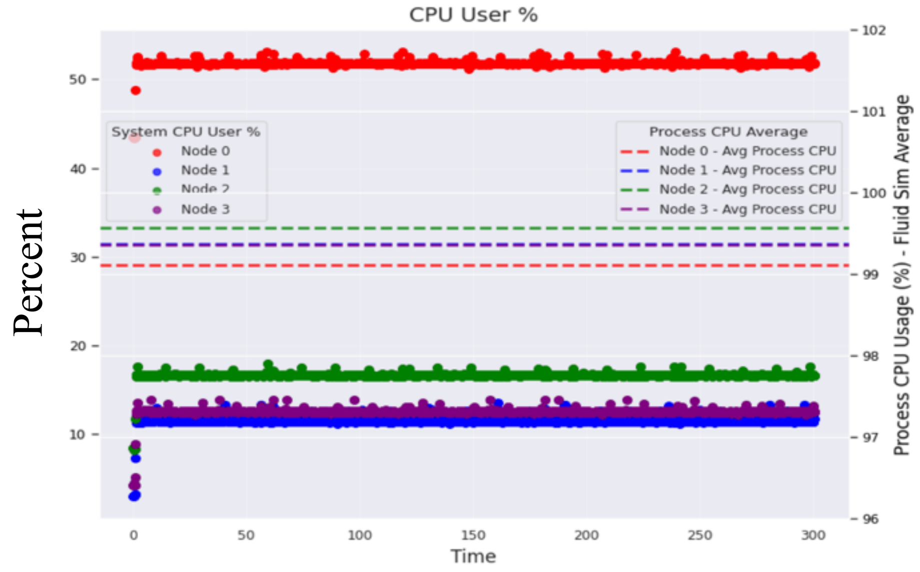
Launch, Configuration, and Measurement Provider

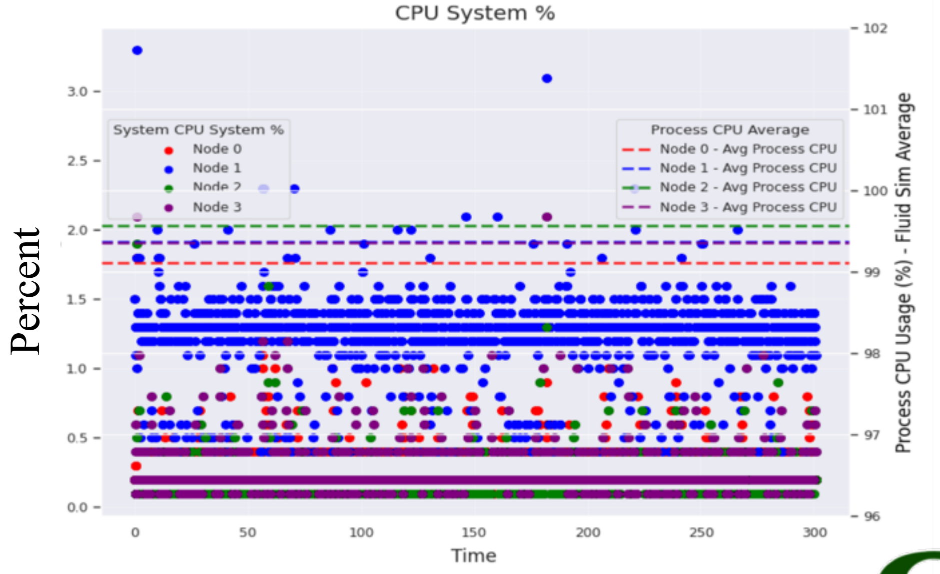
- SIMON collector processes and aggregator starts first
  - Establish OMQ connections
- Application node measurement daemon starts
  - Discovers SIMON collector process
  - Establishes OMQ connections
- Application processes starts (w/ measurement provider)
  - Special magic to set up SIMON at init
  - Discovers SIMON collector process
  - Establishes OMQ connections
- Measurement data converted to structured text
  - Structure determined by type of data
  - Example: CSV, key-value, formatted trace records, …
- Uses ZeroMQ API to send asynchronously to collector
- Each message is timestamped with provider ID



Test Case: Distributed "Top" Monitor

- Multi-node heterogeneous application
  - Complex fluid field simulation on image
  - 16 instances (processes) on 4 nodes
  - Ran for 300 seconds
- Experiment on Talapas cluster
  - Each node has 2x 32-core AMD EPYC
  - Script ran with other programs active
- Local SIMON daemon measurement
  - Ran `top -b -n 1` every second
- Deliver telemetry to SIMON aggregator
  - Allocated its own node
  - >1 MB / second total "top" telemetry
- Realtime analytics and visualization





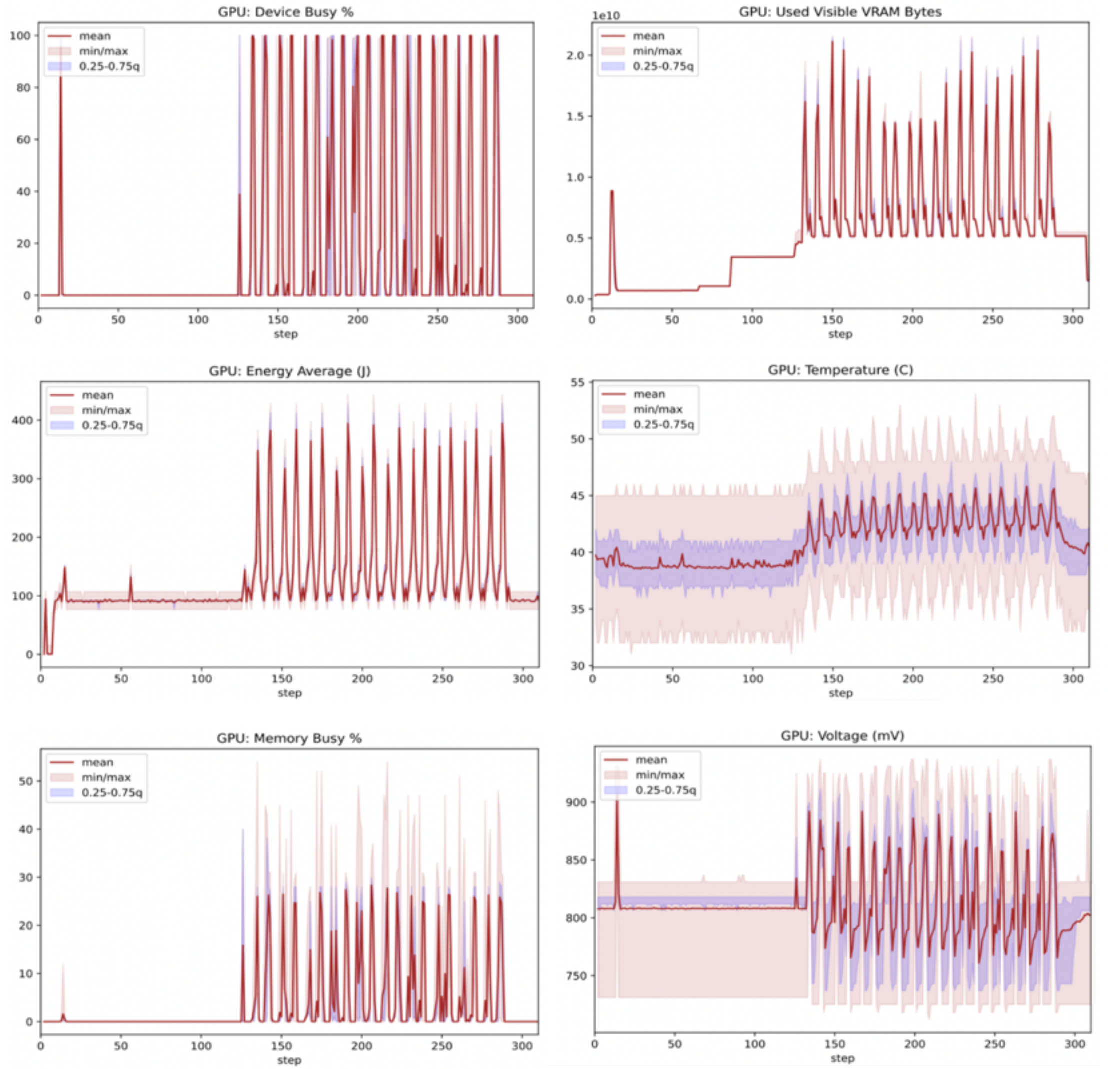
ZeroSum

*K. Huck, A. Malony, "ZeroSum: User-space Monitoring of Resource Utilization and Contention on Heterogeneous HPC Systems," HUST, SC workshop, 23. https://dl.acm.org/doi/10.1145/3624062.3624145

- Certain application observation requires application and system measurement
  - Can not diagnose performance problems otherwise
  - Need for periodic sampling
- Need to associate application performance view with system behavior
  - Resource utilization monitors for applications
  - Check for misconfiguration and efficient utilization
  - Confirmation of expected hardware / operating system behavior
  - Aid to identify failure scenarios
- Monitor coverage (per process, all nodes, sampling)
  - Application threads (LWP)
  - CPU hardware (HWT)
  - Memory
  - GPU hardware for all processes, all nodes in the allocation
- Utilizes an extra thread for monitoring

Test Case: XGC Monitor

- SIMON integration in heterogeneous HPC application
  - XGC gyrokinetic PIC code
  - C++ / Fortran 90, MPI, GPU, Kokkos, Cabana
- Measurement providers for SIMON telemetry data
  - ZeroSum modified to write out sample windows
  - Each process samples every second
  - Writes to SIMON every 10 seconds
- Experiment on Frontier
  - 64 MPI ranks distributed on 8 nodes
  - Each with 64-core AMD EPYC and MI250X GPU
  - Each rank assigned graphic compute die (GCD)
- Deliver telemetry to SIMON aggregator
  - Node collectors gather telemetry for 8 ranks
  - Deliver to aggregator runnin on its own node
- Realtime analytics and visualization



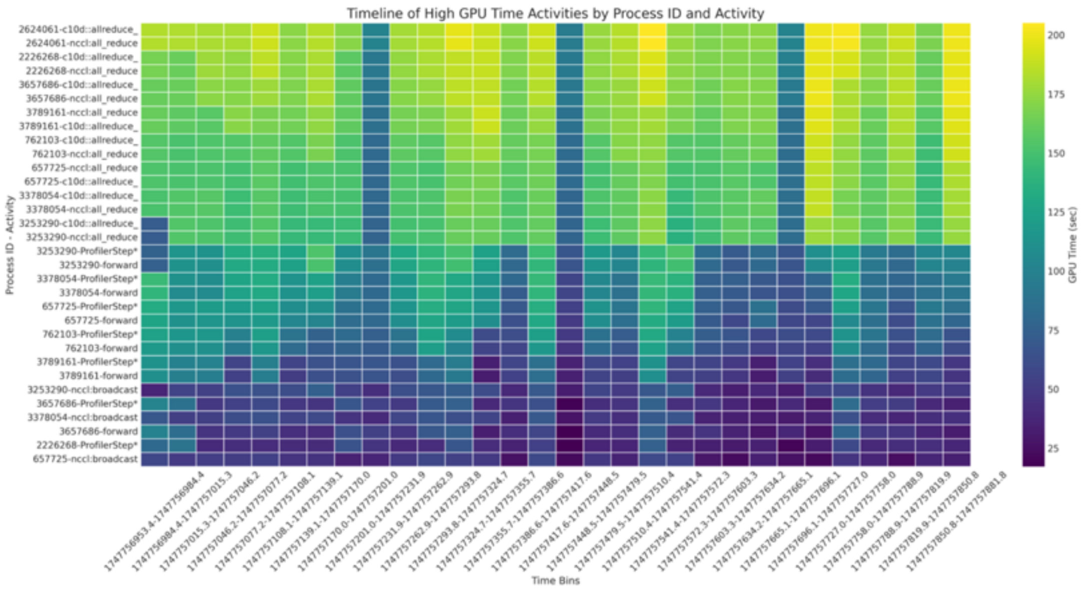
PyTorch Profiler

https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html

- Built into PyTorch
  - Analyze deep learning models during training and inference
  - Identify performance bottlenecks, optimize operations, improve memory efficiency, and accelerate training and inference of PyTorch models
- Detailed insights into various aspects of model execution
  - Time spent on operations (different layers, CPU and GPU)
  - Memory usage (allocation/deallocation for tensors, CPU and GPU)
  - CUDA kernel activity
  - Tensor shapes to operations
  - Python tack traces
- Context manager to define code region to be profiled

Test Case: Distributed PyTorch Training (ResNet-50)

- Distributed PyTorch machine learning application
  - Training using ResNet-50 for image classification
- Measurement providers for telemetry data
  - Embedded in the application code (per process)
  - Daemon capturing node-level data
- PyTorch Profiler to capture model-level profiling information
  - Worked with profiling context manager
  - Capture profiling data for specific code regions
  - Wrapped epoch training code to see each layer and PyTorch routines used
  - Telemetry output for each epoch
- Daemon to observe GPU utilization
  - NVIDIA SMI monitor
  - GPU utilization, power consumption, and GPU memory usage
  - Telemetry output per second
- Experiment on 8 Talapas GPU nodes
  - Each with 24-core AMD EPYC and NVIDIA A100 GPU
- Deliver telemetry to SIMON aggregator
  - Allocated its own node
- Realtime analytics and visualization



Preliminary Takeaways and Possible Issues

- Took some trial-and-error to get things configured properly
- Relatively straightforward to hook up providers
- Mostly small-scale experiments
  - Real-time telemetry transport, analysis, and visualization
  - No noticeable performance effects
- Did not run experiments at moderate or large scale
- Obvious potential problem with overloading single aggregator

Related Work (and many more)

- Systems monitoring
  - Ganglia (https://github.com/ganglia)
  - Nagios (https://www.nagios.org)
- HPC systems monitoring
  - LDMS (https://github.com/ovis-hpc/ldms)
  - ClusterCockpit (https://clustercockpit.org)
  - Pika (https://gitlab.hrz.tu-chemnitz.de/pika)
  - Omnistat (https://github.com/AMDResearch/omnistat)
- Enterprise-ready observability technologies
  - Grafana (https://grafana.com)
  - Prometheus (https://prometheus.io)
  - VictoriaMetrics (https://victoriametrics.com)
  - Dynatrace (https://www.dynatrace.com)
  - OpenTelemetry (https://opentelemetry.io)
  - InfluxDB (https://www.influxdata.com)

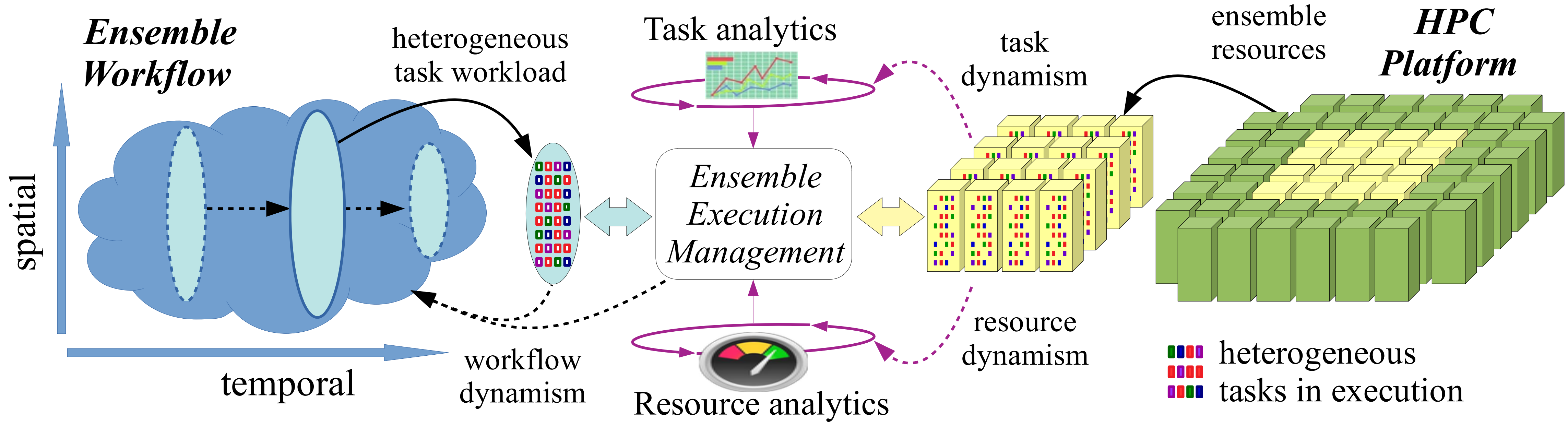
SIMON Next Steps

- Work on SIMON prototype was accepted for Heteropar 2025
- Interested in applying approach more use cases and scenarios
- Involve others in building out SIMON framework and tools
- Measurement providers
  - Develop library of telemetry data sources
  - Add other HPC performance data: Caliper, Score-P, HPCToolkit, …
- In situ telemetry processing
  - Variety of actions at collectors: filtering, merging, sampling control, …
  - Routing of telemetry types to specialized aggregators
  - Develop library of telemetry processing modules
- Aggregation
  - Address scaling concerns through replication, specialization, …
  - Develop analytics and visualization modules

HPC Workflow Performance Observability

- Heterogeneous tasks dynamic heterogeneous platforms
  - Adaptive and dynamic response needed to break free of static execution
  - Heterogeneity and dynamism over many spatio-temporal scales
- Middleware architecture for decision-making and information-access
  - For task execution that provides performance-assurance
  - Optimal and robust decision-making depends on Quality of Information (QoI)
  - QoI only as good as performance monitoring and introspection
- Dynamism, heterogeneity, and state uncertainty in workflows and resources make performance monitoring and introspection challenging
  - Monitoring and introspection → modeling, prediction, and actuation

Canonical Ensemble Execution Environment



- Initial attempt with RADICAL Pilot and SOMA
- SOMA stack more complicated than SIMON
- Re-attempt with RP and SIMON

D. Yokelson, et al., "SOMA: Observability, Monitoring, and in situ Analytics for Exascale Applications," CCPE, Vol. 36, Issue: 19, August 2024.

D. Yokelson, et al., "Enabling Performance Observability for Heterogeneous HPC Workflows with SOMA,", ICPP, August 2024.

Summary

- HPC applications are growing more diverse and heterogeneous
  - More asynchronous and dynamic
  - Observation needs will be different
- Post-mortem analysis is incomplete and limited
  - Ineffective in dynamic environments
  - Timely data needed to make dynamic decisions
- Interesting observability technology in cloud world
- Concern for HPC observability impact should not deter how a monitoring system is developed and used
  - Plenty of under-utilized resources
- Monitoring technology that is portable, robust, and easy-to-integrate will be more productively applied in many cases