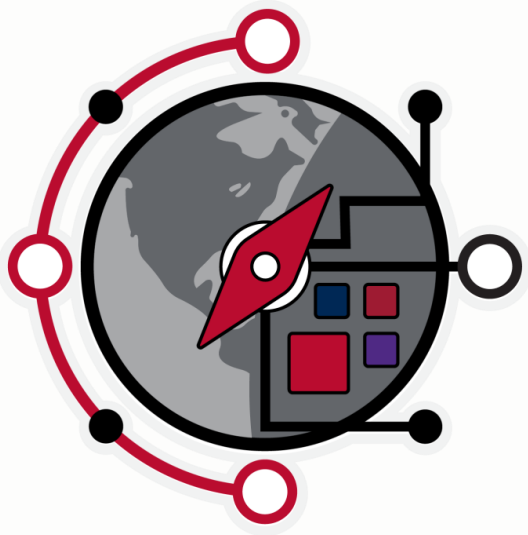




Center for
Optimized Modern
Parallel Adaptive
System Software

Project Foundations

Derek Schafer (University of New Mexico)

Prof. Puri Bangalore (University of Alabama)

Section 01

Training Materials

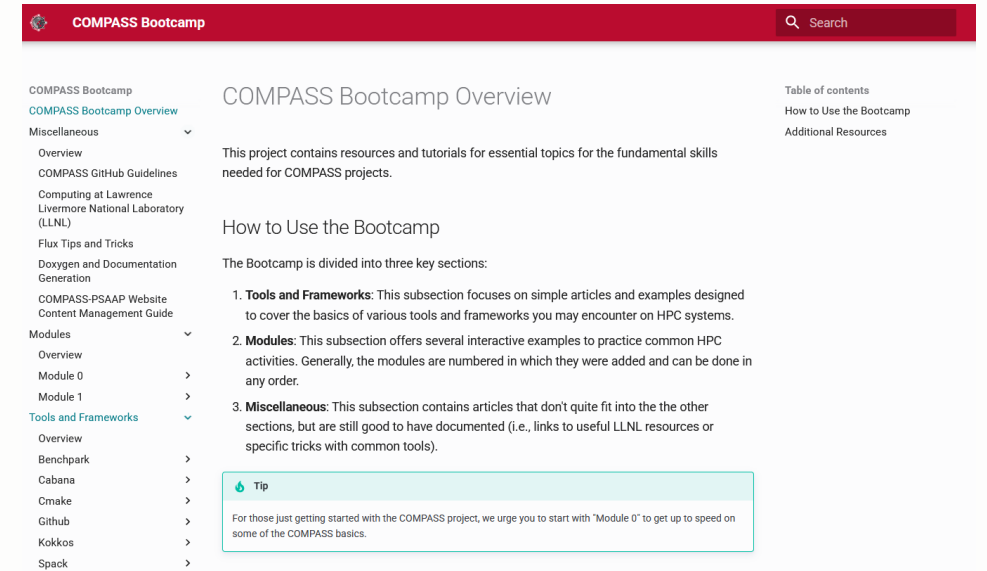


COMPASS Project Roadmap

	Year 1 (FY26)				Year 2 (FY27)				Year 3 (FY28)				Year 4 (FY29)				Year 5 (FY30)						
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4			
AI Performance / Modeling	Analyze AI model sensitivity				Model data collection & augmentation				Decision under uncertainty				Model testing & validation										
					Research model uncertainty quantification				Research model transfer learning				Enhanced model for optimization integration										
Cross-Layer / Optimization	Assess sparse solver optimizations		Assess Beatnik optimization				Layer-by-layer optimization		Search-based optimization				Integrate model uncertainty				Fully integrate models						
			Scheduler optimization sensitivity		Assess NNSA app optimization opportunities																		
Solver Design	Beatnik fast multipole support		Z-Model adaptive meshing		Compare Krylov solver variants				Parallelize topological transitions				Specialize models for physics problems				Integrate models into production codes						
	Develop compressible interface model				Model topological transitions				Multi-scale reactive interface methods				Parallelize multi-scale reactive solver										
					Parallelize compressible interface model				Combine Krylov enhancements into unified solver														
System Software	MPI/Kokkos stream triggered communication				Additional MPI/Kokkos extensions								Time-based communication primitive optimization										
	Create initial RAPIDS primitives				Additional RAPIDS extensions								Cross-layer control of system software APIs										
			Kokkos memory layout optimizations		Develop System Tuning APIs																		
Integration & / Assessment	Design app/system workflow				Assess Z-Model/FIESTA				Assess Krylov Variants														
	Design/deploy system infrastructure				Assess NNSA application optimization								Assess integrated hydro codes										
Education & / Workforce	Create bootcamp materials				Develop tradeoff analysis materials								Create AI model course materials				Offer courses with integrated materials						
			Deploy bootcamp modules		Create conference tutorial materials				Offer HOTI/SIAM/SC tutorials														

Building Bootcamp Documentation

- Collection of knowledge that was typically passed verbally or hidden deep inside other documentation
 - Intended to be complementary to LLNL's HPC documentation, tutorials
 - Place to store materials for hackathons, onboarding, applications
- Three different focus areas:
 - Applications / Framework (details on building, running, or using specific applications or libraries)
 - One-off articles (miscellaneous details, e.g., flux tips, COMPASS documentation)
 - Modules (more interactive examples)
- Now a public website: <https://compass.unm.edu/bootcamp/>



The screenshot shows the COMPASS Bootcamp website. The header is red with the COMPASS Bootcamp logo and a search bar. The left sidebar contains a navigation menu with categories like COMPASS Bootcamp Overview, Miscellaneous, Modules, and Tools and Frameworks. The main content area is titled 'COMPASS Bootcamp Overview' and includes a table of contents, a description of the project, and a section on how to use the bootcamp. A tip box at the bottom of the main content area suggests starting with 'Module 0'.

COMPASS Bootcamp

COMPASS Bootcamp Overview

Miscellaneous

Overview

COMPASS GitHub Guidelines

Computing at Lawrence Livermore National Laboratory (LLNL)

Flux Tips and Tricks

Doxygen and Documentation Generation

COMPASS-PSAAP Website Content Management Guide

Modules

Overview

Module 0

Module 1

Tools and Frameworks

Overview

Benchmark

Cabana

Cmake

Github

Kokkos

Spack

COMPASS Bootcamp Overview

This project contains resources and tutorials for essential topics for the fundamental skills needed for COMPASS projects.

How to Use the Bootcamp

The Bootcamp is divided into three key sections:

1. **Tools and Frameworks:** This subsection focuses on simple articles and examples designed to cover the basics of various tools and frameworks you may encounter on HPC systems.
2. **Modules:** This subsection offers several interactive examples to practice common HPC activities. Generally, the modules are numbered in which they were added and can be done in any order.
3. **Miscellaneous:** This subsection contains articles that don't quite fit into the other sections, but are still good to have documented (i.e., links to useful LLNL resources or specific tricks with common tools).

Tip

For those just getting started with the COMPASS project, we urge you to start with "Module 0" to get up to speed on some of the COMPASS basics.

Table of contents

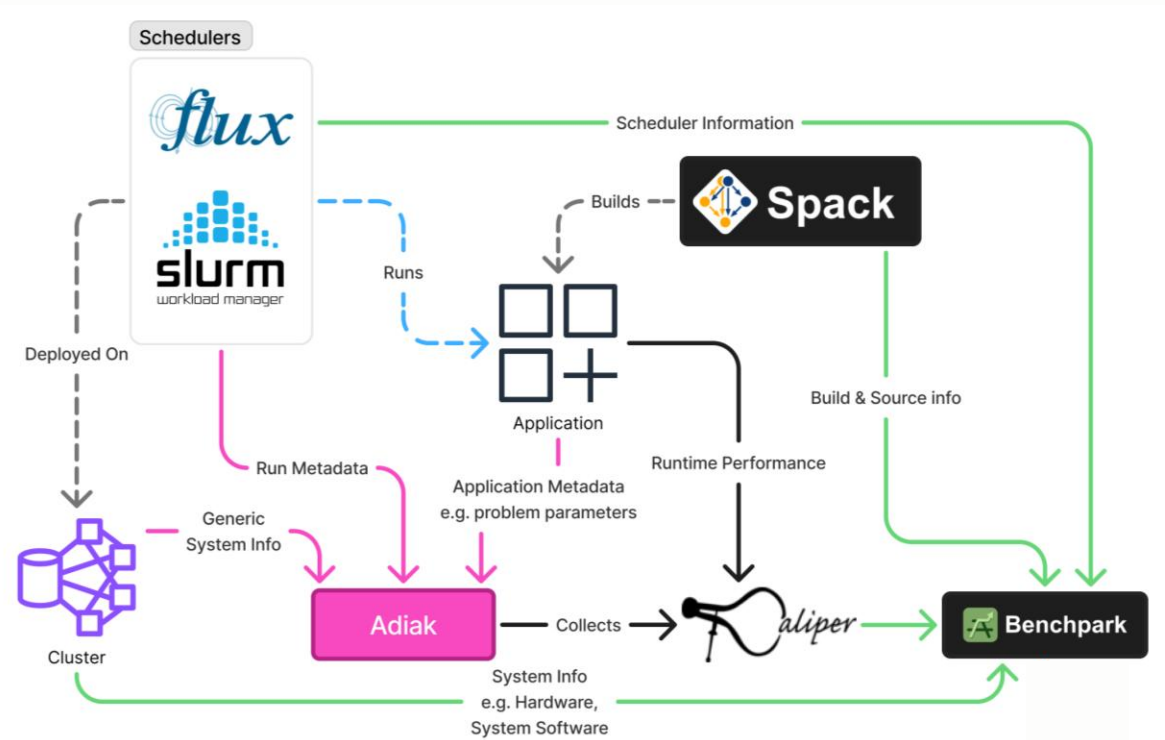
How to Use the Bootcamp

Additional Resources



March 2026 Hackathon: Building and Running AMG2023

- Very early, exploratory look into experiment workflow using an example application
 - Goal 1: Use AMG2023 as a prototype application for understanding what an application workflow may look like
 - Goal 2: Figure out what data to capture from an experiment run (application's own output, Caliper, system metrics, etc.)
- Introduced students to many potential components that our overall infrastructure may need to deal with
 - Flux/Slurm, Spack, Caliper, Benchpark, etc.
- Good hackathon that helped everyone start to visualize how aspects of the various applications and frameworks interact with one another



August 2026 Hackathon: Learning the ins-and-outs of Tuo's compilers

- Goal: Uncover as many of the headaches, pitfalls, and flags to use when compiling programs on LLNL's Tuolumne as we could
 - GPU programs, MPI + GPU programs, CMake + MPI + GPU programs
 - Provided sample codes for each step
- Each student was asked to pick different a compiler (or wrapper)
 - CC, mpic++, amdclang++, hipcc, mpiamdclang++, clang++, g++
- Briefly touched on tricks for allocating GPUs with flux (and how to debug what was given)

```
# Output from "project(BootcampBuilds LANGUAGES HIP CXX)"
-- The HIP compiler identification is Clang 19.0.0
-- The CXX compiler identification is Clang 19.0.0
-- Check for working HIP compiler: /opt/rocm-6.4.3/lib/llvm/bin/clang++ - skip
-- Check for working CXX compiler: /opt/rocm-6.4.3/lib/llvm/bin/clang++ - skip
-- Found MPI_CXX: /opt/cray/pe/mpich/9.0.1/ofc/cray/20.0/lib/libmpi_cray.so (f
-- Found MPI: TRUE (found version "4.0")
# Output from "project(BootcampBuilds LANGUAGES CXX HIP)"
-- The CXX compiler identification is GNU 8.5.0
-- The HIP compiler identification is Clang 19.0.0
-- Check for working CXX compiler: /usr/lib64/ccache/c++ - skipped
-- Check for working HIP compiler: /opt/rocm-6.4.3/lib/llvm/bin/clang++ - skip
-- Found MPI_CXX: /opt/cray/pe/mpich/9.0.1/ofc/cray/20.0/lib/libmpi_cray.so (f
-- Found MPI: TRUE (found version "4.0")
```

Handful of Classes Taught So Far

UA

CS 481/581 High-Performance Computing (Spring 2026)
CS 691 Advanced Parallel Algorithms (Fall 2026)

UNM

CS 429/529 Introduction to Machine Learning (Fall 2025)
CS 442/542 Introduction to Parallel Processing (Fall 2026)

TN Tech

CSC 4760/5760 Parallel Programming (Spring 2026)

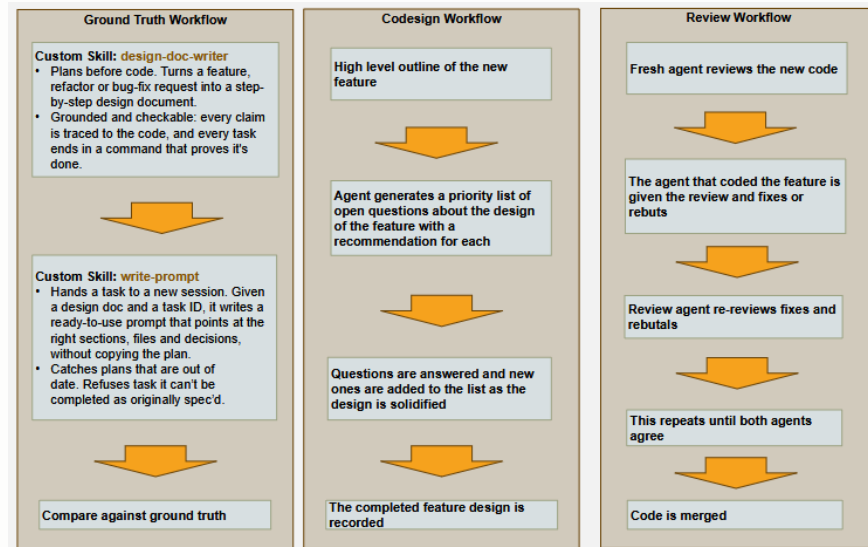
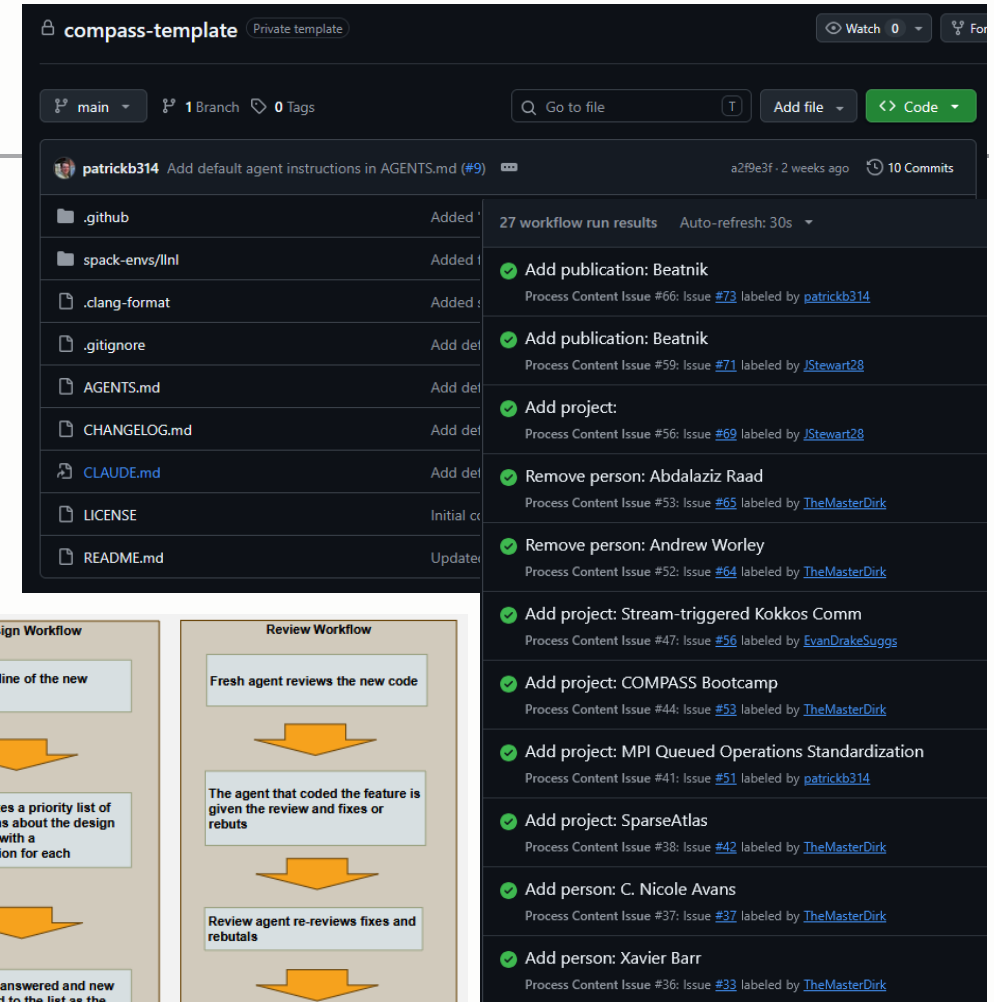
Tutorials and Other Trainings

- Several Tutorials:
 - Understanding Communication Performance on Emerging Heterogeneous Architectures (HotI 2026)
 - Understanding Communication Performance in Large-Scale Codes: A Practical Guide (SC 2026)
- Transitioned the Weekly Meetings from our prior CUP-ECS center into working groups
- Collaborate with CRT to gain insight into system capabilities and requirements:
 - Worked with CRT team for a DST (1024 nodes of Tuolumne last December)
 - Meet with LLNL (Jane Herriman, Ryan Quinlan) to discuss our thoughts on experimental workflow



Early Use of AI in Our Center

- Finding best practices for using AI:
 - How do we want to use AI for code prototyping and analysis?
 - See Jason and Thomas poster on our learnings for guiding AI models to do their best work
- General project management with GitHub
 - Tracking issues, proper feature branching, outlining of projects
 - Forms for updating information on the COMPASS website
 - Created a git template-repository for all new projects in COMPASS to start from
 - Formatting standards, basic GitHub files
 - Template repository with default AI agent instructions



Additional Foundational Material Development

- Understanding and documenting behaviors about the plethora of tuning knobs:
 - Many parameters at every level:
 - Application Level (solver choice, solver parameters)
 - MPI Level (various Cray-MPICH flags)
 - Lower Network layer (libfabric, GPU flags)
 - How does run-to-run variation affect times and parameter testing? – see Xavier's poster
- Statistical rigor in benchmark experiment design and decisions that use them
 - MPI benchmark statistical data collection problems are well documented but not well known
 - Using this data to make principled optimization decisions is even harder (See Nick B's poster)
- Developing documentation for guidance on statistical best practices for center personnel

Section 02

Sensitivity Studies and System Modeling



COMPASS Project Roadmap

	Year 1 (FY26)				Year 2 (FY27)				Year 3 (FY28)				Year 4 (FY29)				Year 5 (FY30)			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
AI Performance / Modeling	Analyze AI model sensitivity				Model data collection & augmentation				Decision under uncertainty				Model testing & validation							
					Research model uncertainty quantification				Research model transfer learning				Enhanced model for optimization integration							
Cross-Layer / Optimization	Assess sparse solver optimizations		Assess Beatnik optimization				Layer-by-layer optimization		Search-based optimization				Integrate model uncertainty				Fully integrate models			
			Scheduler optimization sensitivity		Assess NNSA app optimization opportunities															
Solver Design	Beatnik fast multipole support		Z-Model adaptive meshing		Compare Krylov solver variants				Parallelize topological transitions		Specialize models for physics problems				Integrate models into production codes					
	Develop compressible interface model			Model topological transitions			Multi-scale reactive interface methods					Parallelize multi-scale reactive solver								
				Parallelize compressible interface model					Combine Krylov enhancements into unified solver											
System Software	MPI/Kokkos stream triggered communication				Additional MPI/Kokkos extensions						Time-based communication primitive optimization									
	Create initial RAPIDS primitives			Additional RAPIDS extensions								Cross-layer control of system software APIs								
			Kokkos memory layout optimizations			Develop System Tuning APIs														
Integration & / Assessment	Design app/system workflow						Assess Z-Model/FIESTA						Assess Krylov Variants							
	Design/deploy system infrastructure						Assess NNSA application optimization						Assess integrated hydro codes							
Education & / Workforce	Create bootcamp materials					Develop tradeoff analysis materials						Create AI model course materials			Offer courses with integrated materials					
			Deploy bootcamp modules					Create conference tutorial materials			Offer HOTI/SIAM/SC tutorials									



Agents Plane

Interpret specifications

Generate experiments

Operate within guardrails

Adapt based on results

Application Specification

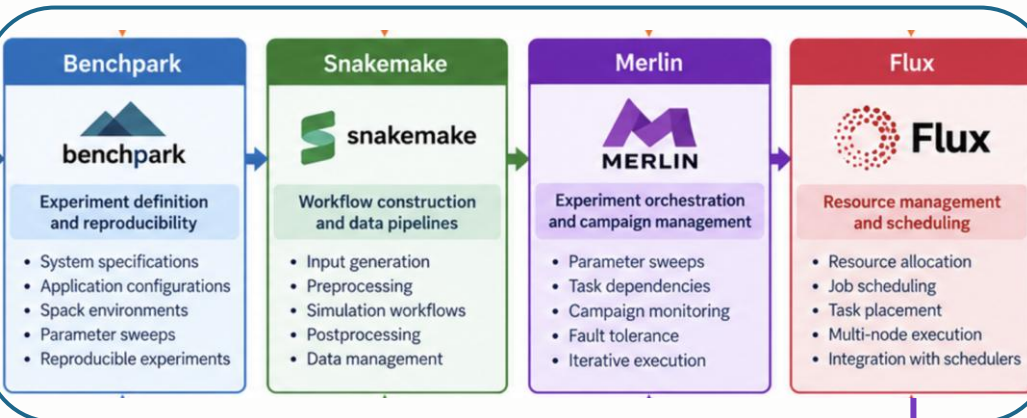
Example (TOML)

```
description = "AMG Solver"
docs = ["docs/amg_docs.pdf"]
[run]
template = "amg -P {Px} {Py}
          {Pz} -n {nx} {ny} {nz}
          -problem {problem}"

[run.params.problem]
type = "choice"
choices = [1, 2]
description = "Problem number"
```

- Application documentation
- Parameter space and allowed values
- Resource and execution constraints

Experiment Planning and Orchestration
(existing tools - optional)



Analysis and Adaptation

Performance Models

- Predict performance
- Quantify uncertainty
- Identify informative experiments

Experiment results

- Runtime and scalability
- System telemetry
- Metadata and logs
- Structured dataset

Data Collection



Applications



HPC Platforms

Agents Plane

Interpret specifications

Generate experiments

Operate within guardrails

Adapt based on results

Application Specification

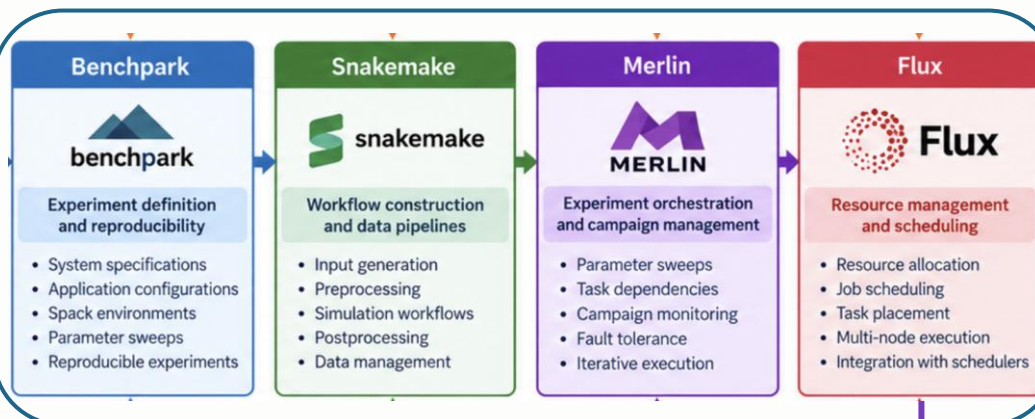
Example (TOML)

```
description = "AMG Solver"
docs = ["docs/amg_docs.pdf"]
[run]
template = "amg -P {Px} {Py}
{Pz} -n {nx} {ny} {nz}
-problem {problem}"

[run.params.problem]
type = "choice"
choices = [1, 2]
description = "Problem number"
```

- Application documentation
- Parameter space and allowed values
- Resource and execution constraints

Experiment Planning and Orchestration (existing tools - optional)



Analysis and Adaptation

- Performance Models**
- Predict performance
 - Quantify uncertainty
 - Identify informative experiments

- Experiment results**
- Runtime and scalability
 - System telemetry
 - Metadata and logs
 - Structured dataset

Data Collection



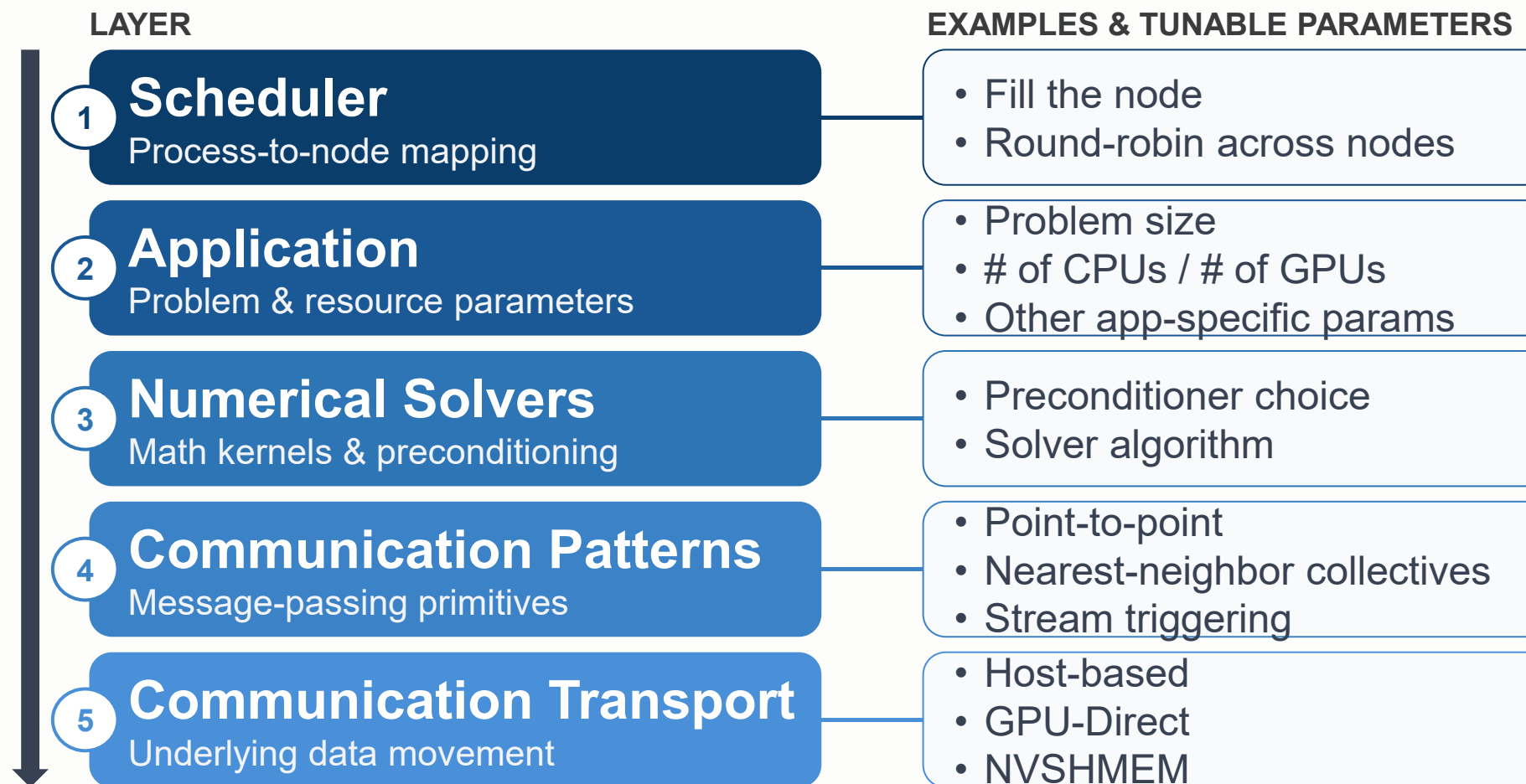
Applications



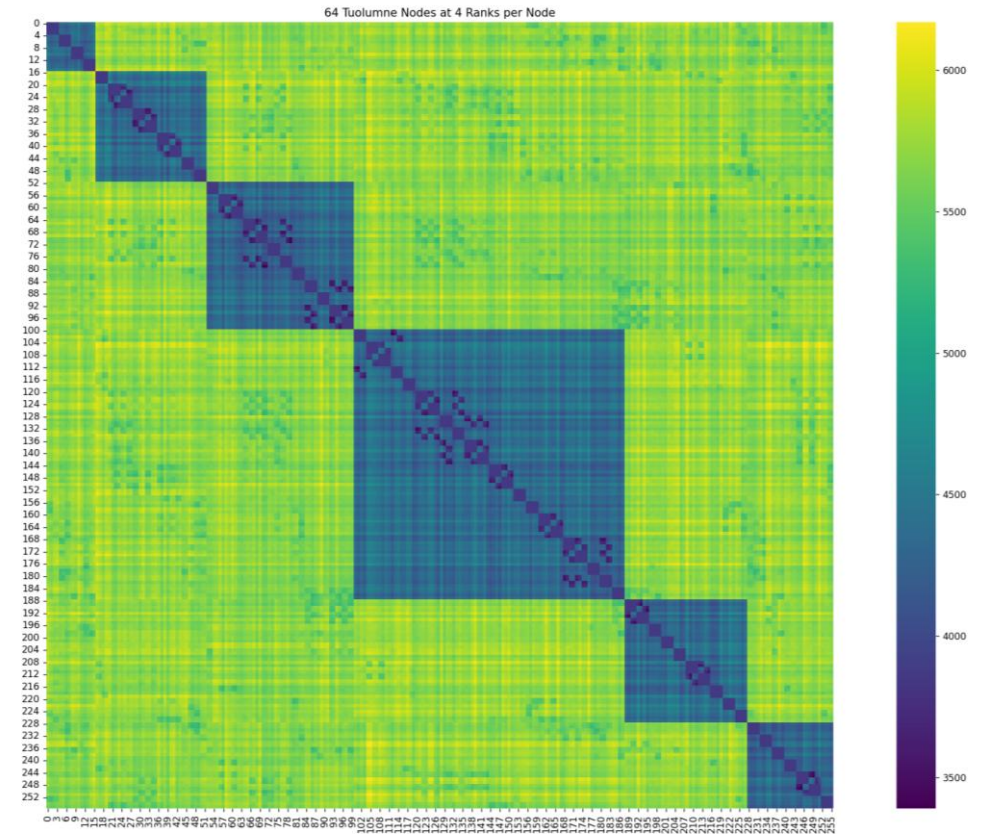
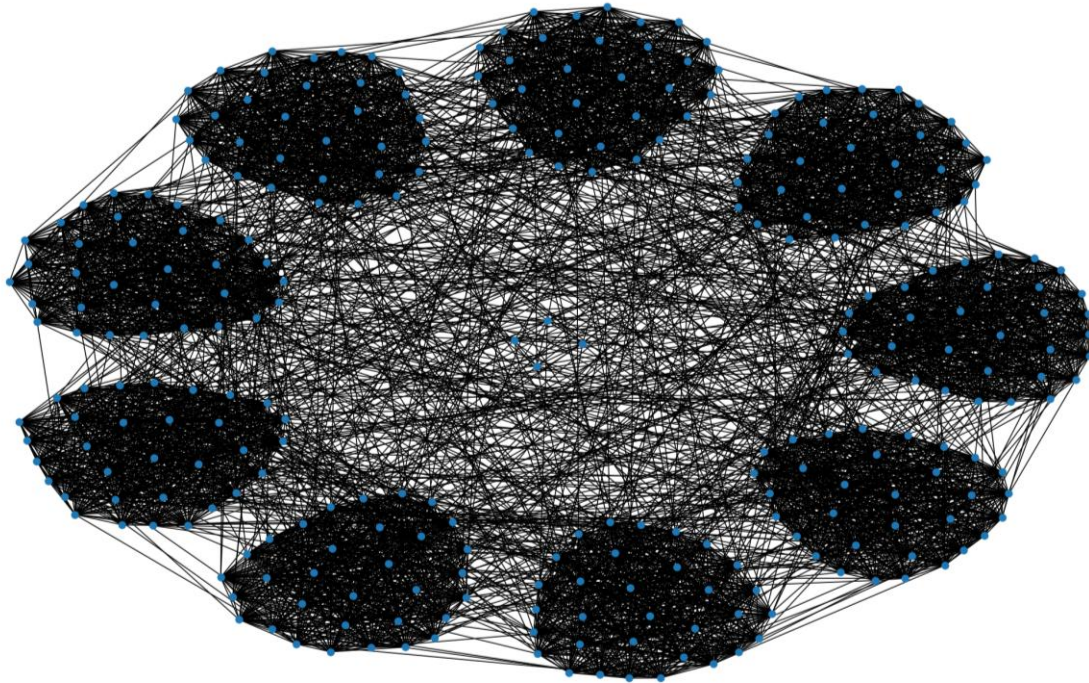
HPC Platforms

HPC Software Stack Dependencies

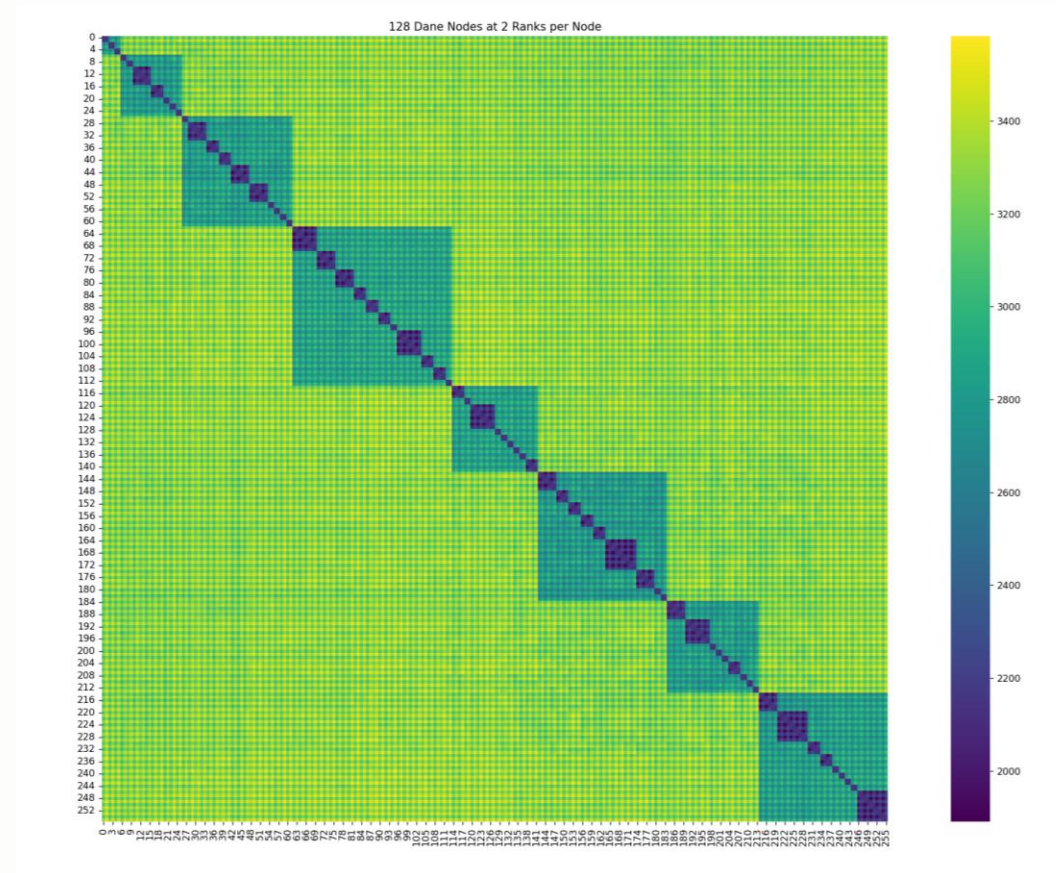
Each layer's tunable parameters depend on choices in the layers below



Network Topology and Latency Effects on Tuolumne



Latency Effects on Dane



Available Telemetry – System Perspective

Compute & Accelerators

Monitoring core processing power, tracking GPU utilization, memory activity, power consumption, and thermal thresholds.

Nvidia DCGM, proc stat

Interconnect & Network

Tracking data flow efficiency by measuring congestion, packet retries, network stalls, and overall communication activity.

Slingshot, InfiniBand Counters

Storage & I/O

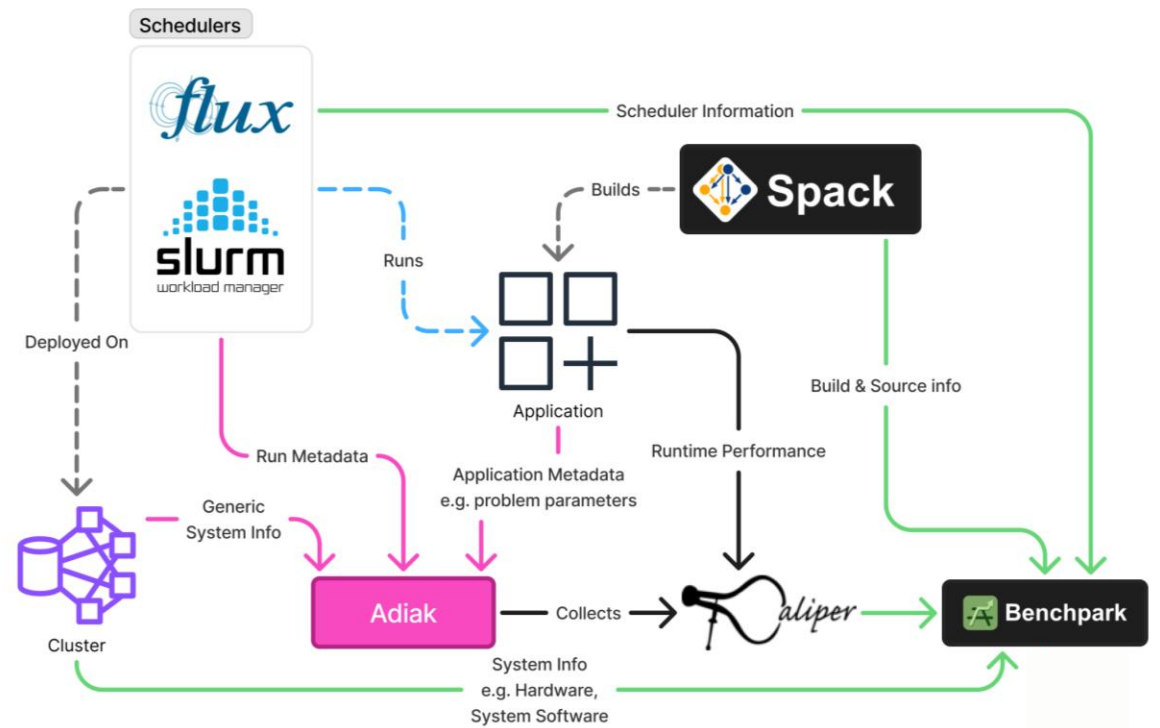
Analyzing persistent data pathways through read/write speeds, complex access patterns, and metadata operations.

Darshan

Facilities collect this information in isolation and often lack a unified job level view.

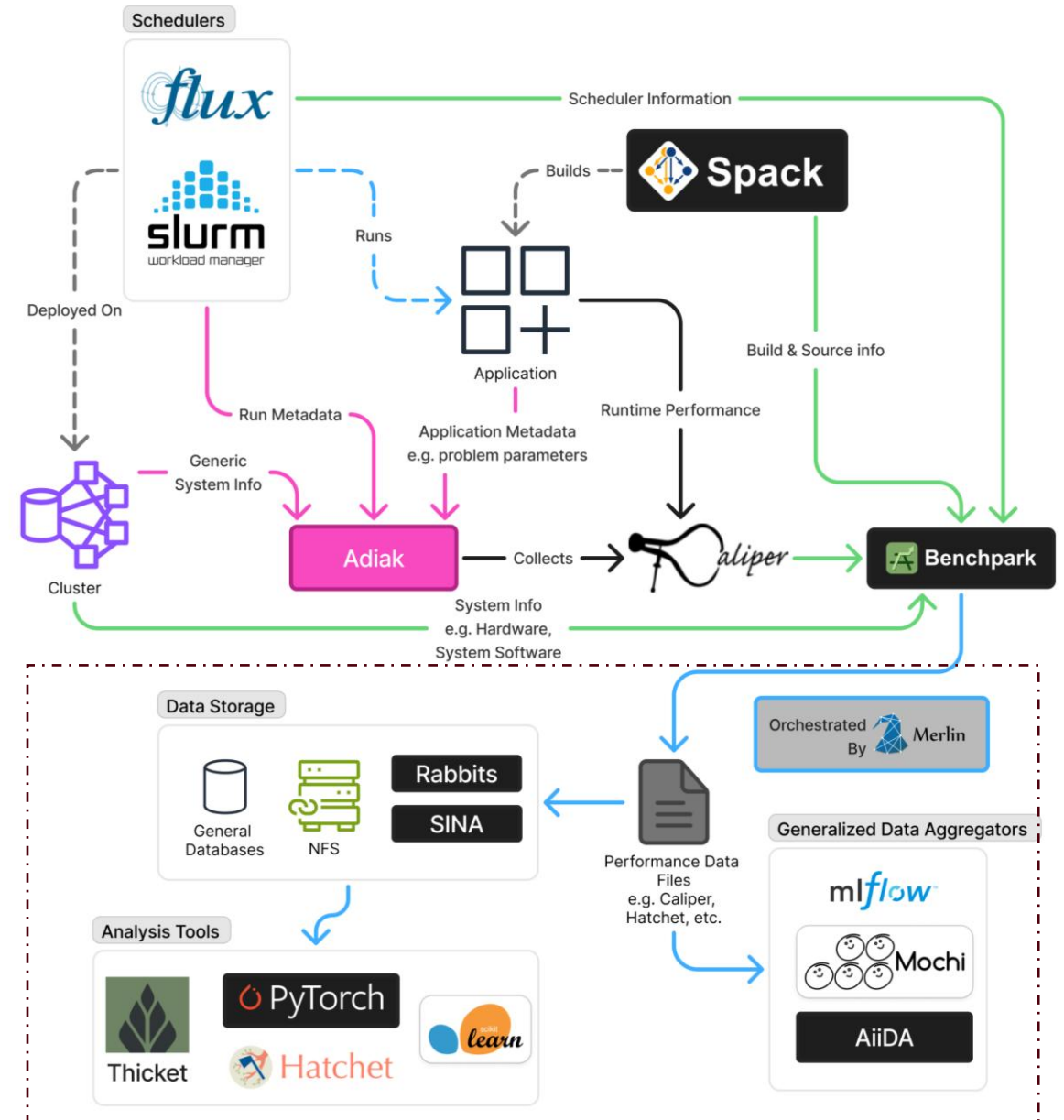
Available Telemetry – Application Perspective

- Application generated performance data
- Non-intrusive profiling data (e.g., TAU, Caliper, HPCToolkit)
- Intrusive/Explicit profiling data
- Scheduler-level job metadata (e.g., nodes details, wall clock time)



Experiment Management

- Many tools and frameworks
- Current gaps are in the following areas:
 - Organizing profiling data
 - Tools to visualize the data
 - Setting up automated AI/ML pipelines
- Efforts currently underway to address these issues



HPC Workflow Tool Comparison

Running Experiments

Tool	Generic Apps	Headless	Custom Scripts	Scalable	Scheduler	Monitoring	Failure Recovery	Adaptive	Cross-Machine
AiiDA	✗	✓	✓	✓	Slurm Flux LSF PBS SGE	CLI Queryable	Yes	Yes	Partial
Benchpark	✗	✗	✗	✓	Slurm Flux LSF	None	Partial	No	Yes
Fireworks	✓	✓	✓	✓	Slurm Flux LSF PBS SGE	CLI Web GUI	Yes	Yes	Partial
Merlin	✓	✓	✓	✓	Slurm Flux LSF PBS Inside allocation	CLI	Yes	Yes	Partial
Snakemake	✓	✗	✓	✓	Slurm Flux LSF PBS SGE	CLI TUI Report Third-party	Yes	Partial	Partial

<https://cs.unm.edu/~nlivingstone/hpc-tools/hpc-workflow-tool-comparison.html>

HPC Workflow Tool Comparison

Data & Reproducibility

Tool	Database	Reproducible	Provenance	ML Export	Build	Config
AiiDA	✓	✓	Automatic Inputs/Outputs DB	Low	None	Python
Benchpark	✗	✓	Software Inputs/Outputs Config/Spec Caliper/Adiak	Medium	Spack	Python CLI
Fireworks	✓	✓	Software Inputs/Outputs DB	Medium	None	YAML JSON Python
Merlin	✗	✓	Software Config/Spec	Medium	None	YAML
Snakemake	✓	✓	Software Inputs/Outputs Config/Spec	Medium	Spack (WIP) Conda Containers Modules	Python DSL

<https://cs.unm.edu/~nlivingstone/hpc-tools/hpc-workflow-tool-comparison.html>

HPC Workflow Tool Comparison

Effort & Upkeep

Tool	New App	New Sweep	Run Steps	Infrastructure
AiiDA	Medium	Low	1	PostgreSQL RabbitMQ Daemon LC can host
Benchpark	High	Medium	5	None
Fireworks	Low	Medium	2	MongoDB LC can host
Merlin	Low	Low	2	RabbitMQ Redis LC can host
Snakemake	Low	Low	1	None

Key Takeaways

- Many tools available each with specific feature sets – no clear winner
- We must incorporate multiple tools to support the AI/ML workflows
- Exploration with AI/ML agents currently underway

Applications Considered for Evaluation

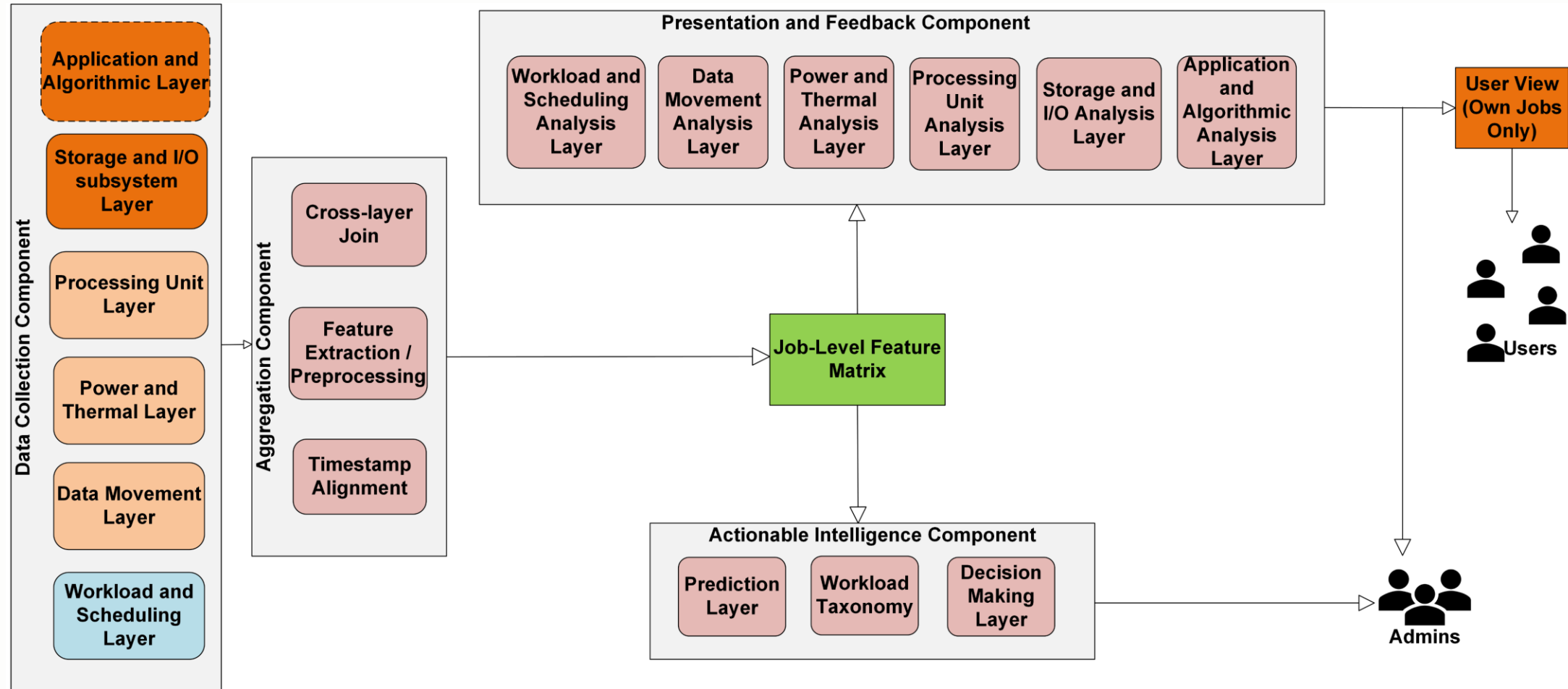
- AMG2023, HYPRE → MFEM
- Trilinos Block Jacobi/Block Tridiagonal → SPARC
- Trilinos MueLu, MiniEM → EMPIRE
- Parthenon/VIBE, Parthenon/RIOT → LANL finite volume multi-physics application

Ongoing Projects

- Node-level performance measurements
 - Do AMD and NVIDIA Count the Same Thing? – Abouelmagd, TN Tech
 - Partitioning the AMD MI300A: Locality Rules, Power Throttles – Abouelmagd, TN Tech
- Communication-related optimizations
 - Optimizing MPI Parameters under Uncertainty – Barr, UNM
 - Guided Analysis for Understanding Global Communication Efficiency – Bacon, UNM
 - Multi-Strategy Adaptive Optimization of High Performance Communication – Kinkead, UNM
- Optimization across multiple layers
 - Cross-Layer Characterization Framework for HPC Applications – Jajula, UA

Please see posters for more details.

Cross-Layer Characterization Framework



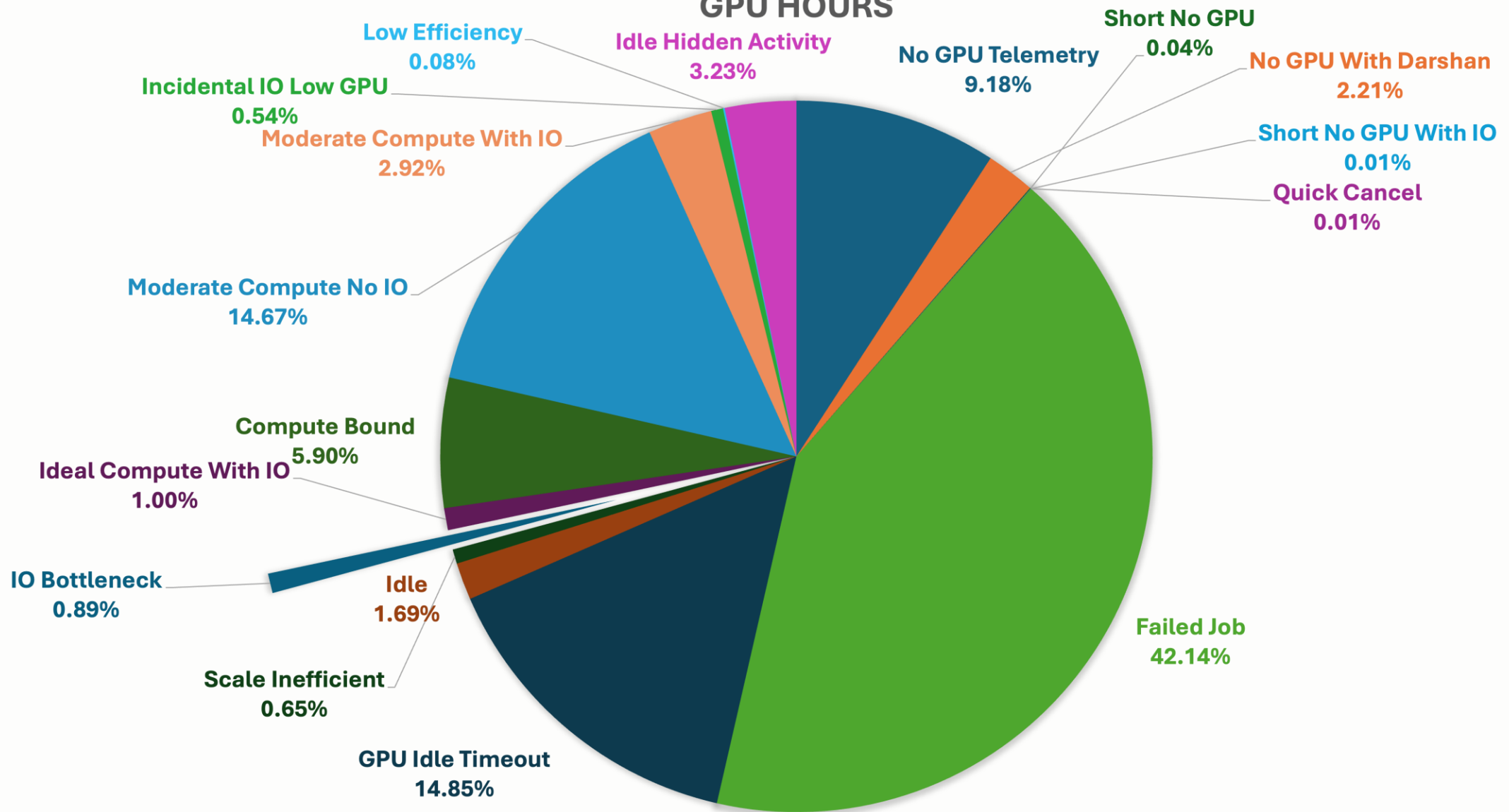
ALCF Polaris Public Data – Evaluation

262,634 jobs across 13 months, totaling 16.53M allocated GPU-hours

Source	Job-level information	Coverage
Scheduler	Allocation, runtime, exit status	All jobs
DCGM	GPU utilization, power, thermal state	78.1%
Darshan	I/O timing, bytes, metadata, imbalance	25.2%

Joint DCGM + Darshan coverage: 18.4%. Missing telemetry is dismissed.

GPU HOURS



Conclusions

- Current focus is on deploying infrastructure to support Cross-Layer Optimization and AI Performance Modeling
- Applications to evaluation identified (in consultation with TST) and initial evaluation work underway
- Continue with system infrastructure development and evaluation activities to support AI/ML performance modeling



This work is supported in part by the Predictive Science Academic Alliance Program (PSAAP) award DE-NA0004267

Questions?

