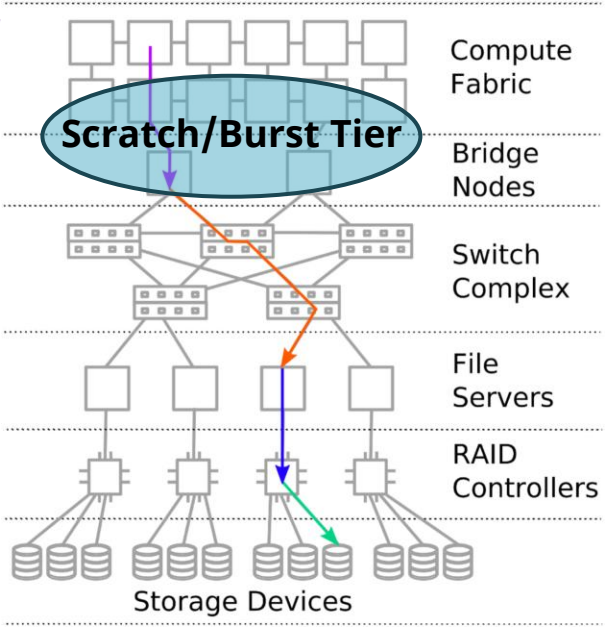
Redefining HPC Observability: Integrating Monitoring, Modeling, and Meaning

Sarah M. Neuwirth
Johannes Gutenberg University Mainz, Germany
neuwirth@uni-mainz.de

MODA 2025 Workshop, ISC High Performance, June 2025

Motivation
Heterogeneous and Complex HPC Infrastructures

- HPC infrastructure ***too complex***, humans are ***overwhelmed***
- Complexity and scope increase the ***urgency***
  - *New computational paradigms* (AI/ML apps vs. BSP-style HPC)
  - *New architectural directions* (e.g., IPU, RISC-V, data flow)
  - *Heterogeneity overall*: node architectures, within the system, storage and parallel file system during application design (e.g., ML within HPC applications)
  - *New operations paradigms* (e.g., cloud, container)
  - Simplistic approaches to increasing compute demand result in *unacceptable power costs*
- Difficult for humans to optimally adapt applications to systems and to detect and diagnose vulnerabilities

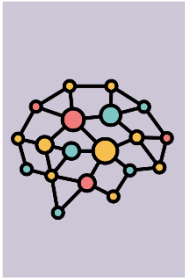


Carns, P., 2023. *HPC Storage: Adapting to Change*. Keynote at REX-IO'23 Workshop.

Ciorba, F., 2023. *Revolutionizing HPC Operations and Research*. Keynote at HPCMASPA'23 Workshop.

B. Settlemyer, G. Amvrosiadis, P. Carns and R. Ross, 2021. *It's Time to Talk About HPC Storage: Perspectives on the Past and Future*, in Computing in Science & Engineering, vol. 23, no. 6, pp. 63-68.

Motivation
Why Redefine Observability?



HPC is no longer just HPC

- From simulation → **simulation + AI + data analytics**
- **Complex workflows** across diverse compute, storage and network layers
- Traditional metrics (e.g., FLOPS, bandwidth) do not capture this complexity

Monitoring ≠ Observability

- Monitoring tells **what** happened
- Observability helps us understand **why** it happened and what to do next
- We need more than logs and counters – we need **explainability** and **context**

New Questions need new tools

- Why is my workflow slow now, but not yesterday?
- Is the problem in the application, the file system, or the kernel?
- Can I trust this model's I/O behavior across systems?

Actionable Insight Requires Integration

- No single tool sees the full stack
- We need to connect:
  instrumentation → modeling → decision-making

Monitoring:
The Foundation – But Not the Full Picture

Monitoring: The Foundation

Vision: Holistic Monitoring for Intelligent HPC Operations

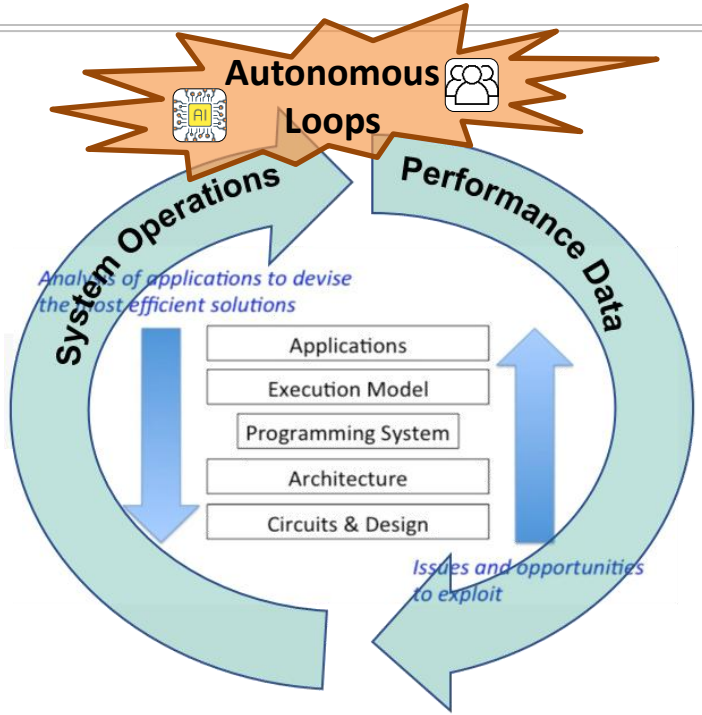
What We Need:

- Continuous monitoring, archiving, and analysis of operational + performance data
- Unified visibility into applications, system software, and hardware layers

Why It Matters:

- Enables automated feedback loops using AI/ML
- Supports dynamic workload & architecture analysis
- Powers adaptive, actionable responses

Goal: Efficient, explainable HPC operations driven by *autonomous analyze-feedback-response loops*

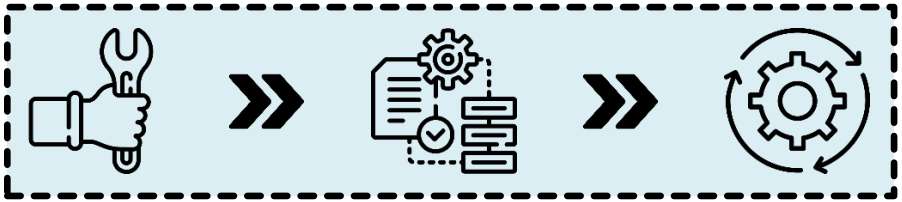


Gentile, A., 2021. *Enabling Application and System Data Fusion*. Keynote at MODA'21 Workshop.

Ciorba, F., 2023. *Revolutionizing HPC Operations and Research*. Keynote at HPCMASPA'23 Workshop.

Dagstuhl Seminar 23171, 2023. *Driving HPC Operations With Holistic Monitoring and Operational Data Analytics*. https://www.dagstuhl.de/23171

Monitoring: The Foundation
Different Perspectives



User Interests and Concerns:

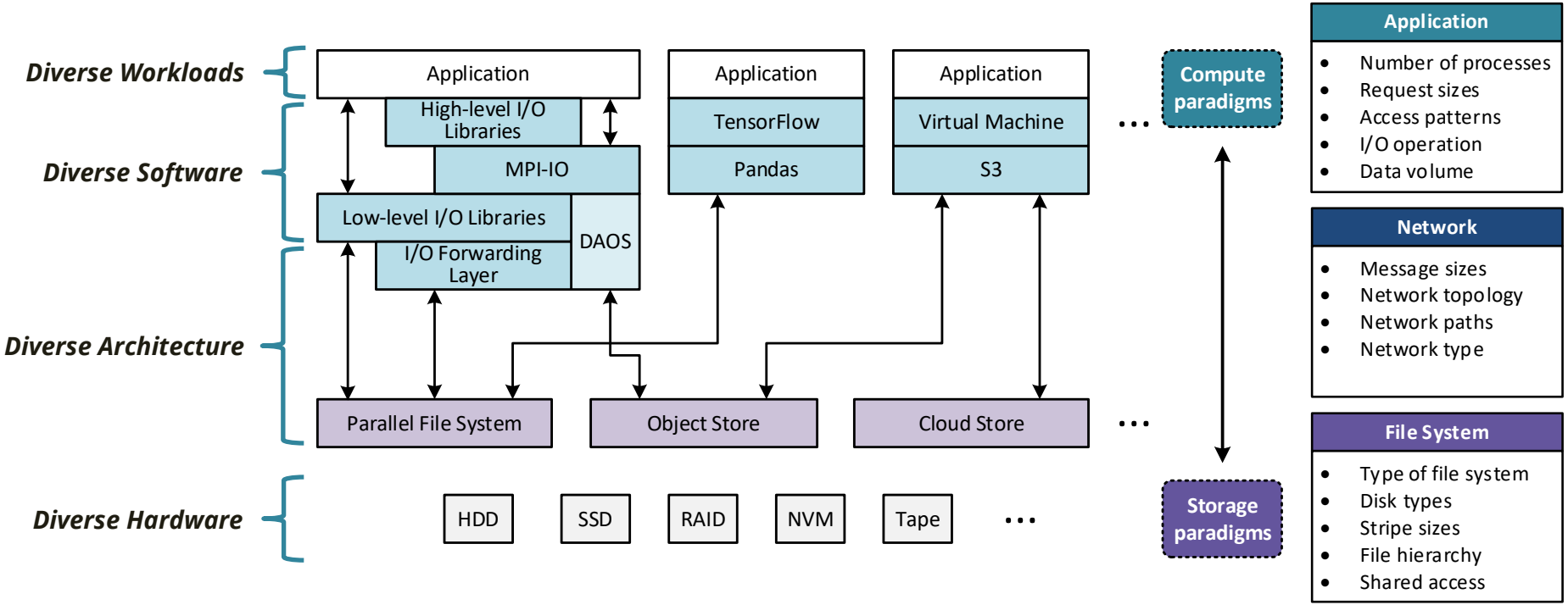
- Ease of use
- Application performance
- Portability
- Reproducible science
- Accessibility of the system
- Data persistence
- Core hour usage

System Interests and Concerns:

- Installation, configuration, and operation of the production-ready system, e.g.:
  - Software requirements
  - System configuration
  - High availability service
- System monitoring
- System security
- Benchmarking and anomaly detection

Monitoring: The Foundation

Parallel I/O System and Performance Factors



Monitoring: The Foundation

Example: Darshan I/O Characterization Tool

Blue Waters, Mira, and Theta popular Darshan log sources used for research:

- https://bluewaters.ncsa.illinois.edu/data-sets
- https://reports.alcf.anl.gov/data/
- ftp://ftp.mcs.anl.gov/pub/darshan/data

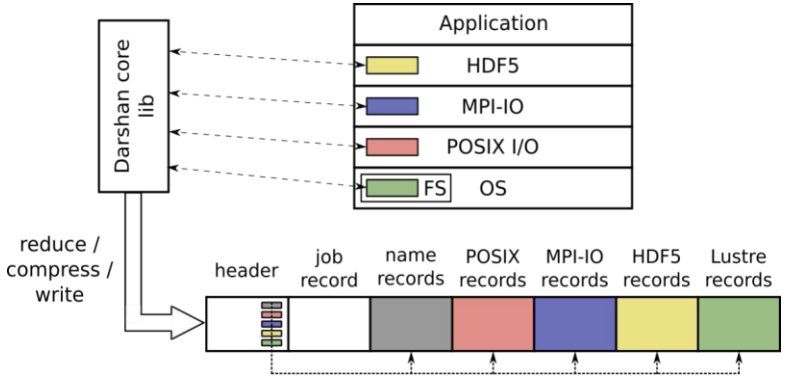
Open questions:

- How relevant are the logs to current systems?
- How do we know the integrity of the logs?

Community comments:

- "Darshan is one of the first tools to be deactivated in the event of I/O problems."
- "Darshan cannot grasp the complexity of state-of-the-art parallel storage systems."

Darshan I/O Characterization Tool



Snyder, S., 2022. *Darshan: Enabling Insights into HPC I/O Behavior*. ECP Community BoF Days.

What are the implications of these questions and observations?

Monitoring: The Foundation

Fragile Pipelines – Tool-Driven, Not Insight-Driven

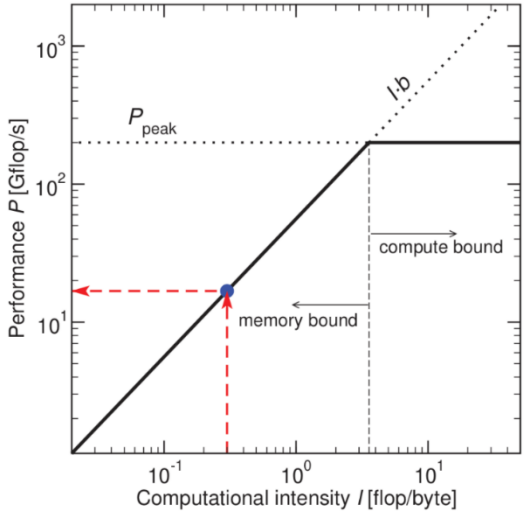
Category	Examples	Strengths	Limitations
Application-level	Darshan, Recorder, Score-P	Fine-grained function tracing	No visibility into system-wide interactions
System-level	LDMS, DCDB, TACCStats	Aggregated I/O performance metrics	Cannot correlate application performance with system metrics
End-to-end	Ganglia, Nagios, Apollo	Holistic view of system utilization	Lacks deep profiling at kernel and network levels

- **Siloed Tool Views:** *Each tool sees a layer. None explain the whole system.*
  => For example, in case of I/O: App-level profilers (e.g., Darshan) vs. system tools (LDMS, DCDB)
- **Workflow Scripts Instead of Workflows:** *Custom scripts per experiment = unscalable, unrepeatable.*
  => Benchmarking tools (e.g., iperf, sockperf) often require client/server logic incompatible with SLURM
- **Discarded Insights:** *Performance data is ephemeral; models are not reused.*
  => No structure for reuse → repeated effort, lost opportunities

Modeling: From Data to Understanding

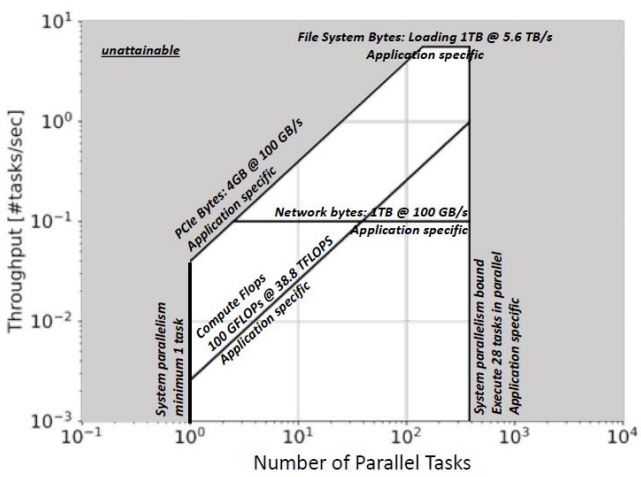
Modeling: From Data to Understanding
Roofline Characterization

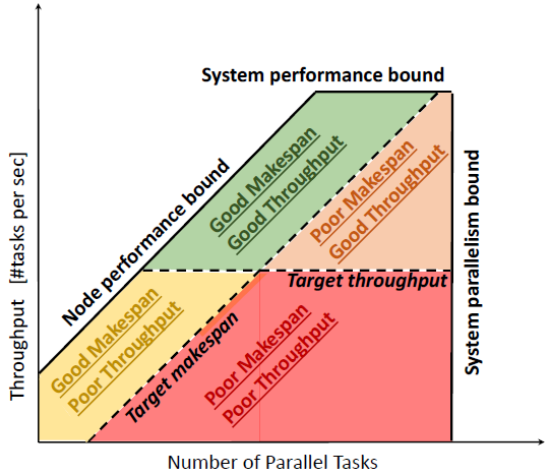
Traditional Roofline Characterization



Williams, S., Waterman, A. and Patterson, D., 2009. *Roofline: an insightful visual performance model for multicore architectures.* Communications of the ACM, 52(4), pp.65-76.

Workflow Roofline Characterization





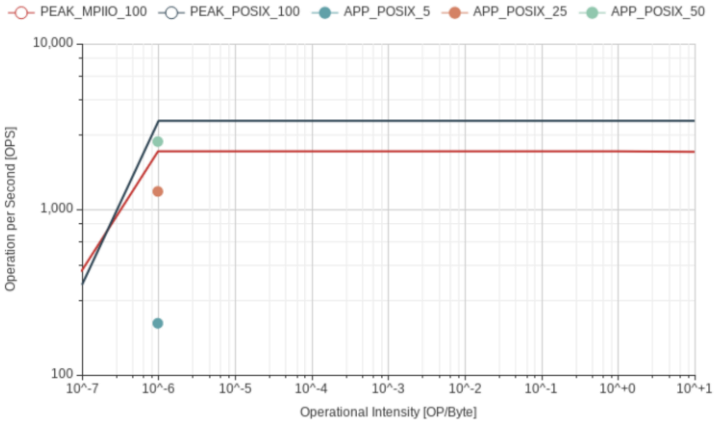
Ding, N., Austin, B., Liu, Y., Mehta, N., Farrell, S., Blaschke, J.P., Oliker, L., Nam, H.A., Wright, N.J. and Williams, S., 2024, November. *A Workflow Roofline Model for End-to-End Workflow Performance Analysis*. In SC24: International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-15). IEEE.

Modeling: From Data to Understanding
I/O Roofline Characterization: Initial Concept

- Traditional Roofline Model…
  - is based on looking at the *relationship between work and traffic*
  - provides intuitive approach through simple bound and bottleneck analysis
- ***I/O Roofline Characterization*** is based on IOPS and the I/O bandwidth => *I/O interface specific*, i.e., POSIX, MPIIO, etc.
  - *IOPS*: number of reads and writes that a storage system can perform per second
  - *Bandwidth*: total amount of data read or written per second
- **X-axis:** *I/O Operational Intensity* $= \frac{\text{Total I/O Operations}}{\text{Read Bytes+Write Bytes}}$
- **Y-axis:** $P = \min(\text{Peak IOPS}, \text{Peak I/O Bandwidth} \times \text{I/O Intensity})$ where P is the attainable perf., P_{peak} is the peak perf., b is the peak bandwidth, and I is the arithmetic intensity

Zhu, Z., Bartelheimer, N. and Neuwirth, S., 2023. *An Empirical Roofline Model for Extreme-Scale I/O Workload Analysis.* IPDPSW'23.

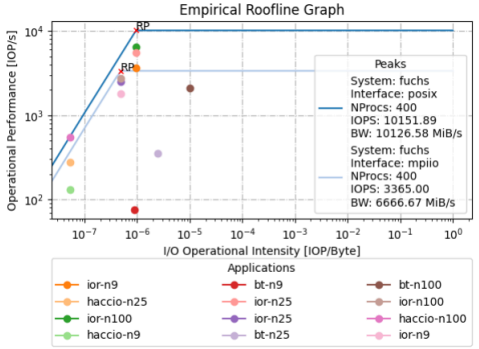
Zhu, Z., and Neuwirth, S., 2023. *Characterization of Large-Scale HPC Workloads With Non-Naïve I/O Roofline Modeling and Scoring.* ICPADS'23.

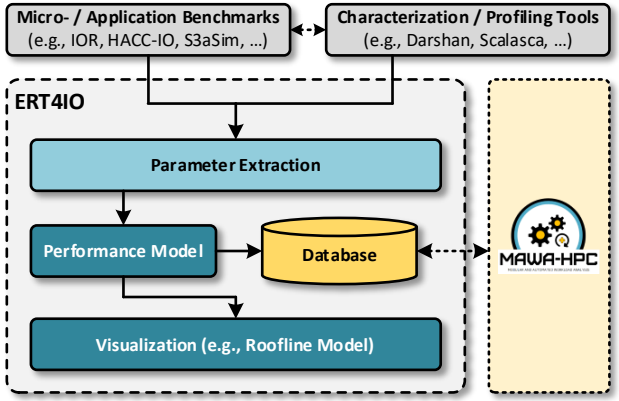


Modeling: From Data to Understanding

I/O Roofline Characterization: Workflow Implementation

- *ERT4IO (Empirical Roofline Tool for I/O)* provides automated I/O Roofline characterization
  - Parameter extraction from Darshan logs
  - Generates Roofline visualization
- Forwards results to MAWA-HPC framework
  - Enables further data analysis
  - Preserves and shares knowledge via performance history database with the HPC community
- Applicable to various use cases
  - Systems with different configurations and hardware can be compared and evaluated
  - Intuitive estimation of an application's I/O performance
  - Identifying performance bottlenecks and anomalies

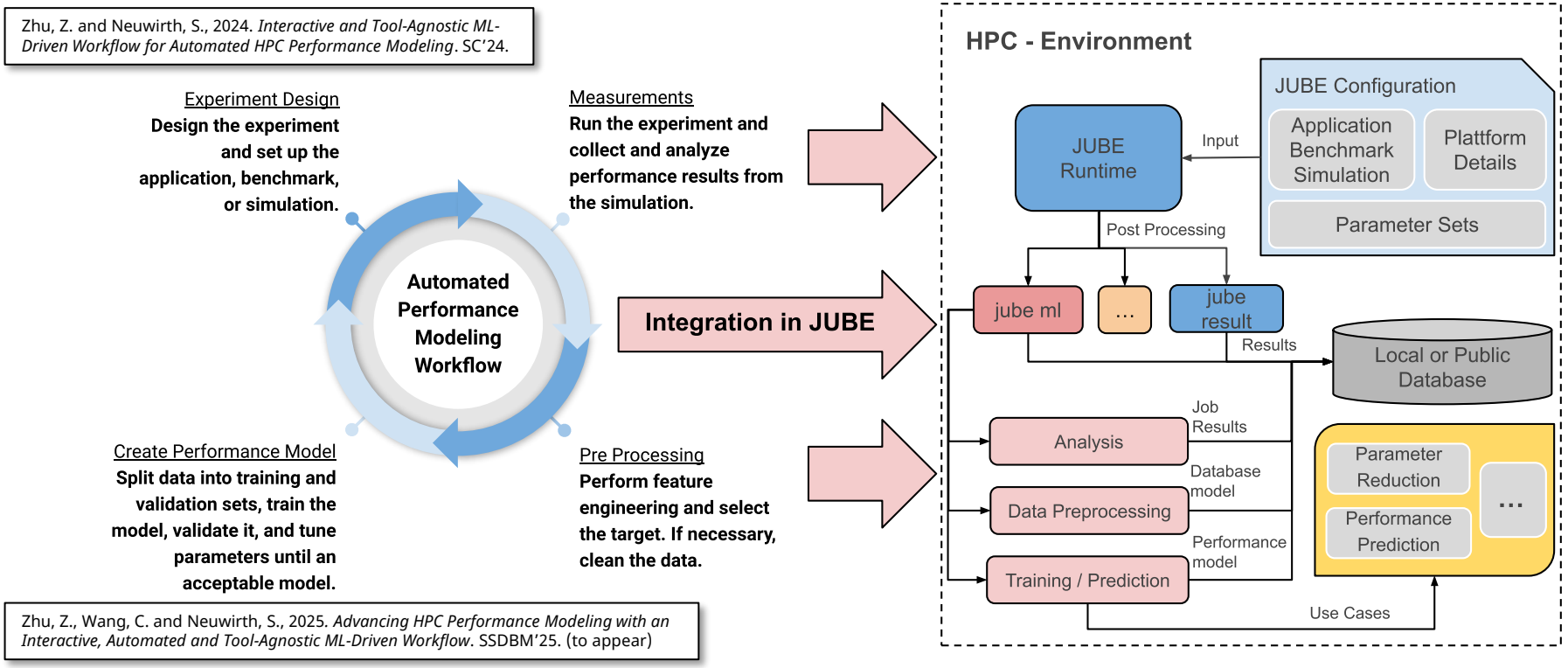




Modeling: From Data to Understanding

Automated ML-Driven Performance Modeling Workflow

Zhu, Z. and Neuwirth, S., 2024. *Interactive and Tool-Agnostic ML-Driven Workflow for Automated HPC Performance Modeling*. SC'24.



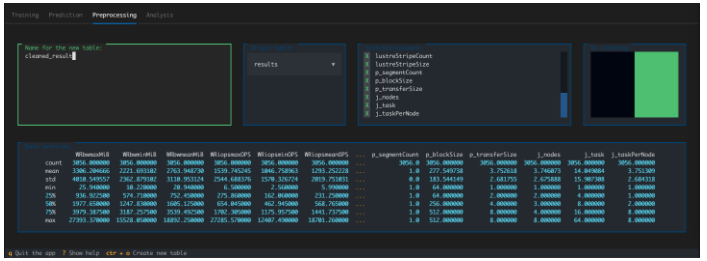
Zhu, Z., Wang, C. and Neuwirth, S., 2025. *Advancing HPC Performance Modeling with an Interactive, Automated and Tool-Agnostic ML-Driven Workflow*. SSDBM'25. (to appear)

Modeling: From Data to Understanding

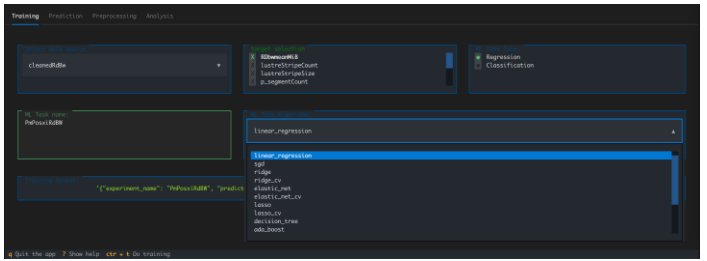
JUBE-ML Prototype Implementation

- Prototype *extends the JUBE framework* with support for automatic ML-based performance modeling and supports variety of ML algorithms
- JUBE-ML is enhanced by the *sqlite-ml extension*
- JUBE-ML stage provides *four key functionalities*:
  - ***Data analysis:*** insights into performance results (e.g., min, max, mean) and lambda functions
  - ***Data preprocessing:*** cleans and filters data for ML, creates new tables with selected features & targets
  - ***ML model training:*** applies ML models (regression or classification) to build a performance model
  - ***Prediction:*** validates the model & enables different analysis scenarios, such as identifying irrelevant parameters and predicting system performance

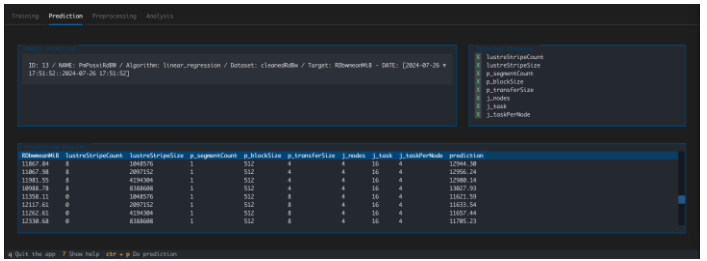
PRE-PROCESSING AND ANALYSIS



TRAINING PHASE



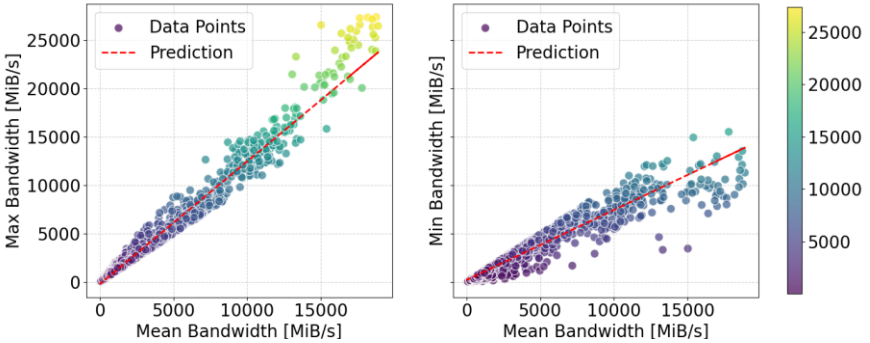
PREDICTION AND VALIDATION



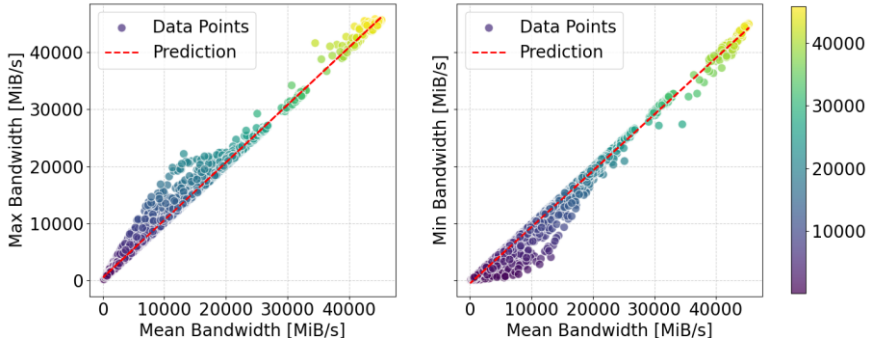
Modeling: From Data to Understanding
JUBE-ML Case Study: I/O Bandwidth Prediction

- *Use Case:* I/O bandwidth modeling and prediction to demonstrate the JUBE-ML workflow
  - IOR benchmark to simulate various I/O workloads and generate performance data
  - Linear regression: 25% for training, 75% for validation
- Despite the small training sample, the model predicts both minimum & maximum bandwidths well

Block size	64MB, 256MB, 512MB
Transfer size	1MB, 2MB, 4MB, 8MB
Lustre striping count	0, 2, 4, 8
Lustre striping size	1MB, 2MB, 4MB, 8MB
Nodes	1, 2, 4, 8
Tasks per node	1, 2, 4, 8



POSIX write performance prediction with JUBE-ML.



POSIX read performance prediction with JUBE-ML.

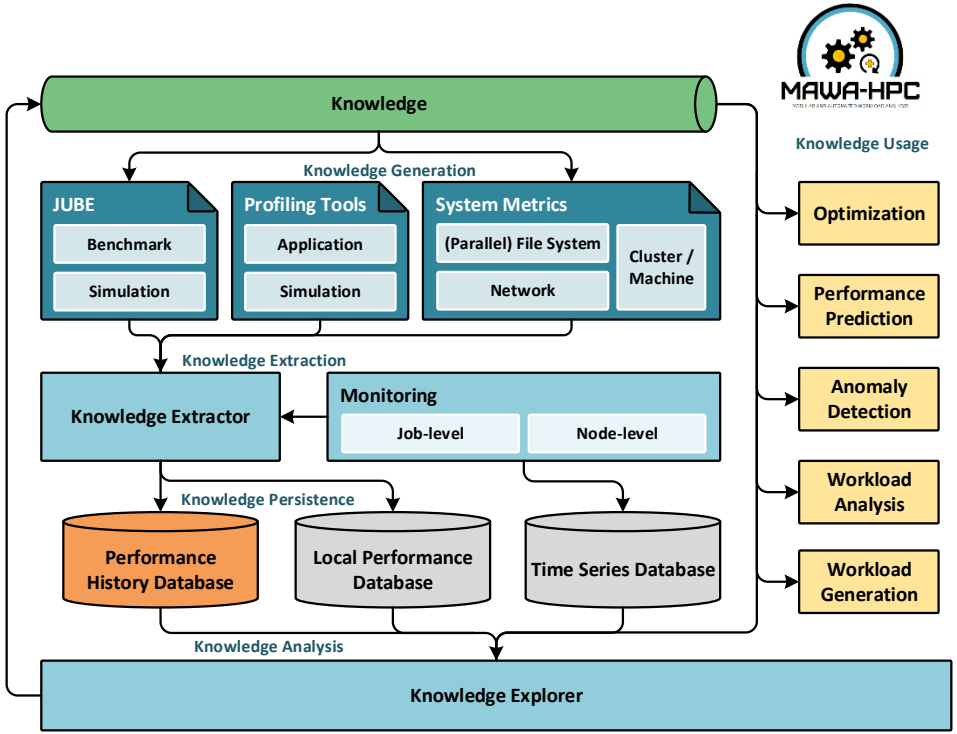
Integration: Bridging Layers and Silos

Integration: Bridging Layers and Silos

Goal: Holistic & Automated Monitoring and Analysis Cycle

Idea: Develop and implement standardized and tool-independent approach for HPC workload and application analysis

Goal: Establish a ***performance history database*** to categorize systems, workload behaviors, and characteristic patterns for different science domains



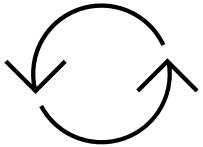
Zhu, Z., Bartelheimer, N. and Neuwirth, S., 2023. *MAWA-HPC: Modular and Automated Workload Analysis for HPC Systems*. ISC'23.

Bartelheimer, N., Zhu, Z., and Neuwirth, S., 2023. *Toward a Modular Workflow for Network Performance Characterization*. IPDPSW'23.

Zhu, Z., and Neuwirth, S., 2023. *Characterization of Large-Scale HPC Workloads With Non-Naïve I/O Roofline Modeling and Scoring*. ICPADS'23.

Integration: Bridging Layers and Silos

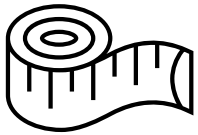
Reproducible Benchmarking: Integration is Key



Repeatability

- Ensures consistent results from the same setup
- Builds trust in computational outcomes
- Forms the basis for scientific validation

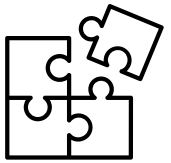
Schifrin, A., 2023. *Automated Performance Characterization of HPC Systems.* Bachelor thesis, Goethe University Frankfurt.



Comparability

- Standardizes benchmarking across models or methods
- Allows fair evaluation of new approaches
- Helps identify performance trade-offs

Bartelheimer, N. and Neuwirth, S., 2023. *Toward Reproducible Benchmarking of PGAS and MPI Communication Schemes.* ICPADS'23.



Modularity

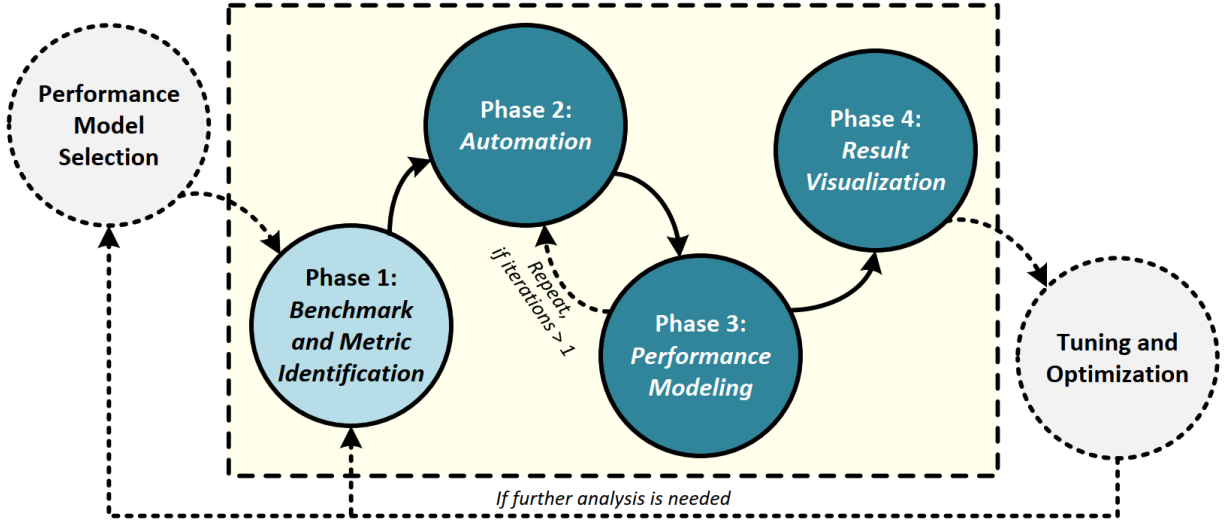
- Enables flexible software architecture
- Facilitates integration of AI/ML components
- Supports code reuse and maintainability

Zhu, Z., Wang, C. and Neuwirth, S., 2025. *Advancing HPC Performance Modeling with an Interactive, Automated and Tool-Agnostic ML-Driven Workflow*. SSDBM'25. (to appear)

Integration: Bridging Layers and Silos

Workflow Design for Reproducible Benchmarking

Core component of the benchmarking framework is the **JUBE Benchmarking Environment**




```
Data: Message Size m > 0, Iterations i > 0, Window
      Size w > 0
M ← 1;
while M ≠ m do
    I ← 0;
    B ← allocate(M);
    while I < i do
        T ← Time();
        W ← 0;
        while W < w do
            C ← B + W · M;
            communicate(C, M);
            W ← W + 1;
        end
        T ← Time() − T;
        I ← I + 1;
    end
    deallocate(B);
    M ← M · 2;
end
```

Algorithm 1: Typical benchmark loop structure.

Bartelheimer, N., Zhu, Z., and Neuwirth, S., 2024. *Automated Network Performance Characterization for HPC Systems*. International Journal of Networking and Computing.

JUBE Documentation: https://apps.fz-juelich.de/jsc/jube/jube2/docu/index.html

Integration: Bridging Layers and Silos

Reproducible Benchmarking: Example Configuration

Benchmark-independent, Platform-specific

`platform.xml`

```
<parameterset name="systemParameter">
    <parameter name="nodes" type="int">2</parameter>
    <parameter name="tasks" mode="python" type="int">$nodes * $taskspernode</parameter>
    <parameter name="timelimit">00:30:00</parameter>
</parameterset>
```

```
<parameterset name="executeset">
    <parameter name="submit">sbatch</parameter>
    <parameter name="submit_script">submit.job</parameter>
    <parameter name="starter">srun</parameter>
</parameterset>
```

Benchmark-specific, Platform-independent

`likwid-specs.xml`

```
<parameterset name="copy_params">
    <parameter name="benchmark_set_cp" tag="copy">       copy</parameter>
    <parameter name="benchmark_set_cp" tag="copy_avx">   copy_avx</parameter>
    <parameter name="benchmark_set_cp" tag="copy_avx512">copy_avx512</parameter>
</parameterset>
```

```
<parameterset name="alloc_params">
    <parameter name="alloc" tag="W">-W</parameter>
    <parameter name="alloc" tag="w">-w</parameter>
</parameterset>
```

```
<patternset name="likwid_pattern">
        <pattern name="likwid_cycles" type="int" dotall="False" mode="pattern">Cycles:\s+$jube_pat_int</pattern>
        <pattern name="likwid_cpu_clock" type="int" dotall="False" mode="pattern">CPU Clock:\s+$jube_pat_int</pattern>
</patternset>
```

Benchmark-specific, Platform-specific

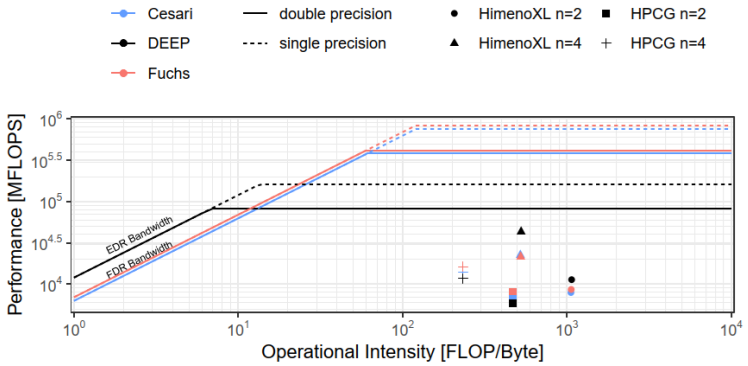
`platform-likwid-specs.xml`

```
<parameterset name="thread_domain_params">
    <parameter name="thread_domain" tag="td-n">  N</parameter>
    <parameter name="thread_domain" tag="td-s0">S0</parameter>
    <parameter name="thread_domain" tag="td-s1">S1</parameter>
    <parameter name="thread_domain" tag="td-d0">D0</parameter>
    <parameter name="thread_domain" tag="td-d1">D1</parameter>
    <parameter name="thread_domain" tag="td-c0">C0</parameter>
    <parameter name="thread_domain" tag="td-c1">C1</parameter>
    <parameter name="thread_domain" tag="td-m0">M0</parameter>
    <parameter name="thread_domain" tag="td-m1">M1</parameter>
</parameterset>
```

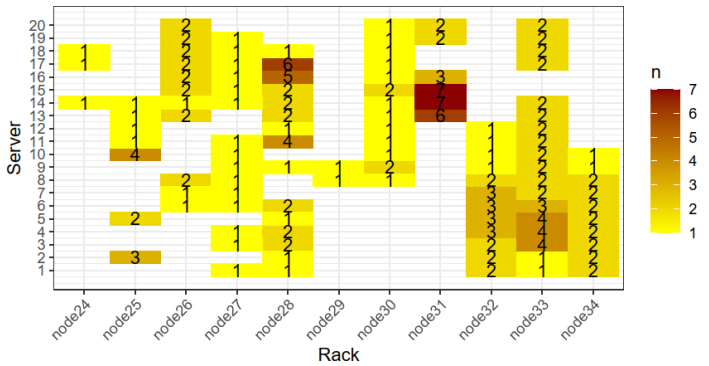
```
<parameterset name="suffix_params">
    <!--
        CHANGEME: this parameter can be changed and represents the parameter part after the <size>
        if not empty, the suffix must start with the ":" (colon) character
    -->
    <parameter name="suffix"></parameter>
</parameterset>
```

Integration: Bridging Layers and Silos

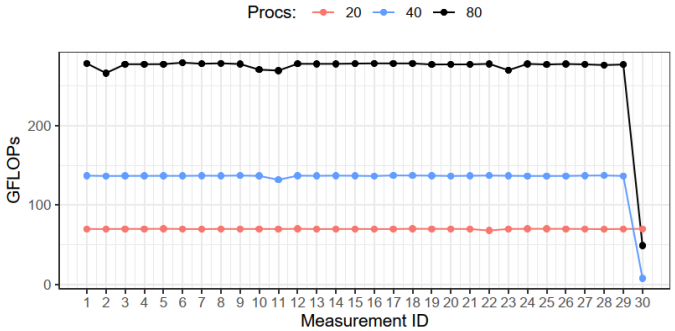
Automated Performance Characterization



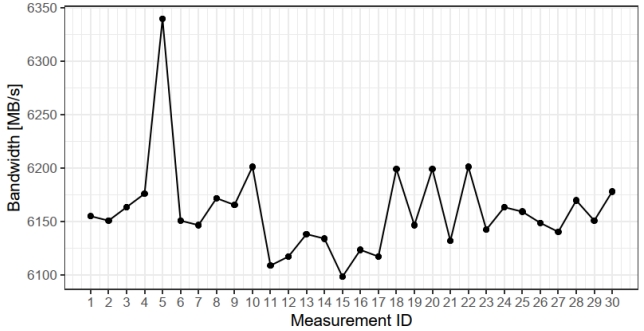
Roofline model for Himeno and HPCG results.



Heat map of the allocated nodes (overall benchmark runs).



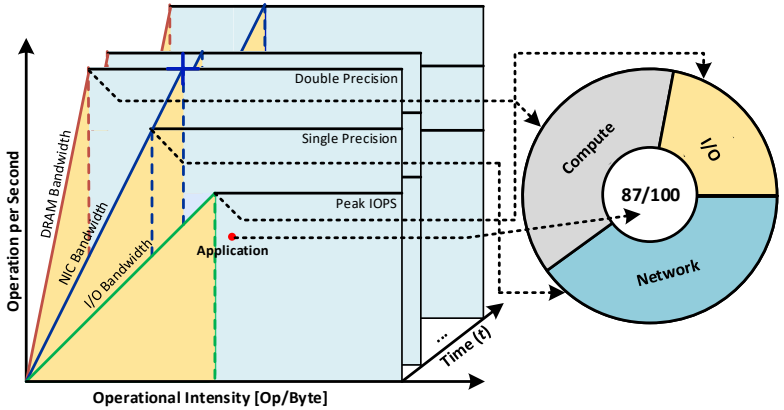
Himeno benchmark over 15 days / 2 measurements per day.



RDMA point-to-point performance over 15 days / 2 measurements per day.

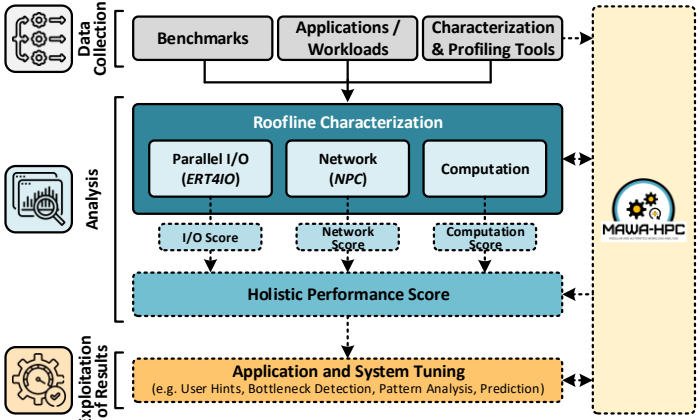
Integration: Bridging Layers and Silos
Multi-dimensional Performance Modeling

- *Goal:* provide a comprehensive view of application and system performance ⇨*emerging workloads*
- Multi-dimensional performance models, for example Roofline model, to account for multiple performance factors (e.g. network, compute power, and parallel I/O)
- Including time as an additional dimension, the Roofline model can provide insight into an application's performance over time, enabling the identification of performance anomalies



Zhu, Z., Bartelheimer, N. and Neuwirth, S., 2023. *MAWA-HPC: Modular and Automated Workload Analysis for HPC Systems*. ISC'23.

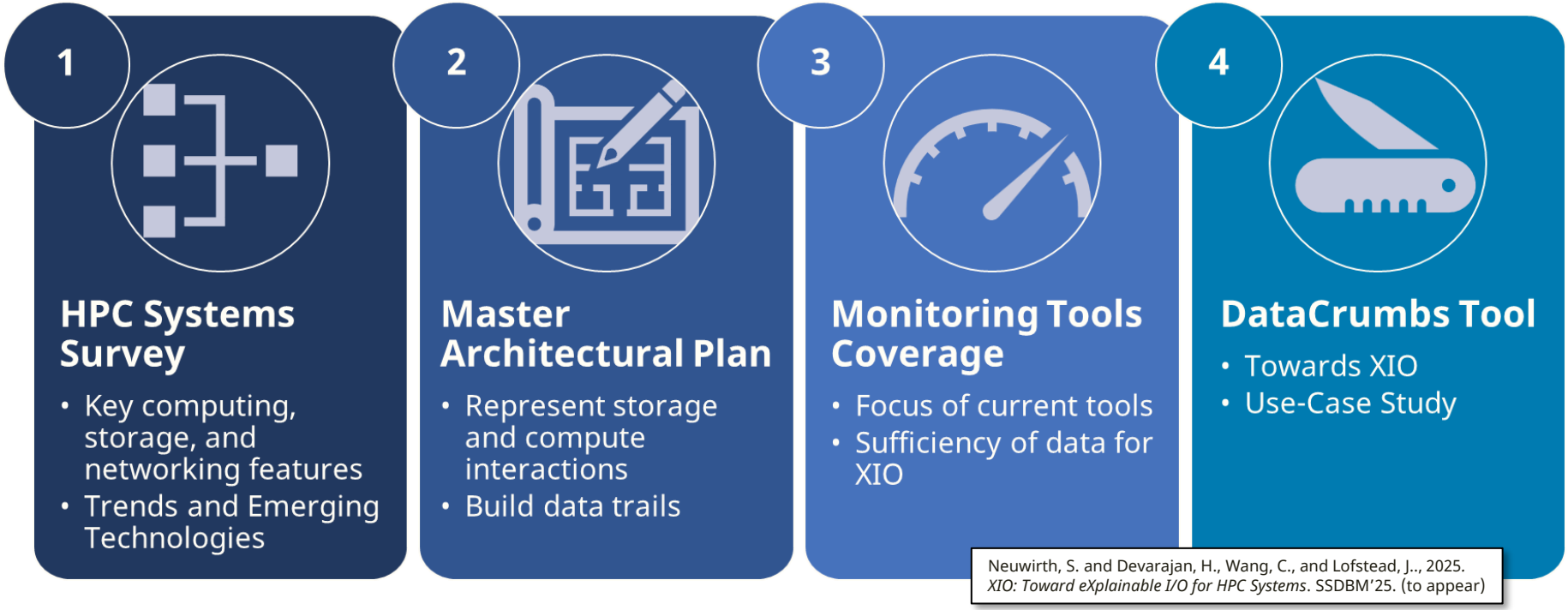
Bartelheimer, N., Zhu, Z., and Neuwirth, S., 2023. *Toward a Modular Workflow for Network Performance Characterization*. IPDPSW'23.



Meaning: Turning Observability into Insight

Meaning: Turning Observability into Insight

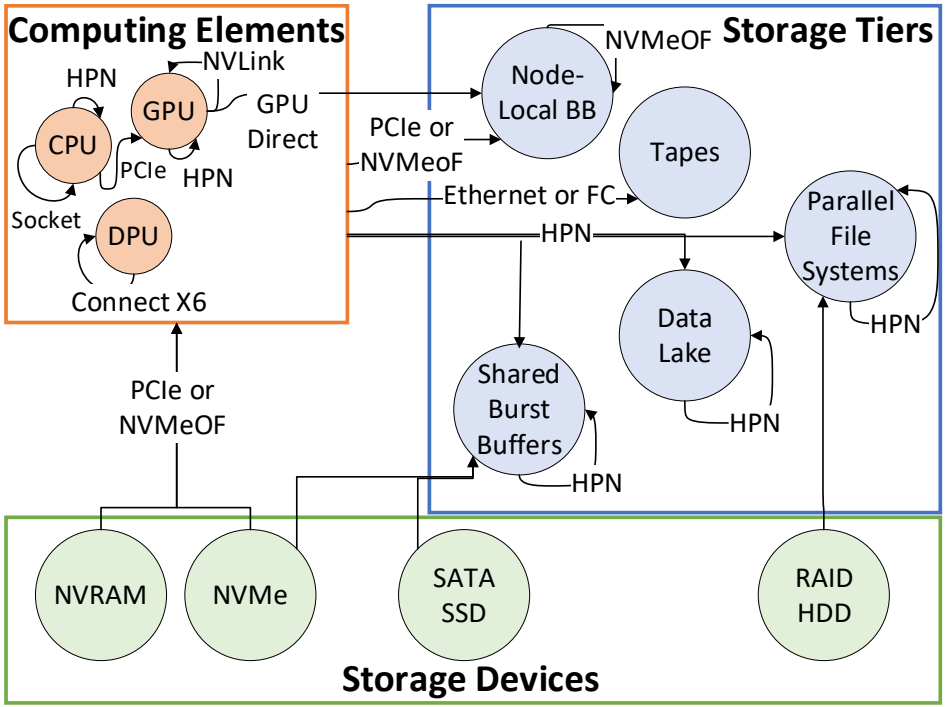
Toward eXplainable I/O



Neuwirth, S. and Devarajan, H., Wang, C., and Lofstead, J.., 2025. *XIO: Toward eXplainable I/O for HPC Systems*. SSDBM'25. (to appear)

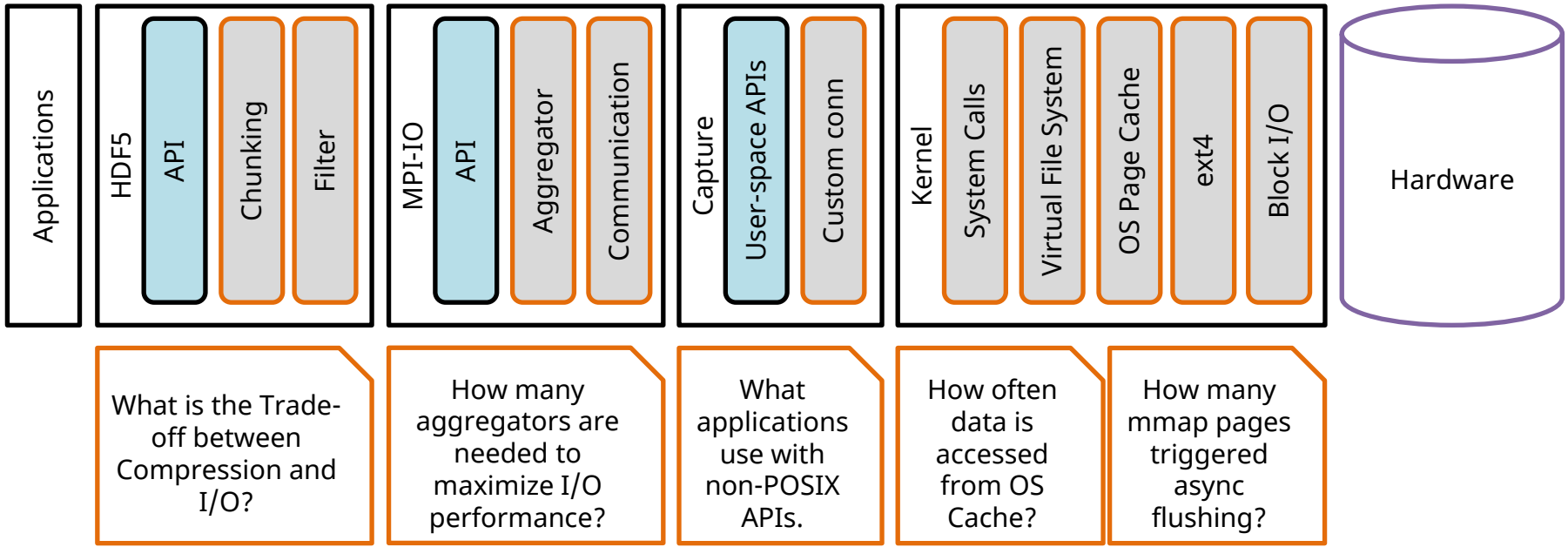
Meaning: Turning Observability into Insight
Master Architectural Plan (MAP)

- Derived from current TOP500 and IO500
- Primary objective of MAP is to provide a *standardized yet adaptable blueprint* to align software and monitoring tools across various HPC configurations
- *Graph-based representation* of components and interconnects for modeling data flows in HPC systems
- Emphasizes data flow, not just system layout => *connects observation points across the stack*



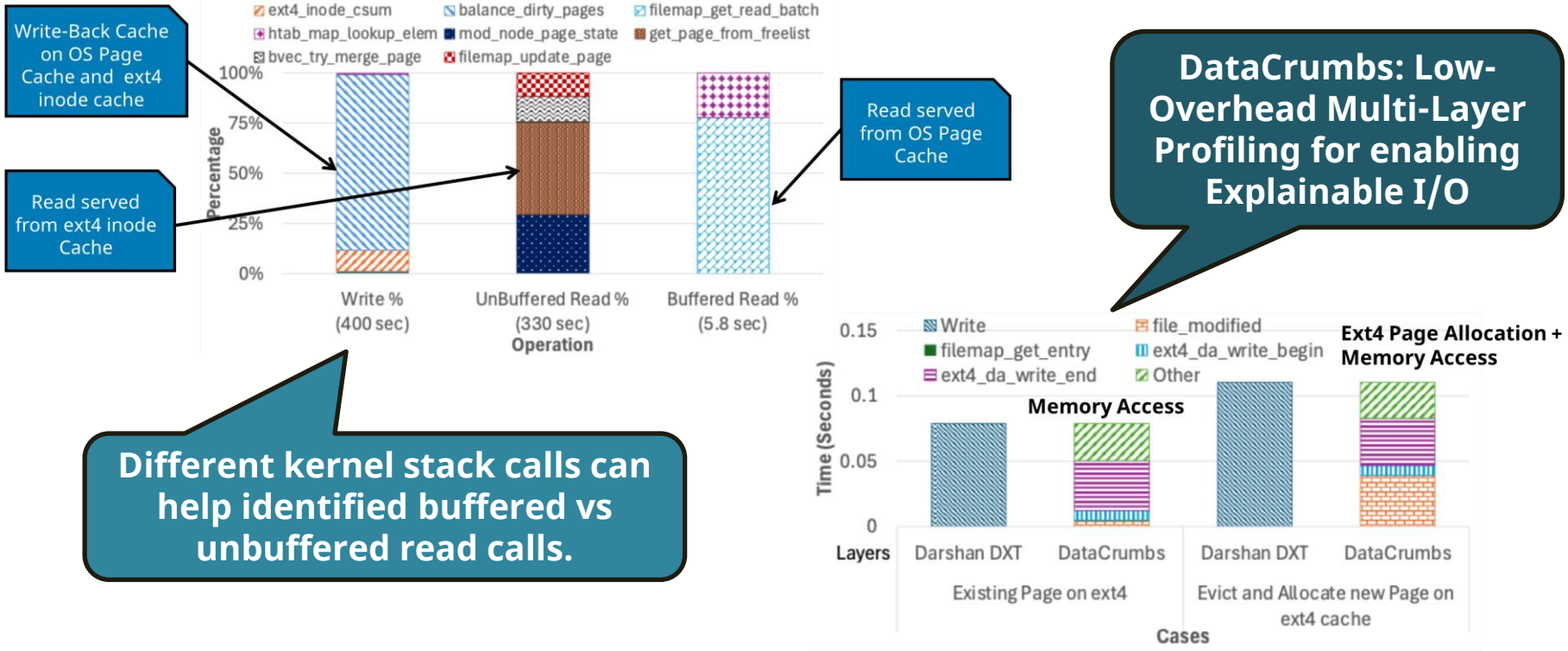
Meaning: Turning Observability into Insight

Observability vs. I/O Performance Variability



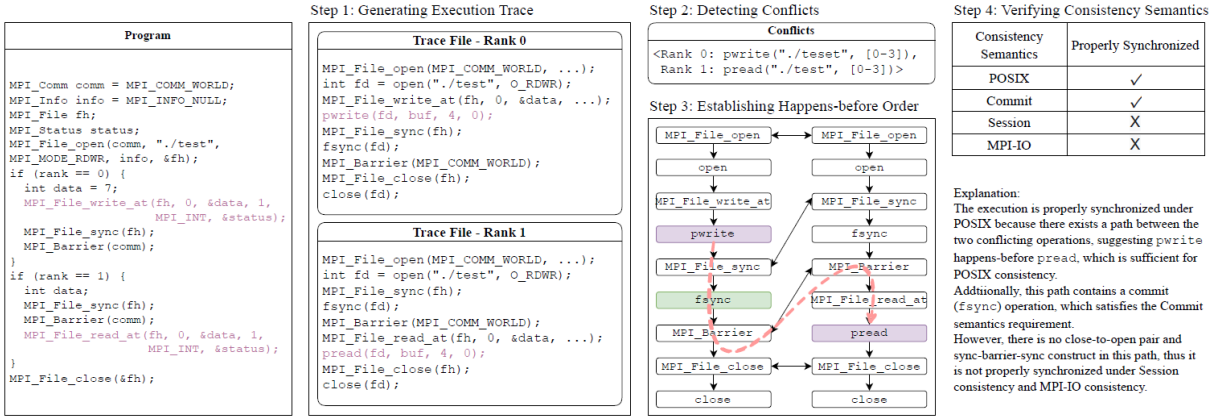
Meaning: Turning Observability into Insight

Toward better Observability with DataCrumbs



Meaning: Turning Observability into Insight

Verifying Performance Meaningfully, Beyond Metrics

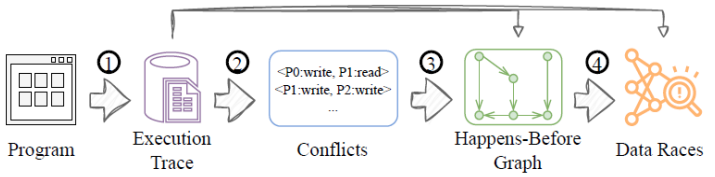


Goals:

- ***Ensure correctness*** on relaxed consistency systems
- ***Diagnose portability issues*** across backends
- ***Support future file systems*** beyond POSIX
- ***Enable developers*** to detect and fix semantic violations

What is VerifyIO? An open-source tool for trace-based verification of I/O consistency semantics in HPC applications.

Why does it matter? Emerging file systems and libraries relax consistency models (e.g., MPI-IO, Session), which can silently break application correctness.



Wang, C., Zhu, Z., Mohror, K., Neuwirth, S., and Snir, M., 2025. *VerifyIO: Verifying Adherence to Parallel I/O Consistency Semantics*. IPDPS'25.

Meaning: Turning Observability into Insight

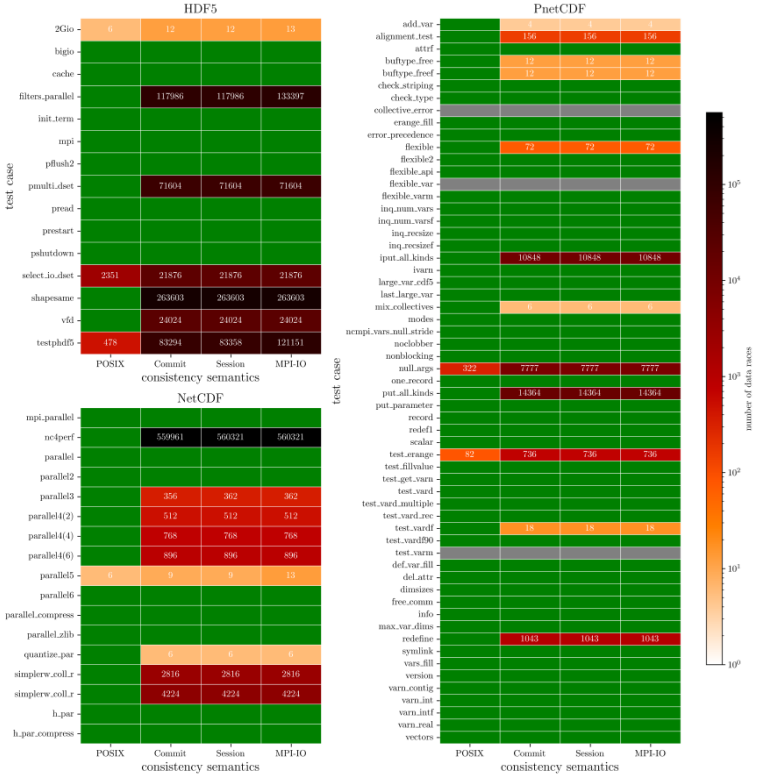
Verifying Adherence to Consistency Semantics – Results

- 91 built-in test programs from three production-level I/O libraries: HDF5, NetCDF, PnetCDF
- 4 consistency models: POSIX, Commit, MPI-IO, Session
- Do I/O libraries comply with the MPI standard?

I/O Library	HDF5	NetCDF	PnetCDF	Total
# Tests	15	17	59	91
Programs w. data races	7	9	12	28

Both test cases and library implementations have consistency bugs

Also identified a communication bug in PnetCDF



Meaning: Turning Observability into Insight

FlexBench – Sandbox for What-If Analysis

FlexBench is…

- A benchmark generator that reconstructs and manipulates I/O patterns from execution traces
- Built on top of Recorder+, leveraging Context-Free Grammars (CFGs) to compress and describe I/O behavior

Why do we need FlexBench?

- Traditional tracing tools like Darshan or Recorder capture detailed I/O but lack replay and what-if capabilities
- FlexBench enables users to replay, analyze, and tune the I/O behavior of applications, even without modifying code

Core Goals:

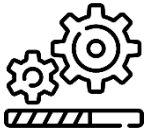
1. Use CFGs to precisely describe application I/O patterns
2. Reproduce original I/O performance from filtered traces
3. Expose optimization opportunities (e.g., parameter tuning)

Step 1: Recording the Full Trace

Step 2: Filtering for a Clean Trace

Step 3: Deriving the Benchmark

Step 4: Performance Optimization



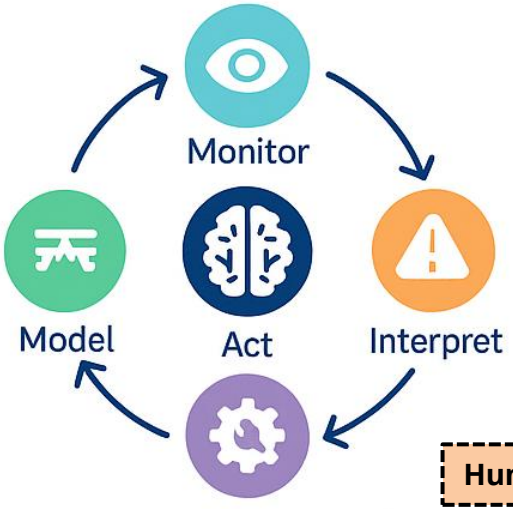
Vision and Closing

Vision and Closing
Toward Self-Aware HPC Systems

What is a Self-Aware HPC System?

- A system that can monitor itself, understand performance context, and adapt intelligently
- Observability + models + feedback = self-optimization

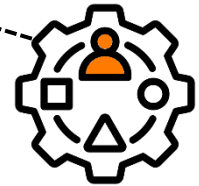
Core Capabilities:



Why it matters:

- Enables resilience: early detection of failure patterns
- Drives efficiency: energy-aware scheduling, performance tuning
- Builds trust: traceable, explainable decisions

Humans on top of the loop



A Community Vision:

- Requires shared efforts: open tools, standardized metadata, cross-layer collaboration

"We do not need smarter tools – we need smarter systems."

Thank you for your Attention!

Dr. Sarah M. Neuwirth
Professor of Computer Science
Johannes Gutenberg University Mainz
Email: neuwirth@uni-mainz.de
Website: https://www.hpca-group.de/
NHR South-West HPC Center: https://nhrsw.de/