


Holistic HPC Performance Engineering and Reproducible Benchmarking: Opportunities and Challenges

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

ZIH Colloquium Series @ TU Dresden, November 2023

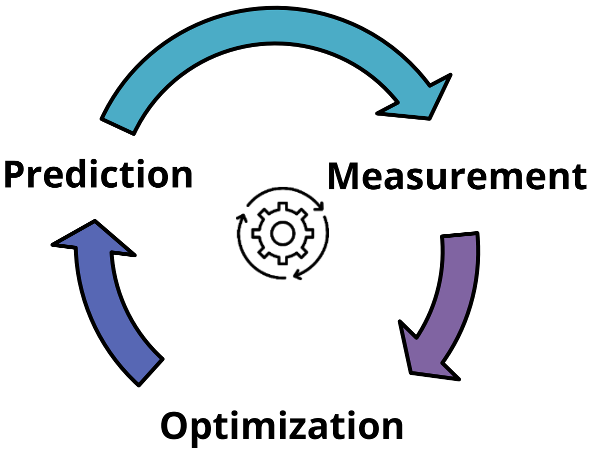
Introduction
Who am I?

- *Since 10/2023*: Professor and Group Lead, Johannes Gutenberg University
  - *"High Performance Computing and its Applications"* Group
  - *Website*: https://www.hpca-group.de/
  - *Research interests include*: parallel file and storage systems, modular supercomputing, performance engineering, high performance computing and networking, reproducible benchmarking, parallel I/O, and parallel programming models
- *Since 06/2021*: Visiting Research, Jülich Supercomputing Centre, Germany
- *03/2021 – 09/2023*: Deputy Group Leader, Goethe University Frankfurt, Germany
  - *"Modular Supercomputing and Quantum Computing"* Group
- *12/2018*: Ph.D. in Computer Science, Heidelberg University, Germany
- *2015 and 2016*: Internships at Oak Ridge National Laboraty, USA

Introduction
What is Performance Engineering?

Performance engineering encompasses the techniques applied during a systems development life cycle to ensure the non-functional requirements for performance (such as throughput, latency, or memory usage) will be met. Often, it is also referred to as ***systems performance engineering*** within systems engineering, and ***software performance engineering*** or application performance engineering within software engineering.

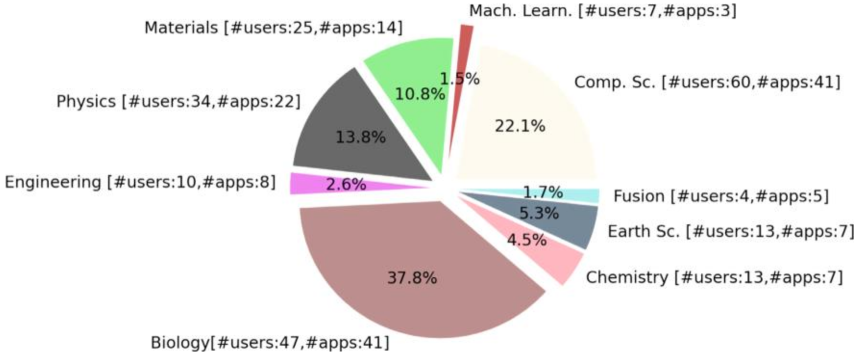
- Increase research output by ensuring the system can process transactions within the requisite time frame
- Eliminate system failure requiring scrapping and writing off the system development effort due to performance objective failure
- Eliminate avoidable system tuning efforts
- Reduce increased software maintenance costs due to performance problems in production
- Reduce additional operational overhead for handling system issues due to performance problems
- Identify future bottlenecks by simulation over prototype



Motivation

Motivation
Emerging HPC Workloads

- **<u>Traditional HPC:</u>**
  - Dominated by bulk-synchronous I/O phases
  - Simulation workloads
  - Checkpoint / Restart Files
  - Data Input / Data Output (bulky reads or writes)
- **<u>Emerging HPC workloads also encompass:</u>**
  - Artificial Intelligence
  - Data Analytics and Big Data
  - Deep Learning
  - Multi-step Workflows
  - In-situ analysis
- *<u>Vastly different I/O and performance characteristics</u>* (random small file accesses, non-sequential, metadata-intensive, and small-transaction reads and writes)



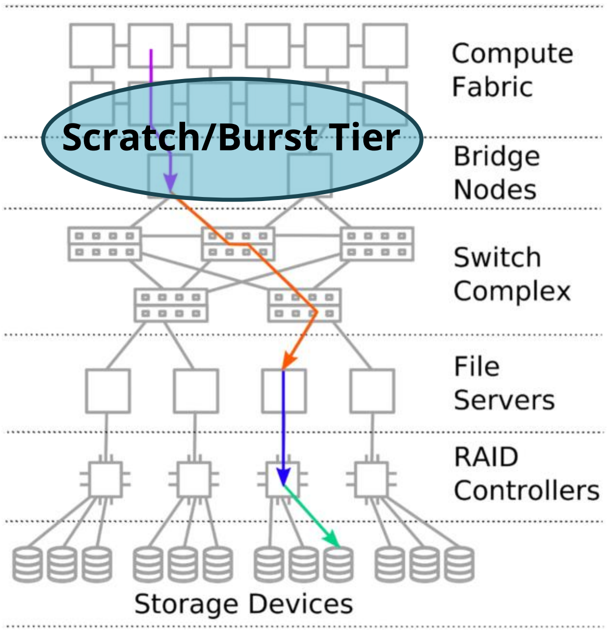
Classification of 23,389 ML jobs on Summit by science domains.

Karimi, A.M., Paul, A.K. and Wang, F., 2022. *I/O performance analysis of machine learning workloads on leadership scale supercomputer*. Performance Evaluation, 157, p.102318.

=> HPC I/O is more than just checkpointing and bulk-synchronous parallel I/O phases

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



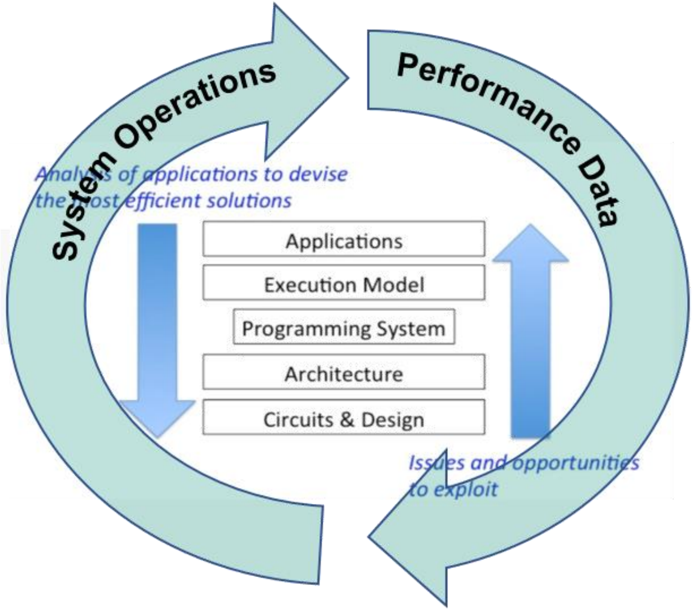
Carns, P., 2023. *HPC Storage: Adapting to Change*. Keynote at REX-IO'23 Workshop.

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
Holistic Monitoring and Operational Data Analytics

- Continuous and holistic *monitoring*, *archiving*, and *analysis* of operational and performance data open up interactivity with applications, system software, and hardware through
  - Automated feedback
  - Dynamic analysis of workloads and application demands, architecture and resource state
  - Actionable analytics and adaptive response
- Enable *efficient HPC operations*



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

Motivation

Holistic Performance Engineering – Challenges

- Heterogeneous Architectures
- Parallelism and Scalability
- Diversity of Workloads
- Dynamic Resource Allocation
- Software Stack Complexity
- Large-scale Data Movement
- Benchmark Suitability

?

Reproducibility

Comparability

Portability

Performance Engineering

– State of the Art –

Performance Engineering

Perspectives

User Interests and Concerns:

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

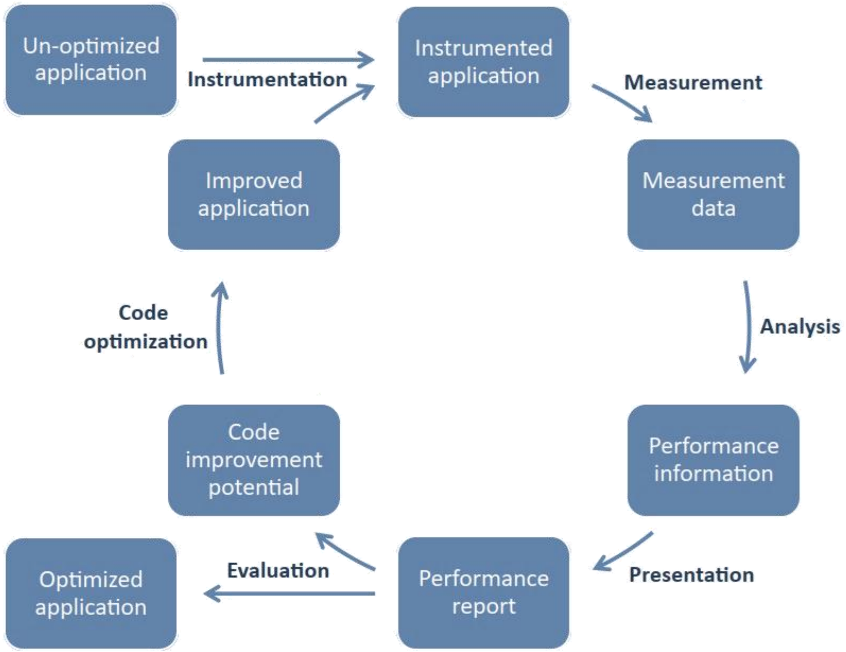
System Interests and Concerns:

- 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

Performance Engineering
Performance Optimization Cycle

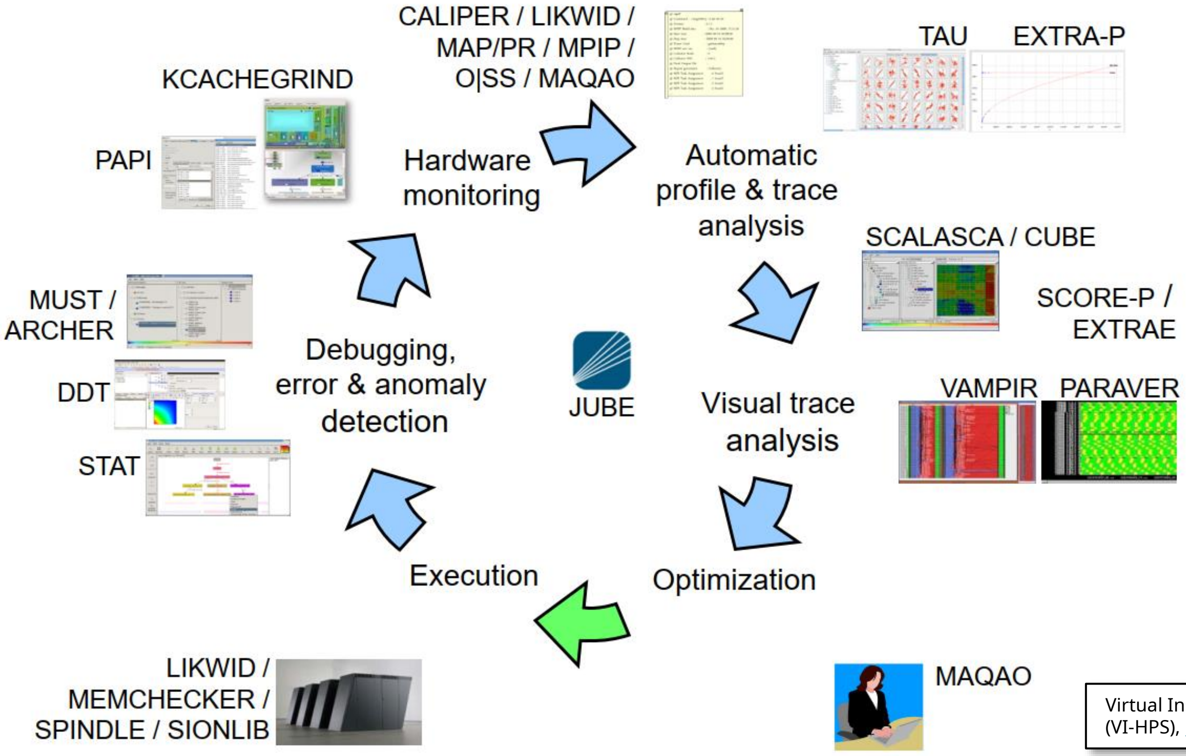
<u>Performance engineering typically is a cyclic process:</u>

- ***Instrumentation:*** common term for preparing the performance measurement
- ***Measurement:*** During measurement, raw performance data is collected
  - **Profiles:** hold aggregated data (e.g. total time spent in function foo())
  - **Traces:** consist of a sorted list of timed application events/samples (e.g. enter function foo() at 0.11 s)
- ***Analysis:*** Well defined performance metrics are derived from raw performance data during analysis
- ***Presentation:*** Presenting performance metrics graphically fosters human intuition
- ***Evaluation (and Code Optimization):*** Requires tools and lots of thinking



Performance Engineering Overview, https://doc.zih.tu-dresden.de/software/performance_engineering_overview/

Performance Engineering
VI-HPS Tools Overview



Virtual Institute – High Productivity Supercomputing (VI-HPS), https://www.vi-hps.org/tools/tools.html

Performance Engineering

I/O Performance Analysis – Overview

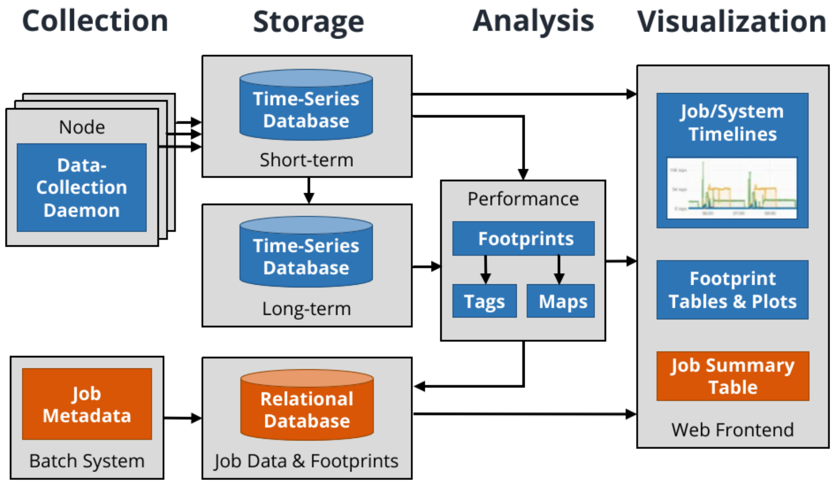
- I/O Performance depends on many different factors:

Application	Network	File System
• Number of processes • Request sizes • Access patterns • I/O operation • Data volume	• Message sizes • Network topology • Network paths • Network type	• Type of file system • Disk types • Stripe sizes • File hierarchy • Shared access

- Multiple tools to record, characterize and analysis I/O are available
  - Darshan, https://github.com/darshan-hpc/darshan
  - PIKA, https://gitlab.hrz.tu-chemnitz.de/pika
  - Vampir and Score-P (OTF2 – Open Trace Format 2)
    - https://vampir.eu/
    - https://score-p.org

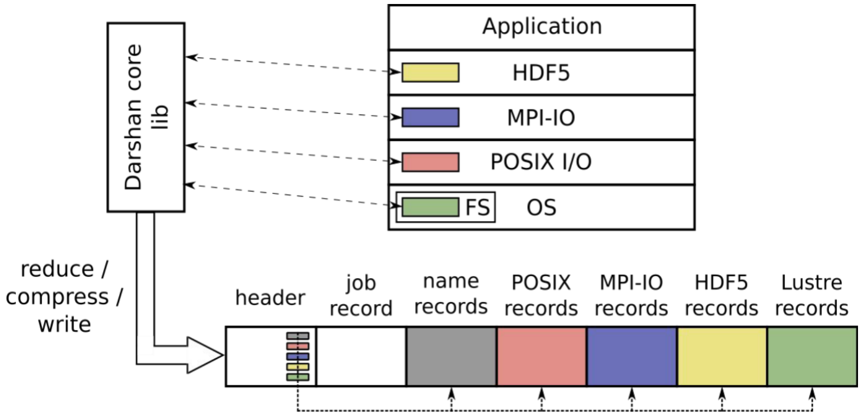
Performance Engineering
I/O Performance Analysis – Example Tools

PIKA: Center-Wide and Job-Aware Cluster Monitoring



Oeste, S., 2022. *Introduction on parallel I/O and distributed file systems*. NHR Lecture.

Darshan I/O-Characterization Tool

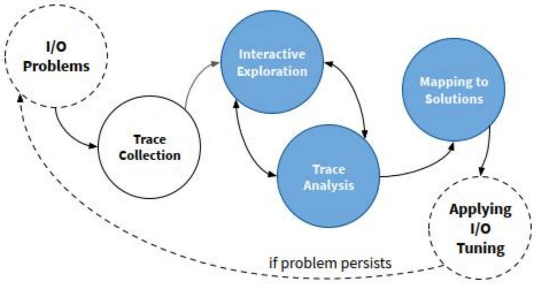


Snyder, S., 2022. *Darshan: Enabling Insights into HPC I/O Behavior*. ECP Community BoF Days.

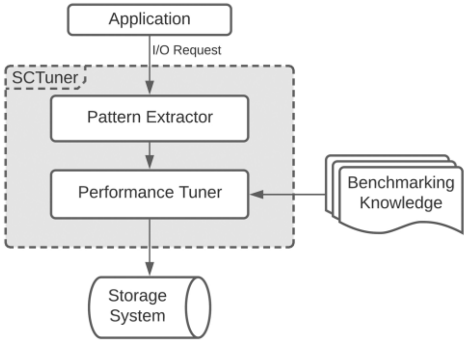
Performance Engineering
[Traditional] I/O Performance Optimization

<u>Performance optimization typically is a cyclic process:</u>

1) During the evaluation phase, the metrics and findings in a performance report are compared to the expected behavior/performance
2) Application is considered to behave sufficiently well or weaknesses have been identified which potentially can be improved
   - An application or its configuration is changed in the later case
3) After evaluating an application's performance, the cyclic engineering process is either completed or restarted from beginning



Iterative workflow to identify I/O performance issues based on the interactive visualization of DXT traces.



Key components of SCTuner.

Tang et al., 2021. *SCTuner: An Autotuner Addressing Dynamic I/O Needs on Supercomputer I/O Subsystems*. In 2021 IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW).

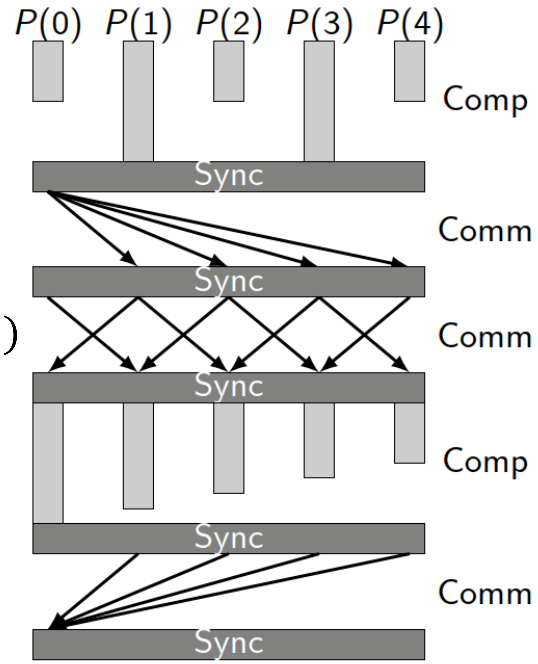
Bez et al., 2021. *I/O Bottleneck Detection and Tuning: Connecting the Dots using Interactive Log Analysis*. In 2021 IEEE/ACM Sixth International Parallel Data Systems Workshop (PDSW).

Performance Engineering
Performance Modeling

- Focus is on ***resource-based analytic loop performance models***
- Performance models generate knowledge about the *software-hardware interaction*
- "Mathematical description" often based on a simplified machine model that ignores most of the details of what is going on under the hood
  - Makes certain assumptions, which must be specified so that the range of applicability of the model is clear
- Main purpose is to produce a quantitative *estimate for an expected performance*
  - For example: resources consumed, contention for resources, and delays introduced by processing or physical limitations (such as speed, bandwidth of communications, access latency, etc.)
- A common feature of these performance models is that they are *discrete event systems*
  - View of the system is characterized by variables that take distinct values which change at distinct times or events in the system

Performance Engineering
Performance Modeling – Bulk Synchronous Parallel

- ***Busk Synchronous Parallel*** (BSP) machine…
  - provides model for designing parallel algorithms
  - operates by performing a sequence of *supersteps*
- Superstep consists of three consecutive phases:
  local computation phase, *global* communication phase, *barrier* synchronization
- h-relation to model the cost of comm superstep: $h = \max(h_s, h_r)$
  - h_s: max. number of data words sent by the processor
  - h_r: max. number of data words received by the processor
- Cost of an h-relation can be estimated by $T(h) = hg + I$
  - g: time per data word
  - I: global synchronization time

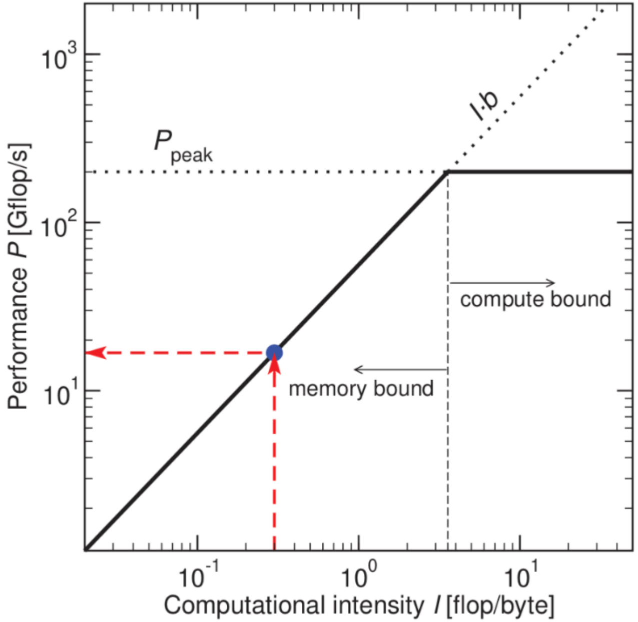


Bisseling, R.H., 2020. *Parallel Scientific Computation: A Structured Approach Using BSP*. Oxford University Press, USA.

Performance Engineering
Performance Modeling – Roofline Model Analysis

- Intuitive approach through simple bound and bottleneck analys
- 2D graph showing the *computational intensity* on the x-axis and the *attainable floating-point performance* on the y-axis
- X-axis: *Computational Intensity* $= \frac{\text{Floating−point operations}}{\text{Memory bytes transferred}}$
- Y-axis: $P = \min(P_{peak}, I \times b)$

  where P is the attainable perf., P_{peak} is the peak perf., b is the peak bandwidth, and I is the arithmetic intensity
- *Ridge point* analysis offers insight on the machine's overall performance, by providing the minimum arithmetic intensity required to be able to achieve peak performance, and by suggesting at a glance the amount of effort required by the programmer to achieve peak performance



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.

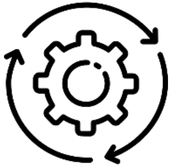
Performance Engineering

– Challenges and Vision –

Challenges and Vision
Reproducible Benchmarking

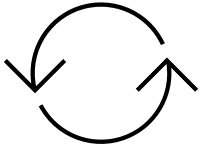
- ***Benchmarking:*** Process of comparing system performance using standardized tests and metrics.
- ***Reproducibility:*** Ability to obtain the same results with the same system and test conditions.
- ***Importance of Reproducibility:***
  - *Consistency:* Enables fair and accurate comparisons between systems
  - *Confidence:* Trust in benchmark results for decision-making
  - *Research Validity:* Essential for scientific studies and product evaluations

- ***Key Principles of Reproducible Benchmarking:***
  - *Documentation:* Record hardware and software configurations, test settings, and data
  - *Version Control:* Maintain consistent test suites and tools
  - *Automation:* Minimize human error by automating test execution
  - *Standardization:* Use industry-standard benchmarks and metrics
  - *Multiple Runs:* Conduct tests multiple times to verify results



Challenges and Vision

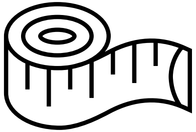
Reproducible Benchmarking – Design Discussion



Repeatability

Benchmark runs can be repeated with little con-figurational overhead

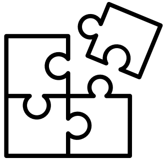
Schifrin, A., 2023. *Automated Performance Characterization of HPC Systems.* Bachelor thesis, Goethe University Frankfurt.



Comparability

A simplified structure amplifies comparability of results between runs or even machines

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



Modularity

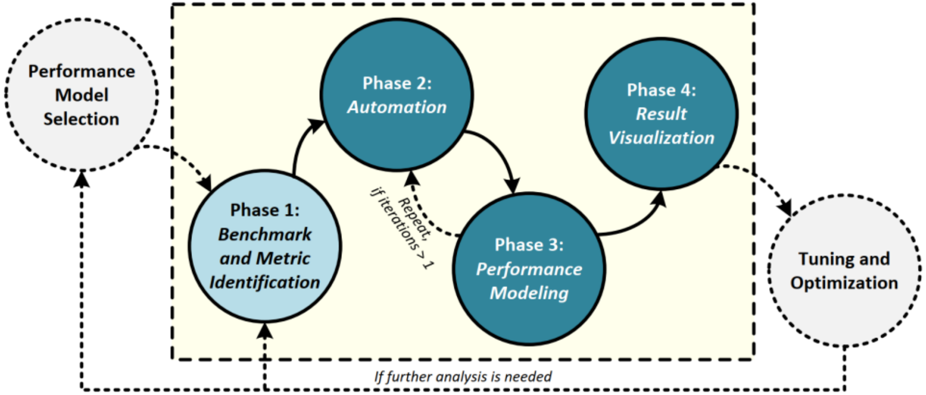
Enhanced modularity increases possiblities for further extensions

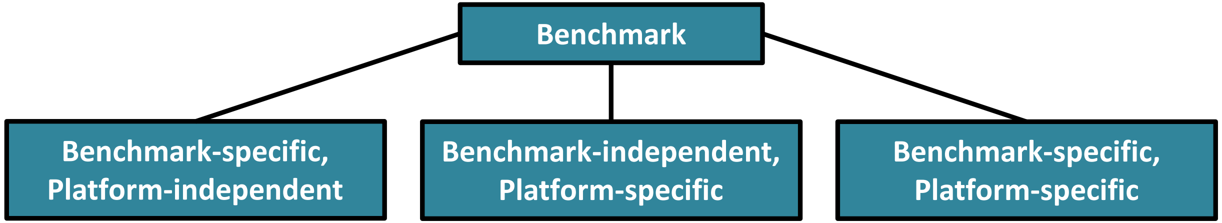
Bartelheimer, N., Zhu, Z., and Neuwirth, S., 2023. *Automated Network Performance Characterization for HPC Systems*. IJNC Special Issue on APDCM 2023 (accepted for publication).

Challenges and Vision

Reproducible Benchmarking – Workflow Design

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






```
config
├─ platform
│  ├─ fuchs
│  │  ├─ platform.xml
│  │  └─ platform-likwid-specs.xml
│  └─ cesari
│     ├─ platform.xml
│     └─ ...
├─ likwid-benchmark.xml
├─ likwid-specs.xml
├─ qperf-benchmark.xml
├─ qperf-specs.xml
├─ ...
├─ likwid.run.in
└─ ...
```

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

Challenges and Vision

Reproducible Benchmarking – Example Configuration

Benchmark-independent, Platform-specific

`platform.xml`

```xml
<parameterset name="systemParameter">
    <parameter name="nodes" type="int">2</parameter>
    <parameter name="taskspernode" type="int">1</parameter>
    <parameter name="threadspertask" type="int">1</parameter>
    <parameter name="tasks" mode="python" type="int">$nodes * $taskspernode</parameter>
    <parameter name="timelimit">00:30:00</parameter>
</parameterset>
```

```xml
<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`

```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>
```

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

```xml
<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

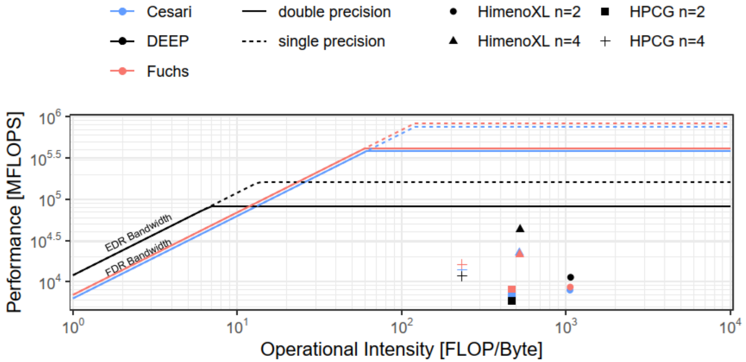
`platform-likwid-specs.xml`

```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>
```

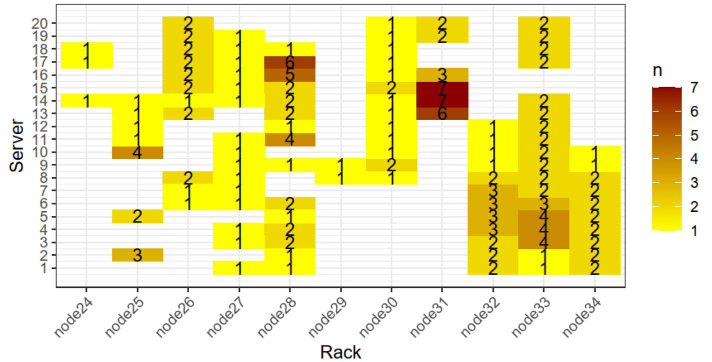
```xml
<parameterset name="suffix_params">
    <!--
        CHANGEME: this parameter can be changed and represents the parameter part after the <size>
        if not empty, the suffix must start with the ":" (colon) character
    -->
    <parameter name="suffix"></parameter>
</parameterset>
```

Challenges and Vision

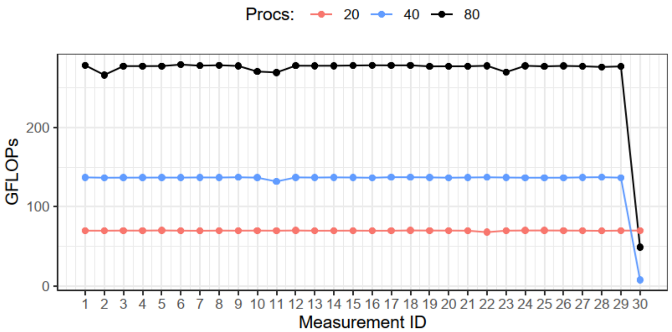
Reproducible Benchmarking – Example Results



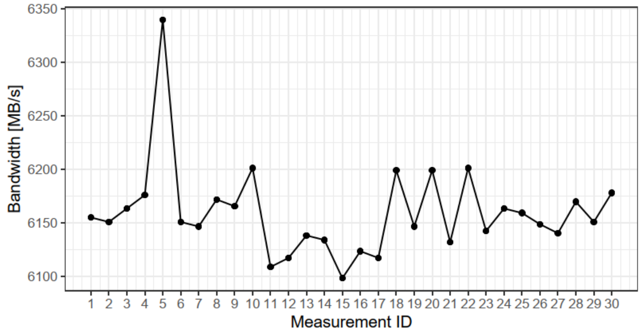
Roofline model with Himeno and HPCG results.



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



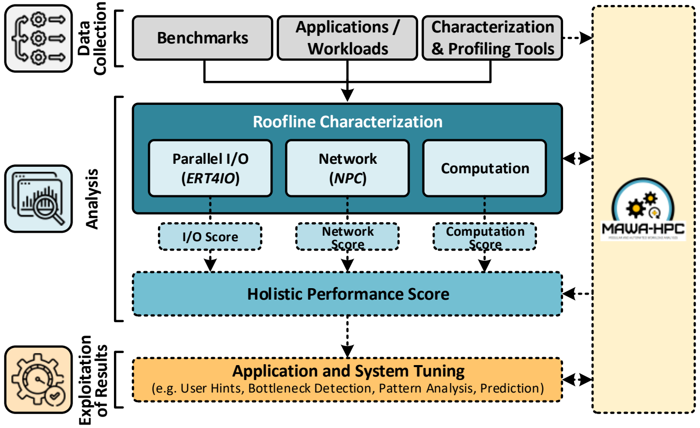
Himeno benchmark over 15 days / 2 measurements per day.



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

Challenges and Vision
Multi-dimensional Performance Modeling

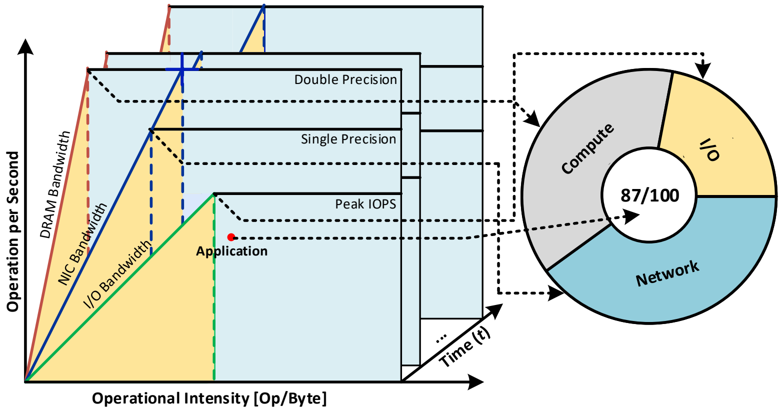
- *Goal:* provide a comprehensive view of application and system performance ⇨ *emerging workloads*
- Multi-dimensional performance models, for example Roofline model, to account for multiple performance factors (e.g. network, compute power, and parallel I/O)
- Including time as an additional dimension,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

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

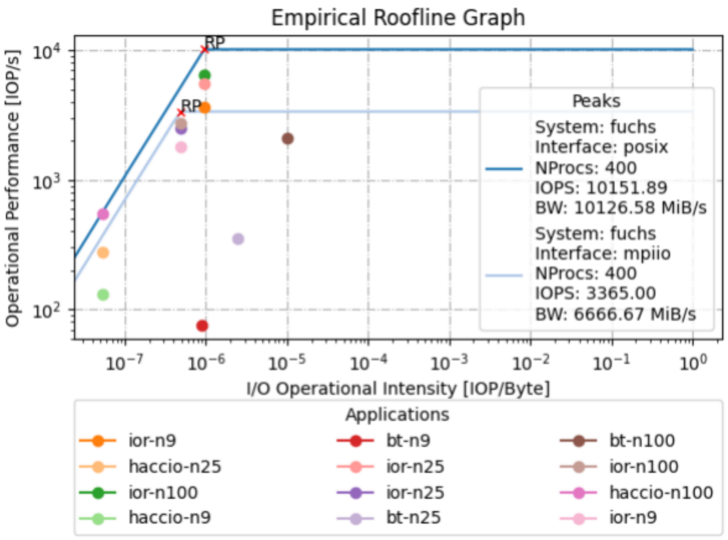
Bartelheimer, N., Zhu, Z., and Neuwirth, S., 2023. *Toward a Modular Workflow for Network Performance Characterization*. IPDPSW'23.



Challenges and Vision

I/O Roofline Model and Scoring Approach

- ***I/O Roofline Model*** is based on IOPS and the corresponding bandwidth
  - *IOPS*: number of reads and writes that a storage system can perform per second
  - *Bandwidth*: total amount of data that can be 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 Operational Performance
- ***I/O Score***: I/O ridge point analysis at the system level
  - *Vector-based score (simplest concept):* ridge point is represented as a vector
  - *I/O bandwidth-based score*: product of peak IOPS and inverse of I/O intensity

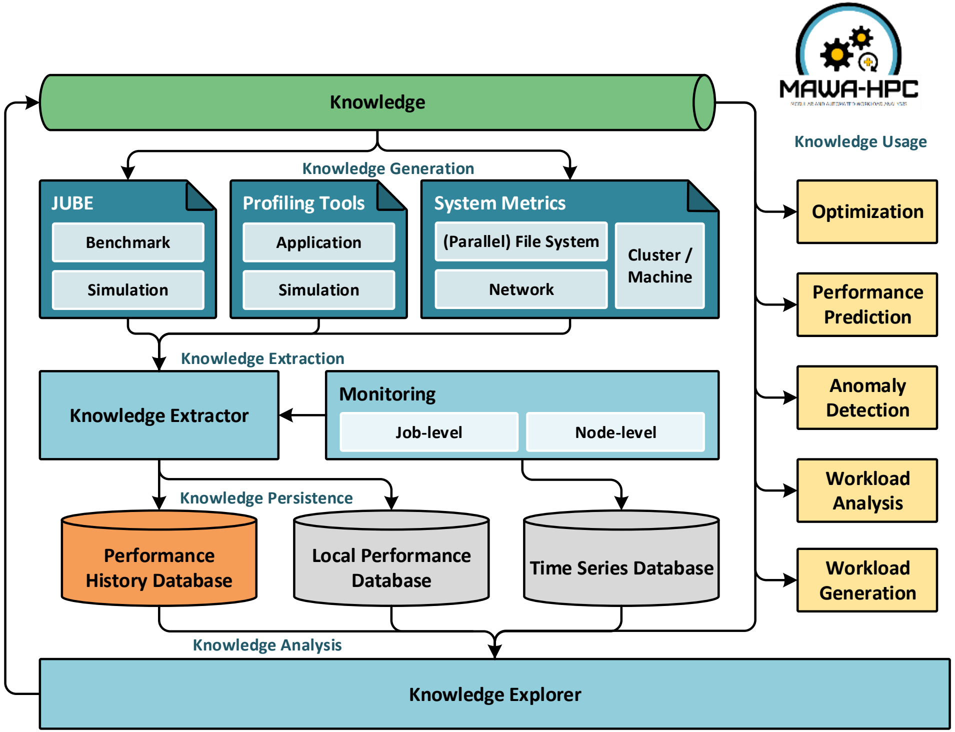


Zhu, Z., and Neuwirth, S., 2023. *Characterization of Large-Scale HPC Workloads With Non-Naïve I/O Roofline Modeling and Scoring.* ICPADS'23 (accepted).

Challenges and Vision

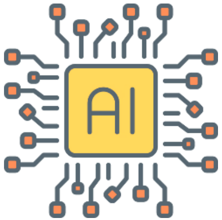
Holistic Performance Engineering and Analysis

- *Idea:* design and implement standardized and tool-independent approach for HPC workload and application analysis
- Support and integration of various community tools, increasing the compatibility and coverage of new use cases
- Intuitive performance modeling and visualization so that users without prior knowledge can understand the results
- *Goal:* establish *performance history database* to categorize systems, workload behaviors, and characteristic patterns for different science domains and applications



Challenges and Vision
Artificial Intelligence and Large Language Models

- ***Large language models*** …
  - are a form of generative artificial intelligence (AI) that focus on generating human-like text in ways that make contextual sense
  - consist of an artificial neural network with many parameters (tens of millions to billions), trained on large quantities of unlabeled text containing up to trillions of tokens, using self-supervised learning or semi-supervised learning achieved by parallel computing
- Level of proficiency in various tasks, as well as the breadth of tasks they can handle, rely less on the model's design and more on the *size of the training corpus*, the *quantity of parameters*, and the computational power achieved by parallel computing
- Questions from a panel discussion at HPDC’23:
  - HPC and AI – a powerful combination?
  - How far should we go with digital twins?
  - Why not just use ChatGPT for automatic system and workload performance tuning?

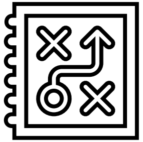


Conclusion

Conclusion

Challenges and Opportunities – What's next?

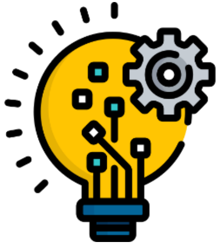
- Data often collected haphazardly, with gaps, without configuration data, logs, etc.
  - How can we make sure that recorded data (e.g., Darshan logs) is complete?
  - Want to collect data systematically, without added effort, leveraging the collection processes and practices currently in state of the practice
- Work toward intelligent system and applications co-design
  - What do we need to collect in a performance history database such that we can use AI frameworks to run autonomously (with human-on-the-loop)?
- Performance history database will enable:
  - Learn cost models
  - Overall system and application / workflow optimization



- Challenges of reproducibility requirements:
  - Maintaining a consistent testing environment can be challenging (i.e., background processes, system load, and external factors)
  - Selecting appropriate benchmarks and metrics that are accepted by the community is crucial

Conclusion
Are we Reinventing the Wheel?

- Significant trends emerging in HPC:
  - *Architectural complexity*: heterogeneity, modularity, power management, virtualization
  - Raise of *artificial intelligence* and large language models
  - *Application complexity*: traditional scale-up and emerging scale-out workloads
  - *Sustainability* of large-scale HPC systems
- Performance engineering is critical to bridging those gaps
  - Measurement, Prediction, and Optimization
  - Feedback to architects and system software designers
- Holistic end-to-end performance engineering is essential for improved user experience and automatic system / workload optimization
  - Interdisciplinary collaborations with researchers from human-computer interaction and data visualization needed
  - Community database for categorization of systems and workloads needed



Thank you for your Attention! Questions?