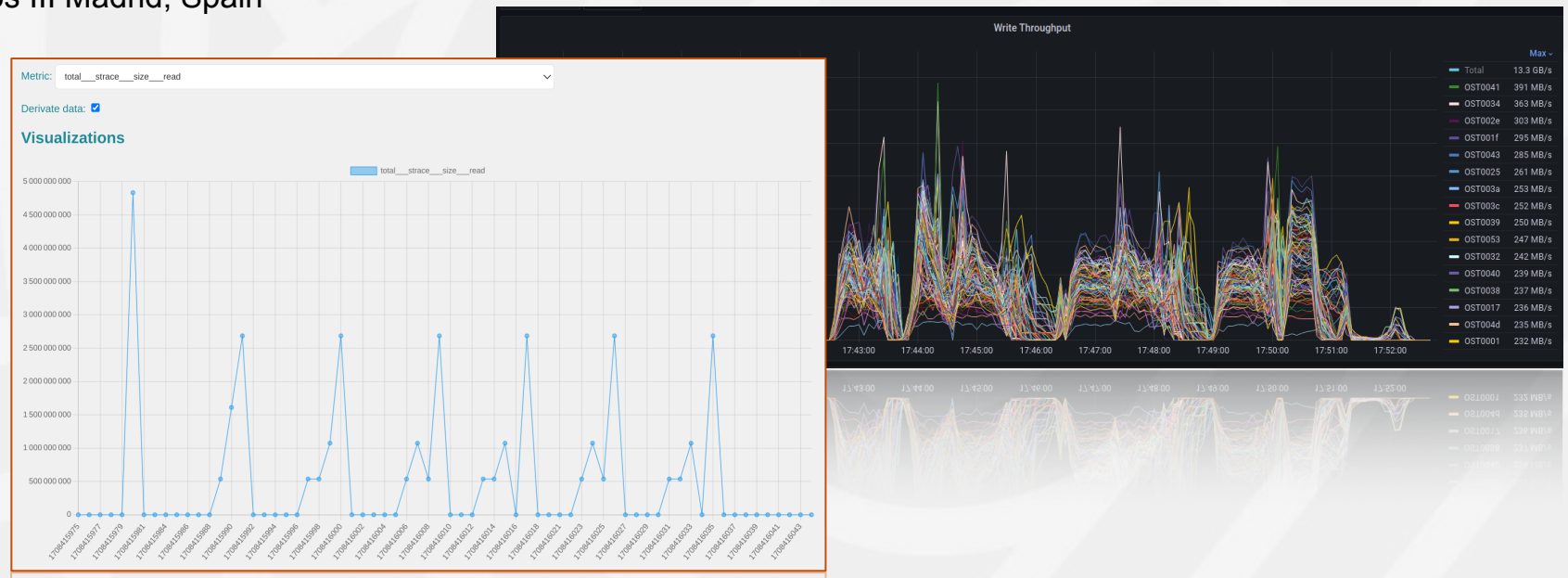
Introducing the Metric Proxy for Holistic I/O Measurements

Jean-Baptiste Besnard[1], **Ahmad Tarraf**[2], Alberto Cascajo[3] and Sameer Shende[1]

[1]ParaTools SAS, France
[2]Technical University of Darmstadt, Germany
[3]University Carlos III Madrid, Spain



TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

ADMIRE
malleable data solutions for HPC

ADAPTIVE MULTI-TIER INTELLIGENT DATA MANAGER FOR EXASCALE

EuroHPC Joint Undertaking

EXTRA NOISE

HPC I/O in the Data Center Workshop

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

- I/O are more and more complex to track in HPC systems as associated performance metrics are spreading the whole HPC stack
- There is a need to understand and predict potential I/O bottlenecks to further optimize the I/O subsystem
- We consider malleable payloads, able to collaborate with the storage back-end to further optimize I/O
- In the **ADMIRE** project, which serves as the framework for this work, we aim at defining a **holistic I/O optimization framework** based on monitoring and fine-tuning of the I/O stack, including the use of ad-hoc file systems

- Development of a versatile monitoring tool (the Metric Proxy)
- Use of its output data (among others) to generate online performance predictions (i.e., models) from profiles and traces

→ **Continuous Modelling**

Motivation (cont')

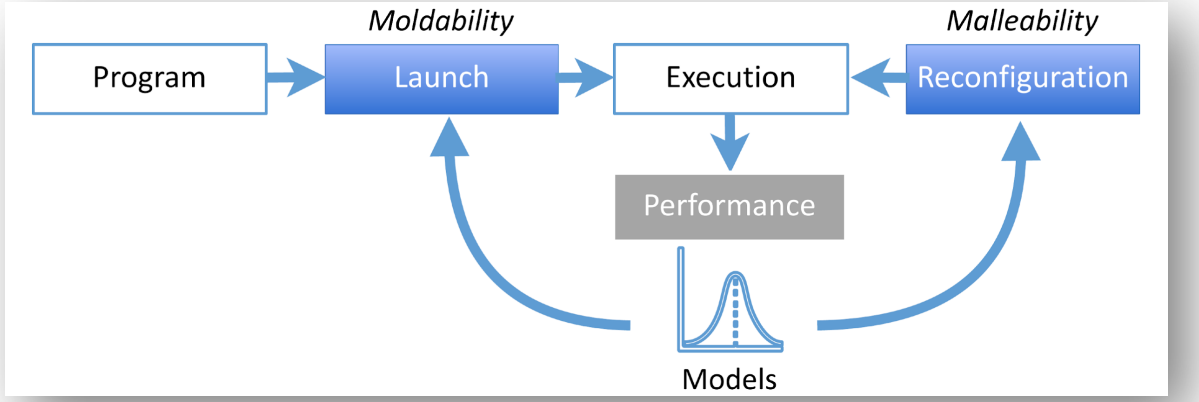
TECHNISCHE UNIVERSITÄT DARMSTADT

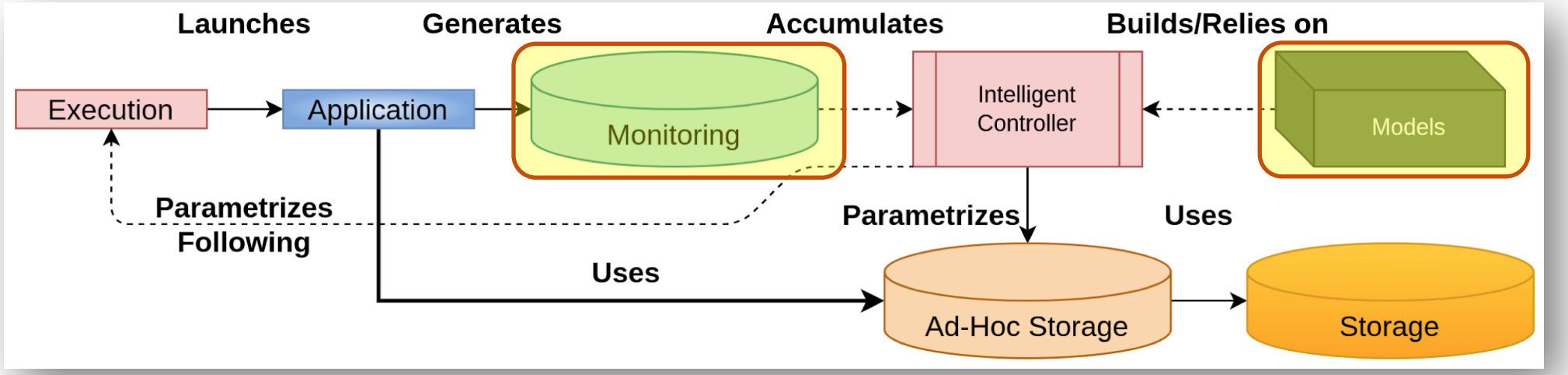
Parallel Programming

ADMIRE

malleable data solutions for HPC

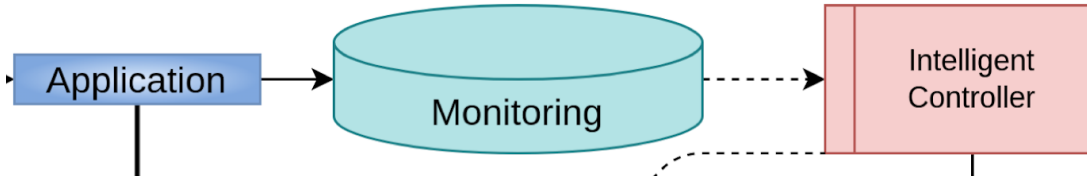
ADAPTIVE MULTI-TIER INTELLIGENT DATA MANAGER FOR EXASCALE



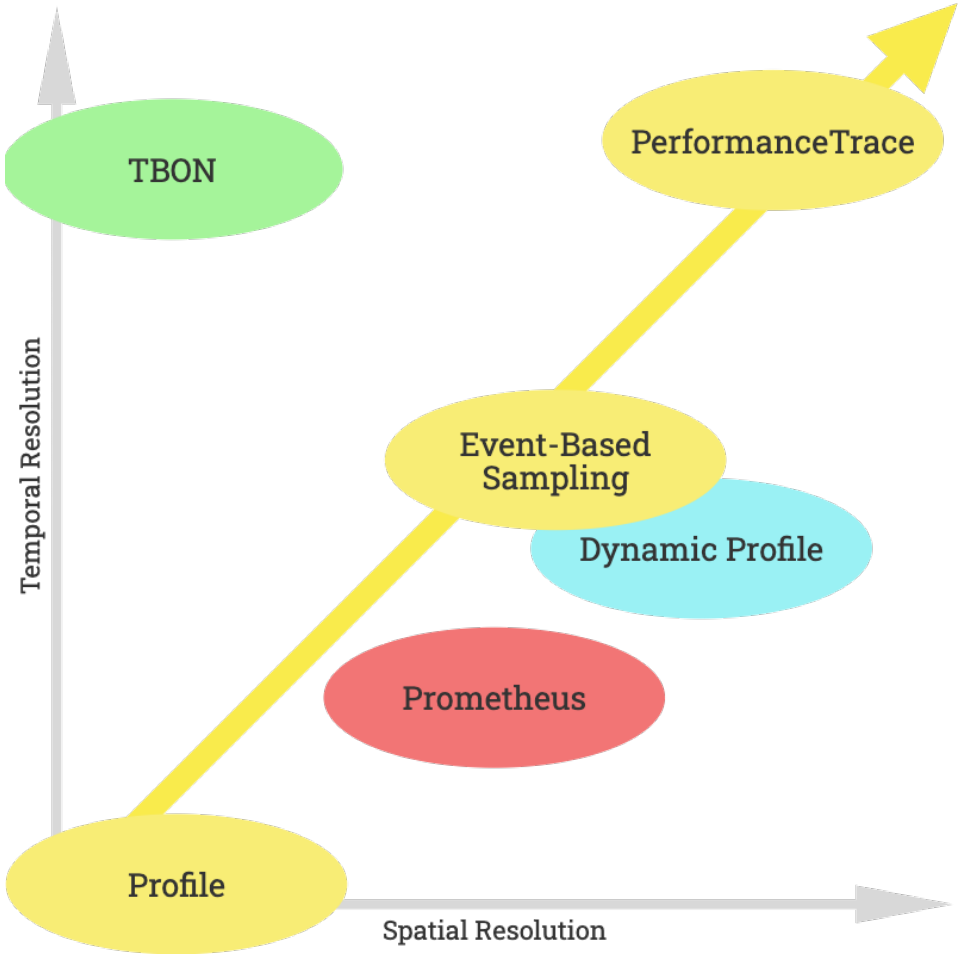


ADMIRE's

Monitoring Infrastructure
The Metric Proxy



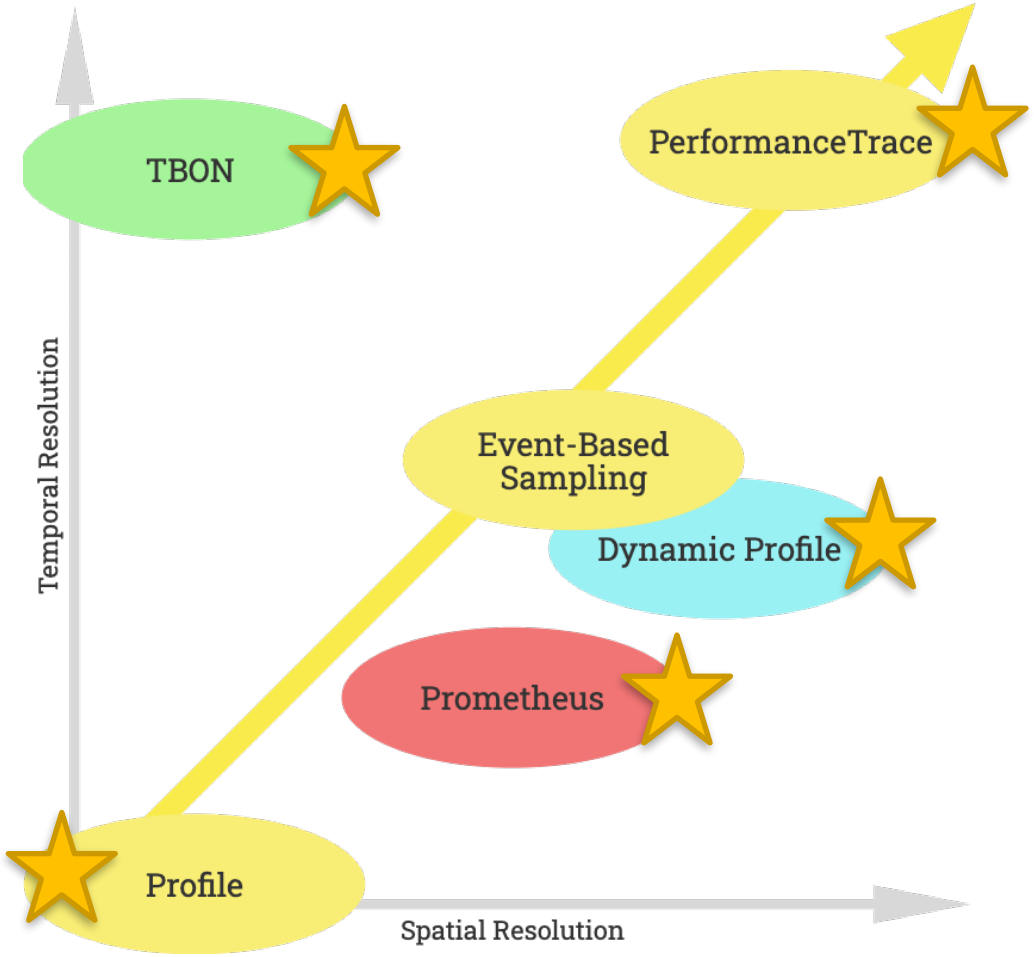
Monitoring Granularities



Performance measurements are always a compromise between verbosity and measurement/storage overhead.

Inspired by the "Cube Model"

Using the Metric Proxy for Flexible Measurements



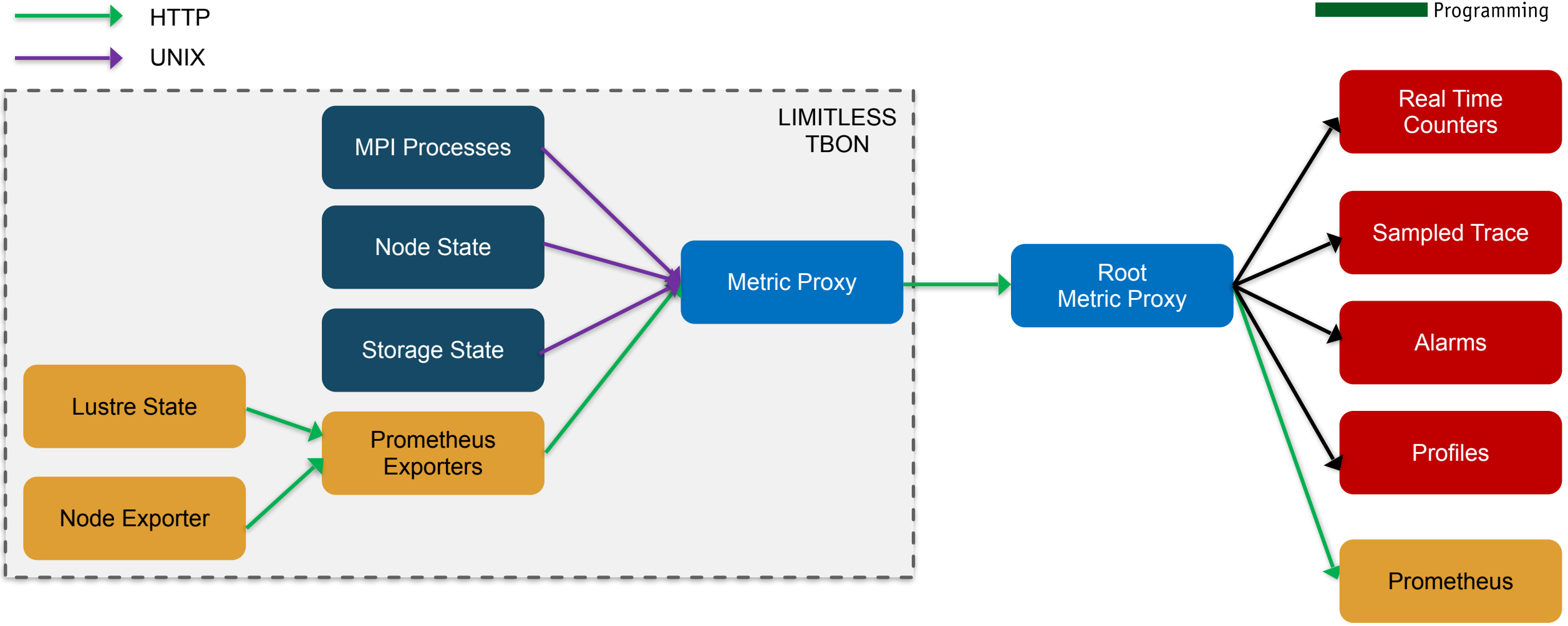
We gathered in a single tool:

- **TBON** for real-time reduction of performance data using LIMITLESS
- **Resampled performance traces** for temporal series
- **Profiles** to describe each run
- **Prometheus** storage for historization
- Real-time summative **profiles** (a.k.a snapshots) for current state

Metric Proxy's Architecture

TECHNISCHE UNIVERSITÄT DARMSTADT

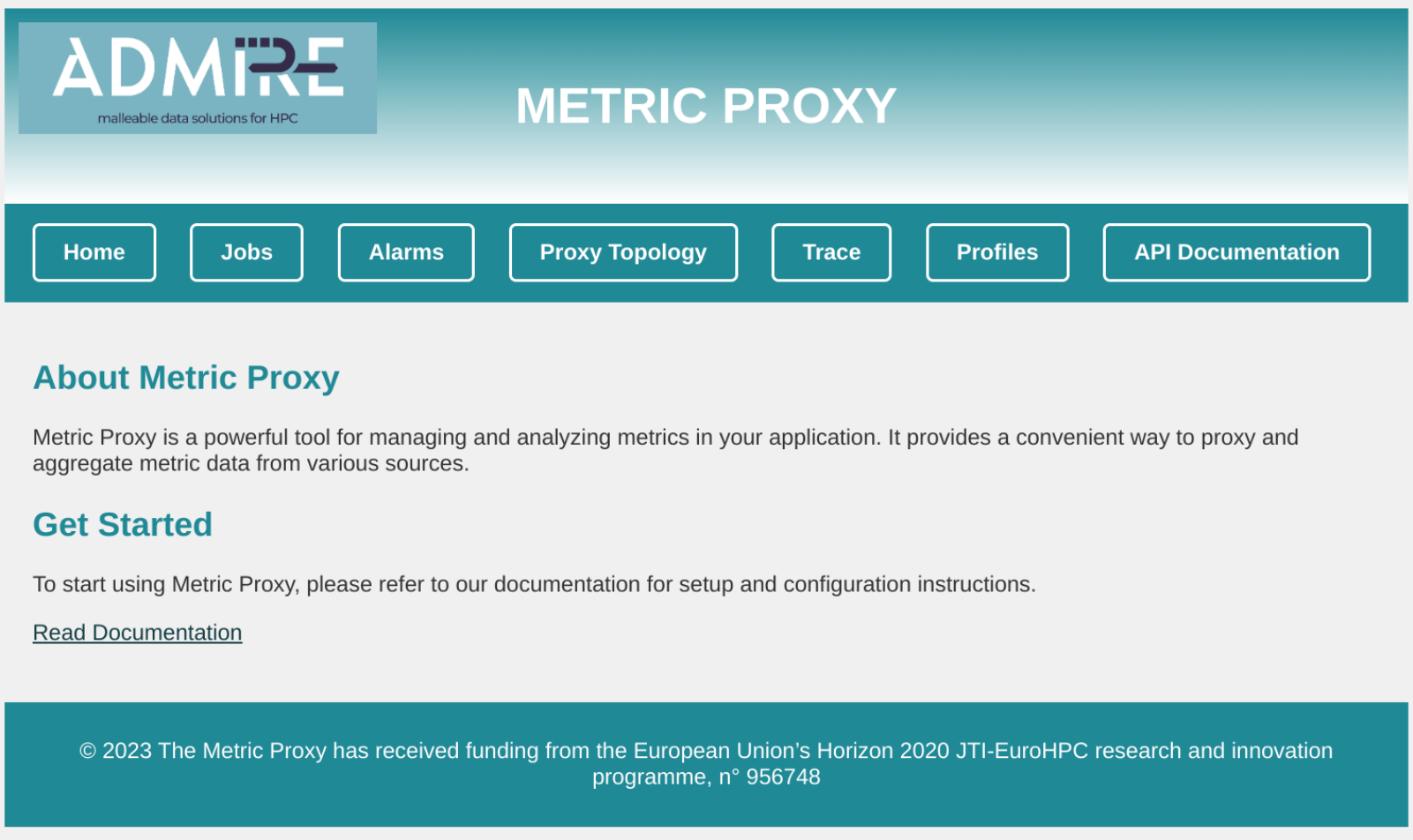
Parallel Programming



Embedded Web Interface

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

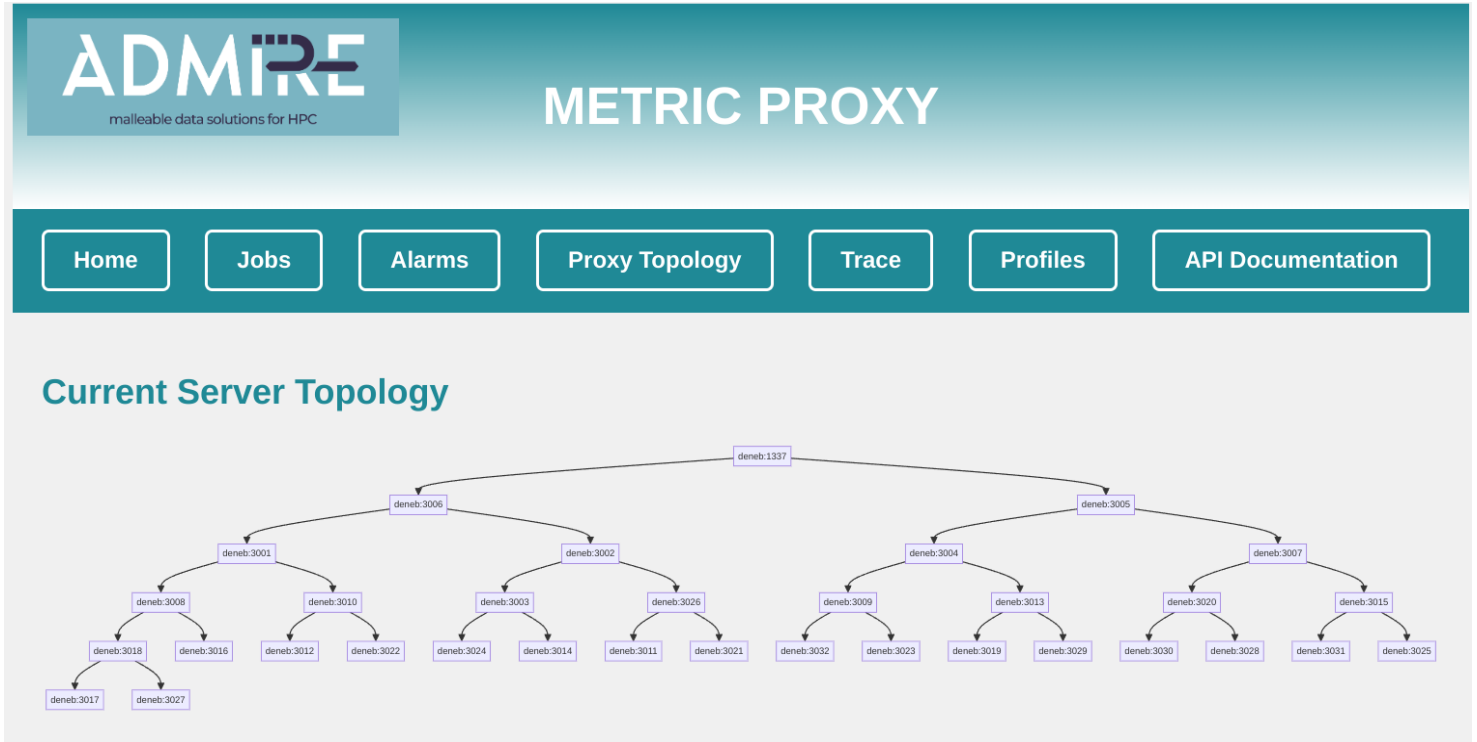


Each metric proxy on each node provides an HTTP endpoint on port 1337 by default.

Monitoring the TBON and Scrapes

The proxy reduction tree is built automatically by «pivoting» the nodes on a root server which then returns the address of one of the proxy.

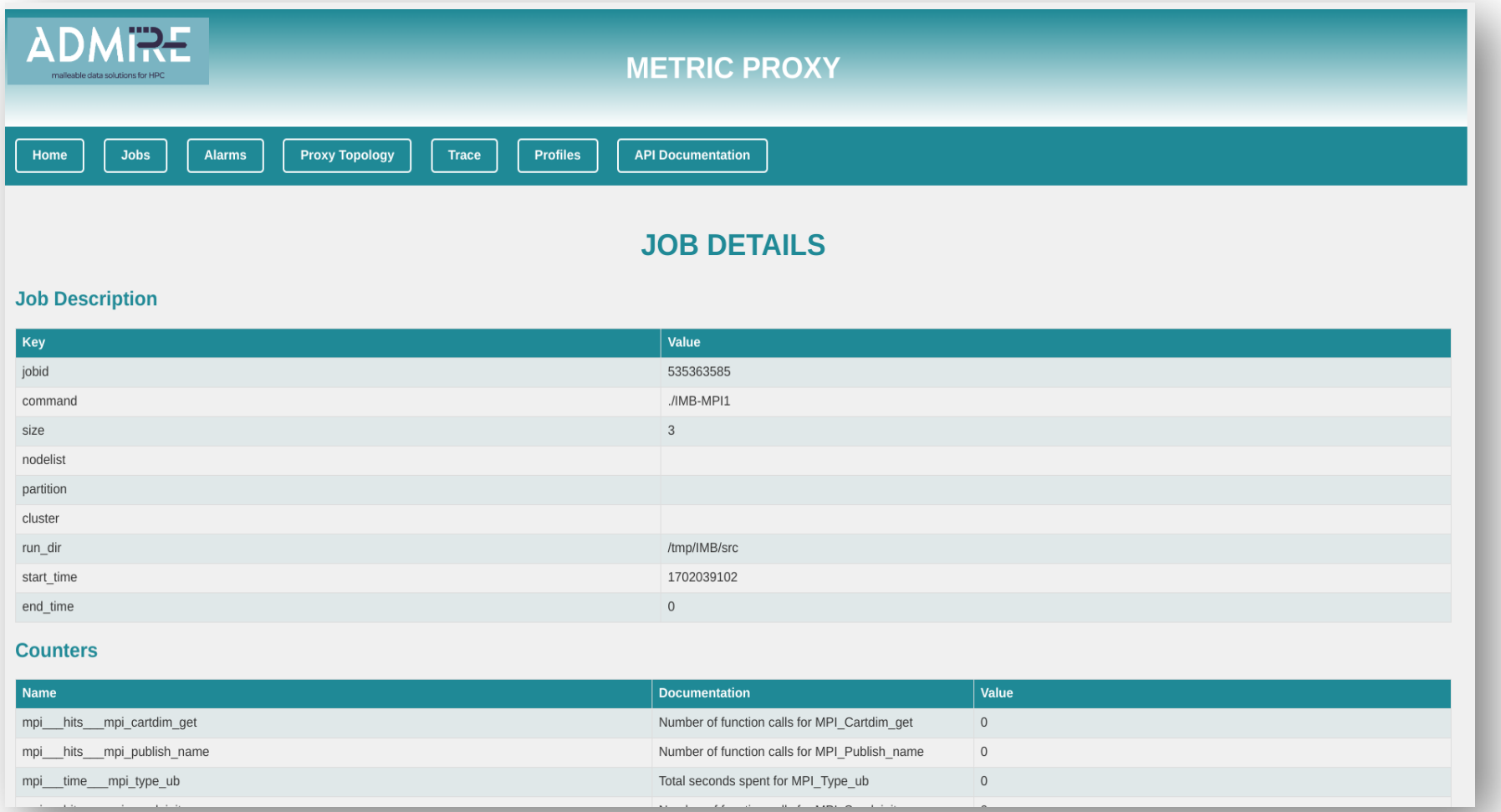
Here an example with 32 nodes, seen from the root.



Sample Profile Snapshot

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming



Sample Profile Snapshot (cont’)

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

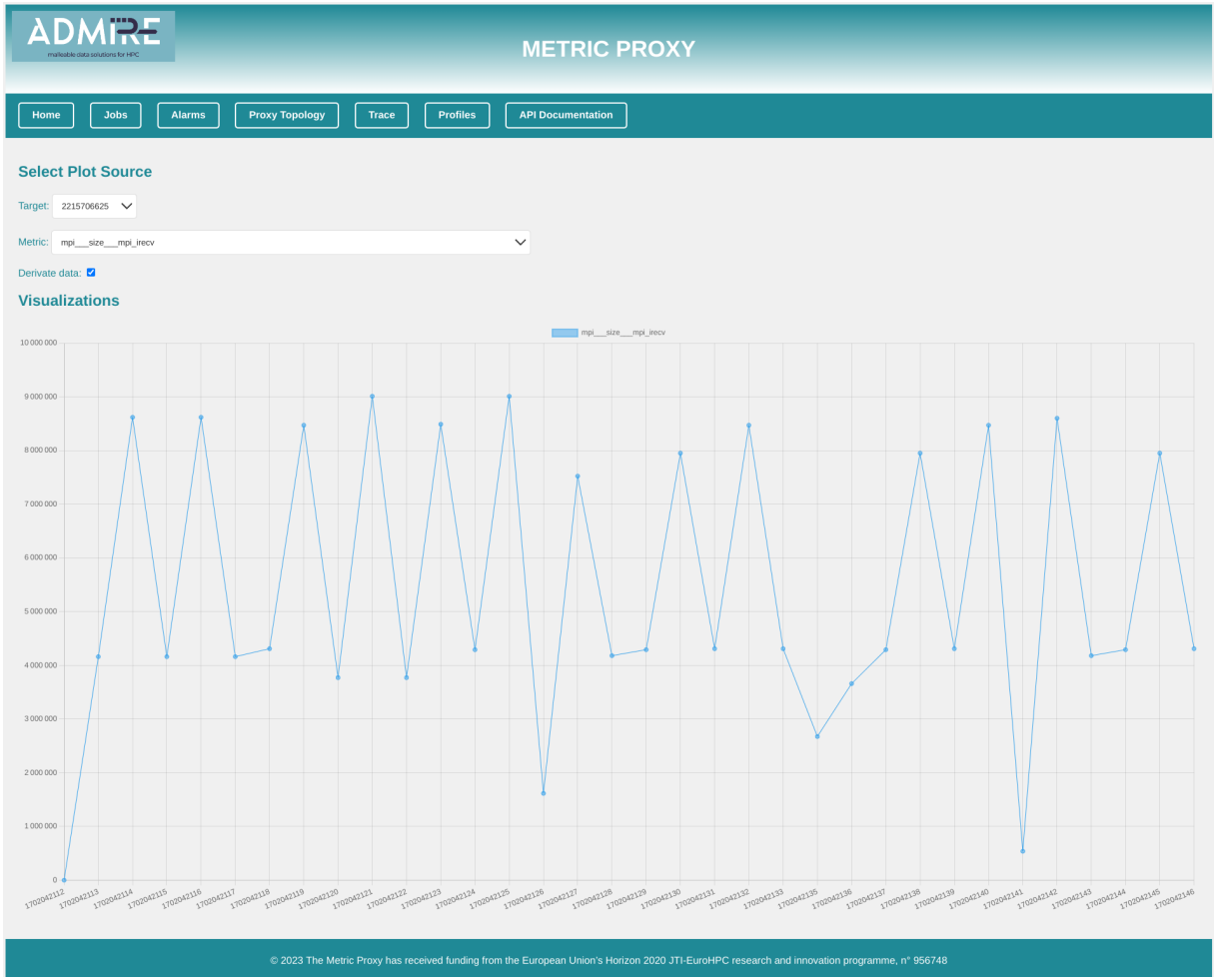
⋮

Name	Documentation	Value		
mpi___hits___mpi_cartdim_get	Number of function calls for MPI_Cartdim_get	0		
mpi___hits___mpi_publish_name	Number of function calls for MPI_Publish_name	0		
mpi___time___mpi_type_ub	Total seconds spent for MPI_Type_ub	0		
mpi___hits___mpi_send_init	Number of function calls for MPI_Send_init	0		
mpi___hits___mpi_status_set_elements_x	Number of function calls for MPI_Status_set_elements_x	0		
mpi___time___mpi_comm_set_attr	Total seconds spent for MPI_Comm_set_attr	0		
mpi___time___mpi_cart_create	Total seconds spent for MPI_Cart_create	0		
mpi___hits___mpi_raccumulate	Number of function calls for MPI_Raccumulate	0		
mpi___hits___mpi_type_create_subarray	Number of function calls for MPI_Type_create_subarray	0		
mpi___time___mpi_win_delete_attr	Total seconds spent for MPI_Win_delete_attr	0		
proxy_memory_swap_used_percent	Total swap usage on the system in percent	AVG: 23.154901660001453	Min: 23.154901660001453	Max: 23.154901660001453
mpi___hits___mpi_win_fence	Number of function calls for MPI_Win_fence	0		
mpi___hits___mpi_allreduce	Number of function calls for MPI_Allreduce	147047		
mpi___time___mpi_allgatherv	Total seconds spent for MPI_Allgatherv	20.255583867873455		
mpi___time___mpi_dist_graph_create	Total seconds spent for MPI_Dist_graph_create	0		
mpi___time___mpi_comm_split_type	Total seconds spent for MPI_Comm_split_type	0		
mpi___time___mpi_file_call_errhandler	Total seconds spent for MPI_File_call_errhandler	0		
mpi___hits___mpi_comm_rank	Number of function calls for MPI_Comm_rank	75		
mpi___hits___mpi_file_write_at	Number of function calls for MPI_File_write_at	0		
mpi___time___mpi_get_elements	Total seconds spent for MPI_Get_elements	0		
mpi___hits___mpi_group_intersection	Number of function calls for MPI_Group_intersection	0		
mpi___time___mpi_group_union	Total seconds spent for MPI_Group_union	0		
mpi___hits___mpi_info_create	Number of function calls for MPI_Info_create	0		
mpi___time___mpi_type_create_darray	Total seconds spent for MPI_Type_create_darray	0		

Sample Trace View

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

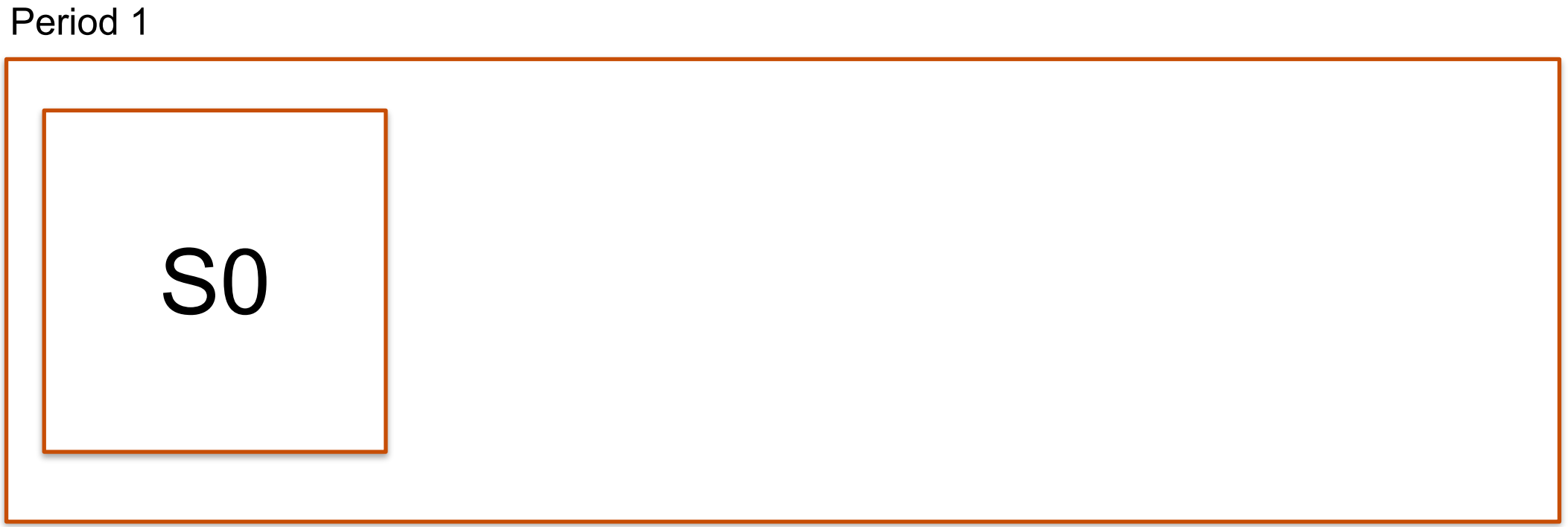


- A maximum fixed size of 32MB per job, filled with sample every one second and slowing down by a factor 2 on resampling.
- This maintains full-range traces by dynamically decreasing resolution and bounded size.

Trace Resampling

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming



Trace Resampling

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Trace Resampling

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Trace Resampling

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Trace Resampling

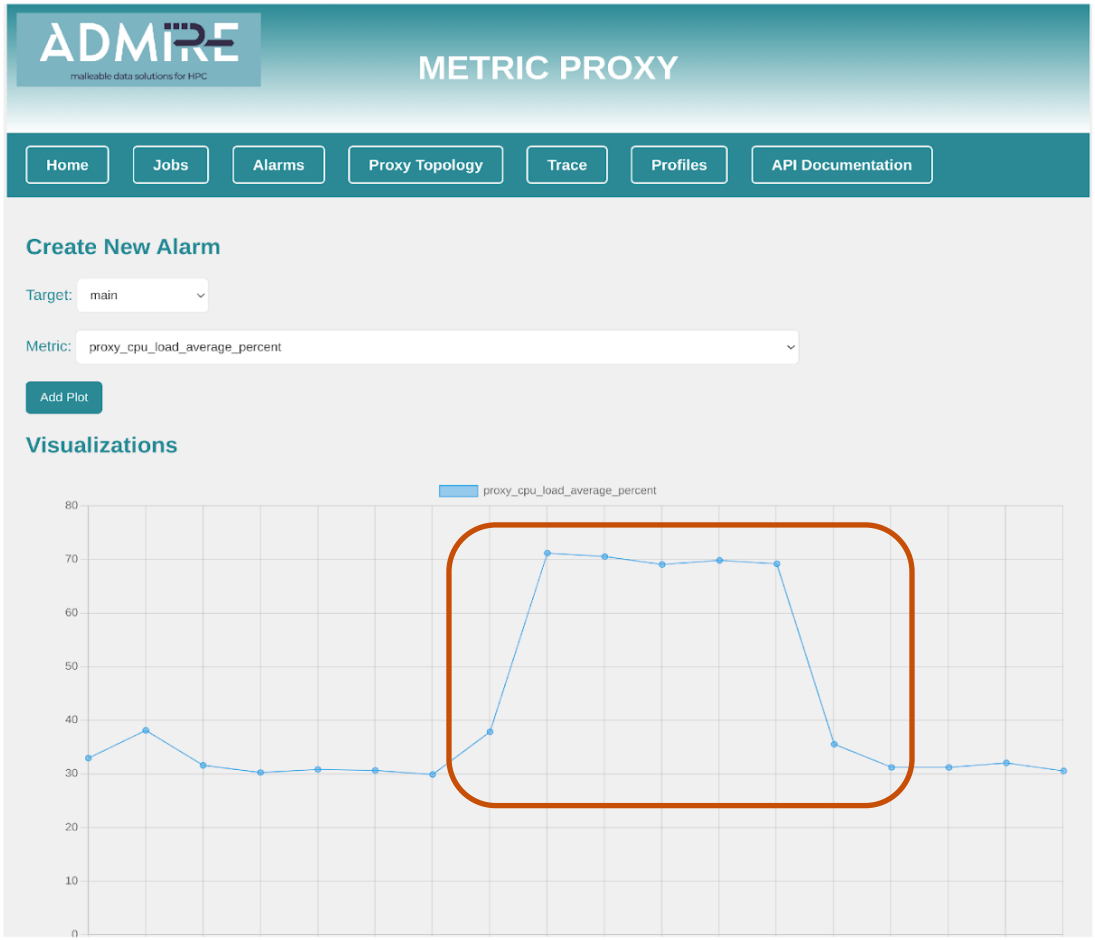
TECHNISCHE UNIVERSITÄT DARMSTADT

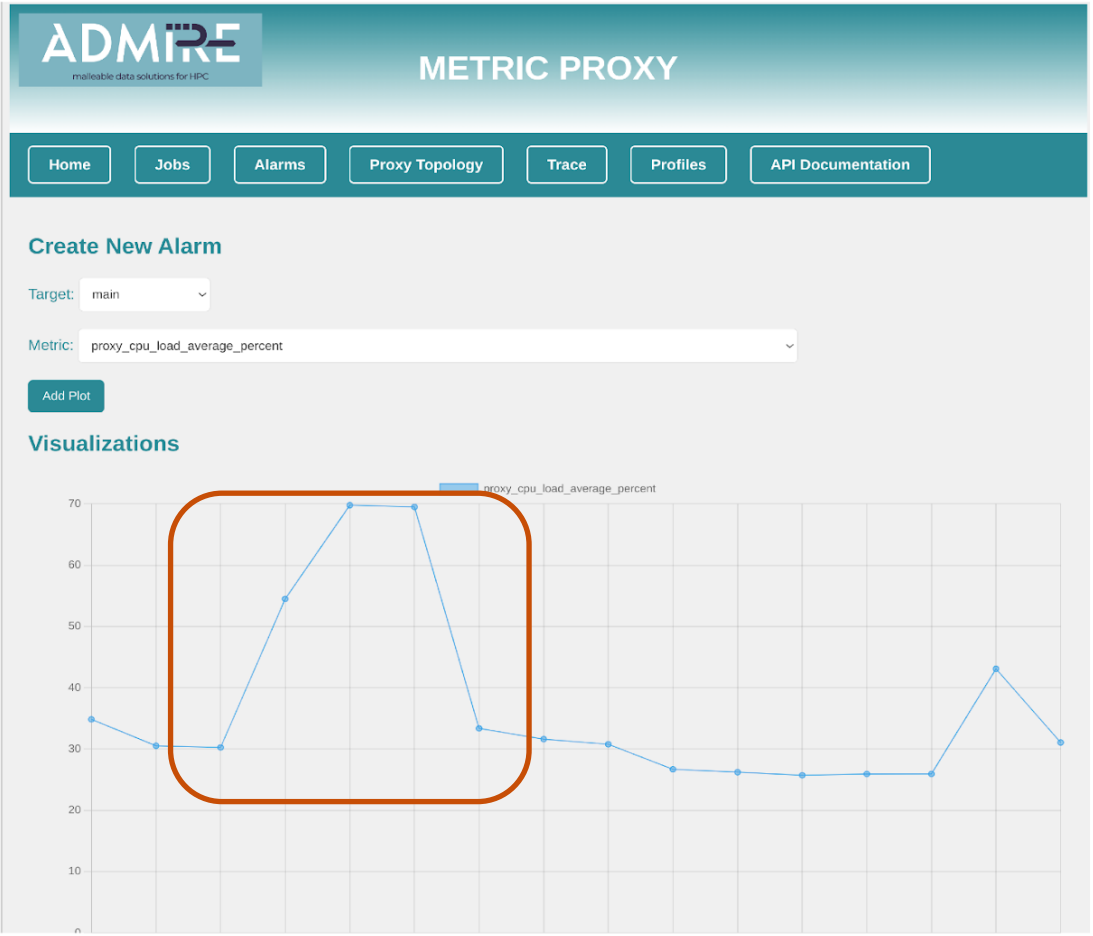
Parallel Programming

Trace Resampling

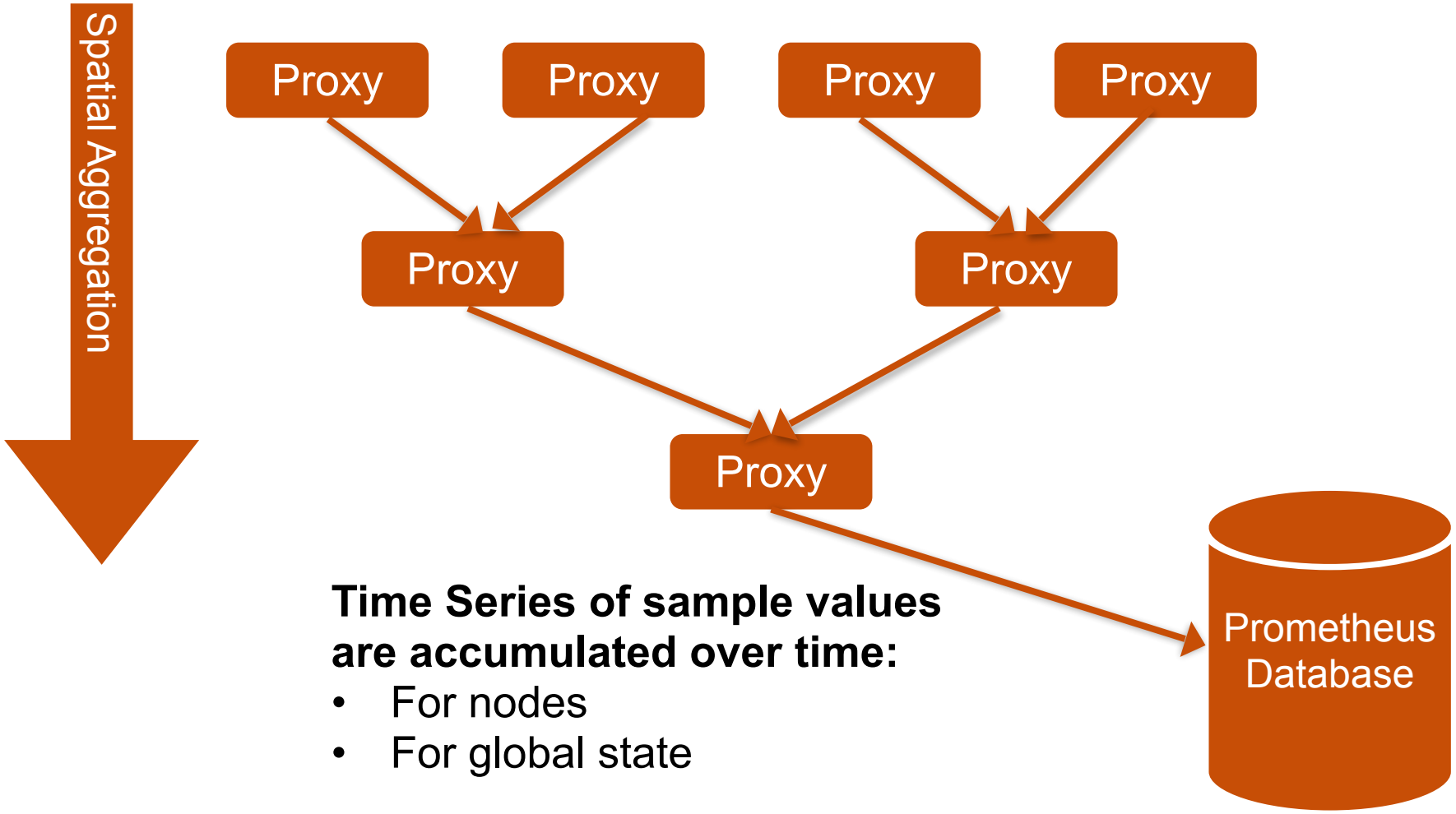
TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming





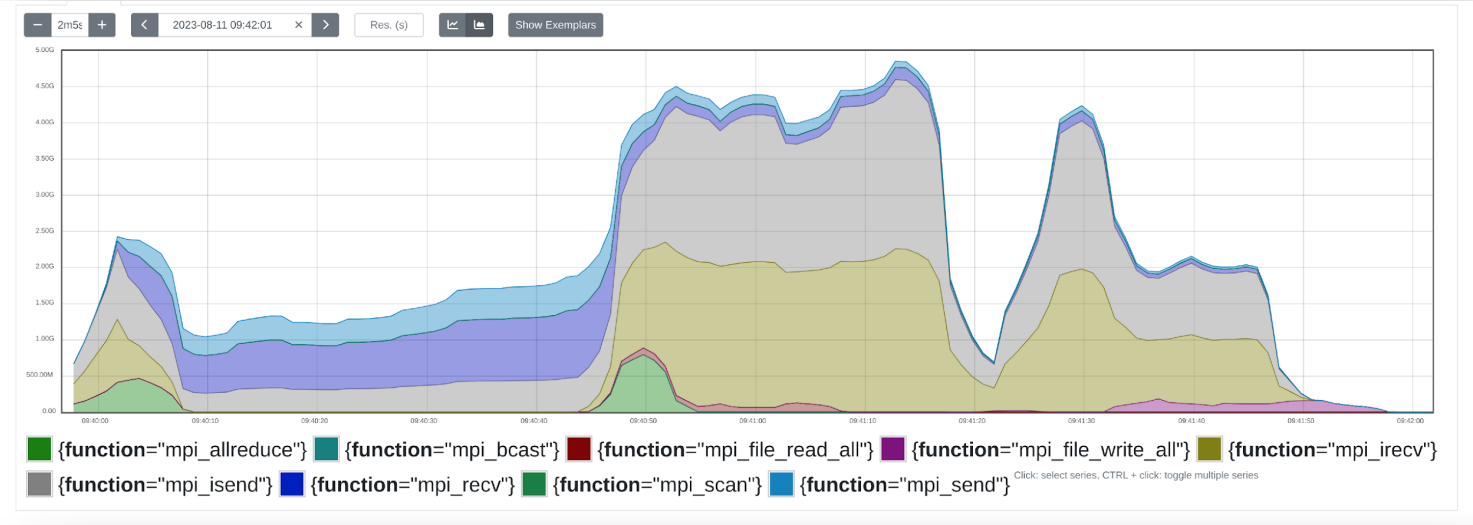
Prometheus Aggregation

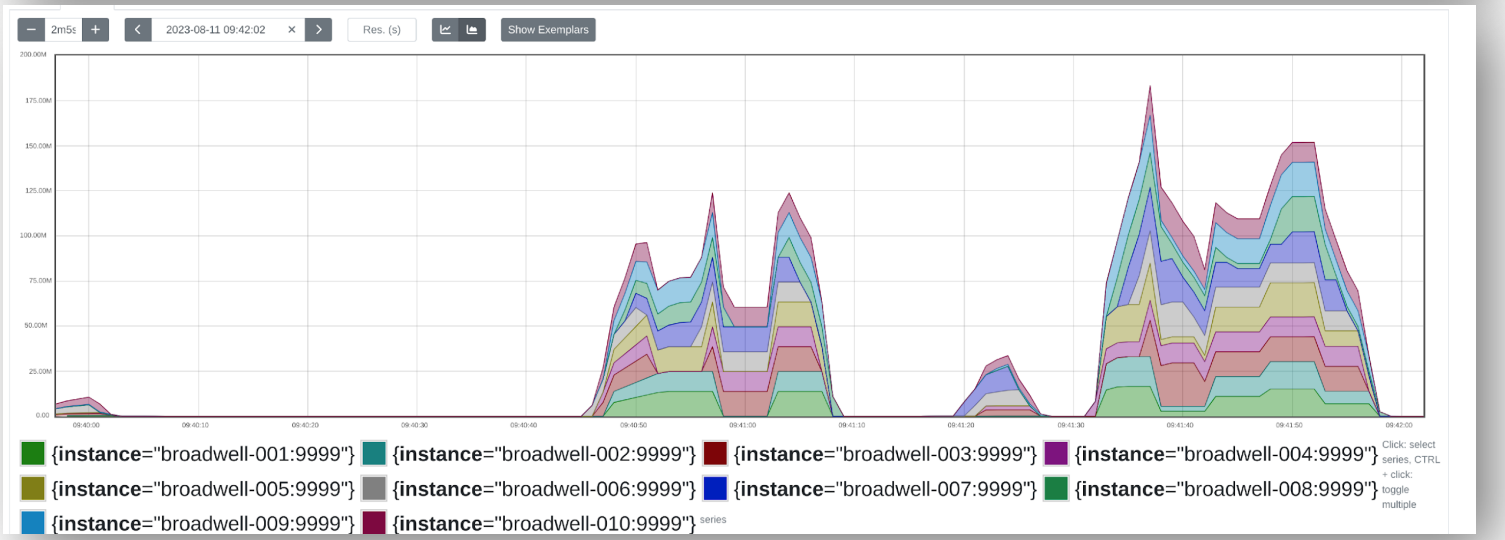


Time Series of sample values are accumulated over time:

- For nodes
- For global state

Nek5000 in Prometheus GUI





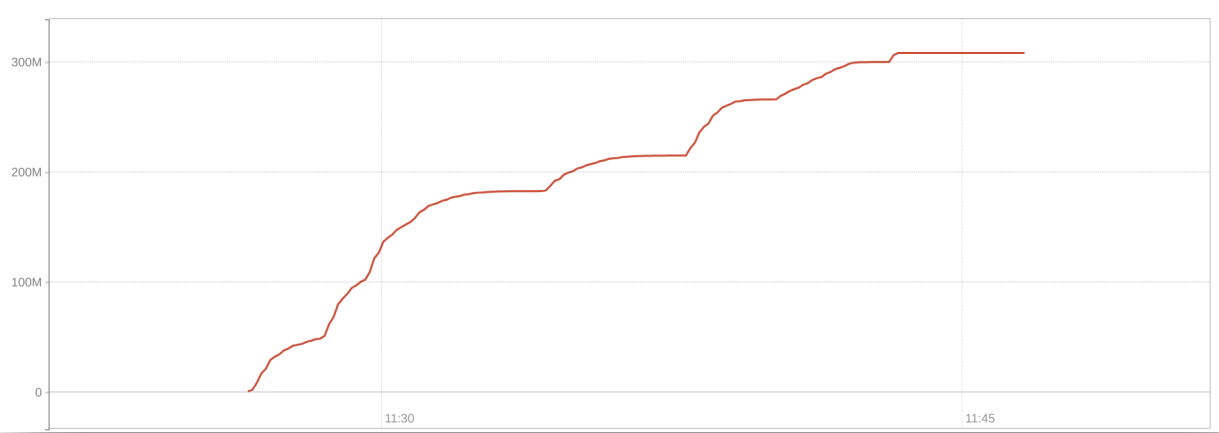
- Proxy examines MPI and I/O information simultaneously
- Example: **Nek5000**
  - is fast and scalable open source CFD solver
  - turbPipe test case (round turbulent pipe flow) running on 10 nodes (360 MPI ranks)

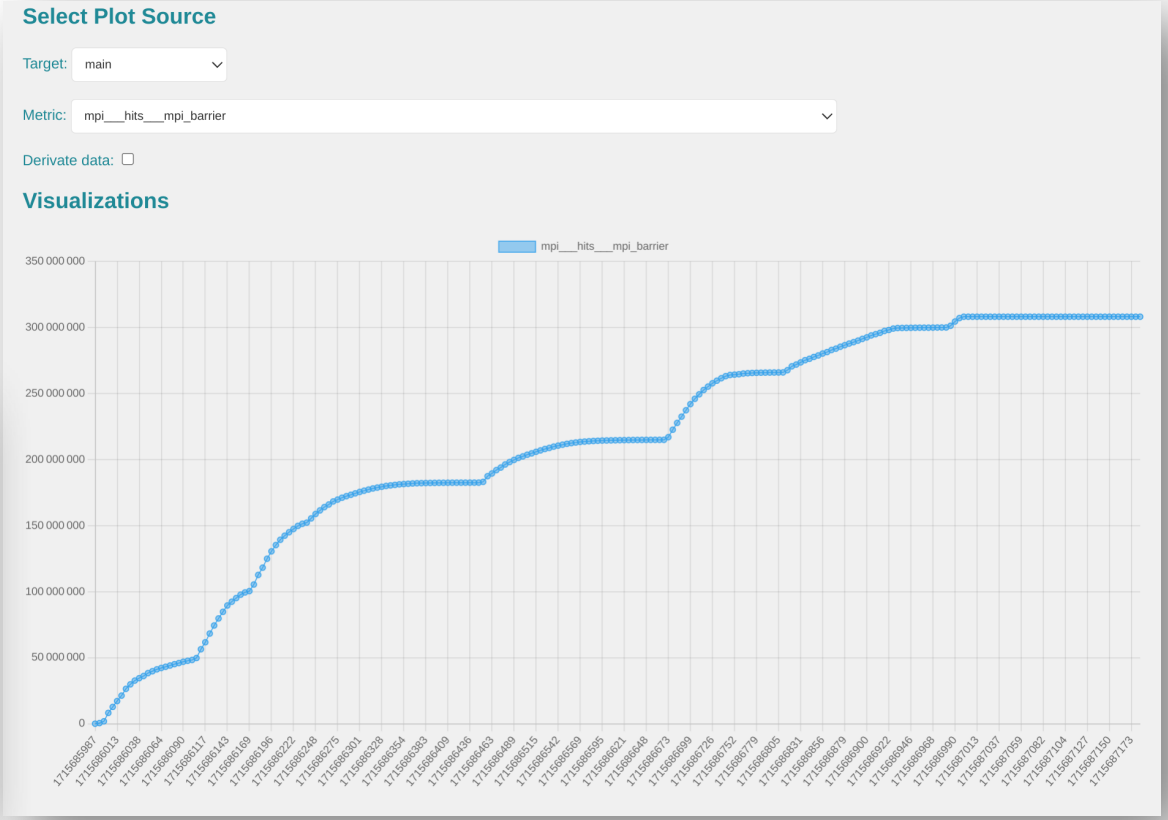
IMB-MPI1 with 50 Nodes (1800 Processes)

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

- Comparison between the Prometheus view and the trace view inside the proxy (GUI) for the total number of MPI_Barriers over time **summed** over 50 nodes with a **1 second** resolution
- It is possible to see the various steps of IMB

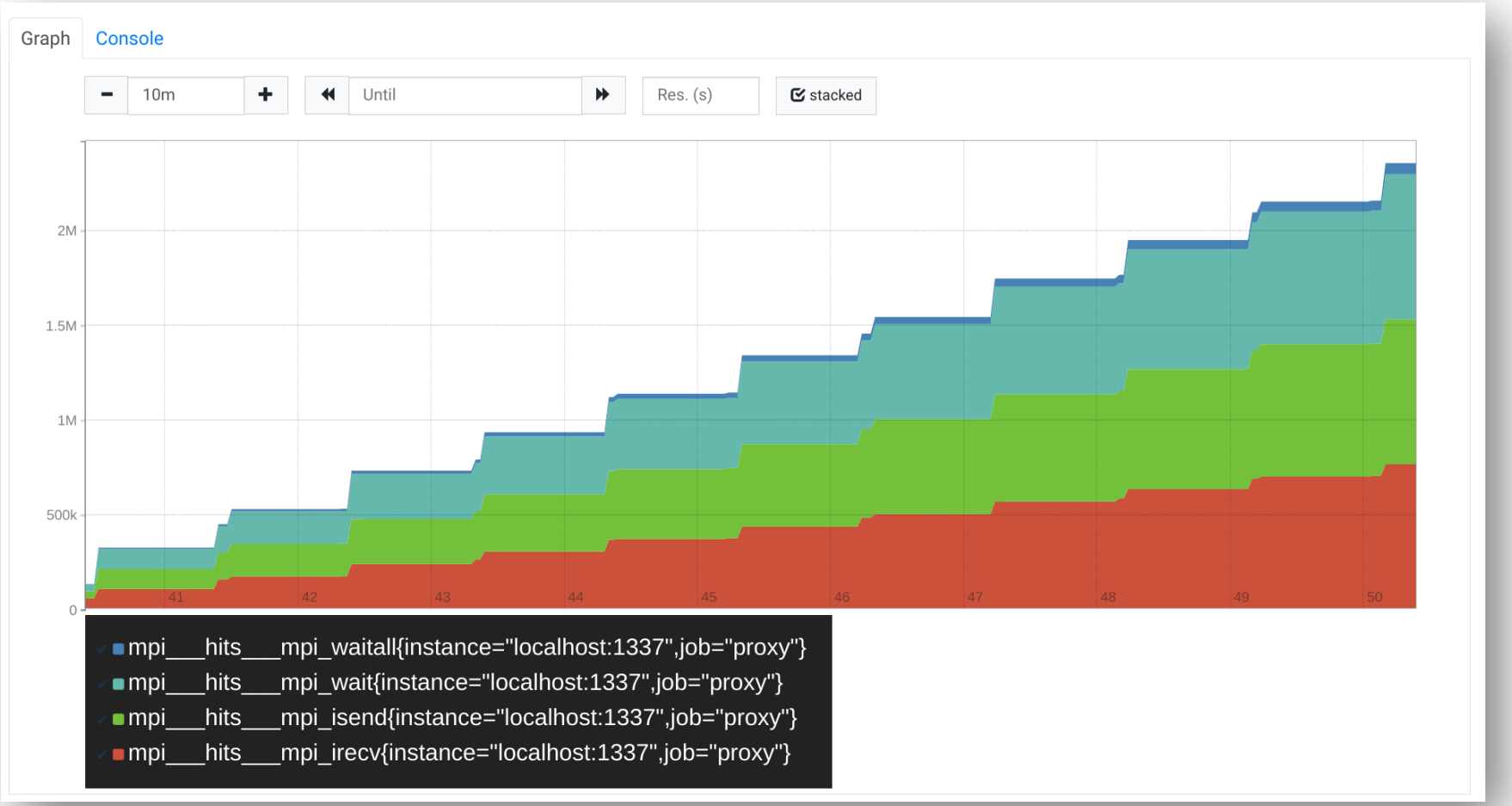




LULESH (MPI) 1728 Processes

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

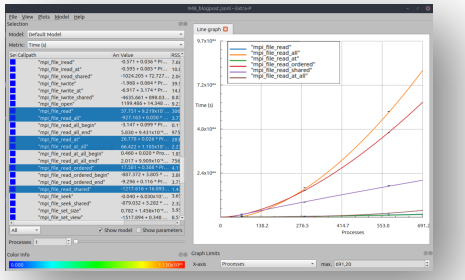


Stacked view of the number of calls for various MPI functions, timesteps are visible

TECHNISCHE UNIVERSITÄT DARMSTADT

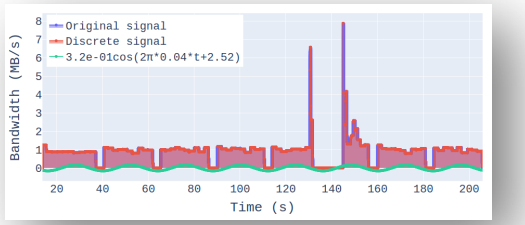
Parallel Programming

Performance Modeling



Extra-P

Metric Proxy

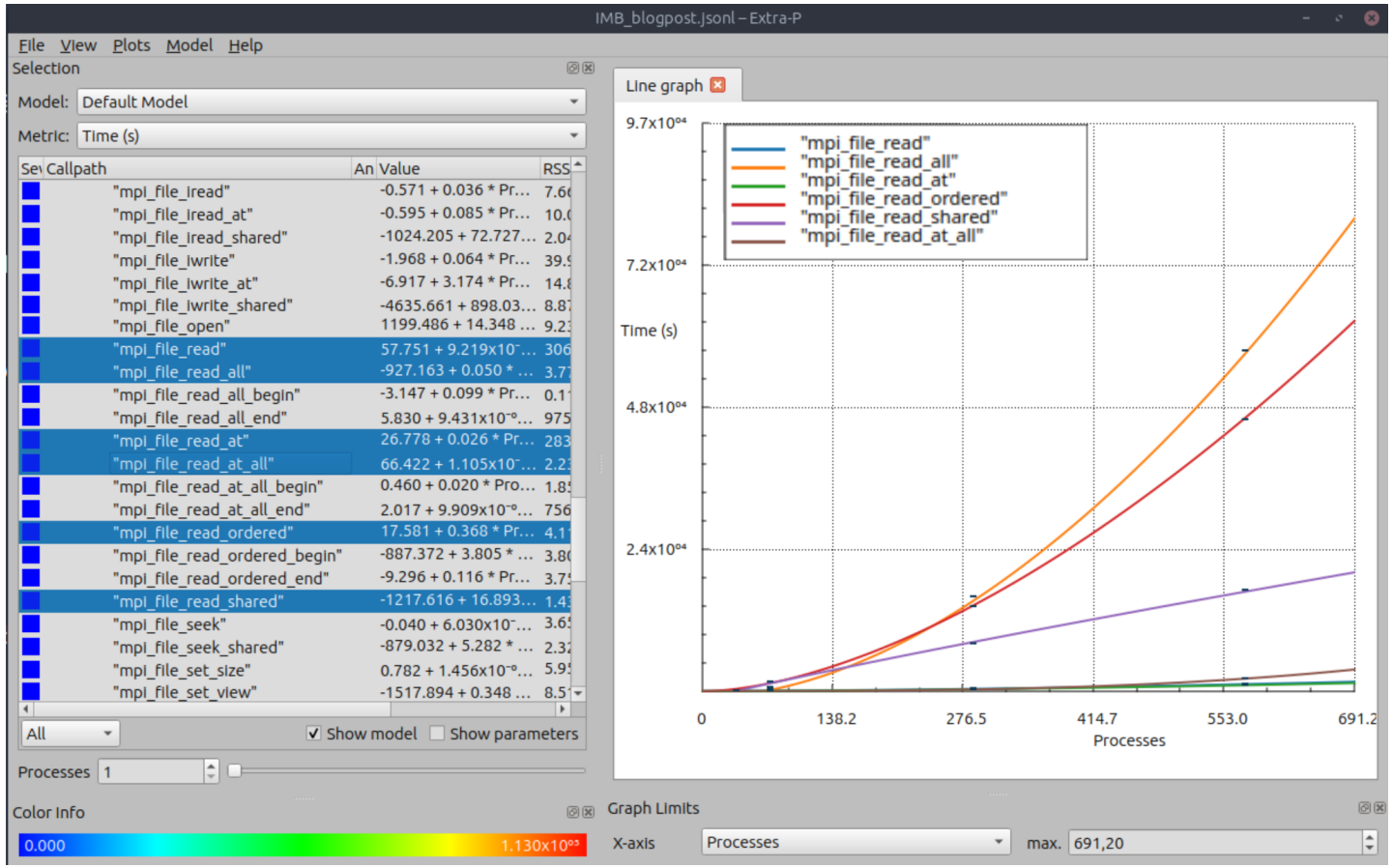


FTIO

Modeling From Profiles

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming



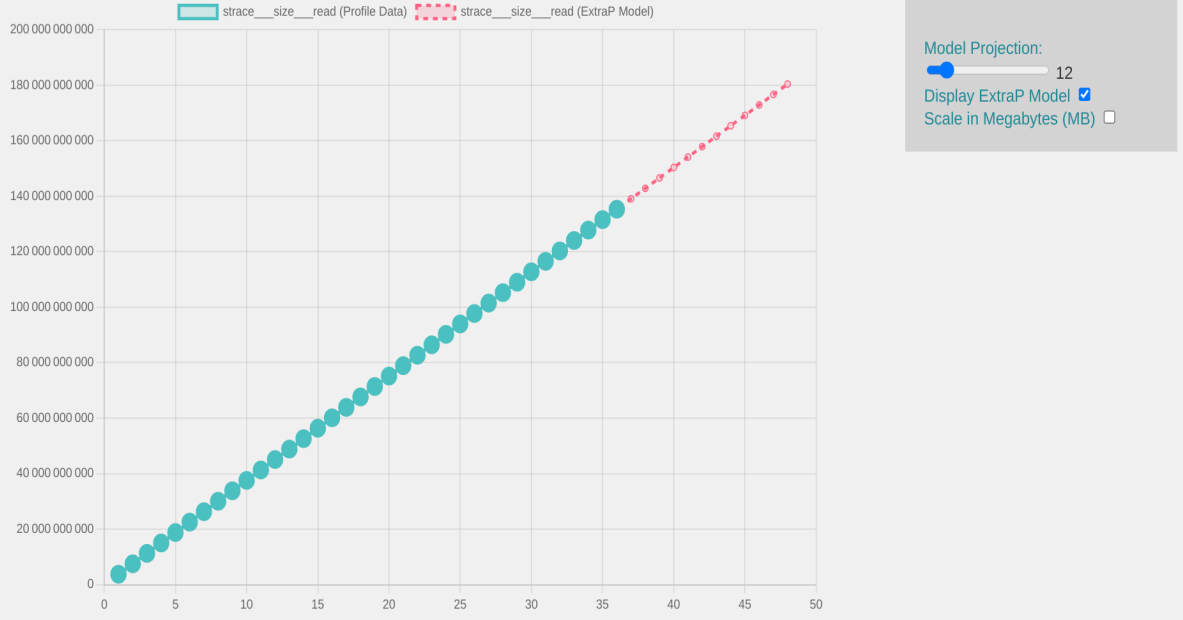
- Final state of job counters are saved by the metric proxy
- **Profiles** can be gathered by command line for various scales
- The Metric Proxy can leverage **Extra-P** to generate and expose performance models for all metrics of a given trace

Sample Model for Bytes Read

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

proxy_network_receive_bytes_total{interface="lo"}	3060492741632.313 + 64355661391.892494 * size^(2/3)	5.77e+23	Display Model
proxy_component_temperature_celcius{component="coretemp Core 9"}	22.561957780017977 + 0.48225689512319364 * (ln(size)/ln(2))^(2)	30	Display Model
strace___size___read	-1.2131179788106714e-05 + 3758116134.0 * size^(1)	4.56e-8	Display Model
proxy_cpu_frequency_ghz{name="cpu32",vendor="GenuineIntel",model="Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz"}	3089.0000000000023	1.86e-22	Display Model
proxy_component_critical_temperature{component="coretemp Core 9"}	0.0	0	Display Model



- The proxy has functions for various metrics and can use **Extra-P** interpolate values at scales not yet run
- This information can be valuable for dimensioning burst buffers or controlling ad-hoc file systems

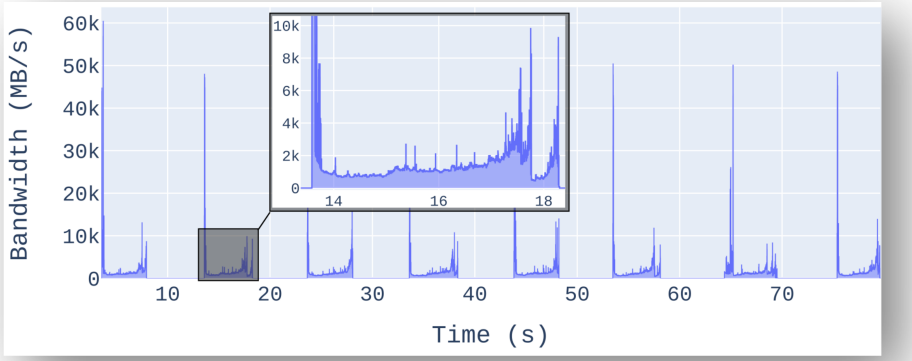
Predicting Temporal Behavior Using Frequency Techniques (FTIO)

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

→ Periodic I/O is often encountered in HPC!

Information about applications' periodicity, **even if not perfectly precise,** leads to good contention-avoidance techniques [1, 2, 3]



→Frequency Techniques for I/O:

- Examine the I/O behavior in the **frequency domain**
- Describes the temporal behavior of the **I/O phases** through a single metric, namely the **period** (T_d)
- Online (**prediction**) and offline (**detection**) realizations with low overhead
- Additional metrics quantify confidence in the results and **further characterize** the I/O behavior

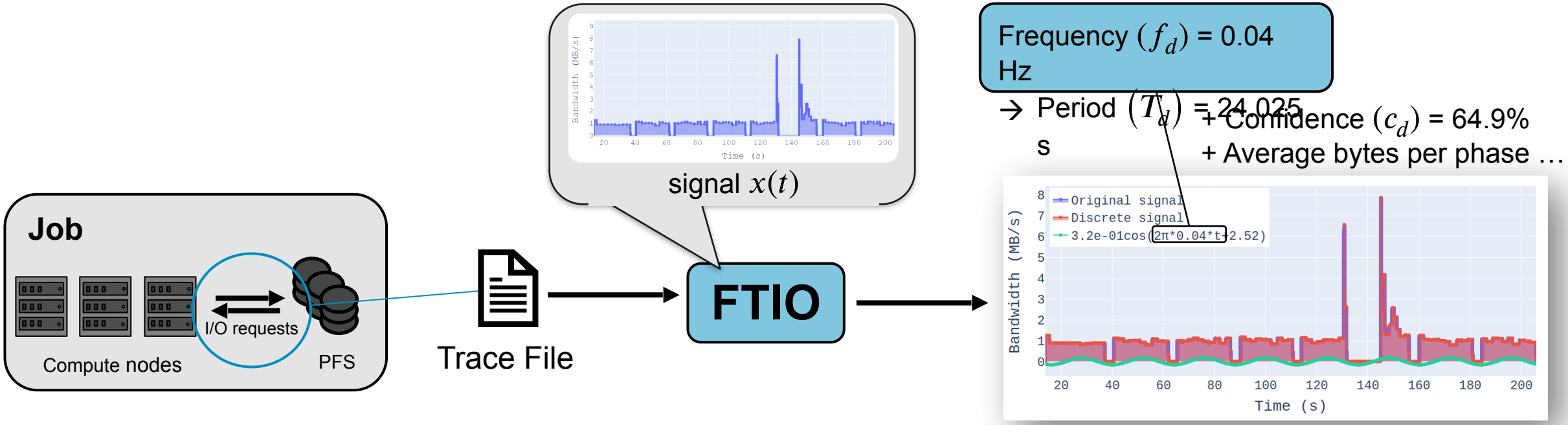
IPDPS 2024

Period (T_d) of I/O phases:
The time between the start of consecutive I/O phases

FTIO: Output (Simplified)

TECHNISCHE UNIVERSITÄT DARMSTADT

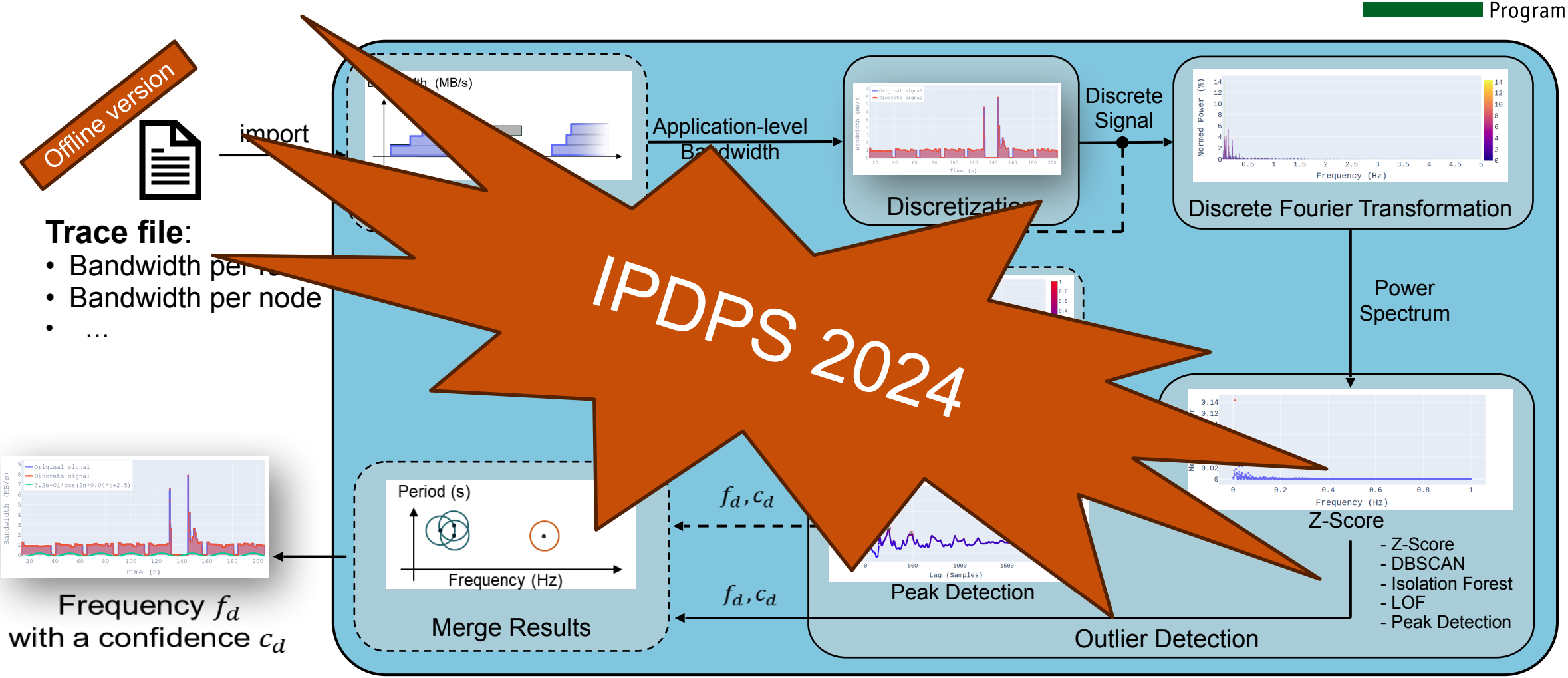
Parallel Programming



FTIO: The Core

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming



FTIO: Required Input

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Supported Formats/Tools for **online prediction**:

- TMIO (JSONL, MessagePack, ZeroMQ)
- **Metric Proxy**

TMIO:
- **T**racing **M**PI-**IO**
- C++ library that uses the PMPI interface
- Flushes I/O data online
- Can be easily attached to existing code
- Will be made publicly available

Supported Formats/Tools for **offline detection**:

- Darshan
- Recorder (folder)
- TMIO (JSON, JSONL, MessagePack, ZeroMQ)
- **Metric Proxy**

demo.json

```
12  "bandwidth": {
13      "b_rank_avr": [1.496276, 2.013454, 2.062243, ...],
14      "t_rank_start": [1.950272, 1.964889, 1.975749, ...],
15      "t_rank_e": [1.964871, 1.975739, 1.986342, ...]
16  }
```

FTIO: Required Input (cont')

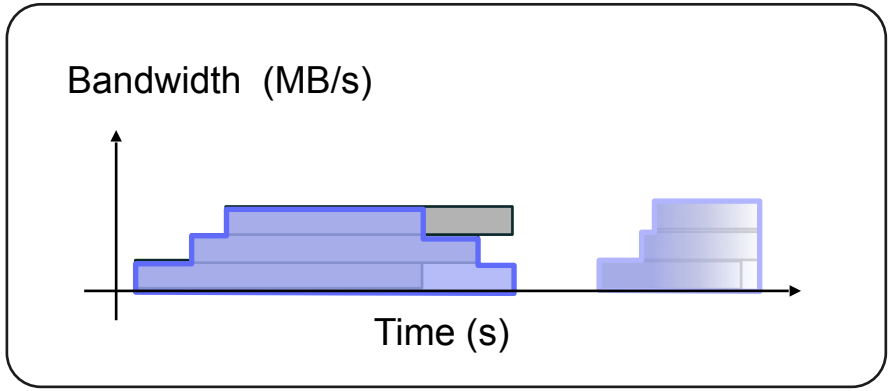
TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Trace file containing:

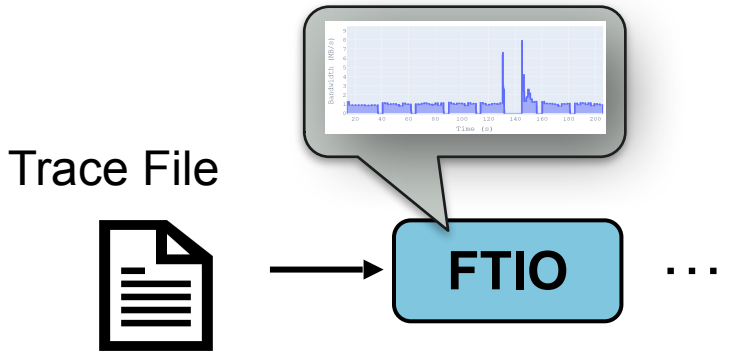
- Bandwidth per rank
- Time (start and end) when the bandwidth changed

→FTIO calculates internally the **application-level bandwidth** by **overlapping** the rank-level metrics



Application-level bandwidth and (start) time can also be provided directly → Any level is ok!

→ **Metric Proxy provides the application-level bandwidth!**

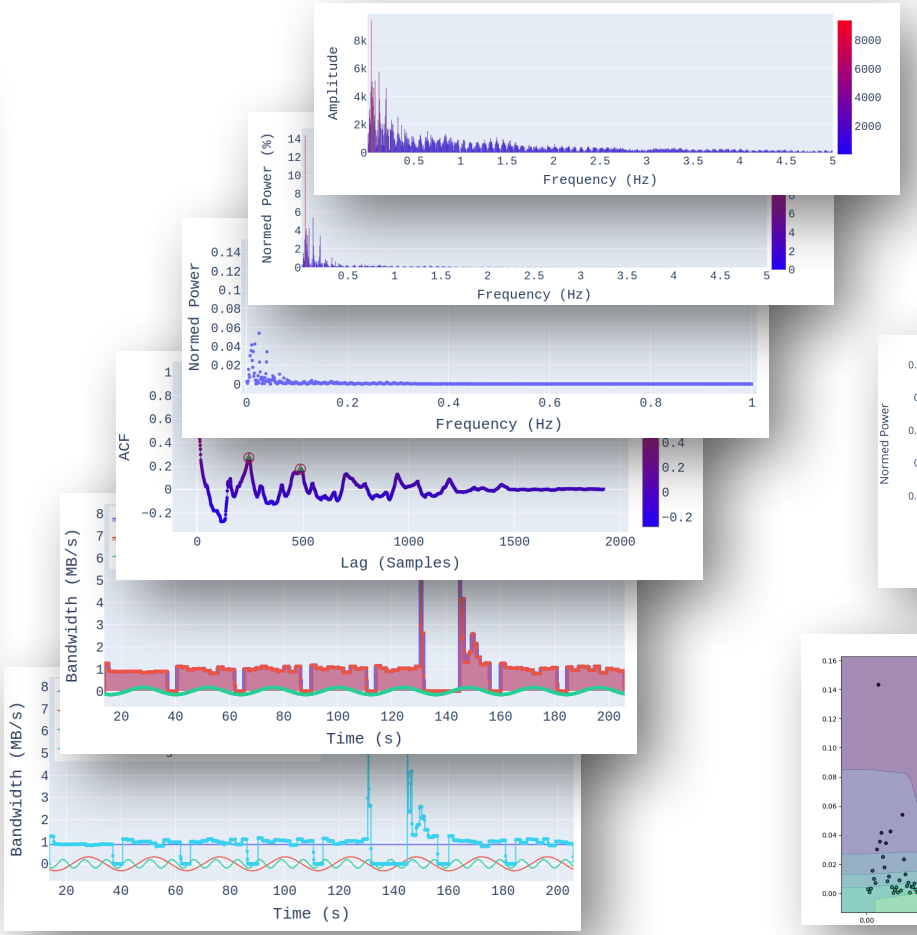


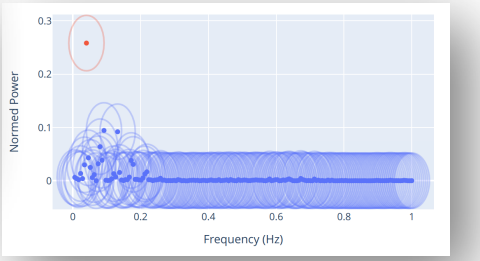
FTIO: User Interface

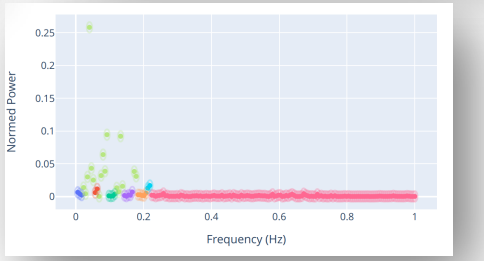
TECHNISCHE UNIVERSITÄT DARMSTADT

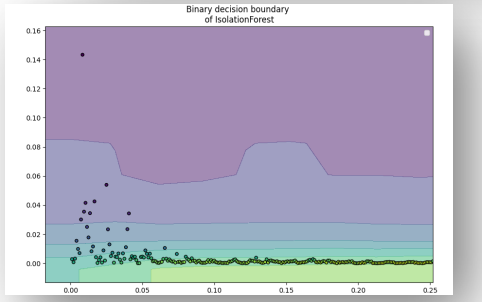
Parallel Programming

```
Tilix
 Discretization
  Time window : 192.223398 s
  Frequency step: 0.005202 Hz
  Sampling frequency:  10.0 Hz
  Expected samples: 1922
  Abstraction error: 0.000788

Discretization finished: 0.054 s
Executing: DFT + Z-score

 Dft
  Ranks: 1536
  Start time: 13.76 s
  End time: 205.98 s
  Ignored bytes: 0.000000

  Z-score
   Spectrum: Power spectrum
   mean: 1.041e-03
   std: 1.874e-03
   Frequencies with Z-score > 3 -> 11 candidates
            + Z > Z_max*80.0% > 3 -> 1 candidates
   Dominant frequency at:  4.162e-02 Hz (T = 24.025 s, k = 8) -> confidence: 64.953%

  Precision of 0.04 Hz is 11.16% (Positive only: 11.33%)

DFT + Z-score finished: 0.006 s
Total ellapsed time: 0.838 s

Prediction results:
Frequency: 4.162e-02 Hz -> 24.025 s
Confidence: 64.95 %
```







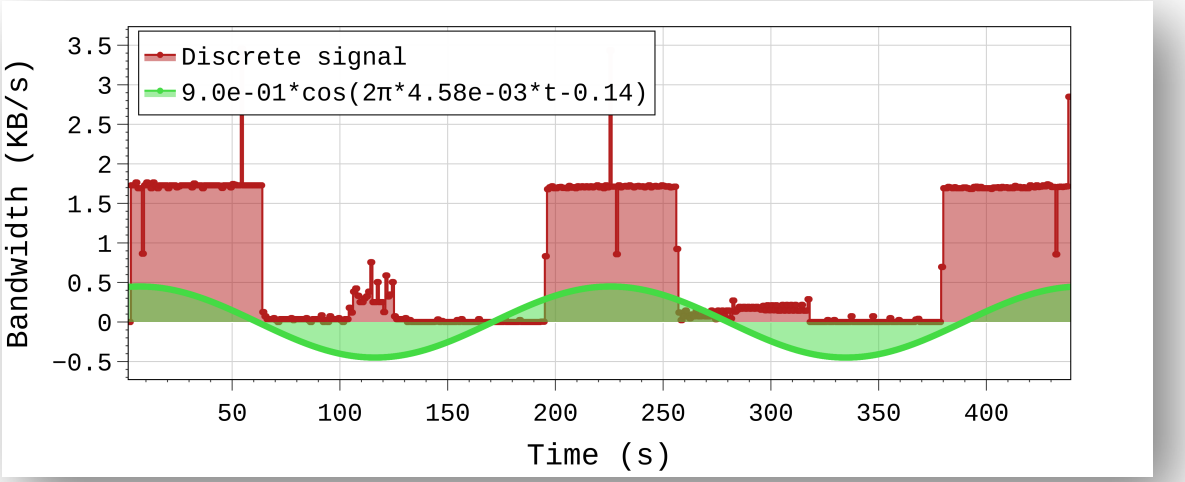


FTIO and the Metric Proxy:

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

- The paper shows a *basic* prediction for the total bytes for a small benchmark
- So far, the proxy provides the trace to FTIO, but this **interaction could be enhanced**
- We could also build a **feedback loop** to utilize FTIO's predictions for:
  - Enhanced modeling approaches
  - System components



Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

- We describe the **Metric Proxy**, a machine-wide transversal monitoring infrastructure able to generate several performance outputs:
  - Prometheus Integration (Both for scrape and as an exporter)
  - Performance trace and Profiles
  - Spatial aggregation with TBON
  - Job aware

- We listed associated performance prediction efforts:
  - Performance models using Extra-P integrated in the Metric Proxy
  - I/O Phase prediction using FTIO by using the traces from the Metric Proxy

  - Contact: ahmad.tarraf@tu-darmstadt.de

Future Work

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

- Utilize the phase detection for contention avoidance using live data from the metric proxy coupled with FTIO analysis (feedback loop)
- Work on the TBON to evaluate alternative topologies and examine the dynamics of the approach
- Work on enhancing integrations of the Metric Proxy with other tools

Available in Open Source

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Metric Proxy:

- https://github.com/besnardjb/proxy_v2
- Rust: can be easily installed using cargo
- Spack recipe is also available (contact us)

FTIO:

- https://github.com/tuda-parallel/FTIO
- Python: can be easily installed using pip

Extra-P:

- https://github.com/extra-p/extrap
- Python: can be installed using pip

Users, contributions, and collaborations are welcomed!

Acknowledgment

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

We acknowledge the support of the European Commission and the German Federal Ministry of Education and Research (BMBF) under the EuroHPC Programmes DEEP-SEA (GA no. 955606, BMBF funding no. 16HPC015) and ADMIRE (GA no. 956748, BMBF funding no. 16HPC006K), which receive support from the European Union's Horizon 2020 programme and DE, FR, ES, GR, BE, SE, GB, CH (DEEP-SEA) or DE, FR, ES, IT, PL, SE (ADMIRE). This work was also funded by the French National Research Agency (ANR) in the frame of DASH (ANR-17-CE25-0004), by the Project Région Nouvelle Aquitaine 2018-1R50119 "HPC scalable ecosystem". This work was also supported by the Spanish Research Agency (AEI) through the SCIOT project, with reference PID2022-138050NB-I00. This work was also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project No. 449683531 (ExtraNoise).

We gratefully acknowledge the computing time provided to them on the high-performance computer at the University of Turin from the laboratory on High-Performance Computing for Artificial Intelligence. Furthermore, We gratefully acknowledge the computing time provided on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities *(www.nhr-verein.de/unsere-partner*). Moreover, we also gratefully acknowledge the computing time on the PlaFRIM experimental testbed supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (https://www.plafrim.fr)

ADMIRE malleable data solutions for HPC

EuroHPC Joint Undertaking

Federal Ministry of Education and Research

EXTRA NOISE

AGENCIA ESTATAL DE INVESTIGACIÓN

Agence Nationale de la Recherche ANR

DEEP-SEA

References

1. Anne Benoit, Thomas Herault, Lucas Perotin, Yves Robert, and Frédéric Vivien. 2023. Revisiting I/O bandwidth-sharing strategies for HPC applications. Technical Report RR-9502. INRIA. 56 pages. https://hal.inria.fr/hal-04038011
2. Matthieu Dorier, Gabriel Antoniu, Rob Ross, Dries Kimpe, and Shadi Ibrahim. 2014. CALCioM: Mitigating I/O interference in HPC systems through crossapplication coordination. In IPDPS'14. IEEE, 155–164.
3. Emmanuel Jeannot, Guillaume Pallez, and Nicolas Vidal. 2021. Scheduling periodic I/O access with bi-colored chains: models and algorithms. J. of Scheduling 24, 5 (2021), 469–481.

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Questions?

Thank you for your attention!