

What Happens in MPI Shouldn't Stay in MPI

Making Communication Visible for Learning and Performance Analysis

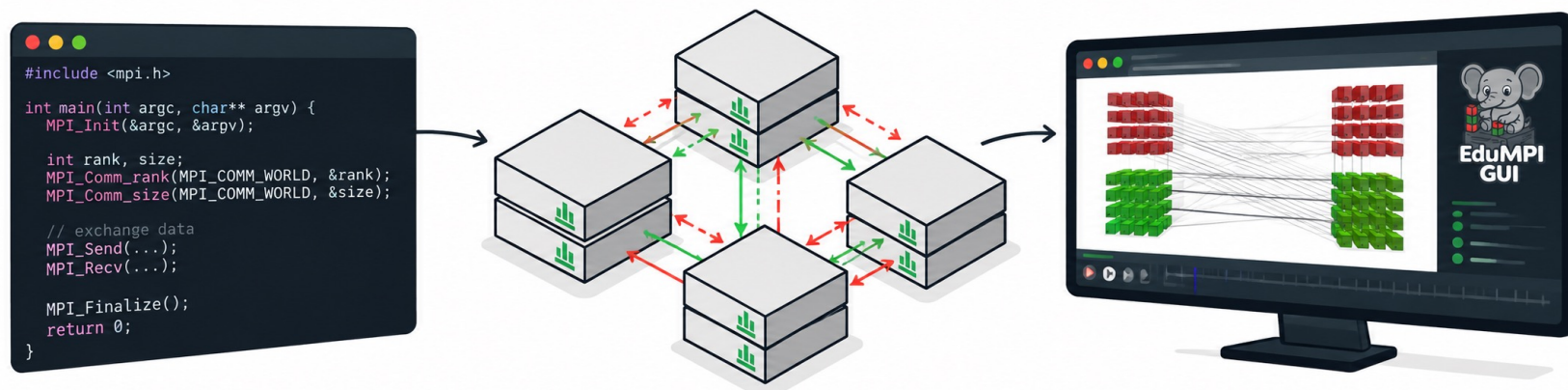
Dr. Anna-Lena Roth

Postdoctoral Researcher
anna-lena.roth@cs.hs-fulda.de
www.annalenaroth.com

Fulda University of
Applied Sciences



Fulda
HPC

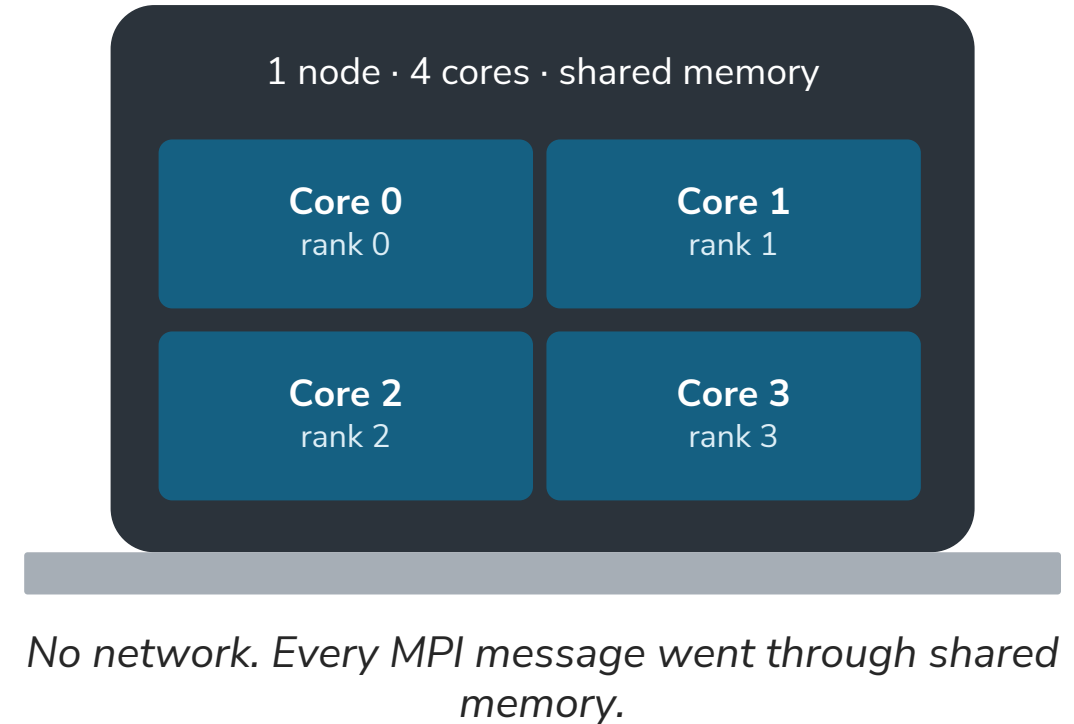


Invited Talk
EuroMPI 2026

TU Wien (University of Technology, Vienna)
October 8, 2026

How I Learned MPI in 2020

- Pandemic semester: MPI on my laptop with 4 cores
- No cluster at the university
 - Scalability testable up to 4 processes only
- Learning MPI meant writing code
 - Syntax, compile, run, check the output
- MPI felt just like OpenMP
 - Threads vs. processes: no visible difference
- Clusters and HPC: theory on lecture slides



What My Laptop Could Not Show Me

The network

Only shared memory: every message stayed inside one "node".

Processes vs. threads

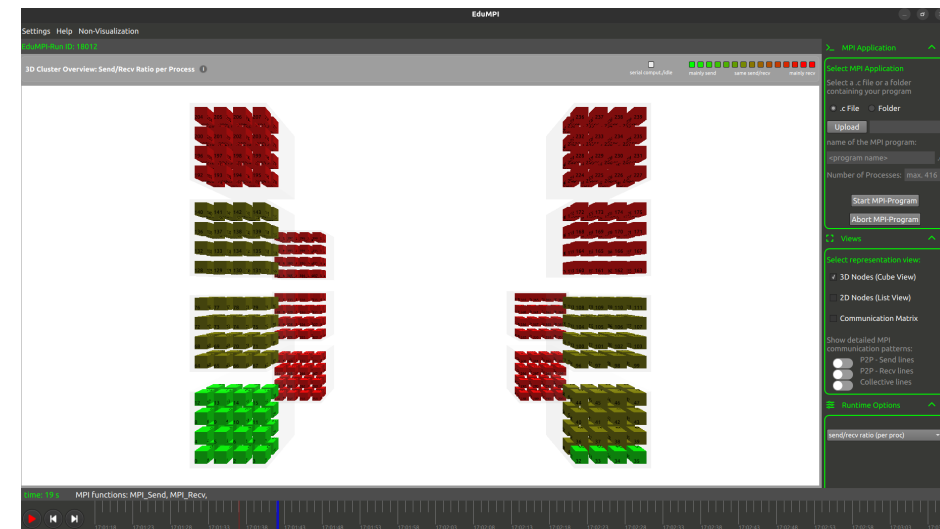
Separate processes, but on one laptop they behaved just like OpenMP threads.

Scalability

Four processes on one node: no placement, no inter-node cost.

Performance

Nothing worth analyzing on four cores: no network, no scale, no real bottlenecks. And the tools were too complex at that stage.



The same kind of program on a cluster: processes spread across several nodes (EduMPI GUI)

My laptop hid exactly the part of MPI that makes MPI MPI.

What Completely Changed My Understanding of MPI

Research project during my studies

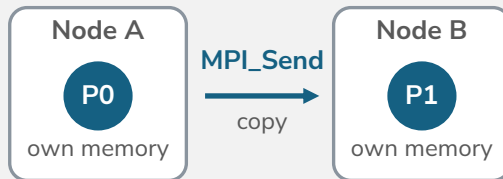


Real HPC clusters



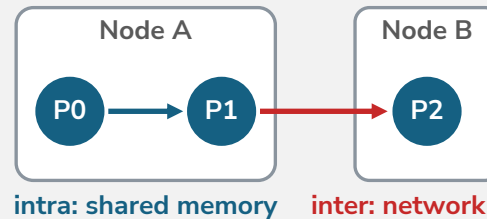
Performance analysis tools

Processes ≠ threads



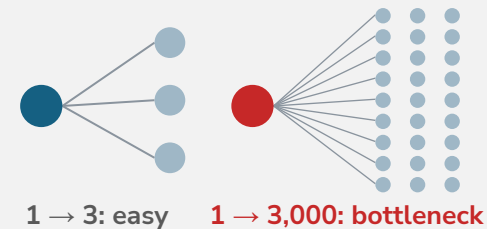
Separate address spaces: data is copied and sent explicitly.

Inter-node ≠ intra-node



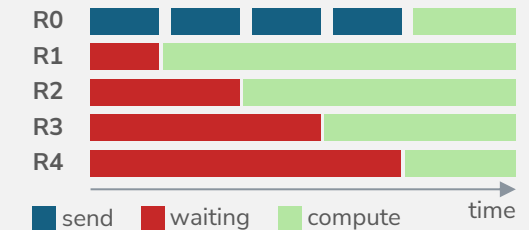
Latency, bandwidth, and process placement suddenly mattered.

Performance at scale



Sending to 3 processes is easy, to 3,000 a bottleneck. Late senders get more frequent, load balancing matters more.

Tools make it visible



Hotspots and waiting times I never saw while coding, and data on why they cost time.

*I needed cluster access and performance analysis tools to understand MPI.
In many HPC courses, students get neither.*

Few Universities of Applied Sciences Have a Cluster

Institutions operating a local HPC cluster

Universities (n = 64)

85,9 %

Universities of Applied Sciences (n = 38)

21,1 %

23.0 %

of the 87 local clusters are documented as available for teaching

11.2 %

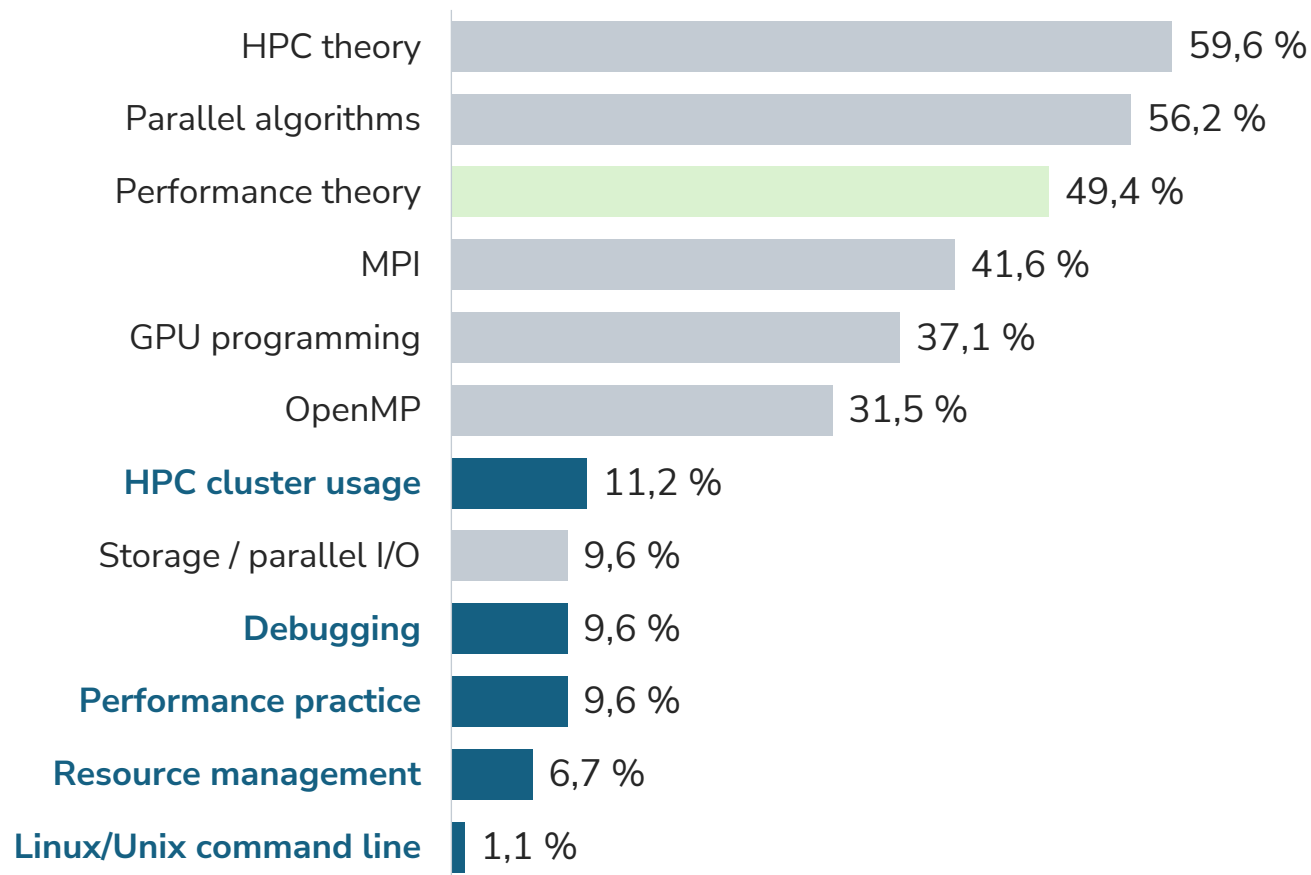
of 178 HPC courses explicitly use a cluster

Only **8** of **38** Universities of Applied Sciences operate a cluster, compared to **55** of **64** Universities.



Performance Analysis: Taught in Theory, Rarely in Practice

Share of HPC courses covering the topic (n = 178)



49.4 %

cover performance theory: speedup, scalability, Amdahl

9.6 %

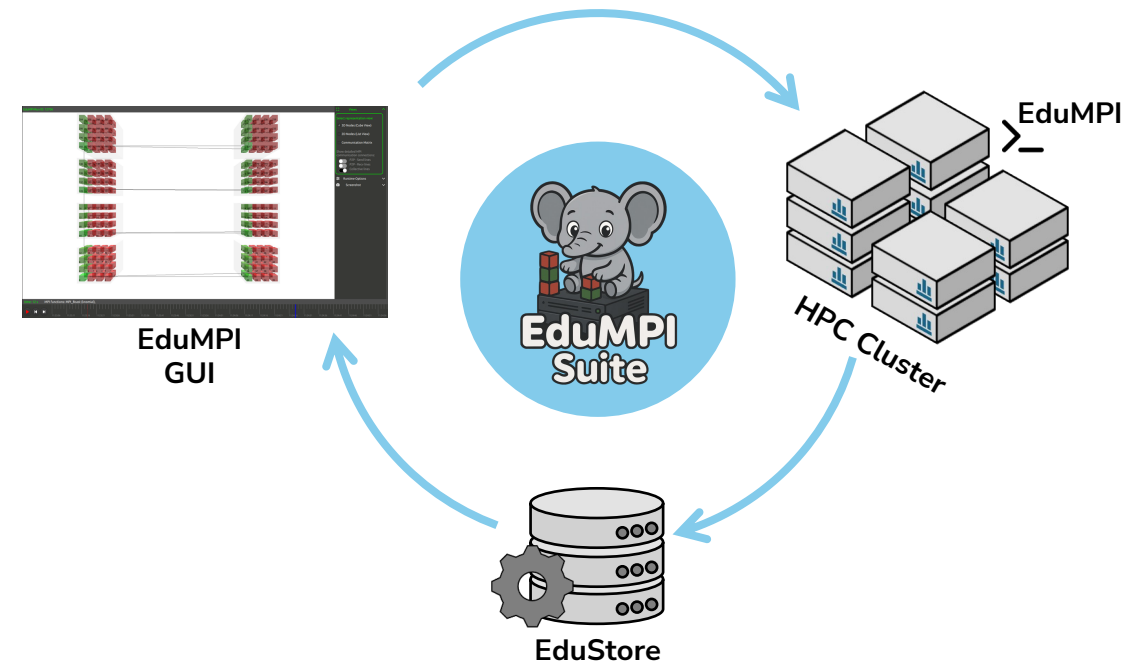
cover performance practice: profiling, tracing, optimization

Dark blue: practical competencies



Using Clusters and Integrating Performance Analysis into HPC Education

- Parallel thinking is a core competence
 - Students must learn how to divide computational work across many processes and coordinate their execution.
- MPI requires conceptual understanding
 - Efficient MPI programs depend on data distribution, communication and synchronization.
 - Learning the syntax alone is insufficient.
- Performance belongs in the curriculum
 - A correct parallel program can still scale poorly.
 - Students need to recognize communication overhead, bottlenecks and load imbalance.

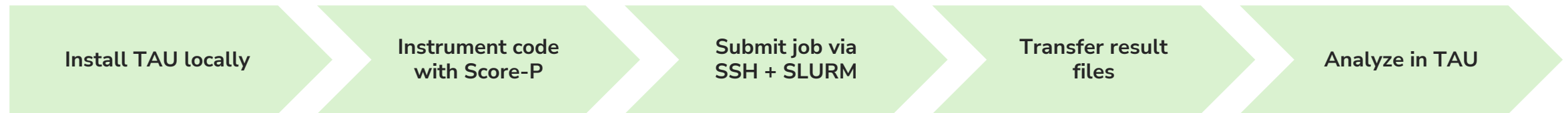


Our First Attempt: Performance Analysis in the Last Week

Parallel Programming course (master), SuSe 2022 and WiSe 2022/23



What students had to master in that one final exercise:



Visualization only after a successful post-mortem pipeline, and students still had to know which view answers which question.

≈ 26 min

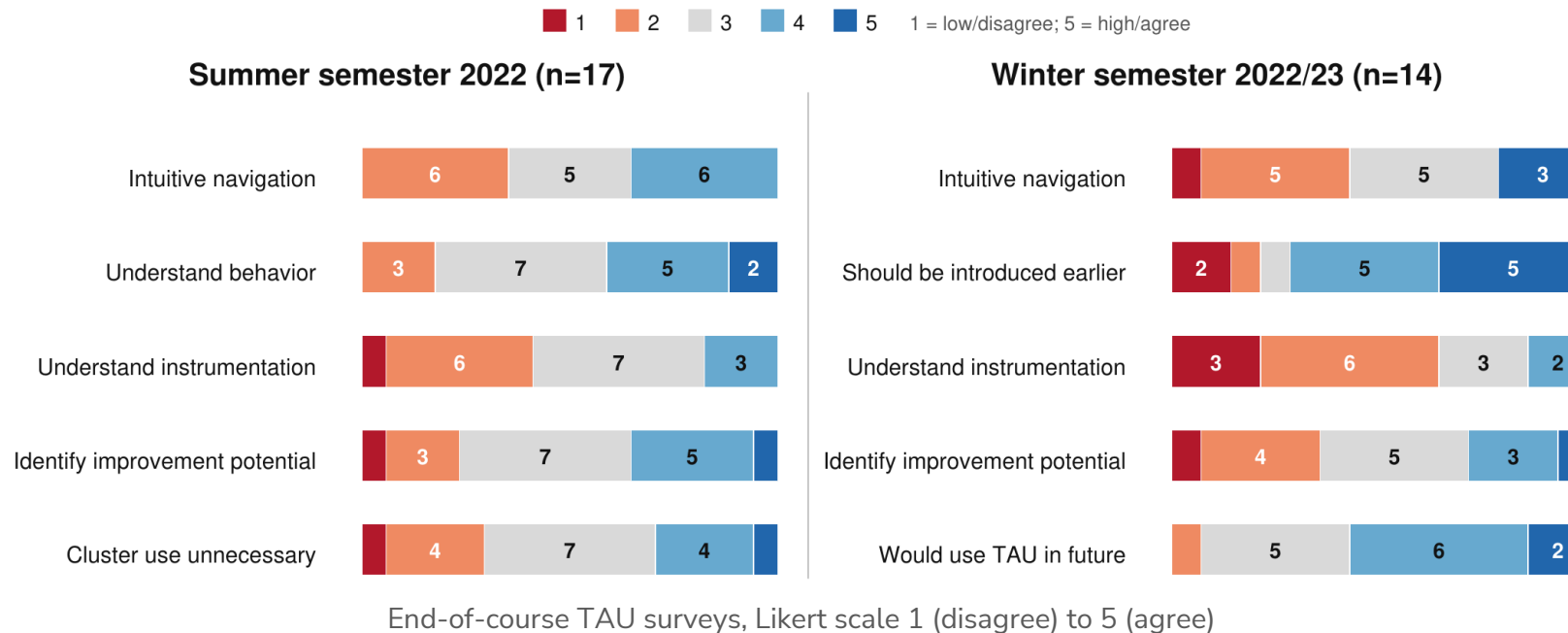
to the first visualization

8

moderator interventions (median)

Conventional workflow (Score-P, Scalasca, CUBE), exploratory pilot study WiSe 2023/24

Late Placement: Interest, But No Competence



10 of 14

wanted performance analysis
earlier in the course

3/17 · 2/14

understood the instrumentation
workflow

Late placement generated interest, but left no time to build independent competence.

Four Barriers to Learning MPI Performance Analysis

1 Cluster usage

- SSH and SLURM job scripts
- Few nodes for teaching, or busy research clusters
- Long waiting times before jobs start

2 Analysis workflow

- Post-mortem: results only after the run
- Instrumentation and measurement overhead
- What to look for, and where?
- Hard to interpret

3 Black-box MPI

- How do processes actually communicate?
- Intra-node vs. inter-node
- Collectives are a black box: tools show the call, not the internals

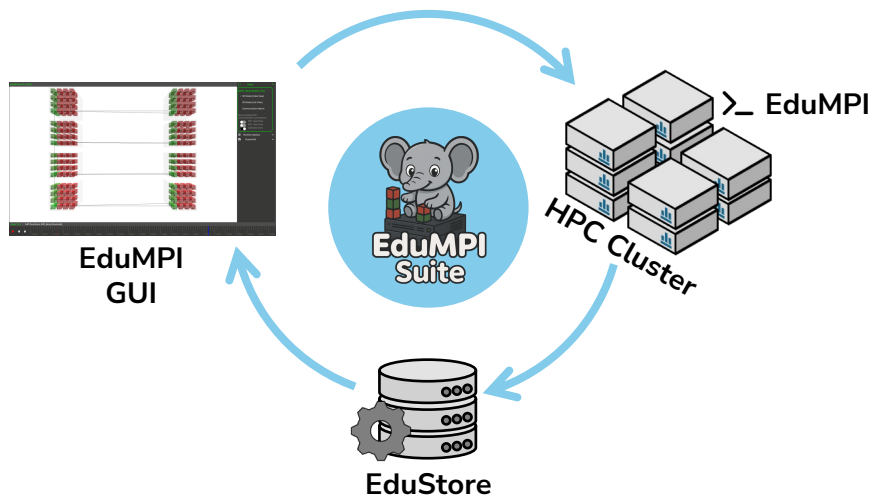
4 Expert-only tools

- Few tools for HPC education
- Score-P, CUBE, TAU, Vampir: complex, assume prior knowledge
- Overwhelming for beginners

In a last-week exercise, all four barriers hit students at once, on top of learning MPI itself.

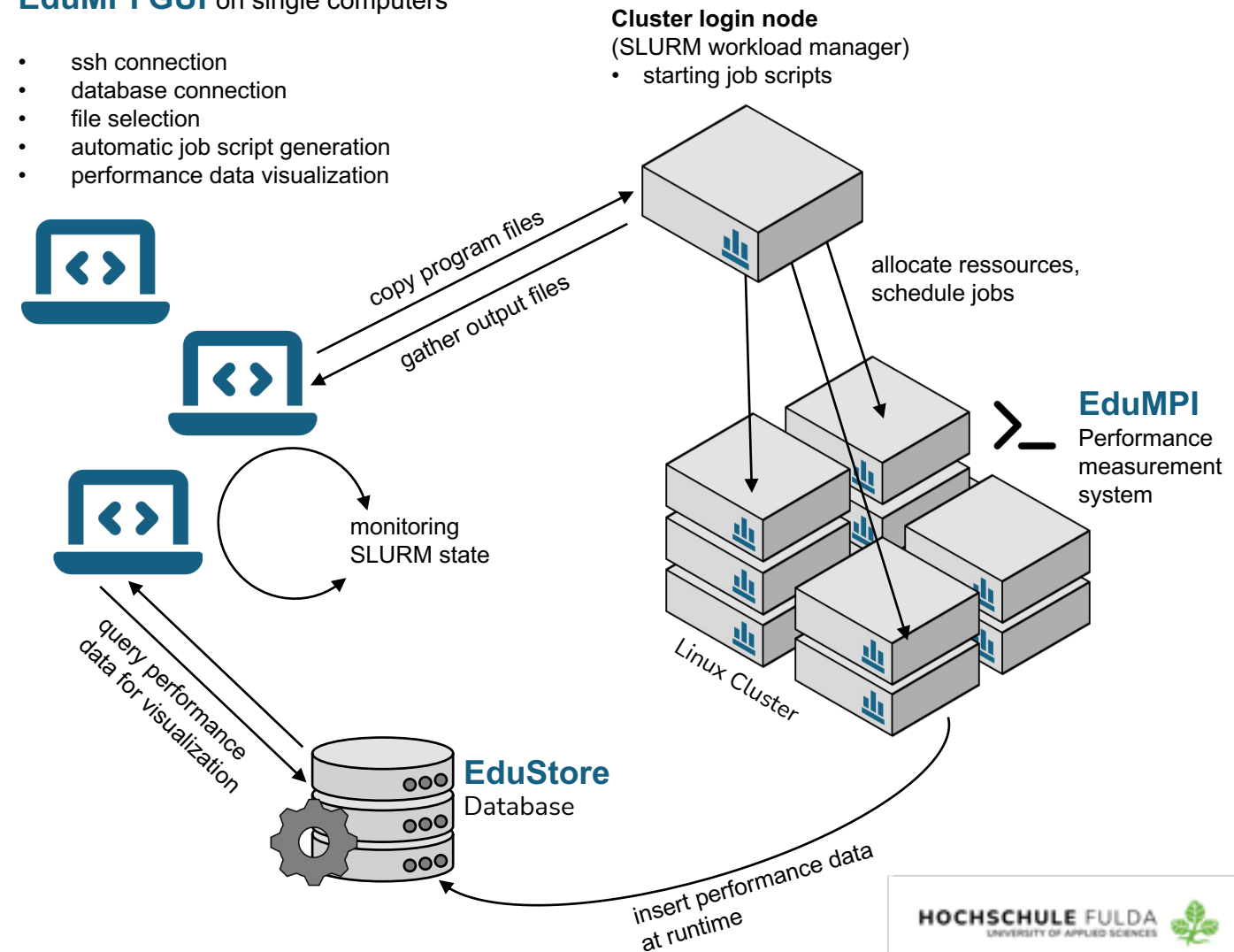
EduMPI Suite

- **EduMPI** – Performance measurement and data collection within an Open MPI fork
- **EduStore** - Time-series database based on:
 - PostgreSQL
 - Timescale DB
- **EduMPI GUI**



EduMPI GUI on single computers

- ssh connection
- database connection
- file selection
- automatic job script generation
- performance data visualization



EduMPI GUI - Operation

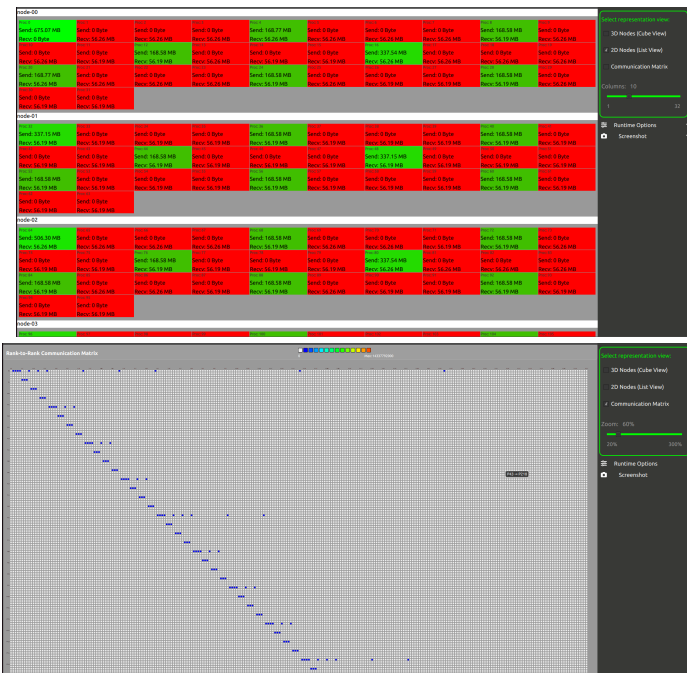
Cluster connection

EduStore connection

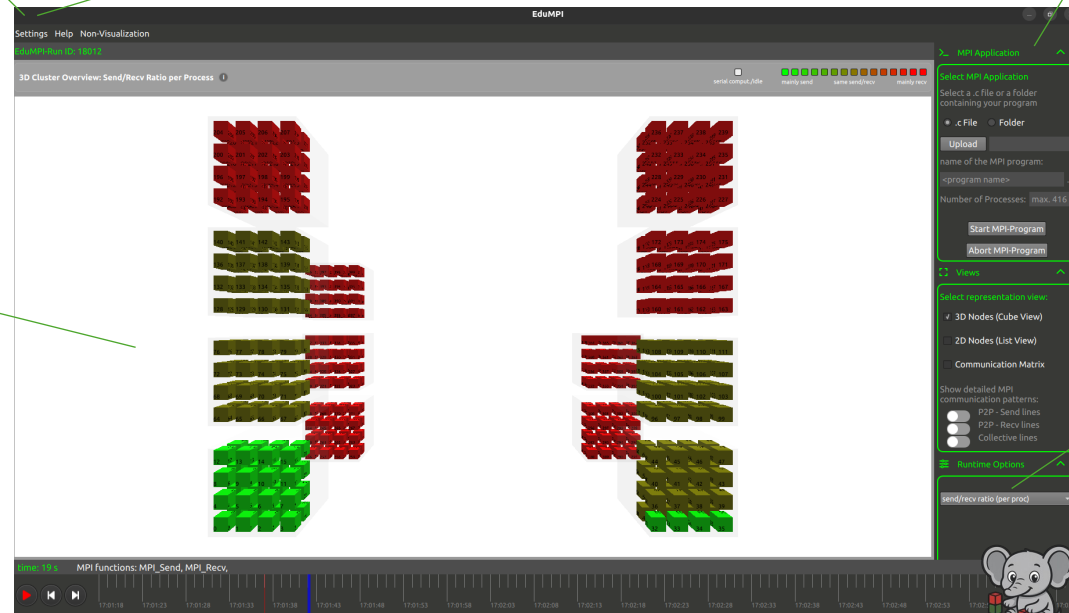
Establish connection to cluster and database

EduMPI Runs		
Job ID: 12321	broadcast_coll.c	Start-Timestamp: 22.06.2025 17:11:38
Job ID: 12322	broadcast_p2p.c	Start-Timestamp: 22.06.2025 17:19:09
Job ID: 12323	broadcast_coll.c	Start-Timestamp: 22.06.2025 17:30:55
Job ID: 12324	broadcast_coll.c	Start-Timestamp: 22.06.2025 17:32:28
Job ID: 12325	broadcast_coll.c	Start-Timestamp: 22.06.2025 17:33:40
Job ID: 12327	matrix_mult.c	Start-Timestamp: 23.06.2025 10:10:12
Job ID: 12604	jacobi.c	Start-Timestamp: 25.06.2025 14:42:14
Job ID: 12605	jacobi.c	Start-Timestamp: 25.06.2025 14:52:11
Job ID: 12606	jacobi.c	Start-Timestamp: 25.06.2025 15:16:54
Job ID: 12706	matrix_mult.c	Start-Timestamp: 27.06.2025 11:21:26

Load previous EduMPI runs from the database



Views – 3D / 2D / communication matrix



MPI Application

Select MPI Application
Select a .c file or a folder containing your program

☐ .c File ☐ Folder

Upload

name of the MPI program:
<program name> .c

Number of Processes: max. 416

Start MPI-Program

Cancel MPI-Program

Start an EduMPI program

Runtime Options

send/recv ratio (per proc)

max send ratio (across all pro...
max recv ratio (across all procs)
wait for late sender (per proc)
wait for late receiver (per proc)

Send Recv

Communication-Type:

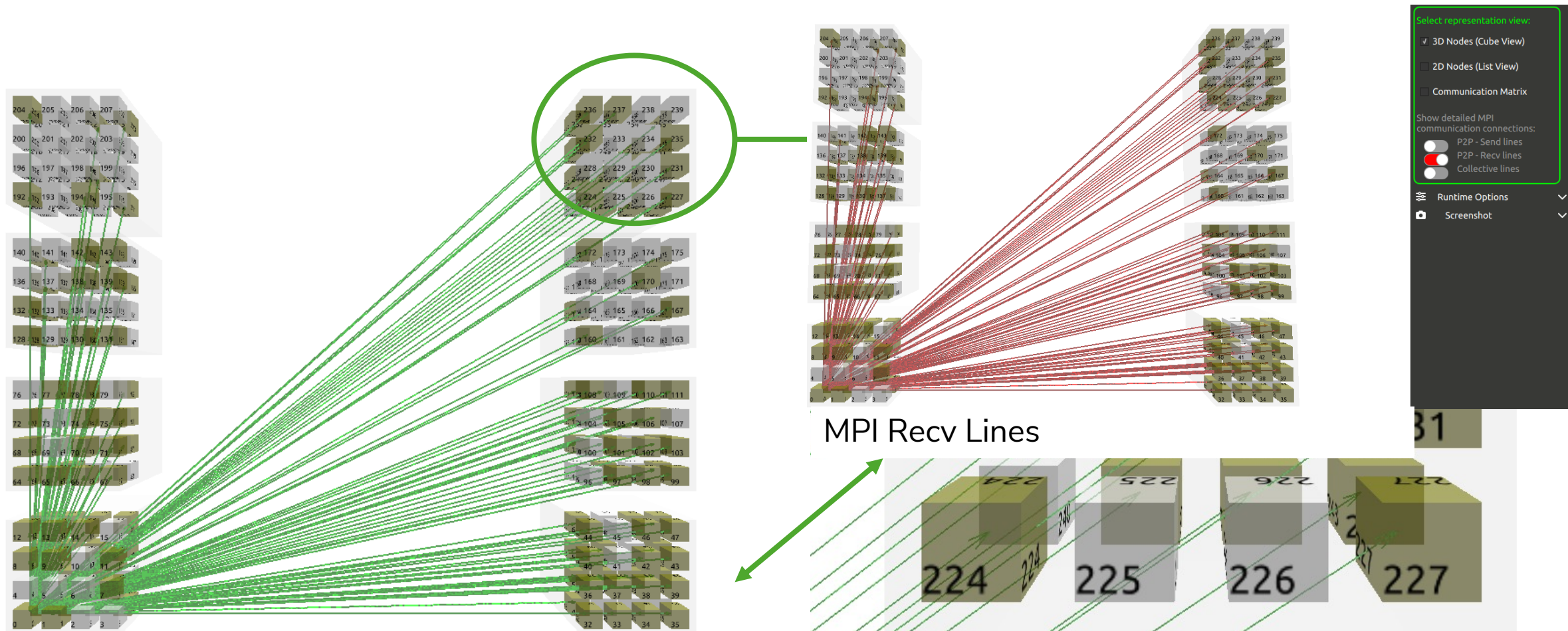
☒ P2P
☒ Collective
☒ One-Sided

Time display:

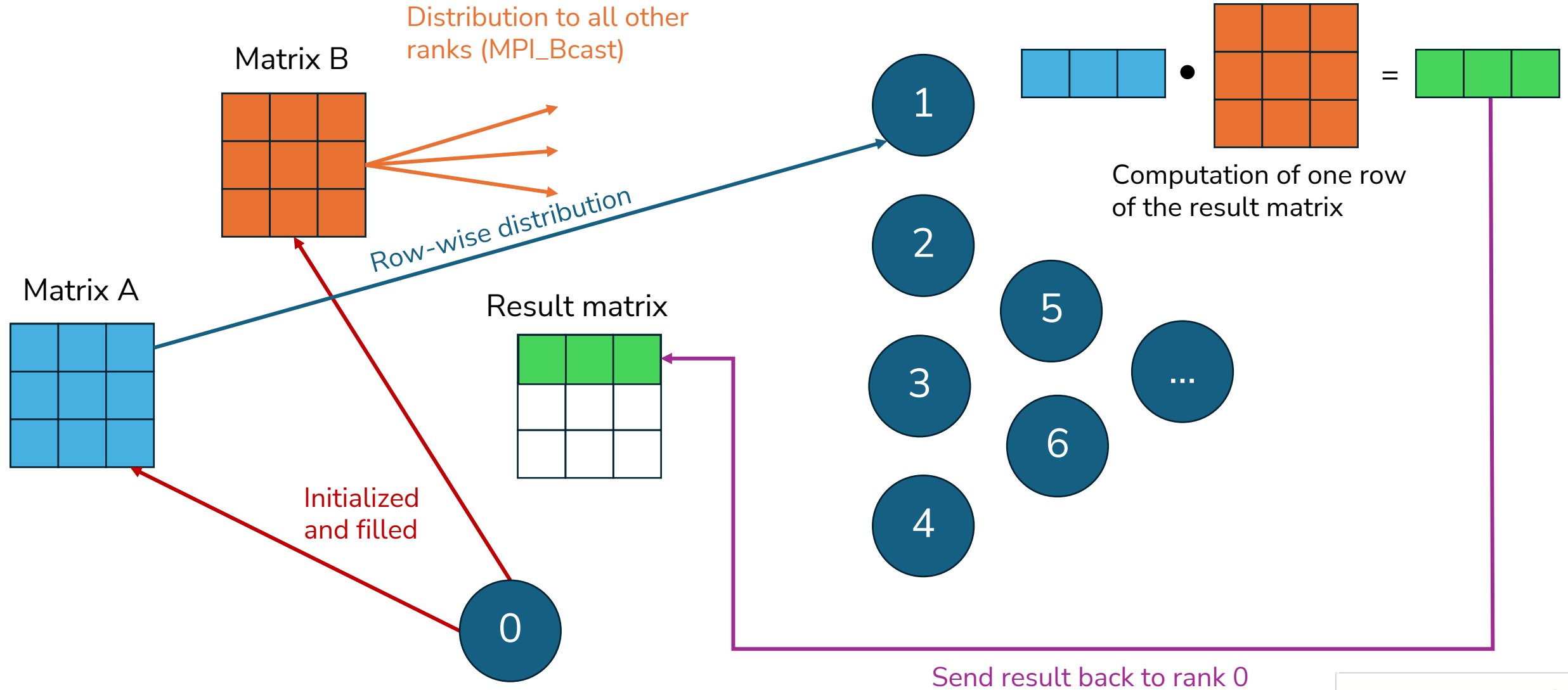
☒ per second
☐ total data

Display options

EduMPI GUI – Send/Recv Communication Arrows



Example Program: Row-wise Matrix Multiplication





MPI Application

Select MPI Application

Select a .c file or a folder containing your program

☒ .c File ☐ Folder

Upload

name of the MPI program:

.c

Number of Processes:

Start MPI-Program

Cancel MPI-Program

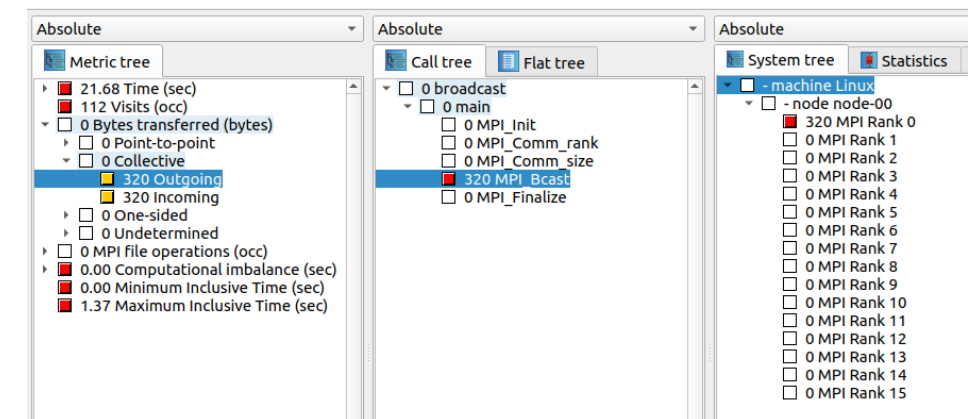
Views

Runtime Options

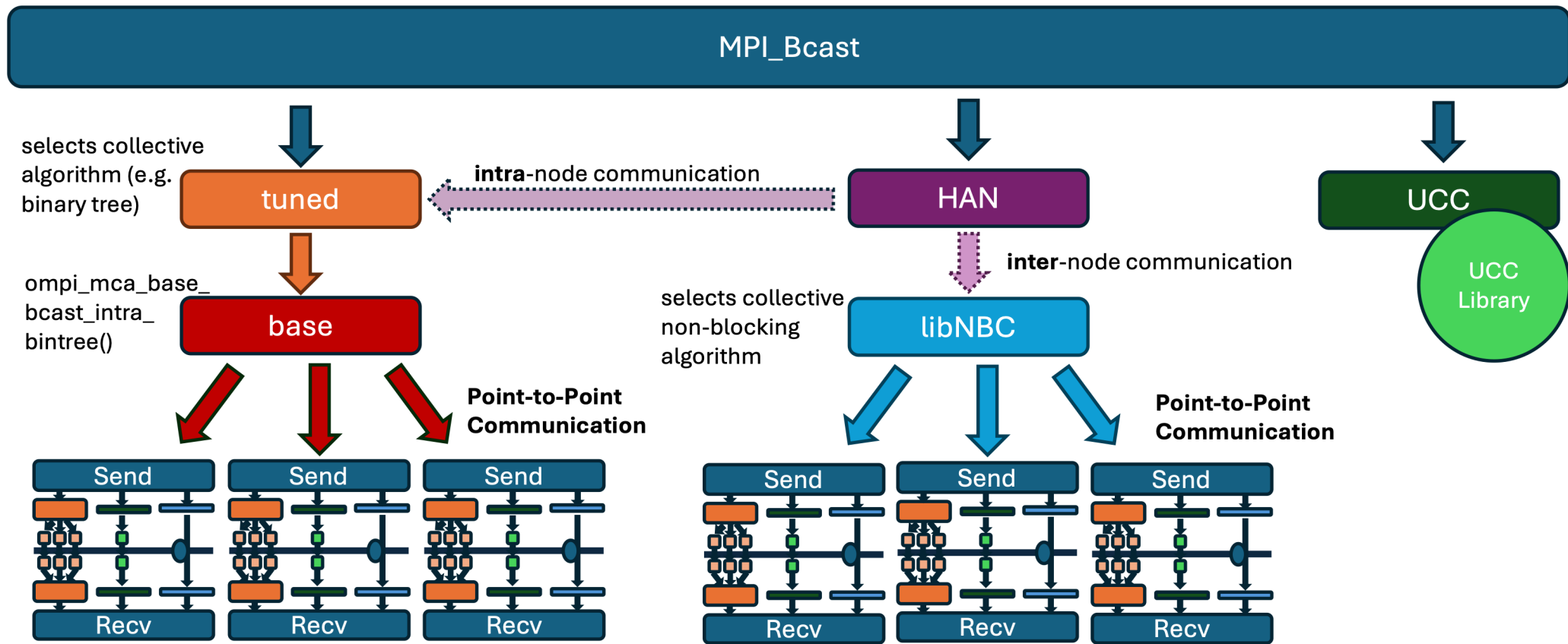
Screenshot

Collective Communication: a Black Box in Performance Analysis Tools

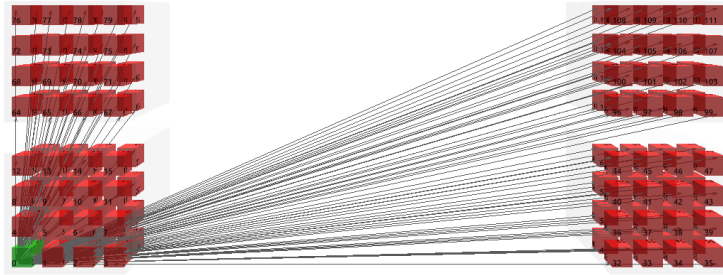
- MPI_Bcast: what does the function do vs. how does it do it?
- Open MPI uses different broadcast algorithms, selected according to fixed criteria.
- Performance analysis tools show only the collective function call, not the internal MPI communication.
- The efficiency of an algorithm depends on the system (architecture, topology, network) → The default is not always optimal.
- Challenge for HPC education:
 - Students often reimplement collective algorithms with point-to-point operations.
 - Misconception: both do exactly the same thing.
- Goal in EduMPI Suite: make internal collective algorithms *visible* and *intuitively understandable*.



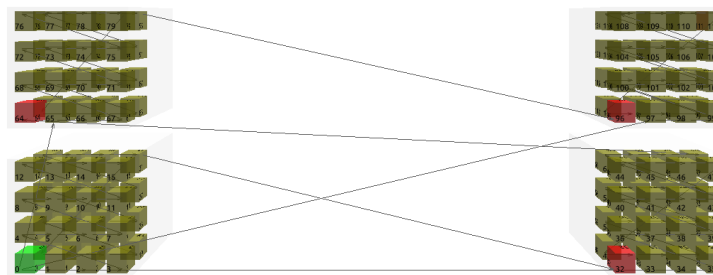
Open MPI – COLL Framework (COLL)



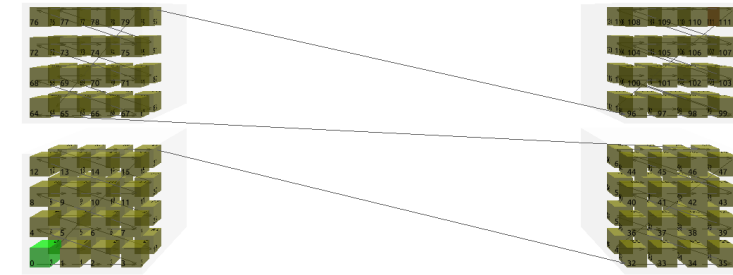
Collective Communication in EduMPI GUI – MPI_Bcast Algorithms



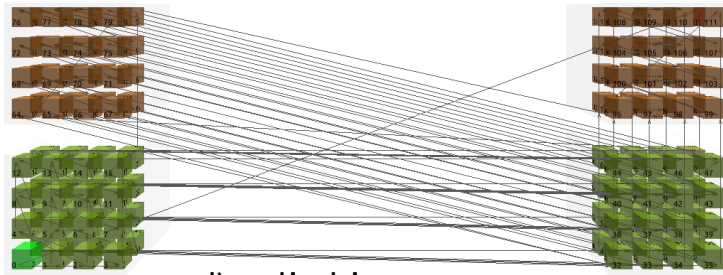
a) basic_linear



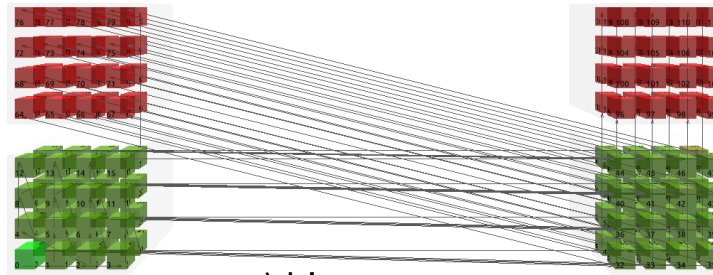
b) chain



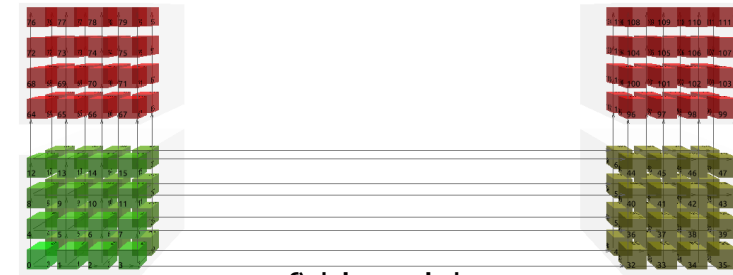
c) pipeline



d) split_binary_tree



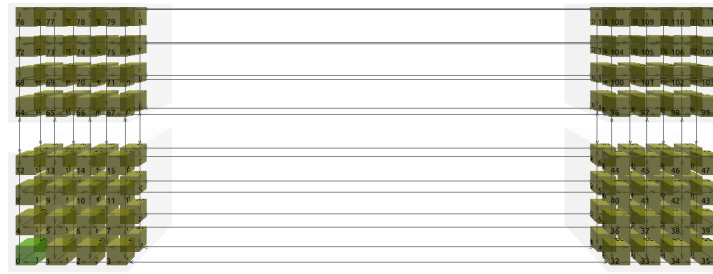
e) binary_tree



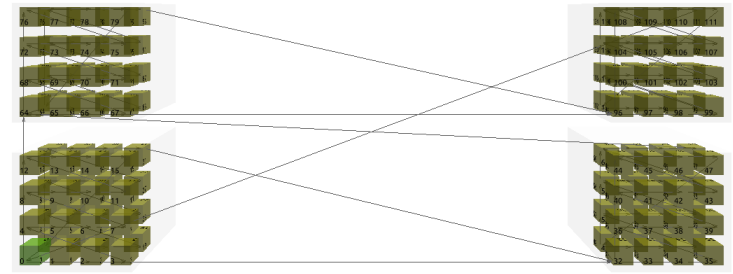
f) binomial



g) knomial



h) scatter_allgather



i) scatter_allgather_ring

EduMPI and EduStore – Performance Measurements and Data Processing

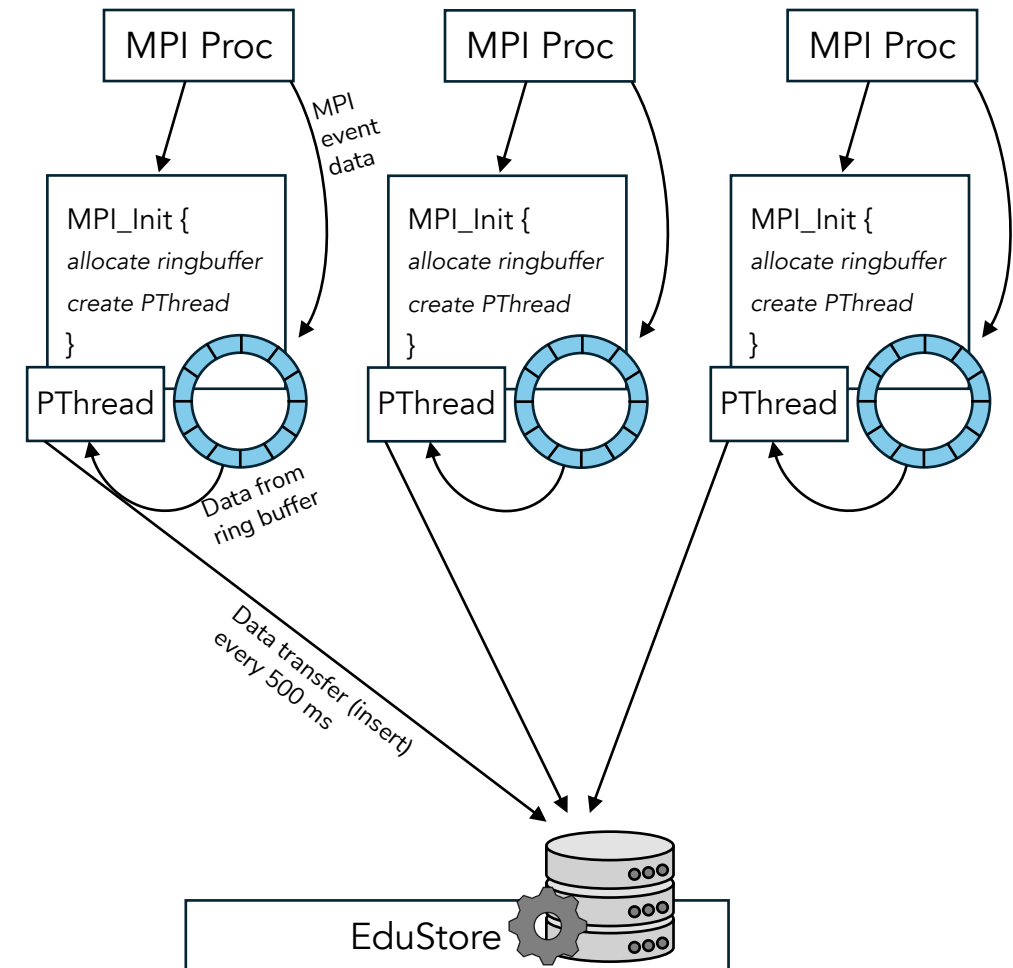
EduMPI: performance measurement at runtime

- Open MPI fork – extended with measurement functionality, captures all MPI communication events
- Each process temporarily stores events in individual ring buffers
- For each process, a Pthread transfers the data from the ring buffer into the database (every 500 ms)

EduStore: data management

- Time-series database based on PostgreSQL and TimescaleDB
- Centralized storage of the MPI events of all processes
- Binary COPY allows very fast data storage (high throughput)
- Continuous data aggregation provides aggregated data per second

→ Overhead stays below 3.8% in benchmarks



EduMPI Measurement Functionality within the Open MPI Source Code

- create item pointer to ring buffer element
- MCA_PML_CALL (point-to-point operation)
- dispatch to ob1 PML
- request handling
- eager/rendezvous decision
- check available BTLs between processes
- select BTL
- call TCP BTL send

```

MPI_Ssend(...)      rb_element *item = getPointerRingPos();

                    rc = MCA_PML_CALL(send(buf, count, type, dest, tag,
                    MCA_PML_BASE_SEND_SYNCHRONOUS, comm, item));

PML                mca_pml.pml_send(buf, count, type, dest, tag,
                    MCA_PML_BASE_SEND_SYNCHRONOUS, comm, item);

ob1                mca_pml_ob1_send(buf, count, type, dest, tag,
                    MCA_PML_BASE_SEND_SYNCHRONOUS, comm, item);

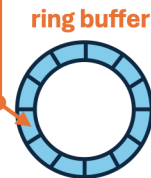
Request handling   MCA_PML_OB1_SEND_REQUEST_INIT(sendreq, buf, count, datatype,
                    dst, tag, comm, sendmode, false, ob1_proc);

                    if(item!=NULL) sendreq->q = item;

BML                rc = mca_bml_base_send(bml_btl, des, hdr->hdr_common.hdr_type, item);

BTL                rc = btl->btl_send(btl, bml_btl->btl_endpoint, des, tag, item);

TCP                mca_btl_tcp_send(btl, bml_btl->btl_endpoint, des, tag, item);
  
```



Profiling Interface (PMPI) and Tool Information Interface (MPI_T)

- **PMPI** (Profiling MPI Interface)

- Intercepts MPI calls via wrapper functions
- Used for runtime behavior, debugging and performance monitoring
- Provides full control over arguments and the execution of MPI functions
- MPI internals are a black box

```
1 int MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest,  
2 int tag, MPI_Comm comm) {  
3     double start = MPI_Wtime();  
4  
5     log("Sending", count, datatype, dest);  
6  
7     int ret = PMPI_Send(buf, count, datatype, dest, tag, comm);  
8  
9     double end = MPI_Wtime();  
10    log_duration(start, end);  
11  
12    return ret;  
13 }
```

- **MPI_T** (MPI Tool Information Interface)

- Provides access to internal performance and control variables
- Non-intrusive, standardized interface, but PVARs and CVARs are implementation-specific
- Suitable for long-term metrics (e.g. buffer utilization, message rate)
- Not yet mature: only partially integrated in many MPI implementations

```
1 MPI_Init_thread(&argc, &argv, MPI_THREAD_SINGLE, &provided);  
2  
3 MPI_T_init_thread(MPI_THREAD_SINGLE, &provided);  
4  
5 int pvar_index, count;  
6 MPI_T_pvar_handle handle;  
7 MPI_T_pvar_session session;  
8  
9 MPI_T_pvar_session_create(&session);  
10  
11 MPI_T_pvar_get_index("bytes_sent", MPI_T_PVAR_CLASS_COUNTER, &pvar_index);  
12  
13 MPI_T_pvar_handle_alloc(session, pvar_index, MPI_COMM_WORLD, &handle, &count);  
14  
15 MPI_T_pvar_start(session, handle);  
16  
17 // ... MPI Communication ...  
18  
19 MPI_T_pvar_read(session, handle, &value);  
20  
21 MPI_T_pvar_stop(session, handle);  
22 MPI_T_pvar_handle_free(session, &handle);  
23 MPI_T_pvar_session_free(&session);  
24 MPI_T_finalize();
```

Profiling Interface (PMPI) and Tool Information Interface (MPI_T) - My Idea

- Read the MPI_T performance variable (PVAR) before and after collective operations to obtain the number and size of the exchanged messages:



- Implementation of a custom PVAR to store the name of the collective algorithm that is used.
- Approach works for blocking communication, but not for non-blocking communication
 - For this, the PVAR would have to be bound to the request (MPI_T_BIND_REQUEST)
 - Request lifetime problem: non-blocking requests disappear after they have been used (termination of the operation), so the PVAR data assigned to them can no longer be accessed.
 - To fix this problem, MPI_T sessions and handles would have to persist beyond the lifetime of the request and only be deleted explicitly when the handle is freed.

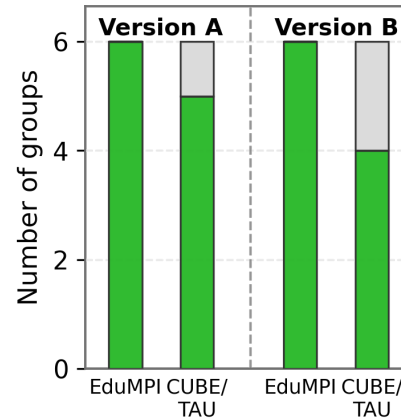
Extensive Comparative Usability Study (SS 2025) - Tasks

- T1 MPI process distribution:** How and across how many nodes?
- T2 Color coding:** What do the colors mean in the respective tools?
- T3 Matrix B distribution:** Duration and MPI function(s) used?
- T4 First phase (broadcast):** Identify the bottleneck and the reasons for it?
- T5 Second phase (row-wise distribution of matrix A):** Problems and reasons?
- T6 Centralized vs. distributed communication:** Efficiency and scalability?
- T7 Dominant MPI communication in the entire program:** Which MPI function takes the most time and transfers the most data

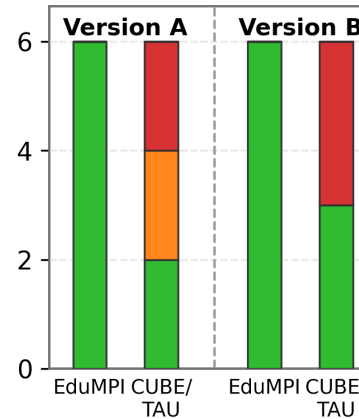
Usability Study (SS 2025) – Results (Effectiveness)

- EduMPI Suite:
96% correct
- CUBE/TAU:
60% correct
- Version A (EduMPI Suite first)
subsequently achieves
better results with
CUBE/TAU

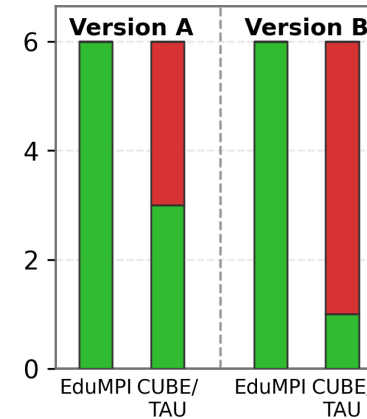
T1: MPI process distribution



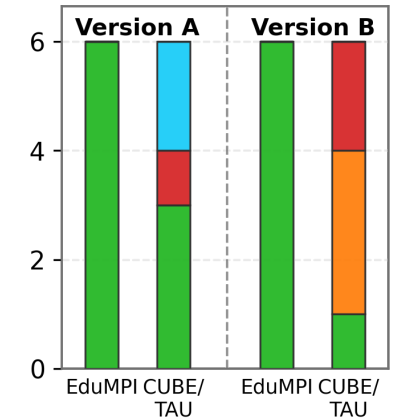
T2: Color coding



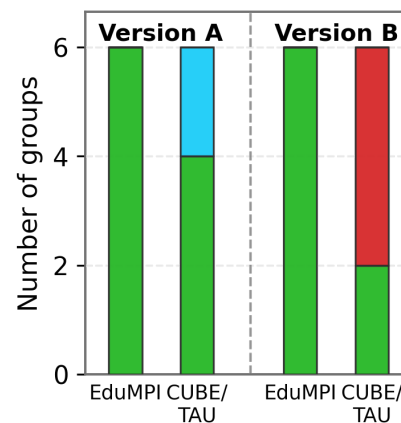
T3: Matrix B distribution



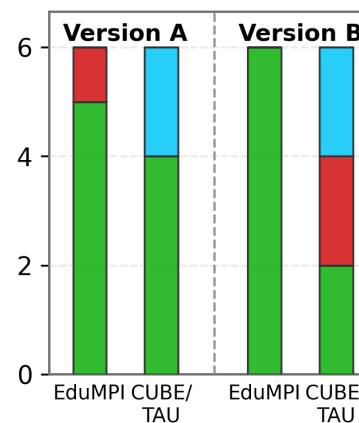
T4: Broadcast (bottleneck)



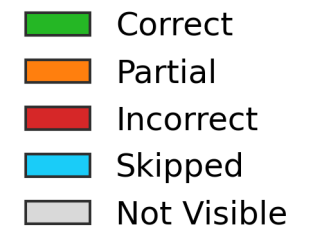
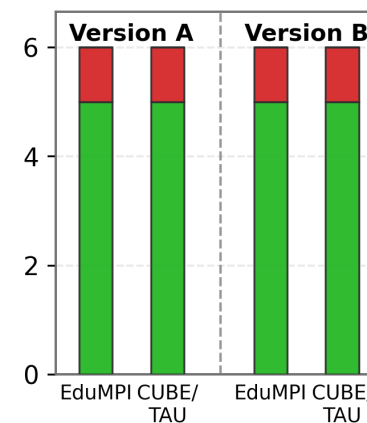
T5: Row-wise distribution of matrix A



T6: Centralized vs. distributed communication



T7: Dominant MPI communication

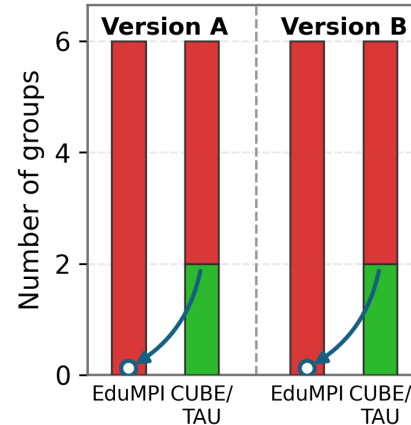


Version A:
EduMPI Suite → CUBE/TAU
Version B:
CUBE/TAU → EduMPI Suite

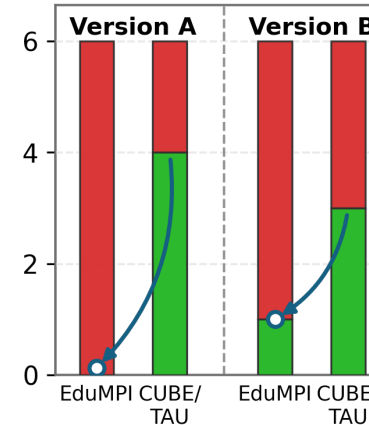
Usability Study (SS 2025) – Results (Efficiency)

- EduMPI Suite: **29%** assistance
- CUBE/TAU: **55%** assistance
- Most frequent assistance: T4
 - EduMPI Suite: **67%**
 - CUBE/TAU: **83%**
 - Motivation to try out different views
- Exception: T7:
 - Students need help more often with EduMPI Suite than with CUBE/TAU

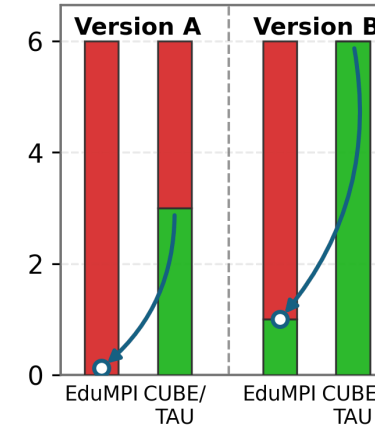
T1: MPI process distribution



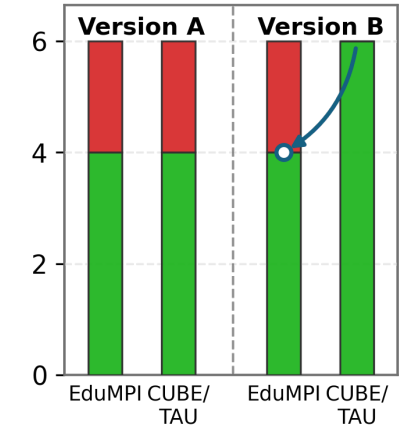
T2: Color coding



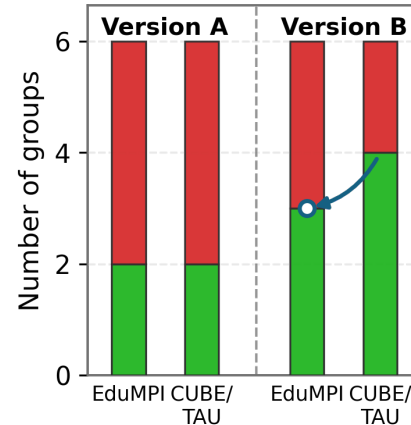
T3: Matrix B distribution



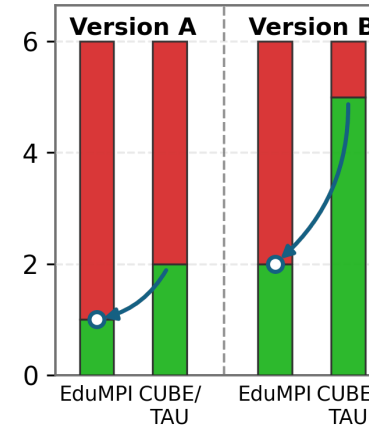
T4: Broadcast (bottleneck)



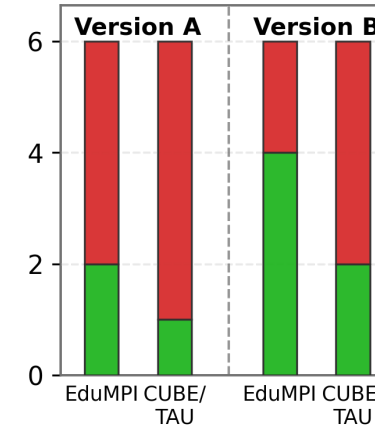
T5: Row-wise distribution of matrix A



T6: Centralized vs. distributed communication



T7: Dominant MPI communication

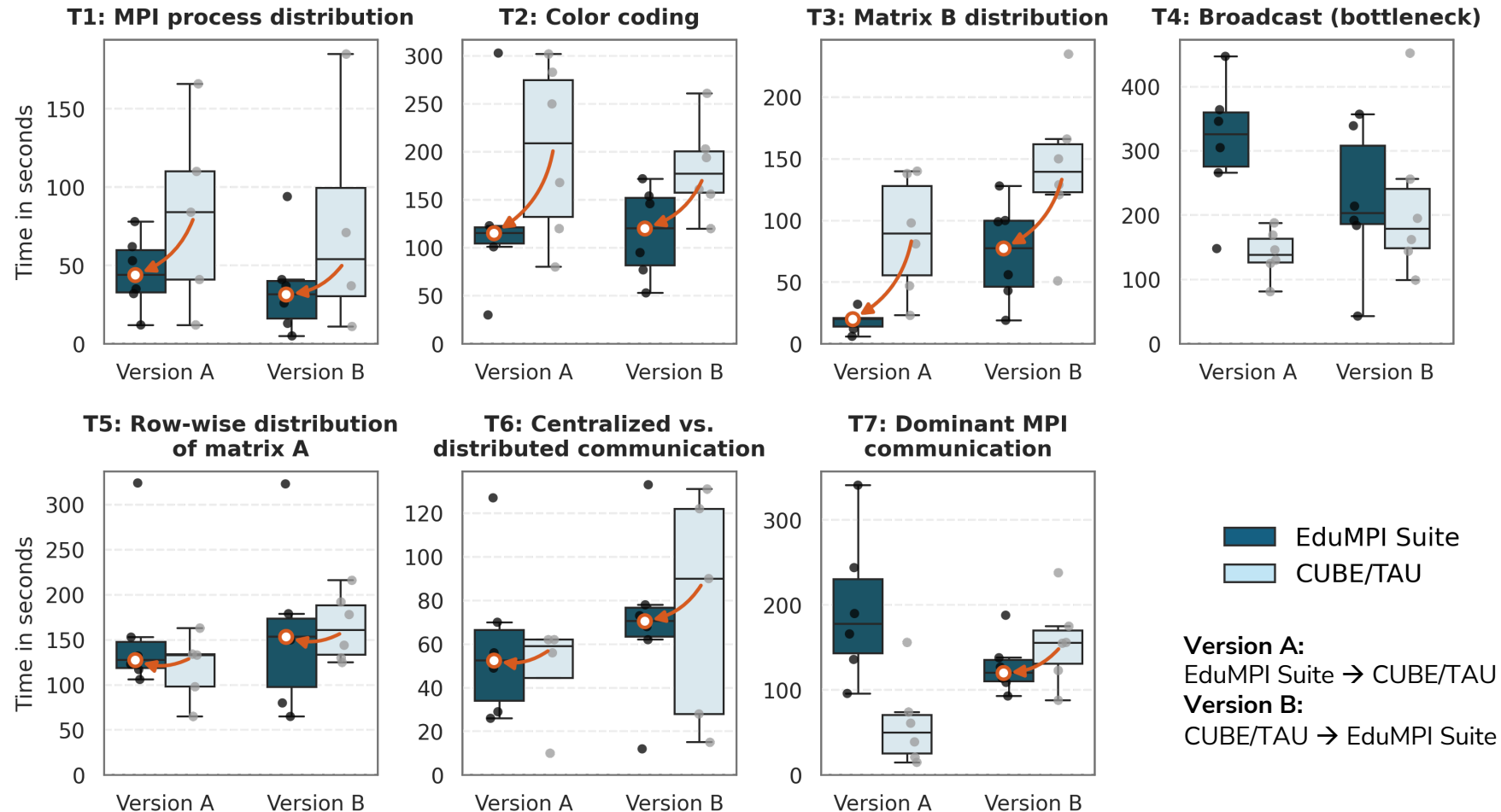


■ Assistance needed
■ No assistance
○ EduMPI needs less assistance

Version A:
 EduMPI Suite → CUBE/TAU
Version B:
 CUBE/TAU → EduMPI Suite

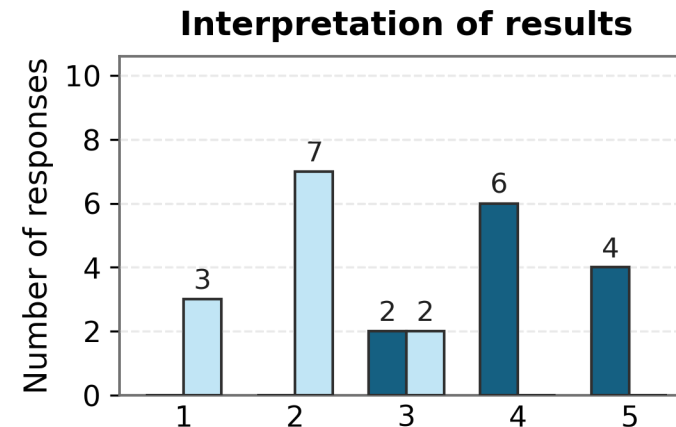
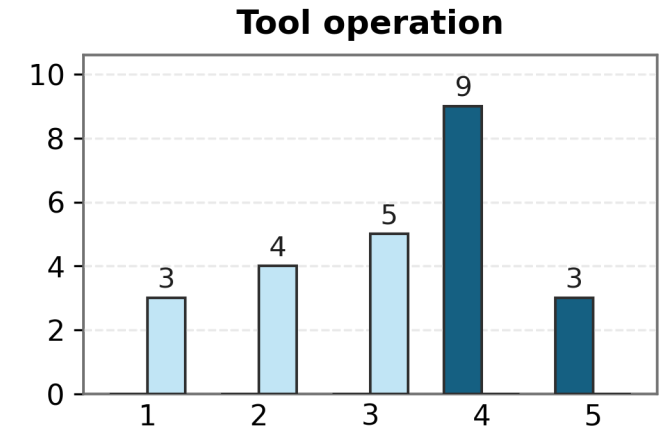
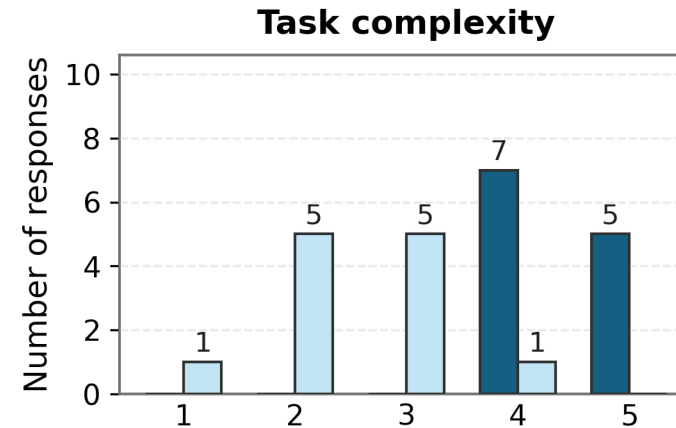
Usability Study (SS 2025) – Results (Efficiency)

- Faster completion of the tasks with EduMPI Suite
- Exception: T4 and T7 (version B)
 - T4 was answered correctly far more often with EduMPI Suite
 - T7 shows that there are some difficulties in EduMPI Suite in gaining an overall overview (profile) of the performance data.



Usability Study (SS 2025) – Results (User Satisfaction)

- EduMPI Suite is predominantly rated as **easy to very easy** in all categories
- CUBE/TAU is predominantly rated as **very difficult to moderate** in all categories
- All 33 participants recommend EduMPI Suite for use in teaching (1 of them with improvements to the tool)
- 15 participants recommend CUBE/TAU for use in teaching (14 of them with improvements to the tool)



■ EduMPI Suite
■ CUBE/TAU

Rating (1 = very difficult, 3 = moderate, 5 = very easy)



Summary

Students understand MPI when they can see it communicate, from the first exercise on.

- Performance analysis from the first day
 - A laptop allows no realistic MPI runs, a last-week add-on leaves no time. Cluster and tools belong in the first MPI exercise.
- Accessible performance analysis
 - One GUI to run, monitor and analyze MPI programs in near-real-time.
- Collective algorithms become visible
 - Communication matrices, arrows and send/receive ratios reveal which algorithm MPI_Bcast runs, and where processes wait.
- The MPI standard could open the black box
 - PMPI sees only the call. MPI_T is not mature yet: PVARs and CVARs are implementation-specific and often barely integrated.

COMPARATIVE USABILITY STUDY

96% vs. 60%

correct answers with EduMPI Suite vs. CUBE/TAU.

32 of 33

recommend EduMPI Suite for teaching without reservation (CUBE/TAU: 1 of 33).

What happens in MPI shouldn't stay in MPI.

What Academia and the MPI Standard Could Contribute

WHAT ACADEMIA CAN DO

Tangible tools

Performance analysis and HPC tools built for learners: visual, interactive, low barrier to entry.

EXAMPLE

EduMPI Suite: MPI communication visualized in near-real-time

Shared curricula

Performance analysis as a core competence, and more exchange of courses and material.

EXAMPLE

EUMaster4HPC: 9 universities, 43 partners, specialisation “Performance Analyst and Advisor”

Real clusters for students

Reserved partitions or shared teaching clusters, so students run MPI where communication matters.

EXAMPLE

bwUniCluster 3.0 (KIT): shared teaching resource for universities and UAS in Baden-Württemberg

WHAT THE STANDARD CAN DO

Open the black box for tools

PMPI

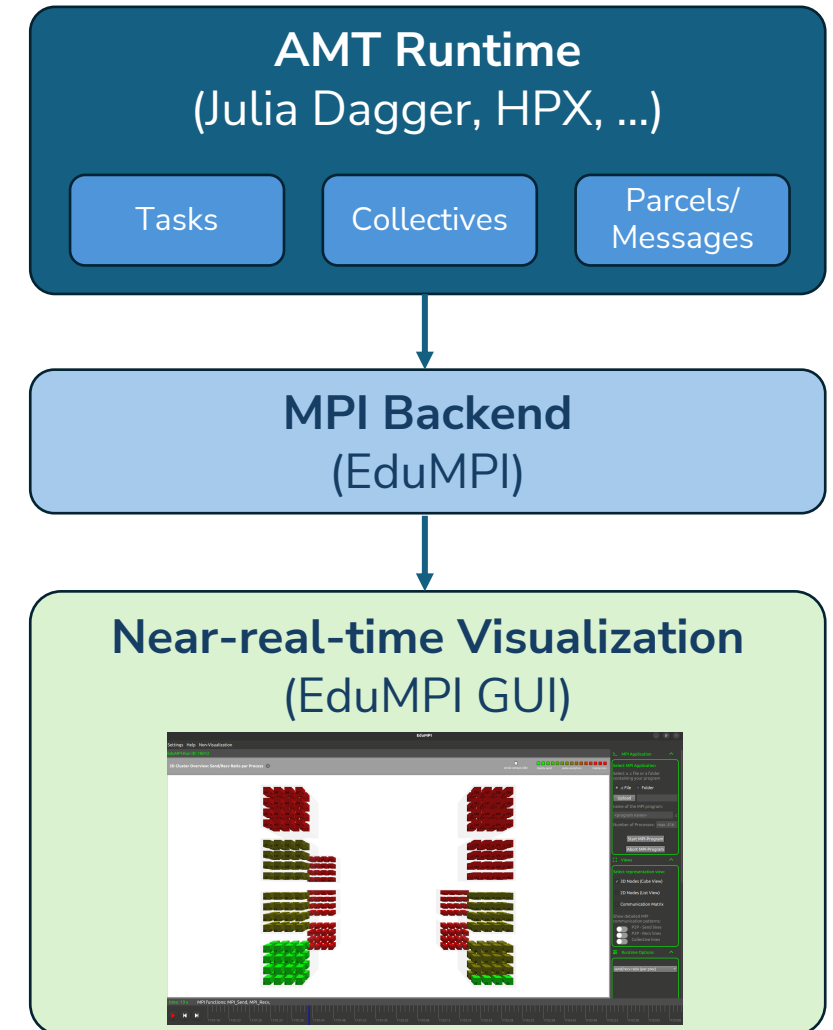
- Several tools at once (today: one PMPI layer; cf. QMPI proposal)
- Link a call to the internal messages it triggers, e.g. collective → point-to-point

MPI_T

- Common variables and events, incl. the selected collective algorithm
- Data that outlives non-blocking requests
- Documented, consistent support across implementations

Future Work – EduMPI Suite for AMT Runtime Observability

- EduMPI Suite instruments the MPI layer and visualizes MPI communication in near-real-time.
- Asynchronous Many-Task (AMT) runtimes such as HPX and Julia Dagger can use MPI as a communication backend.
 - This allows us to observe the actual MPI traffic generated by the runtime.
- EduMPI Suite can reveal communication topology, message volume, message sizes, timing, and synchronization behavior.
 - This helps identify hotspots, imbalances, excessive fine-grained communication, and inefficient communication patterns.
- Comparison with native MPI programs can help determine whether overhead originates in the MPI transport layer or higher-level runtime mechanisms.



EduMPI Suite – Links and Publications



EduMPI GUI



EduMPI + EduStore

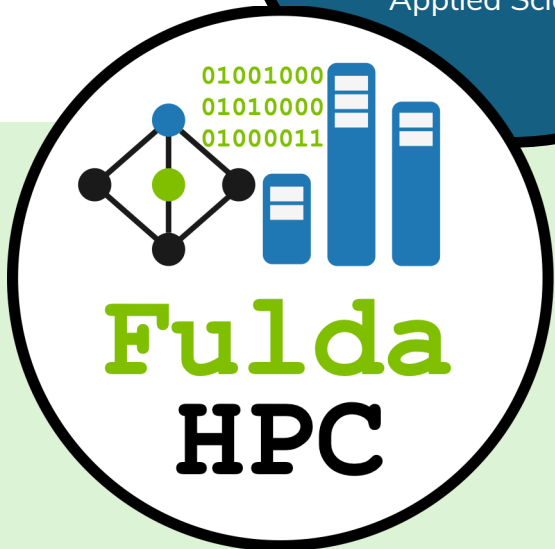
1. [Roth, Anna-Lena](#); Süß, Tim (2023): **Performance Analysis Tools for MPI Applications and their Use in Programming Education**. In Companion of the International Conference on Performance Engineering (ICPE '23 Companion). ACM, New York, NY, USA, 361–368. <https://doi.org/10.1145/3578245.3584358>
2. [Roth, Anna-Lena](#); James, David; Kuhn, Michael; Konert, Johannes (2024): **Enhancing Parallel Programming Education with High-Performance Clusters Utilizing Performance Analysis**. Proceedings of DELFI 2024. Gesellschaft für Informatik e.V. https://doi.org/10.18420/delfi2024_42
3. [Roth, Anna-Lena](#); James, David; Kuhn, Michael (2025): **EduMPI – Simplifying the Use of High-Performance Clusters and Focusing Performance Analysis in Parallel Programming Education**. PARS-Mitteilungen: Vol. 37. Gesellschaft für Informatik e.V., Fachgruppe PARS. ISSN: 0177-0454 <https://dl.gi.de/handle/20.500.12116/46131>
4. [Roth, Anna-Lena](#); James, David; Kuhn, Michael; Frisch, Dustin (2025): **Making MPI Collective Operations Visible: Understanding Their Utility and Algorithmic Insights**. Euro-Par 2025. LNCS, vol 15900. Springer, Cham. https://doi.org/10.1007/978-3-031-99854-6_5
5. [Roth, Anna-Lena](#); James, David; Posner, Jonas; Kuhn, Michael (2026): **Performance Analysis in Parallel Programming Education: A Comparative Usability Study**. PARS-Mitteilungen: Vol. 38. Gesellschaft für Informatik e.V., Fachgruppe PARS. ISSN: 0177-0454 <https://dl.gi.de/handle/20.500.12116/46131>
6. [Roth, Anna-Lena](#); Posner, Jonas; Kuhn, Michael (2026): **Enabling Interactive Visualization for Near-Real-Time Performance Analysis of MPI Applications**. In Companion of the International Conference on Performance Engineering (ICPE '26 Companion). ACM, New York, NY, USA, 120 – 129. <https://dl.gi.de/handle/20.500.12116/48841>
7. [Roth, Anna-Lena](#); Posner, Jonas (2026): **An Empirical Analysis of High-Performance Computing Education in Germany**. In Computational Science (ICCS) Workshops (WTCS). LNCS, vol 16789. Springer, Cham. https://doi.org/10.1007/978-3-032-29918-5_32
8. [Roth, Anna-Lena](#); Posner, Jonas (2026): **From a Last-Week Add-On to First-Day Practice: Three Years of Integrating MPI Performance Analysis into HPC Education with EduMPI Suite**. EduHPC, SC '26 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. (to appear). <https://doi.org/10.48550/arXiv.2610.06352>
9. [Roth, Anna-Lena](#); Posner, Jonas (2026): **Near-Real-Time Performance Visualization of MPI Applications: Integrating Performance Analysis into HPC Education** (Poster). (to appear). <https://doi.org/10.5281/zenodo.23059851>



Dr. Anna-Lena Roth

Postdoctoral Researcher
anna-lena.roth@cs.hs-fulda.de
www.annalenaroth.com

Fulda University of
Applied Sciences



Thank you for your attention!

Questions & discussion



annalenaroth.com



EduMPI + EduStore



EduMPI GUI