MPI-BugBench: A Framework for Assessing MPI Correctness Tools

Tim Jammer[1], Emmanuelle Saillard[3], Simon Schwitanski[2],
Alexander Hück[1], Joachim Jenke[2], Radjasouria Vinayagame[3] and Christian Bischof[1]
[1]Technical University Darmstadt & [2]RWTH Aachen University & [3] Inria of the University of Bordeaux

tim.jammer@tu-darmstadt.de

https://git-ce.rwth-aachen.de/hpc-public/mpi-bugbench

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

- MPI's API is error prone

⇒ Correctness checking tools are important

- Tools are hard to compare
- Real world applicability not sufficiently evaluated

⇒ Correctness Benchmark Suite: MPI-BugBench

Previous Work
Unified in MPI-BugBench

MPI-CorrBench (COBE)

- 202 correct, 308 incorrect codes
- hand written codes

MPI Bugs Initiative (MBI)

- 682 correct, 986 incorrect codes
- replacement of placeholder tokens

RMARaceBench (RRB)

- 43 correct, 64 incorrect codes
- Only for RMA Operations

Categories of MPI Errors

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Single call errors:**
  - **Invalid parameters**
- Process-local errors:
  - Resource leak
  - Initialization of MPI
  - Request lifecycle
  - Local concurrency
  - Epoch lifecycle
- Multi-process errors:
  - Message race
  - Parameter matching
  - Call ordering
  - Global concurrency

```
MPI_Send(buf, -1, dtype, dest, tag, comm);
```

Categories of MPI Errors

TECHNISCHE UNIVERSITÄT DARMSTADT

- Single call errors:
  - Invalid parameters
- **Process-local errors:**
  - Resource leak
  - Initialization of MPI
  - **Request lifecycle**
  - Local concurrency
  - Epoch lifecycle
- Multi-process errors:
  - Message race
  - Parameter matching
  - Call ordering
  - Global concurrency

```
MPI_Request req;
MPI_Isend(buf, count, dtype,   dest, tag, comm, &req);
MPI_Irecv(buf, count, dtype, source, tag, comm, &req);
```

Categories of MPI Errors

TECHNISCHE UNIVERSITÄT DARMSTADT

- Single call errors:
  - Invalid parameters
- Process-local errors:
  - Resource leak
  - Initialization of MPI
  - Request lifecycle
  - Local concurrency
  - Epoch lifecycle
- **Multi-process errors:**
  - Message race
  - Parameter matching
  - **Call ordering**
  - Global concurrency

Rank 0:

```
MPI_Barrier(comm);
MPI_Allreduce(/* parameters */);
```

Rank 1:

```
MPI_Allreduce(/* parameters */);
MPI_Barrier(comm);
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  def generate(self, generate_level, real_world_score_table):
2    for func in mpi_send_funcs :
3      for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
4        tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
5        for call in tm.get_instruction(identifier="MPICALL", return_list=True):
6          # send is executed by rank 1 in default template
7          if call.get_rank_executing() == 1:
8            call.set_arg("buf", buf_to_use) # set buffer to invalid value
9            call.set_has_error() # mark where the error is
10       # set an appropriate description for the error:
11       tm.set_description("InvalidParam-Buffer-" + func, # short description
12                          "Invalid Buffer: "+buf_to_use) # long description
13       yield tm
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  def generate(self, generate_level, real_world_score_table):
2    for func in mpi_send_funcs :
3      for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
4        tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
5        for call in tm.get_instruction(identifier="MPICALL", return_list=True):
6          # send is executed by rank 1 in default template
7          if call.get_rank_executing() == 1:
8            call.set_arg("buf", buf_to_use) # set buffer to invalid value
9            call.set_has_error() # mark where the error is
10       # set an appropriate description for the error:
11       tm.set_description("InvalidParam-Buffer-" + func, # short description
12                          "Invalid Buffer: "+buf_to_use) # long description
13       yield tm
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```python
def generate(self, generate_level, real_world_score_table):
  for func in mpi_send_funcs :
    for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
      tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
      for call in tm.get_instruction(identifier="MPICALL", return_list=True):
        # send is executed by rank 1 in default template
        if call.get_rank_executing() == 1:
          call.set_arg("buf", buf_to_use) # set buffer to invalid value
          call.set_has_error() # mark where the error is
      # set an appropriate description for the error:
      tm.set_description("InvalidParam-Buffer-" + func, # short description
                         "Invalid Buffer: "+buf_to_use) # long description
      yield tm
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  def generate(self, generate_level, real_world_score_table):
2    for func in mpi_send_funcs :
3      for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
4        tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
5        for call in tm.get_instruction(identifier="MPICALL", return_list=True):
6          # send is executed by rank 1 in default template
7          if call.get_rank_executing() == 1:
8            call.set_arg("buf", buf_to_use) # set buffer to invalid value
9            call.set_has_error() # mark where the error is
10       # set an appropriate description for the error:
11       tm.set_description("InvalidParam-Buffer-" + func, # short description
12                          "Invalid Buffer: "+buf_to_use) # long description
13       yield tm
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  def generate(self, generate_level, real_world_score_table):
2    for func in mpi_send_funcs :
3      for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
4        tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
5        for call in tm.get_instruction(identifier="MPICALL", return_list=True):
6          # send is executed by rank 1 in default template
7          if call.get_rank_executing() == 1:
8            call.set_arg("buf", buf_to_use) # set buffer to invalid value
9            call.set_has_error() # mark where the error is
10       # set an appropriate description for the error:
11       tm.set_description("InvalidParam-Buffer-" + func, # short description
12                          "Invalid Buffer: "+buf_to_use) # long description
13       yield tm
```

Test Case Generation

Illustration for invalid buffer error

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  def generate(self, generate_level, real_world_score_table):
2    for func in mpi_send_funcs :
3      for buf_to_use in [ "NULL", 'MPI_BOTTOM', 'MPI_IN_PLACE']:
4        tm = get_send_recv_template(func, "mpi_irecv") # get a TemplateManager
5        for call in tm.get_instruction(identifier="MPICALL", return_list=True):
6          # send is executed by rank 1 in default template
7          if call.get_rank_executing() == 1:
8            call.set_arg("buf", buf_to_use) # set buffer to invalid value
9            call.set_has_error() # mark where the error is
10       # set an appropriate description for the error:
11       tm.set_description("InvalidParam-Buffer-" + func, # short description
12                          "Invalid Buffer: "+buf_to_use) # long description
13       yield tm
```

Different Instances of Same Error

TECHNISCHE UNIVERSITÄT DARMSTADT

- Test multiple instances of *same* error
  - E.g.: different datatype missmatches

⇒ Test tool with different usage patterns

! Can lead to high number of test-cases

```
MPI_Send(buf, count, MPI_INT,
         dest, tag, comm);
MPI_Recv(buf, count, MPI_LONG_LONG,
          src, tag, comm, status);
```

```
MPI_Send(buf, count, MPI_INT,
         dest, tag, comm);
MPI_Recv(buf, count, MPI_UNSIGNED,
          src, tag, comm, status);
```

Generation Levels

TECHNISCHE UNIVERSITÄT DARMSTADT

- Test multiple instances of *same* error
  - E.g.: different datatype missmatches

⇒ Test tool with different usage patterns

! Can lead to high number of test-cases

level 1 Basic cases

level 2.1 **Sufficient coverage of real-world patterns**

level 2.2 Sufficient coverage of all patterns

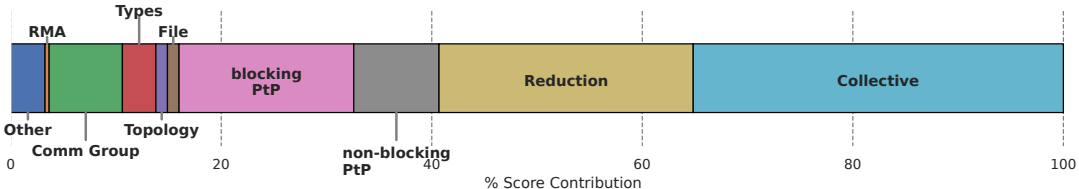
level 3.1 Full coverage of real-world patterns

level 3.2 Full coverage of all patterns

Table: Number of test cases per feature.

	Level 1	**Level 2.1**	Level 2.2	Level 3.1	Level 3.2
Point-to-point	49	**870**	24,116	2,789	3,004,968
Collective	40	**774**	23,937	3,246	1,121,865
One-sided	39	**415**	415	1,898	1,898
Total	128	**2,059**	48,539	7,933	4,128,731

Real-world Usage Patterns

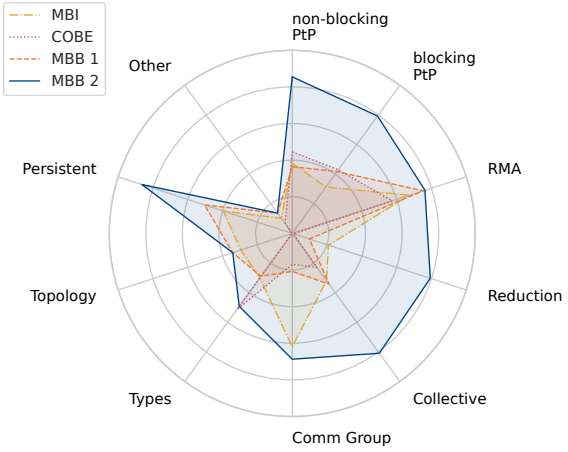


- Usage pattern:
  - MPI call including the relevant parameters
  - `datatype`, reduction `operator`, `count`, matching wildcards, `rank` (collectives only)
- Scoring:
  - Percentage of real world usage patterns covered by the Correctness Benchmark
  - Weighted according to usage frequency

Real-World Applicability

Improvements over previous correctness benchmarks

TECHNISCHE UNIVERSITÄT DARMSTADT



Overall Scores:

COBE	25.61%
MBI	28.32%
MBB lv 1	28.45%
MBB lv 2.1	75.94%

Currently missing:

- MPI I/O
- Fortran

NHR4CES

Correctness Checking Tools

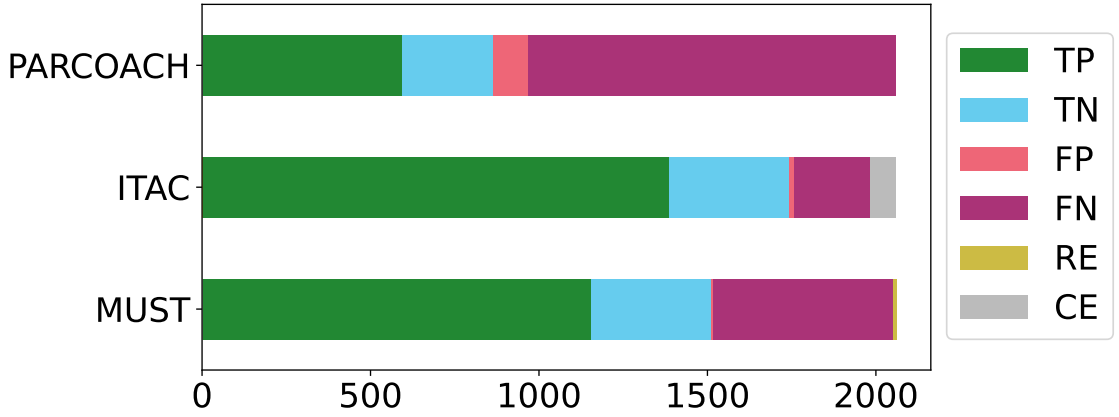
Most relevant state of the art tools

TECHNISCHE UNIVERSITÄT DARMSTADT

- PARCOACH (PARallel COntrol flow Anomaly CHecker)
  - Static tool (performs static analysis at compile time)
  - Specialized tool: control flow related errors with collective or one-sided operations
  - Capability for dynamic data race detection was recently added
- ITAC (Intel Trace Analyzer and Collector)
  - Dynamic tool (intercepts MPI calls at runtime)
  - General tool: Checks for many different kinds of errors
  - development will be discontinued
- MUST (Marmot Umpire Scalable Tool)
  - Dynamic tool (intercepts MPI calls at runtime)
  - General tool: Checks for many different kinds of errors
  - Integrated ThreadSanitizer to detect data races with MPI in hybrid applications

Tool Results

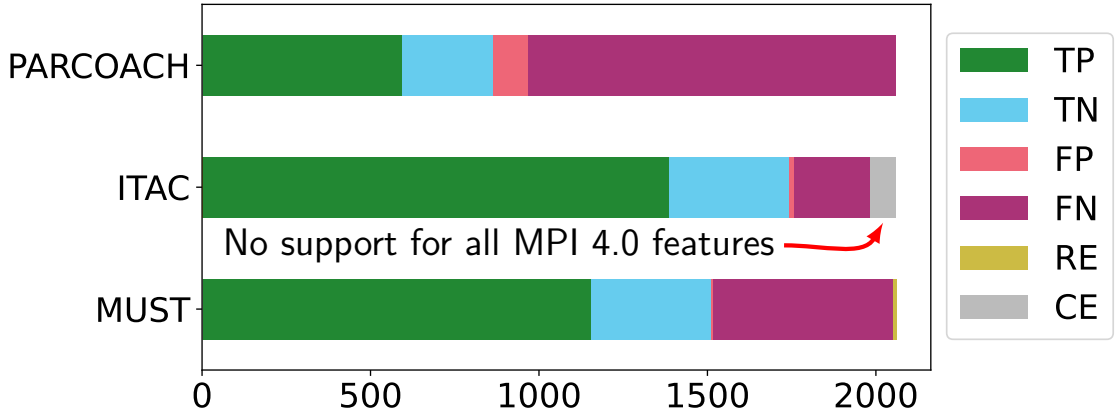
Overall



Tool Results

Overall

TECHNISCHE UNIVERSITÄT DARMSTADT

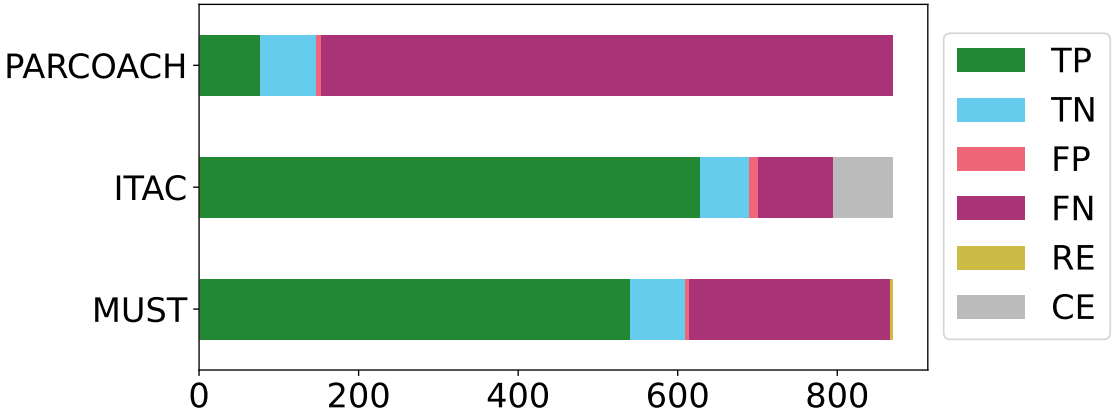


NHR4CES Unit for Computational Engineering Science

Tool Results

Point-to-point operations

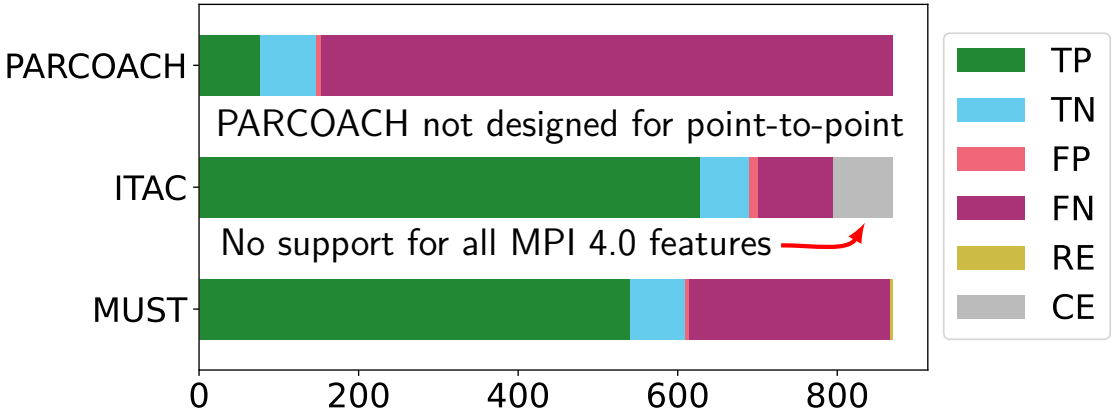
TECHNISCHE UNIVERSITÄT DARMSTADT



NHR4CES

Tool Results

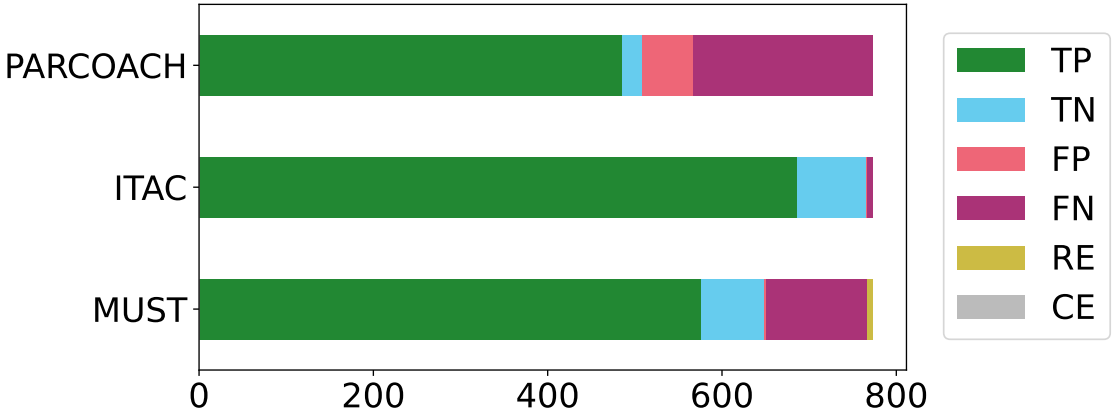
Point-to-point operations



Tool Results

Collective operations

TECHNISCHE UNIVERSITÄT DARMSTADT

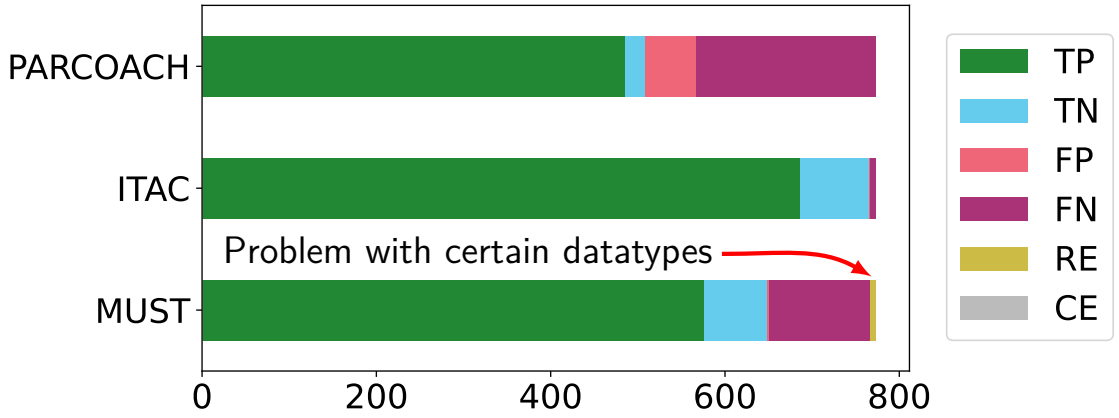


NHR4CES

Tool Results

Collective operations

TECHNISCHE UNIVERSITÄT DARMSTADT

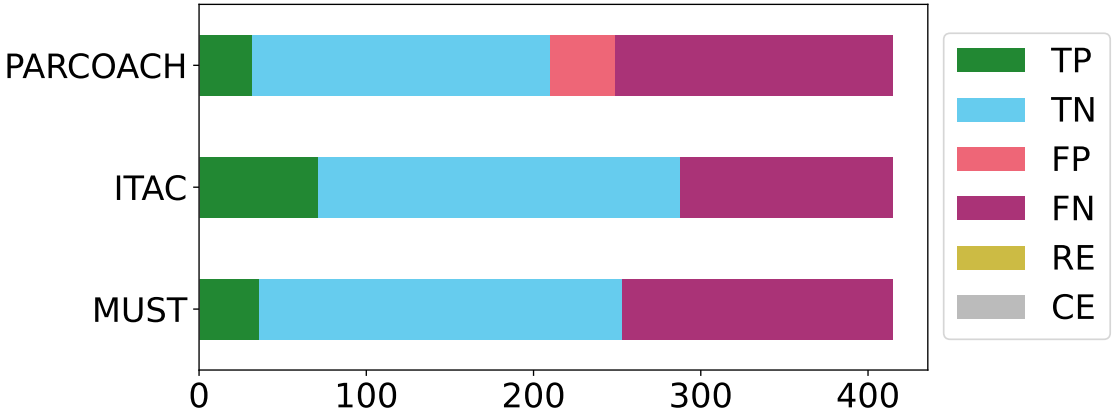


NHR4CES

Tool Results

One-sided operations

TECHNISCHE UNIVERSITÄT DARMSTADT



NHR4CES Unit for Computational Engineering Science

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- ✓ Unified MPI correctness benchmark suite: MPI-BugBench

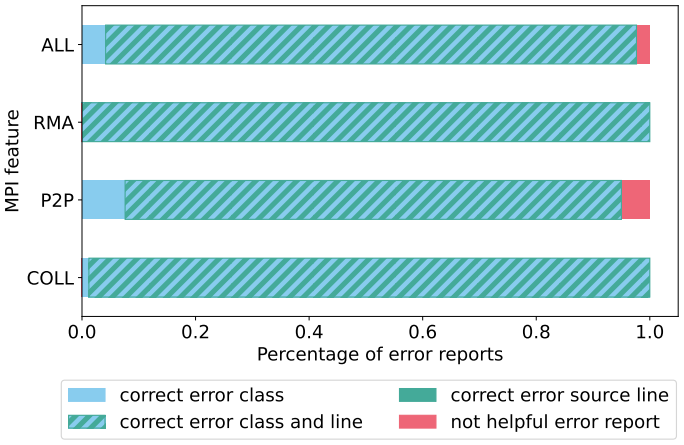
- ✓ Improved real-world applicability
- ✓ Multiple instances of the same error
- ⇒ Points out implementation defects of tools

- ✓ Tools perform reasonable:
  covering the most important aspects of MPI

`https://git-ce.rwth-aachen.de/hpc-public/mpi-bugbench`

Helpfulness of Error Reports
MUST

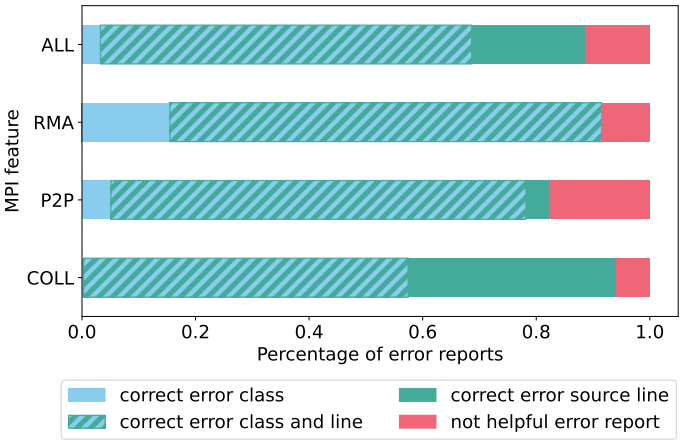
TECHNISCHE UNIVERSITÄT DARMSTADT



Helpfulness of Error Reports

ITAC

TECHNISCHE UNIVERSITÄT DARMSTADT



Helpfulness of Error Reports
PARCOACH

TECHNISCHE UNIVERSITÄT DARMSTADT