MUST: Compiler-aided MPI correctness checking with TypeART

TECHNISCHE UNIVERSITÄT DARMSTADT

RWTH AACHEN UNIVERSITY

FOSDEM'23 HPC, Big Data, and Data Science Devroom
Sunday, 5th February 2023, Brussels, Belgium

Alexander Hück
Scientific Computing, TU Darmstadt
alexander.hueck@sc.tu-darmstadt.de

Joachim Jenke
IT Center, RWTH Aachen
jenke@itc.rwth-aachen.de

MUST
https://itc.rwth-aachen.de/must

TypeART
https://github.com/tudasc/typeart

IT Center | RWTH AACHEN UNIVERSITY | SC SCIENTIFIC COMPUTING

How many errors in this example?

TECHNISCHE UNIVERSITÄT DARMSTADT

```c
#include <mpi.h>
#include <stdio.h>
int main (int argc, char** argv) {
  int rank, size, buf[8];

  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);

  MPI_Datatype type;
  MPI_Type_contiguous (2, MPI_INTEGER, &type);

  MPI_Recv (buf, 2, MPI_INT, size - rank, 123, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  MPI_Send (buf, 2, type, size - rank, 123, MPI_COMM_WORLD);

  printf ("Hello, I am rank %d of %d.\n", rank, size);
  return 0;
}
```

How many errors in this example?

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#include <mpi.h>
#include <stdio.h>
int main (int argc, char** argv) {
  int rank, size, buf[8];

  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);

  MPI_Datatype type;
  MPI_Type_contiguous (2, MPI_INTEGER, &type);

  MPI_Recv (buf, 2, MPI_INT, size - rank, 123, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  MPI_Send (buf, 2, type, size - rank, 123, MPI_COMM_WORLD);

  printf ("Hello, I am rank %d of %d.\n", rank, size);
  return 0;
}
```

No MPI_Init before first MPI call

How many errors in this example?

TECHNISCHE UNIVERSITÄT DARMSTADT

```c
#include <mpi.h>
#include <stdio.h>
int main (int argc, char** argv) {
  int rank, size, buf[8];

  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);

  MPI_Datatype type;
  MPI_Type_contiguous (2, MPI_INTEGER, &type);

  MPI_Recv (buf, 2, MPI_INT, size - rank, 123, MPI_COMM_WOR
  MPI_Send (buf, 2, type, size - rank, 123, MPI_COMM_WORLD);

  printf ("Hello, I am rank %d of %d.\n", rank, size);
  return 0;
}
```

No MPI_Init before first MPI call
Fortran type in C
Recv-Recv deadlock
Rank 0: src=size (out of range)
Type not committed
Type not free'd before end
Send 4 int, Recv 2 int: trunc.
No MPI_Finalize before end

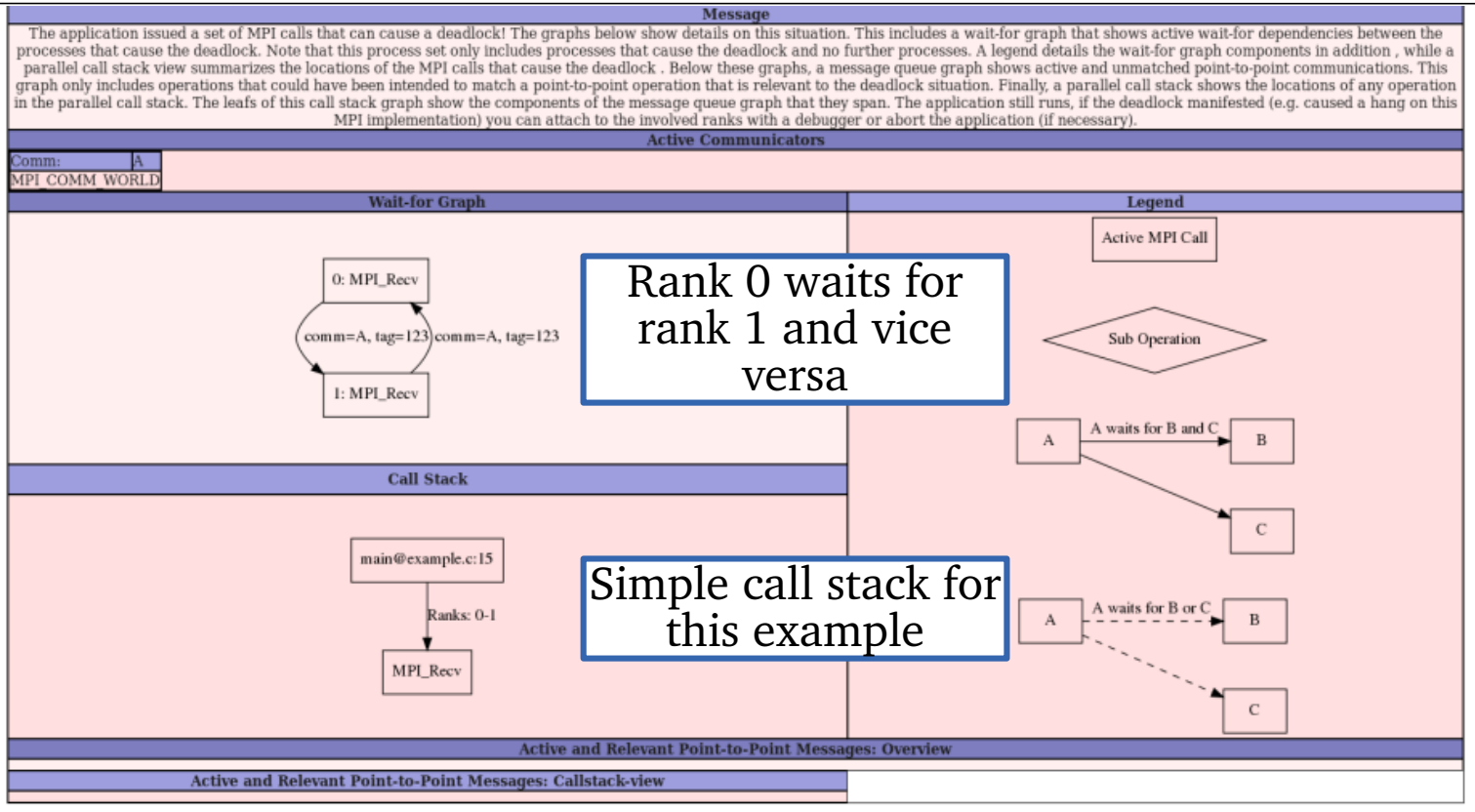
How many errors in this example?

TECHNISCHE UNIVERSITÄT DARMSTADT

```c
#include <mpi.h>
#include <stdio.h>
int main (int argc, char** argv) {
  int rank, size, buf[8];

  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);

  MPI_Datatype type;
  MPI_Type_contiguous (2, MPI_INTEGER, &type);

  MPI_Recv (buf, 2, MPI_INT, size - rank, 123, MPI_COMM_WOR
  MPI_Send (buf, 2, type, size - rank, 123, MPI_COMM_WORLD);

  printf ("Hello, I am rank %d of %d.\n", rank, size);
  return 0;
}
```

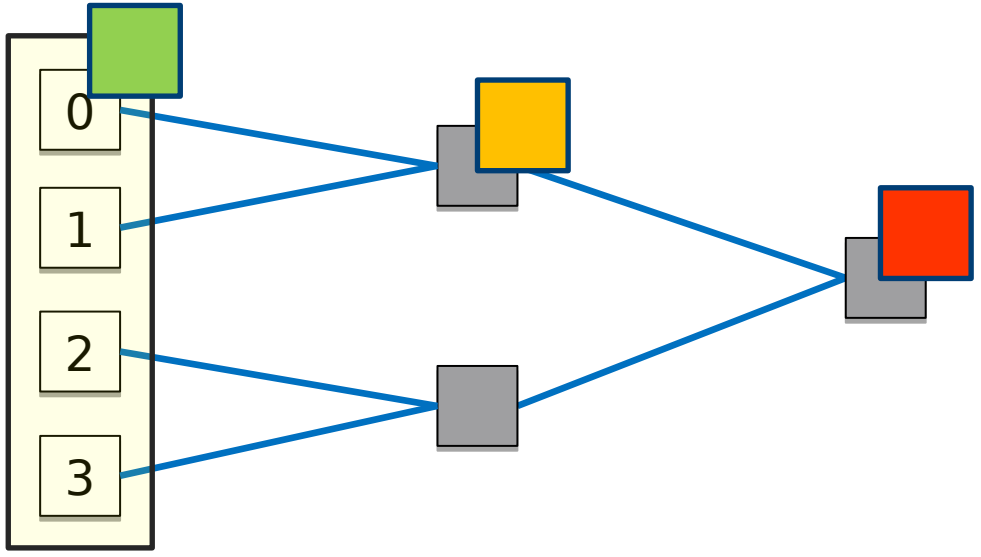
No MPI_Init before first MPI call
Fortran type in C
Recv-Recv deadlock
Rank 0: src=size (out of range)
Type not committed
Type not free'd before end
Send 4 int, Recv 2 int: trunc.
No MPI_Finalize before end

MUST report of deadlock



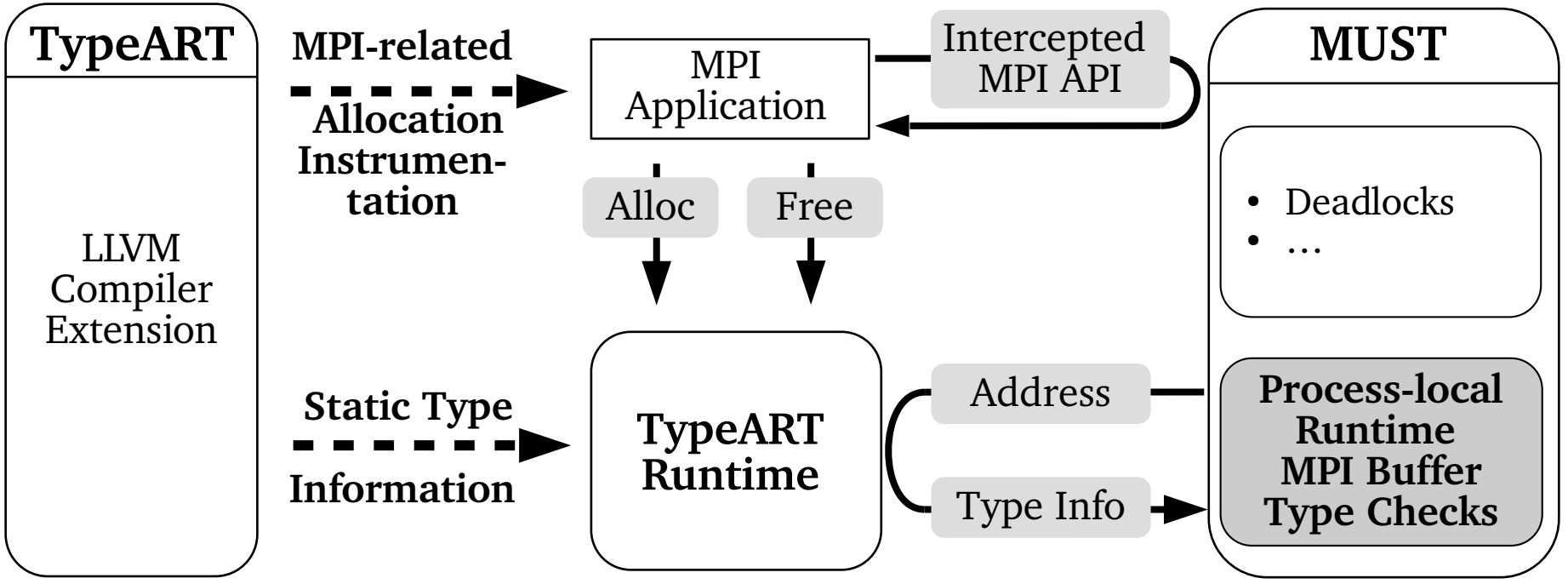
MUST: Agent-based distributed analsysis

TECHNISCHE UNIVERSITÄT DARMSTADT



- Local analysis (e.g., TypeART MPI buffer type correctness)
- Distributed analysis for scalability (e.g., matching MPI send-recv types)
- Centralized analysis for global knowledge (e.g., global deadlocks)
- MUST specific communication (hidden from application)

MUST and TypeART



MUST and TypeART usage

TECHNISCHE UNIVERSITÄT DARMSTADT

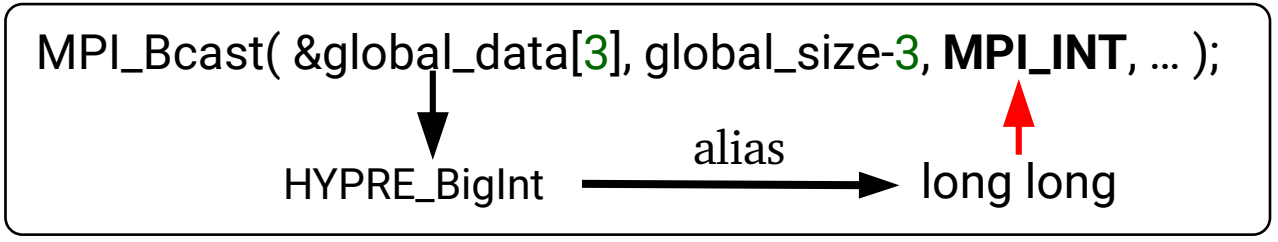
1) Compile and link with TypeART wrapper (optional, if full MPI type checking wanted)
   - mpicc → typeart-mpicc
2) Replace “mpiexec” with “mustrun”
   - e.g., mustrun -np 4 --must:typeart my-app.exe
3) Inspect “MUST_Output.html” in run directory

- The mustrun script will use an extra process for non-local checks (invisible to application)
  - “mustrun −np 4 …” will issue a “mpirun −np 5 …”
- Make sure to allocate the extra task in batch jobs

Examples

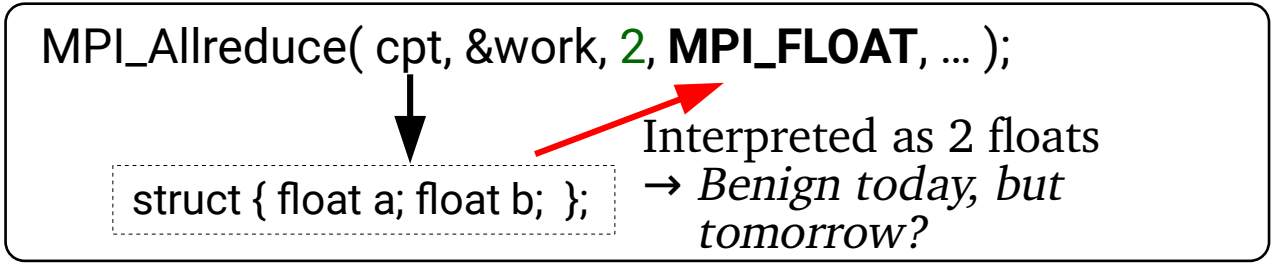
AMG2013, a CORAL performance and parallel scaling benchmark [Coral'20]

- In parcsr_mv/par_csr_matrix.c:1236, reported by [DKL LLVM'15]:



104.milc, a SPEC MPI benchmark [SpecMPI'07]

- In com_mpi.c:480

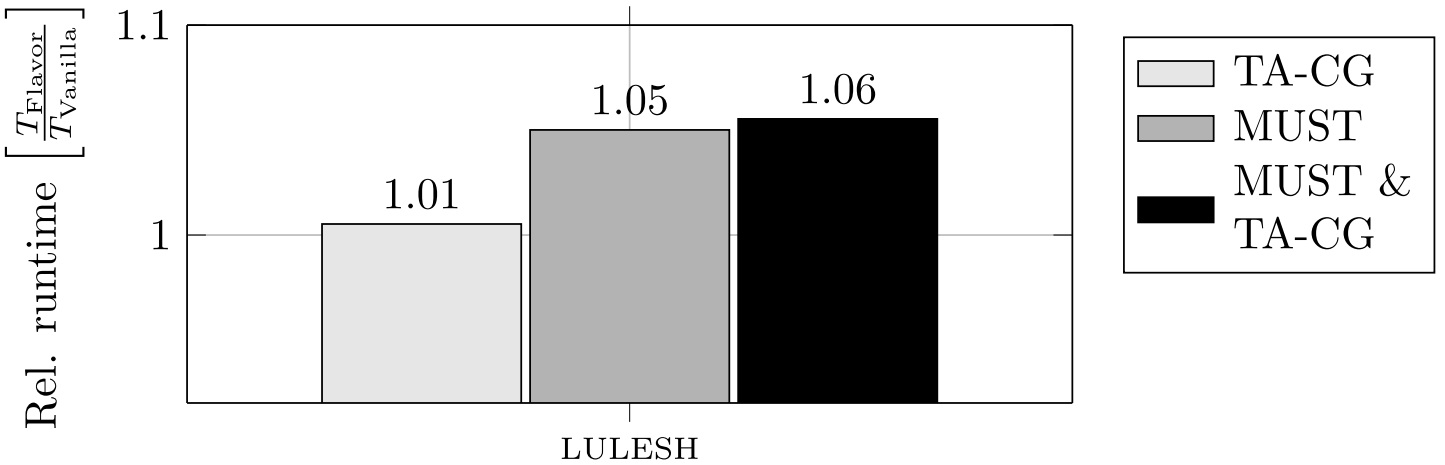


Brief evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

Runtime/memory impact depends on number of tracked allocations

- Runtime overhead for LULESH run, checking MPI communication with MUST (see **[2]**)



- Memory overhead **< 1.1**

IT Center | RWTH AACHEN UNIVERSITY | SC SCIENTIFIC COMPUTING

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

MUST

https://itc.rwth-aachen.de/must

License BSD 3-Clause

"Dynamic MPI correctness checking"

TypeART

https://github.com/tudasc/typeart

License BSD 3-Clause

"C/C++ type and memory allocation tracking"

References

[1] A. Hück, J. Protze, J.-P. Lehr, C. Terboven, C. Bischof, M. S. Müller. **"Towards compiler-aided correctness checking of adjoint MPI applications"** In 4th International Workshop on Software Correctness for HPC Applications (Correctness). IEEE/ACM, 2020, pp. 40–48. DOI: **10.1109/Correctness51934.2020.00010**

[DKL LLVM'15] A. Droste, M. Kuhn, and T. Ludwig, **"MPI-Checker: Static Analysis for MPI"**, In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15. New York, NY, USA: ACM, 2015, pp. 3:1–3:10.

[Coral'20] **"CORAL benchmark codes"**, Last accessed Feb 2020. [Online]. Available: https://asc.llnl.gov/CORAL-benchmarks/

[SpecMPI'07] M. S. Müller, et al, **"SPEC MPI2007 — an application benchmark suite for parallel systems using MPI"**, Concurrency and Computation: Practice and Experience, vol. 22, no. 2, pp. 191–205, 2009.