

MPI Application Development with MARMOT

Bettina Krammer

University of Stuttgart

High-Performance Computing-Center Stuttgart (HLRS)

www.hlrs.de

Matthias Müller

University of Dresden

Centre for Information Services and High Performance Computing (ZIH)

www.tu-dresden.de/zih



Höchstleistungsrechenzentrum Stuttgart



H L R | S



Outline

- Motivation
- Approaches and Tools
- Examples
- Future work and comparison with other tools
- Conclusion



Slide 2

Höchstleistungsrechenzentrum Stuttgart



H L R | S



Motivation



Slide 3

Hochleistungsrechenzentrum Stuttgart



H L R I S



Motivation - Problems of Parallel Programming

- Additional problems:
 - Increased complexity
 - New parallel problems:
 - **deadlocks**
 - **race conditions**
 - **Irreproducibility**
- Portability issues: MPI standard leaves some decisions to implementers, e.g.
 - Message buffering for send/rcv operations
 - Synchronising collective calls
 - Implementation of “opaque objects”



Slide 4

Hochleistungsrechenzentrum Stuttgart



H L R I S



Approaches & Tools



Slide 5

Hochleistungsrechenzentrum Stuttgart



H L R I S



(Parallel) Debuggers

- Most vendor debuggers have some support
- Debugging MPI programs with a “serial” debugger is hard but possible
 - MPIch supports debugging with gdb attached to one process
 - manual attaching to the processes is possible
- Commercial debuggers: Totalview, DDT



Slide 6

Hochleistungsrechenzentrum Stuttgart



H L R I S



Special MPI tools & libraries

- MPI implementations offer (limited) support, e.g.
 - NEC Collectives Verification Library
 - **Only for NEC MPI/SX, MPI/EX**
 - mpich2 profiling library for collective functions
 - **Checks correctness of collective calls and datatypes**
 - **Portable library**
- Tools specially dedicated to analysis of MPI applications:
 - MPI-Check
 - Umpire
 - IMC
 - Marmot



Slide 7

Hochleistungsrechenzentrum Stuttgart



H L R | S



MARMOT

MPI Analysis and Checking Tool

Bettina Krammer



University of Stuttgart

High-Performance Computing-Center Stuttgart (HLRS)

www.hlrs.de



Hochleistungsrechenzentrum Stuttgart



H L R | S



What is MARMOT?

- Tool for the development of MPI applications
- Automatic runtime analysis of the application:
 - Detect incorrect use of MPI
 - Detect non-portable constructs
 - Detect possible race conditions and deadlocks
- MARMOT does not require source code modifications, just relinking
- C and Fortran binding of MPI -1.2 is supported, also C++ and mixed C/Fortran code
- Development is still ongoing (not every possible functionality is implemented yet...)
- Tool makes use of the so-called *profiling interface*



Slide 9

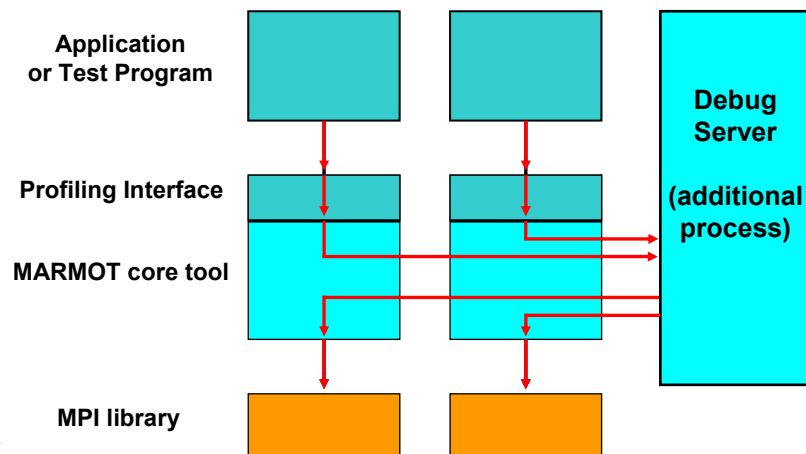
Hochleistungsrechenzentrum Stuttgart



H L R I S



Design of MARMOT



Slide 10

Hochleistungsrechenzentrum Stuttgart



H L R I S



Examples of Server Checks: verification between the nodes, control of program

- Everything that requires a global view
- Control the execution flow, trace the MPI calls on each node throughout the whole application
- Signal conditions, e.g. deadlocks (with traceback on each node.)
- Check matching send/receive pairs for consistency
- Check collective calls for consistency
- Output of human readable log file



Slide 11

Hochleistungsrechenzentrum Stuttgart



H L R I S



Examples of Client Checks: verification on the local nodes

- Verification of proper construction and usage of MPI resources such as communicators, groups, datatypes etc., for example
 - Verification of **MPI_Request** usage
 - invalid recycling of active request
 - invalid use of unregistered request
 - warning if number of requests is zero
 - warning if all requests are **MPI_REQUEST_NULL**
 - Check for pending messages and active requests in **MPI_Finalize**
- Verification of all other arguments such as ranks, tags, etc.



Slide 12

Hochleistungsrechenzentrum Stuttgart



H L R I S



Availability of MARMOT

- Tests on different platforms, using different compilers (Intel, GNU,...) and MPI implementations (mpich, lam, vendor MPIs,...), e.g.
 - IA32/IA64 clusters
 - Opteron clusters
 - Xeon EM64T clusters
 - IBM Regatta
 - NEC SX5,..., SX8
- Download and further information
<http://www.hlr.de/organization/tsc/projects/marmot/>



Slide 13

Hochleistungsrechenzentrum Stuttgart



H L R | S



Examples



Slide 14

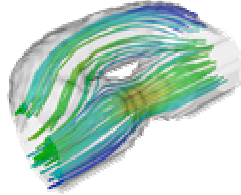
Hochleistungsrechenzentrum Stuttgart



H L R | S



Example - Medical Application B_Stream



- Calculation of blood flow with 3D Lattice-Boltzmann method
- 16 different MPI calls:
 - MPI_Init, MPI_Comm_rank, MPI_Comm_size, MPI_Pack, MPI_Bcast, MPI_Unpack, MPI_Cart_create, MPI_Cart_shift, MPI_Cart_rank, MPI_Send, MPI_Recv, MPI_Barrier, MPI_Reduce, MPI_Sendrecv, MPI_Wtime, MPI_Finalize
- Around 6500 lines of code
- We use different input files that describe the geometry of the artery: tube, tube-stenosis, bifurcation



Slide 15

Höchstleistungsrechenzentrum Stuttgart



H L R S



Example: B_Stream (blood flow simulation, tube)

- Tube geometry: simplest case, just a tube with about the same radius everywhere
- Running the application without/with MARMOT:

```
mpirun -np 3 B_Stream 500. tube
mpirun -np 4 B_Stream_marmot 500. tube
```
- Application seems to run without problems



Slide 16

Höchstleistungsrechenzentrum Stuttgart



H L R S



Example: B_Stream (blood flow simulation, tube-stenosis)

- Tube-stenosis geometry: just a tube with varying radius
- Without MARMOT:
`mpirun -np 3 B_Stream 500. tube-stenosis`
- Application seems to be hanging
- With MARMOT:
`mpirun -np 4 B_Stream_marmot 500. tube-stenosis`
- Deadlock found



Slide 17

Hochleistungsrechenzentrum Stuttgart



H L R I S



Example: B_Stream (blood flow simulation, tube-stenosis)

```
9310 rank 1 performs MPI_Sendrecv
9311 rank 2 performs MPI_Sendrecv
9312 rank 0 performs MPI_Barrier
9313 rank 1 performs MPI_Barrier
9314 rank 2 performs MPI_Barrier
9315 rank 1 performs MPI_Sendrecv
9316 rank 2 performs MPI_Sendrecv
9317 rank 0 performs MPI_Sendrecv
9318 rank 1 performs MPI_Sendrecv
9319 rank 0 performs MPI_Sendrecv
9320 rank 2 performs MPI_Sendrecv
9321 rank 0 performs MPI_Barrier
9322 rank 1 performs MPI_Barrier
9323 rank 2 performs MPI_Barrier
9324 rank 1 performs MPI_Comm_rank
9325 rank 1 performs MPI_Bcast
9326 rank 2 performs MPI_Comm_rank
9327 rank 2 performs MPI_Bcast
9328 rank 0 performs MPI_Sendrecv
```

Iteration step:
Calculate and exchange results with neighbors

Communicate results among all procs

WARNING: all clients are pending!



Slide 18

Hochleistungsrechenzentrum Stuttgart



H L R I S



Example: B_Stream (blood flow simulation, tube-stenosis) deadlock

Node
0

```
timestamp= 9319: MPI_Sendrecv(*sendbuf, sendcount = 7220,
    sendtype = MPI_DOUBLE, dest = 1, sendtag = 1, *recvbuf,
    recvcnt = 7220, recvtype = MPI_DOUBLE, source = 2, recvtg
    = 1, comm = self-defined communicator, *status)
timestamp= 9321: MPI_Barrier(comm = MPI_COMM_WORLD)
timestamp= 9328: MPI_Sendrecv(*sendbuf, sendcount = 7220,
    sendtype = MPI_DOUBLE, dest = 2, sendtag = 1, *recvbuf,
    recvcnt = 7220, recvtype = MPI_DOUBLE, source = 1, recvtg
    = 1, comm = self-defined communicator, *status)
```

Node
1

```
timestamp= 9322: MPI_Barrier(comm = MPI_COMM_WORLD)
timestamp= 9324: MPI_Comm_rank(comm = MPI_COMM_WORLD, *rank)
timestamp= 9325: MPI_Bcast(*buffer, count = 3, datatype =
    MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)
```

Node
2

```
timestamp= 9323: MPI_Barrier(comm = MPI_COMM_WORLD)
timestamp= 9326: MPI_Comm_rank(comm = MPI_COMM_WORLD, *rank)
timestamp= 9327: MPI_Bcast(*buffer, count = 3, datatype =
    MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)
```



Slide 19

Hochleistungsrechenzentrum Stuttgart



H L R I S



Example: B_Stream (blood flow simulation) - Code Analysis

```
main {
```

```
...
```

```
num_iter = calculate_number_of_iterations();
```

```
for (i=0; i < num_iter; i++) {
```

```
    computeBloodflow();
```

```
}
```

```
writeResults();
```

```
.... // communicate results with neighbors
MPI_Bcast(...);
}
```

if (radius < x) num_iter = 50;
if (radius >= x) num_iter = 200;
// **ERROR:** it is not ensured here that all
// procs do the same (maximal) number
// of iterations

CalculateSomething();
// exchange results with neighbors
MPI_Sendrecv(...);
MPI_Barrier(...);

Be careful if you call functions with hidden MPI calls!



Slide 20

Hochleistungsrechenzentrum Stuttgart



H L R I S



Example: B_Stream (blood flow simulation, bifurcation)

- Bifurcation geometry: forked artery
- Without MARMOT:
mpirun -np 3 B_Stream 500. bifurcation



Segmentation fault

(platform dependent if the code breaks here or not)

- With MARMOT:
mpirun -np 4 B_Stream_marmot 500. bifurcation
- Problem found at collective call MPI_Gather



Slide 21

Hochleistungsrechenzentrum Stuttgart



H L R I S



Example: B_Stream (blood flow simulation, bifurcation)

Last calls on node 0:

```
timestamp= 9327: MPI_Bcast(*buffer, count = 3, datatype =  
MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)  
timestamp= 9330: MPI_Bcast(*buffer, count = 3, datatype =  
MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)  
timestamp= 9333: MPI_Gather(*sendbuf, sendcount = 266409,  
sendtype = MPI_DOUBLE, *recvbuf, recvcoun = 266409,  
recvtype = MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)
```

Last calls on node 1:

```
timestamp= 9334: MPI_Gather(*sendbuf, sendcount = 258336,  
sendtype = MPI_DOUBLE, *recvbuf, recvcoun = 258336,  
recvtype = MPI_DOUBLE, root = 0, comm = MPI_COMM_WORLD)  
timestamp= 9336: MPI_Gather(*sendbuf, sendcount =  
13455, sendtype = MPI_DOUBLE, *recvbuf, recvcoun = 13455,  
*recvbuf, recvcoun = 13455, MPI_DOUBLE,  
source = 2, recvtags = 1, MPI_DOUBLE,  
communicator, *status)  
timestamp= 9338: MPI_Sendrecv(*sendbuf, sendcount =  
13455, sendtype = MPI_DOUBLE, dest = 2, sendtag = 1,  
*recvbuf, recvcoun = 13455, recvtype = MPI_DOUBLE,  
source = 0, recvtags = 1, comm = self-defined  
communicator, *status)
```

ERROR: Root 0 has different
counts than rank 1 and 2



Slide 22

Hochleistungsrechenzentrum Stuttgart



H L R I S



MARMOT

Performance with real applications



Slide 23

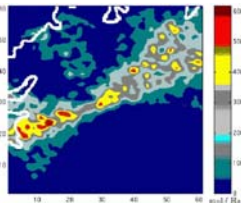
Hochleistungsrechenzentrum Stuttgart



H L R I S



Air pollution modelling



- Air pollution modeling with STEM-II model
- Transport equation solved with Petrov-Crank-Nikolson-Galerkin method
- Chemistry and Mass transfer are integrated using semi-implicit Euler and pseudo-analytical methods
- 15500 lines of Fortran code
- 12 different MPI calls:
 - MPI_Init, MPI_Comm_size, MPI_Comm_rank, MPI_Type_extent, MPI_Type_struct, MPI_Type_commit, MPI_Type_hvector, MPI_Bcast, MPI_Scatterv, MPI_Barrier, MPI_Gatherv, MPI_Finalize.



Slide 24

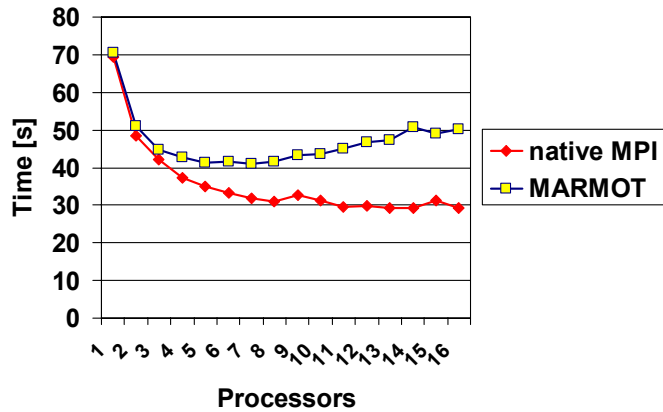
Hochleistungsrechenzentrum Stuttgart



H L R I S



STEM application on an IA32 cluster with Myrinet



Slide 25

Hochleistungsrechenzentrum Stuttgart



H L R I S



Comparison with other approaches and Future Directions



Slide 26

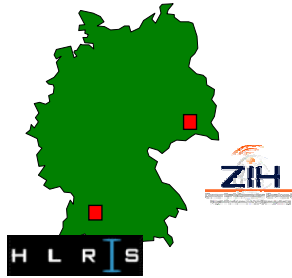
Hochleistungsrechenzentrum Stuttgart



H L R I S



Future Direction of MARMOT



- MARMOT will continue to be developed jointly by ZIH and HLRS



Slide 27

Hochleistungsrechenzentrum Stuttgart



H L R | S



Future Directions

Functionality

- More checks
- MPI-2
- OpenMP/MPI

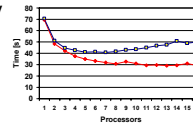
Usability

- GUI



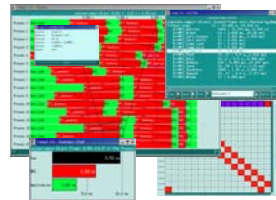
Performance

- Scalability



Combination with other tools

- Debugger
- Vampir



Slide 28

Hochleistungsrechenzentrum Stuttgart



H L R | S



Online vs. Offline Checking

- Offline Checking (Intel Message Checker):
 - + History information available
 - + Less intrusive at runtime
 - Large file space needed for trace file
 - Limited scalability of analysis
- Online Checking (Marmot):
 - Reduced performance of running application
 - Only limited history/time line information
 - + No limit for the runtime of the program
 - + The program is still „alive“ when the error is detected, further online analysis is possible



Slide 29

Hochleistungsrechenzentrum Stuttgart



H L R | S



MPI implementation with checking

- + Scalability and performance
- + Avoids duplicating internal management tasks
- + Avoids re-implementation of existing functionality
- No history/time line information
- Performance and reliability is always more important than checks
- Portability problems are not a major focus



Slide 30

Hochleistungsrechenzentrum Stuttgart



H L R | S



Comparison of systems

	Online	Offline	Library
Checks	++	++	0
Scalability and Performance	+-	+-	++
Runtime scalability	++	-	++
Timeline information	-	++	--
Combination with other tools	+	-	-



Slide 31

Hochleistungsrechenzentrum Stuttgart



H L R I S



Conclusion

- Parallel programming in general and the MPI standard contains enough pitfalls to raise a need for checking/correctness/confidence tools
- Different tools and approaches with different advantages and disadvantages
- MARMOT is a freely available solution that has demonstrated its usefulness with various real applications
- A combination of tools will offer the best solution for the program developer
- Not every error can be detected by tools



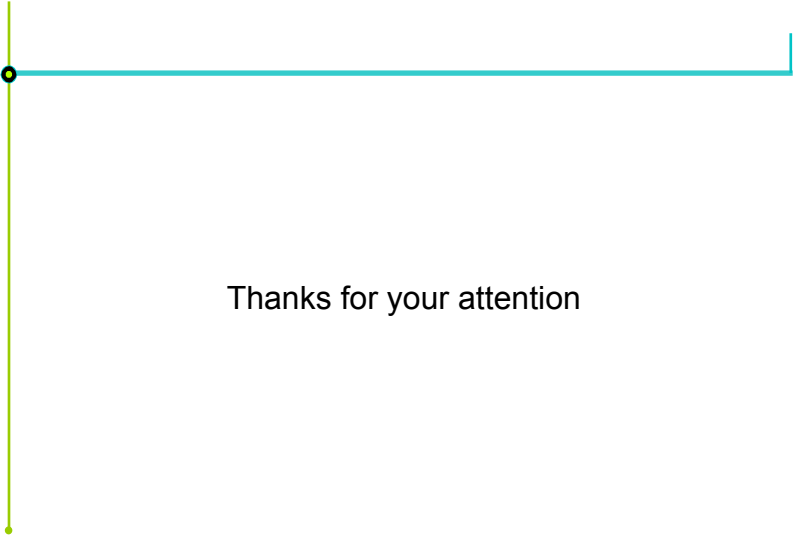
Slide 32

Hochleistungsrechenzentrum Stuttgart



H L R I S





Thanks for your attention



Slide 33

Hochleistungsrechenzentrum Stuttgart



H L R I S

