Coupling DDT and Marmot for Debugging of MPI Applications

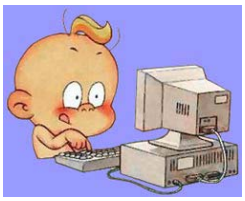
Bettina Krammer, Valentin Himmler
University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de
David Lecomber
Allinea Software
www.allinea.com

Outline

- Motivation
- Introduction to DDT and Marmot
- Coupling DDT and Marmot
- Conclusions

Motivation

- Parallel programming is complex and error-prone
- Couple stand-alone tools to cover different levels of debugging and correctness checking of MPI applications
- Need for a user-friendly environment



Introduction
to
DDT and Marmot

DDT – Distributed Debugging Tool

- Source-level debugger for scalar, multi-threaded and large-scale parallel C, C++ and Fortran codes
- Support for all MPIs
- Easy-to-use graphical interface
- www.allinea.com

What is Marmot?

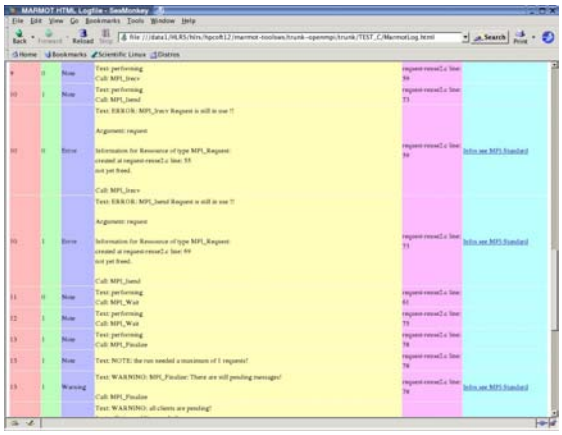
History of Marmot

Now – what Marmot REALLY is

- Tool for the development of MPI applications
- Automatic correctness checking at runtime:
  - Detect incorrect use of MPI
  - Detect non-portable constructs
  - Detect possible race conditions and deadlocks
- no source code modification, just relinking and run with 1 additional process
- Different output formats (txt, html, xml)
- 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*

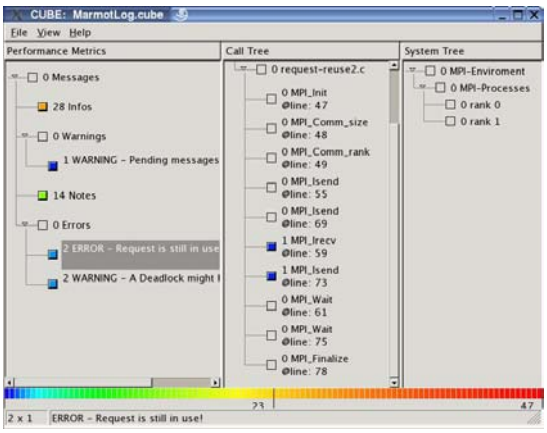
output html

```
export MARMOT_LOGFILE_TYPE=1
```



output cube

```
export MARMOT_LOGFILE_TYPE=2
```

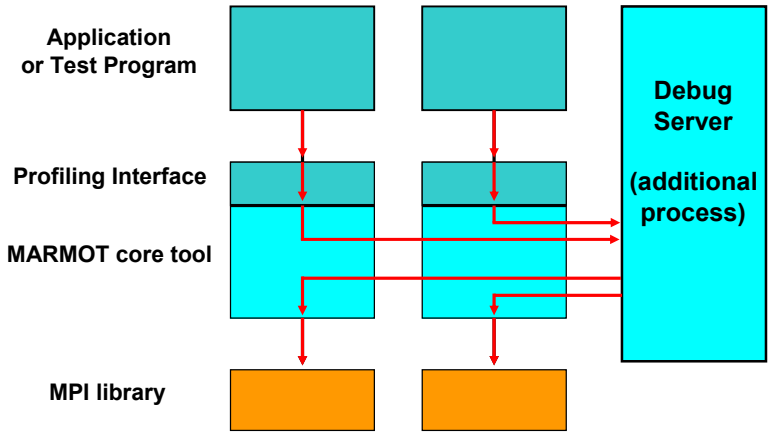


Marmot

- Portable tool , tested on many platforms (ia32/ia64, Opteron/Xeon, IBM, SX6, SX8,...)

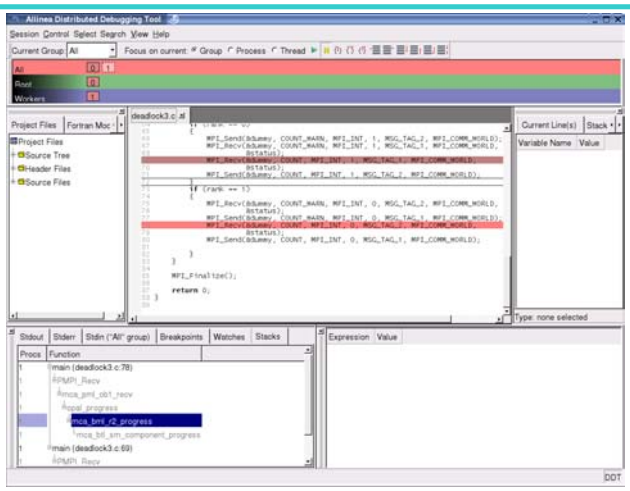
  - Different compilers (Intel, GNU, NEC,...)
  - Different MPIs (mpich, Open MPI, Lam-MPI, NEC MPI, intel MPI, Voltaire MPI,...)
- Marmot works basically anywhere, main challenges:
  - Guess all configure options right
    - **not possible to find everything automatically, e.g. --with-mpi-dir=... etc.**
  - Link examples correctly
- www.hlrs.de/organization/amt/services/tools/debugger/marmot

Design of MARMOT (1)



DDT  Marmot

Simple example – with DDT, without Marmot



Design of MARMOT (2)

Client (Application)

```
...
MPI_Init{
  PMPI_Init();
  initComm;
  MapResources;
  ...

}
...
MPI_Recv{
  doSomeChecks;

  PMPI_Recv;
}
...
```

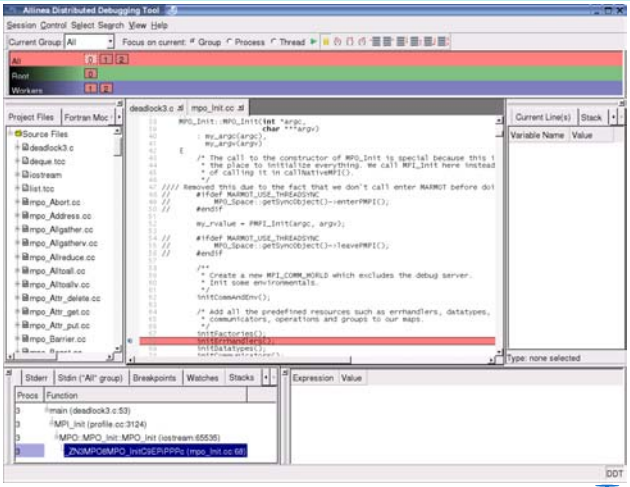
Debug Server

```
...
MPI_Init{
  PMPI_Init();
  initComm;
  MapResources;
  ...

  //don't leave MPI_Init
  DebugServerCode;

  logging;

  ...

}
```


Integration with DDT

- Marmot's special MPI_Init (last process):
  `export DDT_MPI_Init=PMPI_Init`
- Compile everything with `-g` (including marmot code)

Simple example – with DDT & Marmot (scary, isn't it)



Design of MARMOT (3)

Client (Application)

```
...
MPI_Init{
  PMPI_Init();
  initComm;
  MapResources;
  ...
}
...
MPI_Recv{
  doSomeChecks;

  PMPI_Recv;
}
...
```

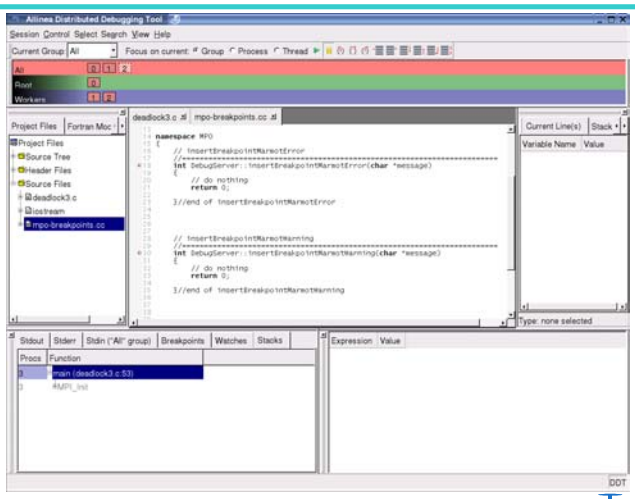
Debug Server

```
...
MPI_Init{
  PMPI_Init();
  initComm;
  MapResources;
  ...
  //don't leave MPI_Init
  DebugServerCode;
  logging;
  if (error)
    insertBreakpointError;
  if (warning)
    insertBreakpointWarning;
  ...
}
```

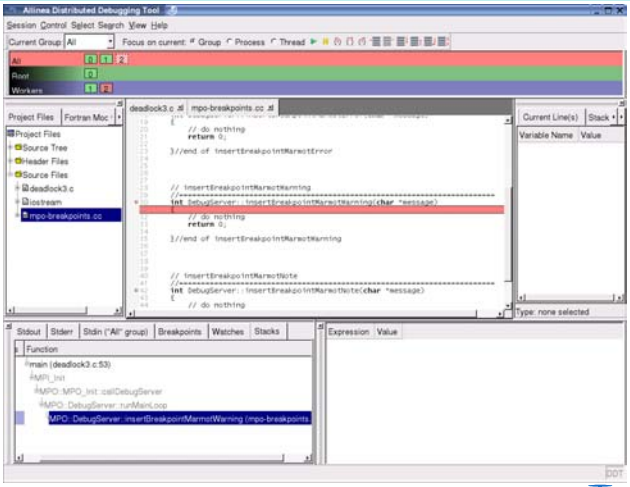
DDT & Marmot

- Insert breakpoint calls in our debug server (highest rank) when error/warning is detected (mpo-breakpoints.cc ~ 30 lines of code)
- Just compile Marmot's mpo-breakpoints.cc source file with `-g`
- Display Marmot warnings (stderr/variables windows, further ideas)

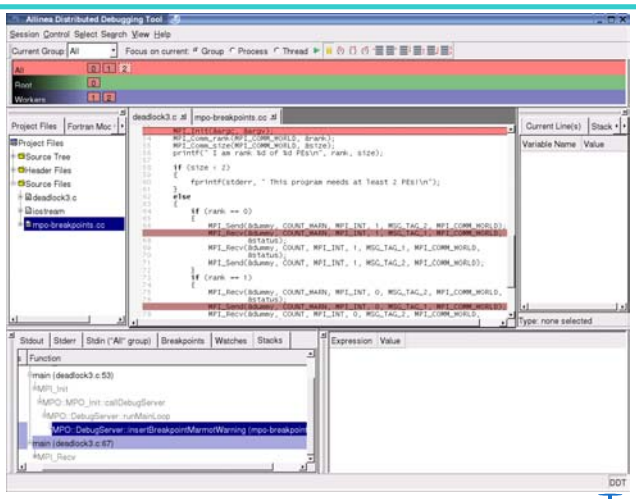
Simple example – with DDT & Marmot (breakpoints)



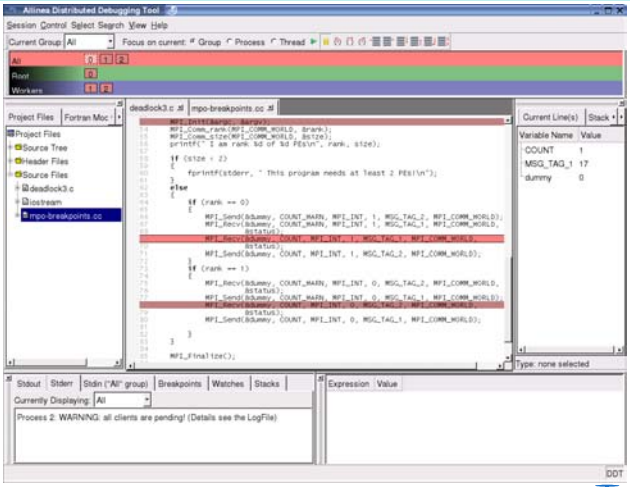
Simple example – with DDT & Marmot (warning detected)



Simple example – with DDT & Marmot (warning detected)



Simple example – with DDT & Marmot (error display)



Conclusions and Future Work

- DDT & Marmot – it works!
- It's not rocket science... but takes some considerations, e.g.
  - how to handle this last Marmot process (hide it completely eventually? DDT to add breakpoints automatically)
  - how to handle error dialogues? Add information? Message queues?
  - adapt Marmot's build process (shared libraries) to be able to switch Marmot on/off through DDT GUI (`LD_PRELOAD`)

Conclusions and Future Work

- Tested with simple examples on various platforms
  - Tests with real applications to be performed
  - Shared lib approach tested on few platforms

Conclusions and Future Work

- Marmot is alive and kickin'
- It's under active development by HLRS and ZIH
- Also within the ParMA project – Parallel Programming for Multi-core Architectures

  www.parma-itea2.org

  (collaboration with Vampir, Kojak, DDT, OPT,...)

Thanks for your attention