GWDG
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Debugging complex simulation codes in parallel

Patrick Höhn (patrick.hoehn@gwdg.de)



04.07.2025

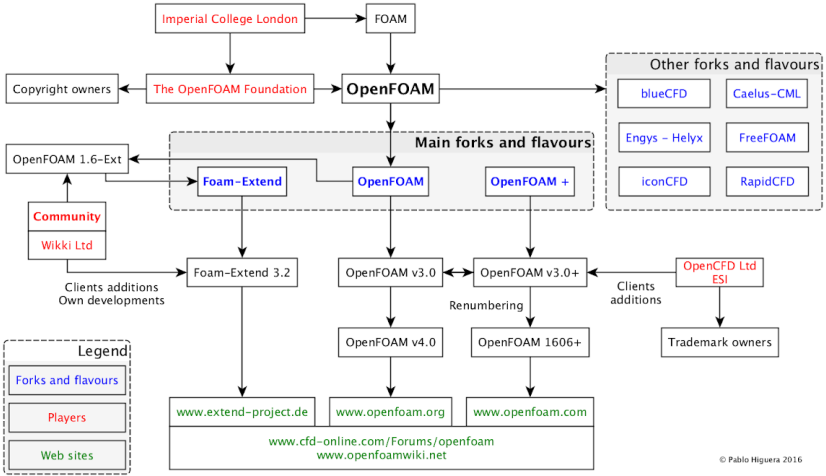
Agenda

- Introduction
- Debugging Tools
- Hands-on
- Parallel Debugging
- Application to preCICE

Motivation

- Code not working as intented
- Unterstanding of running code
- Fixing of runtime errors
- Difficult to study compiled code

OpenFOAM



Source: https://olaflow.github.io/blog/what-is-what-and-who-is-who-in-the-openfoam-environment/

Motivation for Parallel Debugging

- Faster execution running in parallel
-

Basic Debugging by Text Messages to Terminal

```
1 Info<< "\nStarting time loop\n" << endl;
2
3 while (runTime.loop())
4 {
5     Info<< "Time = " << runTime.timeName() << nl << endl;
6     ...
7 }
```

Downsides:

- no control of code during execution
- requires modification of code and recompilation
- cleaning of code required after successful debugging

DebugSwitches

- 0 means no debug information
- Different debug levels available (1,2,3...)
- No recompilation of coded needed unlike `Info` statements
- List and activate debugSwitches of solver (OpenFOAM foundation and ESI)

```
1 pisoFoam -listRegisteredSwitches
2 pisoFoam -debug-switch <name=val>
```

- List and activate debugSwitches of solver (foam-extend)

```
1 pisoFoam -dumpControlSwitches
2 pisoFoam -DebugSwitches <key1=val1,key2=val2, ...>
```

gdb

- Study code run-time behaviour
- Examining variables at run-time
- Changing program at run-time
- BUT: more disc space required and slower execution time
- For own solver or library modification of `Make/options` needed:

```
1 EXE_INC = -O0 -fdefault-inline -ggdb -DFULLDEBUG
```

gdbOF and valgrind

gdbOF

- Simplification of commands from pure GDB usage by additionally implemented additional macros
- Easier inspection of data structures of OF at run-time
- Dumping of data at run-time
- Currently unmaintained and buggy

valgrind

- Open-Source framework for memory debugging, memory leak detection and profiling
- No profiling demonstrated as part of this talk

Visual Studio Code

- Multiplatform (Windows, macOS, Linux) IDE (Integrated Development Environment) Visual Studio Code
- Open source variant without telemetry VSCodium
- Supported languages: C, C#, C++, JavaScript, Julia, Perl, Rust, ...
- Syntax highlighting, auto completion, revision management
- Graphical debugging

debugSwitches

```
module load gcc/11.5.0 openmpi foam-extend/4.1-debug # or locally fe41_debug
source $WM_PROJECT_DIR/etc/bashrc
wget https://owncloud.gwdg.de/index.php/s/yzG9wBfypM0YrZY/download -O code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz
cd $HOME/pitzDaily
blockMesh
scalarTransportFoam -DebugSwitches volScalarField=1,volVectorField=2

# Output without DebugSwitches:
Time = 0.05

BiCGStab:  Solving for T, Initial residual = 1, Final residual = 0, No Iterations 1

# Output with DebugSwitches:
Time = 0.05

GeometricField<Type, PatchField, GeoMesh>::GeometricBoundaryField::updateCoeffs()
GeometricField<Type, PatchField, GeoMesh>::GeometricBoundaryField::updateCoeffs()
GeometricField<Type, PatchField, GeoMesh>::GeometricBoundaryField::updateCoeffs()
GeometricField<Type, PatchField, GeoMesh>::GeometricBoundaryField::GeometricBoundaryField(const
      ↪ GeometricBoundaryField<Type, PatchField, BoundaryMesh>&)
GeometricField<Type, PatchField, GeoMesh>::GeometricField : constructing as copy resetting IO params
IOobject: volScalarField T_0 "/home/hoehn7/foam/foam-extend-4.1/tutorials/basic/scalarTransportFoam/swirlTest/0.05"

BiCGStab:  Solving for T, Initial residual = 1, Final residual = 0, No Iterations 1
```

gdb

```
module load gcc/11.5.0 openmpi gdb foam-extend/4.1-debug # or locally fe41_debug
source $WM_PROJECT_DIR/etc/bashrc
wget https://owncloud.gwdg.de/index.php/s/yzG9wBfypM0YrZY/download -O code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz
cd $HOME/pitzDaily
blockMesh
gdb myScalarTransportFoam
```

command	**meaning**
[h]elp	list all classes of commands
help class	one-line description for commands in class
help command	description of command
[l]ist	shows ten lines after previous listing
list main	shows first ten lines around function
list 30,50	shows defined line range
list createTime.H:1,20	shows first twenty lines of createTime.H

gdb 2

command	**meaning**
[r]un run arglist	start your program with current argument list start your program with arglist
[c]ontinue	resumes execution of program until next breakpoint
ctrl-x ctrl-a tui enable ctrl-l	starting of Text User Inferface (TUI) Mode starting of Text User Inferface (TUI) Mode redraw TUI window
[b]ack[t]race backtrace n	print trace of all frames in stack print trace of n frames in stack
frame n	select frame number n

gdb 3

command	**meaning**
[b]reak (file:)line break (file:)line if var==value	set breakpoint at line (in file) conditional breakpoint at line (in file) if var equal value
clear	clear all breakpoints
whatis expr ptype expr	show datatype of expression show more details on datatype compared to whatis
watchpoint expr	set a watchpoint for expression expr
[s]tep step n	run next line, stepping into function calls run n lines, stepping into function calls

gdb 4

command	meaning
[n]ext	run next line, stepping over function calls
next n	run n lines, stepping over function calls
info break	show defined breakpoints
info watch	show defined watch points
finish	run until stack frame returns
[p]rint *T_.v_@nvalues	print first n values of T_ variable
p *T_.v_@(begin,end)	print values from begin to end of T_ variable
call Foam::min(1,2)	call min function from inside OpenFOAM
[q]uit	quit gdb

Start Debugging with gdb

```
module load gcc/11.5.0 openmpi gdb foam-extend/4.1-debug # or locally fe41_debug
cd $HOME
wget https://owncloud.gwdg.de/index.php/s/yzG9wBfypM0YrZY/download -O code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz
cd ~/myScalarTransportFoam/
{vi|nano|emacs|gedit} # editor of your choice
wmake
# start debugging with gdb
```

Parallel Execution Basics

- Shared Memory Programming Model (Multithreading)
  - OpenMP
  - POSIX Threading Library
- Message Passing Programming Model
  - MPI (OpenMPI, MPICH)
- SIMD Programming Model
  - CUDA
  - OpenCL

Compiler Options

To debug a code, extra debug information is necessary

- GCC
  - For serial code: `$ gcc/g++ -g xxx.c -o xxx`
  - For multithreaded code: `$ gcc/g++ -g -fopenmp xxx.c -o xxx`
  - For MPI code: `$ mpicc/mpic++ -g xxx.c -o xxx`
- Intel [oneAPI]
  - For serial code: `$ icc/icpx -g -02 xxx.c -o xxx`
  - For multithreaded code: `$ icc/icpx -g -Qopenmp xxx.c -o xxx`
  - For MPI code: `$ mpicc/mpiicc -g xxx.c -o xxx`

MPI Fundamentals

MPI: **M**essage **P**assing **I**nterface

Core functionality:

- Process management
- Parallel file I/O
- Collective Communication
- Point to Point Communication
- One Sided Communication
- Derived Datatypes
- Synchronization

Supported Languages:

- C
- C++
- Fortran
- Python
- MATLAB
- Julia
- ...

Challenges of Debugging Parallel Codes

- Deadlocks
- Concurrent write memory access
- Communication overhead
- Load balancing
- Unpredictable result based on execution order

Available Parallel Debugging Tools

- Commercial:
  - Linareo Forge DDT
  - Totalview
- Open Source:
  - The Eclipse Parallel Tools Project
  - pgdb
  - padb
  - mpigdb
  - tmpi

tmpi

- Introduction:
  - Based on tmux
  - Runs executables with MPI as a grid in a tmux window
  - Multiplexes keystrokes to whole grid
  - Can be combined with GDB for parallel debugging
- Installation:

```
module load gcc/11.5.0 openmpi gdb git
git clone https://github.com/Azrael3000/tmpi.git
echo "export PATH=\$HOME/tmpi:\$PATH" >> $HOME/.bashrc
```

Demo tmpi

- Usage:

```
module load gcc/11.5.0 openmpi gdb
tmpi [number] [command]
tmpi 4 gdb executable
tmpi 4 gdb -x script.gdb executable
```

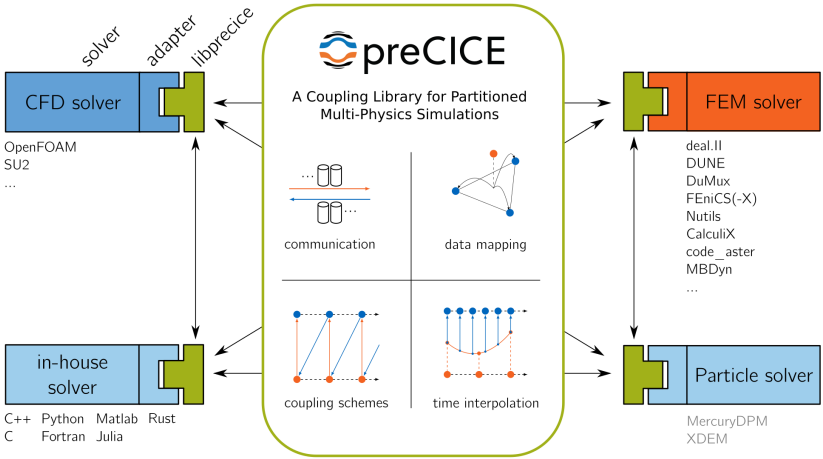
- Important commands

```
# Close debugging windows
Ctrl-b + &
# Switch to decoupled windows
Ctrl-b + z
```

Demo tmpi

```
module load gcc/11.5.0 openmpi gdb foam-extend/4.1-debug
source $WM_PROJECT_DIR/etc/bashrc
wget https://owncloud.gwdg.de/index.php/s/yzG9wBfypM0YrZY/download -O
    ↪ code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz
cd $HOME/pitzDaily
blockMesh
decomposePar
tmpi 4 gdb --args myScalarTransportFoam -parallel
reconstructPar
```

preCICE fundamentals



Configuration Check

- available built-in since version 2.4.0.
- Usage:

```
precice-tools check FILE [ PARTICIPANT [ COMMSIZE ] ]
precice-tools check precice-config.xml
precice-tools check precice-config.xml Fluid
precice-tools check precice-config.xml Fluid 2
```

- Examples of detected errors:
  - Misspelled tags
  - Misspelled data names
  - Incorrect attribute combinations
  - Incorrect meshes used in mapping definitions

Config Visualization - Introduction

- Motivation:
  - Cover issues not checked for in configuration check
  - Understand configurations by collaborators visually
  - Can be included in documentation and presentations
- Implementation:
  - Generation as dot graph
  - Tooling provided by preCICE:
    `https://precice.org/tooling-config-visualization.html`
  - Two modes: CLI and interactive GUI

Config Visualization - Installation

- Install dependencies:
  - `python3`, `pip` or `pipx`, `graphviz`
  - Additonally on Ubuntu:

```
sudo apt-get install libcairo2-dev libgirepository1.0-dev gcc
    ↪ libcairo2-dev pkg-config python3-dev gir1.2-gtk-4.0
```

- Install the python packages:

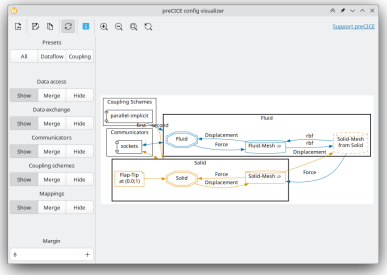
```
pip install precice-config-visualizer
# For GUI addtionally install:
pip install precice-config-visualizer-gui
```

Config Visualization - Usage GUI

- Open the GUI:

```
precice-config-visualizer-gui
# Or directly open the precice-config.xml
precice-config-visualizer-gui precice-config.xml
```

- Opened Window:



Config Visualization - Usage CLI

- Render a preCICE configuration as a PNG image directly:

```
precice-config-visualizer -o graph.png precice-config.xml
```

- Customize the graph further first generate the dot graph:

```
precice-config-visualizer precice-config.xml > graph.dot
# Customize using dot language
dot -Tpng -O graph.png graph.dot
```

Software Setup

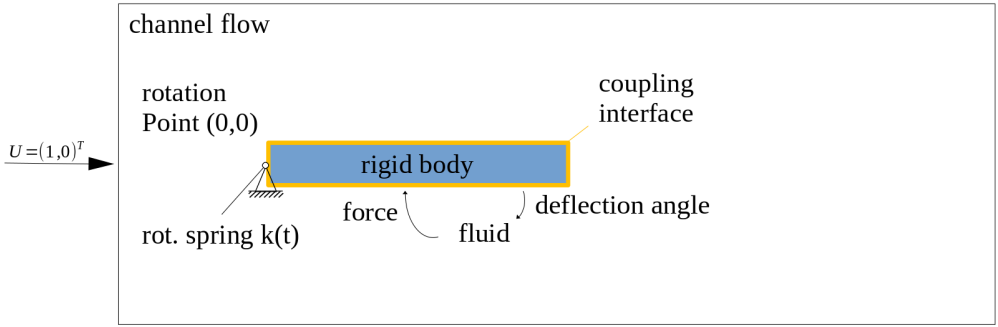
- Clone repository for foam-extend adapter

```
git clone https://github.com/precice/foam-extend-adapter
    ↪ $HOME/fe-adapter
cd $HOME/fe-adapter
git checkout foam-extend-4.1
./Allwmake
```

- Clone tutorial GitHub repository in your home folder

```
git clone https://github.com/precice/tutorials.git
    ↪ $HOME/tutorials
```

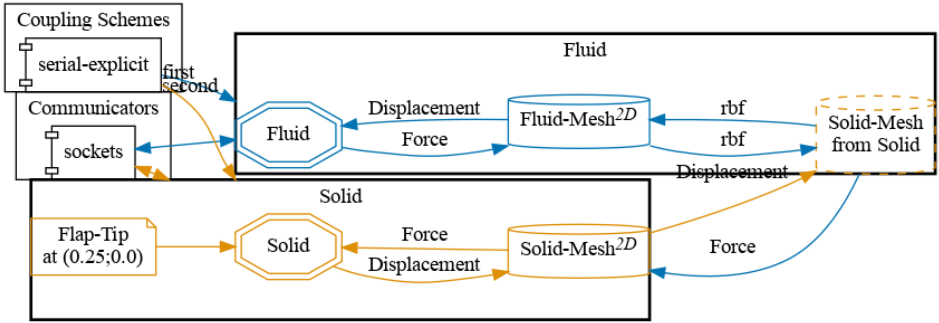
Case Setup


channel flow
rotation
Point (0,0)
coupling
interface
$U = (1,0)^T$
rigid body
force
deflection angle
rot. spring k(t)
fluid

preCICE Configuration Checking and Visualization Hands-On

- Go to respective folder
- Run configuration check
- Run Visualization

Case Setup - Visualization



Running the Case

- Building the rigid body solver

```
cd tutorials/quickstart/solid-cpp
cmake . && make
```

- Running the coupled simulation
  - Terminal window 1

```
cd tutorials/quickstart/solid-cpp
./run.sh
```

  - Terminal window 2

```
cd tutorials/quickstart/fluid-openfoam
./run.sh
# Alternative, in parallel: ./run.sh -parallel
```

Debugging the Case - Serial

- Terminal window 1

```
module load precice
cd tutorials/quickstart/solid-cpp
. ../../tools/log.sh
exec > >(tee --append "$LOGFILE") 2>&1

gdb ./rigid_body_solver
close_log
```

- Terminal window 2

```
module load gcc/11.5.0 openmpi/4.1.7 foam-extend/4.1-debug gdb git precice
source $WM_PROJECT_DIR/etc/bashrc
cd tutorials/quickstart/fluid-openfoam
blockMesh
gdb pimpleFoam
```

Debugging the Case - Parallel

Terminal window 1

```
1 module load gdb precice
2 cd tutorials/quickstart/solid-cpp
3 . ../../tools/log.sh
4 exec > >(tee --append "$LOGFILE") 2>&1
5
6 gdb ./rigid_body_solver
7 close_log
```

Terminal window 2

```
1 cd tutorials/quickstart/fluid-openfoam
2 module load gcc/11.5.0 openmpi/4.1.7 foam-extend/4.1-debug gdb precice
3 source $WM_PROJECT_DIR/etc/bashrc
4 %cd tutorials/quickstart/fluid-openfoam
5 blockMesh
6 decomposePar
7 tmpi 2 gdb pimpleFoam -parallel
8 reconstructPar
```

References

- Training Session "OpenFOAM Code Debugging and Profiling" 18th OpenFOAM Workshop
- https://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2022/lectureNotes/24_debugging.pdf
- https://openfoamwiki.net
- https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf
- R. Stallman, R. Pesch, and S. Shebs. Debugging with GDB: The GNU Source-Level debugger. GNU Press, Free Software Foundation Inc., 9th edition, 2002.

References

- https://wikis.ovgu.de/lss/doku.php?id=guide:qtcreator_for_openfoam
- https://github.com/Rvadrabade/Debugging-OpenFOAM-with-Visual-Studio-Code
- Damián, S. M., Giménez, J. M., Nigro, N. M. (2012). gdbOF: A debugging tool for OpenFOAM®. Advances in Engineering Software, 47(1), 17-23.
- https://github.com/FoamScience/foamUT
- https://users.ece.utexas.edu/~adnan/gdb-refcard.pdf

References

- https://precice.org/quickstart.html
- https://precice.org/tooling-performance-analysis.html
- https://precice.org/tooling-config-visualization.html
- https://sourceware.org/gdb/current/onlinedocs/gdb.html/Process-Record-and-Replay.html
- https://rr-project.org/
- https://github.com/rr-debugger/rr