GWDG
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

# OpenFOAM Debugging

Dr. Patrick Höhn (patrick.hoehn@gwdg.de)



November 13, 2026

# Agenda

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

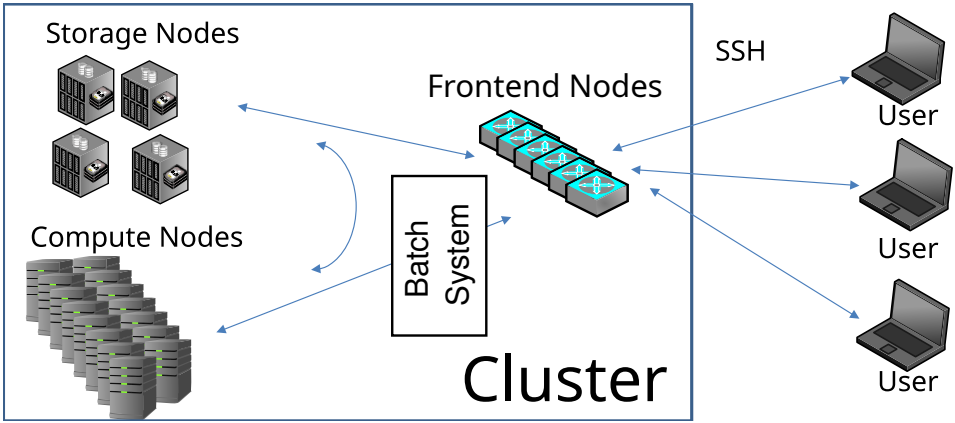
# Motivation

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

# Motivation debugging on clusters

- Limited local resources -> Execution on cluster
- Debugging in production environment
- Introduction to SLURM
- Workflows:
  - Job submission and debugging from Visual Studio Code
  - Attaching to a remote running job

# What is a Cluster?



# Basic Concepts

| | |
|---|---|
| Cluster | A collection of networked computers intended to provide compute capabilities. |
| Node | One of the computers in the cluster, also called host or server. |
| Frontend | Special node provided to interact with the cluster. `glogin.hpc.gwdg.de` in our case. |
| Job | Execution of a program consisting of one or several parallel tasks. |
| Partition | A pool of nodes of equal or similar type. |
| Batch System | Management system distributing job tasks across job slots. |

# Basic Concepts 2

**Serial job** Job consisting of one task using one job slot.

**SMP job** Job with shared memory parallelization, i.e. all processes need access to the memory of the same node.
$\Rightarrow$ several job slots **on the same node**.

**MPI job** Job with distributed memory parallelization, realized with MPI. Can use several job slots on several nodes. Needs to be started with a helper program, e.g., `mpirun or srun`.

# Telling Slurm what to do

- `srun` submits information on your job to Slurm.
  - What is to be done? (path to your program and required parameters)
  - What are its requirements? (e.g. which nodes, number of tasks, maximum runtime)
- Slurm matches the jobs requirements against the capabilities of the nodes.
- When suitable free resources are found, the job is started.
- Slurm prioritizes the jobs based on a number of factors.

# Your first job

Use `srun` to submit a job to Slurm.

`srun <program>`

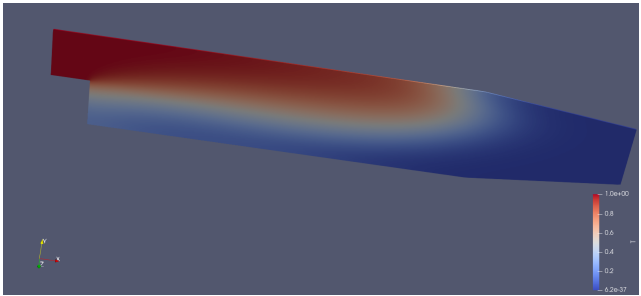
Example:

```
ssh uXXXX@glogin.hpc.gwdg.de
glogin1:27 12:53:50 ~ > hostname
glogin1
glogin1:27 12:53:53 ~ > srun -p medium96s:test hostname
c0005
glogin1:27 12:53:56 ~ > srun -p medium96s:test hostname -f
c0005.usr.hlrn.de
```

# Demo Problem: Pitzdaily

scalarTransportFoam:

$$\frac{\delta T}{\delta t} + \nabla \cdot (uT) - \nabla \cdot (\nabla D_T T) = S$$



# Two Modes of Operation

- Own local installation on Ubuntu 22.04
  - Described below
  - No limitations by cluster access
- Pre-installed on cluster
  - No additional effort
  - Availability depending on cluster usage
  - Restrictions regarding walltime

# Requirements - OpenFOAM Base Installation Ubuntu 22.04 LTS

## Install base packages

```
sudo apt-get update
sudo apt-get install git-core build-essential binutils-dev \
      cmake flex zlib1g-dev libncurses5-dev curl bison \
      libxt-dev rpm mercurial graphviz python3 python3-dev \
      paraview gdb
```

## Clone foam-extend repository

```
mkdir ~/foam
cd ~/foam
git clone http://git.code.sf.net/p/foam-extend/foam-extend-4.1 \
      foam-extend-4.1
```

# Requirements - OpenFOAM Base Installation Ubuntu 22.04 LTS

## Modify settings

```
cd ~/foam/foam-extend-4.1
source etc/bashrc
sed -i "150s/private List/public List/" ./src/foam/containers/Lists/PackedList/PackedList.H
sed -i -e 's,FULLDEBUG,FULLDEBUG \
     -I$(WM_THIRD_PARTY_DIR)/packages/openmpi-1.8.8/platforms/linux64GccDPInt32Opt/include,' \
     wmake/rules/linux64Gcc/cDebug
sed -i -e 's,FULLDEBUG,FULLDEBUG \
     -I$(WM_THIRD_PARTY_DIR)/packages/openmpi-1.8.8/platforms/linux64GccDPInt32Opt/include,' \
     wmake/rules/linux64Gcc/c++Debug
```

# Requirements - OpenFOAM Base Installation Ubuntu 22.04 LTS

## Define aliases

```
echo "alias fe41='source \
      \$HOME/foam/foam-extend-4.1/etc/bashrc'" >> ~/.bashrc
echo "alias fe41_debug='source \
      \$HOME/foam/foam-extend-4.1/etc/bashrc \
      WM_COMPILE_OPTION=Debug'" >> ~/.bashrc
```

## Compile foam-extend

```
cd ~/foam/foam-extend-4.1
source ~/foam/foam-extend-4.1/etc/bashrc
./Allwmake.firstInstall
WM_COMPILE_OPTION=Debug ./Allwmake
```

# Requirements - Tools

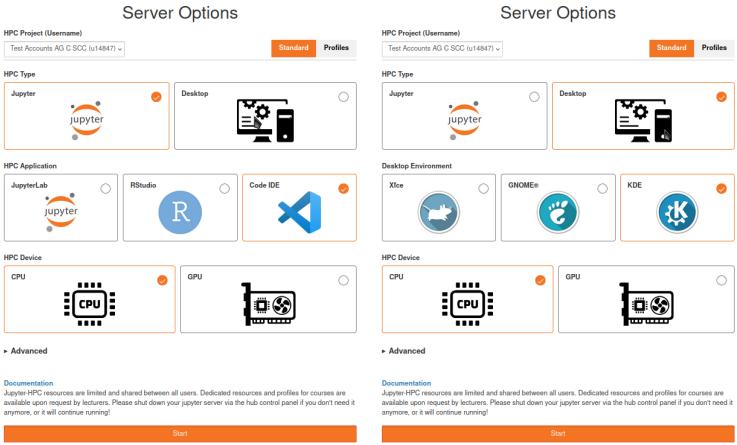
## Using package management system

```
sudo apt-get install valgrind
snap install code --classic
```

## Manual installation - gdbOF

```
source $HOME/foam/foam-extend-4.1/etc/bashrc \
      WM_COMPILE_OPTION=Debug
cd ~/foam/
git clone https://gitlab.com/flowcrunchpublic/gdbof.git gdbOF
cd gdbOF
./installgdbOF.sh
```

# The JupyterHPC Desktop




Link: https://jupyter.hpc.gwdg.de/

# 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)

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

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

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

# Environment variables

- `FOAM_ABORT`
  - This makes every error an abort and produces a stack-trace
  - When you're developing you don't want the solver to be tolerant
- `FOAM_SIGFPE`
  - This environment variable makes sure that Floating point exceptions are raised and properly handled
  - Some compilers generate code that calculates on even if the results are NaN
- `FOAM_SETNAN`
  - This environment variable helps finding uninitialized memory
  - Initializes allocated memory with NaN values, so without proper initialization a SIGFPE is triggered

# 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
```

# valgrind

- Open-Source framework for memory debugging, memory leak detection and profiling
- Dynamic binary instrumentation: no recompilation, works on any compiled binary
- Language agnostic: C, C++, Java, Python, Fortran, Ada, assembly, ...
- 100% coverage of user-space code including system libraries
- Reports exact location and cause of detected bugs
- Built-in tools: **Memcheck** (memory errors), **Helgrind**/**DRD** (thread races), **Cachegrind**/**Callgrind** (cache profiling), **Massif**/**DHAT** (heap profiling)

# rr - Record and Replay Debugger

- Developed by Mozilla Research
- Records a complete execution of a program
- Replay is fully deterministic - same execution every time
- Supports reverse debugging - step backwards through execution
- Useful for hard to reproduce and non deterministic bugs

# 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://s.gwdg.de/EGMmSj -O code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz
module load gdb/15.2
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
```

# Hands-On gdb

```
module load gcc/11.5.0 openmpi foam-extend/4.1-debug # or locally fe41_debug
source $WM_PROJECT_DIR/etc/bashrc
wget https://s.gwdg.de/EGMmSj -O code_openfoam_debugging.tar.gz -O
tar xzvf code_openfoam_debugging.tar.gz
module load gdb/15.2
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 | start your program with current argument list |
| run arglist | start your program with arglist |
| [c]ontinue | resumes execution of program until next breakpoint |
| ctrl-x ctrl-a | starting of Text User Inferface (TUI) Mode |
| tui enable | starting of Text User Inferface (TUI) Mode |
| ctrl-l | redraw TUI window |
| [b]ack[t]race | print trace of all frames in stack |
| backtrace n | print trace of n frames in stack |
| frame n | select frame number n |

# gdb 3

| **command** | **meaning** |
|---|---|
| [b]reak (file:)line<br>break (file:)line if var==value | set breakpoint at line (in file)<br>conditional breakpoint at line (in file)<br>if var equal value |
| clear | clear all breakpoints |
| whatis expr<br>ptype expr | show datatype of expression<br>show more details on datatype compared to whatis |
| watchpoint expr | set a watchpoint for expression expr |
| [s]tep<br>step n | run next line, stepping into function calls<br>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 foam-extend/4.1-debug # or locally fe41_debug
source $WM_PROJECT_DIR/etc/bashrc
cd $HOME
wget https://s.gwdg.de/EGMmSj -O code_openfoam_debugging.tar.gz -O
tar xzvf code_openfoam_debugging.tar.gz
module load gdb/15.2
cd ~/myScalarTransportFoam/
{vi|nano|emacs|gedit} # editor of your choice
wmake
# start debugging with gdb
```

# Hands-On rr

```
module load gcc/11.5.0 openmpi foam-extend/4.1-debug
source $WM_PROJECT_DIR/etc/bashrc
module load gdb/15.2
export PATH=$HOME/rr/build/bin:$PATH
export LD_LIBRARY_PATH=$HOME/capnproto/lib:$LD_LIBRARY_PATH
cd ~/pitzDaily && blockMesh
rr record -n $FOAM_USER_APPBIN/myScalarTransportFoam
rr replay -F
```

*Inside gdb (rr replay):*

```
break myScalarTransportFoam.C:57
continue
reverse-continue
reverse-next
```

# rr - Commands

| **command** | **meaning** |
|---|---|
| `rr record -n ./program` | record (`-n`: perf_event_paranoid=2 workaround) |
| `rr replay -F` | replay (`-F`: bypass freeze_on_smi check) |
| `reverse-continue` | run backwards to previous event |
| `reverse-next` | step backwards over function calls |
| `watch var` + `reverse-continue` | find where variable last changed |

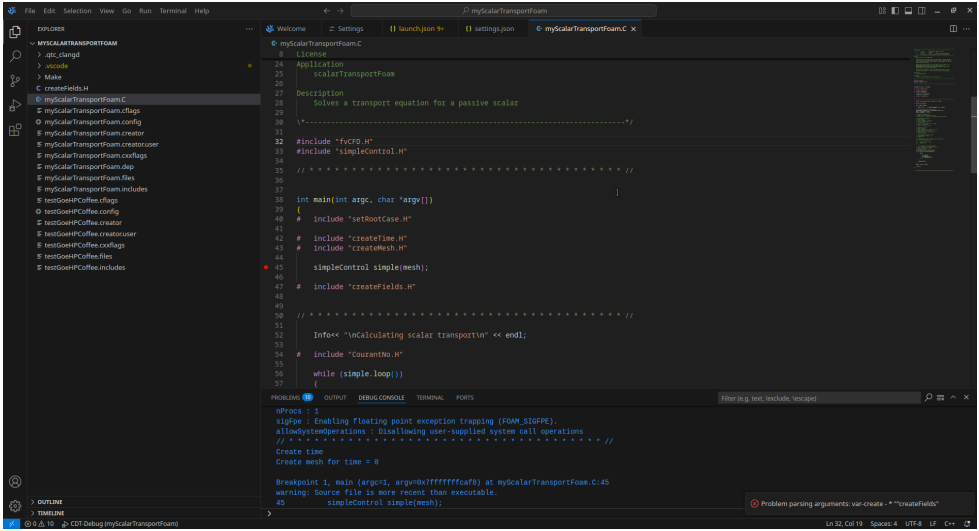
# Hands-On Valgrind

```
cd $FOAM_TUTORIALS/basic/scalarTransportFoam/pitzDaily/
blockMesh
valgrind --leak-check=full --show-leak-kinds=all \
    --track-origins=yes --log-file="memcheck.txt" \
    myScalarTransportFoam
```

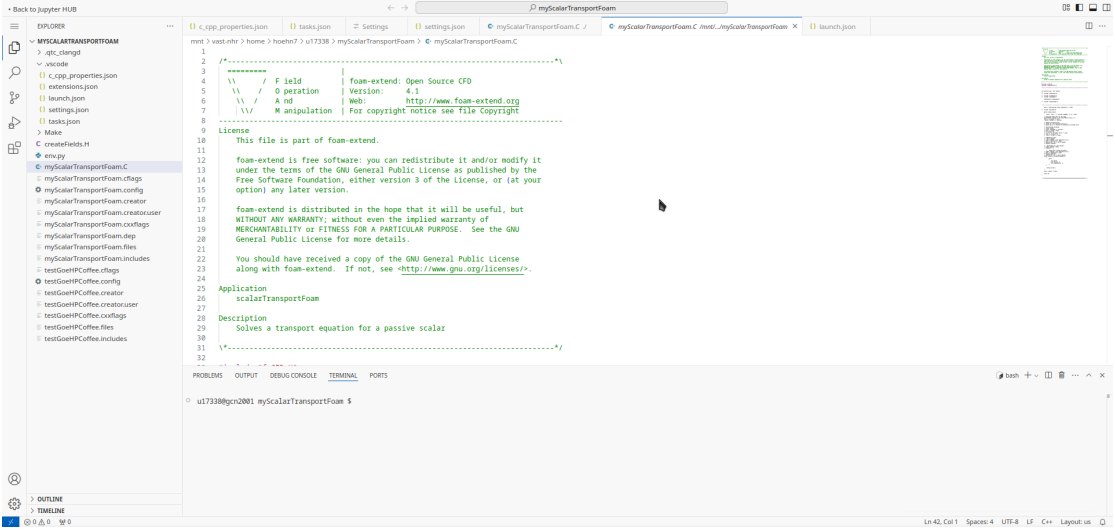
# General Instructions

- Open HPCDesktop in browser: `https://jupyter.hpc.gwdg.de/`
- Login via AcademicCloud
- Start VSCodium via web interface in next slide
- Load the respective modules: `module load gcc/11.5.0 openmpi/4.1.7 foam-extend/4.1-debug`
- Load the environment variables: `source $WM_PROJECT_DIR/etc/bashrc`
- Load the gdb module: module load gdb/15.2
- Open the project folder
- Run `python3.11 env.py` to update the editor settings.json
- Run `blockMesh` in terminal in Editor

# Start (Local)


File Edit Selection View Go Run Terminal Help
myScalarTransportFoam
EXPLORER
MYSCALARTRANSPORTFOAM
.qtc_clangd
.vscode
Make
createFields.H
myScalarTransportFoam.C
myScalarTransportFoam.cflags
myScalarTransportFoam.config
myScalarTransportFoam.creator
myScalarTransportFoam.creator.user
myScalarTransportFoam.cxxflags
myScalarTransportFoam.dep
myScalarTransportFoam.files
myScalarTransportFoam.includes
testGoeHPCoffee.cflags
testGoeHPCoffee.config
testGoeHPCoffee.creator
testGoeHPCoffee.creator.user
testGoeHPCoffee.cxxflags
testGoeHPCoffee.files
testGoeHPCoffee.includes
Welcome
Settings
launch.json 9+
settings.json
myScalarTransportFoam.C
myScalarTransportFoam.C
License
Application
scalarTransportFoam
Description
Solves a transport equation for a passive scalar
#include "fvCFD.H"
#include "simpleControl.H"
int main(int argc, char *argv[])
{
#include "setRootCase.H"
#include "createTime.H"
#include "createMesh.H"
simpleControl simple(mesh);
#include "createFields.H"
Info<< "\nCalculating scalar transport\n" << endl;
#include "CourantNo.H"
while (simple.loop())
{
PROBLEMS 10
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter (e.g. text, !exclude, \escape)
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
allowSystemOperations : Disallowing user-supplied system call operations
Create time
Create mesh for time = 0
Breakpoint 1, main (argc=1, argv=0x7fffffffcaf8) at myScalarTransportFoam.C:45
warning: Source file is more recent than executable.
45 simpleControl simple(mesh);
Problem parsing arguments: var-create - * ""createFields"
OUTLINE
TIMELINE
CDT:Debug (myScalarTransportFoam)
Ln 32, Col 19 Spaces: 4 UTF-8 LF C++

# Steps (HPC Desktop)



# tasks.json (Local and HPC Desktop)

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "shell",
            "label": "wmake-build",
            "command": "wmake",
            "problemMatcher": [],
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}
```

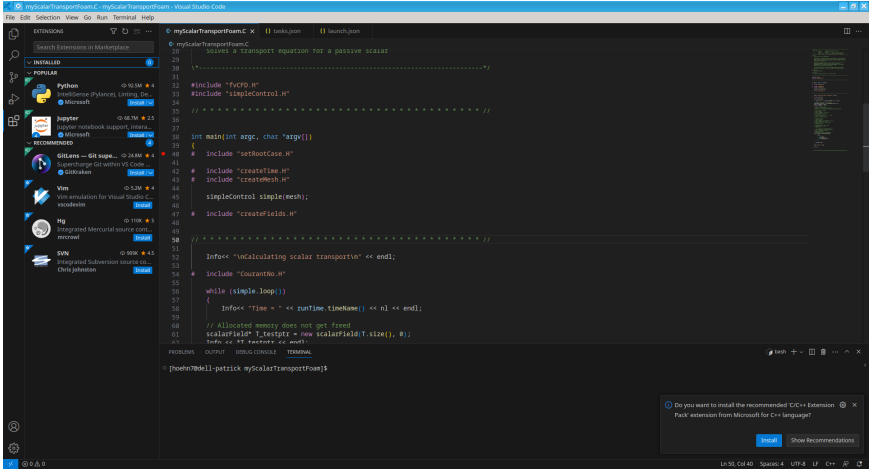
# launch.json (Local with VSCode)

```
1  {
2    "version": "0.2.0",
3    "configurations": [
4          {
5              "name": "OF-Debug",
6              "type": "cppdbg",
7              "request": "launch",
8              "program": "${env:FOAM_APPBIN}/myScalarTransportFoam",
9              "args": [],
10             "stopAtEntry": false,
11             "cwd": "${env:FOAM_TUTORIALS}/basic/scalarTransportFoam/pitzDaily/",
12             "environment": [],
13             "externalConsole": false,
14             "MIMode": "gdb",
15             "miDebuggerPath": "/usr/bin/gdb",
16             "setupCommands": [
17                 {
18                     "description": "Enable pretty-printing for gdb",
19                     "text": "-enable-pretty-printing",
20                     "ignoreFailures": true
21                 }
22             ],
23             "preLaunchTask": "wmake-build"
24         }
25     ]
26
27 }
```

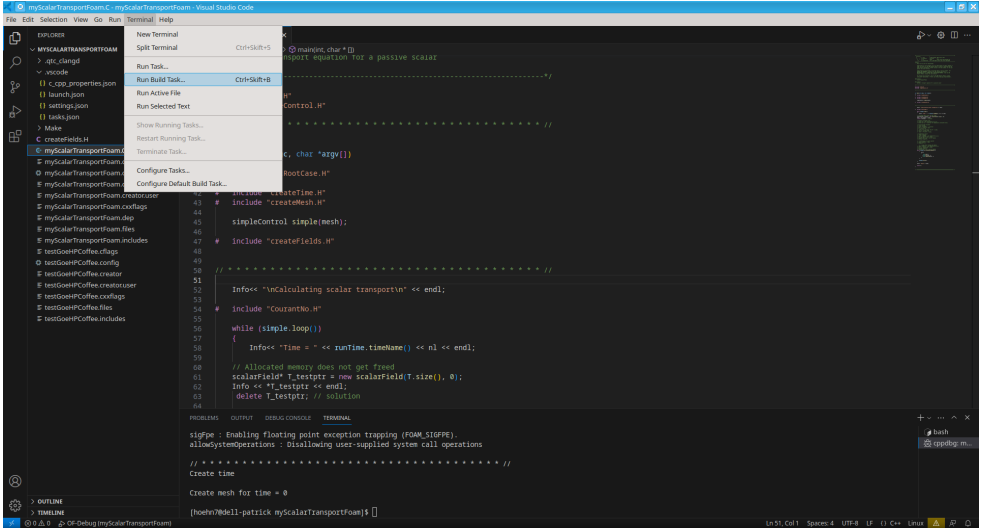
Description:
- 8: Location of executable program
- 9: Arguments to pass to the programm
- 10: Should program stop at the beginning of the main function?
- 11: Path to case folder
- 12: Environment variables to add to the environment of the program
- 14: What debugger should Visual Studio Code connect to?
- 15: Path of debugger. Don't know? Use which gdb in another terminal
- 16-22: Array of commands to execute in order to setup GDB
- 23: Task to be performed before debugging

# Steps (Local)

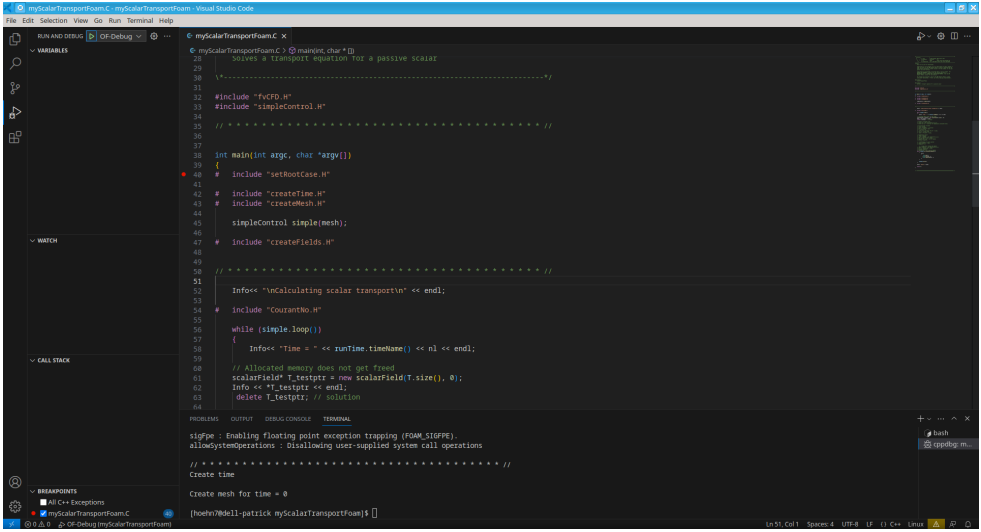
After first start after installation click "Install" to add c++ specific modules



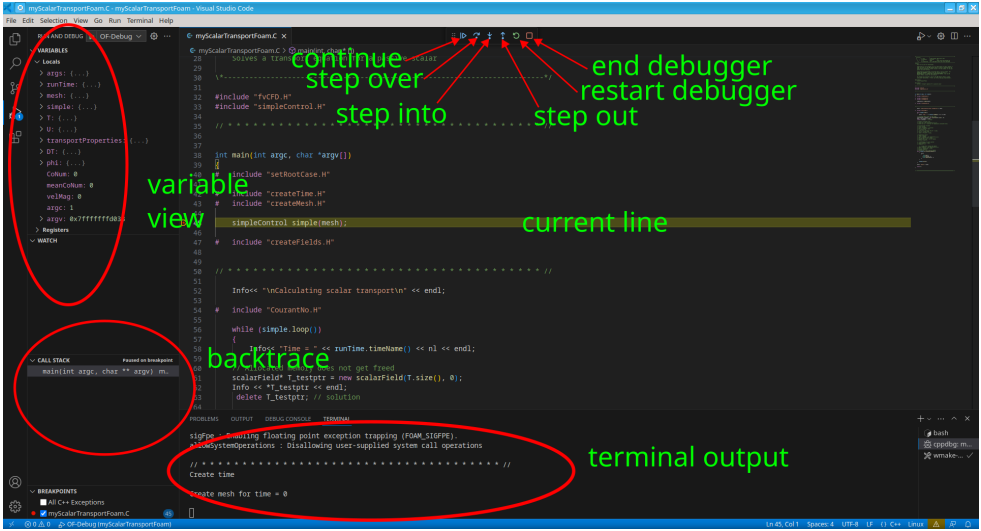
# Steps (Local)


myScalarTransportFoam.C - myScalarTransportFoam - Visual Studio Code
File Edit Selection View Go Run Terminal Help
New Terminal
Split Terminal Ctrl+Shift+5
Run Task...
Run Build Task... Ctrl+Shift+B
Run Active File
Run Selected Text
Show Running Tasks...
Restart Running Task...
Terminate Task...
Configure Tasks...
Configure Default Build Task...
EXPLORER
MYSCALARTRANSPORTFOAM
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
allowSystemOperations : Disallowing user-supplied system call operations
Create time
Create mesh for time = 0
OUTLINE
TIMELINE

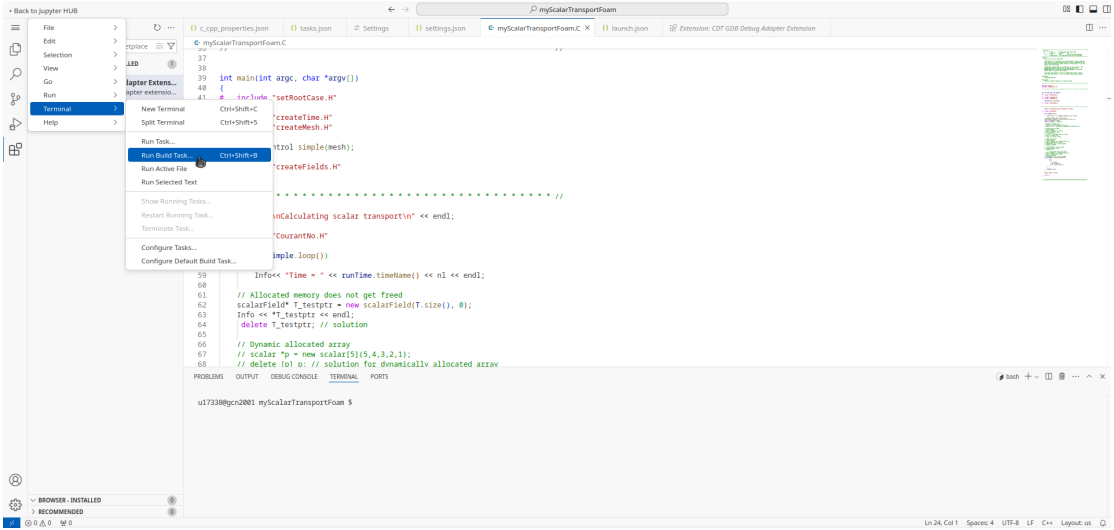
# Steps (Local)


myScalarTransportFoam.C - myScalarTransportFoam - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN AND DEBUG
OF-Debug
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
All C++ Exceptions
myScalarTransportFoam.C
OF-Debug (myScalarTransportFoam)
myScalarTransportFoam.C
myScalarTransportFoam.C > main(int, char * [])
solves a transport equation for a passive scalar
#include "fvCFD.H"
#include "simpleControl.H"
int main(int argc, char *argv[])
#include "setRootCase.H"
#include "createTime.H"
#include "createMesh.H"
simpleControl simple(mesh);
#include "createFields.H"
Info<< "\nCalculating scalar transport\n" << endl;
#include "CourantNo.H"
while (simple.loop())
Info<< "Time = " << runTime.timeName() << nl << endl;
// Allocated memory does not get freed
scalarField* T_testptr = new scalarField(T.size(), 0);
Info << *T_testptr << endl;
delete T_testptr; // solution
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
allowSystemOperations : Disallowing user-supplied system call operations
Create time
Create mesh for time = 0
[hoehn7@dell-patrick myScalarTransportFoam]$
bash
cppdbg: m...
Ln 51, Col 1 Spaces: 4 UTF-8 LF () C++ Linux

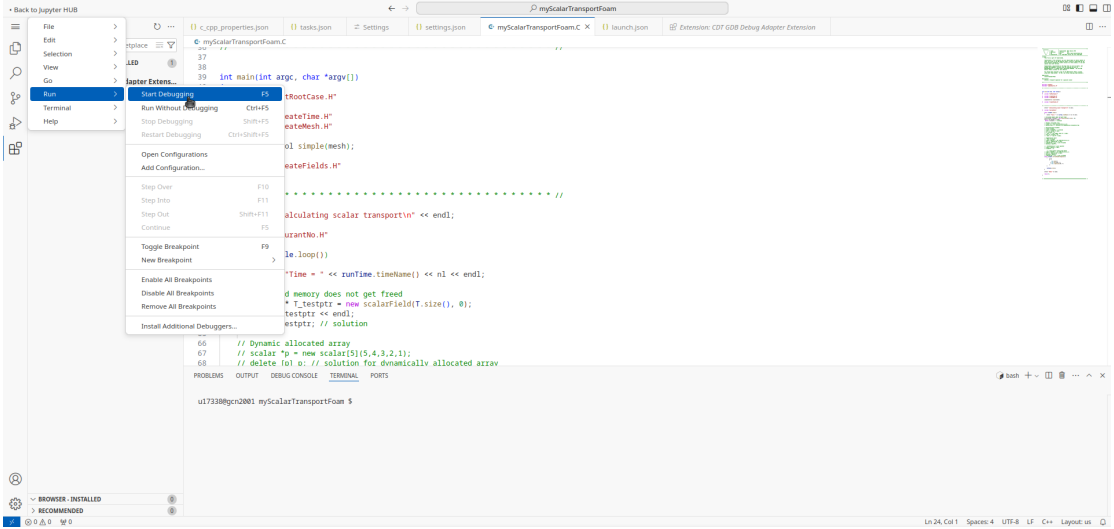
# Steps (Local)


continue
step over
step into
step out
end debugger
restart debugger
variable view
current line
backtrace
terminal output

# Steps (HPC Desktop)

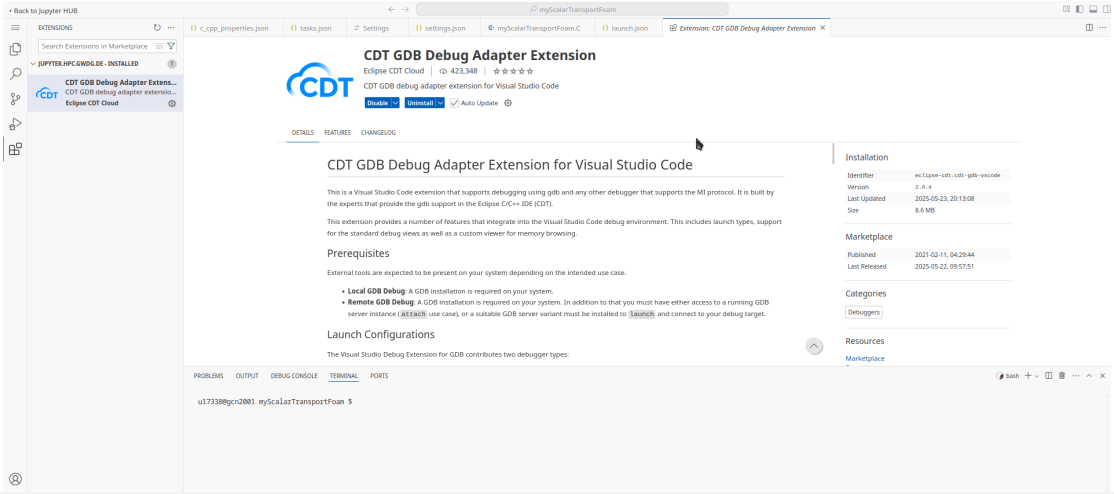


# Steps (HPC Desktop)



# Steps (HPC Desktop)

## Install the GDB Debug Adapter extension when prompted



# Steps (HPC Desktop)



# 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
git clone https://github.com/Azrael3000/tmpi.git
echo "export PATH=\$HOME/tmpi:\$PATH" >> $HOME/.bashrc
```

# Hands-On tmpi

- Usage:

```
module load gcc/11.5.0 openmpi
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
```

# Hands-On tmpi

```
module load gcc/11.5.0 openmpi foam-extend/4.1-debug
CORRECT_MPIRUN=$(which mpirun)
source $WM_PROJECT_DIR/etc/bashrc
wget https://s.gwdg.de/EGMmSj -O code_openfoam_debugging.tar.gz
tar xzvf code_openfoam_debugging.tar.gz && cd $HOME/pitzDaily
module load gdb/15.2
blockMesh && decomposePar
MPIRUN=$CORRECT_MPIRUN tmpi 4 gdb -q myScalarTransportFoam
```

*Inside gdb:*

```
run -parallel
```

# mpigdb

- Lightweight wrapper around GDB for MPI programs
- Opens one GDB instance per MPI rank in separate terminal windows
- Each rank can be debugged independently or in sync
- Provides `mpi*` convenience commands for operating across all ranks

# Hands-On mpigdb

```
module load gcc/11.5.0 openmpi foam-extend/4.1-debug gdb/15.2
CORRECT_MPIEXEC=$(which mpiexec)
CORRECT_GDBSERVER=$(which gdbserver)
CORRECT_GDB=$(which gdb)
source $WM_PROJECT_DIR/etc/bashrc
export PATH=$(dirname $CORRECT_MPIEXEC):$HOME/mpigdb/target/debug:$PATH
cd ~/pitzDaily && blockMesh && decomposePar
mpigdb --mpigdb_gdb $CORRECT_GDB \
     --mpigdb_gdbserver $CORRECT_GDBSERVER \
     -np 4 -- $FOAM_USER_APPBIN/myScalarTransportFoam -parallel
```

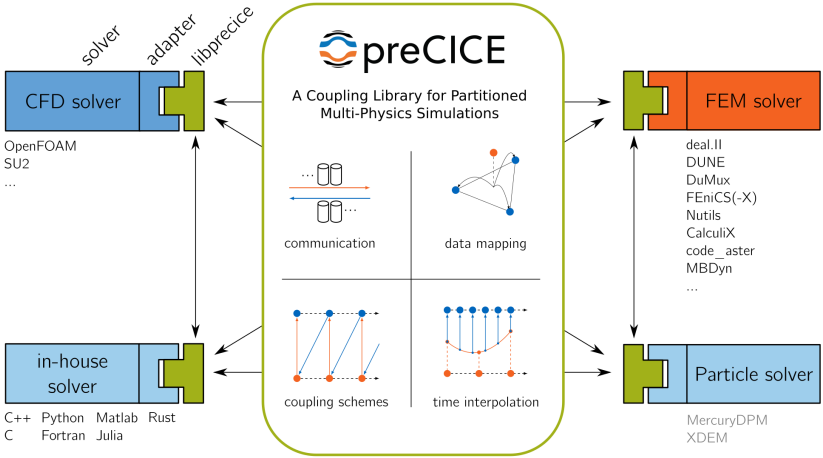
*Inside gdb:*

```
mpib main
mpic
mpip argc
```

# mpigdb - Commands

| **command** | **meaning** |
|---|---|
| `mpib main` | set breakpoint on all ranks |
| `mpic` | continue all ranks |
| `mpip var` | print variable on all ranks |
| `thread apply all bt` | backtrace on all ranks |

# preCICE fundamentals



# Important preCICE Tooling

- Configuration Check
- Config Visualization

# 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 libgirepository-2.0-dev
    ↪ gcc libcairo2-dev pkg-config python3-dev gir1.2-gtk-4.0
```

- Install the python packages:

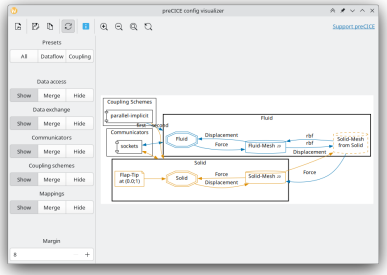
```
pip install precice-config-visualizer
# For GUI addtionally install:
pip install precice-config-visualizer-gui
```

# 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
```

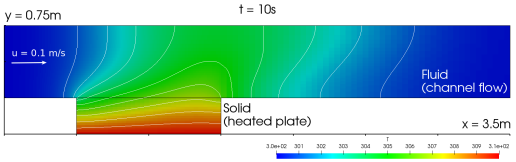
- Customize the graph further first generate the dot graph:

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

# Case: Flow Over a Heated Plate

Setup

- Conjugate Heat Transfer (CHT) benchmark
- Fluid flows over a solid heated plate
- Heat exchanged at fluid-solid interface




Solvers

- Fluid: `buoyantPimpleFoam`
- Solid: `laplacianFoam`

Exchanged data

- Fluid $\rightarrow$ Solid: Heat Flux
- Solid $\rightarrow$ Fluid: Temperature

# gdb on complete preCICE workflow

*Terminal 1 - Fluid side*

```
module purge
module load gcc/11.5.0 openmpi/4.1.7
source ~/OpenFOAM/OpenFOAM-v2412/etc/bashrc WM_COMPILE_OPTION=Debug
module load precice/3.2.0
module load gdb/15.2
cd ~/tutorials/flow-over-heated-plate/fluid-openfoam
gdb ~/OpenFOAM/OpenFOAM-v2412/platforms/linux64GccDPInt32Debug/bin/buoyantPimpleFoam
```

*Inside gdb:*

```
(gdb) break buoyantPimpleFoam.C:1
(gdb) run -case ~/tutorials/flow-over-heated-plate/fluid-openfoam
(gdb) continue
```

# gdb on complete preCICE workflow

*Terminal 2 - Solid side*

```
module load gcc/11.5.0 openmpi/4.1.7 foam-extend/4.1-debug
source $WM_PROJECT_DIR/etc/bashrc
module load precice/3.2.0
module load gdb/15.2
cd ~/tutorials/flow-over-heated-plate/solid-openfoam
gdb $(which laplacianFoam)
```

*Inside gdb:*

```
(gdb) break laplacianFoam.C:1
(gdb) run -case ~/tutorials/flow-over-heated-plate/solid-openfoam
(gdb) continue
```

# Wrap-Up

- Serial debugging with `gdb`, `valgrind`, and `rr` demonstrated on `scalarTransportFoam` / pitzDaily
- IDE workflow: VSCode with `tasks.json` + `launch.json` on JupyterHPC
- Parallel debugging: `tmpi` + GDB and `mpigdb` for per-rank MPI inspection
- preCICE config check, visualization, and GDB across coupled fluid-solid solvers demonstrated

# 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:qtcreater_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
- https://github.com/robertu94/mpigdb

# 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
- https://openfoamwiki.net/images/9/99/BernhardGschaider-OFW16_swakPyFoam4Developers.pdf