FORTRANTESTGENERATOR: AUTOMATIC AND FLEXIBLE UNIT TEST GENERATION FOR LEGACY HPC CODE

CHRISTIAN HOVY
UNIVERSITÄT HAMBURG
christian.hovy@uni-hamburg.de

JULIAN KUNKEL
GERMAN CLIMATE COMPUTING CENTER
kunkel@dkrz.de

ISC HPC 2017, Poster Presentation
Frankfurt, Germany

2017-06-20
Obstacles for Testing

• Limited knowledge of expected results
• Effort of setting up of test data for unit testing
  – Large and complicated data layouts
    (i.e., unstructured grids, cache blocking etc.)
• Long run times
• Parallelism
• Legacy Code

Example snippet from the earth system model ICON
[https://www.mpimet.mpg.de/en/science/models/icon]
FortranTestGenerator (FTG)

- Automatic unit test generation for subroutines of Fortran applications
- **Strategy: Capture & Replay**
  - Extract a set of input data by running the original application
  - Run subroutine independently with one process feeding captured data
- **Features**
  - Instrumentation for capture code
    - FTG generates code for serializing and storing a subroutine's input data
    - Developer can define an event upon which the input data is captured
    - FTG captures only necessary variables using static code analysis
  - Test driver creation: loads this data and runs the subroutine (replay code)
    - Checks and modification of data can be done by the developer
  - FTG uses templates to generate instrumentation and test driver
    - Customizable
Capture & Replay with MPI

• Results of subroutines may depend on MPI communication
• Enables unit testing with only a single process
• Capture/Replay MPI communication (Ongoing effort)
  – Use either HDF5 to store data or create code to initialize data
  – Replay using a dummy MPI implementation
    • Feeds in recorded responses from MPI
    • In the order it has been recorded
  – Python editor to add / modify input
Process $n=i$  

Process $n=j$  

SubroutineUnderTest execution $x$  

SubroutineUnderTest execution $y=f(x, a_1, a_2, a_3, \ldots)$  

SubroutineUnderTest execution $z=f(x, b_1, b_2, b_3, \ldots)$