codee

Codee Training:
Write Accelerated Code at Expert Level

Codee: Automated Analysis of Large-Scale Fortran/C/C++ Codes

NERSC Codee Training Series

September 5-6, 2024

Schedule

Day 1 (Thursday 5th, 9:00 - 12:30 PDT)

Codee: Automated Code Inspection for Modernization and Optimization

- Lecture:
  - *Codee's command-line tool*
  - *Open Catalog of Best Practices for Fortran/C/C++ Modernization and Optimization for CPU and GPU*
- Demo using Fortran:
  - *HIMENO modernization*
  - *HIMENO optimization through GPU parallelism*
- Demo using C/C++:
  - *MATMUL optimization through CPU parallelism*
- Hands-on: PI, MATMUL, COULOMB, HIMENO

Day 2 (Friday 6th, 9:00 - 12:30 PDT)

Codee: Automated Analysis of Large-Scale Fortran/C/C++ Codes

- Lecture:
  - *Codee's command-line tool using compilation databases*
  - *Automated testing of large codes using Codee on Perlmutter*
  - *Use case: Optimizing the Weather Research and Forecasting Model with OpenMP Offload and Codee*
- Demo using Fortran:
  - *Putting it all together with HYCOM*
- Demo using C/C++:
  - *Putting it all together with MBedTLS*
- Hands-on: HYCOM, NUCCOR, ATMUX, LULESHmk, MBedTLS
- Bring your own applications!

Extracting compilation commands from large codes

- Codee requires the compiler invocation for each file to analyze.
- Simulation codes usually contain a large numbers of files.
- **Compilation Database Format:**
  - Part of the LLVM ecosystem.
  - Details the **compilation invocation for each file**.
  - Used by tooling such as clang, clang-tidy, etc.
  - Codee can ingest a compilation database to **process all files in a project** and generate reports **aggregating the results of the codebase**.

Compilation Database Format

```
$ cat compile_commands.json
[
 {
   "arguments": ["/usr/bin/gfortran", "-DREAL8", "-DENDIAN_IO", "-DTIMER", "-DRELO", "-DEOS_SIG0",
"-DEOS_7T", "-fPIC", "-fno-second-underscore", "-O2", "-fdefault-real-8", "-fdefault-double-8", "-c",
"mod_dimensions.F90"],
   "directory": "HYCOM/src",
   "file": "HYCOM/src/mod_dimensions.F90"
 },
 {
   "arguments": ["/usr/bin/gfortran", "-DREAL8", "-DENDIAN_IO", "-DTIMER", "-DRELO", "-DEOS_SIG0",
"-DEOS_7T", "-fPIC", "-fno-second-underscore", "-O2", "-fdefault-real-8", "-fdefault-double-8", "-c",
"mod_xc.F90"],
   "directory": "HYCOM/src",
   "file": "HYCOM/src/mod_xc.F90"
 },
 <...>
```

Generating a Compilation Database

- Depends on the build system.
- **CMake:**
  - `$ cmake [...] -DCMAKE_EXPORT_COMPILE_COMMANDS=ON .`
- **Makefiles:**
  - Not natively supported.
  - We recommend using bear.
  - `$ bear -- make`
- Both approaches generate a **`compile_commands.json` file**.

codee

Using the Compilation Database with Codee

Common option:

```
  -p, --compile-commands <filepath>
          Load the analysis targets options from the specified path. The path
          can either be a JSON compilation database file (compile_commands.json)
          or a directory that contains a file named 'compile_commands.json'
```

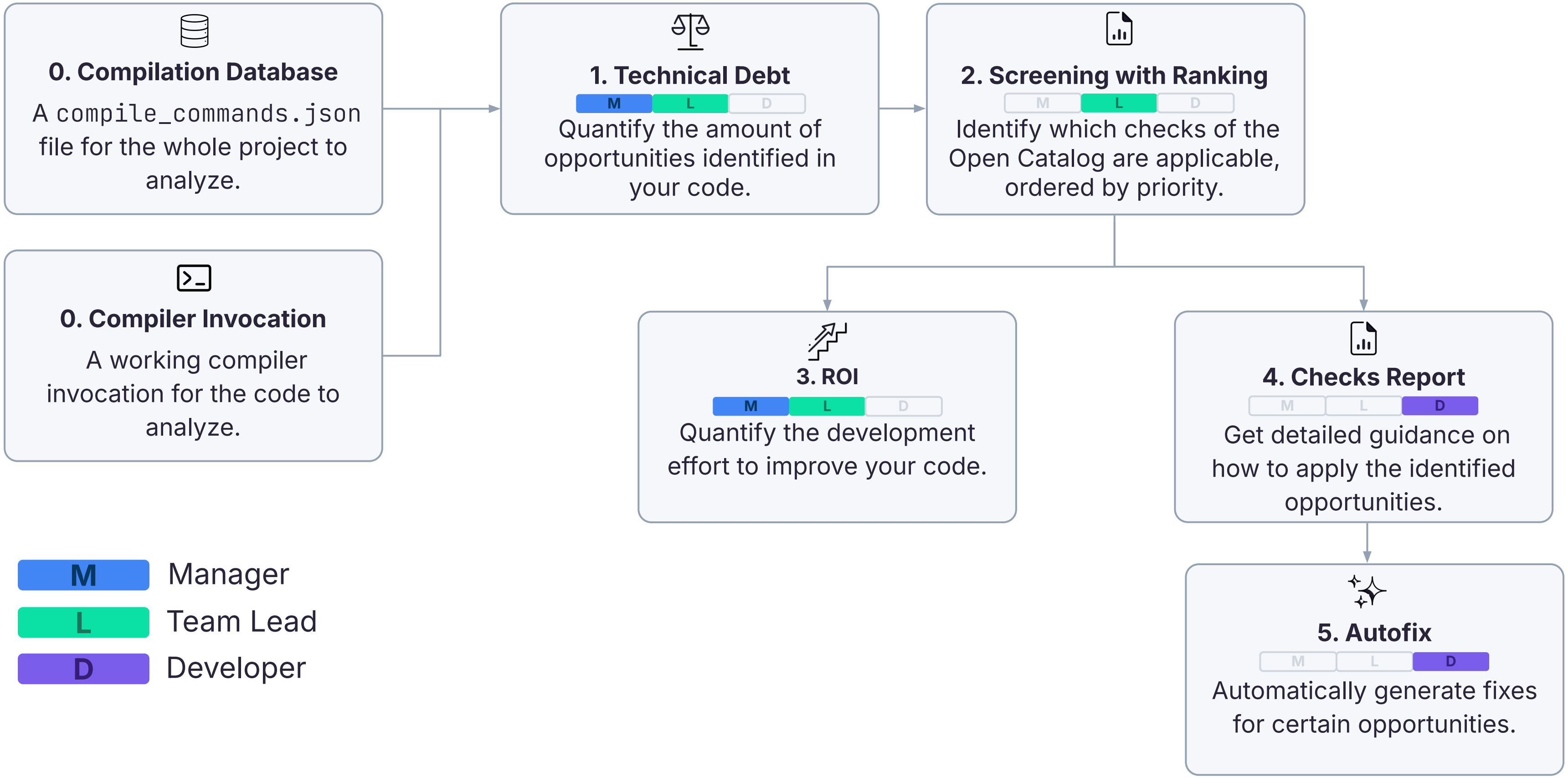
Compiler Invocation:

```
$ codee <command> -- <compiler invocation>
```

Compilation Database Invocation:

```
$ codee <command> -p compile_commands.json
```

Codee: Suggested Advanced Workflow



0. Compilation Database

A `compile_commands.json` file for the whole project to analyze.

0. Compiler Invocation

A working compiler invocation for the code to analyze.

1. Technical Debt

M **L** D

Quantify the amount of opportunities identified in your code.

2. Screening with Ranking

M **L** D

Identify which checks of the Open Catalog are applicable, ordered by priority.

3. ROI

M **L** D

Quantify the development effort to improve your code.

4. Checks Report

M L **D**

Get detailed guidance on how to apply the identified opportunities.

5. Autofix

M L **D**

Automatically generate fixes for certain opportunities.

M Manager

L Team Lead

D Developer

SUGGESTED ADVANCED WORKFLOW

0. Compilation Database

```
$ bear -- make ARCH=codee TYPE=demo
gfortran -DREAL8 -DENDIAN_IO -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -fPIC -fno-second-underscore -O2
-fdefault-real-8 -fdefault-double-8 -c mod_dimensions.F90
gfortran -DREAL8 -DENDIAN_IO -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -fPIC -fno-second-underscore -O2
-fdefault-real-8 -fdefault-double-8 -c mod_xc.F90
gfortran -DREAL8 -DENDIAN_IO -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -fPIC -fno-second-underscore -O2
-fdefault-real-8 -fdefault-double-8 -c mod_za.F90

<...>

gfortran -DREAL8 -DENDIAN_IO -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -fPIC -fno-second-underscore -O2
-fdefault-real-8 -fdefault-double-8 -c hycom.F90
gfortran  -fPIC -fno-second-underscore -O2 -fdefault-real-8 -fdefault-double-8 -o hycom  hycom.o
mod_dimensions.o mod_xc.o mod_za.o mod_cb_arrays.o mod_pipe.o mod_incupd.o mod_floats.o mod_stokes.o
mod_tides.o mod_mean.o mod_archiv.o mod_tsadvc.o mod_momtum.o mod_barotp.o mod_asselin.o mod_restart.o
mod_hycom.o bigrid.o blkdat.o  cnuity.o convec.o diapfl.o dpthuv.o  dpudpv.o forfun.o  geopar.o hybgen.o
icloan.o inicon.o inigiss.o inikpp.o   inimy.o latbdy.o matinv.o mxkprf.o  mxkrt.o  mxkrtm.o  mxpwp.o
overtn.o poflat.o  prtmsk.o  psmoo.o thermf.o trcupd.o machine.o  wtime.o machi_c.o  isnan.o s8gefs.o

$ grep "file" compile_commands.json | wc -l
50
```

SUGGESTED ADVANCED WORKFLOW

1-5. Codee Reports

```
$ codee technical-debt -p compile_commands.json
[ 1/50] mod_dimensions.F90 ... Done
[ 2/50] mod_xc.F90 ... Done
[ 3/50] mod_za.F90 ... Done
[ 4/50] mod_cb_arrays.F90 ... Done
[ 5/50] mod_stokes.F90 ... Done
[ 6/50] mod_pipe.F90 ... Done
[ 7/50] mod_incupd.F90 ... Done
[ 8/50] mod_floats.F90 ... Done
[ 9/50] mod_tides.F90 ... Done
<...>

$ codee screening -p compile_commands.json

$ codee roi -p compile_commands.json

$ codee checks -p compile_commands.json

...
```

SUGGESTED ADVANCED WORKFLOW

Focus the Analysis: Select a subset of code (I)

Filters:

```
$ codee checks HYCOM/src/mod_stokes.F90 -p compile_commands.json
```

Filter by file

```
$ codee checks HYCOM/src/mod_stokes.F90:stokes_vertical_j -p compile_commands.json
```

Filter by function

```
$ codee checks HYCOM/src/mod_stokes.F90:675 -p compile_commands.json
```

Filter by loop

Common options:

```
--lang <language>
    Filter the input files by language (C, C++, Fortran)

--exclude <file|directory>
    Skip the specified file or directory. `--exclude` may be set several
    times

--show-progress, --show-progress=<none|files|functions>
    Show how the analysis progresses by printing a message for each input
    file or function (defaults to `files`)
```

SUGGESTED ADVANCED WORKFLOW

Focus the Analysis: Select a subset of code (II)

```
$ codee technical-debt -p compile_commands.json mod_hycom.F90 mod_archiv.F90 mod_mean.F90 mod_stokes.F90
mod_za.F90

<...>

[1/5] mod_za.F90 ... Done
[2/5] mod_stokes.F90 ... Done
[3/5] mod_mean.F90 ... Done
[4/5] mod_archiv.F90 ... Done
[5/5] mod_hycom.F90 ... Done

TECHNICAL DEBT REPORT

This report quantifies the technical debt associated with the modernization of legacy code by assessing the
extent of refactoring required for language constructs. The score is determined based on the number of
language constructs necessitating refactoring to bring the source code up to modern standards.
Additionally, the metric identifies the impacted source code segments, detailing affected files, functions,
and loops.

Score Affected files Affected functions Affected loops
----- -------------- ------------------ --------------
106   5              17                 14

<...>
```

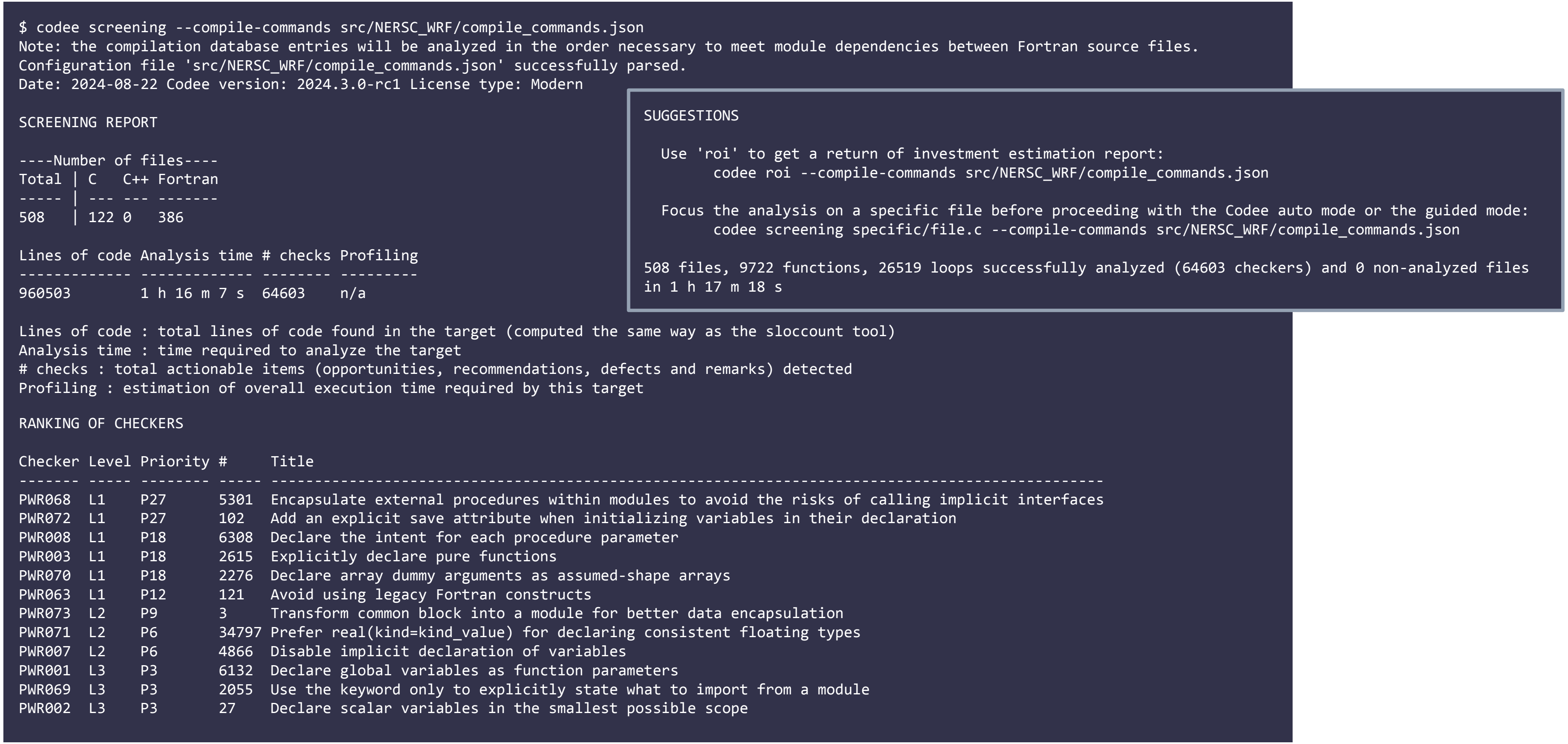
5 files analyzed (out of 50)

USE CASE: FORTRAN MODERNIZATION

Success Stories using Open Source Software

Code	Domain	Metrics with Codee 2024.3.0 (Aug. 2024)
CP2K 1.3M lines of code	Quantum chemistry and solid state physics software package	1344 files, 5431 functions, 9549 loops successfully analyzed and 17 non-analyzed files in 13 m 33 s
OpenRadioss 1.1M lines of code	Finite element solver for dynamic event analysis	3477 files, 6541 functions, 39636 loops successfully analyzed and 0 non-analyzed files in 31 m 6 s
WRF 960K lines of code	Weather Research and Forecasting	508 files, 9722 functions, 26519 loops successfully analyzed (64603 checkers) and 0 non-analyzed files in 1 h 17 m 18 s
ICON 646K lines of code	Weather, climate, and environmental prediction	1143 files, 6959 functions, 7801 loops successfully analyzed (6098 checkers) and 7 non-analyzed files in 9 m 34 s
SIESTA 398K lines of code	First-principles Materials Simulation	967 files, 2956 functions, 2254 loops successfully analyzed (3291 checkers) and 25 non-analyzed files in 2 m 16 s
PHASTA 64K lines of code	Parallel Hierarchic Adaptive Stabilized Transient Analysis of compressible and incompressible Navier Stokes equations	284 files, 608 functions, 1086 loops successfully analyzed (1420 checkers) and 0 non-analyzed files in 6 m 35 s
HYCOM 44K lines of code	HYbrid Coordinate Ocean Model	50 files, 251 functions, 2058 loops successfully analyzed (2965 checkers) and 0 non-analyzed files in 53.85 s
EAP-patterns 4K lines of code	Patterns from an Eulerian cell AMR application	12 files, 88 functions, 164 loops successfully analyzed and 0 non-analyzed files in 1037 ms

WRF: Screening with Ranking Report



```
$ codee screening --compile-commands src/NERSC_WRF/compile_commands.json
Note: the compilation database entries will be analyzed in the order necessary to meet module dependencies between Fortran source files.
Configuration file 'src/NERSC_WRF/compile_commands.json' successfully parsed.
Date: 2024-08-22 Codee version: 2024.3.0-rc1 License type: Modern

SCREENING REPORT

----Number of files----
Total | C   C++ Fortran
----- | --- --- -------
508   | 122 0   386

Lines of code Analysis time # checks Profiling
------------- ------------- -------- ---------
960503        1 h 16 m 7 s  64603    n/a

Lines of code : total lines of code found in the target (computed the same way as the sloccount tool)
Analysis time : time required to analyze the target
# checks : total actionable items (opportunities, recommendations, defects and remarks) detected
Profiling : estimation of overall execution time required by this target

RANKING OF CHECKERS

Checker Level Priority #     Title
------- ----- -------- ----- ------------------------------------------------------------------------------------------------
PWR068  L1    P27      5301  Encapsulate external procedures within modules to avoid the risks of calling implicit interfaces
PWR072  L1    P27      102   Add an explicit save attribute when initializing variables in their declaration
PWR008  L1    P18      6308  Declare the intent for each procedure parameter
PWR003  L1    P18      2615  Explicitly declare pure functions
PWR070  L1    P18      2276  Declare array dummy arguments as assumed-shape arrays
PWR063  L1    P12      121   Avoid using legacy Fortran constructs
PWR073  L2    P9       3     Transform common block into a module for better data encapsulation
PWR071  L2    P6       34797 Prefer real(kind=kind_value) for declaring consistent floating types
PWR007  L2    P6       4866  Disable implicit declaration of variables
PWR001  L3    P3       6132  Declare global variables as function parameters
PWR069  L3    P3       2055  Use the keyword only to explicitly state what to import from a module
PWR002  L3    P3       27    Declare scalar variables in the smallest possible scope
```

```
SUGGESTIONS

  Use 'roi' to get a return of investment estimation report:
        codee roi --compile-commands src/NERSC_WRF/compile_commands.json

  Focus the analysis on a specific file before proceeding with the Codee auto mode or the guided mode:
        codee screening specific/file.c --compile-commands src/NERSC_WRF/compile_commands.json

508 files, 9722 functions, 26519 loops successfully analyzed (64603 checkers) and 0 non-analyzed files
in 1 h 17 m 18 s
```

Main Takeaways

- **Codee is a production-ready tool for automated testing of Fortran/C/C++ code**, designed for integration with CI/CD, IDEs and programming environments.
- Current **success stories of Codee** include: WRF, ICON, PHASTA, CP2K, HYCOM, OpenRadioss...
- Identify how your **build system** can generate a **compilation database**:
  - **CMake:** `cmake -DCMAKE_EXPORT_COMPILE_COMMANDS=ON [...]`
  - **Makefiles:** `bear -- make [...]`
- Provide the **compilation database to Codee** to seamlessly **analyze complete projects** and **jump from one source file to another**:
  - `codee -p compile_commands.json`
- **Focus the analysis** to fit your needs (e.g., **source file filters**):
  - `codee file.f90:function`
  - `codee --exclude`

codee

Automated Code Inspection for
Modernization and Optimization

www.codee.com

info@codee.com

Subscribe: codee.com/newsletter/

Spain

codee_com

in /codee-com/