

Quickstart - Fortran Performance with Himeno

1. Produce the Screening Report of Codee (screening command):

```
codee screening himeno.f90 --target-arch cpu -- -O3
```

```
Date: 2024-04-25 Codee version: 2024.2.2
Compiler flags: -O3

[Fortran] target compiler: <none> (Compiler Agnostic Mode)

[1/1] himeno.f90 ... Done

SCREENING REPORT

---Number of files---
Total | C C++ Fortran
-----|-----
1      | 0 0 1

Lines of code Analysis time # checks Profiling
-----|-----
214      201 ms      11      n/a

CHECKS PER CATEGORY AND PRIORITY LEVELS

| -----Checks per category----- | Priority |
| Scalar Control Memory Vector Multi Offload Quality | L1 L2 L3 |
| -----|-----|-----|-----|-----|-----|
| 0      0      2      1      2      n/a      6      | 2 1 8 |

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
-----|-----|-----|-----
PWR054 L1 P12 1 Consider applying vectorization to scalar reduction loop
PWR063 L1 P12 1 Avoid using legacy Fortran constructs
PWR051 L2 P6 1 Consider applying multithreading parallelism to scalar reduction loop
PWR001 L3 P3 5 Declare global variables as function parameters
RMK001 L3 P3 1 Loop nesting that might benefit from hybrid parallelization using...
PWR035 L3 P2 2 Avoid non-consecutive array access to improve performance

SUGGESTIONS

Use 'checks' to find out details about the detected checks:
codee checks himeno.f90 --target-arch cpu -- -O3

Consider using Codee with a target compiler in order to filter out optimizations that are already
applied by your compiler. For example, for GCC:
codee screening --target-compiler-fc gfortran himeno.f90 --target-arch cpu -- -O3

1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 201 ms
```

2. Follow the suggestions in order to produce the Checkers Report (checks subcommand) that lists all the checks applicable to your code.

```
codee checks --target-arch cpu himeno.f90 -- -O3
```

```
Date: 2024-04-25 Codee version: 2024.2.2
Compiler flags: -O3

[Fortran] target compiler: <none> (Compiler Agnostic Mode)
```

```
[1/1] himeno.f90 ... Done
```

CHECKS REPORT

```
himenof.f90:295:12 [PWR054] (level: L1): Consider applying vectorization to scalar reduction loop
himenof.f90 [PWR063] (level: L1): Avoid using legacy Fortran constructs
himenof.f90:293:6 [PWR051] (level: L2): Consider applying multithreading parallelism to scalar
reduction loop
himenof.f90:136:1 [PWR001] (level: L3): Declare global variables as function parameters
himenof.f90:164:1 [PWR001] (level: L3): Declare global variables as function parameters
himenof.f90:223:1 [PWR001] (level: L3): Declare global variables as function parameters
himenof.f90:255:1 [PWR001] (level: L3): Declare global variables as function parameters
himenof.f90:275:1 [PWR001] (level: L3): Declare global variables as function parameters
himenof.f90:295:12 [RMK001] (level: L3): Loop nesting that might benefit from hybrid parallelization
using multithreading and SIMD
himenof.f90:293:6 [PWR035] (level: L3): Avoid non-consecutive array access to improve performance
himenof.f90:294:9 [PWR035] (level: L3): Avoid non-consecutive array access to improve performance
```

SUGGESTIONS

Use `--verbose` to get more details, e.g:
`codee checks --verbose --target-arch cpu himenof.f90 -- -O3`

Use `--level` to filter checks with a specific level of priority, e.g:
`codee checks --level L1 --target-arch cpu himenof.f90 -- -O3`

More details on the defects, recommendations and more in the Open Catalog of Best Practices for Performance:
<https://github.com/codee-com/open-catalog/>

Consider using Codee with a target compiler in order to filter out optimizations that are already applied by your compiler. For example, for GCC:
`codee checks --target-compiler-fc gfortran --target-arch cpu himenof.f90 -- -O3`

1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 186 ms

3. Show the detailed Checkers Report (option `--verbose`). As an example, focus on the checker *PWR051*, related to parallelizing a loop with multithreading. The detailed Codee output, which includes links to the open catalog available in the website, precise location in the source code, etc..., is as follows:

```
codee checks --verbose --target-arch cpu himenof.f90 -- -O3
```

```
...
himenof.f90:293:6 [PWR051] (level: L2): Consider applying multithreading parallelism to scalar
reduction loop
Suggestion: Use 'rewrite' to automatically optimize the code
Documentation: https://github.com/codee-com/open-catalog/tree/main/Checks/PWR051
AutoFix (choose one option):
  * Using OpenMP 'for' with built-in reduction (recommended):
    codee rewrite --multi omp-for --in-place himenof.f90:293:6
  * Using OpenMP 'for' with explicit privatization:
    codee rewrite --multi omp-for --in-place --explicit-privatization gosa himenof.f90:293:6
  * Using OpenMP 'taskwait':
    codee rewrite --multi omp-taskwait --in-place himenof.f90:293:6
  * Using OpenMP 'taskloop':
    codee rewrite --multi omp-taskloop --in-place himenof.f90:293:6
...
```

4. Note that the check PWR051 contains an *AutoFix* section that remarks the existence of source code rewriting capabilities that facilitate the usage of Codee as a coding assistant. Focus on PWR051 and implement multithreading with OpenMP “for” option, by executing the following command:

```
codee rewrite --multi omp-for -o himenof_codee.f90 himenof.f90:293:6 -- -O3
```

5. Compile the original source code of Himeno (*himenof.f90*) and the optimized source code of Himeno (*himenof_codee.f90*). For instance, using the GFortran compiler:

```
gfortran himenof.f90 -o himeno -O3
gfortran himenof_codee.f90 -o himenof_codee -O3 -fopenmp
```

6. Run the original MATMUL (`matmul`) and the optimized MATMUL (`matmul_codee`):

Linux: `./himeno`
 `./himeno_codee`

Windows: `.\himeno`
 `.\himeno_codee`

Press L to select the large input dataset.