Intel C/C++ Compiler (ICC) Python Package

Aleksandr Maramzin

November 11, 2018

1 Overview

Python package *intel_compiler* is Intel C/C++ Compiler (ICC) optimization report processor. The format and description of ICC optimization report can be found on the official Intel's website [1]. The package is implemented as a simplistic compiler.

It does not strictly follow in it's internal design accepted compiler's front-end organisation practices. It performs minimal input file syntax verification and mostly relies on its strict conformance to the grammar. The package reads optimization report from the disk and transforms it into an intermediate representation of a loop nesting structure in the Python's memory. It populates built loop nesting structure with all ICC optimizations [2] applied to given loops.

2 Intel C/C++ Compiler (ICC) optimization report structure

This section roughly approximates the structure of Intel C/C++ Compiler optimization report. This approximation is presented in a form of a BNF context-free grammar. This grammar approximates only those detailes, needed for extraction and omits all unnecessary ones.

⟨*optimization-report*⟩ ::= ⟨*loop-report-list*⟩

⟨*loop-report-list*⟩ ::= ⟨*loop-report*⟩ ⟨*loop-report-list*⟩
| ε

⟨*loop-report*⟩ ::= ⟨*loop-begin*⟩ ⟨*loop-partition-tag*⟩ ⟨*remark-list*⟩ ⟨*loop-report-list*⟩ ⟨*loop-end*⟩

⟨*loop-begin*⟩ ::= '`LOOP BEGIN at`' ⟨*filename*⟩ '`(`' ⟨*line-number*⟩ '`)`' ⟨*inlined-into*⟩

⟨*loop-tag*⟩ ::= ⟨*peel-mark*⟩
| ⟨*remainder-mark*⟩
| ⟨*distributed-chunk-mark*⟩
| ⟨*distributed-chunk-remainder-mark*⟩
| ε

⟨*remark-list*⟩ ::= ⟨*remark*⟩ ⟨*remark-list*⟩
| ε

⟨*loop-end*⟩ ::= '`LOOP END`'

⟨*remark*⟩ ::= ⟨*loop-report*⟩
| ⟨*unimpotant-diagnostic*⟩
| ⟨*parallel-diagnostic*⟩
| ⟨*vector-diagnostic*⟩

| ⟨*parallel-potential-diagnostic*⟩
| ⟨*vector-potential-diagnostic*⟩
| ε

⟨*peel-mark*⟩ ::= '`<Peeled loop for vectorization>`'

⟨*remainder-mark*⟩ ::= '`<Remainder loop for vectorization>`'

⟨*distributed-chunk-mark*⟩ ::= '`<Distributed chunk([0-9]+)>`'

⟨*distributed-chunk-remainder-mark*⟩ ::= '`<Remainder loop for vectorization, Distributed chunk([0`

⟨*inlined-into*⟩ ::= '`inlined into`' ⟨*filename*⟩ '`(`' ⟨*line-number*⟩ '`)`'
| ε

Increase the two lengths

⟨*statement*⟩ ::= ⟨*ident*⟩ '`=`' ⟨*expr*⟩
| '`for`' ⟨*ident*⟩ '`=`' ⟨*expr*⟩ '`to`' ⟨*expr*⟩ '`do`' ⟨*statement*⟩
| '`{`' ⟨*stat-list*⟩ '`}`'
| ⟨*empty*⟩

⟨*stat-list*⟩ ::= ⟨*statement*⟩ '`;`' ⟨*stat-list*⟩ | ⟨*statement*⟩

References

[1] Intel(R). Getting the Most out of your INTEL(r) Compiler with the New Optimization Reports. https://software.intel.com/en-us/articles/getting-the-most-out-of-your-intel-compiler-with-the-new-optimization-reports.

[2] David F. Bacon, Susan L. Graham, and Oliver J. Sharp. Compiler transformations for high-performance computing. *ACM Comput. Surv.*, 26(4):345–420, December 1994.