Assist: A Feedback-Directed Optimization Source-to-Source Transformation Tool for HPC Applications

Contact Info
Youenn Lebras
Exascale Computing Research (ECR) Lab
University of Versailles Saint Quentin (UVSQ), FRANCE
youenn.lebras@uvsq.fr

Assist: Automatic Source-to-Source Assistant

Traditionally, users try to optimize their source code according to the metrics/hints reported by performance evaluation tools. Manually implementing optimization transformations using these hints can be tedious, when not error prone for a programmer. Also, the source code's readability is greatly impacted by the intricacies of such transformations; for example, handling special cases or multiple architectures will surely bloat the code. One clean way to optimize a target code is to use Feedback Data Optimization (FDO) (i.e. the Intel Compiler Profile Guided Optimization or PGO) to better steer the compiler during its optimization pass.

We present Assist (Automatic Source to Source assistant), a directive oriented source to source manipulation FDO tool developed as a MAQAO[1] module. It is able to guide code transformations based on static and dynamic feedback and aims at providing assistance with respect to productivity and code performance efficiency. It is open source and provides a more flexible alternative than compilers FDO modes while still being complementary. Assist support common transformations such as: loop interchange, unroll/fullunroll, strip mine and tile; but also more advanced ones like: function/loop specialization, block vectorization[2] and loop count transformation[2](LCT).

These transformations can be triggered either by manually inserting directives in the source code – also through a configuration file - or automatically by using analyses from MAQAO modules. The MAQAO toolset focuses on the performance evaluation and optimization of binary applications and provides statics and dynamics tools allowing programmers to: locate hotspot functions/loops, perform value profiling, determine the vectorization ratio of target loops, etc.

This work has been previously presented at the 11th Parallel Tools Workshop at Dresden in Germany. Tests were performed on active industrial applications such as: Yales2, a numerical simulator of turbulent reactive flows developed by CORIA; and AVBP, a parallel CFD code developed by CERFACS. The block vectorization has been applied to certain AVBP functions and the speedup ranges from 1.3 to 2.6. We also compared the performance of both our FDO transformations (LCT) and Intel’s PGO which can be complementary, but, if applied separately, sometimes, LCT can be more efficient than the PGO mode.

References & Definitions
[1] www.maqao.org
[2] Block Vectorization Transformation: This transformation aims to increase the vectorization ratio by splitting a loop into multiple ones, adapt the number of iterations to the vector length and force the compiler to vectorize loops using SIMD directive.
[3] Loop Count Transformation: Intel compiler offers the ability to specify a “loop count” (min, max, avg) directive. The compiler can then make that information available to its optimization passes to generate less variants of the loop (e.g. scalar, SSE, AVX, etc).