Abstract. User feedback in program parallelization is very important since often blind compilation may bring to mistakes in the generated code causing performance degradation. This is why it is necessary to provide the user with clear information about how the compiler parallelizes the program and, at the same time, an interface to control and assist program parallelization according to user feedback. To cope with this aspect we developed the HTG Visualization Tool (HTGviz), a graphic user interface for program parallelization based on a joint work of user knowledge and compiler techniques. HTGviz is implemented on top of Parafrase-2 parallelizing compiler, and Tcl/Tk is used as middleware integration language to implement graphical components. The interaction between the user and the compiler is carried out through the use of the Hierarchical Task Graph (HTG) program representation where task parallelism is represented by precedence relations (arcs) among task nodes. HTGviz offers the user an interface to navigate through the program HTG, providing him with information about data/control dependences and task precedences. It allows to tune task partitioning and parallelism by inserting OpenMP directives into the code based on graph manipulation facilities. HTGviz also guides the user through the process of generating valid and efficient parallel code for OpenMP applications.

1 Introduction

Parallel programs are collections of tasks that could run serially or in parallel. The grain size problem [1, 9, 11] is how to determine the best trade-off between task grain (amount of parallelism) and overhead. Sources of overhead in a multiprocessor setting include scheduling, synchronization and communication overhead [7, 2]. Partitioning techniques are necessary to execute parallel programs at appropriate granularity for a target multiprocessor. It is desirable for task partitioning to be performed automatically, so that the programmer is not burdened with machine details, and the same program can be efficiently compiled for different architectures. Unfortunately, blind compilation often fails in this job since small mistakes may cause degradation of performance on multiprocessors.
Therefore, user feedback is very important for the process of program task partitioning and parallelization. This process often involves calibration of program parameters through experimentation, and this work may be repeated several times for the same program and target machine before reaching the optimal parallel code generation and an efficient program execution.

With respect to this problem our approach is to develop a graphical user interface integrated in a host parallelizing compiler to let the user assist and control the program parallelization process. This task is hard for today parallelizing compilers since they transform the program to a great extension. It is difficult for programmers to relate the transformed program to the original source code and, therefore, to determine how the program parallelization can be improved: the correct interpretation of transformed code requires a deep understanding of compiler techniques and, furthermore, the programmer can easily get confused by a large amount of information.

We adopted the user point of view on looking for which kind of program representation the user–compiler interaction should rely on. The task graph could be considered the program intermediate representation closer to the user conceptual program representation in case of parallel execution. Therefore we chose to carry out the user–compiler interaction on the program Hierarchical Task Graph (HTG) as defined in [4, 5, 3, 6]. Since the HTG representation does not depend on the particular input language, it can be considered abstract. HTG nodes are tasks, defined as sections of code delimited by "natural" boundaries (loops, statements, subroutine calls, and basic blocks). HTG arcs represent precedence relations that must be satisfied to execute a task. The HTG could contain all program parallelism, from coarse- to fine-grain parallelism.

In this paper we describe the implementation design of the HTG Visualization Tool\(^1\) (HTGviz), a graphic tool for user interaction with the compilation process. HTGviz is built on top of Parafrase-2\(^2\) parallelizing compiler [16, 17] and uses its internal program representation (HTG) as interaction medium. Parafrase-2 is a research tool for experimenting with program transformations and compiler techniques for parallel computing on shared memory computers. HTGviz is designed as an interface for program parallelization tuning and a visualizer for compiler internal representations of programs. During the HTGviz session the user performs several operations: refine task partitioning and parallelization adding (or removing) precedences, guide parallel code production, selecting program units to be translated, and mark code to be instrumented. After compilation and execution, performance profiling can be used in the next compilation step to improve program parallelization.

The paper is organized as follows: section 2 introduces the notion of Hierarchical Task Graph; section 3 describes the overall HTGviz design criteria and objectives; section 4 gives a detailed overview of the main HTGviz user interfaces and functionalities; section 5 reports some concluding remarks.

\(^1\) Developed within the NANOS Esprit Project NO.21907
\(^2\) developed at the CSRD - University of Illinois at Urbana-Champaign.