if-conversion 研究近况及其在 LLVM 上的实现

高翔

qasdfgtyuiop@gmail.com

February 25, 2014

Abstract

人们追求程序运行的性能的脚步从来都没有停歇过，为了使得程序更快地运行，人们不断的改进处理器的性能。然而，处理器的性能只是程序性能的一部分，要实现高性能，编译器生成的代码的执行效率也至关重要。更先进的构架给编译器提出了更高的要求，人们也一直在探索如何让编译器生成更高效率的代码。由于早先的编译器无法很好地处理控制依赖，Allen 等人于 1983 年提出了 if 转换（if-conversion）的概念 [1]。if-conversion 能够消除程序中的除了后向分支以外的所有分支，这样就使得程序中的控制依赖被转化为数据依赖，编译器可以在此基础上进行进一步优化。1987 年，Ferrante 等人提出了程序依赖图（PDG）的概念 [3]，使得编译器可以轻易的处理控制依赖。尽管现代的编译器可以轻易处理控制依赖，分支语句的存在有时候仍然会严重影响性能。分支语句会中断指令流水，造成严重的性能损失。根据 Bringmann 的统计 [2]，程序中 10% ∼ 30% 的语句是分支语句，如果一个分支语句会造成 2 个周期的延迟，那么会有大约 40% 的机器时间浪费在分支上。此外，分支语句会将程序划分为若干基本块，从而阻断了进一步的优化。If 转换可以通过现代处理器提供的条件执行指令，实现分支消除，这样做不仅可以减少分支造成的延迟，而且可以将若干小的基本块合并成一个大的基本块，进而为后续优化提供便利。由于上述原因，虽然现代编译器能够很好地处理控制依赖，If 转换仍然显得尤为重要。

We have never stopped pursuing the performance of the program, to achieve this goal, the performance of the processor has been highly developed. However, the performance of the processor is only part of the performance of the program. The performance of the code generated by the compiler is also an important factor. The modern architecture has brought greater challenges to the development of compiler. How to design a compiler that generates high performance instructions has been a problem for a long time. Early compiler couldn't process control dependency. To solve this problem, Allen et al. came up with the concept of if-conversion [1], which

can remove all branches in the program except for the backward branch. The deletion of branches converts control dependency to data dependency, making it possible for the compiler to do further optimizations. In 1987, Ferrante et al. came up with the concept of program dependency graph (PDG) [3], making it easier for compilers to handle control dependency. Albeit this improvement, if-conversion is also important since the branch can significantly harm the performance. Branch interrupts the pipeline of instructions, causing great loss in performance. According to the statistics of Bringmann [2], among instructions about 10% $\sim$ 30% are branches. Assume that one branch can cause about 2 cycles delay, then about 40% of machine time will be wasted in branch. What's more, branch divides the program to several basic blocks, making it harder for further optimization. If-conversion removes branches by taking use of predicted execution mechanism of modern processor. This can not only eliminate the delay caused by branch, but also merge several small basic blocks to a large basic block, making a lot of optimizations possible. Due to the reasons stated above, if-conversion is also very important.

目录

参考

[1] John R Allen, Ken Kennedy, Carrie Porterfield, and Joe Warren. Conversion of control dependence to data dependence. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 177–189. ACM, 1983.

[2] Roger A. Bringmann, Scott A. Mahlke, Richard E. Hank, John C. Gyllenhaal, and Wen mei W. Hwu. Speculative execution exception recovery using write-back suppression. In *MICRO*, pages 214–223. ACM/IEEE, 1993.

[3] J. Ferrante, K. Ottenstein, and J. Warren. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349, July 1987.