编译器中的 edge profiling 设计和实现*)

董希谦　张兆庆
（中国科学院计算技术研究所　北京 100080）

The Design and Implementation of Edge Profiling in Compiler

DONG Xi-Qian　ZHANG Zhao-Qing
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

Abstract　Many compiler optimization techniques depend on which part code has been executed frequently. Profiling will trace and record these information that a compiler needs automatically, which is useful to other phases during compile. Profiling will instrument some internal code that gather these information during a program executes. For example, edge profiling computes the frequency of basic block and the probability of edge in control flow graph. Value profiling determine which variables have invariant behavior.

This paper explores the basic block frequency found from edge profiling. We use the edge profiling to perform some optimization for Spec2000 test cases, showing that the optimization base edge profiling can reduce a program's execution time up to 5%.

Keywords　Edge profiling, Compiler, Profiling_based optimization, Dynamic optimization

1. 简介

这篇论文是在从事 ORC(Open Research Compiler)编译器的过程中完成的。ORC 是中科院计算所同 Intel 公司的合作项目，为 Intel 的新一代 64 位芯片开发开放源代码的先进编译器。ORC 的实现目标：为研究机构和团体提供一个可靠稳定的、开放源代码的编译基础结构，方便其在 ORC 的基础上进行编译器和体系结构的研究；与当前其他 IA-64 编译器相比，要有比较高的性能。

代码优化的一个重要目标是为了减少程序的执行时间。许多编译优化技术，如控制流的优化、指令调度等都依赖于分析程序的动态行为，如各个基本块执行的频率高低或者控制流边执行的概率的大小等信息得到更好的性能。profiling 通常分成静态的 profiling 和动态的 profiling：静态的 profiling 通过启发的算法进行估算统计；动态的 profiling 通过在中间语言插入的插装库函数收集程序运行中动态信息。同样，edge profiling 分成静态的 edge profiling 和动态 edge profiling。静态的 edge profiling 采用启发式算法估算程序各个基本块的执行频率和各个边的概率。比如对于循环的检测(loop detection)，可以估计执行的次数，但估计得并不准确。静态的 edge profiling 不能准确地得一个基本块在实际执行过程中的频率，对于有分支(branch)的指令，很难估算出哪一条边的执行的概率要高一些，而且估算出的结果不准确，有些时候估算的结果与实际运行的结果恰恰相反，这些都使优化不能很好地进行。

在没有动态 profiling 支持的编译器中，静态分析估计出各个基本块执行的频率和控制流边的执行概率。静态估计无法获取准确的程序运行信息。而动态 profiling 可以通过选择有代表性的输入，获取正确的程序运行信息。profiling 执行的过程为：选择不同的、有代表性的输入；运行程序：edge profiling 在运行的过程中记录程序运行信息并在程序结束时保存信息；利用获得的运行信息重新编译优化程序。具体的过程如图 1。

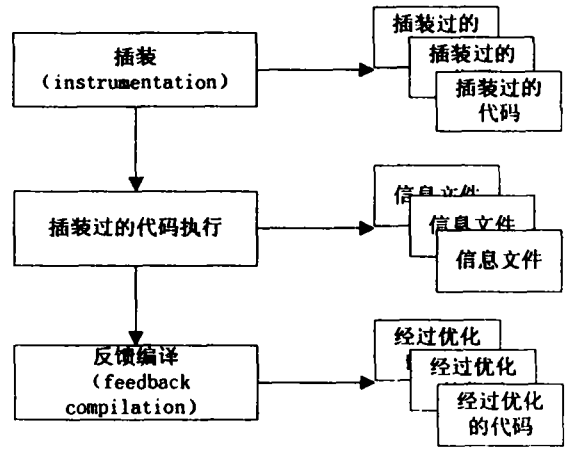

图 1　edge profiling 执行过程

2. 实现概要

一个基于 edge profiling 的编译器的结构如图 2 所示。profiler 向编译器产生的中间代码中插入一些指令，这些指令记录并保存程序在运行中的信息。在第二次编译中，profiler 读取程序运行时信息，基于 profiling 的代码优化器获取 profiler 产生的反馈信息，并利用此信息对代码进行优化。

从 edge profiling 实现的结构上来看，分成三个部分：

a)插装(instrumentation)：向程序的中间表示中插入调用插装库函数的指令，同时设置传递给插装库函数的参数。

*)本课题得到国家自然科学基金(69933020，60103006)资助。董希谦　硕士研究生，主要研究方向为基于 profile 的动态优化。张兆庆　研究员，博士生导师，研究领域包括并行程序设计环境，自动并行化与并行可视化工具以及并行程序的正确性验证等。

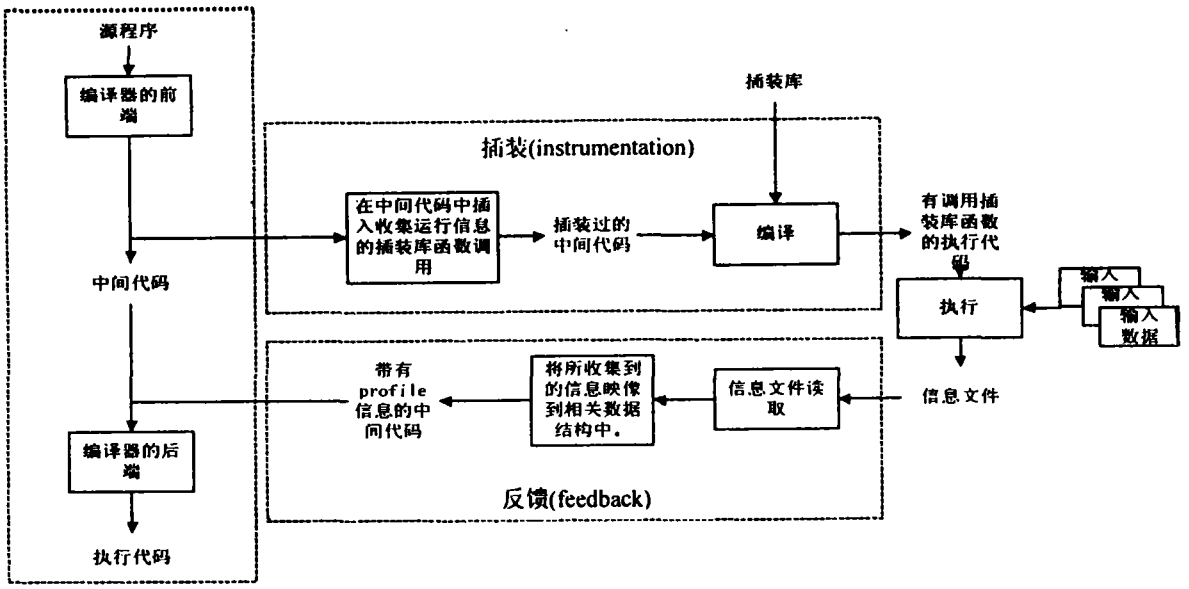

图 2　基于 edge profiling 的编译器结构

b)插装函数库：这部分函数负责统计收集程序运行时的信息；申请和释放保存信息所使用的存储空间；将信息保存到反馈文件中。

c)反馈(annotation)：读取反馈文件，获取程序运行时的信息，并将信息标注到中间数据结构中，编译器可以访问其获取所需信息。

代码优化很大部分基于控制流图(CFG)，在 ORC 编译器中，采用(VORTEX，EDGE，BB_freq，BBLIST_prob，BBLIST_freq)来表示 CFG，每个节点(VORTEX)对应于一个基本块(basic block)，Edge 对应于两个基本块之间的控制关系。BB_freq(bb)对应于一个基本块的执行频率，BBLIST_freq(e)对应于一条控制流边的执行频率，BBLIST_prob(e)对应于一条控制流边执行的概率。每个基本块包含一列指令流，以如下类型的指令为出口：1)跳转(jump)指令；2)分支(branch)指令；3)投机指令(chk.s)；4)函数调用(call)指令。

profiler 完成的功能为：a)每个基本块执行的次数；b)计算每条控制流边的概率；c)每个函数调用的次数。

我们将 profiler 获得的信息，附在控制流图上，当优化器进行优化时，可以参看控制流图上的信息。

edge profiling 自完成以下的工作过程：

I)向中间代码中插入一些调用插装库函数的代码。插入的代码位置在各个分支指令后。在 ORC 中，每个基本块以分支指令结束，分支指令的类型分成：a)指令指针相关(ip_relative)跳转指令；b)函数调用(call)指令；c)非直接跳转指令。

对于 a)，c)类型的指令，插装的过程如下：在插装前，基本块 1(bb1)的分支指令为 branch L1，表示在某个条件下指令跳转到由 L 标识的基本块 2(bb2)；插装时，首先生成由 L1' 标识的基本块 i(bbi)，修改跳转指令 branch 跳转的目标为 bbi。在控制流图中，将每条控制流边赋予唯一的一个 id 标识，同时有一个计数器 count(id)与它对应，用来记录这条控制流边在程序执行的过程中执行过的次数，生成的基本块 i 的指令功能为：将本控制流边对应的计数器值加 1，执行完毕后并跳转到原控制流边的目的基本块 bb2。对于控制流边 e = (bb1->bb2)，在加入 bbi 后，控制流边变为 e=(bb1->bbi)，同时新增控制流边 e'=(bbi->bb2)。如图 3 所示。插入的代码功能为将原控制流边对应的计数器加 1：count(id)=count(id)+1。对于 b)类指令，其插装的过程如图 4 所示。

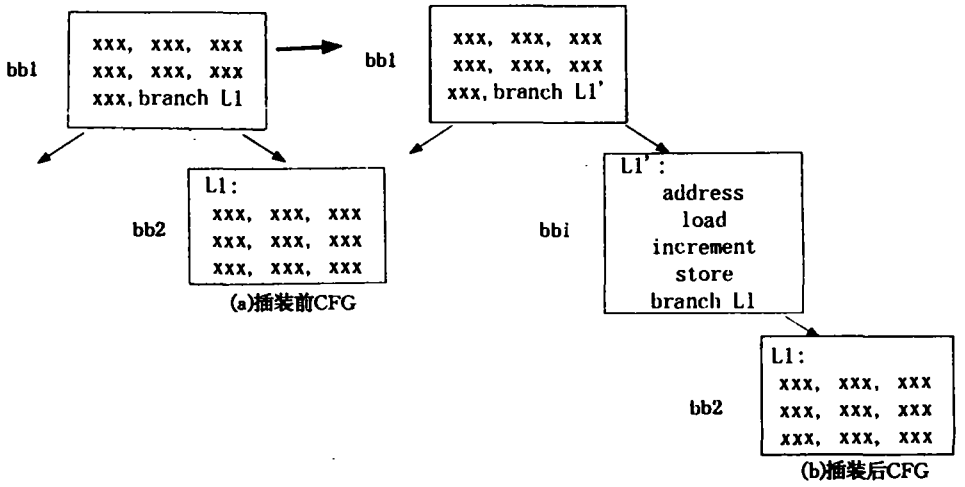

图 3　对 a，c 类型分支指令的插装过程

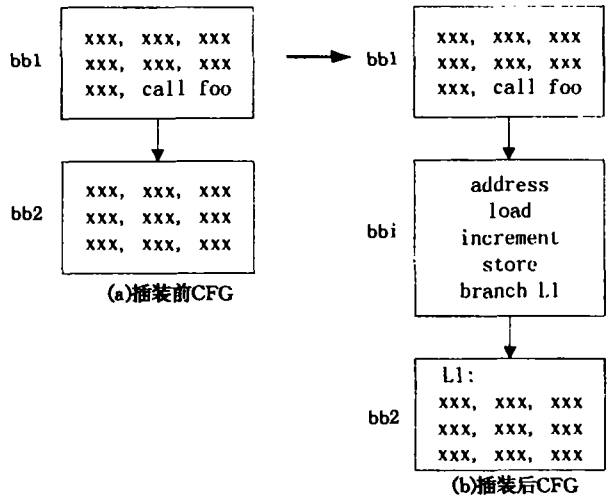

图 4　对 b 类型分支指令的插装过程

如果对每条控制流边都插装代码，获得的信息会出现冗余的情况，同时增加编译和程序运行时间。例如：

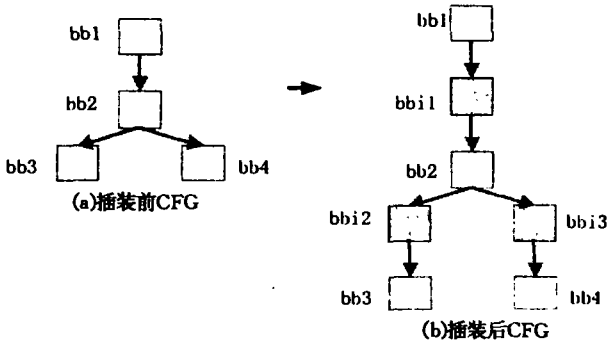

图 5

其中边 e1＝(bb1->bb)，e2＝(bb2->bb3)，e3＝(bb2->bb4)，其对应的计数器为 count(1)，count(2)，count(3)，其满足 count(1)=count(2)+count(3)，根据这个关系，我们可以只对边 e1，e2 或 e1，e3 进行插装代码。另一边的计数器 count(3)的数值可以推算出：count(3)=count(1)-count(2)。为了减少 profiling 的运行时间开销，我们对插装进行优化，并不对所有的边都进行插装。而是由控制流图构造 spanning tree，只对 spanning tree 上的控制流边进行插装。其优化的过程为：

a. 按照启发式算法估算出每条边的权(频率)

b. 构造一条 max_weight spanning tree

c. 对 spanning tree 上的边进行插装。

II)在程序运行的过程中，插装过的代码统计计算各个控制流边的执行次数，每当控制流从一个基本块进入到另一个基本块中时，这两个基本块间的控制流边对应的计数器 count(i)加 1。当程序结束，插装库函数完成收集所需相应信息，即各个控制流边执行的频率，在程序结束之前，插装库中的 finalize 函数将这些信息按照一定的格式存放到指定的文件中，同时释放插装库函数在运行期间申请的存储空间。通常在程序控制流图的入口处插装调用初始化函数，负责分配存储空间。在程序运行结束后，将收集到的信息存放到用户指定或默认的文件，以便以后的编译使用。存取的文件有一定的结构，方便再编译时的读取。反馈文件一般分为四个部分(如图 6)：

文件头信息：a. 标识这个文件是反馈文件，b. PU 头信息的文件的偏移地址，个数和 PU 头信息的大小(定值)，c. PU 函数名表的文件的偏移地址，个数和函数名表的大小 d. Profiling 反馈信息文件的编移地址。

PU 头信息：a. 插装的节点数 b. 该 PU 反馈信息的文件的偏移地址 c. 该函数名在 PU 函数名表中的索引值

PU 函数名表：各个函数名

PU 反馈信息：各个边的频率计数器值

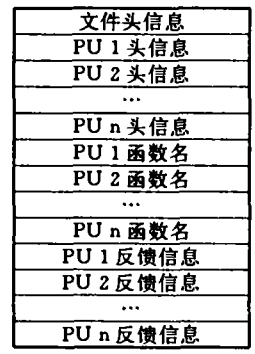

图 6　反馈文件的格式

III)合并多个反馈文件中的信息。将收集到的信息合并在一起，并读取所需信息。也可以在第二次编译时，将程序多次运行产生的多个信息文件合并。

IV)将收集到的运行信息映射到控制流图的基本块和控制流边，并将信息标注在上面。编译器在再编译时，首先根据函数名获得反馈文件的 PU 头信息，再通过 PU 头信息读取到相关控制流边的执行频率。根据各条控制流边的频率计算出各个基本块的运行频率和各个控制流边相对于基本块的概率。

V)对于任何一个节点 bb，都满足：freq(in_edge)＝freq(bb)，即该节点所有入边的频率＝该节点的频率。而对于一般节点 bb(该节点不是程序的终止节点)都满足 freq(out_edge)=freq(bb)。根据这些关系，计算出基本块的频率和没有插装过控制流边的频率和概率。

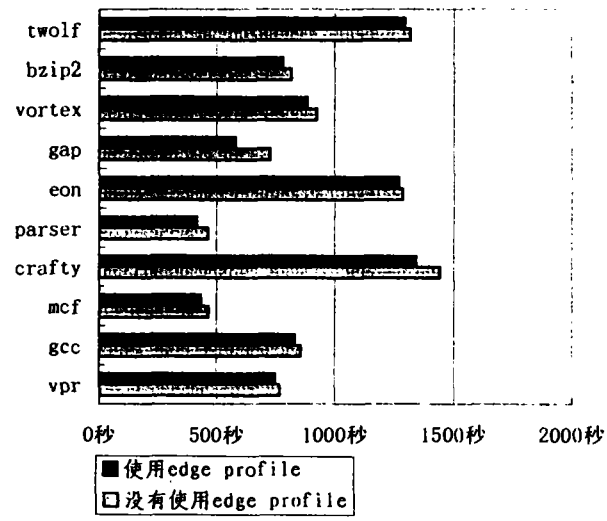

图 7　对 CPU2000 的测试结果

3. 性能测试结果

我们对基准测试程序 CPU2000 进行了比较测试。测试的

(下转第 66 页)

掘具备更高的精确度。由于一般文本挖掘的目标是面向应用的，因此领域启发式知识或知识库的使用能提高分词和标注的精度，已达到文本挖掘所需的要求。

结论和展望 文本挖掘技术是一种综合机器学习、数据挖掘、自然语言处理和其他各种文本处理方法如信息抽取和文本分类的文本自动处理技术。文本挖掘的目标是发现隐含的归纳知识如关联知识、序列知识及完全创新的科学推断和假设等。基于信息抽取和数据挖掘算法的文本挖掘技术是文本知识发现的技术研究趋势，与具体的次语言(sublanguage)领域相结合，从而发现和提炼更具价值的应用领域知识模式是文本挖掘的应用趋势。

参 考 文 献

1 Han Jiawei. Micheline Kamber (2000). Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers. All rights reserved

2 Tukey J W. Exploratory Data Analysis, Addison-Wesley, 1977

3 Ahonen H, Heinonen O, et al. Mining in the Phrasal Frontier. PKDD-97, Trondheim, Norway, June 1997

4 Salton G. Automatic text processing. Reading, MA: Addison-Welsley, 1989

5 Burges C. A tutorial on SVM for pattern recognition. Data Mining and Knowledge Discorvery

6 Chapelle O, Vapnik V. Model selection for SVM. Advances in Neural Information Processing Systems, MIT Press, 2000

7 Freitag D, McCallum A. Information Extraction with HMMs and Shrinkage. In: Proc. Workshop on ML and IE, AAAI-99, 1999

8 Cortes C, Vapnik V. Support-vector networks. Machine Learning. Nov. 1995

9 Vapnik V N. The Nature of Statistical Learning Theory. Springer. New York, 1995

10 Mladenic D. Feature subset selection in text -learning. In: Proc. of the 10th European Conf. on Machine Learning ECML98, 1998

11 Sager N. Sublanguage: Linguistic Phenomenon, Computational Tool. In [Grishman and Kittredge 1986].

12 Swanson D R. Two medical literatures that are logically but not bibliographically connected. Journal of the American Society for Information Retrieval, 1987, 38 (4): 228～233

13 Swanson D R, Smalheiser N R. Assessing a gap in the biomedical literature: magnesium deficiency and neurologic disease. Neuroscience Research Communications, 1994, 15: 1～9

14 Swanson D R, Smalheiser N R. An interactive system for finding complementary Literatures: a stimulus to scientific discovery. Artificial Intelligence, 1997, 91: 183～203

15 Hearst M A. Untangling text data mining. In: Proc. of ACL '99: the 37th Annual Meeting of the Association for Computational Linguistics, New Brunswick, NJ: The Association for Computational Linguistics

16 Armstrong R, Freitag D, Joachims T, Mitchell T. WebWatcher: A Learning Apprentice for the World Wide Web. In: Proc. of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, Stanford

17 Lam W, Low K F, Ho C Y. Using Bayesian Network Induction Approach for Text Categorization. In: Proc. of the 15th Intl. Joint Conf. on Artificial Intelligence IJCAI97 (pp. 745～750)

18 Sorensen H, McElligott M. PSUN A Profiling System for Usenet News. In: Proc. of CIKM'95 Intelligent Information Agents Workshop, Baltimore

19 Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988, 24: 513～523

20 Papadimitriou H, Raghavan P, Tamaki H, Vempala S. Latent Semantic Indexing: A probablistic analysis. In: Proc. of Symposium on Principles of Database Systems (PODS). ACM Press, 1998

21 Feldman R, Hirsh H. Finding Associations in Collections of Text. In: Machine Learning and Data Mining, R. S. Michalski, I. Bratko, M. Kubat, eds. John Wiley & Sons, NY 1997

22 Kodratoff Y. Knowledge Discovery in Texts: A Definition, and Applications. Proc. ISMIS'99, Warsaw, June 1999a

23 Stephen, Potter. A survey of knowledge acquisition from natural language. Artificial Intelligence Applications Institute, Division of Informatics, University of Edinburgh

24 Feldman R. In: Proc. of the Sixteenth Intl. Joint Conf. on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications. 1999

25 Nahm U Y, Mooney R J. Text Mining with Information Extraction To appear in the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002

26 Cowie J, Lehnert W. Information Extraction. Communications of the ACM, 1996, 39(1): 80～91

27 Gaizauskas R, Robertson A M. Coupling IR and IE: A New Text Technology for Gathering Information from the Web. In: Proc. of RIAO 97: Computer-Assisted Information Searching on the Internet, Montreal, Canada, 1997. 356～370

28 Califf M E, Mooney R J. Relational learning of pattern-match rules for information extraction. In: Working Papers of ACL-97 Workshop on Natural Language Learning, 1997

29 Holt J D, Chung S M. Efficient Mining of Association Rules in Text Databases. 1999 ACM 1-581

30 Taira H, Haruno M. Feature Selection in SVM Text Categorization. In AAAI-99

31 Rennie J, McCallum A K. Using Reinforcement Learning to Spider the Web Efficiently. ICML-99 Workshop: Machine learning in Text Data Analysis

32 Yang Y, Pedersen J. A Comparative Study on Feature Selection in Text Categorization. ICML-97 (pp. 412-420)

33 Grishman R. (2001). Adaptive Information Extraction and Sublanguage Analysis. IJCAI-2001 Workshop on Adaptive Text Extraction and Mining

34 Grishman R, Kittredge R. Analyzing Language in Restricted Domains: Sublanguage Description and Processing. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986

35 Text Mining Technology Turning Information Into Knowledge. A White Paper from IBM, 1998

36 Feldman R, Dagan I, Kloegsen W. Efficient Algorithm for Mining and Manipulating Associations in Texts. 13th European Meeting on Cybernetics and Research, 1996

(上接第48页)

环境：733MHz 主频、1G 内存的 Itanium 工作站，操作系统为 Red Hat Linux 6.2。使用 edge profiling 的编译器 ORC 和没有采用 edge profiling 的 ORC 测试结果表示，edge profiling 对所有程序的性能都有一定的提高，性能提高百分比为：2%(vpr)到20%(gap)，平均达到5%。

参 考 文 献

1 Albert G. A Transparent Method for Correlating Profiles with Source Programs. 2nd Workshop on Feedback Directed Optimization, Haifa, Israel, Nov. 1999

2 Anderson J, Berc L, et al. M. Continuous Profiling: Where Have All the Cycles Gone? ACM Trans. on Computer Systems, 1997, 15(4): 357～390

3 Ball T, Larus J. Efficient Path Profiling. In: Proc. 29thAnnual IEEE/ACM Intl. Symp. on Microarchitecture, Dec. 1996. 46～57

4 Conte T M, Menezes K N, Hirsch M A. Accurate and practical profile-driven compilation using the profile buffer. In: Proc. 29th Annual International Symposium on Microarchitecture, Dec. 1996. 36～45

5 Young C, Smith M. Better Global Scheduling Using Path Profile. In: Proc. 31st Annual International Symposium on Microarchitecture, Dec. 1998. 115～123