渐进式智能回溯向量化代码调优方法

赵 博 赵荣彩 徐金龙 高 伟
(信息工程大学 郑州 450002) (数学工程与先进计算国家重点实验室 郑州 450002)

摘 要 为了充分发挥高性能计算机的计算能力,缓解程序员设计和编写并行程序的压力,扩充可用软件集合,设计并实现了利用交互界面深入挖掘程序中的可向量化语句,优化生成代码中的向量化语句,提高生成代码的执行效率。该方法对充分发挥高性能计算机的计算能力,增强系统可用性和扩展应用范围具有重要的意义,同时能够提供有效的辅助手段和工具支持。渐进式智能回溯向量化代码调优架构通过对用户提交的串行程序进行程序分析和变换,采用串行程序分析、数据依赖分析、向量化分析等技术手段,根据分析结果对程序进行变换和优化,自动生成最终的向量化代码。该方法通过分析串行程序中潜在的并行性,将其自动变换为等价的向量化代码形式,大大简化了程序员的工作。

关键词 渐进式,静态调优,动态调优,动静结合
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.011

Method of Progressive Intelligent Backtracking Vector Code Adjustment and Optimization

ZHAO Bo ZHAO Rong-cai XU Jin-long GAO Wei
(PLA Information Engineering University, Zhengzhou 450002, China)
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China)

Abstract In order to fully develop the computing ability of high-performance computer and relieve the pressure of designing and writing parallel programs for programmers, with the expansion of available software sets we designed and realized the vector program through the interactive interface. Using this method, we could optimize the generated vector code to improve the efficiency for the implementation of generated code. Our process is of great significance to express the ability of high-performance computing, enhance the availability and extend range of application. Furthermore, it can provide available supplementary means and tool support. The method of progressive intelligent backtracking vector code adjustment and optimization can automatically generate the parallel code after analysis and transformation of the serial code given by the user using the following methods such as serial code analysis, data dependence analysis and parallelization analysis, etc. The work of this article can greatly reduce the work of programmers by transforming the serial code into parallel code automatically with the analysis of parallelization in the serial code.

Keywords Progressive, Static adjustment, Dynamic adjustment, Static combined dynamic

1 引言

越来越多的通用处理器上集成了SIMD扩展,如2000年AMD在Athlon处理器上集成了类似于SSE的3DNow!指令集[1]。通过并行执行一条指令的多个数据实现,能够实现应用程序并行处理[2]。为了充分发挥高性能计算机的计算能力,缓解程序员设计和编写并行程序的压力,扩充可用软件集合,设计并实现了利用交互界面深入挖掘程序中的可向量化语句,优化生成代码中的向量化语句,提高生成代码的执行效率。如何充分发挥高性能计算机的计算能力,对增强系统可用性和扩展应用范围,提供有效的辅助手段和工具支持具有重要的意义。

当前高性能计算机的构建采用的CPU芯片中集成的单指令多数据流(Single Instruction Multiple Data, SIMD)短向量功能部件能够有效提升系统的整体计算能力,但手工改写/编写高质量的向量化代码对程序员是一种挑战。一方面,由于SIMD并行性的发掘往往需要进行一系列的代码变换与优化,手工向量识别过程对程序员具有较高的要求,各种程序变换和优化措施的实施需要程序员对编译技术有深入的理解;另一方面,半个多世纪以来,计算机技术的飞速发展为研发人员积累了大量的经验和宝贵的财富。现有的标量计算机应用过程中,大量发挥过重要作用的优秀软件急需有效地转移到高性能计算系统上运行,但要让程序员手工变换或重新编写并行程序,则是一件既费时又费力的事情。

为了充分利用给定的CPU提供的SIMD短向量功能部件,减轻程序员手工变换或重新编写向量程序的压力,渐进式

到稿日期:2013-12-26 返修日期:2014-03-15 本文受核高基国家科技重大专项(2009ZX01036)资助。
赵 博(1989—),男,硕士生,主要研究方向为高性能计算、先进编译技术,E-mail:zhaobo197359@gmail.com;赵荣彩(1957—),男,博士,教授,博士生导师,主要研究方向为先进编译技术、软件逆向工程等;徐金龙(1985—),男,博士生,主要研究方向为高性能计算、先进编译技术;高 伟(1989—),男,硕士生,主要研究方向为高性能计算、先进编译技术。

智能回溯向量化代码调优架构通过对用户提交的串行程序进行程序分析和变换，采用串行程序分析、数据依赖分析、向量化分析等技术手段，根据分析结果对程序进行变换和优化，自动生成最终的向量化代码。该工作通过分析串行程序中潜在的并行性，将其变换为等价的向量化代码形式，大大简化了程序员的工作。如今，越来越多的处理器集成了 SIMD 扩展，现有的编译器大多也实现了自动向量化的功能[3]。

2 相关研究

利用微处理器的多媒体扩展实现非多媒体程序的向量化已成为提高程序性能的一种重要手段[4]。通过交互手段或者交互式界面实现并行代码生成和性能调优，是弥补自动并行化不足的重要手段。例如，清华大学的陈文光等开发了一个交互式的 Fortran77 并行化系统。此系统以 Polaris 系统为基础，具有强大的自动并行化底层支持，同时支持共享存储系统和分布存储系统。此系统通过提供一个友好的交互式工具，使用户与编译器紧密协作，在一定程度上弥补了完全自动并行化系统的不足。美国休斯顿大学基于 Open64 开发的 Openuh 编译器，也做了一个程序分析工具 Dragon Analysis Tool，该工具可以提供 C/Fortran77/90 程序的细节信息，可包括 OpenMP/MPI 结构，它利用 Open64 分析和能力优势，以图形的方式为用户提供与源码相关的信息，如控制流、调用关系、依赖关系等。Dragon 能将剖析和调试协作分析提供给用户，使其获得更丰富的信息。这些交互式系统与交互式向量化要处理的对象和达到的目标不同，但对交互式向量化有一定的参考价值。

Intel 公司提供了 2011 版的 Parallel Studio 套装软件，其中主要有 3 部分，Parallel Composer 模块包含了 C/C++编译器、IPP 性能库、TBB 多线程开发库，Parallel Inspector 模块主要是多线程正确性和内存正确性的检查工具，Parallel Amplifier 模块主要是多线程程序的性能分析工具。3 个模块可以依次使用，即先使用 Parallel Composer 实现最基础的并行化开发，然后通过 Parallel Inspector 查找错误并改正，最后通过 Parallel Amplifier 对程序进行优化。虽然 Parallel Studio 提供了较为完整的并行化编译调试流程，但是其使用过于繁琐，而且仅仅提供了对多线程程序的支持。但即使如此，其设计思路对我们的向量化代码性能调优仍具有重要的参考价值。

3 基本组成及工作原理

“渐进式智能回溯向量化代码调优架构”面向高性能巨型计算机系统研发，以充分发挥巨型计算机系统的计算能力、提高系统的好用性、扩充可用软件集合等为研发目标，实现深度发掘程序向量化的目的。“渐进式智能回溯向量化代码调优方法”总体结构如图 1 所示。

向量化代码性能调优交互界面，一方面结合了现有的自动向量化代码生成和优化技术，为程序员提供详细的有针对性的向量化编译提示信息，信息的输出满足了直观、清晰、有用的要求；另一方面，将程序员也作为向量化代码生成和性能优化的重要组成，使程序员能够积极和高效地参与到向量化代码的生成和性能优化中，将程序员所掌握的应用程序特点信息以较为直接的方式传递给向量化工具，进一步帮助自动向量化提高识别率。

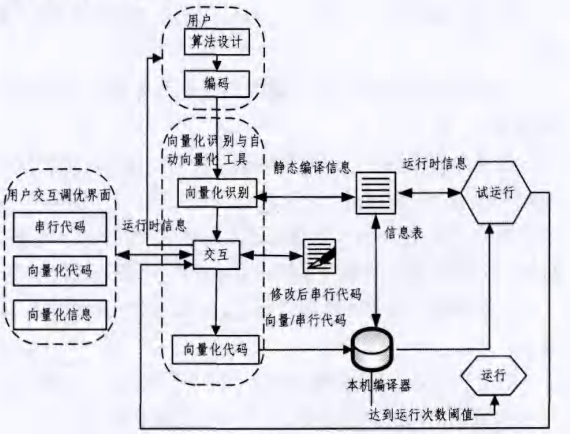

图 1 渐进式智能回溯向量化代码调优架构

3.1 界面

主要功能界面组成如图 2 所示，主要分成 4 个部分显示，最左边的部分是程序列表，程序员可以选择某个程序进行必要的变换；中间部分是程序源码的显示，程序员可以对程序源码做适当的交互式变换，此部分可读可写；右上部分为向量化后的代码显示，此部分只显示向量化结果，向量化后的代码对程序的改动较大，并不主张程序员去修改；最下面的部分显示的是从向量化工具中得到的程序信息，这些信息主要包括循环是否向量化成功以及未向量化成功的原因，通过这些信息可以更好地帮助程序员进行交互式变换。通过用户自定义参数配置，不同的文件采用不同的向量化参数，深度挖掘程序的向量化特性。

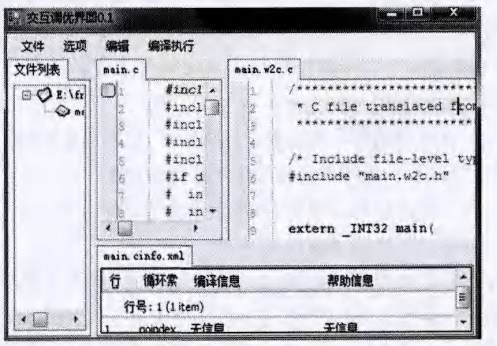

图 2 交互式变换的实现

工作原理：通过上述的交互式变换工具，可以将程序的编译信息与程序员的应用层信息结合起来，以帮助程序员更加准确地理解程序中循环的特点，从而可选择一种相对最有利的变换来提高程序的执行效率。此交互工具既可充分分析程序特点，开发程序的向量化能力，又可大大提高修改程序的效率，对向量化程序员来说是很适用的一个工具。

3.2 静态调优模块

报告的静态编译信息包括向量化报告和阻碍向量化原因报告两部分。其中，向量化报告内容包括向量化代价收益分析结果、静态工作量分析结果、数据引用连续性分析结果、内存地址对齐分析结果等信息；阻碍向量化原因报告包括不支持的循环（含有函数调用，汇编指令等）、不支持的数据类型、不支持的运算类型、不支持的控制流结构（结构化和非结构

化)、阻碍向量化的依赖关系(未知情况和 SCC)等信息。

主要功能:

根据编译指示语句的使用,将编译指示语句分为以下几类:

1. 强制性编译指示,包括强制向量化与强制不向量化编译指示。

强制性编译指示可以使向量化过程不经过向量化预分析过程。在进行强制向量化的时候,将不再进行对齐分析、连续性分析和依赖关系分析,而直接进入向量化代码的变换生成;强制不向量化则按照源程序形式直接输出,不经过任何变换。

2. 数据访问的对齐性编译指示,包括不对齐和对齐编译指示。

在含有指针、结构体的情况下,根据反馈式性能调优可以分析出数据访问的对齐性息,这时可通过数据访问的对齐性编译指示直接说明数据访存是否对齐,使自动向量化直接跳过对齐分析步骤,进行后续的处理。在加入数据访存对齐编译指示后,如果循环不满足可向量化候选循环或者存在阻碍向量化的依赖等情况,则循环仍然不能向量化。当数据访存不对齐时,可根据给出的对齐偏移量直接进入如数组填充等相关的对齐优化阶段,对非对齐访问进行向量化代码生成。

3. 消除数据依赖关系编译指示,包括消除数据依赖关系图的冗余边和冗余回边编译指示。

阻碍向量化的一个重要因素是循环中存在不可破除的依赖环,根据第 3 节所论述的自动向量化数据依赖分析判断方法,通过建立数据依赖关系图并找出其中的强连通分量,强连通分量之外的语句可以被向量化。但在实际应用中,影响数据依赖关系图建立的因素很多,如运行时所获得的输入等都使依赖关系的判断产生不确定性。在分析通常的数据依赖关系时,采用的都是保守方式,假定相关语句之间存在数据依赖,在数据依赖关系图中增加了对应的冗余边和冗余回边而无法进行向量化。

4. 各种向量化相关的编译优化指示,包括函数内联、设定循环展开因子、循环不变量外提等优化编译指示。

在自动向量化过程中,有很多因素都影响了对程序的分析。如第 2 节中关于指针的对齐性和连续性分析,如果函数调用的参数为指针,则随着指针参数的不同,其传递进入被调过程的对齐信息不一定相同。在这种情况下,若采用专门的过程间分析来优化指针的对齐偏移量将非常复杂,实际性能也往往不好。而如果在过程调用前加入函数内联编译指示,将每次调用都内联编译,则大大简化了分析过程,可以在主函数中对不同的调用过程进行向量化变换和优化。

工作原理:在向量性能调优界面中,可以根据静态或者反馈调优的分析结果,在源代码界面窗口中,对源程序的相应位置直接添加向量化编译指示语句,指导自动向量化工具根据指示语句完成 SIMD 向量化代码生成,同时不影响自动向量化工具完成编译指示语句之外原有的 SIMD 自动向量化优化功能。

3.3 动态调优模块

动态调试包括获取程序剖面信息和动态插桩两部分。其中,获取程序剖面信息指的是通过底层运行时系统提供的程序运行信息,向用户提示未向量化循环的实际运行时间等信息,辅助用户确定热点循环;动态插桩是指通过插桩收集程序静态编译时无法确定的数据依赖关系等信息(通过动态插桩获得循环中存在结构体或指针情况下的访存地址信息,通过分析访存地址获取依赖关系、对齐信息和连续访问等信息),以决定再次编译时是否进行向量化。动态调优用到的反馈式编译技术根据当前程序的运行趋势来改变后续动作的技术[5,6]。采用反馈式编译优化技术,能够获取程序运行时的连续性 Profile 信息[7]。动态编译过程分成 3 步:

第 1 步:动态插桩。在首次编译时,根据用户需要,向源程序进行插桩。这时用户可以自行选择所需要的编译器编译程序,生成可执行文件。根据所要收集的信息的不同,选择适当的插桩位置插入收集信息的代码,生成带有插桩函数的源代码文件。

第 2 步:试运行。对于插桩后编译生成的可执行文件进行试运行。在程序的试运行过程中利用插桩代码收集并分析程序运行时的信息,并在程序结束时将 profile 信息存至 profile 信息文件,需要注意的是,这里的 profile 信息是已经将向量化识别分析后的结果信息,而不是未加工的运行时信息。

第 3 步:反馈或交互。在第二次编译时,读取 profile 信息文件。为了能够更好地实现编译器与用户的交互,弥补编译器自动化方面的不足,profile 信息一方面以某种形式提供给编译器进行向量代码的自动生成;另一方面通过特定的函数接口将 profile 信息提供给用户,可以通过调用接口函数在交互界面呈现给用户,由用户通过编译指示等方式对代码进行一些较高层次的修改(如算法级的修改),从而更利于编译器的向量识别和优化。

对静态编译无法确定并且程序员通过交互调优仍无法解决的问题,通过程序员设置插桩标识,在源代码和生成的向量化程序中插桩,进行动态调优。流程如下:第一次编译时,如果需要动态调优,则添加相应的-fb-create 参数进行编译。首先,编译器对源程序进行 SLP 静态分析,之后对未向量化的循环进行插桩位置分析并添加相应的插桩函数,进入代码自动生成阶段。编译产生可执行文件后,进行第一次试运行,试运行程序会自动收集相应循环的剖面信息和动态插桩等信息,并生成反馈信息文件。进行第二次编译时,则添加-fb-opt 参数,由编译器读取反馈文件获取循环相关信息,以此为指导对相应循环进行向量化,自动生成向量化代码,最后产生可执行程序。若进行交互调优,则将循环部分运行时间以及对应循环中动态分析的依赖关系等信息提供给用户,辅助其进行交互调优。目前已实现对未向量化循环的插桩位置分析,进而可以自动地对未向量化循环进行插桩。

4 关键技术和创新点

4.1 主要特点

交互调优方式采用静态分析与动态反馈相结合的方法,使得用户能够积极参与程序的并行性发掘,获得性能更优的并行程序。

4.2 主要关键技术

4.2.1 静态调优

在向量性能调优界面中,可以根据静态或者反馈调优的分析结果,在源代码界面窗口中对源程序的相应位置直接添加向量化编译指示语句,指导自动向量化工具根据指示语句完成 SIMD 向量化代码生成,同时不影响自动向量化工具完成编译指示语句之外原有的 SIMD 自动向量化优化功能。

基于编译指示语句的向量化代码处理过程如下：

(1)前端指示语句的识别：基于 Open64 的编译前端，加入对 SIMD 向量化编译指示语句的识别和分析，确定编译指示语句作用域内的语句序列，将源代码转换成相应的中间表示；

(2)指示语句的分析和预处理：根据编译指示语句，跳过相关的约束条件，强制执行向量化变换过程，或者跳过相关的分析过程不进行向量化变换与优化；

(3)向量化代码的生成：根据分析预处理阶段所获得的信息，调用 SIMD 向量化代码生成功能，将中间代码转换成为向量化代码的中间表示。

4.2.2 动态调优

自动向量化工具收集串行程序中 SIMD 并行时所需要的循环热点信息以及数据依赖信息和引用连续、对齐信息，并对程序中的循环部分进行插桩，一方面获取循环运行时执行时间、执行次数、迭代总次数、单次执行时间、循环嵌套等热点信息，用于确定程序的热点循环，通过交互指导用户针对热点循环代码进行优化；另一方面获取运行时循环内地址 profile 信息，用于循环内的地址引用的连续和对齐分析、依赖关系分析，从而精确分析循环内的依赖关系等制约向量化的关键因素，进一步提高 SIMD 向量的识别率和程序效率。动态调优过程如图 3 所示。

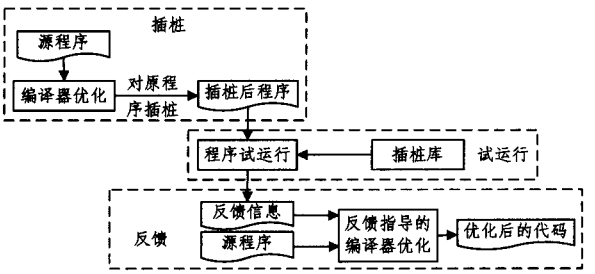

图 3 动态调优过程

4.2.3 动静结合

传统的自动向量化编译器无法产生对用户有意义的调优指导信息，这就增加了用户使用和优化 SIMD 代码的难度。如 INTEL 提供的自动向量化工具 Icc/Ifort 完全依赖自己的分析能力，缺少与用户有效沟通的途径，无法充分挖掘程序中蕴含的 SIMD 并行性。

我们采用静态向量识别信息与动态程序运行信息相结合的方式，为用户提供准确有用的调优信息，并基于调优信息敏感显示的多窗口技术以及面向行业的扩展向量化编译指示集合，提供科学高效的向量化程序优化环境。动静结合调优模型如图 4 所示。

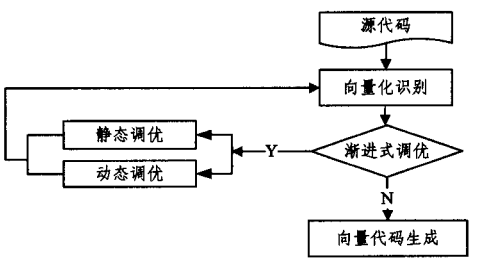

图 4 动静结合调优模型

4.3 本文的创新点

1. 设计实现了 SIMD 性能调优多级框架

支持程序级调优和文件级 SIMD 性能调优，用户既可通过对程序进行向量化参数的统一设置来完成程序级调优，也可对单个文件的向量化参数进行单独配置，实现细粒度的向量化性能发掘；同时在调优过程中，对源程序、向量化程序和向量化编译提示信息进行多窗口的联动显示，帮助获得高效的向量化代码。

2. 提出了静态与动态相结合的 SIMD 调优方法

静态分析时，用户根据向量化提示信息对循环添加强制向量化、强制不向量化等编译指示；在静态分析不足时，采用动态反馈方式，通过插桩获得运行时数据依赖关系、循环内数据对齐与连续访问等剖面信息，并用获得的信息指导精确的性能调优。

5 实验结果及分析

5.1 测试平台

渐进式智能回溯向量化代码调优架构，适用于特定体系结构的带有 SIMD 扩展的实验平台。本节的实验正是基于此平台完成的。编译环境为 linux 操作系统，版本为 Readhat Enterprise 5，实验平台 CPU 主频为 2.0GHz，向量寄存器位数为 256 位。

5.2 测试程序说明

本实验采用的测试集为 spec2000 的部分测试程序，实验时先将源程序转化为向量化程序，然后再用基础编译器编译成二进制代码并在实验平台上运行，最后用串行程序的运行时间除以向量化程序的运行时间便得到向量化的加速比。本实验在源程序转换为向量化程序时在架构的界面上操作实现，具体的变换过程需要根据编译提示信息和试运行反馈结果进行调整。主要包括的测试样例如表 1 所列。

表 1 测试程序集

测试程序	程序说明
mcf	组合优化/单点轮换调度
mgrid	多网格方法求解，3D 位势场
art	神经网络模拟，自适应推理
equak	有限元模拟，地震模型
swim	浅水模型
bzip	压缩文件
gzip	压缩文件

5.3 结果及分析

本文主要对 spec2000 中的 mcf 等 7 个例子的核心循环进行分析，针对不同的测试程序，单独采用静态调优、动态调优以及动静结合调优的方式进行向量化性能调优，通过测试发现，渐进式智能回溯向量化代码调优架构能够有效地提升程序向量化的性能。这是因为相对于单一地进行静态调优或者动态调优，使用结合动态的方法能够根据程序的试运行信息对源程序的结构和编译选项进行调整，实现更优的编译命令组合，得到更优的结果，这充分说明了本实验架构的可行性和有效性。

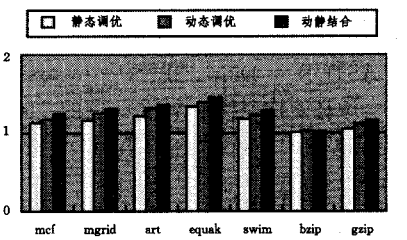

图 5 实验结果

(下转第 58 页)

Gluster 基本上能够满足，但是有一个限制，即在平衡前，对于扩容前创建的老目录不能使用新增加的存储空间，只有新创建的目录才能使用所有存储设备。这一点导致 Gluster 扩容对于高能物理应用来说基本上是无用的，因为实验的目录基本上都是固定的。为此，提出基于版本的扩容方案，扩容后，系统存在新老等多个版本，文件定位在多个版本间进行，使得老目录也可以使用新增加的设备。当平衡完成后，所有版本合并为一个。通过扩容方案的修改，使得 Gluster 扩容能较好地满足高能物理的需求。

3.2.4 应用实例与性能

虽然 Gluster 具有很好的架构设计，但是针对高能物理计算来说，其元数据管理、容错与可靠性、扩容等方面还不能很好地满足需求。在对这些不足进行优化后，将 Gluster 应用到一个实际的高能物理实验（羊八井宇宙线实验）计算中，目前管理了 4 台存储服务器，186TB 的存储空间，近 1500 万的文件，支持 500 多个并发作业访问。实际运行显示，Gluster 能够满足计算的 I/O 需求，计算节点上基本不会出现 IO-Wait，CPU 利用率高。图 7 显示了其中一台存储服务器的数据访问性能，其优化措施与 2.2 节一致，最高时接近 1GB/sec，跑满了整个万兆网络带宽，说明 Gluster 具有很好的数据访问性能。

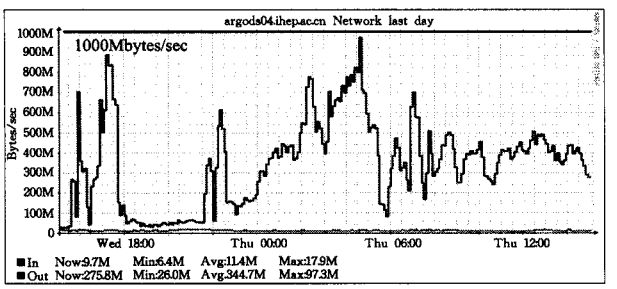

图 7 Gluster 存储服务器性能

结束语 海量的高能物理实验数据对于存储系统提出了很高的要求，既要求有极高的访问性能，还要有良好的可管理性。高能物理领域主要以自主开发、开源软件为主来构建海量存储系统。各种系统有各自的特点，每个应用也有自己的需求，因此希望一种存储系统满足各种应用的需求是不现实的。因此，必须要深入分析应用和存储系统的特点，并进行应用优化才能达到理想的效果。高能物理计算的需求一直在不断变化，存储系统的结构、技术和方法也会随着应用的需求而变化。

参 考 文 献

[1] WLCG -Worldwide LHC Computing Grid [OL]. http://lcg.web.cern.ch/LCG,2013.7

[2] Fuhrmann P, Gülzow V. dCache, storage system for the future [C]// Euro-Par 2006 Parallel Processing. Springer Berlin Heidelberg, 2006: 1106-1113

[3] Peters A J, Janyst L. Exabyte Scale Storage at CERN[J]. Journal of Physics Conference Series, 2011, 331(5)

[4] Schmuck F, Haskin R. GPFS: A Shared-Disk File System for Large Computing Clusters[C]// Proceedings of the Conference on File and Storage Technologies (FAST'02). Monterey, CA, January 2002: 231-244

[5] Schwan P. Lustre: Building a file system for 1000-node clusters [C]// Proceedings of the 2003 Linux Symposium. 2003

[6] IOzone Filesystem Benchmark[OL]. http://www.iozone.org

[7] Shakshober D J. Choosing an I/O scheduler for Red Hat Enterprise Linux 4 and the 2.6 kernel [M]. Red Hat magazine, 2005

[8] Gluster web site[OL]. http://www.gluster.org

[9] 罗象宏，舒继武. 存储系统中的纠删码研究综述[J]. 计算机研究与发展，2012，49(1)：1-11

（上接第 53 页）

结束语 本文利用静态调优和动态调优相结合的方式创建了一个渐进式智能回溯向量化代码调优架构。该架构能够有效地加强用户和编译器之间的交互，极大地减轻程序员编写向量化代码的压力，能够产生高效、易于阅读的向量化代码。利用该系统进行交互式调优，可以便捷、直观地进行代码级调优工作。交互调优界面中的向量化报告和阻碍向量化原因报告对程序员进行代码调试有很大的帮助。

但是我们的工作更多地还要依赖于自动向量化关键技术方法的发展，这样才能有更为丰富的调优方法和手段。我们的架构中用到的静态调优和动态调优的方法，很多是依赖于当前自动化技术的发展水平来定的。我们期待自动向量化技术能有更大的发展和突破，这样我们的调优架构也将更为丰富。

参 考 文 献

[1] Stewart J. An investigation of SIMD instruction sets. University of Ballarat School of Information Technology and Mathematical Sciences, 2005. [OL]. http://noisymime.org/blogimages/SIMD.pdf

[2] Nuzman D, Rosen I, Zaks A. Auto-Vectorization of interleaved data for SIMD[C]// Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation. Ottawa: ACM Press, 2006: 132-143

[3] 魏帅，赵荣彩，姚远. 面向 SLP 的多重循环向量化[J]. 软件学报，2012(7)：1717-1728

[4] 李玉祥，施慧，陈莉. 面向非多媒体程序的 SIMD 向量化算法的研究及改进[J]. 小型微型计算机系统，2009(10)：1927-1935

[5] 白书敬，李中升，漆锋滨. 反馈式编译优化技术浅析[J]. 高性能计算技术，2005，10(5)：1-5

[6] 郝云龙，赵荣彩，侯永生，等. 反馈式编译在循环级性能分析中的应用[J]. 计算机工程，2011，5(5)：32-34

[7] 姚远，赵荣彩. 基于 Profile 信息的连续性分析算法及其优化[J]. 计算机工程，2012(9)：28-31