软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(Suppl.(2)):70−79

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于神威蓝光处理器的向量数学软件包*

解庆春[1,2], 张云泉[1,3], 李 焱[2], 逄仁波[4], 吴再龙[5], 鲁永泉[1], 高鹏东[1]

[1](中国传媒大学 高性能计算中心,北京 100024)

[2](中国科学院 软件研究所 并行软件与计算科学实验室,北京 100190)

[3](中国科学院 计算技术研究所 计算机体系结构国家重点实验室,北京 100190)

[4](国家海洋环境预报中心 网络与计算机部,北京 100081)

[5](中国海洋大学 信息科学与工程学院,山东 青岛 266100)

通讯作者: 解庆春, E-mail: qingchun@iscas.ac.cn

摘 要: 首先介绍了 SIMD 扩展技术,并分析了使用 SIMD 扩展的 3 种方式,认为通过调用特定目标平台优化的第三方库是应用领域软件开发者快速开发高效并行程序的较好的方式;其次,介绍了国产神威处理器 SW-1600 平台,并利用 SIMD 扩展和循环展开等技术开发了 SW-VML(SW Vector Math Library),开发过程中提出了访存对界、简化向量条件分支的优化方法,解决了非对界访存、向量与标量数组转换影响性能的问题,并根据 SW 编译器对 OpenMP 的支持,开发了多线程 OpenMp 版;最后,在 SW-1600 平台上采用不同向量规模对 SW-VML 进行了测试,测试结果显示,SIMD 向量化相对于串行程序加速比为 2.08,4 线程相对单线程平均加速比为 2.26.SW-VML 是在国产神威系列处理器上开发高效程序的向量函数软件包,也是在神威蓝光高性能计算平台单计算节点开发高性能程序的基础软件工具包.

关键词: SIMD 扩展;神威处理器 SW_1600;向量数学库;向量化

中文引用格式: 解庆春,张云泉,李焱,逄仁波,吴再龙,鲁永泉,高鹏东.基于神威蓝光处理器的向量数学软件包.软件学报,2014,25(Suppl.(2)):70−79. http://www.jos.org.cn/1000-9825/14025.htm

英文引用格式: Xie QC, Zhang YQ, Li Y, Pang RB, Wu ZL, Lu YQ, Gao PD. Package of the vector math library based on the sunway processor. Ruan Jian Xue Bao/Journal of Software, 2014,25(Suppl.(2)):70−79 (in Chinese). http://www.jos.org.cn/1000-9825/14025.htm

Package of the Vector Math Library Based on the Sunway Processor

XIE Qing-Chun[1,2], ZHANG Yun-Quan[1,2], LI Yan[2], PANG Ren-Bo[4], WU Zai-Long[5], LU Yong-Quan[1], GAO Peng-Dong[1]

[1](High Performance Computing Center, Communication University of China, Beijing 100024, China)

[2](Laboratory of Parallel Computing, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](State Key Laboratory of Computer Architecture, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[4](Department of Computer and Network, National Marine Environmental Forecasting Center, Beijing 100081, China)

[5](School of Information Sceience and Technology, The Ocean University of China, Qingdao 266100, China)

Corresponding author: XIE Qing-Chun, E-mail: qingchun@iscas.ac.cn

Abstract: This paper first introduces the SIMD (single instruction multiple data) extension technology and presents three ways to use SIMD instructions. It is considered that calling the third party library, which is optimized for the target platform by using those

* 基金项目: 国家自然科学基金(61133005, 61272136); 国家高技术研究发展计划(863)(2012AA010902, 2012AA010903); 中国科学院研究生科技创新与社会实践资助

收稿时间: 2013-08-05; 定稿时间: 2014-03-13

instructions, is the best way to benefit the developers. Next, it introduces the China-developed SW-1600 CPU, and a software package called SW-VML, which consists of many mathematical functions, by using the SIMD extension technology. In order to solve the additional overhead caused by unaligned address access and transformation between vector and scalar array, the paper provides some performance optimized methods, such as aligned address access, simplifying vector condition branch and loop unrolling. An upgrade to SW-VML is also offered to support multi-thread with OpenMP. Finally, functions in the package are tested using arrays of different sizes on SW-1600，and the test results show that high performance is achieved with the technology of the SIMD vectorization. Compared with the traditional methods of the scalar calculation, the average speedup is up to 2.06. The performance speedup of package using 4 threads is up to 2.26 compared to using a single thread. SW-VML is a common vector function package for domestic Sunway processor series, and it can be used as a basic toolkit which is beneficial to high performance computing on Sunway platform.

Key words: SIMD extension; sunway processor SW_1600; vector math libarary; vectorization

随着计算机科学技术的发展,仅仅通过提高处理器主频来提高计算性能的方法已经远远不能满足目前的信息处理需求.由于多媒体程序的任务操作自身具有计算密集,数据长度短,易于数据并行等特点[1],通过在通用处理器内增加基于单指令流多数据流 SIMD(single instruction multiple data)功能单元成为提升性能的主要解决方案[2,3],同时,SIMD 扩展应运而生.自从 1996 年 Intel 在奔腾处理器上集成了 MMX 后,越来越多的通用处理器上集成了 SIMD 扩展.与此同时,随着人们对 SIMD 技术的探索,SIMD 扩展越来越多地应用于高性能浮点计算中,如数值代数、向量矩阵运算、数字信号处理,生物数据分析领域等.

目前主要有 3 种使用 SIMD 扩展的方法,分别是编译器支持的自动向量化、SIMD 扩展指令编程和基于 SIMD 指令开发的第三方库.从开发成本、效率和性能提高对比分析,通过调用优化的第三方库,为应用领域软件开发者快速开发高效并行程序和向量程序较好的方法.同时,这也是各大处理器厂商推出自己的向量数序函数库的原因之一.所以,在国产处理器平台上利用其提供的 SIMD 扩展指令集,开发一套根据该平台结构特征给予优化的高性能向量函数库非常重要.该函数库可以为在此平台进行应用软件开发者提供高性能的基础函数接口.提高开发效率的同时也保证了开发程序的性能.

本文第 1 节介绍 SIMD 扩展和应用方式.第 2 节介绍神威处理器 sw_1600 平台和对 SIMD 扩展的支持,并对开发 SW-VML 的过程中提出访存对界、向量分支简化和多线程等优化算法.第 3 节对 SW-VML 进行详细的性能评测和分析.第 4 节对全文进行总结.

1 SIMD 扩展指令与应用

1.1 SIMD扩展指令

SIMD 扩展是指针对普通处理器的,在普通处理器中增配 SIMD 执行部件、SIMD 寄存器堆和一套 SIMD 多媒体扩展指令集[4].处理器上具有能进行向量计算的处理单元和寄存器,随着 SIMD 技术的发展,一些扩展体系结构开始提供位宽更大的执行部件和寄存器.目前很多微处理器都有自己的 SIMD 扩展指令集,自从 1996 年 Intel 在奔腾处理器上集成了 MMX 之后,越来越多 SIMD 扩展指令集相继出现,例如,Intel 后来推出了 SSE 系列的 SIMD 扩展指令集,AMD 提供了 3DNOW!,SSE5 指令集,SUN 提供的 VIS、ARM 提供 NEON 通用 SIMD 引擎,PowerPC 提供 AltiVec 扩展[5].这些指令提供向前兼容.

SIMD 用来操作数据级别的并行,在执行 SIMD 指令操作时,一条指令可以同时对多个数据(组成一个向量)进行运算,如图 1 所示为最常见的 SSE 指令的垂直计算形式和传统标量指令计算形式的比较,在一个 128 位的向量寄存器下,每执行一次向量操作相当于 4 次 float/integer 或 2 次 double 操作(C 语言数据类型定义),理论上,SIMD 指令性能是普通标量运算的 4 倍.

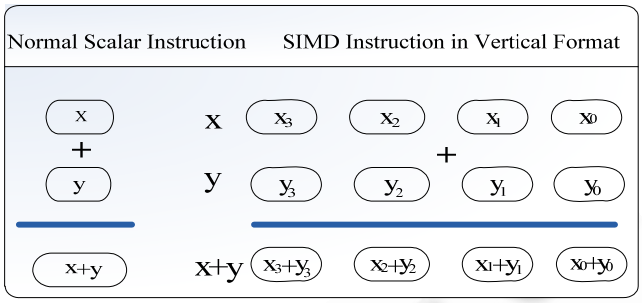

图1 标量处理和 SIMD 操作之间的比较

1.2 SIMD向量化

编译器将串行代码翻译成 SIMD 指令的过程被称为 SIMD 向量化或 SIMD 并行化,SIMD 向量化是针对处理器扩展或者 SIMD 扩展的,是自动向量编译优化中最核心的部分[6].因为通用处理器中,集成的多媒体部件与传统的向量处理器之间存在着显著差异,致使传统的向量化技术无法被简单地移植到 SIMD 编译优化中,所以目前编译器对 SIMD 指令的支持并没有达到足够令人满意的程度.不过,可以通过编译制导语句、手工编写汇编代码、Intrinsic Functions 等方法显示地使用 SIMD 扩展指令来协助编译器完成 SIMD 向量化.

1.3 SIMD Intrinsic Functions

Intrinsic Functions 是处理器厂商随着编译器一块发布的一套接口,方便使用汇编语句,是对 SIMD 扩展指令的封装.这样,就可由编译器来完成向量寄存器的管理工作,降低开发并行程序复杂性.在编译的过程中,编译器会把 Intrinsic 接口函数转为处理器可以支持的 SIMD 扩展指令,同时可以获得与汇编几乎一样的性能.以 Intel SSE 中 SIMD 指令 addsubps 为例,它对应的 Intrinsic Functions 为_mm_addsub_ps,和普通函数调用一样,在需要的地方插入语句:w= _mm_addsub_ps(u,v);即可完成操作,虽然 Intrinsic Functions 的使用类似高级语言的函数调用,但不是真正的函数调用,它会被编译器直接翻译成对应的汇编指令.u 和 v 也不是函数的参数,而是 addsub 指令的两个操作数.

总之,Intrinsic Functions 它是介于高级语言(例如 C,C++)和汇编之间的一种产物,同时具有易编程和高性能的优点[7].

1.4 SIMD扩展的应用

SIMD 指令能够带来较低的功耗、较好的资源利用率、有效提高应用程序的性能和简化编写程序的复杂度.目前,主要存在 3 种利用 SIMD 扩展指令集来生成向量化代码的方法.

(1) 编译器自动向量化、或以向量化制导语句为前提的向量化

编译器对 SIMD 扩展指令集的支持,使之能够利用 SIMD 指令自动优化高级语言编写的代码,编译器的这一功能被称作 SIMD 编译优化.尽管多数商业编译器都宣称能够实现 SIMD 指令自动优化,但在实际应用上却并不尽如人意[8,9].通过编译制导语句指定的向量化是自动向量化的补充,但使用 SIMD 编译制导的向量化带来的性能提升有限,因为编译器为了保证程序正确,通常会绕过一些数据假相关语句的优化.

(2) 直接使用 SIMD 扩展指令

程序员可以通过嵌入汇编代码或 Intrinsic Functions,显式地使用 SIMD 扩展指令.但这种方法存在两方面问题,一方面,该方法需要程序员在清楚算法逻辑的前提下,还需要深入了解 SIMD 部件的体系结构和复杂的 SIMD 扩展指令,开发难度较大.另一方面,由于各种处理器中的 SIMD 部件和 SIMD 扩展指令各不相同,导致代码的可重用度低,移植性差[10,11].

(3) 第三方向量库

应用程序员在开发过程中直接调用已经开发并优化的第三方库,这些库的开发过程中或利用嵌入汇编,或使用 Intrinsic Functions 等方法,并针对具体的硬件体系结构给予优化.在开发过程中,程序员只需要了解应用程

序自身的算法逻辑,而不需要了解向量计算的细节.

从开发效率、性能和稳定性层面比较 3 种 SIMD 扩展使用的方式,调用第三方库,不失为应用领域软件开发者快速开发高效并行程序的较好方式,这也是各大处理器厂商推出自己基础函数库的原因之一.

2 神威平台的向量数学函数库

数学库主要完成科学和工程计算领域中常见的数值密集型运算.在很多应用领域如气象、地震、油藏模拟中具有举足轻重的作用.数学库可以帮助用户减少计算开销,增加计算规模,增强可移植性,提高可扩展性.其中,向量数学库是数学库的一部分,主要用来解决高度结构化、需要大规模向量或者矩阵计算的问题[12].各厂家都针对自己的处理器开发了向量数学库,例如,如 Intel 公司的 MKL 数学库中的 VML,AMD 公司的 ACML 数学库中的向量化函数包,IBM 公司针对 cell 处理器推出的向量化 SDK ESSL,SUN 为 VIS(visual instruction set)扩展提供了一套信号处理库.其中这些库中函数根源是 Cephes 库的函数[12,13]本.

应用程序开发者只需要在使用向量操作的时候,调用向量数学函数库中的对应接口函数就可以带来较高的性能提升,开发难度系数大大降低.除此之外,特定平台的向量函数库都根据特定平台的软硬件结构特性,对函数的性能做了优化,确保链接该库的应用程序在性能上也更加高效,此项工作可是说是一劳永逸.因此,在国产处理器平台上给开发者提供高性能的多核版本向量函数库十分必要.

2.1 SW-1600平台

本文研究的目标平台是国产高性能处理器 SW-1600,目前采用该处理器的神威蓝光高性能计算机已部署在山东省超级计算中心,并已投入使用.SW-1600 增加了支持 SIMD 扩展的执行部件、SIMD 寄存器堆和一套 SIMD 多媒体扩展指令集,编译器还支持一些标准 C 语言的扩展数据类型和 Intrinsic Functions.

SW_1600 处理器支持的 SIMD 处理器长度为 256 位,支持 64×4 的双精度浮点运算,32×4 的单精度浮点运算,32×8 的定点运算.同时,在标准 C 的基础上扩展了 4 种数据类型:intv8(8 个整型),uintv8(8 个无符号整型),floatv4(4 个单精度浮点)和 doublev4(4 个双精度浮点).寄存器存储位宽如图 2 所示.

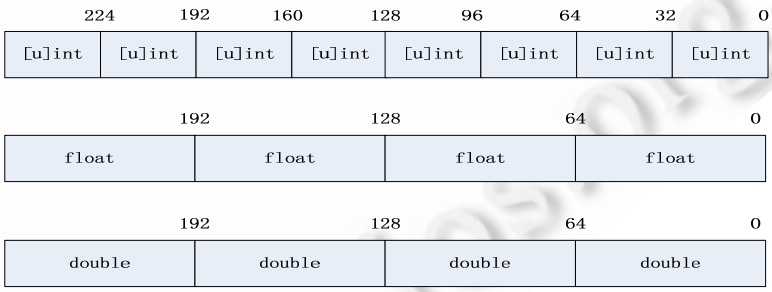

图2 向量寄存器存储宽度

浮点和定点运算,理论上可以达到 4~8 倍的加速比,但是在应用中,因为内存对界、向量变量条件判断开销,再加上数据的 I/O、带宽、流水线和 Cache 容量限制等问题,正常情况下加速比低于理论值.

2.2 基于SW-1600平台的向量数学库

本文在国产 SW-1600 平台上充分利用系统提供 SIMD 扩展,开发了多个基础向量函数,并结合体系结构特征应用多种方法进行了优化.函数接口和 Intel 的 MKL 相似,支持 float 和 double 两种数据类型,共有 6 类函数,分别是算术函数,指数函数、幂函数、三角函数,双曲线函数、取整函数,命名为 SW-VML(SW 向量数学库).这些函数的共同特点是数据高度结构化、数据之间没有依赖,或者有较少依赖.在实际应用中,应用程序开发者可以直接调用该库来充分利用 SIMD 硬件资源,提高程序性能和开发效率.所以,SW-VML 可以作为神威蓝光高性能计算平台单计算节点的基础软件之一,具有重要的理论意义和现实意义.

2.3 优化方法

2.3.1 访存对界优化

SIMD 扩展指令集引入的一个重要问题是数据的对齐问题(alignment).对齐的数据读取有更多性能优势,所以,保证数据读取的对齐是首先要考虑的优化方法.对齐问题也是 SIMD 编译优化的一个难点[14].多媒体扩展指令集都希望处理的这些数据是对齐的,因为它们可以在一个 Cache Line(一般 Cache Line 的大小 32 字节)内快速得到需要的操作数据;否则,需要跨多个 Catch line,这样需要执行至少多次 Load 操作,并把数据进行拼接,这需要处理器等待几倍的访存时间,造成计算资源浪费严重[15].如图 3(a)所示,访问 *A* 的地址,读取连续 32 字节的数据,假设机器的 catch line 长度为 32 字节,但向量 *A* 并不是 32 字节对齐的,这时候需要两次访存操作,SW_1600 处理器在使用进行数据操作的时候,需要保证输入的标准类型变量为 16 字节(4×Sizeof(float))的单精度浮点运算)或 32 字节(双精度浮点运算)对界.对非对界的 SIMD Load 或 SIMD Store 操作会引发异常,执行的时候,操作系统会根据收到的异常信号将这些扩展操作拆分成标准类型的 Load/Store,这样会造成性能严重受影响.

本文使用的优化方法是对开始位置到最近的 32 字节对界部分进行标量化处理,之后再 SIMD 向量化.因为 SW-VML 中的函数操作的对象都是连续存储的数据,比如 *vsSin*(*num N*,*Input Vector A*,*Output Vector B*)函数,输入 *A* 是连续存储的浮点数据组成的向量,但不能保障用户传进的参数 A 的首地址是 32 字节对界的,在函数开始之前,判断是否是 32 字节对界,如果是,直接进行向量化访存,如图 3(b)所示.如果非对界,计算离开始位置最近的满足 32 字节地址对界的位置,该位置之前的元素做标量处理,对齐位置之后的元素开始,做 SIMD 向量化操作,如图 3(c)所示.由于 align 之前和 tail 之后的元素较少,不需要循环展开和向量化,都采取串行执行. 程序语句块如 3(d)所示, 对可以使用显示循环展开的优化技巧.

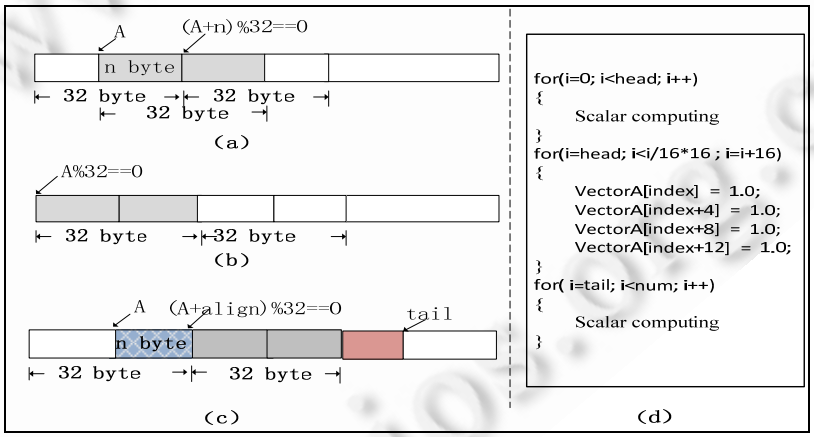

图3 非对界内存访问

2.3.2 化简向量分支

向量数学函数中待处理的元素之间很少有依赖,非常适合并行向量化,但是多数的函数算法本身比较复杂,条件判断分支较多,当不支持条件跳转的 SIMD 扩展指令遇到条件分支语句时,会影响向量操作的性能[5],简化向量分支也是 SIMD 向量化的优化重点.

在 SIMD 向量化过程中,如果遇到条件分支语句,常用的处理方法是把向量转化为标量数组,进行多次的标量操作,完成之后再转化标量数组结果为向量,继续完成后面的向量运算.如果函数的算法中条件判断较多,该处理方法就会使 SIMD 扩展不仅没有带来任何性能提升,还可能因为向量和标量之间的转化造成性能下降.

本文利用编译器给定的 SIMD 扩展位条件分支判断指令,通过逻辑运算,向量算术运算,常数在控制流上的传播等完成简化向量条件分支操作,避免向量和标准数组的相互转化,可以保持向量处理顺利流水.理论上,该方法相对标量和向量相互转化方法,性能提高 50%.比如以图 4(a)的语句块为例,向量化运算需要进行:1 次向量转化为标量操作,4 次取数据,4 次条件判断,4 次计算操作,1 次标量数组转向量操作.而是用该方法只需要 6 次

Op 即可.

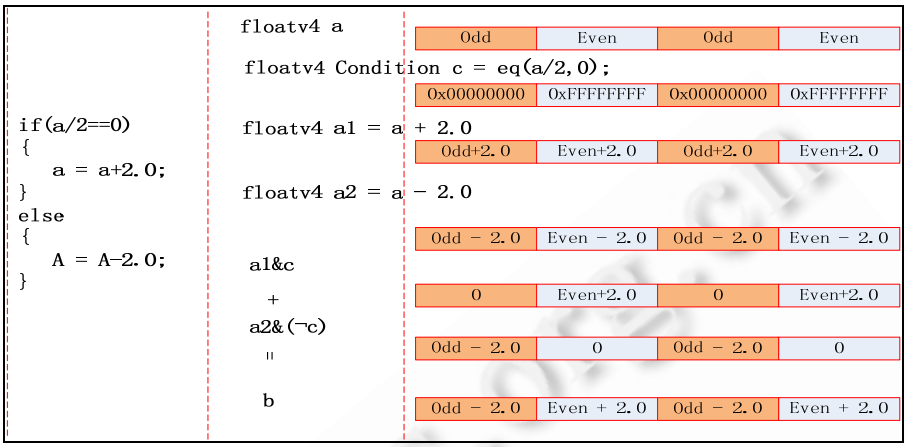

图4 SIMD 条件分支优化

2.3.3 多线程

向量库包含的这些函数的共同特点是数据高度结构化、数据之间很少有依赖,空间局部性强,非常适合并行化,所以考虑从线程级别进行并行优化.在多核处理器采用多线程并行,并行方法相对成熟,主流的有直接操纵轻量级低开销的线程 Pthread 方法,也有利用声明或制导语句通过编译器将串行的程序变换为并行程序的 OpenMP 方法.从硬件上,SW 处理器的每个核组包含 4 个核心,SW-1600 满足多线程运行的条件,同时神威编译器还支持 OpenMP 并行,所以,SW-VML 采用 OpenMP 多线程并行化.同时考虑到向量函数操作目标元素间的空间局部性,SW-VML 采取静态均匀分配线程任务.

2.4 OpenMP与SIMD扩展

OpenMP 是一种共享存储并行系统上的基于线程的编程模型,并不是一种新的编程语言,目前支持 OpenMP 并行化都是通过编译器可识别的编译制导语句来实现的,通过简单地添加 OpenMP 制导语句,来显式指导编译器进行优化,完成数据并行[16].

SW 处理器是四核共享存储系统,神威编译器支持 OpenMP.SW-VML 在核心内部利用 SIMD 扩展实现内层的 SIMD 向量化,实现数据级别并行;通过显式的外层 OpenMP 编译指导语句,实现核心与核心之间多线程并行,充分利用 SW-1600 的硬件特性来挖掘向量函数的性能.

3 实验结果与分析

3.1 实验平台

实验平台是国产高性能处理器 SW-1600,SW-1600 处理器包含 4 个核组,每核组 4 个核心,采用 4 核共享存储系统,系统总线是 128 位宽[17],提供 OpenMP 多线程编程支持.SW-VML 是单核组多线程版本.实验平台的配置信息见表 1.

表1 SW-1600 实验平台

处理器	SW-1600 0MHz
内存	1.7G
L1,L2 缓存	8 KB 的 L1 catch(指令和数据),96 KB 的 L2catch[18]
操作系统	Linux 2.6.15
测试函数	SW-VML,Cephes 库和 C 标准数学函数库
编译器	神威编译器(swcc 2.2.25)

3.2 SW-VML软件包的介绍

针对 SW-1600 开发的向量函数软件包中的接口和 Intel 的 MKL 相似,具体函数说明参考 MKL 使用说明手册.向量函数软件包分为 6 类,分别是算术函数,指数函数、幂函数、三角函数,双曲线函数,取整函数.函数列表见表 2.

表 2 SW-VML 函数库

函数名 / 类型	具体函数(函数名以v(vector)开头,其中? 可以是s:单精度浮点,d:双精度浮点)
算术函数	v?Add, v?Sub, v?Sqr, v?Mul, v?Abs
幂函数	v?Inv, v?Div, v?Sqrt, v?InvSqrt, v?Cbrt, v?InvCbrt, v?Pow2o3, v?Pow3o2, v?Pow, v?Powx, v?Hypot
指数函数	v?Exp, v?Expm1,v?Ln,v?Log10,v?Log1p
三角函数	v?Cos, v?Sin v?, SinCos, v?Tan, v?Acos, v?Asin, v?Atan, v?Atan2
双曲线函数	v?Cosh, v?Sinh, v?Tanh, v?Acosh, v?Asinh, v?Atanh
取整函数	v?Floor, v?Ceil, v?Trunc, v?Round, v?NearbyInt, v?Rint, v?Modf

v?Add,v?Sub,v?Sqr,v?Mul,v?Abs 等常见函数通过调用 C 语言编译器自带的对应函数接口来验证 SW-VML 正确性;其他 C 语言 math 库中缺乏的函数如 v?InvCbrt,v?NearbyInt,v?Expm1 等,我们采用开源的 Cephes Math Library 中相对应的接口来验证 SW-VML 正确性[19].通过测试软件包所有函数不同的向量规模,并在一定规模下,对所有函数的加速比加和求平均来描述软件包在该向量规模下的性能提升情况.

3.3 SIMD扩展性能提升

采用未优化的 netlib 或者 Cephes 库源码为基础,在试验平台编译后与 SW-VML 进行性能对比.使用 SIMD 扩展,在向量规模为 1.0E+6 时,向量化带来的平均加速比达到最好的 2.29,单精度可以达到 2.20,双精度达到 2.38,这里的平均加速比的计算方法是对软件包内 80 个函数在规模一定(这里为 1.0E+6)的情况下加和求平均,图 5 描述了不同规模下,测的 SW-VML 性能的平均加速比.9 种规模下平均加速比为 2.08.

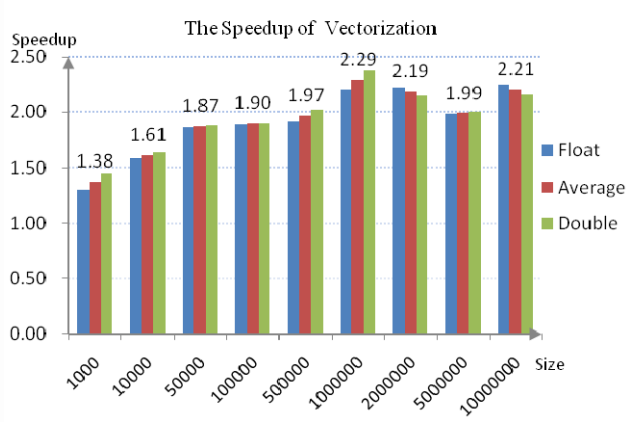

图5 SIMD 向量化性能提升

从图 5 可以看出,随着规模的不断扩大,整个函数库的性能也逐步提高,但达到一定标准后趋于稳定.这是因为 SIMD 扩展发挥强大作用的前提是具有较大的流数据,较高的计算访存比,这样可以通过计算掩藏因为函数对界、向量条件分支处理等向量化操作带来的开销.

图 6 描述了各函数在不同规模下的性能提升情况.从图 6 可以看到,在向量规模为 100 000 时,还存在指数和对数类函数 exp,ln,log10 等函数加速比小于 1 的情况,这是因为选用的这些串行算法都是基于较早的算法,不是太高效,其次这些函数本身比较复杂,在实现的过程中,条件判断,类型转换较多,这类函数也是下一步优化的重点.

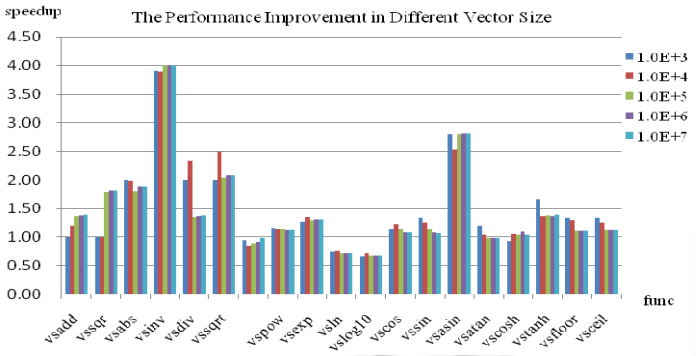

图 6 各函数在不同向量规模时 SIMD 向量化性能提升

3.4 多线程性能提升

SW-VML 采用 OpenMP 完成多线程版本,分别测试向量规模为 1 000,10 000,50 000,100 000,50 000, 1 000 000, 2 000 000,5 000 000,10 000 000 时单精度浮点和双精度浮点的性能.6 类规模的平均加速比为 2.26.其中向量规模为 2.0E+6 时,向量化带来的平均加速比达到最好的 2.84,单精度可以达到 2.90,双精度达到 2.78,这里的平均加速比的计算方法是对软件包内 80 个函数在规模一定的情况下对 1、4 线程加速比加和求平均获得.9 中规模下求平均加速比为 2.26.从图 7 可以看出,随着规模的不断扩大,整个函数库 4 线程的性能也逐步提高,但达到一定标准后趋于稳定.

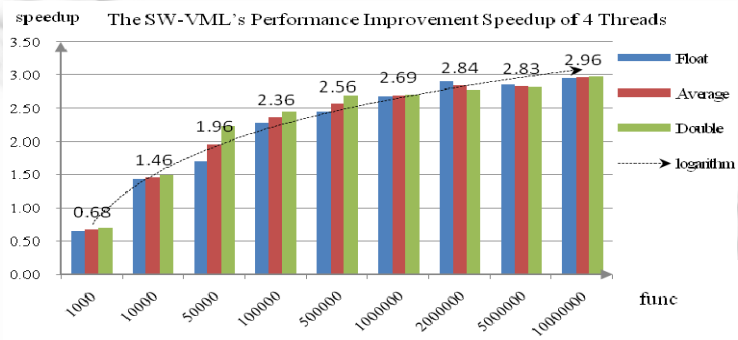

图 7 SW-VML 的 4 线程性能提升加速比

从函数库的每类函数中选择有代表性的函数接口,进行测试,图 8 描述了在向量规模为 1.0E+5 时,每个函数的单精度和双精度类型的 4 线程相对于 1 线程带来的性能提升.

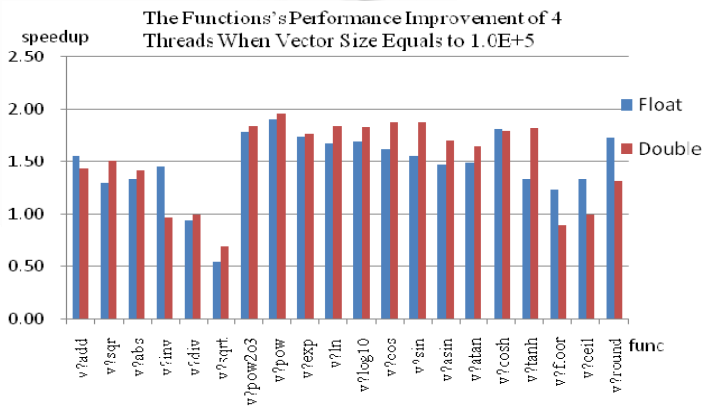

图 8 向量规模为 1.0E+5 时,各函数 4 线程性能提升加速比

从图 8 可以看到,在向量规模为 100 000 时,4 线程相对于 1 线程并行效果明显,加速比在 1.5 左右.虽然整体加速效果不错,但也有个别函数性能较差,比如 sqrt 函数,是因为在最后开根号时调用软件包的 v?Div 所致,函数互相调用性能较差的问题,也是下一步调优的重点.

对多数程序来说,通过编译制导语句来实现完全自动向量化和并行化对编译器来说十分困难,因为编译器很难做到既能保障效率,又能提高程序性能,在编译器静态分析的过程中,对任务相关或者数据相关的语句,编译器为确保结果的正确性而采取保守措施,只编译出非优化的代码.

图 9 描述了部分函数在不过测试向量规模下,4 线程相对 1 线程的并行加速比,总体趋势是:随着向量规模变大,加速效率也越好.这是因为向量规模大,数据计算与线程自身启动、维持管理等开销时间比例较大,线程创建、分配资源,线程计算、文件并行读写和线程同步等开销可以隐藏在大量的计算时间之内,如果向量规模较小,计算和线程开销的比例较小,加速不明显.

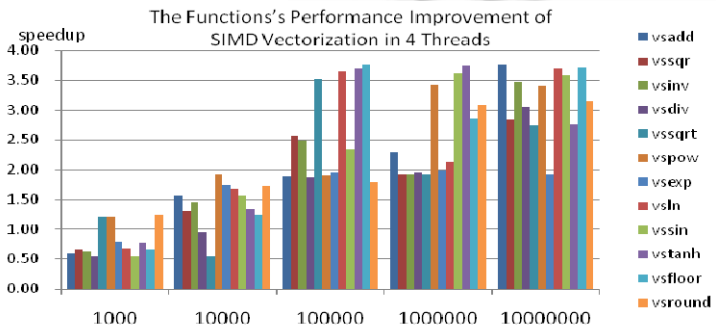

图 9 各函数在不同向量规模时 4 线程向量化性能提升

4 总结与展望

我们认为通过调用特定目标平台优化的第三方库是应用领域软件开发者快速开发高效并行程序的较好的方式.本文在国产 SW-1600 处理器上,应用神威编译器提供的 Intrinsic Functions 开发向量数学库,在函数开发过程中使用访存对界,向量分支简化处理,循环展开等优化方法,并根据该平台对 OpenMP 的支持,开发支持多线程的 OpenMP 版.并分别在 SW-1600 平台上,进行了测试,9 种测试规模下的平均加速比为 2.09,其中在向量规模为 1.0E+6 时,向量化带来的平均加速比达到 2.29.同时,9 种测试规模下 1、4 线程加速比为 2.26,其中向量规模为 2.0E+6 时,向量化带来的平均加速比达到最好的 2.84.

该工作为国产处理器提供常用的数学函数软件包,方便了行业领域应用程序在该平台上通过调用该库而提高开发效率和性能提升,具有重要的理论意义和实践意义.同时,SW-VML 中有少数函数因为算法选择和优化问题,还存在加速效率较差的情况,下一步工作:一是从算法设计方面,利用第 2.3 节介绍的优化方法,简化向量条件判断,减少类型转换等完成 SIMD 向量化调优;二是从线程绑定、多线程任务调度方面,在 OpenMP 线程级别,对函数进行调优.

References:

[1] 魏帅,赵荣彩,姚远.面向国产 CPU SW_1600 的向量重组.计算机应用与软件,2011,28(11):230−275.

[2] Vaidya AS, Shayesteh A, Woo DH, Saharoy R, Azimi M. SIMD divergence optimization through intra-warp compaction. In: Mendelson A, ed. Proc. of the 40th Annual Int'l Symp. on Computer Architecture. New York: ACM, 2013. 368−379.

[3] 张为华,朱嘉华,张宏江,臧斌宇.基于位宽控制提高 SIMD 架构并行度的优化算法.计算机学报,2009,32(11):2168−2175.

[4] Furtak T, Amaral JN, Niewiadomski R. Using SIMD registers and instructions to enable instruction-level parallelism in sorting algorithms. In: Scheideler C, ed. Proc. of the 19th Annual ACM Symp. on Parallel Algorithms and Architectures. New York: ACM, 2007. 348−357.

[5] 王迪.Simd 编译优化技术研究[硕士学位论文].杭州:浙江大学,2008.

[6] Estérie P, Gaunard M, Falcou J, Lapresté JT, Rozoy B. Boost.SIMD: Generic programming for portable SIMDization. In: DeRose L. Proc. of the 21st Int'l Conf. on Parallel Architectures and Compilation Techniques. New York: ACM, 2007. 431−432.

[7] 万木杨.大话处理器.北京:清华大学出版社,2011.

[8] Intel SSE Automation Vctorization Manual, 2001.

[9] 朱嘉华.SIMD 编译优化方法研究[博士学位论文].上海:复旦大学,2005.

[10] Talla D, John LK, Burger D. Bottlenecks in multimedia processing with SIMD style extensions and architectural enhancements. IEEE Trans. on Compuers, 2003,52(8):1015−1031.

[11] 姚远.SIMD 自动向量识别及代码调优技术研究[博士学位论文].郑州:解放军信息工程大学,2012.

[12] Liu Y, Zhang DH, Zhao XB, Mao HP, Liu XP. A rapid parellel ART based on SIMD technology. Journal of Image and Graphics, 2007,12(1):73−77.

[13] Dersch H. Universal SIMD-Mathlibrary. http://webuser.fh-furtwangen.de/%7Edersch/

[14] 姜伟华.针对实际多媒体程序和多媒体扩展指令集的 SIMD 编译优化[博士学位论文].上海:复旦大学,2005.

[15] Ålind M, Eriksson MV, Kessler CW. BlockLib: A skeleton library for cell broadband engine. In: Pankratius V, ed. Proc. of the 1st Int'l Workshop on Multicore Software Engineering. New York: ACM, 2008. 7−14.

[16] The OpenMP homepage. http://openmp.org/wp/

[17] Tsa HS. SW1600 and Alpha 21164. (2011-10-29). http://en.wikipedia.org/wiki/ShenWei

[18] Feldman M. China's indigenous supercomputing strategy bears first fruit. http://www.hpcwire.com/hpcwire/20111101

[19] BSD math collection, Cephes Math Library. http://www.netlib.org/cephes/

解庆春(1982－),男,山东巨野人,博士生,主要研究领域为高性能计算,并行计算.
E-mail: xieqingchun@126.com

吴再龙(1988－),男,硕士生,主要研究领域为异构计算.
E-mail: xiaoxian0321@gmail.com

张云泉(1973－),男,博士,研究员,主要研究领域为高性能计算.
E-mail: yunquan.cas@gmail.com

鲁永泉(1974－),男,博士,教授,主要研究领域为云计算.
E-mail: zjbc2005@163.com

李焱(1985－),男,博士,主要研究领域为并行计算.
E-mail: liyan665@gmail.com

高鹏东(1979－),男,博士,主要研究领域为云计算.
E-mail: pd_gao@cuc.edu.cn

逄仁波(1982－),男,博士生,主要研究领域为大规模并行计算.
E-mail: pangrenbo@gmail.com