SIMD技术与向量数学库研究

解庆春[1] 张云泉[1,2] 王 可[1] 李 焱[1,2,3] 许亚武[4]
(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)[1]
(中国科学院计算机科学国家重点实验室 北京 100190)[2]
(中国科学院研究生院 北京 100190)[3] (广州大学网络与现代教育技术中心 广州 510006)[4]

摘 要 首先,结合Intel,AMD和IBM处理器,介绍了单指令流多数据流(SIMD)向量化技术及其各自的特点。其次,在3种平台上对各自开发的函数库中的部分向量数学函数进行了测试。结果表明,相对传统的标量计算,向量化技术带来的加速比较高,特别是Cell SDK函数,因其独特的体系结构,多个向量处理单元带来的平均加速比为10。最后,通过测试结果的对比,发现不同数学库中的向量函数之间在性能方面也存在着差异,并对差异原因进行了分析,得出性能差异主要是处理器架构和向量计算单元个数和访存等因素造成的。

关键词 向量化,SSE,MMX,3DNow!,SIMD
中图法分类号 TP338.6 **文献标识码** A

Research of the SIMD and Vector Math Library

XIE Qing-chun[1] ZHANG Yun-quan[1,2] WANG Ke[1] LI Yan[1,2,3] XU Ya-wu[4]
(Lab. of Parallel Software and Computational Science, ISCAS, Beijing 100190, China)[1]
(State Key Lab. of Computer Science, CAS, Beijing 100190, China)[2]
(Graduate University of Chinese Academy of Sciences, Beijing 100190, China)[3]
(Network and Modern Education Technology Center, Guangzhou University, Guangzhou 510006, China)[4]

Abstract Firstly, we introduced the single Instruction Multiple Data(SIMD) vectorization technology and the features separately, based on the processors of Intel AMD and IBM Cell. Secondly, some vectorization functions were tested in these three platforms, which were deve-loped by the three vendors separately. Our test results show that we achieve high performance with the technology of the vectorization, compared to the traditional methods of the scalar calculation. Especially, the speedup of the Cell SDK functions is 10 on average, which were achieved by the help of many processing elements and the special processor structure. Lastly, we also found that there are some differences between the vectorial functions, which are in different vector math libraries. We analyzed that there are some reasons caused the difference between the math functions in different platforms, such as processor structure, the number of the processing elements, memery accessing and so on.

Keywords Vectorization, SSE, MMX, 3DNow!, SIMD

1 引言

在科学与工程计算领域,对数值计算的要求较高,如数值代数、偏微分方程数值解、最优化方法、流体力学、数字信号处理、生物数据分析、高性能商业和科学计算方法等许多领域都需要高性能的计算[1],这些大型数值应用问题一般都是高度结构化的计算问题,多数需要数千亿次浮点运算才能达到精度要求,需要大量高速向量或矩阵运算。为了解决这类大型问题,必须提高系统结构的并行性[2]。而传统CPU执行操作相当简单,其指令部件每次只对一条指令进行译码,并且只为一个操作部件分配数据。典型代表就是单指令流单数据流计算机(Single Instruction Single Data即SISD计算机)。以加法指令为例,单指令单数据的CPU对加法指令译码后,执行部件先访问内存,取得第一个操作数;之后再一次访问内存,取得第二个操作数;随后才能进行相关运算。处理器只有在完成前一条指令所规定操作后,方能进行下一条指令的运算操作。即使在处理器的核心中安上100个ALU,也不可能带来性能的提升,因为只有一个ALU是工作的,其它的ALU处于闲置状态,在等待计算结果。为了改善这一情况,单指令流多数据流(Single Instruction Multiple Data,SIMD)技术被提出[2],SIMD计算机主要适用于大量高速向量或矩阵运算的地方[3]。

到稿日期:2010-08-22 返修日期:2010-12-25 本文受国家863项目(2006AA01A125,2009AA01A129,2009AA01A134),国家自然科学基金项目(60303032),国家自然基金重点项目(60533020)资助。

解庆春(1982—),男,硕士,研究实习员,主要研究方向为高性能计算、并行计算,E-mail:xieqingchun@126.com;**张云泉**(1973—),男,研究员,博士生导师,主要研究方向为高性能计算及并行数值软件、并行计算模型、并行数据库,海量数据并行处理等;**王 可**(1980—),男,博士,副研究员,主要研究方向为并行计算;**李 焱**(1985—),男,博士生,主要研究方向为并行计算;**许亚武**(1967—),男,高级工程师,主要研究方向为计算机网络。

2 向量化 SIMD 技术

SIMD 操作数据级别的并行，就是处理器上具有能进行向量计算的硬件单元，在执行向量操作时，一条指令可以对一个向量进行运算，这类计算机称为 SIMD 计算机，如图 1 所示，在一个 128 位的寄存器下，每执行一次向量操作相当于 4 次 float 或两次 double 操作(C 语言数据类型定义)。

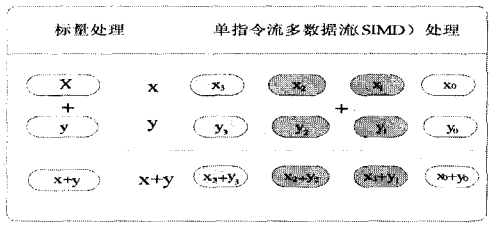


图 1 标量处理和 SIMD 之间的比较

在向量化发展的道路上，主要存在着 Intel，AMD 和 IBM 等几家公司。其中 Intel 向量化技术有 MMX，SSE，AMD 的有 3DNow!，IBM 有支持 SIMD 的 AVX 和 SPE 等。它们之间的差别主要表现在数据格式、寄存器数量、指令集数量上的不同[4,5]。

2.1 Intel SIMD 技术

1997 年 Inter 公司推出了多媒体扩展指令集(MMX)，MMX 主要用于增强 CPU 对多媒体信息的处理能力，提高 CPU 处理 3D 图形、视频和音频信息的能力。

1999 年 Inter 公司在 Pentium III CPU 产品中推出了数据流单指令序列扩展指令(SSE)。SSE 兼容 MMX 指令，它可以通过 SIMD、单时钟周期并行处理多个浮点来有效地提高浮点运算速度。从 Intel MMX 开始，对 SIMD 技术的发展历程、每个阶段新增的指令和完成的功能进行了总结，如表 1 所列。

表 1 Intel SIMD 指令集技术[9]

描述 \ 版本	MMX (1997 年)	SSE (1999 年)	SSE2 (2000 年)	SSE3 (2004 年)	SSSE3 (2006 年)	SSE4(2007 年)		AVX (预计 2011 年)
						SSE4.1	SSE4.2	
指令条数(条)	57	20	144	13	32	47	7	—
向量寄存器位数	128	128	128	128	128	128	256	
功能	主要用于增强对多媒体信息的处理能力，提高 CPU 处理 3D 图形、视频和音频信息的能力。	SSE 指令寄存器可以全速运行，保证了与浮点运算的并行性。	引入新的数据格式，如：128 位 SIMD 整数运算和 64 位双精度浮点运算等。	SSE3 又新增加了 13 条新指令。在奔腾 4 中还新增加了几条缓存指令，允许程序员控制已经缓存过的数据。	首次装载在 Core 2 Duo 处理器，SSSE3 是在 SSE3 命令的基础上又添加了 32 个新命令的产品。	SSE4 的这些指令有助于编译、媒体、字符/文本处理和程序指向加速。		将引入全新的 AVX(高级矢量扩展)指令集。

SSE4 的这些指令有助于编译、媒体、字符/文本处理和程序指向加速。其中，SSE4.1 还改良了插入、提取、寻找、离散、跨步负载及存储等动作，保证了向量运算的专一化。根据 Intel 发展战略，2011 年将要推出的“Sandy Bridge”处理器将采用名字为 AVX(Advanced Vector Extensions)的新型 SIMD 指令集，支持 256 位的指令扩展。

2.2 AMD SIMD 技术

3D Now! 指令集是 AMD 公司 1998 年开发的多媒体扩展指令集，共有 21 条指令。针对 MMX 指令集没有加强浮点处理能力的弱点，重点提高了 AMD 公司 K6 系列 CPU 对 3D 图形的处理能力。3DNow!＋指令集在原有的指令集基础上，增加到 52 条指令，其中包含了部分 SSE 指令，该指令集主要用于新型的 AMD CPU。之后在 SIMD 指令集方面，AMD 不仅为自己的处理器添加 SSE2 和 SSE3 支持，而且不断改进 3DNow!，直到 2007 年 AMD 抢在 Intel 前推出了 SSE5，共 47 条新指令，其具体发展历程与功能特点如表 2 所列。

表 2 AMD Simd 指令集技术

描述 \ 版本	3DNow! (1998 年)	3DNow!＋(1999 年)	EMXX(1999 年)	SSE5(2007 年)
指令条数(条)	21	增加到 52 条	—	170
向量寄存器位数	128	128	—	128
功能	K6-2 处理器中实现 AMD 公司开发的 SIMD 指令集，可以增强浮点和多媒体运算的速度。	AMD 为 3DNow! 增加了 5 条新的指令，用于增强其在 DSP 方面的性能，它们被称为“扩展 3DNow!”(Extended 3DNow!)。	之后开始支持 Intel 推出的 SSE 指令集。	新增的最重要的有两条：一是“三操作数指令”(3-Operand Instructions)。二是“熔合乘法累积”(Fused Multiply Accumulate，FMACxx)。SSE5 规定了加快单线程处理速度的 47 条新指令。

2.3 Cell SIMD 技术

CELL Broadband Engine(CELL BE) 处理器是由 Sony，Toshiba 和 IBM 于 2001 年开始共同研发的新一代处理器。Cell 包含了 3 个主要的单元，即：1 个作为主处理器的 Power Processor Element(Power 处理器元件，简称 PPE)、8 个单指令多数据流(SIMD)向量处理器以及 1 个高度可编程的 DMA 控制器。其中 PPE 包含 VMX(Vector Multimedia eXtension)，它是 POWER 处理器 SIMD 架构的扩展，而 VMX 本身就是一个向量处理单元。VMX 有 32 个 128 位寄存器，每个寄存器可以保存 16 个 8 位，或者 8 个 16 位或者 4 个 32 位数据[6]。

2.4 Intrinsic

Intrinsic 是处理器厂商随着编译器一块发布的，给开发环境提供的 SIMD 指令集的接口，其对 SSE，3DNow! 等指令进行了封装，这样就不用程序员自己去管理 SSE 寄存器，并可以方便地创建出大型的应用程序[2]。在编译过程中，编译程序会把 Intrinsic 接口函数转为处理器可以支持的 SIMD 指令。

3 试验与分析

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、需要大量高速向量或者矩阵计算的问题[7]。我们可以通过测试不同处理器厂商提供的向量函数，来得到 SIMD 带来的并行加速比和效率。

各厂家都针对自己的处理器开发了向量数学库，其中大多数都是基于 Cephes 库来开发的[4]，这些库基本上都实现了常用函数的向量化版本，例如 sin，cos，pow，exp，log2，log10。以下是在各自的平台上对向量数学库中常用的函数进行的测试，精度模式设置为默认的高精度模式，通过 C 语言计时函数 gettimeofday()计时，结果精确到 us，运行结果得到的测试时间为重复计算 1000 次所求的平均值。设机器的时钟周期统一为 10ns，取 cycles per element 作为性能基本单位。图 2—图 4为在 3 种测试平台下向量化方法相对标量方法带来的加速比。

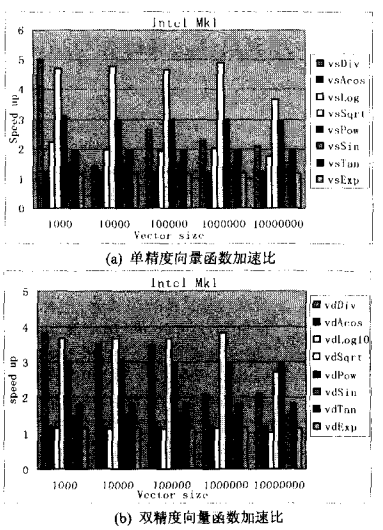
3.1 Intel 平台

表 3 为测试 Intel VML 部分向量函数的平台配置。

表 3 Intel 实验平台

处理器和存储	Xeon(R) CPU X5472 3.00GHz,2GB Memory
操作系统	Linux Ubuntu 9.04
向量数学函数库	MKL 10.1.1.019 VML
测试函数	div,acos,log10,sqrt,pow,sin,tan,exp(single precision float;double precision floats)
编译器	GCC 4.3.3

图 2 为在该测试平台下，应用 SIMD 相对于串行方法带来的加速比。



(a) 单精度向量函数加速比

(b) 双精度向量函数加速比

图 2 Intel 平台向量化带来的性能加速比

从图 2 可知，在向量规模较小时，加速比较小，当向量规模达到一定的程度时，向量化带来的加速比较大；而达到一定的规模之后，加速比趋于稳定。原因在于向量规模较小时，由于 128 位的向量寄存器需要 16 字节边界对齐，对齐操作带来的损失部分抵消了向量化带来的性能提升，因此加速比较小；而规模达到一定程度后，寄存器读写数据和字节对齐等开销相对计算时间的比值较小，向量化操作带来了较高加速比。

图 2(a)所示的几个单精度函数的平均加速比为 2.25，小于理论加速比 4；图 2(b)所示的双精度函数的平均加速比为 1.94，接近于理论加速比 2。

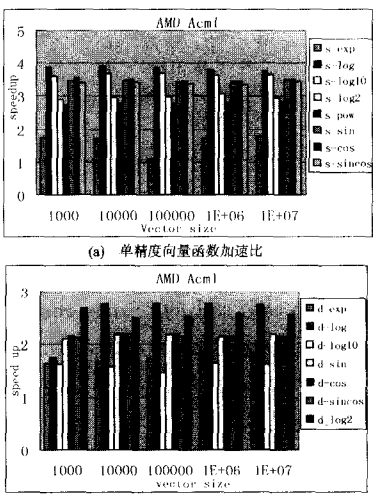
3.2 AMD 平台

表 4 为测试 AMD ACML 库中部分向量函数的试验平台配置。

表 4 AMD 实验平台

处理器和存储	AMD Opteron Processor 8378 2.0GHZ,cache size 512kB
操作系统	Debian GNU/Linux squeeze/sid
向量数学函数库	ACML 4.4.
测试函数	log2,log,log10,pow,sin,cos,sincos,exp(single precision float;double precision floats)
编译器	GCC 4.4.4

在该测试平台下向量化技术相对标量方法带来的加速比测试结果见图 3。AMD 数学库 XLMass 中支持的常用的基本向量数学函数较少，从图 3 可知，向量化函数性能加速比稳定，但同样表现出和图 2 Intel 平台测试所示的趋势，即开始加速比小，随着规模的扩大，向量化带来的效益逐渐明显。



(a) 单精度向量函数加速比

(b) 双精度向量函数加速比

图 3 AMD 平台向量化带来的性能加速比

图 3(a)所示的几个单精度函数的平均加速比为 3.15，小于理论加速比 4；图 3(b)所示的双精度函数的平均加速比为 2.0，等于理论加速比 2。可以看出 AMD 针对自己的平台对部分向量函数的优化效果明显。

3.3 Cell 平台

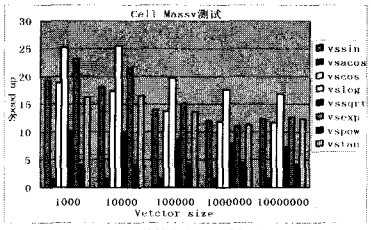
表 5 为测试 Cell SDK 中部分向量函数的试验平台配置。

表 5 Cell 实验平台

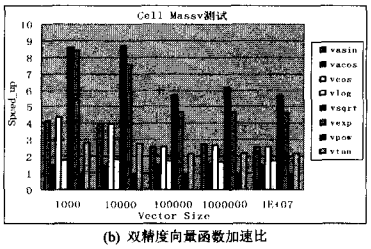
处理器和存储	Cell 3.2GHZ,2GB Memory,32/32kB L1 (data/instruction) Cache,512kB L2 Cache
操作系统	Linux Fedora Core 9.0
Cell SDK	3.1
向量数学函数库	XL Mass
测试函数	div,acos,log,sqrt,pow,sin,cos,tan,exp(single precision float;double precision floats)
编译器	GCC 4.3.0

图 4 为在该测试平台下向量化技术相对于串行方法带来的加速比。

Cell 平台因其独特的处理器架构[8]，有 PPE 的 VMX 和 8 个 SPE 等多个向量处理单元，可以一起完成大规模的向量并行处理。测试函数通过 gcc 编译，没有区分计算单元，没有针对计算单元的编译器分别编译。因为有多个计算单元，并且每个计算单元都支持 SIMD，可以说该测试平台实现了两个层次的并行，一个是指令级别的，一个是多计算单元完成的并行计算。所以从图 4 可知 Cell 处理器所调用的向量化函数加速比非常高，单精度向量函数加速比平均达到 10，而双精度函数向量化函数带来的加速比较少，平均在 2.7 左右。



(a) 单精度向量函数加速比



(b) 双精度向量函数加速比

图 4　Cell/B.E 平台向量化带来的性能加速比

从图 4(a)单精度向量函数的加速比可以看出，函数之间的差别较大，因为在 Cell 上，部分向量函数仅可以在 SPE 执行，例如 sind2，sqrtd2 等函数只能在 SPU 上执行，差距大的原因为各执行单元对函数实现调用的指令的支持程度不同。从图 4 可知，单精度和双精度的加速比差别比较大(10 与 2.7)，因为 VMX 中的 32 个向量寄存器不支持 64 位数据，原因是 IBM 认为两个 64 位数据共同操作获得的性能不足以弥补向量化带来的额外开销[10]。

3.4　对比分析

表 6 所列为数组规模为 100 万时，3 种平台下对各自数学库中的部分向量函数测得的性能值，单位为单个元素计算所需 cycle 值。

表 6　向量规模为 10^6 时不同平台测得函数计算时间

平台 / 函数	平台 1(Intel)		平台 2(AMD)		平台 3(Cell)	
	Float	Double	Float	Double	Float	Double
Sin	13.51	14.05	5.51	11.52	2.33	4.2
Pow	4.19	4.28	3.3	—	17.95	42.1
Log10	3.81	6.95	0.99	2.317	2.5(log)	17.46(log)
Exp	11.2	12.91	1.139	1.17	2.71	3.49
Sqrt	0.193	0.58	—	—	2.49	4.8
acos	11	13.17	—	—	18.29	12.9

由于各个库所支持的函数不同，从表 6 数据可以看出，Cell 在计算规模为 106 时，每个元素花的时间比较长；而 Intel 平台，只有部分 VML 向量库中的函数测得的性能较 AMD，Cell 好。从表 6 所列的几个函数性能来说，AMD 平台下的向量函数性能是最好的。

以 4 字节单精度数值运算为例，在 128 位或 256 位向量寄存器支持的前提下，每执行一次向量操作相当于 4 次或 8 次标量操作。理论可以提高 4 或 8 倍，但由于数据的 I/O、带宽、流水线和字节对齐以及 Cache 容量限制等问题，一般情况下加速比低于理论值。因为要体现 SSE 的速度，因此必须以 Stream 为前提，即大量的流数据，这样才能发挥 SIMD 的强大作用。

结束语　本文结合 Intel，AMD 和 IBM 处理器，介绍了 SIMD 向量化技术及其各自特点，并对各自平台支持的数学库进行了测试。测试发现，相对传统的标量计算，向量化带来的加速比较高，效率提升明显，特别在 Cell/B.E 平台下，因为 Cell 特殊的处理器架构，向量寄存器较多，多个计算单元并行和向量并行共同加速得到的加速比较大，比如单精度 log 函数达到的加速比高达 25.5。因为处理器厂商之间的处理器体系结构不同、SIMD 处理单元不同、指令集不同等原因，造成了他们之间在解决同一问题时的性能也存在着差异。

参 考 文 献

[1] Chiu Jihching, Chou Yuliang, Hua Yitzeng. A Multi-streaming SIMD Architecture for Multimedia Applications[A]// CF '09: Proceedings of the 6th ACM conference on Computing frontiers, 2009[C]. New York: ACM, 2009: 51-60

[2] Parhami B. SIMD Machines: DO They Have a Significant Future [J]. SIGARCH Computer Architecture News, 1995, 23(4): 19-22

[3] 郑纬民. 计算机系统结构[M]. 北京：清华大学出版社，2005：451-479

[4] Dersch H. Universal SIMD-Mathlibrary [EB/OL]. (2008-08-20). http://webuser.fh-furtwangen.de/%7Edersch/, 2010-6-30

[5] Alex Fr, Introduction to MMX Programming [EB/OL]. (2003-07-08)[2010-6-30]. http://www.codeproject.com/mmxintro.aspx

[6] 徐晟. Cell/B.E 处理器编程手册[M]. 北京：电子工业出版社，2009：10-35

[7] 刘远. 一种基于 SIMD 技术的快速并行代数重建算法[J]. 中国图像图形学报，2007，12(1)：73-77

[8] IBM Systems and Technology Group, SIMD Math Library Specification for Cell Broadband Engine Architecture[EB/OL]. https://www-01.ibm.com, 2010-6-30

[9] Furtak T, Amaral J N, Niewiadomski R. Using SIMD Registers and Instructions to Enable Instruction-Level Parallelism in Sorting Algorithms[A]// Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures. 2007 [C]. New York: ACM, 2007: 348-357

[10] AMD Corp, Core Math Library, Version 4.4.0[EB/OL]. http://developer.amd.com/cpu/libraries, 2010-6-30