ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, Supplement, December 2010, pp.214−223

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

BLAS 库在多核处理器上的性能测试与分析*

陈少虎[1,2,3+], 张云泉[1,2], 张先轶[1], 程 豪[1,2,3]

[1](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)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100190)

Performance Testing and Analysis of BLAS Libraries on Multi-Core CPUs

CHEN Shao-Hu[1,2,3+], ZHANG Yun-Quan[1,2], ZHANG Xian-Yi[1], CHENG Hao[1,2,3]

[1](Laboratory of Parallel Software and Computational Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computing Science, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: huhumartinwar@gmail.com

Chen SH, Zhang YQ, Zhang XY, Cheng H. Performance testing and analysis of BLAS libraries on multi-core CPUs. *Journal of Software*, 2010,21(Suppl.):214−223. http://www.jos.org.cn/1000-9825/10023.htm

Abstract: BLAS library is the most basic math library in high performance computing. Its performance has a great impact on the performance of supercomputers. With the multi-core technology development, BLAS' multi-core parallel performance has become more important than single-core performance associated with architecture. The experiment takes X86 multi-core processors like Xeon, Opteron series as platform for example, which are popular in HPC. It fully tests GotoBLAS, Atlas, MKL and ACML BLAS libraries of all 1,2,3-level functions, and covers different scales and multi-core parallel aspect. BLAS source code, material and papers, test results are used to analyze the way of BASL optimization and parallelism, and which platform they are suitable for. Then we will provide some useful suggestions for the use of BLAS, BLAS optimization method or even the development of high-performance CPUs. It was found that compared with a logically powerful and complex CPU, a CPU which has larger and better caches, wider memory bandwidth, smaller memory latency, higher core frequency can often obtain better performance in HPC applications. At the same time, the condition of X86 platform is also a good example for other architectures.

Key words: BLAS; architecture; multi-core parallel; X86; GotoBLAS; Atlas; MKL; ACML; optimization

摘 要: BLAS 库是高性能计算中最基本的数学库,它的性能对超级计算机的性能有着极大的影响.而且随着CPU多核化的发展,BLAS的多核并行性能已经变得比与体系结构相关的单核性能更加重要.实验以流行于高性能计算的 Xeon、Opteron 系列多核 X86 处理器为例,全面测试了 GotoBLAS、Atlas、MKL 和 ACML 四种主流

* Supported by the National Natural Science Foundation of China under Grant No.60533020 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01A125, 2009AA01A134, 2009AA01A129 (国家高技术研究发展计划(863)); the National Key Scientific & Technological Project HEGAOJI of China under Grant No.2009ZX01036-001-002 (核高基项目)

Received 2010-06-15; Accepted 2010-12-10

的 BLAS 库的所有 1,2,3 级函数,并覆盖了不同计算规模和多核并行方面的测试.通过测试结果,分析源代码、BLAS 库资料和论文的方式,分析 BLAS 有效的优化和并行方法,以及它们所适合的平台.为 BLAS 的优化、使用,甚至高性能处理器的发展上提供有益的建议.实验结果表明,比起一个逻辑处理强大但是复杂的处理器,一个 cache 更大、性能更好,内存带宽更宽、延迟更小,主频更高的处理器往往能在高性能计算中取得更好的性能.同时,X86 平台上的状况对其他体系结构也有巨大的借鉴意义.

关键词: BLAS;体系结构;多核并行;X86;GotoBLAS;Atlas;MKL;ACML;优化

基本线性代数函数库 BLAS(Basic Linear Algebra Subprograms)是最基本最核心的数学库之一,它是一组高质量的基本向量、矩阵运算子程序. BLAS 库广泛用于科学及工程计算,是许多数学软件的基本核心,因此它的性能优化对提高程序运行速度、发挥计算机的运算能力有重要的意义.现在流行的 BLAS 库主要有以下几种[1]:

ATLAS 库(Automatically Tuned Linear Algebra Software)[2],由田纳西大学计算机专家 Jack Dongarra 与一群研究员合作开发而成.Atlas 自动替指定的处理器找出有效率的方式完成 BLAS 的高效实现,可以在不同平台上自动生成优化的 BLAS 代码.

GotoBLAS 库[3],是 Texas 大学超级计算中心高性能计算组 Kazushige Goto 开发的一套高性能 BLAS 库,它针对特定的平台以实现高效的性能,支持 Opteron,Xeon,Itanium,Power,Alpha,MIPS 等平台.

MKL 库(Math Kernel Library)[4],是 Intel 为自己的 CPU 专门优化的基本数学运算库,其中包含了 BLAS 库,在 Intel 的 CPU 上性能十分高效.

ACML 库(AMD Core Math Library)[5],是 AMD 为自己的 CPU 专门优化的基本数学运算库,与 MKL 类似.

不少文章都对 BLAS 库进行过测试,但往往只针对一两个函数,只涉及到某个方面特征或少数的计算规模.一些测试会涉及不同的体系结构,但由于体系结构之间的差异太大,测试的可比性不大.这里以 x86 平台为例,进行 BLAS 三级函数全面而详细的测试与实验,找出与性能相关的因素,为 BLAS 的优化甚至高性能处理器的发展上提供有益的建议.

1 性能测试平台与环境

1.1 硬件平台

通过近几年 Top 500 发布的超级计算机榜单即可看出,x86 处理器不光在 PC 领域,在高性能计算领域也十分流行.对于高性能计算,目前用的最多的 x86 处理器就是 Intel 的 Xeon 系列和 AMD 的 Opteron 系列,实验选择的硬件平台都是这两个系列在高性能计算中最为流行的型号.

Table 1 CPU parameters concerned
表 1 实验所关注的 CPU 参数

CPU	Core speed	L1 size	L2 size	L3 size
Opteron 8378	2.4 GHz	128 KB	512 KB	6 MB
Xeon X5472	3.0 GHz	64 KB	2 * 6 MB	——
Xeon X5550	2.67 GHz	64 KB	256 KB	8 MB

高性能计算已经完全向多核化发展,单个节点上倾向于采用更多核的 CPU.由于目前四核处理器已经非常成熟,实验统统采用了两路四核平台(总共 8 个核),这也是许多超级计算机上的一个标准节点.对于 Opteron 系列,选择了 8378 处理器,Xeon 系列则选择了 X5472 和 X5550.它们的主要性能参数如表 1 所示.

从参数上看,Opteron 8378 主频最低和 L3 cache 也最小.对于 X5472,这个差距最为明显.X5472 在 cache 的层次上更少意味着大 Cache 上更快的存取速度,而且它每两个核共享一个 6M 的 L2 Cache 比 8378 四个核共享一个 6M 的 L3 Cache 优势是明显的.

Opteron 8378 的优势在于较大的 L1,L2 Cache 和四核共享的 L3 Cache.多核和 SMP 架构的主要区别就是核间是否共享 Cache,这可以增强数据重用与核间互联的性能.而 X5472 的每两个核共享 Cache 在这方面有所欠缺,X5550 作为 Xeon 的新一代处理器,在架构和关键参数上显得更为均衡.

另一个极为重要的参数是内存带宽,但带宽不光是由 CPU、内存控制器的性能决定,也受内存性能的极大影响.甚至运行情况的不同,表现出来的有效带宽也有极大的波动.理论上 X5550 带宽远大于 X5472,也比 8378 要大不少,但现实测试中往往和理论差别较大.本实验平台的实际带宽情况是:X5550 带宽大约是 X5472 的三倍,8378 的两倍.

1.2 软件平台

为了使影响测试结果的其他因素尽可能的少,实验尽可能选择相同的软件平台.

Table 2 Software versions
表 2 软件版本

Software	Linux	GCC	GotoBLAS	Atlas	MKL	ACML
Version	2.6.32	4.4.3	2-1.18	3.9.24	10.2.2.25	4.4.0

操作系统都采用 Linux,Atlas 与 GotoBLAS 的编译器都使用了 gcc.至于 MKL 与 ACML,都是以编译后的库进行分发.测试库都尽量采用了最新的版本,以提供对较新处理器的优化支持和更好的性能.

2 BLAS 测试

Atlas 与 GotoBLAS 不仅都是开源软件包,而且更是非常著名的学术项目,相关的论文比较多,可以通过源代码和文献了解它们的实现.MKL 与 ACML 作为商业软件,不仅源代码无法得到,关于其实现的文献更是极少,只能根据相关公司公布的少量材料一探究竟.但总的来说,这些 BLAS 库的实现思路都是类似的,就是将调用得最多并直接影响性能的核心函数用目标平台的汇编语言实现,再使用高级语言进行包装整合.上层使用 Pthread 或 OpenMP 并行库进行多核上的多线程并行化.不同的只是优化的方法,算法以及核心汇编与高级语言程序的比例[6,7].

例如,MKL 和 ACML 使用 OpenMP[4,5]实现多核并行,Atlas 为了实现一个可移植的高性能 BLAS 更多的使用了 C 语言,Atlas 和 GotoBLAS 的分块算法由于基于的假设不同造成算法上的差别,等等.这些实现的差别可以通过实验结果的比较进行分析.

BLAS 按操作数的精度来分有 4 种:分别是 S(单精度实数),D(双精度实数),C(单精度复数),Z(双精度复数).如果按不同精度统计来看,BLAS 1 有 46 个子函数,BLAS 2 有 66 个子函数,BLAS 3 有 30 个子函数,共 142 个子函数.

如果完全测试数据量太大,性能的主线很可能被旁枝末节所覆盖,所以实验对测试集做了一些优化和简化.就性能测试来说,许多函数是基于一个函数子集实现的,所以只需要测试这部分的核心函数,性能状况就可以把握.就精度来说,实验中只测试了在科学计算中最常用的双精度 D.

2.1 BLAS 1的测试与分析

BLAS 1 是最难以优化好的部分,这是由于它运算的数据毫无局部性可言,计算访存比又比较低,难以掩盖访存延迟.

由表 3 可以看到,在不同的测试情况中,GotoBLAS 是综合性能最好的.通过分析 GotoBLAS 的源代码得知,它的 BLAS 1 主要是由汇编语言写成的,只有两种主要的优化方法:循环展开和预取.而使用 SIMD(Single Instruction Multiple Data)指令是 x86(SSE 指令)[8]平台上 BLAS 实现的必需方法,应用在所有 BLAS 中,所以之后就不再强调了.除了这两种优化方法再无其他,所以 BLAS 1 函数往往不能很好地利用 CPU 的运算能力,测试中性能最高的为 GotoBLAS,也只能达到 CPU 峰值的(1898.78 / 10666) = 17.8%.

Table 3 Different average performances of BLAS 1 (Mflops/s)

表 3 BLAS 1 在不同测试中的平均性能 (Mflops/s)

Cores	CPU	GotoBLAS	Atlas	MKL	ACML	MAX
1-core	8378	661.12	522.42	530.3	501.58	661.12
	X5472	646.53	669.04	578.14	590.08	669.04
	X5550	1898.78	1881.75	1440.69	1193.35	1898.78
8-core	8378	652.61	481.86	385.19	399.93	652.61
	X5472	782.03	663.86	856.49	589.75	856.49
	X5550	1953.43	1237.66	1210.16	828.03	1953.43

其他的 BLAS 也使用同样的方法,只是实现的细节不同罢了.GotoBLAS 采用手工优化,自然效果最好.

除了程序优化以外,BLAS 1 的效率深受内存带宽的影响,从 X5550 的表现就可以看出.由于 X5550 处理器巨大的带宽优势,它的 BLAS 1 运算性能几乎是其他两个平台的 3 倍.此时计算速度不是瓶颈,而内存吞吐量的优势比 BLAS 的优化要有用得多.

BLAS 1 的实现值得一提的是对多核的利用,特别是 GotoBLAS 和 MKL 在 X5472 上的表现:

GotoBLAS 在 AXPY,SWAP,SCAL 三个主要函数上实现了多线程并行化.而 MKL 更多,AXPY,COPY,SWAP,ROT,SCAL,DOT 实现了并行化,如图 1 所示.这样可以通过多核更充分的利用带宽,使 BLAS 1 性能达到更高.

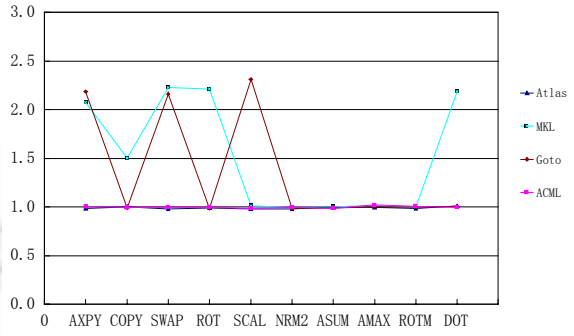

Fig.1 Speedup of BLAS 1 on X5550(8-core/1-core)

图 1 BLAS 1 在 X5550 上的加速比(8-core/1-core)

但由于 BLAS 1 对内存(Cache 利用率低)访问量很大,多线程之间协调不好易于导致未使用的预取数据发生 Cache 冲突替换.图 2、图 3 就出现了这样的情况.

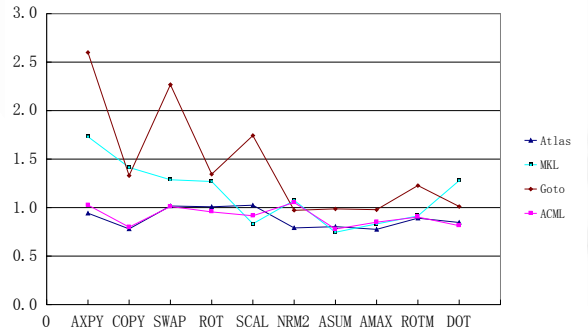

Fig.2 Speedup of BLAS 1 on X5473(8-core/1-core)

图 2 BLAS 1 在 X5473 上的加速比(8-core/1-core)

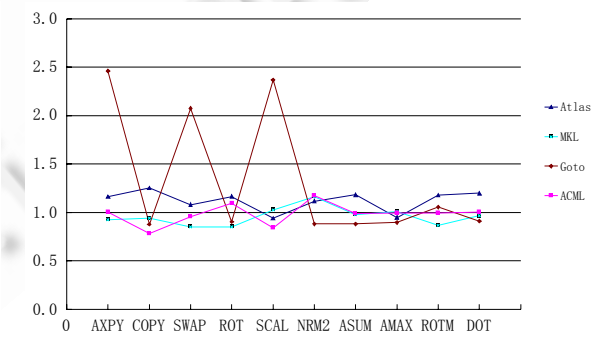

Fig.3 Speedup of BLAS 1 on 8378(8-core/1-core)

图 3 BLAS 1 在 8378 上的加速比(8-core/1-core)

可以发现除了 MKL 在 X5550 上的一小部分函数,MKL 的多线程 BLAS 1 性能上与其对应的单线程提高不大,这就是 Cache 数据冲突造成的.而 MKL 和 ACML 采用 OpenMP 的隐式并行方式更容易产生这种冲突.X5472 上没有产生这种冲突的主要原因是它没有四核共享 Cache,独立的 Cache 避免了冲突.GotoBLAS 采用 Pthread 库复杂的显式控制多线程是值得的,并行比较成功,无论在任何平台上都表现得很好.

一个精心设计的多线程并行才能使 BLAS 1 在多核 CPU 上产生性能提高,但是测试中 8 线程最多才达到 2.5 的加速比,可见瓶颈仍然在内存带宽.当达到带宽极限时再增加线程只能导致资源冲突,反而会使 BLAS 1 性能下降.如果不能很好的实现,那么 BLAS 1 只实现单线程会更好,然后将并行化的控制交给用户.

综合来说,BLAS 1 级函数对 CPU 利用率较低,应当尽量少使用,当能使用更高级 BLAS 时要优先选择更高级的函数.当需要大量使用 BLAS 1 级函数时(当然这种情况比较少见),除了选择一个好的 BLAS 实现以外,一个高内存带宽的机器是取得高性能的关键.

2.2 BLAS 2的测试与分析

BLAS2 在 O(N2)的数据上完成 O(N2)的计算,而 O(N2)次的访存中,时间主要花在内存访问上,因此内存访问方式是否有效对 BLAS 2 函数的性能影响很大.一般在向量机上 BLAS 2 才能够实现接近峰值的高性能.在通用处理器上受限于内存访问速度,性能很难优化.

BLAS 2 的函数非常多,但实际上各个 BLAS 库都是基于以下几个函数或其子集来实现的:GEMV、SYMV、TRMV、TRSV、GER.所以为了优化测试集,实验主要测试这几个函数.

Table 4 Average performance of main subprograms in BLAS 2(Mflops/s)

表 4 单核 BLAS 2 的主要函数的平均性能(Mflops/s)

CPU	BLAS	GEMV	SYMV	TRMV	TRSV	GER
8378	GotoBLAS	1000.1	2000.5	999.8	999.8	670.1
	Atlas	637.2	1000.3	615.3	615.3	615.4
	MKL	727.3	2000.5	750	727.2	500.1
	ACML	727.3	757.9	666.6	695.6	653.1
X5472	GotoBLAS	920.8	2000	1000	1000	666.8
	Atlas	699	2000	1000	800	727.3
	MKL	1000.2	2000	1000	1000	666.8
	ACML	1000.1	2000.5	1000	999.8	734.7
X5550	GotoBLAS	3200.4	7200.6	3600	3200	1896.4
	Atlas	3200.4	1960.1	2133.3	2000	1828.7
	MKL	3600.3	5000.2	3600	3600	1781.9
	ACML	3600.3	3200.4	1800	3600	1920.1

如果分析 BLAS 2 中最重要的函数 GEMV 会发现,对于未经优化的 GEMV,计算量为 $2N^2$,访存为 $3N^2$.而 GEMV 除了采用 BLAS 1 的优化方法,还使用了矩阵分块算法,使访存接近 N^2 [9].即使是这样,计算也很难掩盖访存的延迟,所以 BLAS 2 是一个典型的计算与访存密集型程序.而且测试确实表明 BLAS 2 有与 BLAS 1 相类似的一面,内存带宽强烈的影响着计算性能,X5550 单核运算性能几乎接近其他平台性能的 3 倍.X5472 与 8378 性能相当,这是由于 8378 虽然带宽更大,但是计算峰值较低的缘故.需要注意的是在小规模向量-矩阵运算时,X5550 上 BLAS 2 的性能还不及 X5472 与 8378,这是由于它的内存带宽虽然远大于其他两个平台但是访存延迟差别不大,计算规模较小而无法掩盖访存延迟所致.

一般来说,单核 BLAS 的性能高低与是否针对目标平台专门优化密切相关.通过单核 BLAS 2 综合性能图 4~图 6 可以看到,虽然同为 x86 平台上的 BLAS 库,但每个库都有自己的目标平台,每个平台上性能最好的库也可能不同.GotoBLAS 几乎对 x86 的每个主流平台都进行了优化,所以在每个平台上几乎都是性能最好的.MKL 对 Xeon 处理器专门做了优化,所以在 Xeon 系列上性能十分接近甚至超过 GotoBLAS.ACML 与 MKL 性能类似,但对 Opteron 的优化比 MKL 好,虽然优化效果不如 GotoBLAS.Atlas 永远站在移植性和高性能的平衡点上,所以比起这些手工优化过的 BLAS 库性能偏低也就不奇怪了.

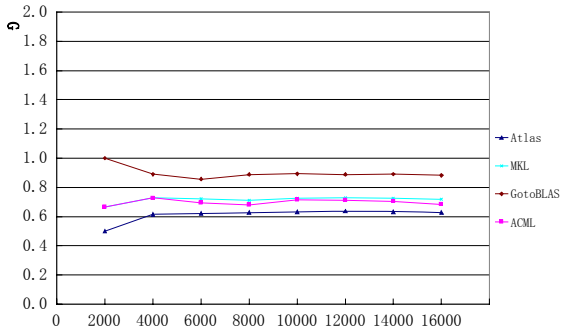

Fig.4 1-core performnce of BLAS 1 on 8378

图 4 单核 BLAS 2 在 8378 上的综合性能(Gflops/s)

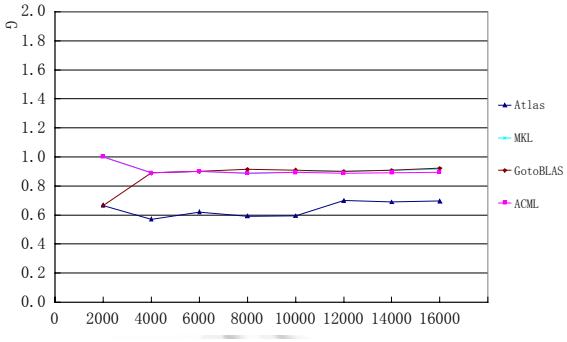

Fig.5 1-core performnce of BLAS 1 on X5472

图 5 单核 BLAS 2 在 X5472 上的综合性能(Gflops/s)

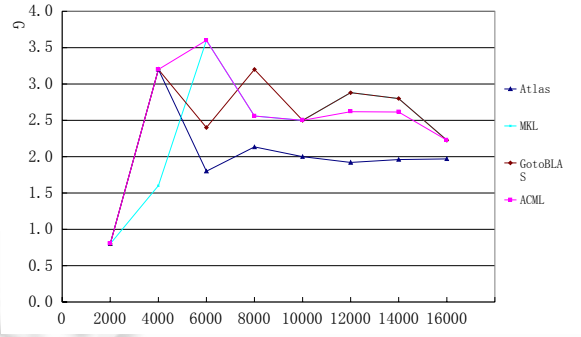

Fig.6 1-core performnce of BLAS 1 on X5550

图 6 单核 BLAS 2 在 X5550 上的综合性能(Gflops/s)

如果 BLAS 1 的多核并行还可以让用户控制的话,从 BLAS 2 开始的 BLAS 函数就不适合由用户控制并行化了.因为矩阵的分块并行方法比较复杂,由用户完成不但会加重用户的负担,往往性能也很难保证.在多核 CPU 已经成为主流的今天,一个高性能的 BLAS 必须也是高度并行化.表 5 测试了 BLAS 2 级函数在各个平台上的多核加速比.

可以发现,并不是每个 BLAS 库都实现了 2 级函数的多线程并行,Atlas 就完全没有并行的举措.GotoBLAS 并行了 5 个函数中的 4 个:GEMV、SYMV、TRMV、GER.其次是 MKL 并行了 3 个函数:GEMV,TRMV,TRSV.最后 ACML 并行了 2 个:GEMV 和 GER.

AVG 一栏统计了各个 BLAS 已并行函数的加速比均值,可以发现 GotoBLAS 并行最好,而 ACML 和 MKL 相当.但还有一些细节需要注意,在四核共享 L3 Cache 的 8378 和 X5550 中,GotoBLAS 都有很好的加速比,而 MKL 和 ACML 的隐式并行方式加速比不佳,也存在 Cache 替换冲突的问题.MKL 在 8378 处理器上没能很好地优化,甚至比不并行还慢.但是在 X5472 的分开 Cache 下,三者的加速比比较接近,可见 OpenMP 这种隐式并行难以处理 Cache 替换冲突问题.

Table 5 Software versions

表 5 BLAS 2 的多核加速比(8-core/1-core)

CPU	BLAS	GEMV	SYMV	TRMV	TRSV	GER	AVG
8378	GotoBLAS	2.79	2.88	2.93	0.85	3.14	2.94
	Atlas	0.97	0.91	1.02	0.95	0.86	—
	MKL	0.89	1.02	0.97	0.97	0.97	0.95
	ACML	2.11	0.96	0.95	0.99	1.56	1.84
X5472	GotoBLAS	1.67	2.11	2.12	1.04	2.44	2.09
	Atlas	1.19	1.04	1.08	1.46	1.08	—
	MKL	2.04	1.05	1.95	1.91	1.01	1.97
	ACML	1.82	1.02	1.02	1.05	2.46	2.14
X5550	GotoBLAS	2.85	3.16	3.22	1.08	2.13	2.84
	Atlas	0.93	0.97	0.88	0.95	0.98	—
	MKL	1.67	1.09	1.84	1.22	1.04	1.58
	ACML	1.67	1.01	1.08	0.91	1.23	1.45

BLAS 2 也存在与 BLAS 1 类似的问题:加速比受到内存带宽的制约.在内存带宽不提升的情况下,BLAS 2 的加速比不可能按 CPU 核数线性增长,在达到一定并行程度后再增加线程数无益于性能的提高.8378 在带宽上的优势弥补了计算上的劣势,在单核上性能接近于 X5472,且多核架构和数据共享上的优势更使并行 BLAS 2 的性能超过了 X5472.X5550 作为一个性能均衡的处理器,使 BLAS 2 的性能又一次达到最高.

2.3 BLAS 3的测试与分析

就函数本身的重要性而言,BLAS 3 函数是最重要的,BLAS 3 实现的好坏几乎决定了 BLAS 的好坏.BLAS 3 进行矩阵之间的基本运算,是整个 BLAS 库中被调用最多的函数.BLAS 3 的计算量达到 $O(N^3)$,访存次数为 $O(N^3)$.与二级函数不同,BLAS3 的访存可以集中于 cache 中,如果能够高效使用 cache,可以达到非常高的运算速度,甚至接近处理器性能峰值.BLAS 3 是整个 BLAS 库的核心,而 GEMM 函数则是 BLAS 3 的核心函数[10].

几乎所有 BLAS 3 级函数的实现都基于 GEMM 和 TRSM[11],特别是需要大量的调用 GEMM,因此 GEMM 函数的优化是整体 BLAS 3 级函数优化的关键,其次是 TRSM,其他函数会对它有一定量的调用.所以这两个函数的好坏影响着整个 BLAS 3 的效率.在实际应用中 BLAS 3 中的矩阵相乘运算(GEMM)时间的确占到了 BLAS 运算总时间的绝大部分.论文[12]对 Linpack 的 HPL 测试中,各个函数所运行的时间比例进行了统计.其中发现 DGEMM 函数的执行时间占到总时间的 93%以上.

Table 6 Average 1-core performance of all BLAS 3 subprograms (Mflops/s)

表 6 单核 BLAS 3 所有函数的平均性能 (Mflops/s)

CPU	BLAS	GEMM	SYMM	SYR2K	SYRK	TRMM	TRSM
8378	GotoBLAS	8 888.6	8 883.4	8 795.5	8 839.7	8 739.6	8 830.1
	Atlas	8 232.3	8 012.3	8 081.7	7 881.3	7 998.5	7 935
	MKL	8 539.1	8 273.3	7 238.5	7 185.4	7 893.9	8 120.8
	ACML	8 953.6	8 719.4	8 754	8 387.4	8 564.8	8 594.7
X5472	GotoBLAS	11 539.6	11 533.9	11 427.9	11 467.2	11 399.4	11 375
	Atlas	10 696.8	10 465.8	10 546.1	10 358.3	10 417.7	10 324.4
	MKL	11 554.2	11 408	10 357	10 834.4	11 187.1	11 200.8
	ACML	11 362.9	10 971.8	11 110.4	10 239	10 693.6	10 799.2
X5550	GotoBLAS	10 322.6	10 295.7	10 258.5	10 261.8	10 213.4	10 227.7
	Atlas	8 743.2	8 602.2	8 713.4	8 387.9	8 567.6	8 487.2
	MKL	10 317.3	10 281.8	9 823.9	10 007.3	10 160	10 032.2
	ACML	10 152.4	10 023.4	9 965.9	9 627.1	9 849.9	9 849.9

由于 BLAS 3 是 BLAS 库中使用最多的函数,每一个都比较重要,所以实验测试了所有函数,见表 6.所有的 BLAS 库都对 3 级函数做了优化,所以 BLAS 3 的计算访存比很高,性能接近峰值.它的性能受 CPU 的计算性能影响极大,X5472 上的 BLAS 3 性能往往大于 X5550,X5550 也比 8378 性能高很多.

除了处理器的峰值(与主频相关)之外,L1,L2,L3 cache 的大小也极大地影响着 BLAS 3 的性能.GotoBLAS、MKL、ACML 都立足于使用 cache 来减少对内存的访问,但它们更加在意利用 L3 cache[3]来执行分块算法以利用访存的局部性,所以 L3 cache 的大小严重影响这 3 种库的性能.通过表 7 可以发现,这 3 种库在 3 个平台上的性能是 X5550 (8M) > X5472 (6M) > 8378 (6M),X5472 比 8378 更高的性能源于 L2 cache 更低的延迟以及 Xeon 工艺和微架构.

Table 7 Ratio of 1-core peak performance of CPU

表 7 单核 BLAS 3 性能达到 CPU 峰值的百分比

BLAS	8378	X5472	X5550
GotoBLAS	92.59%	96.16%	96.83%
Atlas	85.75%	89.14%	82.02%
MKL	88.95%	96.29%	96.79%
ACML	93.27%	94.69%	95.24%

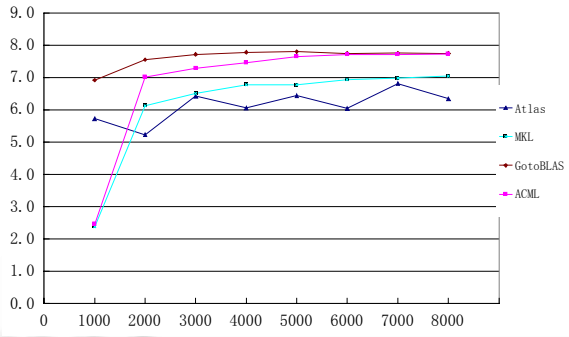

Fig.7 Speedup of BLAS 3 on 8378 (8-core/1-core)

图 7 BLAS 3 在 8378 上的加速比情况(8-core/1-core)

而 Atlas 的分块算法[13]则更倾向于利用 L1 和 L2 cache 来存储局部性数据,所以表现出来的性能是 X5472 (64KB,6M)>8378(128KB,512KB)>X5550 (64KB,256KB).

GEMM 作为最影响 BLAS 3 性能的函数,每个 BLAS 库都要仔细优化它.可以发现 GotoBLAS 因为精心的手工优化[3],几乎是所有平台上性能最好的.但 MKL 对 Xeon 的精心优化使 Xeon 平台上的 GEMM 性能超过 GotoBLAS(X5472)或接近它(X5550),ACML 在 Opteron 上也类似(8378).Atlas 由于过多的使用高级语言,导致编译器的优化能力直接影响 BLAS 3 的性能,但显然编译器还达不到 Atlas 所想象的那样强大.而且文献[12]阐明了使用最大的 cache (L3)存放局部性数据可以使矩阵分块算法更加高效.

此外,BLAS 3 多核并行的性能尤其重要.因为矩阵间的运算计算访存比很高,数据相关性较小,并行性和扩展性都极佳.只要算法得当,就可以很好的利用多核处理器.在矩阵规模较大的情况下接近线性加速.图 7~图 9 展示了 BLAS 3 不同矩阵规模时在各平台上的加速比情况,可以看到几乎所有 BLAS 库的加速比都是随矩阵规模的增加而增加的,GotoBLAS 在所有平台上最后都达到了接近线性的加速比.

Atlas 的多线程并行实现的不是很好,是性能最差的.MKL 和 ACML 基于 OpenMP 的隐式并行开销较大,在小规模矩阵计算时效率较低.而且在共享 Cache 较小的 CPU 上(Opteron 8378),容易产生 Cache 冲突替换而阻碍性能的提高.可见增大多核共享的 Cache 大小不但可以增强单线程函数的性能,也能减少多线程间的 Cache 冲突从而增强并行程序的性能.

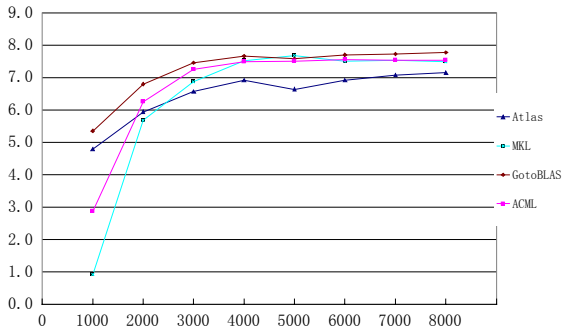

Fig.8 Speedup of BLAS 3 on X5472(8-core/1-core)

图 8 BLAS 3 在 5472 上的加速比情况(8-core/1-core)

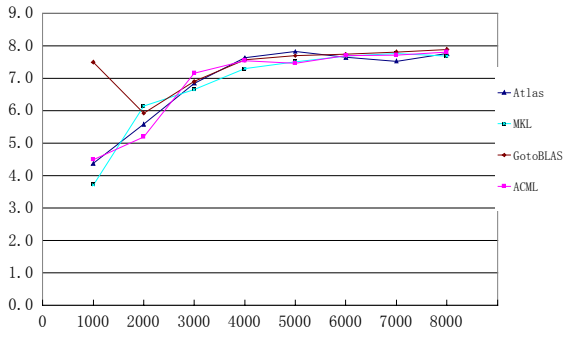

Fig.9 Speedup of BLAS 3 on X5550(8-core/1-core)

图 9 BLAS 3 在 5550 上的加速比情况(8-core/1-core)

表 8 是并行 BLAS 3 性能达到 CPU 峰值的百分比.BLAS 3 的并行化是如此重要以至于 BLAS 的提供者对其并行性能做了大量优化.ACML 达到了 8378 测试性能的最高值,MKL 在 Xeon 上的表现也与 GotoBLAS 不分伯仲.但总体来说,GotoBLAS 的性能最好.

Table 8 Ratio of 8-core peak performance of CPU

表 8 并行 BLAS 3 函数性能达到 CPU 峰值的百分比

BLAS	8378	X5472	X5550
GotoBLAS	89.77%	93.46%	95.23%
Atlas	72.88%	79.72%	77.56%
MKL	78.43%	92.09%	93.92%
ACML	90.18%	88.83%	92.97%

BLAS 3 的测试结果再次证明了 X5550 是一款设计均衡的处理器,实测性能达到了 CPU 峰值的 95.23%.X5472 的大 Cache 使 BLAS 3 无论单核还是多核的性能都很不错,但如果能够合并两个 6M 的 Cache,那么相信性能还会更高.8378 较小的 Cache 限制了处理器计算性能的发挥,而且其较低的主频也使总的性能相比另两个平台显得较低.

3 结束语

ACML 经常能达到 Opteron 测试的最高值,MKL 在 Xeon 上的表现也与 GotoBLAS 不分伯仲.但总体来说,无论是不同计算规模,不同函数之间的性能差距,还是不同平台上的性能,单核性能以及并行加速比,GotoBLAS 的表现都异常稳定,名列前茅.这样高的性能是通过大量核心函数汇编化,各平台上大量的手工调优,优良的算法[5],显式而精心的并行化而达到的.与此相对的是 Atlas,它在尽量保证移植性的情况下,仍能达到较高的性能.虽然在一些流行的、充分优化的处理器上竞争力较弱,但是对于其他不是很流行、还尚未有充分优化的 BLAS 存在的平台上,Atlas 无疑是最好的选择.

对于高性能处理器来说,摩尔定律失效后,cache 和多核的发展就成为增强 CPU 性能的主要途径.作为高性能计算中主要的函数库——BLAS 的性能严重依赖于 cache 的性能.而且在多核架构中,共享 cache 也占有非常重要的地位,是 BLAS 和其他软件能否高效并行化的重要因素.但是通过 BLAS 1,2 的测试也可以看到,访存密集型的程序在共享 cache 的多核架构往往可以达到更好的加速性能,但这是以仔细的线程控制为前提的,显式并行在理论上比隐式并行效率更高.但对于计算密集型程序,多核架构会使隐式并行的也有很好的加速比.

对于 X5472,多核 BLAS 3 的性能已经达到了峰值的 93.46%,即使提高到 100%差别也不会很大.其绝对性能相当于 2.8GHz 的同样处理器运行 BLAS 3 性能达到峰值的 100%时的状况.而往往一个复杂的逻辑处理强大的架构(X5550)是很难到达高主频的,而它的 BLAS 3 性能当然也不可能达到峰值的 100%.

所以可以得出这样一个结论:一个 cache 性能更好、更大,内存带宽更宽、延迟更小,主频更高的处理器往往能在高性能计算中取得更好的性能.

致谢 在此,我们向对本文的工作给予支持和建议的中科院软件所并行软件与计算科学实验室的的老师和同学们表示感谢.

References:

[1] Clifton C, Leavens GT, Chambers C, Millstein T. MultiJava: modular open classes and symmetric multiple dispatch for Java. ACM SIGPLAN Notices, 2000,35(10):130−145.
[2] Wegner P, Zdonik SB. Inheritance as an incremental modification mechanism or what like is and isn't like. In: Gjessing S, Nygaard K, eds. Proc. of the ECOOP'88. LNCS 322, Heidelberg: Springer-Verlag, 1988. 55–77.
[3] Zhang LB, Chi XB, Mo ZY, Li N. Introduction to Parallel Computing. Beijing: Tsinghua University Press, 2006 (in Chinese).
[4] Whaley RC, Petitet A, Dongarra JJ. Automated empirical optimizations of software and the ATLAS project. Parallel Computing, 2001,27:3−35.
[5] Goto K, van de Deijn RA. Anatomy of high-performance matrix multiplication. ACM Trans. on Mathematical Software. 2008,3(3).
[6] http://software.intel.com/zh-cn/intel-mkl/
[7] http://developer.amd.com/cpu/libraries/acml/Pages/default.aspx
[8] Jiang MQ, Zhang YC, Cong G, Li YC. Research on High Performance Implementation Mechanism of GOTOBLAS General Matrix-matrix Multiplication. Computer Engineering, 2008,34(7):84(in Chinese with English abstract).
[9] Kulkarni M, Pingali K. An experimental study of self-optimizing dense linear algebra software. Proc. of the IEEE, 2008,96(5): 1−17.
[10] Kurzak MJ, Alvaro W, Dongarra J. Optimizing matrix multiplication for a short-vector SIMD–CELL processor. Parallel Computing. 2009,35:138−150.
[11] Su B. Memory access optimization of ATLAS on Loongson 2F [MS. Thesis]. Hefei: University of Science and Technology of China, 2009 (in Chinese with English abstract).
[12] Kagstrom B, Ling P, van Loan C. GEMM-Based level 3 BLAS: High-Performance model implementations and performance evaluation benchmark. ACM Trans. on Mathematical Software, 1998,24(3):268−302.
[13] Goto K, van de Deijn RA. High-Performance implementation of the level-3. ACM Trans. on Mathematical Software, 2008,35(1):4.
[14] Yang RQ. Optimization of GotoBLAS math library based on Loongson processor [MS. Thesis]. Beijing: Institute of Computing Technology, the Chinese Academy of Sciences, 2009 (in Chinese with English abstract).
[15] Yotov K, Li XM, Ren G, Cibulskis M. A Comparison of empirical and model-driven optimization. ACM. 2003. 63–76.
[16] Goto K, van de Deijn RA. On reducing TLB misses in matrix multiplicatio. FLAME Working Note# 9.2002.

附中文参考文献

[3] 张林波,迟学斌,莫则尧,李若.并行计算导论.北京:清华大学出版社,2006.
[8] 蒋孟奇,张云泉,宋刚,李玉成.GOTOBLAS 一般矩阵乘法高效实现机制的研究.计算机工程,2008,34(7):84.
[11] 苏波.ATLAS 在龙芯 2F 上的访存优化[硕士学位论文].合肥:中国科技大学,2009.
[14] 杨荣秋.基于龙芯处理器的 GotoBLAS 数学库优化.北京:中国科学院计算技术研究所,2009.

陈少虎(1986－),男,硕士生,主要研究领域为并行软件与计算科学,多核并行,BLAS,GPGPU 并行计算.

张先轶(1983－),男,博士生,助理研究员,主要研究领域为并行数值软件研发,PGPU 软件性能优化.

张云泉(1973－),男,博士,研究员,博士生导师,主要研究领域为并行算法,并行软件.

程豪(1985－),男,硕士,主要研究领域为并行软件与计算科学,BLAS,SpMV,GPGPU 软件性能优化.