计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):101−108 [doi: 10.15888/j.cnki.csa.006045]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于申威众核处理器的 1、2 级 BLAS 函数优化研究①

孙家栋[1,2], 孙　乔[1], 邓　攀[1], 杨　超[1]

[1](中国科学院 软件研究所, 北京 100190)
[2](中国科学院大学, 北京 100049)

摘　要: BLAS(Basic Linear Algebra Subprograms) 是一个以向量和矩阵为操作对象的基础函数库. 该库中函数分为 3 个级别, 各个级别分别提供了向量-向量 (1 级)、向量-矩阵 (2 级)、矩阵-矩阵 (3 级) 之间的基本运算. 本文研究如何在申威众核处理器上 BLAS-1、2 级函数的并行实现, 并充分利用平台特性对它们进行深度的性能调优, 归纳总结程序在申威平台上的并行实现与优化技巧. 申威 26010 CPU 采用了异构众核架构, 众多计算核心提供的大规模并行处理能力, 使单块芯片具有 3 TFLOPS 的双精度浮点计算性能. 实验结果显示 BLAS-1、2 级函数相对于 GotoBLAS 参考实现版的平均加速比分别高达 11.x 和 6.x, 对于每一优化手段, 均有明显的性能加速.

关键词: BLAS; 异构众核; 任务并行; simd 向量化

引用格式: 孙家栋,孙乔,邓攀,杨超.基于申威众核处理器的 1、2 级 BLAS 函数优化研究.计算机系统应用,2017,26(11):101–108. http://www.c-s-a.org.cn/1003-3254/6045.html

Research on the Optimization of BLAS Level 1 and 2 Functions on Shenwei Many-Core Processor

SUN Jia-Dong[1,2], SUN Qiao[1], DENG Pan[1], YANG Chao[1]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: BLAS (Basic Linear Algebra Subprograms) is a specification that prescribes a set of low-level routines for performing common linear algebra operations such as vector addition, scalar multiplication, dot products, linear combinations, and matrix multiplication. The functions in this library are divided into three levels, and each level provides basic operations between vector-vector (level 1), vector-matrix (level 2), and matrix-matrix (level 3), respectively. In this paper, we study the parallel implementation of BLAS level 1 and level 2 functions on Shenwei many-core processor, and make full use of the characteristics of the platform to optimize their performance, and sum up the parallel implementation and optimization techniques of the program on Shenwei platform. Shenwei 26010 CPU uses heterogeneous multi-core architecture, and has an obvious advantage in operating speed. Many computing cores provide large-scale parallel processing capabilities, so that, double precision floating-point computing performance of one single chip can reach 3TFLOPS. The experimental results show that the average speedup of BLAS level 1 and level 2 functions is as high as 11.x and 6.x times of GotoBLAS reference implementations respectively.

Key words: BLAS; heterogeneous multi-core; task parallelism; simd vectorization

1　引言

BLAS (Basic Linear Algebra Subroutines, 基础线性代数子函数集) 是负责向量和矩阵基本运算的核心子函数库, 支持着诸多计算机应用领域的数值线性代数计算工作. 该库分为三个级别, 其中 BLAS-1 级函数负责向量-向量之间的运算, 如两个同维度向量相加、点

① 基金项目: 国家自然科学基金重大研究计划集成项目 (91530323); 国家高技术研究发展计划 (863 计划)(2015AA01A302)
收稿时间: 2017-02-21; 修改时间: 2017-03-09; 采用时间: 2017-03-13

积等; BLAS-2 级函数负责向量-矩阵之间的运算, 如向量-矩阵乘等; 而 BLAS-3 级函数支持矩阵-矩阵之间的运算, 如矩阵-矩阵乘等. 在诸如数值模拟、数值矩阵分析, 乃至当今深度学习与神经网络方面的实践表明, BLAS 库的性能对应用整体的性能有着至关重要的作用. 因此, 各大处理器厂商和研究机构均对 BLAS 函数库开展针对某一特定硬件平台的深度优化工作, 如 Intel 处理器配置的 MKL(Math Kernel Library, 数学核心库) 和 Nvidia 显卡上运行的 CuBLAS 等.

随着人们对处理器计算能力的要求越来越高[1], 对芯片的并行处理能力要求越来越大, 在当今高性能计算领域, 以传统多核 CPU 以及众核协处理构成的异构计算节点平台成为了各大超级计算机的主流配置. 申威 26010CPU 是我国自主研发的新一代众核 CPU, 具有超大规模并行计算能力. 单块 CPU 由 4 个众核 CG(Core Group, 核组) 构成, 而单个 CG 又是由单一 MPE(Manager Processing Element) 和 64 个 CPE(Computing Processing Element) 组成的共享内存异构计算系统. 其理论双精度计算性能峰值为 750 GFLOPS, 并具有超过 25.6 GB/s 的访存带宽. 本文工作开展于单众核 CG 上, 关于单众核 CG 的详细介绍将安排在第 2 节.

由于 BLAS-3 级函数有较高的计算访存比, 可以充分发挥申威众核处理器的浮点计算能力——据记载[2], BLAS-3 级函数 GEMM(一般稠密矩阵乘法) 在单 CG 上可以到达机器理论峰值的 90%(约 650 TFLOPS), 由此可见对于 BLAS-3 级函数的在申威 CPU 上的优化工作已相对成熟. 而本文关注的 BLAS-1、2 级函数首先被认为是访存密集型问题, 其性能受限于系统访带宽, 因此对这些函数的性能调优难度更大; 其次, BLAS-1、2 级函数数量较多且所解决的问题在如计算规模、数据排布等方面存在一定的灵活性, 因此需要成体系的设计一套并行实现及优化技术以成批量并有效地提高这些函数的性能; 第三在实际应用中, 虽然 BLAS-1、2 级函数单次计算量较少, 但其往往会被反复调用多次, 因此它们也容易成为应用程序的性能瓶颈. 现在, 越来越多的应用开始部署在以申威处理器为代表的国产高性能计算平台上, 包括传统的科学计算、气象预报、金融统计等领域, 以及新兴的机器学习等, 均极度依赖底层 BLAS, 因此, 在申威众核架构上并行实现并优化 BLAS-1、2 级函数有着相当的难度以及现实意义. 本文的主要贡献有以下三点:

1) 在申威众核处理器的单一 CG 上并行实现整套 BLAS-1、2 级函数, 使得它们的性能大幅高于 GotoBLAS 库中参考实现——其中 BLAS-1 级函数平均加速比约为 11 倍, BLAS-2 级函数平均加速比为 6 倍.

2) 在实现整套函数库的过程中, 我们采用了包括双缓冲、向量压缩、动态负载均衡等诸多行之有效的性能优化手段充分提高了目标程序的性能, 为今后在申威平台上程序优化工作提供了积极的借鉴意义.

3) 通过在 LAPACK 函数中调用这些高效的 BLAS-1、2 级函数, 进一步提高了 LAPACK 库中函数的运行性能, 也充分地说明了本文工作的应用价值.

本文内容安排如下: 在第 2 节中我们详细地介绍申威众核处理器的硬件、软件特性. 在第 3 节中我们将逐步介绍 BLAS-1、2 级函数的并行实现以及使用的各种优化手段; 在第 4 节中我们将展开性能实验, 展现本文工作的有效性能和实用性; 本文将在第 5 节中进行讨论并总结.

2 申威 CG 介绍

2.1 体系结构特性

本文围绕申威众核处理器的单一 CG 展开工作. 一个 CG 由一个控制核心 MPE 与 64 个轻量级计算核心组成. MPE 结构类似通用单核 CPU, 其主频为 1.5 GHz. 它主要负责程序中逻辑复杂的串行部分.64 个 CPE 被划分成 8*8 二维阵列, 如图 1 所示. 每一个 CPE 主频仅为 1.5 GHz, 但 64 个 CPE 协同能够对程序中计算密集的部分提供大规模并行处理能力. 概括地说, 每一个 CPE 由轻量级逻辑处理单元、向量化计算单元、LDM(Local Data Memory, 局部数据存储器) 构成. 其中, 逻辑处理单元的能力相对通用 CPU 而言较弱, 因此 CPE 不适合处理逻辑结构复杂的程序片段. 向量化计算单元宽度为 512-bit, 支持常用的单、双精度向量运算符和向量乘加融合操作. 因此从理论上说, 一个 CPE 能够提供 4*1.5GHz*2=12GFLOPS 双精度浮点计算能力. 与通用 CPU 不同的是, CPE 私有的 64-KB LDM 作为高速数据缓存可被程序员控制, 因此程序设计需要考虑显式地考虑如何高效地复用 LDM 中的数据. MPE 访存方式与通用 CPU 类似, CPE 阵列可与 MPE 共享进程的内存空间. 每一个 CPE 除了读写内存的常规方式外, 系统还提供专门的 DMA(Data Memory

Access, 直接内存访问) 通道为 CPE 与内存高速地传输成块的数据.64 个 CPE 并发 DMA 操作可以在理论上提供 32GB/s 的访存带宽. 多个并发 DMA 请求受到系统的 DMA Controller 管理, 确保每一个请求被正确地应答. 在图 1 中我们还看到, 每相邻两行共计 16 个 CPE 共享一条数据子总线, 其作为 CPE 与内存数据传输的通路. 因此并发 DMA 请求的处理还受到总线仲裁协议的制约. 但由于 CPE 常规内存操作的带宽较低, 因此 CPE 程序应尽可能使用 DMA 方式访问数据. 另外值得一提的是同一行 (列) 上的 8 个 CPE 共享一条控制总线, 能够支持同行 (列) 上的 CPE 同步栏栅 (Synchronization Barrier) 操作.

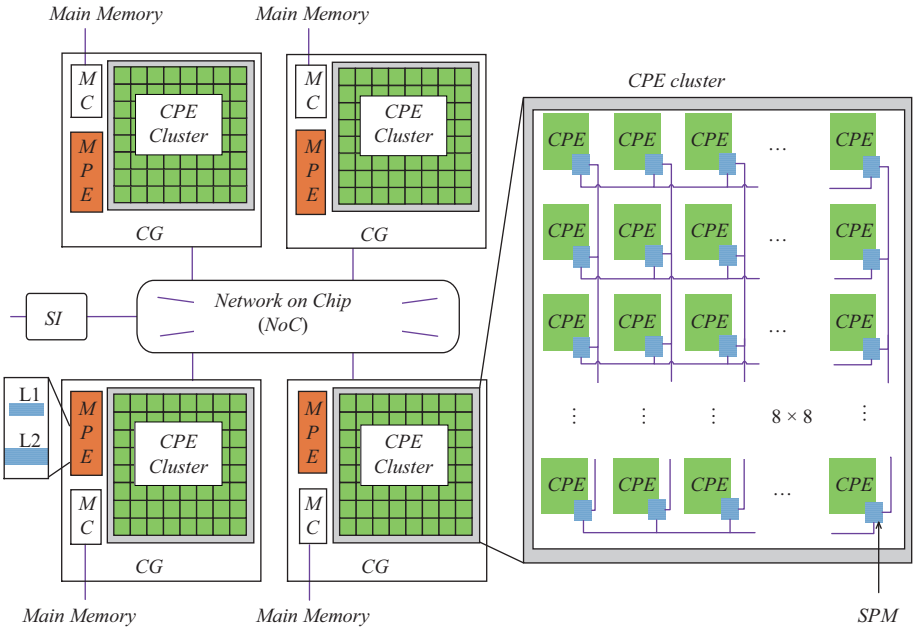


图 1 申威 CPU 总体架构

2.2 基础编程接口

由于申威处理器属于新兴的异构众核平台, 开发者实现了基于此平台的用于并行编程的接口, 即控制 MPE、CPE 阵列协同计算的编程接口, 名为 athread. 在 athread 中, 程序员可以手动的指定并行区域所使用的线程数量, 并通过线程启动接口开启线程并行处理. 此外, CPE 也通过 athread 库中提供的接口使用 DMA 机制. athread 库的各常用 API 由表 1 列出.

表 1 athread 库主要接口函数汇总.

函数名称	说明
athread_init()	初始化CPE阵列
athread_set_num_threads()	设置下一并行区线程数量
athread_spawn()	MPE开启CPE线程执行并行区
athread_join()	MPE等待并行区结束
athread_halt()	关闭CPE阵列
athread_get()	CPE使用DMA从内存读取数据
athread_put()	CPE使用DMA向内存写入数据

一个基于由 athread 接口编写的并行程序的执行流程大致如图 2 所示, 其中 MPE 在初始化 CPE 阵列之后, 设置线程数并由 CPE 阵列执行并行区, 最后等待并行区执行结束. 常规地, 类似图 2 的并行方式叫做 Fork-and-Join. 另外 CPE 还支持若干内存的原子操作, 如“取并加 1”和“比较并交换”, 以这些原子操作为基础可以为若干 CPE 并行计算提供更灵活的同步机制.

3 并行实现与性能优化

3.1 BLAS-1、2 级函数特点

BLAS-1 级函数处理标量-向量、向量-向量运算. 基于 BLAS-1 级各个函数的相似性, 我们仅以函数 daxpy(双精度向量乘加) 为例来说明在 BLAS-1 级函数中常见的数据结构, 其接口如公式 (1) 所示, 它的功能则实现了公式 (2):

$$\text{daxpy}(\text{int n, double alpha, double} * \text{x, int incx, double} * \text{y, int incy}) \quad (1)$$

$$\mathbf{y} = \text{alpha} * \mathbf{x} + \mathbf{y}, \quad (2)$$

其中 alpha 为标量, **x**, **y** 为 n-维向量.

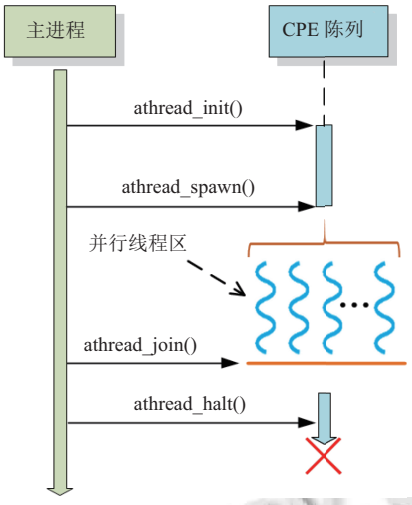


图 2 athread 的 Fork-and-Join 并行模式

BLAS-2 级函数处理向量-矩阵间的计算. 与 BLAS-1 级类似, 我们以 BLAS-2 级函数中具有代表性的方法 dgemv(双精度一般矩阵向量乘) 来说明在 BLAS-2 级函数中常见的数据结构, 其功能实现了公式 (3):

$$\mathbf{y} = \text{alpha} * \mathbf{A} * \mathbf{x} + \text{beta} * \mathbf{y} \quad (3)$$

为了方便讨论并不失一般性, 此处讨论矩阵 A 非转置访问的情况.

从本质上说, BLAS-1、2 级函数仅仅涉及到一维、二维线性数据结构的遍历操作. 但对于 BLAS-1 级函数而言, 由于其计算访存比低并在实际应用中输入规模一般较小而调用次数多, 因此对于 BLAS-1 级函数我们在对其并行化的过程需要尽可能地提高内存带宽利用率和降低函数调用开销. 而对于 BLAS-2 级函数来说, 除以上问题之外更需要其他优化技术, 以处理数据复用及由矩阵的特殊性质而产生的负载不均衡. 在接下来的章节中, 我们将逐一介绍 BLAS-1、2 级函数在申威众核处理器上的并行化和性能调优手段.

函数的代码架构设计如图 3 所示.

3.2 数据划分与并行化

针对 BLAS 库所处理的对象存储特点, 我们采用“段”(Segment) 结构来表示向量的一个分段, 用“片”(Tile) 结构来表示矩阵的一个子块, 如图 4 所示. 在实际的 (并行) 计算开始前, 我们使用 MPE 进行数据划分并在合适时将各个段、片上的计算任务指派到对应的处理核心——任务调度. 虽然使用 MPE 串行地完成以上操作在提高函数性能方面乏善可陈, 但这样做的好处在于, 可以尽早地得到问题分解的详细信息并决定: 1) 在当前问题规模情况下, 是否需要开启 CPE 阵列或开启多少线程进行并行处理; 2) 在开启 CPE 阵列之后, 是否需要使用动态负载均衡策略.

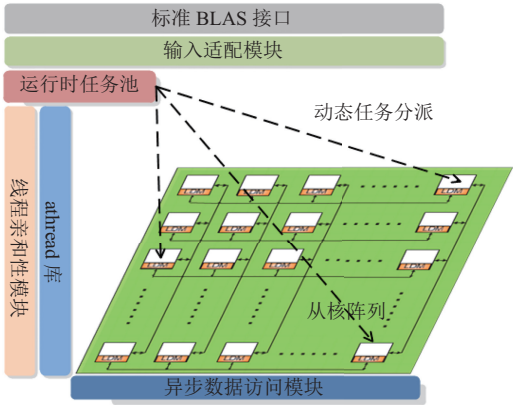


图 3 BLAS-1、2 级函数基本架构

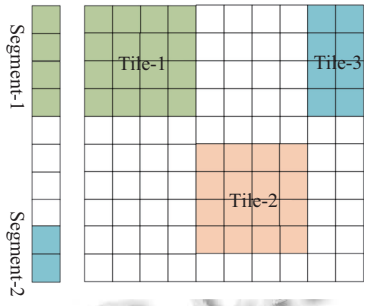


图 4 Tile 和 Segment 示意图

对于 BLAS-1 级函数, 当输入向量被划分成若干等长段后 MPE 仅决定使用多少 CPE 进行并行处理. 为此, 本文专门实现了线程数自动设置功能, 为各个规模的输入设置恰当的线程数量, 尽可能地减少线程使用的开销. 由于 BLAS-1 级函数问题相对规整, 可以在并行区开始前让 MPE 执行静态任务调度, 即可保证负载的平衡.

对于 BLAS-2 级函数, 首先以矩阵为中心进行数据划分, 将矩阵分为 M*N 块. 依据计算规模设置 CPE 数量后, MPE 将选择使用静态或动态的任务调度策略: 在矩阵为一般稠密矩阵的前提下, 依然可以使用任务的静态分派策略以保证负载均衡; 如果矩阵为三角矩阵或对称矩阵, 简单任务静态分派策略将很难保证 CPE 阵列的负载均衡性, 因此本文提出基于“动态工作共享”(dynamic work-sharing) 的任务动态调度. 关于该调度技术的实现, 将在后面的章节进行讨论.

3.3 向量压缩 (compact)

如上文所述, BLAS 库支持非连续储存的向量的运算, 而在申威平台上向量相邻元素间的跨步将会严重影响 DMA 访存带宽. 对于 BLAS-1 级函数, 由于仅涉及对向量的遍历操作, 故向量的非连续性带来的性能下降无法避免. 而对 BLAS-2 级函数, 由于相关向量将被反复访问, 因此有必要采取向量压缩手段对输入向量进行预处理.

在图 5 中, 带跨步的向量 **x** 在经过压缩之后写回新的内存区域 $\mathbf{x}_1$. 在接下来的矩阵-向量乘过程中, 对向量 **x** 的访问被重定向到 $\mathbf{x}_1$, 因此避免了向量跨步对计算流程的反复影响. 在第 4 节的实验部分, 可以看到向量压缩这一手段有效地提高了 BLAS-2 级函数的执行效率.

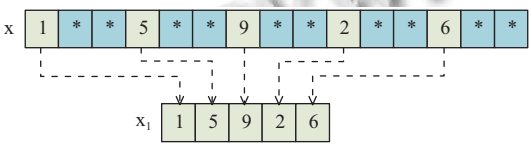


图 5 向量压缩示意图

3.4 LDM 双缓冲技术

由于各个 CPE 中有程序员可控制的 LDM 并且 DMA 为异步通信, 因此在一定程度上可以使 CPE 的计算和其 DMA 访存过程相重叠. 本文对 BLAS-1、2 级函数采用了分段和分片处理, 因此在 CPE 处理当前段或片的同时, 可以加载下一段或片的数据进入 LDM, 其过程如图 6 所示. 这样至少在语义上实现了程序的计算通信重叠, 但实际对程序性能带来的增益还要考察程序本身的计算访存比. 为了方便地为各个函数确定双缓冲机制是否会带来显著的性能增益, 本文使用 C 语言宏定义抽象了 CPE 双缓冲逻辑并支持各种数据类型.

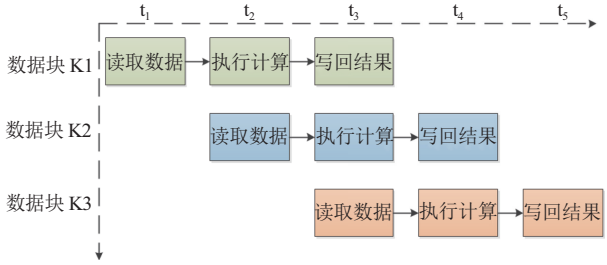


图 6 计算访存重叠

3.5 任务动态调度

在任务量不均匀的情况下, 本文采用任务动态调度机制以确保程序的负载均衡. 图 7 中展示了一个上三角矩阵 A 与向量 **x** 相乘并更新向量 **y** 这一过程 (trmv). 首先将矩阵 **A** 进行分片, 形成多个正方形小片, 相应于若干小片形成的一行, 将向量 **y** 分段. 为了避免多个线程在更新向量 **y** 产生竞写, 本文设计一个 CPE 一次性完全更新 **y** 的一个分段.

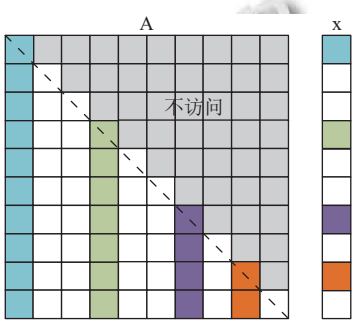


图 7 三角矩阵向量乘中的负载不均情况

可以看到, 在如图 8 所示的这种情况下, 更新各个 **y** 分段的计算量并不一致. 为了简易地保证各个 CPE 之间的负载均衡, 本文专门开发了支持动态任务共享的任务池. 以任务池为基础, 每个 CPE 在完成当前向量段的更新之后, 能够及时地在任务池中获取下一个向量段更新任务, 避免 CPE 空闲.

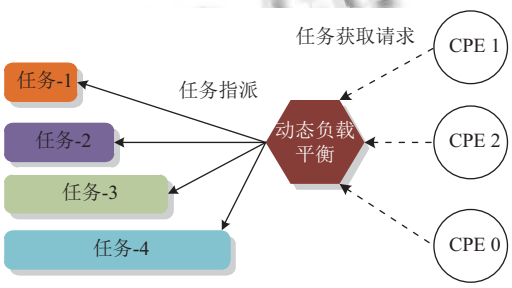


图 8 动态负载均衡组件及示意

3.6 CPE 阵列数据子总线利用 (optional)

在实际应用中, 由于 BLAS-1、2 级处理的数据规模相对较少, 在一般情况下并不需要开启 64 个 CPE 进行计算, 否则并行线程将对系统共享资源 (如 DMA 控制器、数据总线等) 产生过强的争用, 影响程序性能. 第 2 节提到整个 CPE 阵列设有 4 套数据子总线, 每两行共计 16 个 CPE 共享一套数据子总线. 而在默认情况下, athread_spawn() 函数以行优先的方式启动 CPE, 不利于充分利用 4 套数据子总线. 因此本文基

于 athread 库的接口, 开发设置线程亲和性的函数, 将开启的线程更为均匀的分配在各套数据子总线上, 这种方法得益于申威众核处理器的物理结构, 是该平台特有的优化手段.

4 实验与讨论

本文关注的 BLAS-1、2 级函数首先被认为是访存密集型问题, 其性能受限于系统访带宽, 因此对这些函数的性能调优难度更大; 其次, BLAS-1、2 级函数数量较多且所解决的问题在如计算规模、数据排布等方面存在一定的灵活性, 因此需要成体系的设计一套并行实现及优化技术, 以成批量并有效地提高这些函数的性能; 第三, 在实际应用中, 虽然 BLAS-1、2 级函数单次计算量较少, 但其往往会被反复调用多次, 因此它们也容易成为应用程序的性能瓶颈.

经过以上对优化手段的描述, 借助神威太湖之光异构众核平台, 本文进行了一系列的对比实验. 特别的, 针对 BLAS1 级函数, 本文进行了访存带宽的测试, 设计了访问主存数据的对比实验, 并以检测到的带宽作为对比基准, 并对 iamax 等几个 1 级函数进行实验, 结果表明一级函数在访存方面已经达到足够优秀的性能.

如图 9 及图 10, 对 gemv 函数的单精度和双精度, 本文分别实现了使用了双缓冲技术和未使用双缓冲的两个版本, 矩阵大小规模范围在 300~1200, 可以看到优化手段使得计算性能得到了一定的加速.

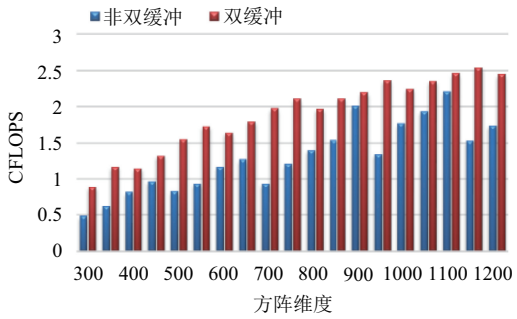


图 9 双缓冲优化——dgemv 性能对比图

对于向量压缩存储优化方案, 本文以 sgemv 函数为研究对象, 对比是否进行向量压缩两种情况. 矩阵大小规模在 600~3000, 而对两个参与计算的向量 x, y 的存储增量也有几种不同的情况, 分别设置如下: (a) incx=11, incy=15; (b) incx=13, incy=15; (c) incx=11, incy=17; (d) incx=13, incy=17.

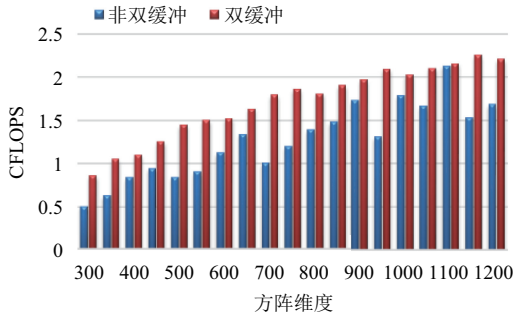


图 10 双缓冲优化——dgemv 性能对比图

对比实验结果如下图所示, 可以得出结论, 进行向量压缩优化手段之后, 计算性能得到了 1~2 倍的加速.

本文以 trmv 为实验载体, 进行了负载均衡的优化对比实验, 三角矩阵规模范围为 100~3900. 如图 12 及图 13, 可以看到, 相比于数据静态分配, 优化方案在性能上有 1.2~1.6 倍的加速.

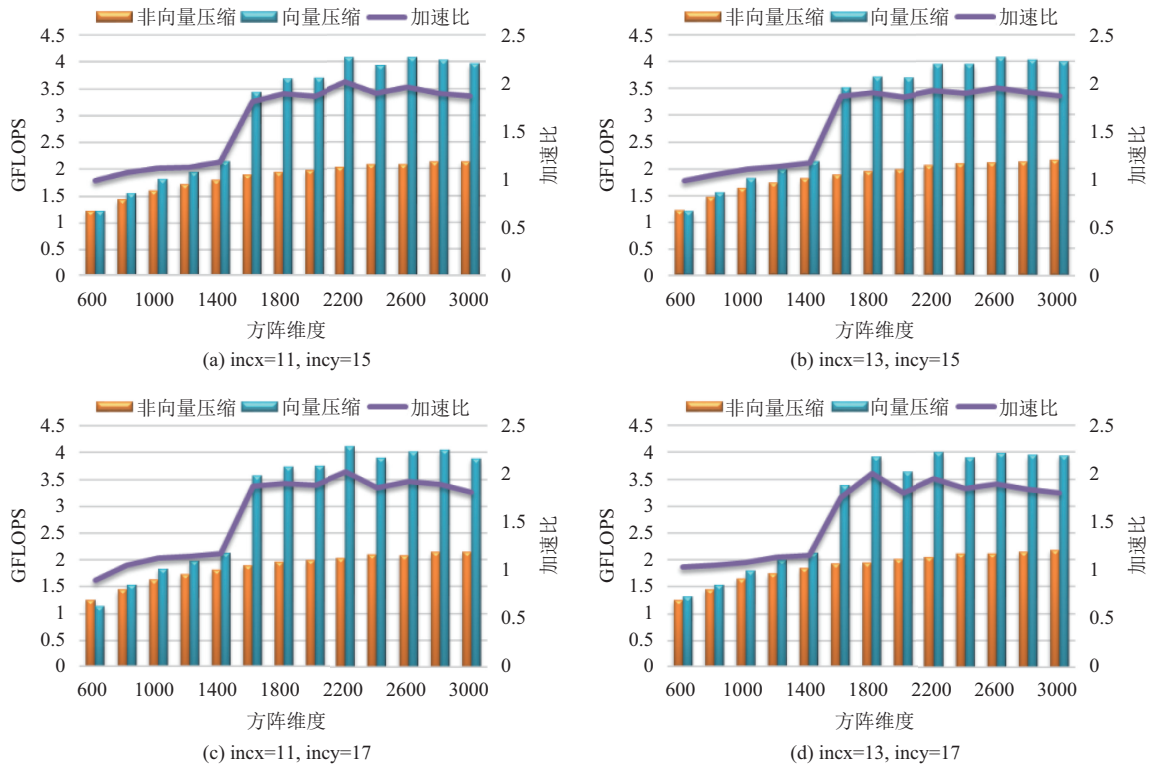
在应用数据复用手段的时候, 要注意矩阵的规模, 当矩阵估摸足够小, 数据复用的优势就不明显了, 因为此时矩阵以及向量的分块比较少, 重复存取的时间消耗不算特别大. 当规模增加到 4096 及以上时, 数据复用优化手段的优势开始显现, 如下图, 性能加速会达到 1.2 倍.

5 相关应用

BLAS-2 级函数在矩阵分解方面被广泛调用, 比如 LU 分解, QR 分解以及 SVD 分解等等, 均大量调用了 gemv 等子函数.

线性代数中, LU 分解 (LU Decomposition) 可以将一个矩阵分解为一个下三角矩阵和一个上三角矩阵的乘积 (有时是它们和一个置换矩阵的乘积). LU 分解主要应用在数值分析中, 用来解线性方程、求反矩阵或计算行列式. LU 分解在本质上是高斯消元法的一种表达形式, 实质上是将 A 通过初等行变换变成一个上三角矩阵, 其变换矩阵就是一个单位下三角矩阵. 基本思想: 从下至上地对矩阵 A 做初等行变换, 将对角线左下方的元素变成零, 然后再证明这些行变换的效果等同于左乘一系列单位下三角矩阵, 这一系列单位下三角矩阵的乘积的逆就是 L 矩阵, 它也是一个单位下三角矩阵.

QR 分解是将矩阵分解成一个正规正交矩阵 Q 与上三角形矩阵 R, 它是目前求一般矩阵全部特征值的最有效并广泛应用的方法, 一般矩阵先经过正交相似变化成为 Hessenberg 矩阵, 然后再应用 QR 方法求特征值和特征向量.



(a) incx=11, incy=15 (b) incx=13, incy=15

(c) incx=11, incy=17 (d) incx=13, incy=17

图 11 向量压缩——sgemv 性能对比图

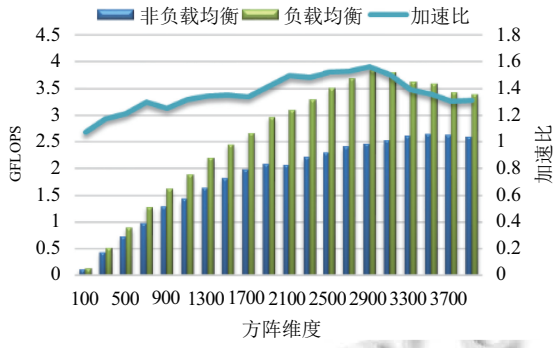


图 12 负载均衡—dtrmv 性能对比图

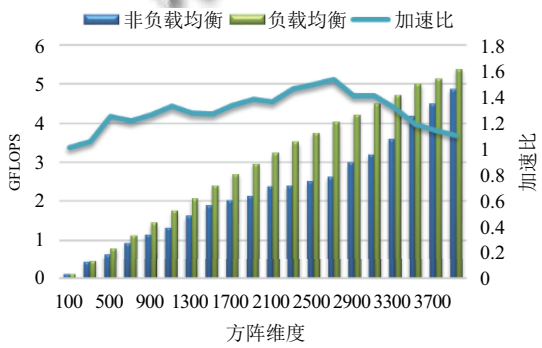


图 13 负载均衡——strmv 性能对比图

以上述两种分解为实验对象, 分别调用本文提出改进后的 BLAS 函数和原版 GotoBLAS, 四类精度的性能对比数据如表 2 及表 3.

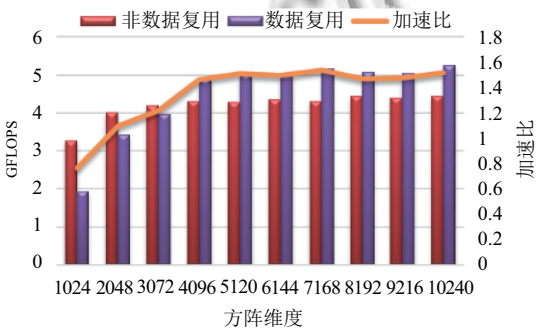


图 14 数据复用——dsymv 性能对比图

表 2 LU 分解性能对比 (GFLOPS)

数据类型	改进版	原版
实数单精	49.54	22.98
实数双精	54.67	19.46
复数单精	68.09	40.09
复数双精	57.46	28.22

6 结论

本文在新兴申威众核处理器 26010 设计实现了 BLAS-1、2 级函数并研究了其性能优化方法, 在该平

台上 BLAS-1、2 级函数的并行化和性能调优有相当的难度和现实意义. 本文首先通过在 CPE 上设计一套适应于 DMA 机制的矩阵和向量的逻辑表达方式, 即矩阵片、向量段, 实现了一系列目标函数的 CPE 并行化方法. 在此之上, 使用了诸如向量压缩、DMA 双缓存、任务动态调度及数据子总线亲和等方法, 大幅地提高了目标函数的性能. 实验证实, 本文实现的算法相对于 MPE 上参考版本在性能有着大幅的提高, 其中: BLAS-1 级函数的加速比高于 11 倍, 而 BLAS-2 级函数的加速比也高于 6 倍, 各个分项的优化手段所带来的性能增益也逐一被实验所验证. 其中, 对在物理存储上不连续的向量, 通过使用向量压缩机制能够避免数据的非连续性对程序访存行为的影响; 数据双缓存机制有能力覆盖一部分计算所耗费的时间. 而动态任务调度、数据子总线亲和性机制能够在数据分布非规整的情况下更充分地利用 CPE 阵列的计算部件、数据通路. 最后, 在确保相同的实验条件下, LAPACK 中典型的矩阵分块函数 dgelqf、dgeqrf 通过调用本文设计实现的 BLAS-1、2 级函数, 性能提高了数倍, 充分说明了本文工作的实用性.

表 3 QR 分解性能对比 (GFLOPS)

数据类型	改进版	原版
实数单精	103.11	33.78
实数双精	120.07	27.87
复数单精	192.46	69.8
复数双精	174.69	35.51

7 相关工作

本文主要探讨 BLAS-1、2 级函数在申威 26010 众核架构上的并行实现与性能优化, 在此过程中, 本文借鉴了以往诸多研究机构及处理器厂商在主流计算机平台上对 BLAS 库开展的性能调优经验. 作为 BLAS 优化工作基石的 GotoBLAS 是由德克萨斯高级计算中心开发的 BLAS 实现库, 它从平台高速缓存组织方式和处理器流水线结构等角度出发, 使用数据分块技术并重排汇编指令, 提高了 BLAS 库在通用多核 CPU 上的执行效率. 对向量和矩阵分别进行分段和分片的设计思路受到了 GotoBLAS 数据分块的启发, 但在申威平台上, 数据的合理划分目的是为了使 DMA 以更高的带宽传输成块的数据, 而非提高数据在高速缓存中的复用率. 并且在申威平台上 BLAS-1、2 级函数的性能主要受限于 DMA 带宽, 因此针对 CPE 的指令优化技巧没有产生显著的性能增益. 在多核、众核处理器上对 BLAS 进行优化的工作还有 Intel MKL 和 AMD ACML, 分别为 Intel 和 AMD 公司为其旗下处理器定制的高性能数学库, 这些商业版数学库软件对硬件的处理能力的挖掘更为彻底. 这些工作也激发我们通过改变线程数量、亲和性等手段降低线程争用开销、充分各套数据子总线. 在运行参数自动调优方面, 本文受到 ATLAS 库的启发. ATLAS 库是一款能够自动探查硬件特性并产生高性能代码的开源 BLAS 库. 类似地, 本文根据输入问题规模, 根据以往运行的结构自动地选取合适的线程数和分块大小, 以确保 BLAS 库在一般的使用场景中能够产生相对较优的性能.

另外, Nvidia 公司为其众核 GPGPU 也配置了高性能数学库 cuBLAS, 我们也在新型众核处理器申威平台上配置了 BLAS-1、2 级函数, 对今后的实践提供了积极的借鉴. 最后, 国内外有很多学者开展了针对 BLAS 1、2 级中某些具体函数的优化工作, 如 Jian Yin[3], Weizhi Xu[4,5], Rajib Nath[6], Li Yi[7], Qian Wang[8]等. 这些的工作共同特点是仅优化程序的计算性能. 但如上文所述在当前申威平台上, BLAS-1、2 级函数的计算效率并不会成为影响整个程序性能的瓶颈, 所以应该把关注点集中在提高降低程序调用开销、提高函数的访存带宽、确保 CPE 负载均衡等方面.

参考文献

1 刘颖, 吕方, 王蕾, 等. 异构并行编程模型研究与进展. 软件学报, 2014, 25(7): 1459–1475. [doi: 10.13328/j.cnki.jos.004608]

2 梁娟娟. 基于 GPU 的 BLAS 库的设计和实现[硕士学位论文]. 北京: 中国科学技术大学, 2010.

3 Yin J, Yu H, Xu WZ, *et al*. Highly parallel GEMV with register blocking method on GPU architecture. Journal of Visual Communication and Image Representation, 2014, 25(7): 1566–1573. [doi: 10.1016/j.jvcir.2014.06.002]

4 Xu WZ, Liu ZY, Wu J, *et al*. Auto-tuning GEMV on many-core GPU. Proc. of the 18th International Conference on Parallel and Distributed Systems (ICPADS). Singapore. 2012. 30–36.

5 Xu WZ, Zhang H, Jiao S, *et al*. Optimizing sparse matrix vector multiplication using cache blocking method on fermi GPU. Proc. of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD). Kyoto, Japan. 2012. 231–235.

6 Nath R, Tomov S, Dong TX, *et al*. Optimizing symmetric dense matrix-vector multiplication. Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. Seatle, WA, USA. 2011. 1–10.

7 李毅, 何颂颂, 李恺. 多核龙芯 3A 上二级 BLAS 库的优化. 计算机系统应用, 2011, 20(1): 163–167.

8 王茜. 多核 CPU 上稠密线性代数函数优化及自动代码生成研究[博士学位论文]. 北京: 中国科学院大学, 2015.