软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(Suppl.(2)):118−126

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

异构平台数学库 MAGMA 性能测试与分析*

肖玄基[1,3], 张云泉[1,2], 李玉成[1,2], 袁 良[1,3]

[1](中国科学院 软件研究所 并行计算实验室,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](中国科学院大学,北京 100190)

通讯作者: 肖玄基, E-mail: growj@126.com

摘 要: MAGMA是第一个面向下一代体系架构(多核CPU和GPU)开源的线性代数软件包,它采用了诸多针对异构平台的优化方法,包括混合同步、通信避免和动态任务调度.它在功能、数据存储、接口上与 LAPACK 相似,可以发挥 GPU 的巨大计算能力进行数值计算.对 MAGMA 进行了测试分析.首先对矩阵分解算法进行分析;然后通过测试结果,分析 MAGMA 有效的优化和并行方法,为 MAGMA 使用、优化提供有益的建议;最后提出了一种对于矩阵分块算法的自适应调优的方法,经过测试,对于方阵的 SGEQRF 函数加速比达到 1.09,对于高瘦矩阵的 CGEQRF 函数加速比达到 1.8.

关键词: MAGMA;GPU;矩阵分解;优化;线性代数软件包

中文引用格式: 肖玄基,张云泉,李玉成,袁良.异构平台数学库 MAGMA 性能测试与分析.软件学报,2013,24(Suppl.(2)):118−126. http://www.jos.org.cn/1000-9825/13030.htm

英文引用格式: Xiao XJ, Zhang YQ, Li YC, Yuan L. Performance testing and analysis of MAGMA library on hybrid architecture (CPU+GPU). Ruan Jian Xue Bao/Journal of Software, 2013,24(Suppl.(2)):118−126 (in Chinese). http://www.jos.org.cn/1000-9825/13030.htm

Performance Testing and Analysis of MAGMA Library on Hybrid Architecture (CPU+GPU)

XIAO Xuan-Ji[1,3], ZHANG Yun-Quan[1,2], LI Yu-Cheng[1,2], YUAN Liang[1,3]

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

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

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

Corresponding author: XIAO Xuan-Ji, E-mail: growj@126.com

Abstract: MAGMA is an open source high performance linear algebra package first developed for next-generation of heterogeneous/hybrid architectures (CPUs+GPUs) with a dense linear algebra library similar to LAPACK in functionality, data storage, and interface. This paper presents performance testing and analysis of MAGMA. It first studies the matrix decomposition algorithm in MAGMA, then provides some useful suggestions of MAGMA usage and optimization through massive testing and source code analysis, and finally proposes a method for auto-tuning matrix decomposition block algorithms. In this test, the speedup is 1.09 for SGEQRF of square matrix and 1.8 for CGEQRF in terms of tall and skin matrix.

Key words: MAGMA; GPU; matrix decomposition; optimization; linear algebra package

高性能计算(high performance computing)是计算机科学的一个分支,随着信息化社会的飞速发展,人类对信息处理能力的要求越来越高,不仅石油勘探、气象预报、航天国防、科学研究等需要高性能计算,而且金融、

* 基金项目: 国家自然科学基金(61133005); 国家高技术研究发展计划(863)(2012AA010902, 2012AA010903); 国家重大专项核高基项目(2009ZX01036-001-002)

收稿时间: 2012-08-05; 定稿时间: 2013-07-22

政府信息化、教育、企业、网络游戏等更广泛的领域对高性能计算的需求也迅猛增长.除了硬件设备决定高性能计算的能力,还要有合适的基础软件来释放其能力.

涵盖矩阵计算理论[1]的线性代数软件库正是发挥高性能计算系统的重要基础软件.它提供矩阵乘法、矩阵分解、特征值求解和线性方程组解法器等功能.以 LINPACK 的产生为标志,线性代数库大致从 19 世纪 70 年代中期产生,从基本实现(LINPACK)到底层并行优化的实现(LAPACK),再到算法级别的并行(PLASMA[2,3]),然后随着 GPU 的出现而产生的 GPU 版本的线性代数库(MAGMA,CULA).如图 1 所示,我们可以明显看出,线性代数库软件一直随着计算机体系结构的变化而发展,它需要适应新的架构,从而发挥性能.

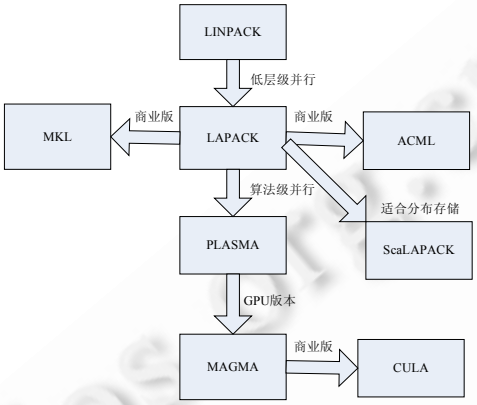

图 1 线性数学库软件发展历史

本文以 QR 分解为例研究 CPU/GPU 异构体系架构上的 MAGMA 软件包.首先对算法进行分析;然后测试 MAGMA 数学库,提出优化和调优的方向;最后对矩阵分解块大小的调优提出了一种简单的自适应方法.

1 MAGMA 算法架构分析

QR 分解算法[1]是最基本的分解算法,也是线性代数库最重要的算法之一,广泛用于解线性方程组、求特征值、求解最小二乘问题等.矩阵 A 的 QR 分解为 $A=QR$,其中,Q 为正交矩阵,R 为上三角矩阵,一般 QR 分解方法主要有 Schmidt 正交化、Givens 变换和 Householder 变换.本文中的算法均采用 Householder 变换法.

1.1 基本QR分解算法

MAGMA 中的基本 QR 分解算法如图 2 所示(以 100×70 为例).一个 100×70 的矩阵,根据算法选取块大小为 $NB\times NB$=32×32.算法每一次迭代计算一个 PANEL(被选为此次进行分解的列块)的分解.

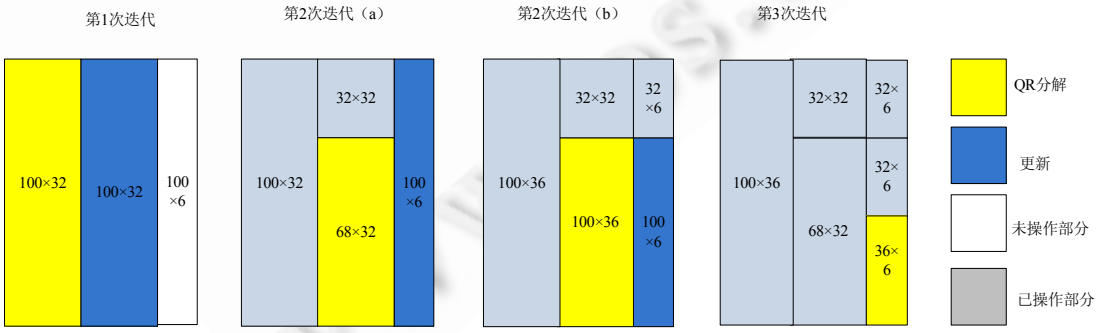

图 2 基本 QR 分解算法

第 1 次迭代,使用 CPU 的 QR 分解函数 lapackf77_sgeqrf 对左边的 100×32 大小的块进行 panel 分解;然后

利用 panel 分解的结果对中间的 100×32 加以更新(使用 GPU 的 kernel 函数 magma_slarfb_gpu),这两者之间串行执行.

第 2 次迭代,首先对最右边的 100×6 进行更新(使用 GPU 的 kernel 函数),然后对中间的 68×32 进行分解(CPU 的 kernel 函数),这两者之间并行执行,如图 2(a)所示.然后算法判断剩余矩阵列数小于 32,不需要进行第 3 次迭代,于是把剩余的 100×6 也进行了更新,如图 2(b)所示.

第 3 次迭代的时候,发现剩余需要分解的矩阵已经小于块大小 *NB*,所以使用 CPU kernel 函数进行分解.

需要注意的是,我们主要的并行计算如第 2 次迭代(如图 2(a))所示,即把上一次分解带来的更新与这一次的分解在 CPU 和 GPU 上并行执行.我们的图示矩阵规模过小,当矩阵规模扩大时,大量的运算均在这一步执行.

1.2 CA(communication avoiding)QR算法[5]

Communication avoiding 算法最近在线性代数库领域被大量研究,是矩阵分解优化的新方向,国内对此还缺乏关注.其应用在单 GPU 和多 GPU 中,均取得显著的性能提升[5].MAGMA 的新版本中也使用了 CA 技术.

高瘦矩阵因为列数相对较少,panel 分解占的比例较大,而 panel 分解是不规整的计算,通信占的比例较大,并不适合密集计算.因而很多 QR 分解算法对方阵的情况表现较好,对高瘦矩阵的情况则不然.而 TSQR 算法通过提高并行性解决了这个问题.CAQR 则在 TSQR 算法上进一步加以扩展,通过隐藏或者减少数据传输来减少处理器之间以及处理器与全局存储器之间的通信,以提高算法效能.

TSQR(高瘦矩阵的 QR 分解)的分解采取完全不同以往分块算法的方式.它不直接对一整列进行分解.主要有以下两步:

(1) 首先把矩阵竖直地分成块,然后对每一个小块做 QR 分解,得到 U_i(即正交矩阵 Q)和 R_i(即上三角矩阵 R).

(2) 然后采用树型规约消去主对角元下面元素,即 R_1 下面的元素.如图 3 所示中的二叉树规约,上面两个块进行规约得到 U_{12} 和 R_{12},下面两个块进行规约得到 U_{34} 和 R_{34},继续规约得到最终结果 U_{1234} 和 R_{1234}.可以采用任何树形结构进行规约,例如四叉树.采用什么样的树结构取得最优性能与体系架构有关.

TSQR 的每一个块的操作都是独立的,显示了极高的并行性.而且 block size 是可调的,选择合适的大小从而匹配 cache,将可以极大地提高性能.

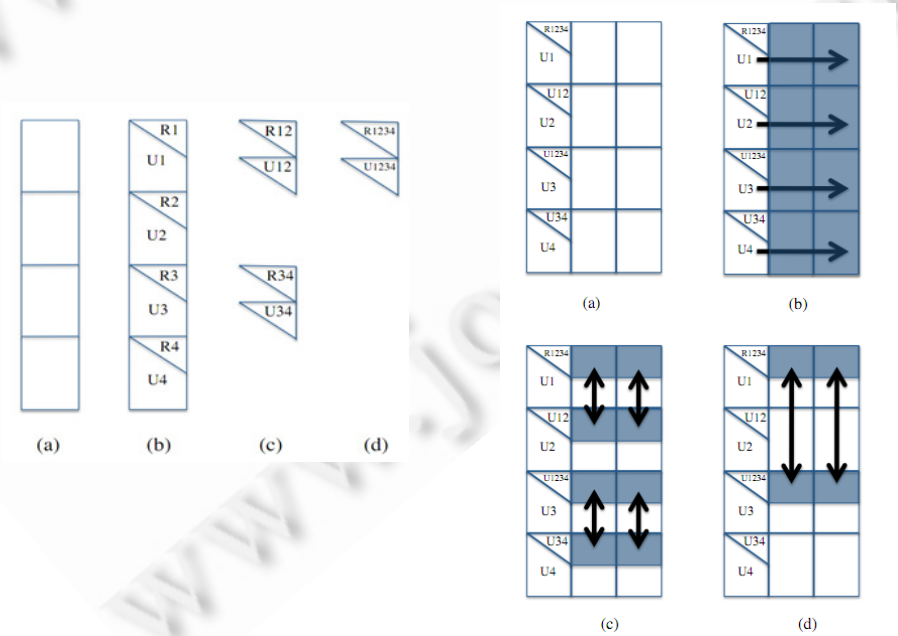

图 3 TSQR 和 CAQR[5]

CAQR 是在使用 TSQR 的基础上发展起来的.它使用 TSQR 来进行 panel 分解.因为 TSQR 在 panel 分解时是以块为单位的,那么尾子矩阵更新可以在整个 panel 分解完成前就开始.这去除了标准分块 householder QR 分解算法中的同步,增加了并行性.尾子矩阵更新有两种可选方式.第 1 种叫作水平更新,如图 3(b)所示.它等待 panel 分解完成,生成所有的更新矩阵 H^T,向右边进行更新操作.这种方法简洁,块的更新是独立的.第 2 种叫作树形规约更新,如图 3(c)所示.右边矩阵的树形规约更新伴随着左边 panel 树形规约分解的每一步.这种方法较复杂,而且对于 GPU 来说,它对矩阵的数据访问不规则且相对稀疏.

2 MAGMA 测试分析

本节使用两个测试平台:平台 1 为两路四核 Xeon 5550 处理器,主频 2.67GHz,内存 16G,GPU 为 TeslaC2050,显存 2687.2MB,平台 2 为八路八核 Xeon7550 处理器,主频 2GHz.平台 1 单精度浮点性能总和为 1.2TFLOPS,平台 2 为 1.02TFLOPS,二者差距不是很大.测试软件为 MKL 10.1.1.019,PLASMA 2.4.5(调用 LAPACK3.4.0 中的自带 BLAS),MAGMA 1.1.0(调用 GOTOBLAS2 和 CUBLAS4.2.9),CULA R14(调用 CUBLAS 4.2.9).

MAGMA 中,QR 分解的矩阵存放在 CPU 内存或者 GPU 内存均可.因为 GPU 承载的计算量较大,我们的计算规模也比较大,为了得到较好的性能,在测试中我们选择把矩阵存放在 GPU 的内存中.

2.1 与CPU数学库的性能对比

为了增加软件包之间对比结果的可靠性,我们在此添加了 LU 和 Cholesky 分解的情况.MKL 和 PLASMA 均运行在 CPU(64 核)平台上,MAGMA 运行在 GPU(C2050)平台上.需要说明的是,MAGMA 中的 Cholesky 分解(图示的 spotrf 函数)因为自身测试函数的 BUG 问题在矩阵规模为 15 000 和 17 000 的情况下无法得到结果.根据图 4 和表 1,我们得到如下分析:

首先,对于 QR 和 LU 分解,在两个系统的浮点性能接近的情况下,在各个规模下,MAGMA 的性能均好于 MKL 和 PLASMA,加速比达到 1.3 以上(见表 1)(需要注意的是,GPU 平台仅使用一块 Tesla C2050 GPU 就达到了 1T 的性能,性价比非常高).对于 Cholesky 分解的情况,MAGMA 却比 MKL 要差一点,这可能是因为 MAGMA 对于 Cholesky 分解做得不够,内核等优化得不好所示.

其次,MKL 的 LU 分解在 CPU 平台的性能达到 777.46 GFLOPS,达到理论峰值 1.02TFLOPS 的 76%,这已经很接近 SGEMM 在 CPU 平台的实测性能峰值 810.13GFLOPS,所以除非 GEMM 的性能继续提升,否则,QR 分解的整体性能不可能继续提升.而 GEMM 继续提升的空间已经很小.反观 MAGMA,性能峰值为 Cholesky 的 656.39GFLOPS,占系统峰值 1.2TFLOPS 的 55%,所以对于 QR,LU,Cholesky,在 GPU 上优化提升的空间比较大.

我们可以看到,通过调用高度优化的 BLAS 的 CPU 上的数学库的性能已经接近机器性能,提升空间不大.GPU 开发的数学库是潜在的提升性能的重要手段,尤其是随着 GPU 的蓬勃发展,浮点计算性能飞速提升.

表 1 MAGMA,MKL,PLASMA 性能对比

	MAGMA	MKL		PLASMA	
	峰值性能 GFLOPS	峰值性能 GFLOPS	加速比 MAGMA/MKL	峰值性能 GFLOPS	加速比 PLASMA/MKL
QR	625.68	463.80	1.35	76.08	8.22
LU	579.87	418.28	1.39	56.37	10.29
Cholesky	656.39	777.66	0.84	79.14	8.29

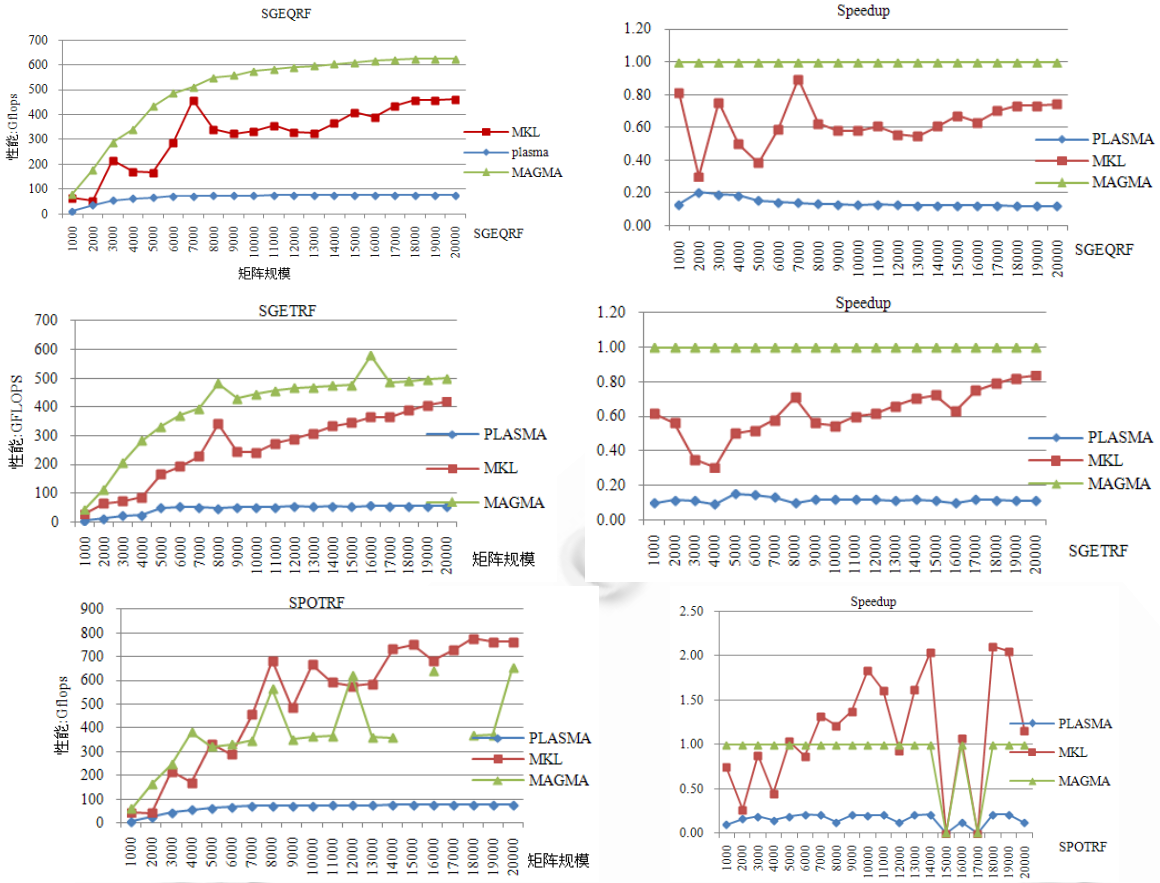

图 4 CPU 数学库性能对比和相对 MAGMA 的加速比

2.2 GPU数学库性能对比

由图 5 我们看到,MAGMA 总体性能好于 CULA.在矩阵规模较小的情况下,CULA 与 MAGMA 相差不大,甚至超过 MAGMA,这是因为 CULA 的 sgemm 做得更好;而随着规模的扩大,MAGMA 好于 CULA,这是因为计算量增大,而 MAGMA 新近采用 CA 算法避免了通信,隐藏了延迟,提高了并行性,从而使性能得到了提升.平均加速比为 1.4.

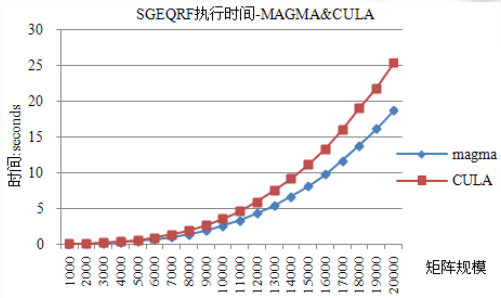

图 5 GPU 数学库性能对比和加速比

2.3 矩阵胖瘦的影响

如图 6 所示,矩阵行数为 20 000,列数从 1000~17000(由于受到显存大小的限制,无法进行更大规模(大于

17 000)的测试).首先,刚开始的时候加速比并不是很明显,二者的性能值较为接近,这是因为起始时的计算量太小,而两个系统的浮点计算能力又很强,主要时间用在数据传输上,数据传输时间差别不大,大概均为 0.4s(矩阵规模为 1 000 时),计算能力的差距体现不出来.然后,当矩阵列数逐渐加大,当增大到 4 000 时我们可以看到,加速比为最大,约为 2.5,这个时候就体现出 MAGMA 矩阵分解中使用 TSQR(tall and skin QR)[5]算法的好处,这种算法对于高瘦矩阵有更好的表现.最后,当矩阵规模大于 4 000 时,加速比逐渐降低,最后达到 1.2~1.3 的水平,稳定下来.这是因为随着矩阵列数的增多,即矩阵变得越来越胖,起决定因素的是矩阵乘的操作.对比的结果显示了二者矩阵乘 SGEMM 函数的性能和机器的浮点计算性能.

综上我们可知,MAGMA 很适宜计算高瘦矩阵的分解.

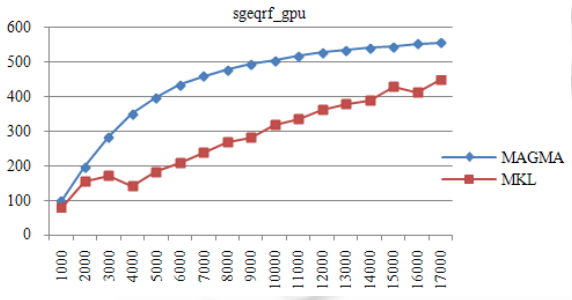

图 6 矩阵胖瘦对性能的影响和加速比

3 矩阵 BLOCK 大小的自适应调优

3.1 实测性能

图 7 是实际测试得到的在各个矩阵规模(1000×1000~10000×10000)下,算法性能随着分块大小的变化示意图.

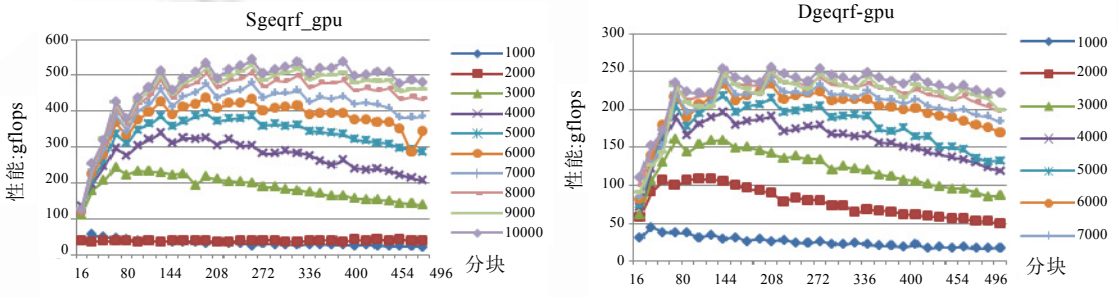

图 7 单精度 QR 分解和双精度 QR 分解实测性能

观察图 7 中单精度的情况(双精度类似).首先从矩阵规模来看,当矩阵规模较小时(小于 2 000),SGEQRF 性能稳定且较低,原因是数据量太小,计算在短时间内完成,函数启动时间占总时间比例较大.其次,SGEQRF 性能随着分块的增大总体上呈简单的上升然后下降趋势,主要是因为较小的分块导致系统把大量时间花在分块和任务划分上,而过大的分块会使块数目减少从而导致并行度不够,所以必须找到适中的分块.另外,虽然随着块大小的增加,性能上升到较高后总体的性能在一定范围内发生变化,但是会出现抖动.例如,当矩阵规模为 10 000 时,分块大小 *NB* 为 128,性能为 512.6GFLOPS,但若增加分块大小 *NB* 为 144,则性能仅为 456.9,性能差距达到 11%.然而,随着分块大小继续增加,当 *NB* 为 192 时,性能为 535.33,性能上升 17%,因此,进行分块调优具有重要的应用价值.

3.2 自适应调优-混合调优法

3.2.1 算法简介

目前,稠密矩阵采用的自适应策略主要有 3 种模型[4]:生成和测试模型(generate-and-test model)、性能分析模型(analytical performance model)、忽略缓存模型(cache-oblivious model)[6].生成测试模型,顾名思义,即使用测试的方法(简单的穷举等都可归于这一类)找到优化的参数.性能分析模型的主要手段是分析,分析当前软、硬件环境(体系结构,编译器等)或者代码的特点(例如,计算密集型还是访存密集型),得到好的参数值.忽略缓存模型方法是对块规模递归减小然后测试.该方法借鉴以往的性能分析方法和生成测试方法,对分块大小提出了一种适合 MAGMA 架构的混合调优方法.

性能分析:

CUDA 实际执行时,block 中线程以 warp 为单位,warp 大小是 32,也就是 32 个 thread 会被群组成一个 warp 来一起执行;同一个 warp 里的 thread 会以不同的数据执行同样的指令.而如果 block 里面的 thread 数量不是 32 的倍数,比如说 thread 数目是 70 的话,就会有 3 个 warp:0-31,32-63,64-69.由于最后一个 warp 里只剩下 6 个 thread,所以其实在计算时,就相当于浪费了 26 个 thread 的计算能力.

TeslaC2050 是一种 Fermi 架构,采用双 warp 调度.每一个 SM 都有两个指令发送单元,可以同时让两个 warp 相互独立地并发运行.Fermi 同时并发调度两个 warp 的一条指令分别在 16 个一组的 CUDA core(即 SP)上进行计算,或者在 16 个存/取单元运行,或者 4 个 SFU 上运行,从而提高并行度,充分发挥计算能力.这样我们可以得到:在一个 GPU 核心周期,一个 SM 可以完成两个 warp 的计算,即 64 个线程的计算.

综上,64 的倍数为理论上的较优点,但为了涵盖较多的点,在测试时选择 32 的倍数为测试样本点.

生成测试:

在测试部分,我们只需搜索分块大小为 32 的倍数的点.以矩阵大小 4 000 为例,首先随机选取一个测试点 *NB*=192,标记为最优块.然后分别向两侧搜索,每边取两个值,即选择 *NB*=160,128 和 *NB*=224,256 进行测试,若出现两个值都比最优块的性能要好,那么舍弃另外一边,仅进行单边搜索,并且把最优值进行替换.虽然我们由测试数据可以看出,性能随着分块的增大呈现较为严格的先上升后下降的趋势,但是为了防止某个点出现异常,我们使用两个点的结果,以保证结果是可靠的.

当然,为了防止最优点附近,两个测试值,一个比最优值好,一个比最优值差(这种情况既可能是搜索到最优点,也有可能是测试的异常或者错误造成),我们选择再增加一个搜索点,即用 3 个点来判断性能走势.搜索结束的标志是最优值不再发生变化.

3.2.2 实验结果

(1) 方阵的情况

图 8 展示了不同分块算法在不同规模下取得的性能.最优分块性能是实际测试的最优性能,“步长 32”算法是指搜索步长为 32(分块大小为 32 倍数).

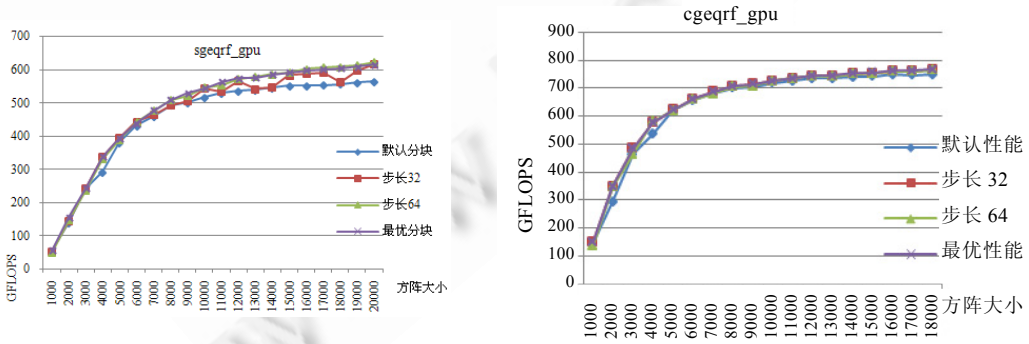

图 8 性能对比结果

图 8 所示对于 *S* 精度的 QR 分解,当矩阵规模为 20 000 时,“步长 32”算法对应默认分块加速比为 1.09.平均加速比为 1.04.*S* 精度规模较大时加速比较好.对于 *C* 精度的 QR 分解函数,自适应调优相对于默认分块的最大加速比在 2 000 处取得,加速比为 1.19.平均加速比为 1.03.规模较小时加速比较好.*C* 精度虽然平均加速比不高,但是可以得到最优性能对于自适应调优性能的平均加速比为 1,这说明,自适应调优在所测试的矩阵大小点均取得了最优性能.同时也说明,通过调优加速的空间不大,但是对于其他类型和函数有可能取得好的性能.

观察第 3.1 节中算法性能随矩阵分块的改变可以看到,每当分块 *NB* 为 64 的倍数时,性能有一个相对急速的上升,这也与我们对 TeslaC2050 的性能分析相吻合.只取 *NB* 为 64 的倍数的点,可以得到一个简单的先上升后下降的曲线.这样就不需要考虑局部最优值,得到的结果基本上是全局最优,而且搜索速度会大大加快,称为“步长 64”算法.

“步长 32”算法与“步长 64”算法除了每次搜索的步长不同之外,更主要的区别是“步长 64”算法仅仅是基于一个简单的假设:对于 QR 分解,局部最优值的曲线呈现简单的先上升后下降的抛物线形式.“步长 32”算法需要在局部最优值处进行繁琐的判断以避免局部最优从而达到全局最优.观察 sgeqrf_gpu 可以看到,“步长 64”算法已经达到最优性能,性能曲线和最优性能曲线基本吻合,但是平均加速比为 1.05.对于 cgeqrf_gpu,“步长 64”算法也达到最优性能,性能曲线和最优性能曲线基本吻合,但是平均加速比更是只有 1.02.

对于方阵来说,得到的结论是:1) 自适应调优是有效的,接近实测最优值.2) 对分块大小进行调优的空间很小,已经没有调优的必要.

(2) 高瘦矩阵的情况

因为 MAGMA 最新版本开始采用 CAQR 算法;由如算法分析可知,CAQR 正是以高瘦矩阵的 QR 分解算法(TSQR)为基础的,所以 TSQR 的性能很重要,有必要进行调优.

单精度测试行数为 20 000,列数从 1 000~20 000.“步长 32”算法的平均加速比是 1.04,最大加速比出现在列数 1 000 和 2 000 处,为 1.10.“步长 64”算法的平均加速比是 1.05.最大加速比出现在 19 000 和 20 000 处,为 1.10.总体来看,加速比都不好.原因应该是 MAGMA 中单精度计算比较规则而且优化得比较好.对单精度复数精度来说,“步长 32”算法的平均加速比是 1.16,“步长 64”算法的平均加速比是 1.15.但是可以明显地看到,主要的加速出现在列数比较少的情况,主要是 1 000~5 000 之间,最大加速比均为 1.8 之多.而随着矩阵的变宽,矩阵乘法的性能对整体性能的贡献占了主要的部分,分块大小的影响不大.这说明,对高瘦矩阵进行参数调优还是很有必要的,如图 9 所示.

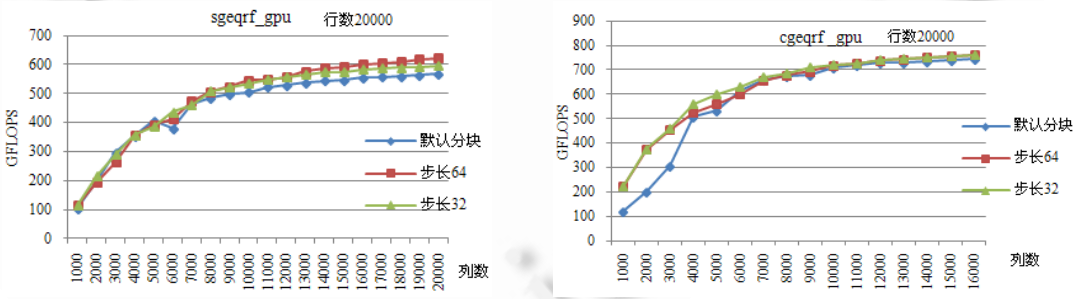

图 9 高瘦矩阵的自适应调优结果

因此,对于高瘦矩阵,可以得到如下结论:对于矩阵很瘦的情况进行调优是有必要的;因为 *S* 和 *C* 精度 QR 分解算法的性能在自适应调优之后达到很高的加速比,尤其是 *C* 精度的不规则运算较多,调优性能更好.

4 结论与未来工作

多核计算已经普及化,由于 GPU 所能提供的巨大计算性能,最新的超级计算机系统均开始采用 CPUs+GPUs 异构的方式构建.可以说,硬件系统构架即将进入多核加 GPU 的异构时代.CPU 的线性数学库已经

达到系统性能峰值的约 80%,很难继续提升.GPU 因为其天然的高浮点计算能力,所以更适于做线性代数运算.

我们根据测试可以看到:首先,MAGMA 性能表现很优异,无论是对比 CPU 的数学库还是 GPU 的数学库,基本上都表现出了较好的加速比.其次,MAMGMA 在使用时,如果矩阵是高瘦矩阵或者规模很大,都表现得更好,更有利于发挥 GPU 强大的能力.最后,我们可以看到,未来的 GPU 需要把显存做得更大或者在 GPU 直接读取内存数据方面做得更好;并且,未来的算法优化方向是减少通信,进一步地提高并行性,把 CPU 和 GPU 之间的任务分配调度做得更好,从而充分利用各部件的计算性能.

我们进行了自适应调优测试,取得了一定的效果,得到的结论是:

(1) 分块调优,对于高瘦矩阵进行调优是非常必要的,但是对于胖矩阵,方阵调优则是不必要的.

(2) 仅仅对于一个参数(*NB*)的调优是不够的,调优性能空间太小,应对多个参数(与体系架构相关)进行,这样才能精细地加以控制,获得好的结果.

未来工作的重点应该是:

(1) 对于运算相对 QR 不规则的其他函数进行调优测试.

(2) 进行多参数调优(对 MAGMA 实现类似 LAPACK 的针对体系架构的多参数控制).

(3) MAGMA 在 AMD 平台上的自适应调优(由于 MAGMA 在 CUDA 平台已经优化得很好,在 OPENCL 平台做得相对薄弱).

(4) 发展针对特定平台的优化方法.

References:

[1] Golub GH, Van Loan CF, Matrix Computations. Beijing: Post & Telecom Press, 1996. 223−226.

[2] Buttari A, Langou J, Kurzak J, Dongarra J. A class of parallel tiled linear algebra algorithms for multicore architectures. ICL Technical Report, UT-CS-07-600, 2007.

[3] Buttari A, Langou J, Kurzak J, Dongarra JJ. Parallel tiled QR factorization for multicore architectures. Concurrency Computat. Pract. Exper., 2008,20(13):1573−1590.

[4] Lü ZC, Zhang YQ, Wang T, Xiao XJ. Design and implementation for PLASMA auto-tuning and performance optimizing. Computer Science, 2012,39(4):282−286.

[5] Orti GQ, Orti EQ, Van Zee ECFG, van de Geijn R. Scheduling of QR factorization algorithms on SMP and multi-core architectures. Technical Report, University of Texas at Austin, Department of Computer Sciences, 2007.

[6] Agullo E, Augonnet C, Dongarra J, Faverge M, Ltaief H, Thibault S, Tomov S. QR factorization on a multicore node enhanced with multiple GPU accelerators. In: Proc. of the 2011 IEEE Int'l Parallel & Distributed Processing Symp. (IPDPS). 2011. 932−943.

附中文参考文献:

[4] 吕渐春,张云泉,王婷,肖玄基.PLASMA 自适应调优与性能优化的设计与实现.计算机科学,2012,39(4):282−286.

肖玄基(1988－),男,河南平顶山人,硕士生,主要研究领域为并行软件,计算数学.
E-mail: growj@126.com

李玉成(1961－),男,研究员,主要研究领域为并行计算.
E-mail: yucheng@iscas.ac.cn

张云泉(1973－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域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.
E-mail: yunquan.zhang@gmail.com

袁良(1984－),男,博士,CCF 会员,主要研究领域为并行计算.
E-mail: ilpiny@gmail.com