基于龙芯 3A 的 LAPACK 函数优化①

张 斌, 顾乃杰, 何颂颂, 刘斌斌

(中国科学技术大学 计算机科学技术学院, 合肥 230027)
(安徽省计算与通信软件重点实验室, 合肥 230027)
(中国科学技术大学 中科院沈阳计算所网络与通信联合实验室, 合肥 230027)

摘 要: 针对龙芯 3A 体系结构, 通过底层 BLAS 库的优化、LAPACK 分块算法中分块大小的改善以及 LAPACK 函数的单独优化这三种途径来提升 LAPACK 函数的性能. 用 LAPACK 自带的性能测试程序进行测试, 实验结果表明, 有 240 个 LAPACK 函数的性能提升达到 30%以上, 占全部性能测试函数的 81%.

关键词: LAPACK; BLAS; 龙芯 3A; 优化; 双单精度

Optimization of LAPACK Based on Loongson 3A

ZHANG Bin, GU Nai-Jie, HE Song-Song, LIU Bin-Bin

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
(Anhui Province Key Laboratory of Computing and Communication Software, Hefei 230027, China)
(USTC & SICT Network and Communication Joint Laboratory, Hefei 230027, China)

Abstract: According to the characteristics of Loongson 3A architecture, this paper shows three ways to improve the performance of LAPACK: optimization of the underlying BLAS library, the selection of the best block size of the block algorithm in LAPACK and optimization of the specific LAPACK functions. By running the LAPACK Timing Programs, experimental results are obtained and it shows that the performance of 240 LAPACK functions, which account for 81% of all the LAPACK Timing Programs, is increased by more than 30%.

Key words: LAPACK; BLAS; Loongson 3A; optimization; paired single

1 引言

LAPACK (Linear Algebra PACKage)[1]是田纳西大学、加州大学伯克利分校、科罗拉多大学丹佛分校等联合开发的以 BLAS 库(Basic Linear Algebra Subprograms)为基础, 用 Fortran 语言编写而成的一个高效的线性代数函数库, 提供了功能非常丰富的矩阵运算函数, 其在数值和工程计算中应用非常广泛, 可用于解决线性方程组、最小二乘法问题、特征值问题和奇异值问题, 以及矩阵分解(LU 分解、Cholesky 分解、QR 分解、SVD 奇异值分解, Schur 分解, Generalized Schur 分解)和条件数估计等问题. LAPACK 的高效性使得它经常作为底层库, 供上层的数学工具调用, 如 MATLAB 就是通过调用 LAPACK 来完成矩阵运算.

函数库的优化需要充分结合目标机器的体系结构特征, 最大程度的发挥特定体系结构的优势. 龙芯 3A 处理器是我国具有自主知识产权的四核通用 CPU, 采用 65nm 制造工艺, 最高运行主频 1GHz, 文献[2]中列出的与程序优化密切相关的龙芯 3A 的主要特征如下:

① 片内集成四个 64 位的四发射超标量 GS464 高性能处理器核;

② 四发射技术, 即每个周期最多可以发射两条定点指令, 两条浮点指令和一条访存指令中的四条指令;

③ 每个处理器核包含大小均为 64KB 的一级指令 Cache 和数据 Cache, 采用 4 路组相联;

④ 片内集成四核共享的 4MB 二级 Cache;

① 基金项目:国家“核高基”重大专项(2009ZX01028-002-003-005);国家自然科学基金(60833004);高等学校学科创新引智计划(B07033)
收稿时间:2012-03-27;收到修改稿时间:2012-05-18

⑤ 访存部件支持 128 位存储访问;

⑥ 支持寄存器重命名、动态调度、转移预测等乱序执行技术;

⑦ 提供了非阻塞预取指令.

为了获得更高的性能, 各大处理器厂商都根据自身产品的特性设计和实现了自己专用的高效数学库, 如 AMD 的 ACML 和 Intel 的 MKL, 龙芯目前还没有自己的专用数学库, 重新设计和实现一套数学库需要投入大量的人力和物力, 因此针对龙芯体系结构的特征, 在开源数学库的基础上作优化, 是一个花费较小, 收效较高的方案. 本文针对龙芯 3A 体系结构特征, 使用双单精度指令、128 位访存指令、循环展开、指令调度、数据预取等优化技术对 LAPACK 以及底层 BLAS 库进行优化, 取得较好的性能提升, 这对提升龙芯平台上科学计算软件的运行速度, 充分发挥龙芯处理器的运算能力具有十分重要的意义.

本文的组织结构如下: 第一节是引言, 介绍 LAPACK 以及龙芯 3A 的相关背景知识; 第二节介绍影响 LAPACK 函数性能的主要因素; 第三节详述三种优化途径; 第四节是实验结果; 第五节进行总结.

2 LAPACK函数性能的影响因素

LAPACK 在设计时就考虑尽可能多的调用 BLAS 库来完成矩阵运算, 因此 LAPACK 的性能在一定程度上取决于 BLAS 库的性能, 选用高效的 BLAS 库至关重要. LAPACK 自带了原始的 FORTRAN 版的 BLAS 库, 由于未经优化, 自带的 BLAS 库非常低效. 目前被广泛使用的 BSD 开源的 BLAS 实现是 ATLAS 和 GotoBLAS.ATLAS(Automatically Tuned Linear Algebra Software)能够在不同平台上自动生成优化代码, 具有良好的移植性. 本文针对龙芯 3A 体系结构的特性, 在 ATLAS 3.8.3 的基础上进行优化, 得到优化版的 ATLAS 库. 在作性能对比时分别连接原版的 ATLAS 库和优化版的 ATLAS 库作为底层的 BLAS 库.

LAPACK 很多函数采用了分块算法, 即将矩阵划分为一个个的子块, 对子块内数据的访问比较集中, 如 Cholesky 分解的分块算法, LU 分解的分块算法, QR 分解的分块算法和特征值问题的分块算法等, 这对于拥有高速缓存的机器来说具有很大的性能优势. 结合问题的规模以及目标机器的体系结构特性, 选取合适的分块大小是提升分块算法性能的一个关键[3,4].

3 优化途径

3.1 BLAS 库的优化

由于大部分 LAPACK 函数通过调用 BLAS 库函数完成矩阵运算, 因此高效的 BLAS 库是非常关键的. 文献[5]结合龙芯 2F 体系结构特性对 ATLAS 的核心函数进行了优化, 使 LINPACK 实测峰值比原 ATLAS 提高 45%; 文献[6]使用龙芯 3A 的 128 位访存指令和非阻塞预取指令等, 将基于龙芯 3A 平台的 ATLAS 的效率平均提升 20%. 本文在以上工作的基础上对 ATLAS 的部分函数进行优化, 以 CGEMM 为例, 使用双单精度指令进行优化, 使该函数的性能显著提升.

龙芯 3A 浮点寄存器有 64 位, 故每个浮点寄存器可以存放两个单精度浮点数, 分别存于高 32 位和低 32 位. 双单精度指令可以将浮点寄存器的高 32 位和低 32 位分别并行执行操作, 一条双单精度指令可以完成两次浮点运算, 指令条数节省一半. 由于 CGEMM 的运算量为 $O(n^3)$, 访存量为 $O(n^2)$, 属于计算密集型函数, 因此采用双单精度指令, 可以大大提升该函数的性能. 由于 CGEMM 是单精度复数类型的三级 BLAS 库的核心函数, 对 CGEMM 的优化会促使单精度复数类型的其它三级 BLAS 库函数的性能提升.

首先通过矩阵分块将规模较大的矩阵划分成一个个子块, 将数据的多次重复访问都集中在同一子块内, 减少 Cache 失效; 然后再通过矩阵拷贝将同一子块内部的数据拷贝到一块连续的存储空间上, 可以提高 Cache 的命中率[5]. 子块内部的计算为核心计算, 考虑到龙芯 3A 有 32 个浮点寄存器可供用户使用, 采用 A 矩阵的每 4 行乘以 B 矩阵的每 2 列, 累加到 C 矩阵的一个 4×2 的子块上. 如图 1 所示为采用 JIK 循环顺序时的寄存器存放矩阵元素的情况. 完成一轮计算, 需要 16 条双单精度乘加指令, 同时需要取 8 个 A 数组值和 4 个 B 数组值. 浮点运算指令与取数指令之比小于 2:1, 为了充分发挥龙芯四发射特性, 使用龙芯特有的 128 位访存指令, 一次取 4 个单精度浮点数, 减少访存指令条数. 完成一轮计算后, 寄存器 c0 的低 32 位和高 32 位分别存放 rx0×ry0 和 ix0×iy0, c1~c7 依此类推, c8 低 32 位和高 32 位分别存放 rx0×iy0 和 ix0×ry0, c9~c15 依次类推. K 循环结束时, c0~c7 各自的低 32 位减去高 32 位得到 C 矩阵子块的实部, c8~c15 各自的低 32 位加上高 32 位得到 C 矩阵子块的虚部. 优化后的 CGEMM 函数平均性能提升 85%.

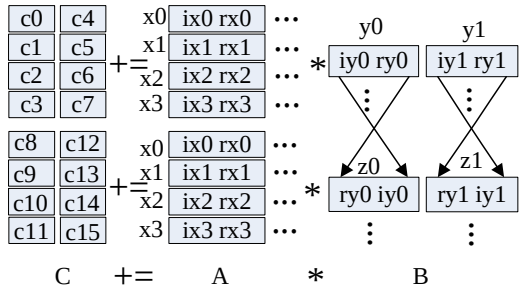

图 1 CGEMM 四行两列相乘寄存器存数情况

LAPACK 函数 CGETRS 是求形如 AX=B, ATX=B 或 AHX=B 的线性方程组的解矩阵, 其中 A 矩阵已由 CGETRF 函数 LU 分解为 A=LU, 原方程组变为 LUX=B, 令 Y=UX 有 LY=B, 先解出 Y, 再由 UX=Y 解出 X. 解 LY=B 和 UX=Y 时, 均可通过调用一次三级 BLAS 函数 CTRSM 来完成. CTRSM 采用递归折半划分的方法将三角阵划分为普通矩阵和小三角阵, 对普通矩阵部分调用 CGEMM 进行计算, 如果划分出来的小三角阵的规模大于预先定义的 RB 值, 那么递归进行这样的划分, 直到小三角阵的规模小于或等于 RB 值时停止划分, 完成小三角部分的计算. 小三角阵运算量总和占总运算量的比例为 RB/M, 其中 M 为大三角阵的规模, 因此当三角阵规模较大时, 小三角阵运算量所占的比例较小, 此时该函数的性能基本取决于 CGEMM 的性能. 图 2 是 CGETRS 在 NRHS 分别为 16、32、64 和 128 时的性能对比情况, 平均性能提升分别为 31%、40%、53%和 64%.

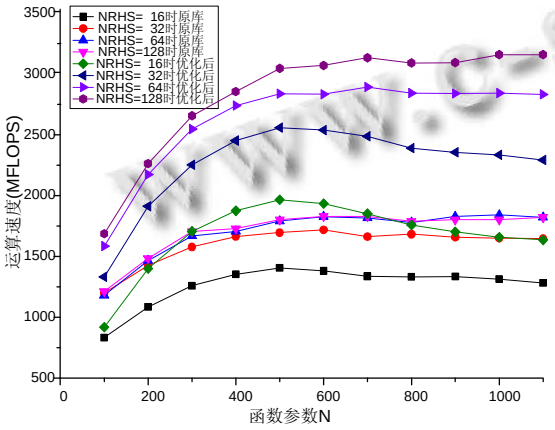

图 2 CGETRS 函数在不同 NRHS 时性能对比图

3.2 分块大小调整

现代通用处理器为了折中性能和造价, 存储系统都采用了层次结构. 龙芯 3A 的每个处理器核有 64KB 的一级数据 Cache, 四核共享 4MB 的二级 Cache. CPU 从一级 Cache、二级 Cache 和内存的取数速度依次减慢, 最理想的情况是要使用的数据都在 Cache 中, 但是由于 Cache 的容量有限, 当矩阵规模较大时, Cache 容纳不下整个矩阵的数据, 可采用分块算法, 将大矩阵划分为一个个能装进 Cache 的子块, 使得程序对数据的多次重复访问都是在块内进行, 降低 Cache 的容量失效和冲突失效. LAPACK 的很多函数都是采用分块算法来实现的, 如 Cholesky 分解的分块算法, LU 分解的分块算法, QR 分解的分块算法和特征值问题的分块算法[3]. 影响分块算法性能的一个重要因素就是分块大小的选取, 而分块大小的选取又跟问题规模和目标机器的特性密切相关. LAPACK 在函数 ILAENV 中对每个分块算法都设定了一组默认的分块大小, 但由于没有考虑到问题规模的变化和目标机器的特性, 经实际测试, 在龙芯 3A 平台上, 其默认分块大小并不一定是最优的. 如图 3 所示为普通矩阵 QR 分解函数 DGEQRF 在不同分块大小时的速度(MFLOPS)图, 可以看出, 分块大小 NB=1 即不分块时, 性能最差; 当矩阵的规模 N 介于 200 到 700 时, 最优分块大小为 NB=32; 当 N 介于 800 到 1100 时, 最优分块大小为 NB=64; 当 N 介于 1200 到 2000 时, 最优分块大小为 NB=96. 由此可见该函数的默认分块大小 NB=32 并非在所有规模下最优, 因此通过分析大量实验数据, 在不同的规模范围内选取最优的分块大小可获得性能提升. 本例中, 当 N 介于 800 到 1100 时, 将分块大小由 32 调整为 64, 可获得平均 6%的性能提升; 当 N 介于 1200 到 2000 时, 将分块大小由 32 调整为 96, 可获得平均 18%的性能提升.

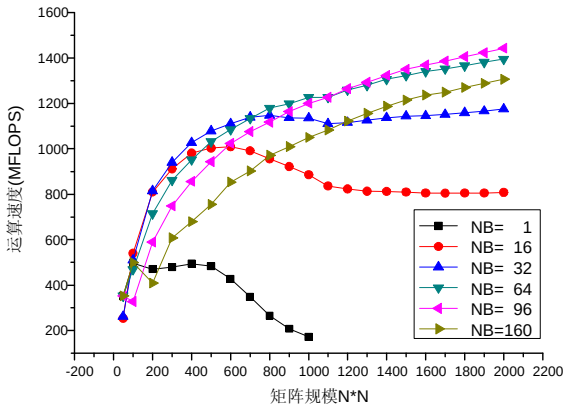

图 3 DGEQRF 函数在不同分块大小时的速度(MLOPS)

3.3 单独优化 LAPACK 函数

部分 LAPACK 函数既没有调用 BLAS 库函数(或者调用的 BLAS 库函数耗时比例并不大), 也没有使用分块算法, 因此只能对这些 LAPACK 函数进行单独优化. 优化方法以循环展开为主, 再辅以指令调度, 变量重命名和数据预取等优化技术. 循环展开是最基本的优化手段, 它以增加代码长度为代价, 减少循环次数以降低循环开销; 由于循环体内部代码变长, 使得有更加充裕的空间进行指令调度, 尽可能减弱数据相关, 同时可以使用寄存器变量保存中间结果供后续使用, 加快取数速度; 对多层循环中的外层循环进行展开, 一次取数可供多次使用, 有效减少访存次数[5].

下面以 LAPACK 中的函数 DGTSV 为例对优化细节进行详述. DGTSV 是用高斯消元法求解三对角方程组 AX=B 的函数, 其中 A 为 N 阶的三对角阵, B 和 X 均为 N×NRHS 的矩阵. 解三对角线性方程组问题是许多科学和工程计算中最基本的问题之一, 在核物理、流体力学、地震数据处理和数值天气预报等领域中有广泛的应用. DGTSV 求解过程主要分两个阶段: 消元和回代. 在消元阶段, 按行自上而下依次比较主对角线上的元素和次对角线上该列的元素, 如果主对角线上的元素绝对值较大, 可直接消元, 反之, 先进行行行交换, 然后消元. 消元阶段结束时, 系数矩阵 A 被更新为最多只含三条对角线(主对角线 D、超对角线 DU 和第二条超对角线 DU2)的上三角阵 U. 在回代阶段, 自下而上依次按公式(1)计算即可求出所有解.

$$\begin{cases} x_{N,j}=\dfrac{B_{N,j}}{D_N} \\ x_{n-1,j}=\dfrac{B_{N-1,j}-DU_{N-1}\gamma X_{N,j}}{D_{N-1}} \\ x_{i,j}=\dfrac{B_{i,j}-DU_i\gamma X_{i+1,j}-DU2_i\gamma X_{i+2,j}}{D_i} \\ i=N-2,N-3,\Lambda\ ,1;\ j=1,2,\Lambda\ ,NRHS \end{cases} \tag{1}$$

现从访存量和运算量对该函数进行分析. 在消元阶段, 每更新一行, 如果没有进行行交换, 需要读取 2×NRHS 个 B 值, 一个 DU 值, 一个 DL 值和两个 D 值, 故访存量为 2×NRHS+4, 运算量为 2×NRHS+3, 如果进行行交换, 需要读取 3×NRHS 个 B 值(通过查看编译出的汇编代码确定), 一个 DL 值, 两个 DU 值和两个 D 值, 故访存量为 3×NRHS+5, 运算量为 2×NRHS+4, 共有 N-1 行需要更新(第一行无需更新), 假设发不发生行交换是等概率的, 则总访存量为(2.5×NRHS+4.5)×(N-1), 总运算量为(2×NRHS+3.5)×(N-1); 在回代阶段, 总访存量为 4×NRHS×(N-1), 总运算量为 5×NRHS×(N-1). 运算访存比约为 1, 由于龙芯 3A 的运算能力远强于访存能力, 因此访存成为影响该函数性能的瓶颈, 主要从减少访存次数和减少 Cache 失效角度进行优化[7]. 令(2.5×NRHS+4.5)×(N-1) < 4×NRHS×(N-1), 即 NRHS 大于 3 时, 回代阶段访存量更多, 其耗时也较多, 故此处是优化的重点. 以下为 DGTSV 函数在回代阶段的伪代码:

```
1   for(J = 1 ; J <= NRHS ; J++ ){
2       B[ N ] [ J ] = B[ N ][ J ] / D[ N ]
3       if(N > 1 ) {
4           B[ N-1 ][ J ] = (B[ N-1 ][ J ]
5               - DU[ N-1 ] *B[ N ][ J ] )/ D[ N-1 ]
6       }
7       for (I = N – 2 ; I >= 1 ;   I-- ) {
8        B[ I ][ J ] = (B[ I ][ J ]-DU[ I ]
9             * B[ I+1 ][ J ] - DL[ I ]
10            * B[ I+2 ][ J ] ) / D[ I ]
11      }
12  }
```

上述的伪代码是公式(1)的代码实现. DU2 和 DL 都是逐行更新的, 由于 DL 逐行消为 0, 与此同时逐行确定 DU2 的值, 因此 DU2 和 DL 可共用数组 DL 保存, 输入时存储次对角线 DL 上的元素, 输出时存储第二条超对角线 DU2 上的元素. 右侧矩阵 B 的元素 B[I][J]在求出与之对应的同一行同一列的解 X[I][J]后就弃置不用, 因此右侧矩阵 B 和解矩阵 X 可共用数组 B 存储, 输入时存储右侧矩阵 B 的元素, 输出时存储解矩阵 X 的元素.

当 NRHS=1 时, 上述的双层循环弱化为单层循环, 此时由于是逆序遍历一次数组, 优化空间很小. 本文考虑 NRHS 稍大情况下 DGTSV 函数的优化, 性能测试中分别取 NRHS=16、32、64 和 128. 本文对回代阶段 JI 循环顺序的双层循环使用了循环展开的方法, 在大幅降低访存次数的同时又逐列访问 B 数组, 采用数据预取技术将 Cache 失效与计算时间重叠, 使得该函数性能显著提升. 循环展开中的关键问题是如何选择合适的展开因子. 假设外层和内层循环展开因子分别为 M 和 K. 由于每 M 列共用一组 DU、DL 和 D 数组,

故总访存量减为(1+3/M)×NRHS×(N-2). 为了尽可能降低访存量, M 的取值越大越好, 但是受到浮点寄存器个数的限制, M×K 应约等于 32. 实测表明取 M=8, K=4 时, 效果最好, 此时总访存量减为 1.375×NRHS×(N-2). 同时, 由于浮点除法比乘法延迟更大, 可将 NRHS×N 次除法变为 NRHS×N/M 次除法和 NRHS×N 次乘法. 图 4 为在 NRHS 分别取 16、32、64 和 128 时, DGTSV 函数按 8×4 展开前后性能对比图, 平均性能提升分别为 175%、221%、250%和 253%.

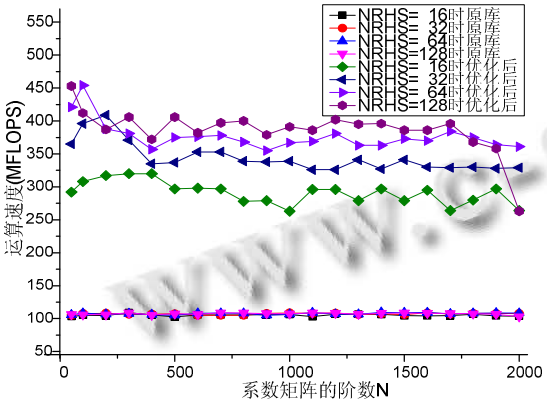

图 4 DGTSV 函数 8×4 循环展开前后性能对比

4 实验结果

实验平台为龙芯3A单机, 其工作频率为900MHz, 内存4G. 操作系统是Linux 2.6.36, 编译器为gcc 4.4.6. LAPACK 版本为 3.2.1, BLAS 库选用 ATLAS 3.8.3.

由于LAPACK 3.2.1没有自带的性能测试程序, 所以本文将 LAPACK 3.0 自带的性能测试程序移植过来. 性能测试中将函数分为两大类: LIN(线性问题)和 EIG(特征值和奇异值问题). 本文中原库性能指的是未经优化的 LAPACK 函数连接未经优化 ATLAS 库所测得的性能, 优化库性能指的是优化后的 LAPACK 库并连接优化后的 ATLAS 库所测得的性能. 大部分 LAPACK 函数都有四种数据类型: S(单精度实数)、D(双精度实数)、C(单精度复数)和 Z(单精度复数). 本文以 D 类型为例给出实验效果. 在有性能测试的 72 个 D 类型的 LAPACK 函数中, 性能提升超过 30%的有 62 个, 未达 10%的仅有 3 个, 算术平均值为 75%. LAPACK 函数名基本上是 TXXYYY 的形式, T 表示数据类型(S、D、C、Z) , XX 表示矩阵类型, YYY 表示函数的计算任务[8]. 表 1 列出了部分具有代表性的函数性能提升百分比, 选取时尽量以计算任务为区分度.

表 1 D 类型的部分 LAPACK 函数性能提升百分比

LIN		EIG	
函数名	性能提升(%)	函数名	性能提升(%)
DGTSV	59.6	DGESDD	64.1
DGELS	43.5	DGGHRD	51.8
DGETRF	77.8	DGEBRD	60.2
DGBTRS	72.3	DSYTRD	27.7
DPPTRI	231.7	DSTEDC	66.9
DGEQPF	34.4	DPTEQR	72.9
DGEQRF	30.6	DHSEIN	32.2
DGEQLF	30.4	DHGEQZ	61.1
DORMHR	180.5	DTGEVC	14.8
DORMTR	204.1	DSTERF	36.7
DORMBR	163.3	DBDSDC	98.4

5 总结

本文针对龙芯 3A 体系结构, 通过以下三种途径来提升 LAPACK 函数的性能: ①底层 BLAS 库的优化; ②LAPACK 分块算法中分块大小的改善; ③LAPACK 函数的单独优化. 具体优化方法有: 特殊指令(双单精度指令, 128 位访存指令)的使用, 循环展开, 指令调度, 数据预取等. 使用 LAPACK 自带的性能测试程序进行测试, 实验结果表明, 有 240 个 LAPACK 函数的性能提升达到 30%以上, 占全部性能测试函数的 81%.

参考文献

1 Anderson E, Bai Z, Bischof C, et al. LAPACK Users' Guide. (3rd Edition), 1999.

2 中国科学院计算技术研究所.龙芯 3A 处理器用户手册 0.1 版. 2009.

3 李玉成.LAPACK 中的分块算法及其效果.数值计算与计算机应用,2001,22(3):172−180.

4 Chi XB, Li YC, Sun JC, et al. Developing High Performance BLAS, LAPACK & ScaLAPACK on HITACHI SR8000. Proceedings HPC-ASIA2000, IEEE Computer Society, Beijing, May 14−17, 2000, (ISTP Include).

5 顾乃杰,李凯,陈国良,等.基于龙芯 2F 体系结构的 BLAS 库优化.中国科学技术大学学报,2008,38(7):855−858.

6 李毅.多核龙芯体系结构下 BLAS 库的优化[硕士学位论文].合肥:中国科学技术大学,2010.

7 苏波,李凯,徐志广,何颂颂.龙芯 2F 上的访存优化.计算机系统应用,2010,19(1):171−175.

8 Blackford S, Dongarra J. Installation Guide for LAPACK (VERSION 3.0), 1999.