# 并行复数矩阵乘性能初探

● 张丹丹 上海超级计算中心 上海 201203
● 蒋荣琳 上海超级计算中心 上海 201203

摘要：

矩阵乘运算是求解线性方程中影响计算性能的关键部分，而复数矩阵乘运算也在多个领域使用，且研究甚少。文中对现有数学库及对复数矩阵乘运算的支持进行了研究，并对比了scaLAPACK、MKL在并行复数稠密矩阵乘运算上的性能表现。该项工作对大规模并行环境下复数稠密矩阵乘运算具有指导和借鉴意义。

关键词：并行，复数矩阵乘，BLAS，ScaLAPACK

## 1. 引言

矩阵乘在科学计算中广泛应用，好的矩阵乘算法往往能很大程度上提高求解线性代数问题的效率。一般情况下，研究多以实数矩阵为对象，而复数矩阵乘也在多个领域中应用，到目前为止，有关复数矩阵乘的研究很少。本文以复数方阵为对象，对现有数学库中串行和并行复数矩阵乘的支持及性能进行了初步分析。

测试以“魔方”超级计算机系统为平台，并行复数矩阵乘选择了支持并行BLAS的MKL和ScaLAPACK两种函数库，后台BLAS库分别采用标准BLAS、MKL及OpenBLAS进行比较分析。

并行复数矩阵乘通过数学库中scaLAPACK中PZGEMM来实现，为了保持完整性，论文后续章节安排如下，第2部分介绍BLAS/ZGEMM，第3部分介绍ScaLAPACK，第4部分介绍并行复数矩阵乘PZGEMM及性能测试，第5部分为结论。

## 2. BLAS

### 2.1 BLAS简介

BLAS[1]（Basic Linear Algebra Subprograms）是基础线性代数程序集，它是采用Fortran编写的一组应用程序接口（API）标准，广泛应用于科学及工程计算，是许多数学软件的核心。该程序集最初发布于1979年，并用于建立更大的数值程序包（如LAPACK）。在高性能计算领域，世界超级计算机TOP500[2]排行Linpack运算成绩很大程度上取决于BLAS库中子程序DGEMM的表现。为提高性能，主要的硬件供应商如Intel、IBM、AMD、NVIDIA针对其硬件对BLAS库进行了高度优化，即MKL、ESSL、ACML、cuBLAS，为商业软件。同时，也有一些组织为各种架构的系统做了不同的BLAS实现，如Atlas、GotoBLAS、OpenBLAS、MAGMA，为开源软件。

其中，OpenBLAS[2]是在GotoBLAS2.1.13BSD版本基础上的开发的优化版本，为开源软件。MKL和其他商业版本一样，将调用得最多并直接影响性能的核心函数采用目标平台的汇编语言进行实现，再使用高级语言进行包装整合。上层使用Pthread或OpenMP进行多核上线程级的并行化[3,4]。

cuBLAS[5]为NVIDIA CUDA基本线性代数子程序，是一款GPU加速版本的完整标准BLAS库；MAGMA[6]为异构环境下的LAPACK库，是在cuBLAS库中GPU加速BLAS例程的基础之上开发的。

BLAS按照功能被分成三个级别：Level 1实现矢量-矢量运算；Level 2实现矩阵-矢量运算；Level 3：矩阵-矩阵运算。按照操作数的精度来分有4种：S（单精度实数），D（双精度实数），C（单精度复数），Z（双精度复数）。BLAS Level1共46个子函数，Level 2有66个子函数，Level 3有30个子函数。

BLAS Level3进行矩阵与矩阵之间的基本运算，是整个BLAS库中被调用最多的函数，计算量达到$O(N^3)$，访存次数为$O(N^3)$。与Level 2有所不同的是，BLAS Level 3集中于Cache的访问，如果能高效使用Cache，其计算峰值与处理器的理论峰值性能接近。BLAS 3是整个BLAS的核心，而GEMM矩阵乘子

函数则是BLAS 3的核心函数。ZGEMM即为复数矩阵乘运算。

文中选取MKL和OpenBLAS作为对比数学库，考查串行复数矩阵乘和并行复数矩阵乘性能。

### 2.2 ZGEMM

ZGEMM实现复数矩阵与矩阵乘，与实数矩阵乘DGEMM类似。假设有一个M×K的矩阵A，一个K×N的矩阵B，完成矩阵乘法之后的结果是一个M×N的矩阵C，即：

$$C := \alpha \cdot op(A) \cdot op(B) + \beta \cdot C \quad (1)$$

其中，op(X)指定矩阵X为矩阵是常规、转置还是共轭；A，B，C为复数矩阵，$\alpha$ 和 $\beta$ 为复数标量。Fortran调用接口为：

SUBROUTINE ZGEMM（TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC）

TRANSA, TRANSB：字符变量，代表是否要转置矩阵，默认为“N”，表示不转置；若为“T”则表示转置输入的矩阵；这里的A、B、C为全局矩阵如公式（1）中A、B、C的一部分，由于为Fortran函数，A、B、C按列存储；ALPHA和BETA对应于公式（1）中的 $\alpha$ 和 $\beta$ ；LDA，LDB和LDC分别描述局部矩阵A、B、C的大小和位置信息。

## 3. ScaLAPACK

### 3.1 ScaLAPACK简介

ScaLAPACK[7][8](Scalable Linear Algebra PACKage, 可扩展线性代数计算软件包)是美国能源部DOE2000支持开发的20多个ACTS工具箱之一，是LAPACK在分布式存储环境中的扩展，主要运行在基于分布式存储和消息传递机制的MIMD计算机及支持MPI或PVM的集群上。LAPACK是适用于向量超级计算机、共享式存储并行计算机和各种单机上的线性代数运算程序包，ScaLAPACK实现了其功能的一个子集。

ScaLAPACK可以求解线性方程组、线性最小二乘问题、特征值和奇异值及其相关问题。它只适用于稠密矩阵和带状矩阵，不适用于普通的稀疏矩阵。与LAPACK类似，ScaLAPACK也是基于块划分算法以减少进程间的通信。

ScaLAPACK的基本组成部分是PBLAS和BLACS。PBLAS是BLAS的分布式存储版本，实现Level 1 到Level 3的函数运算；BLACS（Basic Linear Algebra Communication Subprograms）实现并行线性代数计算中常用的通信。ScaLAPACK的软件组成见图1。

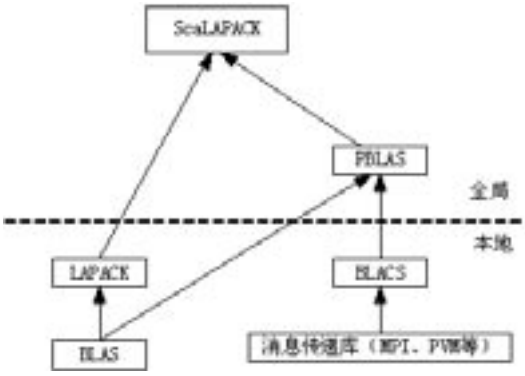


图1 ScaLAPACK软件组成

ScaLAPACK支持4种数据类型如S、D、C、Z（见BLAS部分介绍）的计算程序，包括了求解各种类型线性代数方程组和特征值问题的驱动程序、执行不同计算任务的计算程序以及执行某些子任务或者一般底层计算任务的辅助程序。每个驱动程序一般调用一系列计算程序，计算程序则又调用一系列的辅助程序。驱动程序主要用于测试ScaLAPACK库程序的正确性和效率，计算程序和辅助程序供各类数值分析人员或软件开发者使用。

ScaLAPACK使用基于SPMD模型的Fortran77编程，采用显示消息传递方式进行通信。PBLAS和BLACS使用C语言编程，但提供Fortran77接口。

Intel MKL包含了ScaLAPACK针对Intel平台和编译器优化过的版本。

### 3.2 PZGEMM

PZGEMM和ZGEMM类似，仅多了描述矩阵的参数。调用PZGEMM来计算公式（1）中全局矩阵乘A×B时并不需要输入全局矩阵，而只要输入全局矩阵分配在每个进程的局部矩阵。当把全局矩阵按进程数 $p \times q$ 进行分解的时候并不是直接划分为 $p \times q$ 个局部矩阵，而是按照矩阵块大小依次划分给各个进程。

Fortran调用接口为：

PZGEMM（TRANSA, TRANSB, M, N, K, ALPHA, A, IA, JA, DESCA, B, IB, JB, DESCB, BETA, C, IC, JC, DESCC）

其中，IA，JA为关于从全局矩阵A的描述，指明进行矩阵运算的起始行列位置；DESCA，描述局部矩阵A的大小和位置信息，可通过函数descinit生成，是包含了9个元素的整型数组。数组DESCA各元素描述见表1。关于矩阵B和矩阵C的描述DESCB和DESCC类似与DESCA。

表1 数组DESCA元素描述

| DESCA | 说明 |
|---|---|
| 1 | 描述符，固定为1 |
| 2 | BLACS上下文（进程空间），scaLAPACK可调用此空间的所有进程 |
| 3 | 全局矩阵的行数，即M |
| 4 | 全局矩阵的列数，即K |
| 5 | 矩阵块行数，及MB |
| 6 | 矩阵块列数，及NB |
| 7 | 全局矩阵第一个元素所在进程在进程行列空间 $p \times q$ 的第几行，一般为0 |
| 8 | 全局矩阵第一个元素所在进程在进程行列空间 $p \times q$ 的第几列，一般为0 |
| 9 | 局部矩阵A的行数，可通过函数numroc得到 |

### 3.3 ScaLAPACK编程

在程序中调用ScaLAPACK程序需要4个基本步骤：

（1）初始化进程网格

程序运行时指定运行的进程数P，P可表示为一维数组。在ScaLAPACK中，将进程的一维数组隐射到二维网格中，即进程网格。在使用ScaLAPACK时，首先要初始化进程网格，得到默认的系统上下文。代码如下：

```
CALL BLACS_GET ( -1, 0, ICTXT )
CALL BLACS_GRIDINIT ( ICTXT, ‘ Row-major ’,NPROW，NPCOL )
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
```

其中，BLACS_GET获得上下文ICTXT；BLACS_GRIDINIT初始化进程网格，Row-major表示网格按行优先的次序排列；BLACS_GRIDINFO获得本进程在进程网格中的位置信息( MYROW, MYCOL )。

（2）将矩阵分布到进程网格上

在分布数据之前，每个要分布在进程网格上的全局矩阵都必须分配一个数组描述符，通过DESCINIT 程序进行初始化，如：

```
CALL DESCINIT ( DESCA, M, K, MB, NB, RSRC, CSRC,ICTXT,
MXLLDA, INFO)
CALL DESCINIT ( DESCB, K, NRHS, NB, NBRHS, RSRC, CSRC,
ICTXT, MXLLDB, INFO)
CALL DESCINIT ( DESCC, M, NRHS, NB, NB, 0, 0, ICTXT,
MAX(1, P), INFO )
```

然后，再去矩阵A、B进行初始化，如果从数据文件中读入矩阵数据并将数据分布在进程网格上，可通过调用PDLAREAD函数，如给矩阵A初始化：

```
CALL PDLAREAD ( ‘ input.dat ’, MEM(IPA), DESCA, 0, 0, MEM(IPW) )
```

（3）调用ScaLAPACK程序

如3.2节中的方式调用PZGEMM完成复数矩阵乘运算。

（4）释放进程网格

当所有计算完成时，需释放所有进程网格，代码为：

```
CALL BLACS_GRIDEXIT ( ICTXT )
CALL BLACS_EXIT ( 0 )
```

## 4. 并行复数矩阵乘测试及分析

### 4.1 测试环境

并行复数矩阵乘测试基于“魔方”超级计算机，系统为HPP体系架构，共1920个16个计算核心的刀片节点，每个节点通过InfiniBand高速网络互联。每个节点由4路Quad-core AMD Opteron 8347@1.9GHZ构成，64GB内存，理论计算峰值性能达到128GFlops/Node。魔方的计算节点分布在38个rack中，每个rack包含有5箱刀片（每箱10个刀片），每箱刀片内置有InfiniBand交换模块和千兆以太网交换模块。

测试使用的软件环境如表2。

表2 测试所用软件环境

| | |
|---|---|
| Operating System | SUSE Linux Enterprise Server 10 (x86_64) sp2,kernel 2.6.16.60-0.69.1-smp |
| Compiler | GCC 4.1.2, Intel C++ compiler 11.1, Intel Fortran compiler 11.1 |
| MPI lib | OpenMPI 1.4.1(compiled by Intel C++ 11.1, Intel Fortran 11.1) |
| BLAS lib | Intel mkl 10.2.1, OpenBLAS 0.2.5, |
| LAPACK | Version 3.4.2(including BLAS) |
| Scalapack | Version 2.0.2(compiled by OpenMPI 1.4.1) |

### 4.2 结果及分析

魔方环境下串行复数矩阵乘ZGEMM性能，比较标准BLAS、MKL和OpenBLAS。从测试结果来看，OpenBLAS的性能明显高于MKL，高出约30%，而是标准BLAS库性能的9倍多。

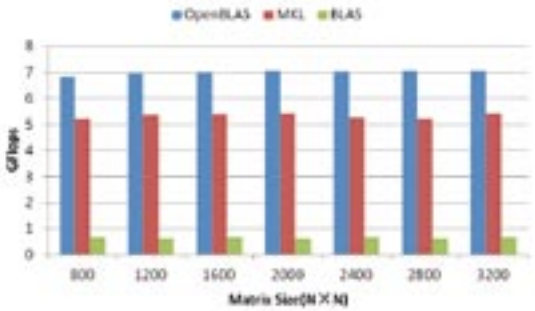


图2 ZGEMM性能

并行复数矩阵乘PZGEMM性能，比较ScaLAPACK分别调用标准BLAS、OpenBLAS及MKL的性能。从图3可以看出，在运行进程数为256核时，ScaLAPACK+OpenBLAS实现PZGEMM的性能最好，而且随着矩阵维度的增加，ScaLAPACK调用OpenBLAS性能呈现上升趋势，在N=75,600时，MKL性能呈现下降趋势；ScaLAPACK调用标准BLAS库，性能持续下降，在N=49,200时，性能基本维持在一个水平。

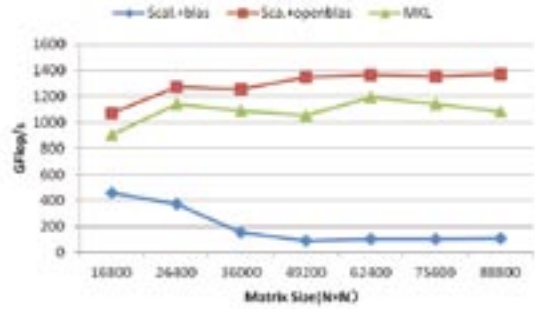


图3 NP=256核PZGEMM性能

图4给出了随着运行进程数的增加，ScaLAPACK+OpenBLAS相对ScaLAPACK+BLAS和MKL实现PZGEMM的加速性能。很显然，处理器规模越大，ScaLAPACK调用OpenBLAS相对调用BLAS的加速越明显，而相对MKL的加速性能则比较稳定。

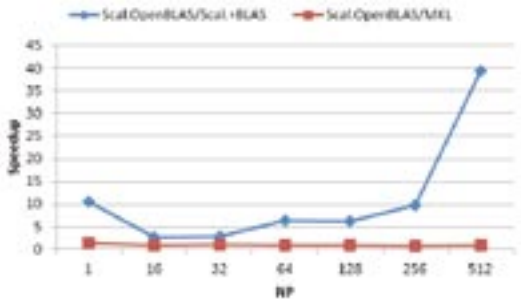


图4 PZGEMM调用不同BLAS库的加速性能

从以上数据来看，PZGEMM在上述3种数学库实现方式下，ScaLAPACK+OpenBLAS性能最好。

## 5. 结论

由于复数矩阵乘在很多领域中得到应用，本文研究了在大规模并行计算环境下“魔方”系统平台下，不同数学库实现串行、并行复数矩阵乘的性能表现。选择了NetLib的标准BLAS库、Intel的MKL及基于GotoBLAS开发的OpenBLAS，比较其串行复数矩阵乘ZGEMM性能，OpenBLAS能获得高于MKL30%、标准BLAS 多于9倍的性能。同时，选择ScaLAPACK调用标准BLAS、OpenBLAS及Intel MKL数学库，比较其并行复数矩阵乘PZGEMM在不同问题规模、不同处理器核心数下的性能，结果表明，ScaLAPACK调用OpenBLAS实现PZGEMM性能最优，且在256核心下，随着矩阵维度N的增加，此种结合方式相比调用标准BLAS和MKL表现出更好性能递增趋势。此外，随着处理器核心数的增加，MKL与ScaLAPACK+OpenBLAS的性能差距基本不变，而ScaLAPACK调用OpenBLAS、标准BLAS的性能差异则逐渐增大。

该项工作对大规模并行环境下并行复数矩阵乘的应用及性能进行了初探，但处理器技术的发展日新月异，在不同处理器平台下并行复数矩阵乘的应用及性能表现将是今后我们要研究的内容。而且，随着加速部件提高高性能计算系统整体性能上的作用越来越受重视，研究CPU+加速部件（如GPGPU、MIC）异构平台下并行复数矩阵乘的应用及性能将成为今后这部分工作的重点。

参考文献：

[1] blas, http://www.netlib.org/blas/

[2] openblas, http://xianyi.github.com/OpenBLAS/

[3] mkl, http://software.intel.com/en-us/intel-mkl

[4] 陈少虎,张云泉,张先佚,程豪. BLAS库在多核处理器上的性能测试与分析. 软件学报 , Vol.21, Supplement, December 2010, 21(zk):214-223

[5] cuBLAS, http://cudazone.nvidia.cn/cublas/,

[6] MAGMA, http://icl.cs.utk.edu/magma/software/index.html