

# 文献综述

## 一、背景介绍

近年来，在工程应用中，求解高阶矩阵的需求日益增长，全矩阵运算脱离了实际的硬件限制，为了满足这一日益增长的需求，同时这些矩阵通常都有着一个特征——非零元远少于零元，稀疏矩阵这门学科便应运而生。在 20 世纪 60 年代研发电子网络的电子工程师们是最早的去利用稀疏性来应用稀疏矩阵进行工程上的计算的。[1] 而在微分方程数值解、线性规划等的有限元分析中，经常出现求解高阶稀疏线性方程组，如利用全矩阵进行存储，则需要  $n^2$  的空间复杂度和  $n^3$  的乘法运算时间复杂度，显然，这种程度的运算量是无法被微型计算机，甚至是工作站所接受的。而利用矩阵的稀疏性，可以有效地减小消耗很多无谓的存储空间以及无谓的计算，在很大的程度上降低了时间和空间复杂度，降低了计算对硬件的需求，使计算成为可能。

## 二、国内外研究现状

### 1、研究方向及进展

在这些年的发展中，出现了很多的存储方法，比如：对角线存储法、对称矩阵的变带宽存储法、坐标存储法、Elipack-Itpack 存储法、CSR 存储法、Shermans 存储法、超矩阵存储法、动态存储方案等 [2]。

如行压缩存储方式 (Compressed Row Storage): CRS 存储可以高效地存取任意一行非零元素，但存取任意一列非零元则需要遍历整个 CRS 存储结构。相应地，与 CRS 存储的稀疏矩阵相关的算法要高效的编程实现，算法的计算顺序必须按行来进行。[3] 下面对应于 CRS 存储：

ROWP	0	2	5	8	9	12						
COLIND	0	1	0	2	4	1	2	3	2	1	2	4
VAL	a	b	c	d	e	f	g	h	i	j	k	l

我们可以发现，ROWP 数组存储的时行非零元的增长量。COLIND 则存储的是列索引值，典型的 C 语言结构实现为

```
struct cr_matrix{
```

```

    int *Ri; /* col index */
    int *Rp; /* length nrow+1 */
    double *Rx;
    int Rncol;
    int Rnrow;
};

```

在早期计算机时，串行的算法占到主流，自然，稀疏矩阵的存储方式以及稀疏矩阵的计算都是为了串行算法服务的。但是随着计算机的发展，计算机集群、多核 CPU、GPU 并行等的出现，让并行算法在计算时间上远远地超越了串行算法，为了更好的适应并行计算中的稀疏矩阵计算，出现了许多新的算法以及存储方式。例如对于非结构化的矩阵，基于 CUDA 框架下的 SCOO 形式的 SpMV 算法比利用基于 Cusp 库的 SCOO 具有更高的效率。[4]

而在做稀疏矩阵的计算时，通常都是做一系列的基本运算，如：矩阵转置、矩阵向量乘法、矩阵矩阵乘法、数乘等。为了能得到更好的效率，许多研究者致力于寻找对于这些计算最优的存储结构及计算算法，同时提供了许多类库供科学计算使用，如：Portable, Extensible Toolkit for Scientific Computation (PETSc)、Boost、GNU Scientific Library (GSL) 等。

例如，在 Boost-uBLAS 中有着稀疏矩阵的模板 `mapped_matrix<T, F, A>`（元素映射矩阵存储形式）、`compressed_matrix<T, F, IB, IA, TA>`（压缩存储格式）、`coordinated_matrix<T, F, IB, IA, TA>`（坐标存储格式）。分别有着如下示例：[5]

**mapped\_matrix:**

```

#include <boost/numeric/ublas/matrix_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    mapped_matrix<double> m (3, 3, 3 * 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;
    std::cout << m << std::endl;
}

```

**compressed\_matrix:**

```

#include <boost/numeric/ublas/matrix_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    compressed_matrix<double> m (3, 3, 3 * 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)

```

```

        m (i, j) = 3 * i + j;
    std::cout << m << std::endl;
}

coordinate_matrix:

#include <boost/numeric/ublas/matrix_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    coordinate_matrix<double> m (3, 3, 3 * 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;
    std::cout << m << std::endl;
}

```

在多年的研究中，各种相关的研究成果层出不穷，例如，根据 Michele Martone 的研究成果，在对称矩阵乘法，或者转置的矩阵向量乘法中，RSB 格式的迭代算法效率是比较高的 [6]。而在李佳佳, 张秀霞, 谭光明, 陈明宇的研究中，得到了影响矩阵性能的参数集，可以利用该参数集提取矩阵特征，并输出最优存储格式，供数值解法器和上层应用调用。[7] 为了适用于大数据集计算以及克服现有稀疏矩阵乘法算法低效的问题，郑建华，朱蓉，沈玉利提出了一种基于向量线性组合（VLC）的矩阵乘法处理模式，同时采用 MapReduce 计算模型实现了基于 VLC 模式的并行矩阵乘法算法。[8] 而在集群计算方面，负载平衡是不能不提的，为了更好地发挥集群计算的计算能力，付朝江博士给出了基于贪婪分配的稀疏矩阵与向量乘的负载平衡的解决方案。[9]

## 2、存在问题

稀疏矩阵的存储没有一种通用的形式，各种形式都各有利弊 [10]，这也就需要我们对于不同的情况去寻找最合适的存储形式来计算，虽然李佳佳, 张秀霞, 谭光明, 陈明宇 [7] 的研究得到了一个参数集用以寻找最优存储形式，但是比较的存储形式比较少，还有多种形式没有计入考量范围。

而在超大规模的稀疏矩阵计算时，计算机的内存可能不能完全存储这些数据，部分数据将会存储在硬盘中，而硬盘的读取效率远低于内存，如何更好地将硬盘中的数据读取到内存中，减少内存与硬盘交互的时间，同时，如何降低内存与缓存间的交互时间也是一个问题。

而随着计算机的普及，计算机进入了家家户户，分布式计算逐渐成为可能，让普通的计算机使用者参与到科学计算中，贡献 GPU 资源来帮助研究人员更快地完成计算，但是这尚未得到普及，同时，中国尚未形成一个成熟的分布式计算平台，也没有培养出普通用户参与到分布式计算项目的习惯。

## 3、研究展望

为了更好的适应现代的工程计算需求，计算更大规模的稀疏矩阵，以及适应新

的 CPU 指令集和 GPU 计算框架,稀疏矩阵的存储形式值得进一步的研究。同时,伴随着分布式计算的发展,更为高效的冗余计算机制和任务分配机制也将带给稀疏矩阵计算新的研究方向。按照摩尔定律,计算机的计算能力将每隔 18-24 个月翻一番,那么计算稀疏矩阵的能力也将相应提高。同时随着量子计算机、光计算机等的出现,更高德计算能力也将随之到来。而为了更好地发挥这些硬件的能力,以往的算法可能并不再适合了,需要开发新的算法以适应新的硬件。而对于硬件商而言,如 Intel、AMD 等 CPU 厂商而言,封装更多的 CPU 指令,让用户可以直接调用 CPU 指令来进行科学计算,更好地发挥硬件的性能。对于普通计算机用户,培养参与到分布式计算的习惯,通过建立一个国家级的分布式计算中心,建立奖励机制,鼓励普通计算机用户参与到分布式计算中,为研究计划中的稀疏矩阵计算贡献自己的计算资源。

### 三、参考文献

- [1]Yousef Saad.Iterative Methods for Sparse Linear Systems[M].SECOND EDITION.USA:Society for Industrial and Applied Mathematics,2003 年.68.
- [2] 张永杰、孙秦. 稀疏矩阵存储技术 [J]. 长春理工大学学报,2006 年,03 期:38-41.
- [3] 冯广祥. 大型稀疏矩阵直接求解算法的研究及实现 [8]. 东北大学: 系统工程, 2010.
- [4]Hoang-Vu Dang, Bertil Schmidt.CUDA-enabled Sparse Matrix-Vector Multiplication on GPUs using atomic operations[J].Parallel Computing, 2013,Vol.39 (11):737-750.
- [5]Joerg Walter and Mathias Koch.Boost 1.57.0 Library Documentation-uBLAS. [http://www.boost.org/doc/libs/1\\_57\\_0/libs/numeric/ublas/doc/index.html](http://www.boost.org/doc/libs/1_57_0/libs/numeric/ublas/doc/index.html),2011
- [6]Michele Martone.Efficient multithreaded untransposed, transposed or symmetric sparse matrix-vector multiplication with the Recursive Sparse Blocks format[J].Parallel Computing, 2014,40:47-58.
- [7] 李佳佳, 张秀霞, 谭光明, 陈明宇. 选择稀疏矩阵乘法最优存储格式的研究 [J]. 计算机研究与发展, Journal of Computer Research and Developmen,2014 年,04 期:882-894.
- [8] 郑建华, 朱蓉, 沈玉利.Sparse matrix multiplication algorithm based on Map Reduce[J]. 仲恺农业工程学院学报,2013 年,03 期:45-50.
- [9] 付朝江. 基于贪婪分配的稀疏矩阵与向量乘的负载平衡 [J]. 福建工程学院学报,2010 年,01 期:79-82.
- [10] 张永杰,孙秦. 稀疏矩阵存储技术 [J]. 长春理工大学学报,2006 年,03 期:38-41.