Spark 平台在单光子成像测量矩阵生成与评估中的应用

王兴达[1,2]　刘雪峰[1]

[1](中国科学院国家空间科学中心　北京 100190)

[2](中国科学院大学　北京 100049)

摘　要　针对单光子成像实验中测量矩阵在高分辨率条件下所面临的生成时间瓶颈问题,提出基于 Spark 平台的测量矩阵的生成和评估算法。在三种高分辨率条件下,进行测量矩阵生成与评估实验。实验表明,使用 Spark 平台进行测量矩阵的生成,大幅缩短了生成时间,测量矩阵的生成质量与单机版本相近,且可拓展性良好,大大降低了单光子成像实验的时间成本。

关键词　Spark　测量矩阵　单光子成像　压缩感知　分布式计算

中图分类号　TP3　　**文献标识码**　A　　DOI:10.3969/j.issn.1000-386x.2019.08.010

APPLICATION OF SPARK IN GENERATION AND EVALUATION OF MEASUREMENT MATRIX FOR SINGLE PHOTON IMAGING

Wang Xingda[1,2]　Liu Xuefeng[1]

[1](*National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China*)

[2](*University of Chinese Academy of Sciences, Beijing 100049, China*)

Abstract　Aiming at the time bottleneck of measurement matrix generation in single photon imaging experiments under high resolution conditions, we proposed a method of generating and evaluating measurement matrix based on Spark platform. In this paper, the experiment of measuring matrix generation and evaluation was carried out under three high resolution conditions. Experiments show that using Spark platform to generate measurement matrix greatly shortens the generation time. The quality of measurement matrix generation is similar to that of single-machine version, and has good extensibility, which greatly reduces the time cost of single-photon imaging experiment.

Keywords　Spark　Measurement matrix　Single photon imaging　Compressed sensing　Distributed computing

0　引　言

近年来,压缩感知(CS)理论得到了越来越多的关注。单光子成像是利用压缩感知理论的一个典型应用。原理是先将物体成像在数字微镜阵列上,利用测量矩阵的调制作用和单光子探测器的测量值,就能够在低采样率条件下利用单点探测器取得良好的成像效果[1]。在实验中起调制作用的测量矩阵是非常重要的一环,通常在理想的单光子成像实验中需要 30% ~ 50% 采样率的测量矩阵才能够恢复出效果比较好的图像[2]。对于高分辨率的单光子成像实验,所需测量矩阵数量可达百万级别。普通计算机利用 MATLAB 等软件生成一个 FHD(1 920 × 1 080)分辨率的全采样率测量矩阵通常需要 20 小时以上的时间,大大降低单光子成像实验的效率,成为单光子成像实验中的一个痛点[3]。

近几年随着以 Apache Spark 为代表的分布式计算技术的发展为解决这一痛点带来了曙光。Spark 平台本身是一个分布式的计算框架,通常用于大规模的数据处理[4],本文通过 Spark 中的 parallel 算子生成简单 RDD,然后通过对该 RDD 的 MapPartition 操作能够将测量矩阵生成和评估任务均匀分配到各个机器节点的 Executor 上进行计算作业,利用 Spark 平台调度集群的

收稿日期:2019 - 01 - 03。国家自然科学基金面上项目(61474123);国家重大科学仪器专项(2013YQ030595)。**王兴达**,硕士生,主研领域:分布式计算。**刘雪峰**,副研究员。

计算性能。实验证明该方法具有良好的加速比和拓展性,满足了单光子成像实验中对于测量矩阵的频繁生成需求,大幅降低了成像实验的时间成本。

1 单光子成像与 Spark 平台

1.1 压缩感知理论

压缩感知(Compressed sensing),也被称为压缩采样(Compressive sampling)或稀疏采样(Sparse sampling),是一种寻找欠定线性系统的稀疏解的技术[5]。压缩感知理论指出:只要信号是可压缩的或在某个变换域是稀疏的[6],那么就可以用一个与变换基不相关的观测矩阵将变换所得高维信号投影到一个低维空间上[7],然后通过求解一个优化问题就可以从这些少量的投影中以高概率重构出原信号,可以证明这样的投影包含了重构信号的足够信息[8]。在过去的几年内,压缩感知作为一个新的采样理论,它可以在远小于 Nyquist 采样率的条件下获取信号的离散样本,保证信号的无失真重建[9]。

1.2 单光子成像

单光子成像是压缩感知理论的一个典型应用。借鉴计算成像中的高通量测量的思想,如图 1 所示,首先用光源打在物体上,通过成像透镜将物体的像成在可调制的数字微镜阵列(DMD)上,再利用测量矩阵对 DMD 进行 0 - 1 的掩膜调制,每个测量矩阵都随机调制图像的部分像素点,再用一个收集透镜将这些像素点的光路聚集到一个探测点上,利用单光子探测技术对其进行测量,得到光子计数测量值[10]。最后,通过 TVAL3 等压缩感知恢复算法及测量矩阵和测量值就能进行图像恢复,这样就能够利用单点探测器取得良好的成像效果[11]。根据压缩感知理论的成像经验,每次成像实验大约需要 30% ~50% 的采样率才能取得比较好的成像效果。测量矩阵的数目与采样率的关系如下:

$$matrix\_num = width \times height \times sample\_rate$$

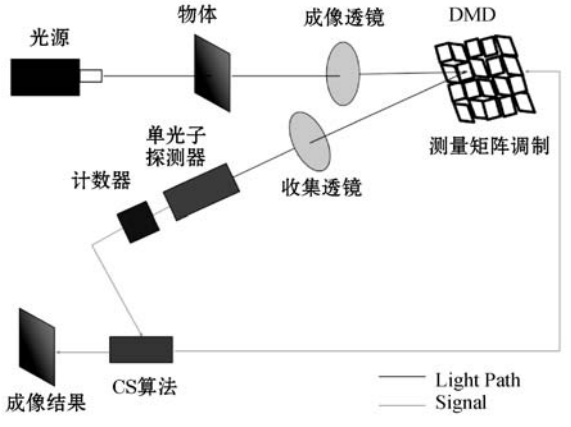

图 1 单光子成像原理图

在高分辨率的情况下,所需的测量矩阵数目往往是百万级别的,普通计算机利用 MATLAB 生成一次 FHD 分辨率的测量矩阵大约需要 20 小时,非常影响单光子成像实验的效率。

1.3 Spark 平台介绍

Spark 是一个分布式计算框架,专为大规模数据处理而设计,具有快速且通用的特点[12]。Spark 原本是加州大学伯克利分校的 AMP 实验室开源的类似于 Hadoop MapReduce 的通用并行框架,它拥有 Hadoop MapReduce 所具有的优点,但不同于 MapReduce 的是 Spark 中间的输出结果可以保存在内存中,从而不再需要读写 HDFS,因此能够获得比 MapReduce 更快的大数据处理速度[13]。图 2 展示了 Spark 的基本框架。用户通过 Driver 向 Master 注册申请资源,由 Master 通过各个 worker 的心跳包获取集群当前的资源状况,找到合适的资源给本任务,然后在各个 worker 上启动 Executor 进程去运行用户提交任务中的每一个 task,Executor 向 Driver 通过心跳包报告运行状态,直到作业完成。分布式计算技术的瓶颈往往在与集群的网络 IO,想提高计算性能应该尽可能地避免 shuffle 的过程出现,而测量矩阵的生成和校验是非常独立的过程,不需要与其他的节点进行通信或数据交互[14],因此非常适合于 spark 这种分布式计算框架,能够非常充分地利用集群机器的计算性能快速生成和评估测量矩阵。

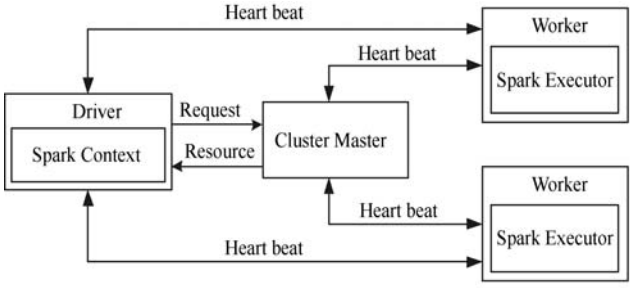

图 2 Spark 框架的工作流程

2 分布式测量矩阵生成算法

2.1 算法设计

测量矩阵生成分为矩阵生成,矩阵评估和数据取回三个阶段。通过 spark-submit 脚本中的参数并配置本次任务的矩阵类型、分辨率和采样率等。Spark 的计算框架通常用于分布式的处理数据,如果想使用该框架进行数据生成可以通过 Spark 中内置的 parallel 算子生成简单 RDD,然后对 RDD 使用 Spark 中的 MapPartition 算子,MapPartition 算子是 Map 算子的一个变种,虽然它们都可进行分区的并行处理,但两者的主要

区别是调用的粒度不一样:Map 的输入变换函数是应用于 RDD 中每个元素,而 MapPartition 的输入函数是应用于每个分区[15]。通过 Spark 中的调度系统将本次矩阵生成与评估任务均匀地分布在各个节点上,在每个节点根据配置的参数进行测量矩阵的生成作业。对于每个生成的测量矩阵需要在本地进行矩阵质量的评估,以随机测量矩阵为例,一般需要做计算矩阵的随机偏差,在偏差允许范围内的矩阵才会被认为是合格的矩阵而保留下来。对于不合格的矩阵,则会进行丢弃然后重新生成一个新的测量矩阵,为了进行对比实验,在进行生成与评估作业中会进行数据统计(包括生成矩阵数目、矩阵质量和矩阵保留个数等)。因为测量矩阵本身是 0 -1 矩阵,为节省存储空间并没有将结果写入 HDFS 中,而是通过 Spark 中的 collect 算子将已生成好的测量矩阵和统计数据拉回提交作业的 Driver 端,通过二进制文件的方式写入本地,供后续进行单光子成像实验使用。图 3 展示了一次完整的测量矩阵的生成和评估作业的流程。

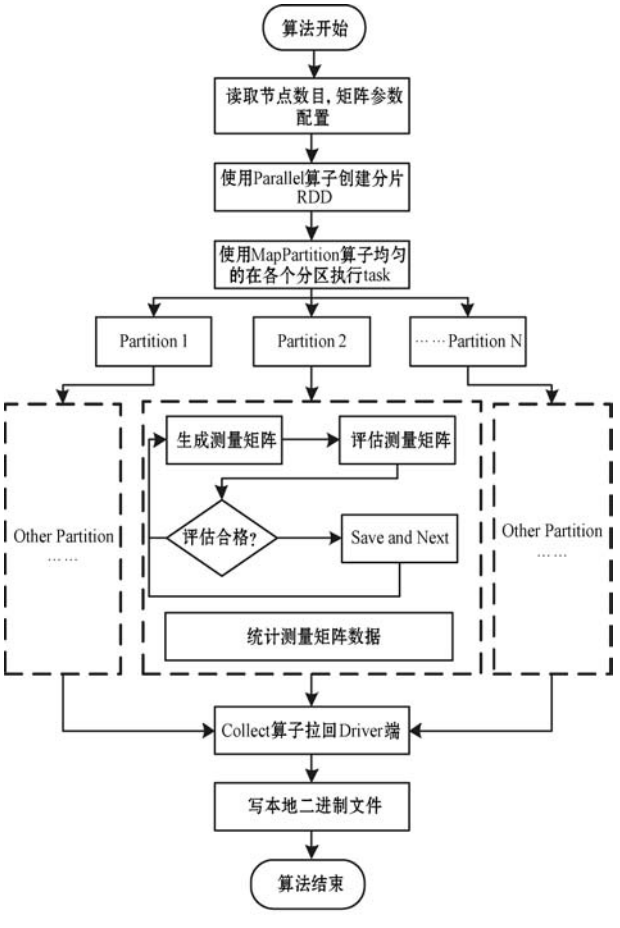

图 3　算法流程图

2.2　工程化实现

工程化实现过程主要是利用 Spark 中的编程模型,主要用到的算子包括 Spark 中的 mapPartition、parallel、collect 等,其算法流程的伪代码如算法 1 所示。

算法 1　测量矩阵生成

```
输入:矩阵大小,节点数,矩阵数目
输出:经过评估合格的测量矩阵二进制文件
1. val sc = new SparkContext( )
2. val config = sc. getConf. get( )
3. val partitionNum = sc. getConf. get( )
4. val matrixNumPerPartition = matrixNum / partitionNum
5. val matrix = sc. parallelize(0 until
       partition,partition). mapPartitions {part = >
         for(0 to matrixNumPerPartition){
           generate matrixes
             if(evaluate matrixes is qualified)
               save
             qualified matrix count ++
           else
             regenerate
         unqualified matrix count ++
       }
           Calculate matrix save rate
     }
6. val result = matrix. collect( )
7. val resultFile = new FileOutputStream( )
8. resultFile. write(result)
```

3　实验和结果分析

3.1　实验环境

由于实验室单光子成像实验的需要,在实验室搭建了 8 台服务器搭建的物理集群,并且安装配置好了 Spark 环境。为了便于进行对比实验,单机实验采用的机器环境和集群中单节点的硬件完全相同,表 1 中列出了单个节点机器的物理硬件配置,表 2 中列出了集群机器的主要软件环境版本信息。

表 1　Spark 集群单节点硬件配置信息

处理器	Intel XeonE5 - 2630
CPU 核心数	8
CPU 频率/Ghz	2.6
硬盘/GB	1 024
内存/GB	16

表 2　Spark 集群单节点软件配置信息

软件名	版本号
操作系统 OS	Centos 7.5.0
Spark 版本	2.3.0
Hadoop 版本	2.7.1
Scala 版本	2.11.0
Java 版本	1.7.0

3.2 评估指标

本文的实验选择了单机、2 节点、4 节点、8 节点等 4 组进行对比的横向对比，选择了三种不同的分辨率进行纵向对比，测量矩阵选择了相对简单的随机测量矩阵。主要选取运行时间、矩阵评估存留率、加速比和拓展性作为评价指标。各评价指标说明如下：

（1）运行时间（Runtime）：该指标是本次实验的主要目的指标，运行时间指从提交作业成功到测量矩阵的二进制文件写入硬盘完毕所需的总时间。

（2）矩阵评估留存率（Matrix Save Rate）：该指标是衡量矩阵生成质量的指标，其值定义为：

$$rate = \frac{save\_num}{total\_num}$$

（3）加速比（Speedup）：该指标主要是用来验证并行算法的效率，根据阿姆达尔定律，其值在理想的状态下通常被定义为：

$$speedup = \frac{base\_time}{parallel\_time}$$

（4）可拓展性（Extension）：该指标主要反映出集群节点的变化对该并行算法的性能影响大小，可以通过此指标预判集群数目与该算法的效率匹配性。

3.3 实验分析

为了验证以上的实验指标，分别选择分辨率在 512×512、1 024×768、1 920×1 080 下进行标准随机测量矩阵算法生成进行实验。

实验 1 本实验对比了不同节点数 N 对于生成随机测量矩阵的运行时间（单位：min），实验数据如表 3 所示。

表 3 运行时间实验结果

分辨率	单机	$N=2$	$N=4$	$N=8$
512×512	42.3	23.8	12.6	6.9
1 024×768	235.1	145.1	77.3	45.1
1 920×1 080	1 203.7	761.8	416.6	204.9

从表中的实验数据的对比来看，可以看出随着节点数的增加，相同分辨率下测量矩阵的生成时间在减小，大致呈现线性关系；测量矩阵的生成时间随着分辨率的提高而大幅度增加，呈现倍数关系，基本符合预期。

实验 2 矩阵评估存留率是指生成的测量矩阵经过评估算法之后合格的比例，实验以常用的随机测量矩阵为例，高分辨情况下选择随机偏差小于 0.001 的矩阵为合格矩阵，保存到最后的测量矩阵结果中，0 -1 随机测量矩阵的随机偏差通常用以下公式计算：

$$random\_rate = \frac{abs(pos\_num - neg\_num)}{pos\_num + neg\_num}$$

式中：pos_num 是矩阵中正元素的个数，neg_num 是矩阵中非正元素的个数。随机测量矩阵的随机偏差一般会随着测量矩阵分辨率的提高而降低。

矩阵评估存留率实验结果如表 4 所示。

表 4 矩阵评估存留率实验结果

分辨率	单机	$N=2$	$N=4$	$N=8$
512×512	0.394	0.382	0.393	0.388
1 024×768	0.625	0.626	0.619	0.625
1 920×1 080	0.852	0.851	0.853	0.847

可以看出，随着节点数的增加，矩阵评估保留率一直表现比较稳定，没有显著有趋势性的变化，说明利用 Spark 进行分布式的测量矩阵生成依然能够获得与单机版测量矩阵生成任务质量相近的矩阵；随着测量矩阵分辨率的提高，矩阵评估存留率会逐渐提高，且与单机版实验的指标保持一致。

实验 3 计算不同节点数目下的加速比关系绘制折线图，如图 4 所示。

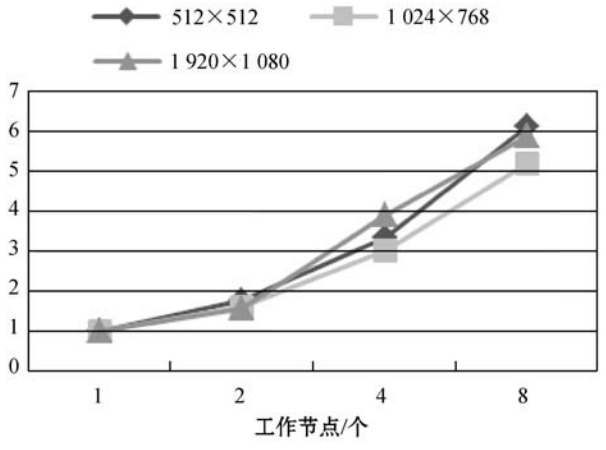

图 4 不同节点数下的加速比

从图中可以看出，利用 Spark 框架生成测量矩阵的过程中，算法的加速比基本是接近线性的，随着分辨率的增加，数据规模成倍的增加，加速比也没有因为数据分散后，最后从各个节点收集数据的时间增加而出现明显下降。这是因为 Spark 这种运行在 JVM 上的平台，随着数据在各个节点上的分散，单个节点所需的内存按比例下降，JVM 中消耗在内存回收（GC）的时间也会逐渐减少了。说明该算法有非常良好的加速比，可以期待该算法在未来高分辨测量矩阵生成中也具有非常好的表现。

实验 4 可拓展性分析。根据实验 1 的数据将集群使用的节点数和作业的运行时间作折线图，如图 5 所示。

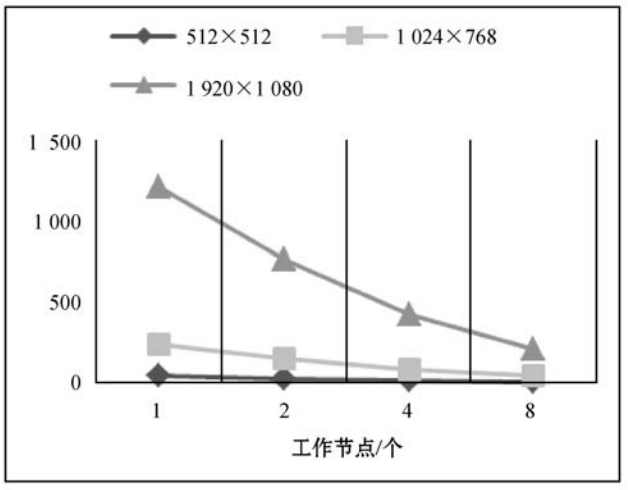

图 5　扩展性实验结果图

由图 5 可以看出,在以上三个分辨率的测量矩阵生成实验中,运行时间随着节点数增加明显下降,基本呈现线性关系。实验说明,该分布式测量矩阵生成算法具有良好的可扩展性。

4　结　语

传统利用 MATLAB 等软件工具生成测量矩阵的算法在面对高分辨率或高采样率等数据规模较大的情况时,完整生成一次测量矩阵所需的时间成本往往非常巨大,严重影响了成像实验的效率。通过 Spark 分布式计算框架可以为测量矩阵的生成和评估提供可靠、简单、有效的解决方案,大大缩短了测量矩阵的生成时间,为调试测量矩阵和成像实验提供了有力支撑。实验表明,利用 Spark 平台通过分布式的生成算法可以近似线性的缩减生成时间,测量矩阵质量基本与单机算法相同,并且可拓展性良好,在可预见的情况下能够支撑更大分辨率的单光子成像实验。

参　考　文　献

[1] 俞文凯. 压缩感知在超灵敏时间分辨成像光谱中的应用[D]. 北京:中国科学院研究生院(空间科学与应用研究中心),2015.

[2] 孙宪坤,陈涛,韩华,等. 基于 Kent 混沌测量矩阵的压缩感知图像重构算法[J]. 计算机应用与软件,2017,34(4):213 - 220.

[3] 陈晓锋. 基于分段线性坐标优化的退化压缩感知水印检测[J]. 计算机应用与软件,2018,35(12):313 - 319.

[4] 黎文阳. 大数据处理模型 Apache Spark 研究[J]. 现代计算机(专业版),2015(8):55 - 60.

[5] 李威,朱岱寅,胡晓琛. 一种 SAR 极坐标格式成像算法的 FPGA 实现[J]. 计算机应用与软件,2018,35(7):256 - 262.

[6] 戴琼海,付长军,季向阳. 压缩感知研究[J]. 计算机学报,2011,34(3):3425 - 3434.

[7] Masahiro H, Yoshinori N O, Yasuo Y, et al. Accurate estimation of energy dependence of quasiparticle interference using compressed sensing[C]// Meeting Abstracts of the Physical Society of Japan,2016.

[8] Ferdian R, Hou Y F, Okada M. Compressed Sensing Based Channel Estimation in ISDB-T System without Cyclic Prefix[R]. ITE Technical Report,2016.

[9] 李珅,马彩文,李艳,等. 压缩感知重构算法综述[J]. 红外与激光工程,2013,42(S1):225 - 232.

[10] 陈涛,李正炜,王建立,等. 应用压缩传感理论的单像素相机成像系统[J]. 光学精密工程,2012,20(11):2523 - 2530.

[11] 俞文凯,姚旭日,刘雪峰,等. 压缩传感用于极弱光计数成像[J]. 光学精密工程,2012,20(10):2283 - 2292.

[12] 冯贵兰,周文刚. 基于 Spark 平台的并行 KNN 异常检测算法[J]. 计算机科学,2018,45(S2):349 - 352,366.

[13] 郑凤飞,黄文培,贾明正. 基于 Spark 的矩阵分解推荐算法[J]. 计算机应用,2015,35(10):2781 - 2783,2788.

[14] 王军. 基于 Apache Spark 的大数据分析引擎应用研究[J]. 电子测试,2018(16):72,66.

[15] 廖湖声,黄珊珊,徐俊刚,等. Spark 性能优化技术研究综述[J]. 计算机科学,2018,45(7):7 - 15,37.

(上接第 40 页)

[8] 周万珍,阚景森. 基于 k-means 与 Apriori 算法的食物 营养成分分析[J]. 科学技术与工程,2018,18(17):211 - 216.

[9] 中国营养学会. 中国居民膳食营养素参考摄入量[M]. 北京:科学出版社,2014: 23 - 168.

[10] Kennedy J, Eberhart R. Particle swarm optimization[C]// Proceedings of ICNN'95—International Conference on Neural Networks. IEEE, 1995.

[11] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation,2004,8(3):256 - 279.

[12] 程义勇.《中国居民膳食营养素参考摄入量》2013 修订版简介[J]. 营养学报, 2014, 36(4):313 - 317.

(上接第 54 页)

[16] LI Y D. Research and Implementation of a science and technology information base management system based on ontology[D]. Beijing university of posts and telecommunications, 2018: 32 - 36.

[17] 李亚东. 基于本体模型的科技信息知识库管理系统研究与实现[D]. 北京:北京邮电大学,2018.

[18] 项灵辉,顾进广,吴钢. 基于图数据库的 RDF 数据分布式存储[J]. 计算机应用与软件,2014,31(11):35 - 39.