第 33 卷 第 5 期
2018 年 10 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol. 33 No. 5
Oct. 2018

DOI:10.13364/j.issn.1672-6510.20170159
数字出版日期：2018–06–26；数字出版网址：http://kns.cnki.net/kcms/detail/12.1355.N.20180626.1440.006.html

基于 Spark 的大规模天文数据天区覆盖生成算法

熊聪聪[1]，田祖宸[1]，赵 青[1]，冯 阔[1]，崔辰州[2]
（1. 天津科技大学计算机科学与信息工程学院，天津 300457；2. 中国科学院国家天文台，北京 100012）

摘 要：天区覆盖生成是天文数据归档中的重要一环，其结果对天文数据检索、计算等后续处理流程至关重要. 由于天文数据的海量性，应用传统科学计算方法处理这一问题通常耗时较长，效率不高，且受存储空间的制约，扩展性差. 为解决这一问题，本文提出了一种基于 HEALPix 索引和 Spark 框架的高效分布式天区覆盖生成算法. 实验证明：该算法可以在短时间内完成大规模天文数据的天区覆盖生成，为实现海量天文数据的快速归档提供了支持；同时，所生成的结果还可以用于数据可视化，直观地展现星表中的天文数据在天区上的分布情况.

关键词：Spark；大数据；HEALPix；天文；天区覆盖
中图分类号：TP311.13；P114 **文献标志码**：A **文章编号**：1672-6510(2018)05-0063-05

A Sky Coverage Generation Algorithm for Large-scale Astronomical Data Based on Spark

XIONG Congcong[1], TIAN Zuchen[1], ZHAO Qing[1], FENG Kuo[1], CUI Chenzhou[2]
(1. College of Computer Science and Information Engineering, Tianjin University of Science & Technology, Tianjin 300457, China; 2. National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China)

Abstract: The sky coverage generation is an important part of astronomical data archiving. Its results are very important for the subsequent processing of astronomical data retrieval and calculation. Due to the massive nature of astronomical data, it is usually time-consuming and inefficient to deal with this problem with traditional scientific computing methods, and it is limited by the constraints of storage space and poor scalability. In order to solve this problem, this paper presents an efficient sky coverage generation algorithm based on HEALPix index and Spark. Experiments show that the algorithm can complete the generation of large-scale astronomical data in a short time. The generated results of the algorithm can be used to accelerate the processes of dealing with astronomy data and can also be used for data visualization of the sky coverage.

Key words: Spark; big data; HEALPix; astronomy; sky coverage

近年来科技的发展大大提高了天文数据的采集能力，各波段的数据量呈指数级增长，天文学逐渐走向全波段巡天的“大数据”时代[1]. 如此庞大的数据量，对传统科学计算技术处理大规模天文数据的做法提出了新的挑战. 一般而言，天文数据的计算主要分为数据导入、数据预处理和数据计算三个阶段，如果面对大规模天文数据，在预处理阶段对数据没有进行合理的安排，会降低数据计算阶段的执行效率，制约天文发现的进展. 天区覆盖是望远镜所拍摄区域在天区上的一种表征方式，它是将大量的星体记录以一定的规则对杂乱无序的天体数据重新整理、归档的过程，对后续的数据查询、处理、分析等工作具有重要的支持作用. 但望远镜拍摄能力的逐渐提升带来了庞大的数据量，如何提高天区覆盖计算的效率成为另一重要问题.

天文计算具有数据量大、I/O 敏感性、内存占用高等特点，早期的计算通常在超级计算机或高性能计算机集群上完成的[2]，但这种方案费用较高，而且程序设计复杂. 云计算的出现让研究人员得以用廉价、高效的方式处理海量数据，其中 Spark 框架凭借可以

收稿日期：2017–06–02；**修回日期**：2017–12–12
基金项目：国家自然科学基金资助项目(61402331，61402332)；天津市教委科研计划项目(2017KJ035)
作者简介：熊聪聪（1961—），女，四川泸州人，教授；**通信作者**：赵 青，讲师，zhaoqing@tust.edu.cn

在内存中进行计算的优势，具有比 Hadoop MapReduce 更高的计算效率，为天文大数据处理提供了前所未有的潜力. 近年来已有一些天文研究工作使用 Spark 分布式计算框架处理大规模天文数据，如利用决策树实现星系的分类[3]、利用聚类算法实现光谱分类[4]、还有结合 Hadoop 与 Spark 利用聚类算法实现了星象图片分类[5]. 此外，从基于 Spark 的天文研究的软件平台 AstroSpark[6]、高性能计算机集群 SDSC Comet[7]中也不难看出，分布式计算技术已成为海量数据下进行天文研究工作的一种发展趋势. HEALPix 作为一种天文学常用的天区索引方法，被应用在了如星表交叉证认[8]、天文图像显示[9]等多种天文相关的场景中. 由此可见，HEALPix 与 Spark 的结合尤其适用于处理大规模天文数据. 但是，现有的研究多是为了解决某一种天文问题，或针对计算环节进行分析，而对天文原始数据的预处理及归档方面却鲜有提及，实验也只是直接读取原始数据并进行计算. 通常情况下，原始天文数据往往是分散、无序存放的，而对这种杂乱数据频繁读取会大大降低计算效率.

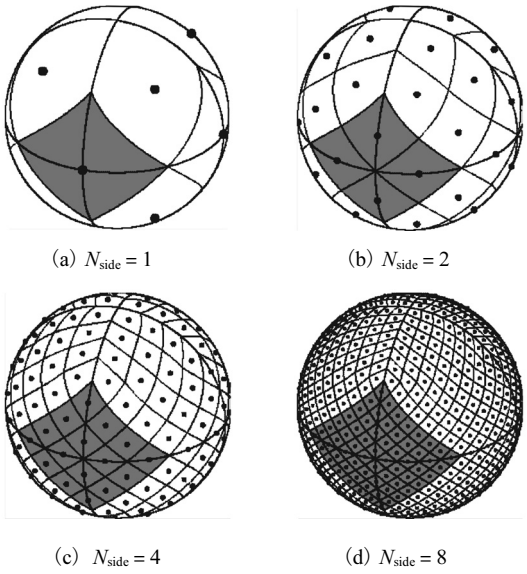
鉴于此，本文在借鉴前人工作的基础上，围绕对原始天文数据的预处理及归档，结合文献[10]和文献[11]中介绍的层次化索引思想，提出一种基于 Spark 天区覆盖生成的数据预处理与归档方法：在 HEALPix 分层索引的基础上，将天文数据层次化、分块、连续存放，从而提升后期交叉证认、漏源监测等天文计算中对数据进行访问、处理的效率；利用 Spark 的快速计算及迭代计算的特点，将天区覆盖生成方法设计成一系列弹性分布式数据集的转换操作，使其可以在并行环境下处理大规模天文数据. 同时，索引的结果还可以用于数据可视化，方便研究人员直观了解各个星表中的天文数据在天区上的分布情况.

1　天区覆盖生成算法

1.1　HEALPix 层次化球面索引

天文数据以每一条天体的信息为单位，如果在天文计算中以词为单位逐条处理，过细的粒度会大大降低对数据访问查询的效率，消耗不必要的资源. 针对这一问题，可以采用对数据进行分块的方法，适当增加处理单位数据的粒度，这样在抽取数据的时候可以分段读取，本文利用了一种天文领域常见的球面索引方法 HEALPix（hierarchical equal area ISO latitude pixelation）[12].

HEALPix 根据星体坐标和区块块号之间的映射关系，采用四叉树结构，以层层递归的方式对天球进行等面积划分（图 1）. 其具有层次化、等面积和等纬度分布等特点. HEALPix 在初始层级将天球划分成 12 个面积相等的基准球面四边形，每一个四边形被看作一个 pixel（这里称为“区块”，下同），从 0 至 11 为其编号. 到了第二层级，每个四边形再被等分为 4 个子四边形，随着层级深度的增加，每层的四边形会不断地被细分. 最终到第 k 层时，对天球基准四边形任一边细分的次数即为当前索引的分辨率，用参数 N_{side} 表示（$N_{\text{side}} = 2^k$）[12].



（a）$N_{\text{side}} = 1$　（b）$N_{\text{side}} = 2$

（c）$N_{\text{side}} = 4$　（d）$N_{\text{side}} = 8$

图 1　HEALPix 天球分区示意图

Fig. 1　HEALPix partition of the sphere

HEALPix 具有 2 种编码规则：RING 和 NESTED. 其中 RING 是按照自西向东、自北向南的顺序对区块编号；NESTED 是以层次化递归的方式对区块编号. 因顺序编码的 RING 规则不适应于天区覆盖生成的迭代操作，这里采用 NESTED 编码方法，其编码形式如图 2 所示. 每一层级的区块都会被分配唯一的二进制编码，每一个区块的块号由父块号和当前层级编码两部分组成. 当父块（记为 highOrder）被分为 4 个子块后，每个子块根据其位置会被赋予 00、01、10 或 11 的二进制编号（记为 lowOrder），也就是当前层级编码，父级块号作为前缀，当前层级编码作为后缀，即执行了 lowOrder≪2&highOrder 移位操作，形成了新层级的编码. 随着层数深度的增加，对应层级的块号位数也会越来越大. 如果以 k 表示划分的层级，那么 $N_{\text{side}} = 12 \times 2^{2k}$ 表示当前层级可以最多划分出的区块数.

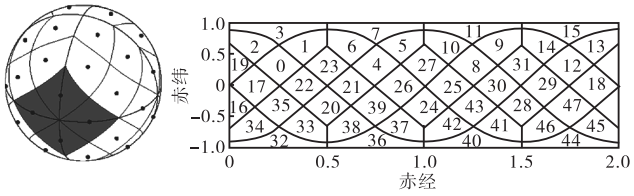
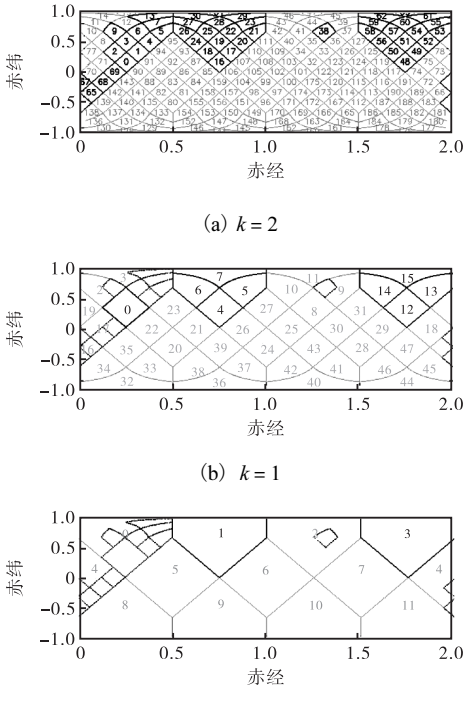


图 2　HEALPix 索引及 NESTED 编码方式示意图（$N_{side}=2$）

Fig. 2　HEALPix index and NESTED numbering schemes ($N_{side}=2$)

1.2　天区覆盖生成算法的基本原理

从最低层级 k 开始，利用上文提到的位运算对当前层每个区块的块号进行分割操作，获得相应的父块块号（highOrder）和子块块号（lowOrder）. 当所有区块处理完毕后，将相同父块块号的区块聚合在一起，分析相同父块块号下的子块分布情况，如果这些子块中的星体覆盖了 00、01、10、11 位置时，当前区域内的所有区块的原始块号全部由父级块号代替，同时这些区块会被保留至下一次迭代操作. 如果这些子块没有满足上一条件则保留原始块号，舍弃并进行其他处理，不代入下一次迭代. 如此往复，每一次迭代后满足条件的数据会聚集起来，成为下一次迭代的输入，这一迭代操作持续到没有可聚集的数据为止. 图 3 以一组第 2 层级下的小规模数据为例展示了这一变换过程.



（a）$k=2$

（b）$k=1$

（c）$k=0$

图 3　天区覆盖生成过程示意图

Fig. 3　The process of sky coverage generation

图 3（a）的深色区域表示初始数据在第 2 级的区块覆盖情况，图 3（b）与图 3（c）实线区域为满足条件聚合后的区块，即原 4 个子块的 HEALPix ID 更新为父块的 HEALPix ID，虚线表示为无法继续聚合但被保留的区块. 当算法执行到第 0 级或者无法聚合时结束.

1.3　位运算

在聚合操作中主要通过二进制的移位操作获得子块自身以及对应的父块信息. 以层级为 2、块号为 19 的区块为例，对应的二进制表示为 010011，那么从低两位 11 就可以知道其作为上一级的子块时的位置，而剩余的高位 0100 即为对应上一级区块的块号. 以此类推，如果再一次进行移位操作，就可以得知 19 这个区块在 0 和 1 层级时对应的块号为 1 和 4. 具体的对应关系见表 1.

表 1　某个区块的层级对应关系表（以层级 2，块号 19 为例）

Tab. 1　Hierarchical correspondence of a pixel (No. 19, level 2)

层级	当前层块号		父块块号		子块块号	
	十进制	二进制	十进制	二进制	十进制	二进制
2	19	010011	4	0100	3	11
1	4	0100	1	01	0	00
0	1	01	—	—	—	—

2　基于 Spark 的算法设计与实现

对天文数据的处理量通常比较庞大，如果在单台计算机上进行处理，往往受存储空间和处理能力的限制. 由于本算法中的迭代操作会产生大量中间数据，如果使用 Hadoop MapReduce 框架执行这一任务，系统会将中间变量存储在磁盘中并产生大量的读写操作. 由于磁盘的存取速度慢，频繁的磁盘读写操作会制约数据的处理效率. 因此，本算法使用 Spark 作为计算框架.

天区覆盖生成算法在 Spark 平台下的实现主要分为 2 个阶段，算法中涉及到的 RDD（Resilient Distributed Datasets）及算子如图 4 所示，具体实现代码可以从网站（https://github.com/ZuChen93/CrossMatch）获得.

2.1　第 1 阶段

数据预处理. 首先用 textFile 算子将原始数据从 HDFS 中导入，转换为 RDD，再用 map 算子根据天体的赤经 ra、赤纬 dec 添加 HealPix 索引. 由于从网站获得的天文数据中未含有 HEALPix 索引信息，所以

需要根据星体的坐标逐条生成某一给定层级 k 下的 HEALPix 索引编号，k 根据数据密度或者天区覆盖的最小精度决定. 此外，由于原始数据是未经压缩的文本数据，在 RDD 中直接以字符串形式处理会占用大量内存空间，而且除坐标以外的大部分信息在实际处理过程中并未参与运算. 为了提高数据的传输效率，在添加 HEALPix 索引之后，将每条天体数据中除索引以外的全部信息以新生成的唯一 ID 替代.

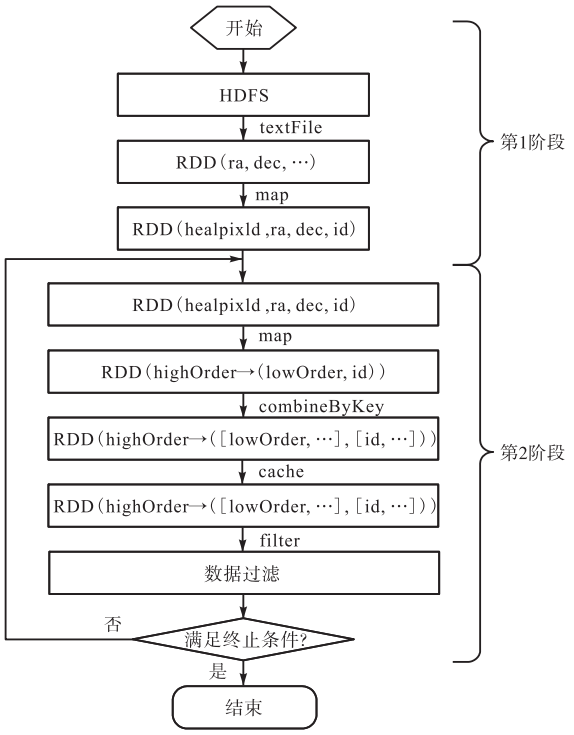


图 4　天区覆盖生成算法流程图

Fig. 4　Flow chart of sky coverage generation

2.2　第 2 阶段

根据天区分布生成算法的原理，对数据执行迭代聚合操作，并根据条件对数据进行过滤. 在本阶段采用了多个 map 操作，用于根据计算需要变换键值. combineByKey 算子作为核心操作，主要用于将相同 highOrder 的数据整合为一条记录，并统计 lowOrder 的分布情况. 最后使用 filter 算子，当 lowOrder 完全分布在 00、01、10、11 的时候，对应 highOrder 的记录可以用于下一次迭代，反之会被剔除并保存至 HDFS 中.

需要说明的是，算法在迭代进行前添加了一个 longAccumulator 累加器，用于统计每次迭代完成后可以继续聚合的数据条数. 当数值为 0 时，说明没有可以继续处理的数据，迭代终止.

另外，本阶段会有一些需要重用的 RDD 数据，如 combineByKey 执行后生成的 RDD 会被 filter 算子调用两次. 这里需要用到 Spark 中特有的持久化特性，即在 RDD 需要被频繁使用的时候，使用 cache 持久化操作将数据缓存至内存中，这样如果后面需要再次调用这组数据的话，就可以直接到内存中访问，从而加快对此 RDD 访问的速度. 如果没有这一操作的话，Spark 中的 lazy 特性会使 RDD 在每次被调用的时候，都要根据它的 DAG 映射重新开始计算[13]，这样不但占用系统资源，还会降低程序运行效率. 算法中对类似的 RDD 都进行了持久化处理.

3　实验与分析

3.1　数据集与集群配置情况

为了验证算法的性能，选取在中国虚拟天文台网站（http://casdc.china-vo.org/mirror/2MASS）中公开的 2 MASS 星表中 Point Source Catalog（PSC）数据集进行实验. 共 152 GB 数据中包含 4.7 亿条星体信息，以文本形式存储，每一行数据表示为一个星体，每一行数据中的各个属性信息以竖线分割.

实验在阿里云的 E-MapReduce 平台下进行，集群版本是 EMR-3.4.2（Spark 2.1.1，Hadoop 2.7.2），每个节点的配置为“通用型 n2，4 核处理器，16 GB 内存，SSD 云盘 80 GB × 4，series Ⅲ”，1 个 Master 节点，多个 Core 节点，集群资源使用 YARN 进行托管. 实验数据预先存储至集群下的 HDFS 文件系统中，Spark 以 YARN-client 模式运行，通过 spark-submit 中的 num-executors 参数调整计算节点数，以模拟算法的并行执行情况.

3.2　初始层级 k 对天区覆盖生成结果的影响

1.1 节已提到 $N_{\mathrm{side}} = 2^k$ 是影响 HEALPix 对天区细分精度的首要因素，其中 k 代表细分的层数. 为了测试这一参数是否会对天区覆盖生成算法的结果有影响，选定计算节点数为 64，令 k 取值分别为 2，4，6，···，20，实验结果如图 5 所示.

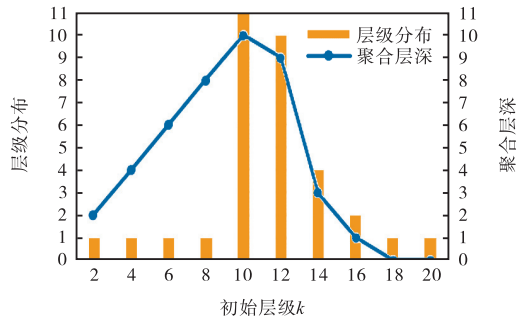


图 5　初始层级 k 与算法生成结果的关系

Fig. 5　Relation between k and results of algorithm

图中的“层级分布”表示数据从初始层级 k 不断聚合，直到程序结束时，能够聚合出数据的层数分布；“聚合层深”表示程序结束时的层级距离初始层级 k 的深度. 可以看出：当 $2<k<10$ 时，“聚合层深”与 k 相等，其中“层级分布”为 1 的情况表明数据全部聚合到了第 0 层；当 $k>18$ 时，“聚合层深”为 0，表示数据完全无法聚合，全部数据都停留在了初始 k 层. 以上两种情况都不是最理想的聚合结果. 而 k 的取值为 10 和 12 时，数据的划分程度更好，k 取 10 的时候数据的聚合结果最理想，且在不同层级间均有所分布. 通过分析实验结果猜测当 k 取值过低时，由于区块范围过大，大量数据都集中在同一区块中；而 k 取值过大时，由于区块过小，星体分散严重，大量数据在最低层级就无法向上聚合. 实际情况下 k 的取值除了要考虑处理时间和层级分布外，还应根据数据归档的需要设定.

3.3　计算节点数对算法执行时间的影响

为了比较集群节点数对计算效率的影响，分别在 1、2、4、8、16、32、64 计算节点数下用相同数据测试了算法的性能，初始层级 k 定为 10. 实验结果如图 6 所示.

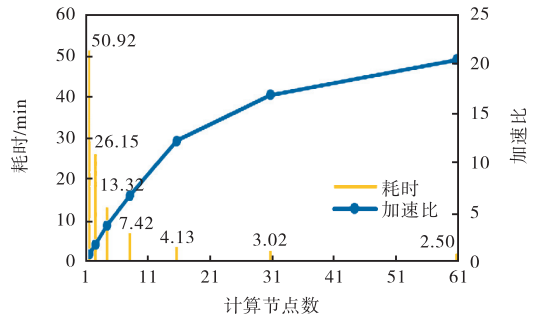


图 6　算法在不同节点下的耗时情况（$k=10$）

Fig. 6　Time elapsed on different nodes（k = 10）with the algorithm

从图 6 可以看出：随着节点数的增加，耗时越来越少；但是从节点数为 16 开始，增速逐渐放缓. 在本算法的迭代计算中，reduce 算子会在集群中产生大量的数据交换操作，这种操作会造成节点间的通信更频繁，从而产生大量的网络开销，并且随着计算节点数的增加而网络开销越来越大，会在整个程序执行流程中占据较多时间. 这可能是造成计算节点达到 16 后，加速比逐渐放缓的主要因素. 由此可见：合理地增加计算节点数可以加快处理数据的速度，但是要想在增加计算节点后获得更好的加速比，需要进一步优化算法或者集群资源分配；可以根据需要，人为规定聚合结束时停止的层级，使得聚合操作提前结束，避免耗费不必要的计算资源.

4　结　语

本文探讨了一种基于 Spark 的天区覆盖生成算法，在 HEALPix 分层索引的基础上，对天文数据层次化、分块、连续存放. 实验证明，本算法借助 Spark 可以在较短的时间内处理大规模数据. 将整理后的数据用于后期相关天文计算，可明显提升访问、处理数据的效率. 此外，算法处理后的数据还可以用于数据可视化，绘制天区覆盖图，以便直观地了解天文数据在天区上的分布情况.

分布式计算的性能受算法优化、参数调优等多种因素制约，例如文献[6]提到 Spark 对数据块划分的合适与否会影响计算的并行性，因此还可以从以上方面继续优化本文算法.

致谢：本研究受天津市自然科学基金项目（17JCQNJC00400）资助，特此致谢！

参考文献：

[1] 崔辰州，于策，肖健，等. 大数据时代的天文学研究[J]. 科学通报，2015，60(5)：445–449.

[2] Zhang Z，Barbary K，Nothaft F A，et al. Scientific computing meets big data technology：An astronomy use case [C]//Proceedings of IEEE International Conference on Big Data. New York，USA：IEEE，2015：918–927.

[3] 黄智昌，王俊义，郑霖，等. 基于 Spark 深度感知决策树的恒星/星系分类应用研究[J]. 计算机应用研究，2017，34(3)：765–768.

[4] 刘聪. 基于 SPARK 平台的 LAMOST 早 M 型光谱聚类的研究[D]. 济南：山东大学，2016.

[5] Sharma T，Shokeen V，Mathur S. Multiple K means++ clustering of satellite image using Hadoop MapReduce and Spark[J]. International Journal of Advanced Studies in Computer Science and Engineering，2016，5(4)：23–31.

[6] Brahem M，Lopes S，Yeh L，et al. AstroSpark：Towards a distributed data server for big data in astronomy [C]//Proceedings of the 3rd ACM SIGSPATIAL PhD Symposium. New York，USA：ACM，2016：3003823.

[7] Tatineni M，Lu X，Choi D，et al. Experiences and benefits of running RDMA Hadoop and Spark on SDSC

（下转第 78 页）