中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

基于优化RDD分区的Spark并行K-means 大尺度遥感图像分割

李玉, 崔书琳, 赵泉华

引用本文:
李玉,崔书琳,赵泉华. 基于优化RDD分区的Spark并行K-means 大尺度遥感图像分割[J]. 控制与决策, 2024, 39(5): 1612-1619.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.1717

基于优化RDD分区的Spark并行K-means大尺度遥感图像分割

李 玉†, 崔书琳, 赵泉华

(辽宁工程技术大学 测绘与地理科学学院，辽宁 阜新 123000)

摘 要: 大尺度遥感图像分割对单机处理方式而言是巨大挑战. Spark平台为在单机上构建用于大数据处理的分布式计算环境提供了可能. 当Spark平台内置的K-means算法用于数字图像处理时, 其中的Spark Shuffle弹性分布式数据集(RDD)分区一般采用缺省设置, 尽管这种RDD设置简单便捷, 但对大尺度图像分割任务容易造成“多分区、小数据”现象, 极大影响图像分割速度. 为此, 采用覆盖部分上海市区的WorldView-3遥感图像为测试数据, 在K-means算法初始化聚类中心阶段自定义影响RDD分区的参数spark.sql.shuffle.partitions, 在迭代计算阶段调用coalesce( )算子减少分区数; 与串行K-means算法对比验证单机处理大数据的可行性与有效性, 与优化前的Spark并行K-means算法对比实现了大尺度遥感图像快速分割. 实验结果表明, 在K-means算法初始化聚类中心和迭代计算阶段, 将RDD分区数设置在CPU核数的1～10倍, 总用时由优化前的145 s缩减到97 s, 尤其在初始化聚类中心阶段的时间效率上, 优化后是优化前的500～1 000倍.

关键词: Spark平台；单机大数据处理；大尺度遥感图像；RDD优化；图像分割；并行K-means算法

中图分类号: TP751 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.1717

引用格式: 李玉, 崔书琳, 赵泉华. 基于优化RDD分区的Spark并行K-means大尺度遥感图像分割[J]. 控制与决策, 2024, 39(5): 1612-1619.

Spark parallel K-means large scale remote sensing image segmentation based on optimized RDD partition

LI Yu†, CUI Shu-lin, ZHAO Quan-hua

(School of Geomatics，Liaoning Technical University，Fuxin 123000，China)

Abstract: It is a great challenge for segmentating large scale remote sensing images on a single computer. The Spark platform makes it possible to build a distributed computing environment for big data processing on a single computer. When the K-means algorithm built in Spark platform is used for digital images processing, the Spark Shuffle resilient distributed dataset (RDD) partition generally adopts the default setting. Although this RDD setup is simple and convenient, it is easy to cause the phenomenon of “excessive partition and too little data” in the large scale images segmentation task, which greatly affects the image segmentation speed. Therefore, this paper utilizes the built-in K-means algorithm for segmenting the WorldView-3 images coving part of Shanghai city, which properly defines the RDD partition parameter spark.sql.shuffle.partitions during the initializing clustering centers stage and adaptively calls the coalesce() operator to adjust the number of RDD partitions during iteration. Comparing with the serial K-means algorithm, the feasibility and effectiveness of single computer processing of big data are verified. Comparing with the Spark parallel K-means algorithm with the parameters for the default setting, the proposed algorithm can realize large-scale image segmentation faster. The experimental results show that, in the both stages of initialization and iterative computation of cluster centers for the K-means algorithm, the RDD partition number is set at 1-10 times of the CPU core number, which reduces the total time from 145s before optimization to 97s. Especially in the time efficiency of the initializing cluster center stage, the time efficiency after optimization is 500-1000 times that before optimization.

Keywords: Spark platform; single computer big data processing; large-scale remote sensing images; RDD optimization; image segmentation; parallel K-means algorithm

收稿日期: 2022-09-29；**录用日期:** 2023-02-10.

基金项目: 辽宁省自然科学基金项目(2022-M S-400)；辽宁省教育厅重点攻关项目(LJ2020ZD003).

†通讯作者. E-mail: liyu@lntu.edu.cn.

0 引 言

卫星遥感技术革命性地拓展了人类观测地球的手段和能力[1]. 随着传感器网络、云计算、大数据的日益发展,人类可以高效获取区域甚至全球尺度的对地观测数据. 根据UCS (union of concerned scientists)统计,截止2020年3月,全球范围已有928颗对地观测卫星在轨运行[2]. 研究者亟需依赖计算机和遥感信息智能提取技术挖掘蕴含在这些海量数据中的地表信息和知识,而针对大数据处理的并行计算正是顺应了这种信息需求. 目前,主流大数据并行处理技术包括Hadoop MapReduce[3]、Spark和机器学习框架等[4]. 其中,Spark平台有3大优势: 1) 在K-means算法的迭代优化求解时,Spark平台会将需要迭代的数据存储在内存中,减少了磁盘I/O操作[5],从而提高了计算效率; 2) Spark平台提供了一个可并行计算的弹性分布式数据集(resilient distributed dataset, RDD)[6],相比于传统的MapReduce框架,Spark在RDD中内置很多操作函数,使任务并行执行在RDD的各个分区上[7-8],最终整合各个分区的计算结果; 3) RDD还具有容错性[9],一旦数据丢失,它会根据内部依赖关系重新计算某个分区丢失的数据,不会因此而导致程序崩溃. 尽管如此,在实际使用中RDD划分多采用系统提供的缺省分区,由于受spark.sql.shuffle.partitions参数的影响,Shuffle过程中的RDD分区数均默认为200,一旦选择缺省分区,Spark平台不能根据所处理的数据规模自适应地确定RDD分区数. 为此,文献[10]提出一种高效的RDD自主缓存替换机制,解决了由于RDD重复使用导致的作业执行效率低的问题. 文献[11]通过改进RDD淘汰机制和缓存方式,降低了Java对象序列化开销. 文献[12]提出一种新的RDD分区权重缓存替换算法,以提高内存资源利用率和作业执行效率. 文献[13]将模糊K-means算法分别搭建在MapReduce和Spark框架下,通过处理文本数据比较分析二者差别. 文献[14]提出一种自适应缓存策略,同样提高了Spark的任务执行效率. 目前,大多数研究学者更多地是从RDD的缓存角度提高作业执行效率,且处理的数据多为文本大数据. 本文将RDD优化分区应用在基于K-means算法的遥感图像分割[15]中,在初始化聚类中心和迭代计算聚类中心两个阶段自定义RDD分区,从而节省图像分割的执行时间,提升了工作效率.

1 RDD分区优化方案

1.1 Spark并行K-means算法

Spark平台提供了可扩展的机器学习库(machine learning library, MLlib),其中包括K-means算法. 本文直接调用Spark MLlib中的K-means算法实现遥感图像分割,并在Spark平台的分区内并行执行图像的快速分割. 本文实验中用到的核心算法是K-means聚类算法,它是一种无监督聚类算法[16],具有简洁高效性. 给定遥感图像$\boldsymbol{Z}=\{\boldsymbol{Z}_i, i=1,2,\dots,n\}$. 其中: i为像素索引,n为像素数,$\boldsymbol{Z}_i=(Z_{ij},\ j=1,2,\dots,m)$为像素$i$的光谱测度矢量,$j$为波段索引,$m$为波段数,$Z_{ij}$为像素$i$波段$j$的光谱测度. 将$\boldsymbol{Z}$划分到$k$个不同的聚类,聚类中心矢量集合记为$\boldsymbol{C}=\{\boldsymbol{C}_l, l=1,2,\dots,k\}$. 其中: l为聚类索引,$\boldsymbol{C}_l=(C_{lj},\ j=1,2,\dots,m)$为聚类$l$的聚类中心矢量,$C_{lj}$为$\boldsymbol{C}_l$在波段$j$的分量. 设$\boldsymbol{L}=\{L_i,\ i=1,2,\dots,n\}$为像素类属标识集合,其中$L_i\in\{1,2,\dots,k\}$为像素$i$的类属标识,即$L_i=l$表示像素$i$隶属于聚类$l$. 综上,$K$-means聚类算法的具体步骤如下.

step 1: 随机获取k个初始聚类中心,记为$\boldsymbol{C}^{(0)}=\{\boldsymbol{C}_i^{(0)},\ l=1,2,\dots,k\}$.

step 2: 计算$\boldsymbol{Z}_i$与$\boldsymbol{C}_l^{(0)}$之间的欧氏距离

$$D_{il}^{(0)}=\sqrt{\sum_{j=1}^{m}(Z_{ij}-C_{ij}^{(0)})^2}, \tag{1}$$

将像素i划分到$D_{il}^{(0)}$最小的聚类,即

$$L_i^{(0)}=\arg\min\{D_{il}^{(0)}, l=1,2,\dots,k\}. \tag{2}$$

更新聚类中心矢量为

$$\boldsymbol{C}_l^{(1)}=\frac{1}{N_l^{(0)}}\sum_{L_i^{(0)}=l}\boldsymbol{Z}_i, \tag{3}$$

其中$N_l^{(0)}=\#\{\boldsymbol{Z}_i; L_i^{(0)}=l\}$,#为求集合中元素数操作符.

step 3: 对于$\boldsymbol{C}^{(t)}=\{\boldsymbol{C}_l^{(t)}, l=1,2,\dots,k\}$,利用式(1)计算$D_{il}^{(t)}$,并依据式(2)决定第$t$次迭代时像素$i$的类属标号$L_i^{(t)}$,由式(3)计算$\boldsymbol{C}^{(t+1)}$.

step 4: 重复step 3,直到满足预先给定的终止条件,如聚类中心不再变化或者达到设定的最大迭代次数,得到k个聚类中心.

1.2 RDD分区数对程序执行时间的影响

Spark平台实现图像分割与Hadoop MapReduce大体相同,都是由Map阶段经过Shuffle操作,最终汇总到Reduce并输出最优的结果[17]. 不同的是,Spark平台可以处理图像数据[18],通过DataFrame API加载指定目录中的图像文件,生成一个DataFrame对象,利用该对象对图像数据进行聚类处理. 本文实验中用到的彩色遥感图像定义在RGB色彩空间中[19],RGB色彩空间[20]是一种加法色彩空间,其中红色、绿色

和蓝色波段的光以各种方式加在一起,再现广泛的颜色. 该空间为彩色遥感图像的每个像素设置一个彩色矢量,而彩色矢量中分量分别为R、G、B强度. Spark平台获取彩色遥感图像数据并将其转化为可处理的数据样本,具体流程如下.

step 1: 读取图像路径,通过decode方法将图像文件解析成一个Row字段,分别包括: origin图片路径信息、height图片高度信息、width图片宽度信息、nChannels图片通道数量信息、mode即openCV兼容的类型、data即以openCV兼容的方式排列,大多数情况下按行排列BGR.

step 2: 自定义一个用于转换图像信息的工具类,创建存储RGB数值的数组.

step 3: 从元数据中提取特征指标数据,用VectorAssembler方法将给定数据信息组合成单个向量列,转换后得到一个DataFrame. 此时的图像数据转化为大数据样本,转化结果示例如表1所示.

表 1 样本数据示例

B	G	R	特征向量
43	43	43	[43.0, 43.0, 43.0]
109	109	109	[109.0, 109.0, 109.0]
103	103	103	[103.0, 103.0, 103.0]
106	106	106	[106.0, 106.0, 106.0]

step 4: 利用MLlib中的K-means算法实现对数据的聚类,输出分割之后的图像.

在Spark平台中,RDD是最基本的抽象数据集[21],分区是RDD内部并行计算的一个基本单元,每个RDD都由若干个分区组成. RDD中的算子主要分为两类: 行动(Action)算子和转换(Transform)算子[22-24]. 出现一个Action算子就会产生一个任务(Job), 一个Job中包括很多阶段(Stage), Transform算子不会产生Job,只是用于完成Stage的中间处理.

Spark的本地模式本身具有较好的自动调节性[25],但因为Spark中Shuffle分区数是静态的,不会随着所处理的数据规模而变化,所以分区方式对Spark程序执行时间的影响较大,合理的分区是提升图像分割效率的关键. 根据Spark UI界面显示,RDD的分区内数据分布均匀,且无溢出现象,但过小的分区块将导致分区数增多. 这是因为spark.sql.shuffle.partitions参数的影响,解析并分配数据处理任务时RDD分区为200. 汇总每个分区数据进入迭代计算时,由于RDD的依赖关系,分区数不变. 而RDD分区数过多,线程间通信会占据大量时间,任务调度也会更耗时,导致程序开始执行时便浪费时间,降低了程序整体的执行效率.

1.3 Spark并行K-means算法RDD的优化方案

数据分区只作用于⟨key, value⟩形式的数据,当一个Job中包含Shuffle操作类型的算子时会产生静态分区. 在本地模式下,综合考虑数据的存储位置、CPU核数、内存大小等因素, Spark提供RDD缺省分区. 优化前的Spark UI界面显示,在执行K-means算法初始化聚类中心的任务时,默认生成两个Stage, 400个Task,执行时间高达分钟量级. 各Job、Stage和Task的关系如图1所示,其中Spark平台提交任务,通过Action算子创建一个Job,每个Job会根据宽依赖算子,即Shuffle类型的算子产生一个Stage, Stage会根据RDD分区的数量生成相应数量的Task.

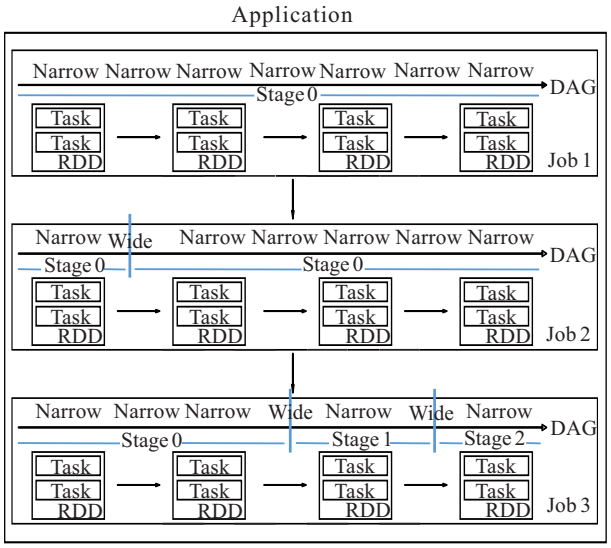


图 1 Job、Stage和Task的关系

在执行迭代计算任务中,由于线程间的大量通信,用时反而远超串行算法的执行时间,这便失去了Spark平台快速处理大数据的意义. 为此,本文在K-means算法筛选初始化聚类中心及其迭代计算聚类中心阶段采用自定义RDD分区,优化后程序执行的具体流程如表2所示. 表2中: Par表示分区数,w和h分别表示图像的宽度和高度,a为不包括0的自然数,最好是硬件CPU核数的整数倍,aPar表示a个分区, maxIter表示最大迭代次数,本文设定为20次. 经实验验证,程序执行过程中在初始化聚类中心和迭代优化聚类中心两个阶段,导致整体执行效率低下. 本文在图像像素数据转化生成⟨key, value⟩数据之后,在执行聚类中心迭代优化计算之前自定义分区.

2 实验与结果分析

2.1 实验数据与环境

本文采用的实验数据是分辨率为0.5 m覆盖部分上海市区的WorldView-3遥感图像. 实验内容包括: 确定聚类数,串行与并行迭代时间对比,验证分区

表 2　优化后程序流程

优化后的各Job详细任务
Job 0: 以数组的形式返回数据集的所有元素(1Par)
Job 1: 按数据集中colName=“w”, colName=“h”对数据分组,将相同key对应的值放入一个迭代器
自定义参数spark.sql.shuffle.partitions (aPar)
Job 2: K-means算法对数据集采样,初始化聚类中心 (numPartitions= $a, k = 5$, maxIter=20, seed=1, replicas=1)
Job 3: 随机抽样
Job 4: 统计数据集中元素的总数
Job 5: 将RDD类型的数据转化为数组,同时将数据拉取到driver端
Job 6: 对Row字段进行加和,并保存其结果
Job 7: 将RDD类型的数据转化为数组,同时将数据拉取到driver端
Job 8: 统计相同value的个数
while (Job 9: collectAsMap, reduceByKey) **do**
加入coalesce()算子,将〈key,value〉格式的RDD回收到driver端形成一个Map,并对相同key的数据
进行处理,最终每个key只保留一条记录,自定义RDD分区数,迭代计算,循环设定的maxIter次(aPar)
end while Job 28
Job 29: 将RDD类型的数据转化为数组,同时将数据拉取到driver端
while (Job 30: first算子)
返回RDD中的第一个元素,即最终的聚类中心,循环k次,输出k个聚类中心(1Par)
end while
Job 35: 按照每个聚类中心的RGB色彩为该类所有像素点重新上色,输出分割结果图像,关闭sparkContext

数影响. 本文使用的Spark实验平台是由加州大学伯克利分校AMP (algorithms, machines, and peoplelab)实验室[26-27]开发的通用内存并行计算平台,将其搭载在Windows 10操作系统下,硬件资源配置信息: Inter Core i5-5200U四核2.20 GHz处理器,16 GB内存;Spark环境的相关配置版本信息: Spark-2.4.1, JVM-1.8.0, Hadoop-2.7.7, Scala-2.11.11. 本文实验均在Java编程语言的集成开发环境IntelliJ IDEA中执行. 由于Spark的内存管理模块在整个操作系统中也占有主要的地位,程序执行前Java虚拟机(Java virtual machine, JVM)的参数设置如下.

1) -Xms: 初始堆大小, 4 096 MB;

2) -Xmx: 最大堆大小, 14 336 MB;

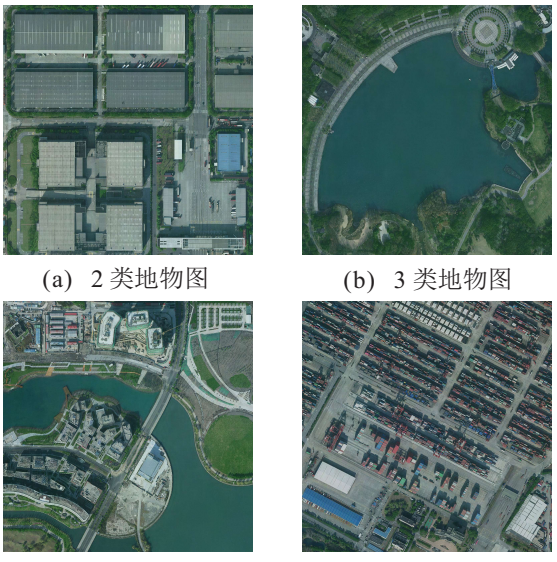
3) -XX: MaxPermSize: 持久代最大值, 6 144 MB;

4) spark.driver.maxResultSize: 影响MemoryStore的值, MemoryStore专门负责内存数据存储和读写,即负责存储RDD, 8 GB.

2.2　实验结果与分析

2.2.1　聚类数确定实验

在遥感图像分割中,聚类数k的选择是K-means算法的关键,本文采用计算代价函数的方法. 该方法首先计算所有像素点到其最近聚类中心点的平方和,并以此为测度评估聚类分割效果. 一般来说,相同迭代次数,该值越小代表聚类成本越低,但结合遥感图像分割实际情况,还需要考虑聚类结果的可解释性,不只满足使computeCost结果值最小的聚类数k. 本文选取图2中覆盖上海不同地区的WorldView-3遥感图像,其尺寸为1 500×1 500像素. 其中: 图2(a)主要包含建筑物、道路;图2(b)主要包含建筑物、植被和湖泊;图2(c)主要包含建筑物、植被、湖泊和道路;图2(d)主要包含建筑物、植被、道路、集装箱和机器设备.



(a) 2类地物图　(b) 3类地物图　(c) 4类地物图　(d) 5类地物图

图 2　上海市WorldView-3遥感图像

如图3所示,各图像均取40为最大聚类数,得到聚类数与聚类成本的关系,纵轴表示聚类成本,横轴表示聚类数. 随着聚类数的增多,聚类成本逐渐减小,当聚类数大于20以后,聚类成本都会趋于稳定. 根据肘部法则[28],图2(a)～图2(d)图像分割较为合适,聚类数分别为6, 4, 5, 4. 由此可得,实现包含不同种类地物的图像分割时聚类数有所不同,但后文实验中主要

验证RDD分区的影响,所以后文所有实验的聚类数统一设定为5.

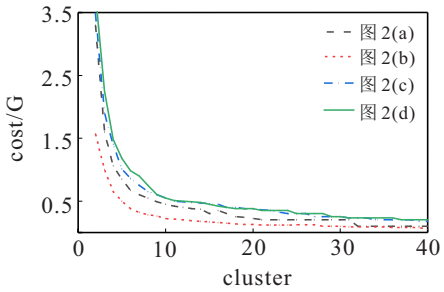


图 3 最大聚类40次代价函数统计

2.2.2 串行与并行迭代时间对比实验

本节实验旨在验证Spark平台采用并行K-means实现遥感图像分割的有效性,并将其与串行K-means算法作对比,均用Scala语言编写程序.为统一K-means算法中的主要影响参数,全文统一设定串行与并行算法的聚类数k为5,最大迭代次数为20. 图4包含10组测试图像,其中a～j为不同组的图像,1～3分别为原始图像、Spark并行K-means算法分割图像以及K-means算法串行分割图像,图4(a1)～图4(j1)对应的图像尺寸分别为876×699、$1\,987\times1\,874$、$3\,330\times2\,657$、$3\,920\times3\,843$、$5\,142\times4\,784$、$6\,117\times5\,755$、$7\,005\times6\,619$、$8\,133\times7\,555$、$8\,853\times8\,875$、$10\,314\times10\,212$.

从视觉上看,串行与并行算法分割遥感图像结果相差不大,本文将串行分割的图像按照原分割效果重新上色便于区分. 本文着重研究Spark并行K-means在分割时间上的优势,而非追求并行分割算法的高精度. 为此,计算串行与并行分割10次所用迭代时间的平均值,以此作为统计数据. 并行算法对串行算法的加速比定义为

$$T_{sp}=\frac{T_{并}}{T_{串}}. \tag{4}$$

其中:$T_{并}$表示并行算法的迭代时间,$T_{串}$表示串行算法的迭代时间.

图5所示为并行与串行执行时间和并行与串行的加速比统计图. 柱状图为不同尺寸图像利用串行和并行算法实现图像分割的迭代时间,对应的纵轴在左,折线图为串行与并行算法的加速比,对应的纵轴在右. 当图像像素数小于8.848×10^6时,串行算法在执行时间上占优势,这是因为在启动Spark平台时,需要启动相关的主从节点、Scala语言、JVM等等一系列的相关配置,如果采用并行算法分割小尺度遥感图像,则并行操作节省的时间不足以弥补程序启动消耗的时间. 当图像像素数大于8.848×10^6时,并行算法在执行时间上开始占优,并且随着图像尺度的进一

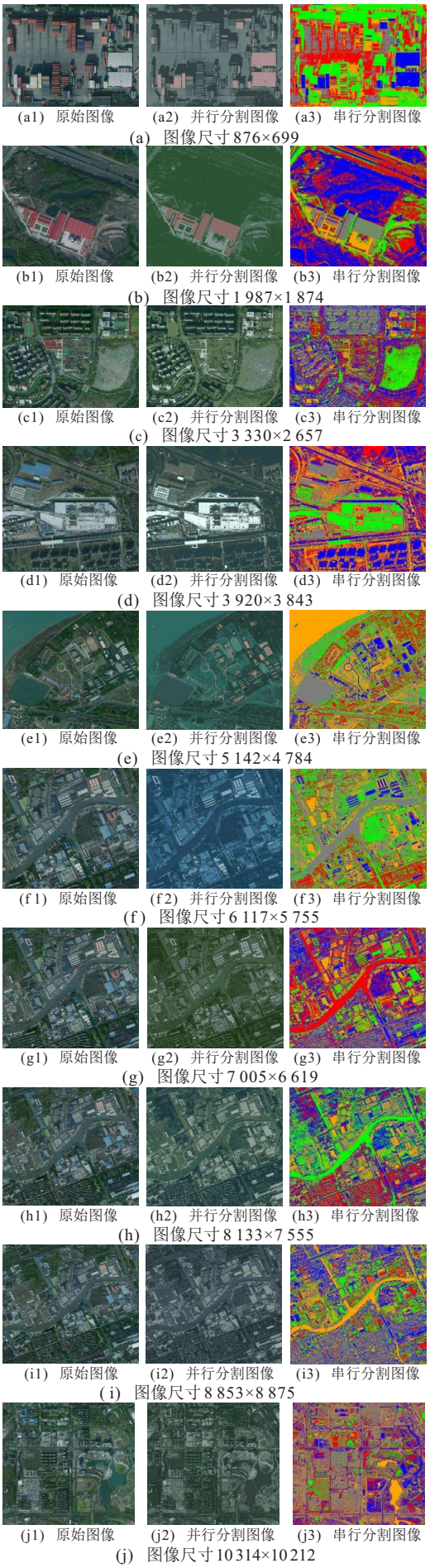


图 4 原始图像及分割结果

步增大,并行算法在时间效率上的优势越发突出,而随着图像尺度的增大串行算法的迭代时间增幅明显大于并行算法的增幅. 当图像像素数达到10^8时,并行算法的耗时有一段剧增,加速比减小,表示并行算法所能处理的图像尺寸接近最大. 当图像尺寸达到8 853×8 875像素(像素数为7.857×10^7)时,加速比达到峰值,可以看出此时Spark并行K-means算法效率最高. 而当图像尺度再增大时,加速比走向呈下坡趋势,意味着并行算法逐渐达到最大限度. 实际上,在本文的实验中,处理最大尺度图像为10 866×10 908像素,此时程序崩溃. 这是由于实验所用的个人计算机硬件配置过低,在调取不同分区数据的过程中频繁出现卡顿现象,而且程序执行多次暂停,导致即使是合理调节和分配RDD分区也无法充分利用并行算法带来的计算优势. 综合来看,就本文实验所采用的个人计算机而言,在单机上完成Spark并行K-means分割遥感图像,最大的实验数据集可以达到10^8.

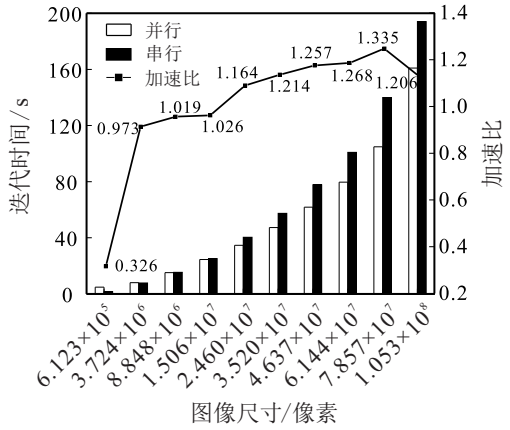


图 5　串行与并行时间统计及加速比统计

为定量评价串行和并行K-means算法的分割结果,选取图4(a)～图4(e)作为测试数据集,分别随机选取4幅图像中10%的像素点作为样本数据点,目视解译所属聚类,建立标准分割数据,分别计算串行与并行算法分割结果的混淆矩阵,并根据此矩阵分别计算出两种算法的总精度和Kappa值作为评价指标. 由表3可知,Spark并行K-means算法不仅在时间效率上优于串行K-means,且在分割精度上也占据明显优势. 但本文优化Spark RDD旨在提升K-means并行计算的速度,在分割精度方面有待进一步提升.

表 3　精度评价

数据集/px	评价指标	串行K-means	本文算法
876×699	总精度/% Kappa值	95.64 0.91	97.79 0.95
1 987×1 874	总精度/% Kappa值	94.95 0.90	97.84 0.96
3 330×2 657	总精度/% Kappa值	94.94 0.90	97.67 0.95
3 920×3 843	总精度/% Kappa值	96.17 0.91	98.17 0.96

2.2.3　RDD分区实验

本节实验的目的是验证图像分区数对程序执行时间的影响和验证优化方案的有效性,均以10次分割时间的平均值作为统计数据. 图6所示为分割图2(a)图像的平均执行时间随RDD分区数的变化趋势. 结果表明,优化前默认的RDD分区数为200,执行时间为145 s,比优化后的执行时间最高多出48 s.

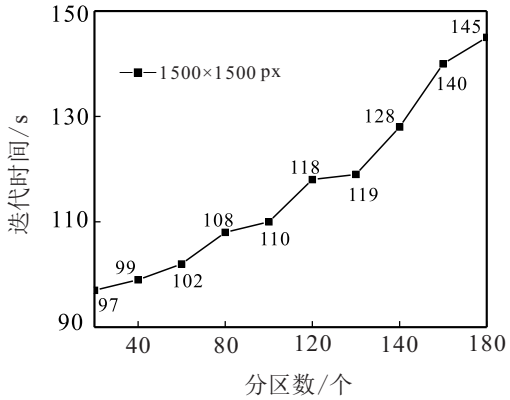


图 6　执行时间随分区数的变化趋势

将初始化聚类中心阶段的RDD分区数设置为CPU核数的1～25倍,最大达到100个分区,如图7所示. 结果表明,在Spark本地模式下,RDD分区数设定在CPU核数的1～10倍能有效减少程序执行时间,当RDD分区数大于40时,执行时间逐渐加大.

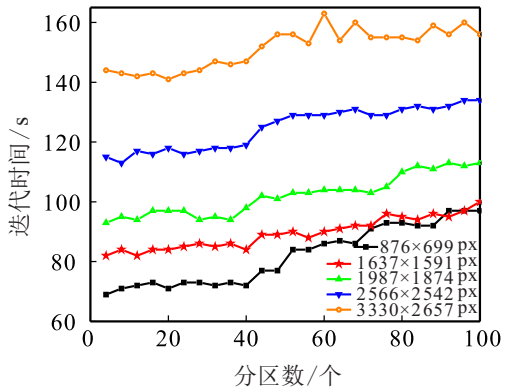


图 7　不同分区数执行时间统计

为分析图像地类数与执行时间的关系,本节实验数据选用图2中的4幅图像,将K-means算法初始化聚类中心阶段和聚类中心迭代优化计算阶段的RDD分区数设置为CPU核数的1～25倍,观察每组图像的执行时间的变化趋势,如图8所示. 由图8中曲线变化趋势可知,RDD分区数为CPU核数的1～10倍时,执行时间平均值较小,分区数越大执行时间越长. 因为硬件配置的限制,在本文实验中无法验证更大尺度图

像分割的执行时间,因此只分析了像素数在10^8以下的遥感图像分割.

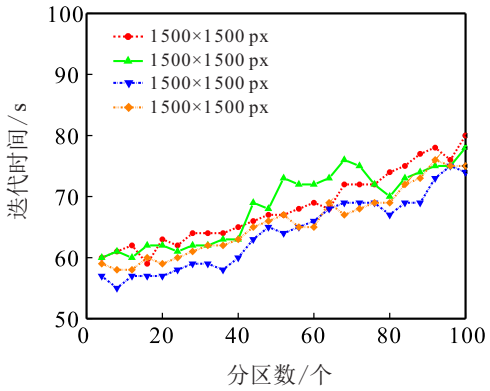


图 8 不同地类数图像分割执行时间统计

对比优化前后算法执行时间,本节选择图4(a1)～图4(e1)5幅图像做优化前后对比实验,根据Spark UI界面的显示结果统计出的时间数据如表4所示.在Job 3的时间效率上,优化后是优化前的500～1 000倍,验证了本文优化算法大大减少了Job 3阶段的执行时间,极大地提升了算法的执行效率.

表4 Job 3优化前后时间对比 s

	(a1)	(b1)	(c1)	(d1)	(e1)
优化前	222	234	231	236	231
优化后	0.2	0.3	0.3	0.4	0.3

3 结 论

为解决大尺度遥感图像的分割与分析问题,采用Spark并行K-means算法.首先借助DataFrame API加载所要定义路径下的图像文件,并生成DataFrame对象,将图像像素信息转化为数字信息,计算且并行更新指向不同聚类中心的像素隶属度,并进行迭代计算.最终,分割图像基于聚类数据进行重构.结果表明,Spark并行K-means算法实现图像分割时,迭代速度比串行K-means算法有明显提升.此外,通过设置RDD分区数,分区数为CPU核数的1～10倍程序执行总用时的平均值,比其他的分区数总用时的平均值减少了12.349%～20.61%,且在初始化聚类中心阶段的时间效率上,优化后是优化前的500～1 000倍.

参考文献(References)

[1] 王志华, 杨晓梅, 周成虎. 面向遥感大数据的地学知识图谱构想[J]. 地球信息科学学报, 2021, 23(1): 16-28.
(Wang Z H, Yang X M, Zhou C H. Geographic knowledge graph for remote sensing big data[J]. Journal of Geo-Information Science, 2021, 23(1): 16-28.)

[2] 王冰冰, 喻文勇, 龙小祥, 等. 高分辨率卫星地面处理系统研制[J]. 遥感学报, 2021, 25(9): 1946-1963.
(Wang B B, Yu W Y, Long X X, et al. Development of high-resolution satellite ground processing system[J]. Journal of Remote Sensing, 2021, 25(9): 1946-1963.)

[3] 黄基诞, 郑斐峰, 徐寅峰, 等. 基于MapReduce模型带任务分割的平行机调度优化[J]. 控制与决策, 2019, 34(7): 1514-1520.
(Huang J D, Zheng F F, Xu Y F, et al. Parallel machine scheduling with splitting jobs in MapReduce system[J]. Control and Decision, 2019, 34(7): 1514-1520.)

[4] 钱文君, 沈晴霓, 吴鹏飞, 等. 大数据计算环境下的隐私保护技术研究进展[J]. 计算机学报, 2022, 45(4): 669-701.
(Qian W J, Shen Q N, Wu P F, et al. Research progress on privacy-preserving techniques in big data computing environment[J]. Chinese Journal of Computers, 2022, 45(4): 669-701.)

[5] 黄廷辉, 王玉良, 汪振, 等. 基于内存与文件共享机制的Spark I/O性能优化[J]. 计算机工程, 2017, 43(3): 1-6.
(Huang T H, Wang Y L, Wang Z, et al. Spark I/O performance optimization based on memory and file sharing mechanism[J]. Computer Engineering, 2017, 43(3): 1-6.)

[6] Zhao Y, Dong J, Liu H W, et al. Improving cache management with redundant RDDs eviction in spark[J]. Computers, Materials & Continua, 2021, 68(1): 727-741.

[7] Zaharia M, Chowdhusry M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]. USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2012: 141-146.

[8] Zaharia M, Das T, Li H, et al. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters[C]. Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing. USENIX Association, 2012: 10.

[9] 陈天宇, 张龙信, 李肯立, 等. Spark框架中RDD缓存替换策略优化[J]. 小型微型计算机系统, 2019, 40(6): 1248-1253.
(Chen T Y, Zhang L X, Li K L, et al. Optimization of RDD cache replacement strategy optimization in spark framework[J]. Journal of Chinese Computer Systems, 2019, 40(6): 1248-1253.)

[10] 魏赟, 丁宇琛. Spark中一种高效RDD自主缓存替换策略研究[J]. 计算机应用研究, 2020, 37(10): 3043-3047.
(Wei Y, Ding Y C. Research on efficient RDD self-cache replacement strategy in Spark[J]. Application Research of Computers, 2020, 37(10): 3043-3047.)

[11] 赵俊先, 喻剑. 基于RDD非序列化本地存储的Spark存储性能优化[J]. 计算机科学, 2019, 46(5): 143-149.
(Zhao J X, Yu J. Optimization of spark RDD based on

non-serialization native storage[J]. Computer Science, 2019, 46(5): 143-149.)

[12] 刘恒, 谭良. 并行计算框架Spark中一种新的RDD分区权重缓存替换算法[J]. 小型微型计算机系统, 2018, 39(10): 2279-2284.
(Liu H, Tan L. New RDD partition weight cache replacement algorithm in spark[J]. Journal of Chinese Computer Systems, 2018, 39(10): 2279-2284.)

[13] 翟俊海, 田石, 张素芳, 等. 基于MapReduce和Spark的大数据模糊K-means算法比较[J]. 河北大学学报: 自然科学版, 2020, 40(4): 433-440.
(Zhai J H, Tian S, Zhang S F, et al. A comparison on big data fuzzy K-means algorithm based on MapReduce and Spark[J]. Journal of Hebei University: Natural Science Edition, 2020, 40(4): 433-440.)

[14] 卞琛, 于炯, 英昌甜, 等. 并行计算框架Spark的自适应缓存管理策略[J]. 电子学报, 2017, 45(2): 278-284.
(Bian C, Yu J, Ying C T, et al. Self-adaptive strategy for cache management in spark[J]. Acta Electronica Sinica, 2017, 45(2): 278-284.)

[15] 王玉, 李玉, 赵泉华. 基于区域的多尺度全色遥感图像分割[J]. 控制与决策, 2018, 33(3): 535-541.
(Wang Y, Li Y, Zhao Q H. Region-based multiscale segmentation of panchromatic remote sensing image[J]. Control and Decision, 2018, 33(3): 535-541.)

[16] 陈玉明, 蔡国强, 卢俊文, 等. 一种邻域粒K均值聚类方法[J]. 控制与决策, 2023, 38(3): 857-864.
(Chen Y M, Cai G Q, Lu J W, et al. A neighborhood granular K-means clustering method[J]. Control and Decision, 2023, 38(3): 857-864.)

[17] 孟红涛, 余松平, 刘芳, 等. Spark内存管理及缓存策略研究[J]. 计算机科学, 2017, 44(6): 31-35.
(Meng H T, Yu S P, Liu F, et al. Research on memory management and cache replacement policies in spark[J]. Computer Science, 2017, 44(6): 31-35.)

[18] 黄文辉, 冯瑞. 基于Spark Streaming的视频/图像流处理与新的性能评估方法[J]. 计算机工程与科学, 2015, 37(11): 2055-2060.
(Huang W H, Feng R. A novel performance evaluation method for processing video/image stream based on Spark Streaming[J]. Computer Engineering and Science, 2015, 37(11): 2055-2060.)

[19] Maia D, Trindade R. Face detection and recognition in color images under Matlab[J]. International Journal of Signal Processing, Image Processing and Pattern Recognition, 2016, 9(2): 13-24.

[20] Liu B, He S R, He D J, et al. A Spark-based parallel fuzzy C-Means segmentation algorithm for agricultural image bigdata[J]. IEEE Access, 2019, 7(7): 42169-42180.

[21] 景维鹏, 霍帅起. 基于自定义RDD的海量遥感图像并行镶嵌方法[J]. 地球信息科学学报, 2017, 19(10): 1346-1354.
(Jing W P, Huo S Q. A model of parallel mosaicking for massive remote sensing images based on self-defined RDD[J]. Journal of Geo-Information Science, 2017, 19(10): 1346-1354.)

[22] Jiang K, Du S F, Zhao F, et al. Effective data management strategy and RDD weight cache replacement strategy in Spark[J]. Computer Communications, 2022, 194(194): 66-85.

[23] Xu Y N, Liu H, Long Z H. A distributed computing framework for wind speed big data forecasting on Apache Spark[J]. Sustainable Energy Technologies and Assessments, 2020, 37: 100582.

[24] Singh P, Singh S, Mishra P K, et al. A data structure perspective to the RDD-based apriori algorithm on Spark[J].International Journal of Information Technology, 2022, 14(3): 1585-1594.

[25] Wang N, Chen F, Yu B, et al. A Strategy of parallel SLIC super pixels for handling large-scale images over Apache Spark[J]. Remote Sensing, 2022, 14(7): 1568-1568.

[26] Li X H, Yu B W, Feng G Y, et al. LotusSQL: SQL engine for high-performance big data systems[J]. Big Data Mining and Analytics, 2021, 4(4): 252-265.

[27] Ramírez-Gallego S, García S, Benítez J M, et al. Aistributed evolutionary multivariate discretizer for big data processing on Apache Spark[J]. Swarm and Evolutionary Computation, 2018, 38(38): 240-250.

[28] 周玉, 朱文豪, 孙红玉. 一种基于目标函数的局部离群点检测方法[J]. 东北大学学报: 自然科学版, 2022, 43(10): 1405-1412.
(Zhou Y, Zhu W H, Sun H Y. A local outlier detection method based on objective function[J]. Journal of Northeastern University: Natural Science, 2022, 43(10): 1405-1412.)

作者简介

李玉 (1963−), 男, 教授, 博士生导师, 从事基于生物视觉、信息几何、大数据理论等研究, E-mail: liyu@lntu.edu.cn;

崔书琳 (1998−), 女, 硕士生, 从事遥感影像处理的研究, E-mail: jc_211130@163.com;

赵泉华 (1978−), 女, 教授, 博士生导师, 从事随机几何、 模糊集在遥感图像建模和解译算法等研究, E-mail: zhaoquanhua@lntu.edu.cn.