Journal of System Simulation

Volume 30 | Issue 4 Article 32

1-4-2019

Parallel Pattern Recognition of Leak Current Data Using Spark-KNN

Li Li
School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China;

Yongli Zhu
School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China;

Yaqi Song
School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China;

Follow this and additional works at: https://dc-china-simulation.researchcommons.org/journal

 Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, Numerical Analysis and Scientific Computing Commons, Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Science Commons

This Paper is brought to you for free and open access by Journal of System Simulation. It has been accepted for inclusion in Journal of System Simulation by an authorized editor of Journal of System Simulation.

Parallel Pattern Recognition of Leak Current Data Using Spark-KNN

Abstract

Abstract: With the rapid development of smart grid, the status monitoring data of power grid equipment increase exponentially and gradually form the big data. Traditional computing architectures are no longer to meet the demand of computing performance. *This paper explores how Spark and Cloud computing can accelerate performance of missive insulator leak current data pattern recognition.* The Parallel KNN (k-Nearest Neighbor) algorithm is designed and implemented by using Spark and Aliyun E-MapReduce cloud computing platform. The results from experiments show that the performance of Spark-KNN is 2.97 times of MapReduce-KNN and gains acceleration of 8.8 times. The experimental results confirm that Spark is more suitable for real time data processing tasks than MapReduce.

Keywords

power grid equipment, online monitoring, big data, Spark

Recommended Citation

Li Li, Zhu Yongli, Song Yaqi. Parallel Pattern Recognition of Leak Current Data Using Spark-KNN[J]. Journal of System Simulation, 2018, 30(4): 1473-1481.

This paper is available in Journal of System Simulation: https://dc-china-simulation.researchcommons.org/journal/vol30/iss4/32

第 30 卷第 4 期 系统仿真学报© Vol. 30 No. 4
2018 年 4 月 Journal of System Simulation Apr., 2018

泄漏电流数据的 Spark-KNN 并行模式识别方法

李莉，朱永利，宋亚奇
（华北电力大学控制与计算机工程学院，河北 保定 071003）

摘要：随着智能电网的快速发展，电网设备状态监测数据呈指数级增长，逐渐构成电网设备状态监测大数据。传统计算架构已无法满足计算性能需求。*结合 Spark 大数据处理技术和阿里云 E-MapReduce 云计算平台，提出电网设备状态监测大数据并行模式识别方法，旨在提升电网设备在线监测系统对短时间内骤增的报警监测数据快速批量分析的能力。*设计了基于 Spark 的并行化 k 最近邻分类算法(k-Nearest Neighbor，KNN)Spark-KNN，实现了海量绝缘子泄漏电流数据的并行模式识别。实验结果表明，Spark-KNN 的平均性能是 Hadoop MapReduce 实现的 2.97 倍，获得了最高 8.8 倍的加速比，更适合执行电力设备监测大数据的实时处理任务。

关键词：电网设备；在线监测；大数据；Spark
中图分类号：TM764 文献标识码：A 文章编号：1004-731X (2018) 04-1473-09
DOI: 10.16182/j.issn1004731x.joss.201804032

Parallel Pattern Recognition of Leak Current Data Using Spark-KNN

Li Li, Zhu Yongli, Song Yaqi
(School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China)

Abstract: With the rapid development of smart grid, the status monitoring data of power grid equipment increase exponentially and gradually form the big data. Traditional computing architectures are no longer to meet the demand of computing performance. *This paper explores how Spark and Cloud computing can accelerate performance of missive insulator leak current data pattern recognition.* The Parallel KNN (k-Nearest Neighbor) algorithm is designed and implemented by using Spark and Aliyun E-MapReduce cloud computing platform. The results from experiments show that the performance of Spark-KNN is 2.97 times of MapReduce-KNN and gains acceleration of 8.8 times. The experimental results confirm that Spark is more suitable for real time data processing tasks than MapReduce.

Keywords: power grid equipment; online monitoring; big data; Spark

引言

电网设备状态监测正在从单一参数监测向全方位、群设备监测发展，数据量成几何增长趋势。大规模海量监测数据将涌向远程电网设备监测中心，使之面临繁重的数据收集、处理、存储和分析任务。尤其是暴露于户外的设备监测，如输电线路状态监测等，受天气影响极大。在极端天气或连锁式故障情况下，电网设备或装置由于监测值越限而频繁向监测中心发送报警数据，将在短时间内骤增。当前数据若得不到快速识别和诊断，随着报警的陆续增加，会造成分析任务和数据的堆积，甚至数据丢失，这对监测系统的性能提出了更高要求。

收稿日期：2016-05-11 修回日期：2016-07-15;
基金项目：国家自然科学基金(51677072)，河北省自然科学基金(A2016502001)，中央高校基本科研业务费专项资金(2018MS074)；
作者简介：李莉(1980-)，女，重庆，硕士，讲师，研究方向为现代信号处理方法在电力系统故障诊断等方面的应用。

Published by Journal of System Simulation, 2018

Journal of System Simulation, Vol. 30 [2018], Iss. 4, Art. 32

传统的单机环境下，使用单任务方式，对小样本数据量适用，但遇到紧急或特殊情况，当样本数据量急剧增大后，对存储和运算要求突然提高，一般很难在有限的时间内处理完成，甚至出现无法处理宕机等情况[1]。而目前通用的 Hadoop MapReduce[2]技术，虽然可以有效处理大数据，但针对需要多次循环迭代的输变电设备状态评价计算分析任务，需要频繁的磁盘 I\O 操作，无法在短时间内完成对大量的越限报警数据的分析和设备状态识别，实时性难以满足要求[3]，需要借助内存并行技术加快数据分析的速度。

Spark[4]是一款基于内存计算的大数据并行计算框架。Spark 基于内存计算，使用弹性分布式数据集 RDD(Resilient Distributed Datasets)作为数据载体，并基于 RDD 提供了多种便捷的操作，相比 MapReduce，可以大幅提高数据处理性能，同时保证高容错性和高可扩展性。Spark 与 Hadoop 兼容并且支持多种计算模式，包括流、以图形为核心的操作、SQL 访问以及分布式机器学习等。

很多分类算法都可以用于绝缘子泄漏电流的分类，笔者选择 KNN 算法的主要依据是：1) KNN 属于经典分类算法，应用广泛，被各领域所熟知。为了体现云计算平台处理电力大数据的优势，选择了经典算法，而并非非常新的机器学习算法。2) KNN 算法本身适合利用 Spark 实现并行化(数据并行)。并非所有的机器学习算法都适合在 Spark 上实现。KNN 分类算法[5]具有简洁、参数估计简单等特点，适合对稀有事件、多分类问题进行分类，广泛应用于电力系统数据分析中。

本文基于 Spark 并行计算框架，开展了电力设备状态快速模式识别技术的研究，设计实现了并行化的快速 KNN 算法 Spark-KNN，并以输电线路覆冰绝缘子泄漏电流数据为例，实现了绝缘子状态的快速类型识别。

1 相关工作

对于电力设备监测大数据处理和管理所面临的挑战，目前公认的最有效的解决方法之一是云计算技术。文献[6-7]研究了基于 Hadoop 的广域测量系统 WAMS(Wide Area Measurement System)数据处理平台。文献[8]针对电能质量监测数据海量化问题，基于 Hadoop 大数据技术设计实现了一种层次化的电压暂降并行计算方法，以提高计算效率。文献[9]针对智能配电网的海量数据集，基于 MapReduce 框架设计实现了变断面量测数据的并行化无损压缩。文献[10-11]针对电力系统智能化带来的数据海量化、高维化的问题，采用 MapReduce 框架设计实现了并行化的极限学习机，用于短期电力负荷预测。文献[12]针对海量用户侧电力数据(智能电表数据)，利用 MapReduce 设计了并行化的数据分析和负荷预测。文献[13-14]针对海量电能质量监测数据的查询问题，使用 MapReduce 框架设计实现了并行数据分析算法。文献[15]基于 MapReduce 框架设计实现了并行化的贝叶斯分类器，用于变压器的故障诊断，诊断速度高于传统单机环境下的诊断速度。文献[16]基于 HBase 研究了智能电网时间序列数据的存储和处理方法。

已发表的研究成果大多是以 Hadoop 为平台的。这些系统的架构较为相似，存储层普遍采用 HDFS 和 HBase；计算层采用 MapReduce；控制层采用 Hive、Pig 等高层接口完成计算任务的传递；在应用层运行状态评估、故障诊断等各类监测系统应用。Hadoop 平台在进行海量数据分析时，响应时间往往达到小时级。如果需要对不断到达的巨量监测数据进行实时分析，Hadoop 是不能胜任的。

2 基于 Spark 的状态监测数据并行模式识别

2.1 监测数据在 RDD 中的分布式存储

Spark 的数据处理是建立在统一抽象的 RDD 之上，并以基本一致的方式应对各种数据处理场景，包括 MapReduce，SQL 查询，流计算，机器学习以及图计算等。RDD 是一个容错的、并行的

https://dc-china-simulation.researchcommons.org/journal/vol30/iss4/32
DOI: 10.16182/j.issn1004731x.joss.201804032

数据结构，可以显式地将数据存储到磁盘和内存中，并能控制数据的分区。相对于 MapReduce 编程模式，RDD 通过提供包括 map、flatMap、filter、join、groupBy、reduceByKey 等算子来操作数据，使得编写并行程序更加容易。

电力设备监测的波形数据或者经特征提取之后的特征样本，在执行模式识别之前，以 RDD 的方式分布式存储在 Spark 集群的各数据节点中。RDD 可以被理解为一个大的数组，但这个数组是分布在集群上的。RDD 在逻辑上是由多个分区(Partition)组成的。Partition 在物理上对应某个数据节点上的一个内存存储块。执行 KNN 模式识别的过程，就是对 RDD，使用一系列 Spark 算子，进行转换，最终获得类别的过程。监测数据在 RDD 中的存储如图 1 所示。

在图 1 中，RDD1 包含 4 个 Partition(P1、P2、P3、P4)，分别存储在 3 个节点(Worker Node1、Worker Node2、Worker Node3)中。RDD2 包含 2 个 Partition(P1，P2)，分别存储在 2 个节点(Worker Node3、Worker Node1)中。

2.2 Spark-KNN 快速模式识别算法

KNN 算法的基本思想是：如果一个样本在特征空间中的K个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。由于 KNN 方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN 方法较其他方法更为适合。

模式的建立过程如下：建立并维护一个大小为 K 的、按距离由大到小的优先级队列，用于存储最近邻训练样本。随机从训练样本中选取 K 个样本作为初始的最近邻样本，分别计算测试样本到这 K 个样本的距离，将训练样本标号和距离存入优先级队列。遍历训练样本集，计算当前训练样本与测试样本的距离，将所得距离 L 与优先级队列中的最大距离 L_{max} 进行比较。若 $L>=L_{max}$，则舍弃该样本，遍历下一个样本。若 $L<L_{max}$，删除优先级队列中最大距离的样本，将当前训练样本存入优先级队列，直至遍历完成。

在优先级队列更新并确定后，计算优先级队列中 K 个样本的多数类，并将其作为测试样本的类别，完成模式识别过程。

Spark-KNN 算法的输入、输出数据可以使用本地文件系统，或者 HDFS；如果使用其他存储介质，如阿里云 OSS 等，则需要自行编写输入和输出代码部分。Spark-KNN 算法描述如下：

算法输入：训练样本集 TrainSet；待测样本集 TestSet；结果集 ResultSet 路径；参数 K；

算法输出：结果集 ResultSet。

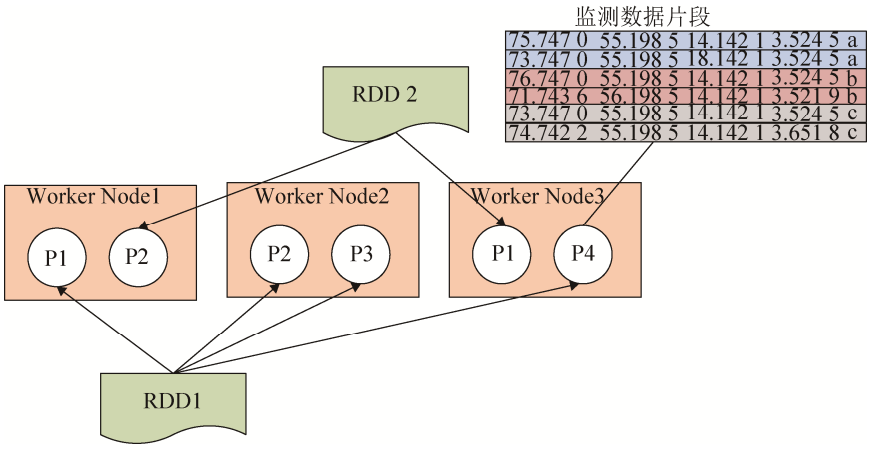


图 1 监测数据在 RDD 中的分布式存储

Fig. 1 Distributed storage of monitoring data in RDD

Journal of System Simulation, Vol. 30 [2018], Iss. 4, Art. 32

算法过程：

(1) 初始化 SparkContext 环境参数：Spark 集群 Master 节点、使用资源规模等；

(2) 加载训练样本集 TrainSet 到 RDD，在 Spark 集群的节点的内存中分布式存储 TrainSet；执行 RDD.map()算子，并行完成 TrainSet 的格式转换，转换结果为多元组形式。RDD.map()算子代码如下：

map(line => {vardatas = line.split(" ") (datas(0), datas(1), datas(2))})

(3) 执行 RDD.collect()算子，将分布式的 RDD 返回到 Driver 程序所在的节点，以 scala Array 数组形式存储，命名为 TrainSet_Array；

(4) 由于待测样本集是分布式存储的，为了计算一条待测样本和 TrainSet 中各样本的距离，需要利用广播(broadcast)算子 SparkContext.broadcast()将 TrainSet_Array 发送到集群中的各个数据节点中，命名为 trainDatas。broadcast 的作用类似于 Hadoop 的 distributed cache，但 broadcast 的内容可以跨作业共享。

(5) 利用广播(broadcast)算子 SparkContext.broadcast()将 KNN 参数 K 发送到集群中的各个数据节点中。

(6) 加载待测样本集 TestSet 到 RDD，在 Spark 集群的节点的内存中分布式存储 TestSet；执行 RDD.map()算子，并行完成 TrainSet 的格式转换，结果为多元组形式。

(7) 对转换后的 TestSet RDD 执行 map()算子，执行并行化的映射，将单条测试样本映射为结果样本(带标记的样本)。map()算子过程描述如下：

1) 解析一条测试样本元组，提取各特征量；

2) 使用 foreach 算子，循环计算测试样本到训练样本的距离：

distanceset=trainDatas.foreach(trainData=>(特征，距离，类别)})

3) 按照距离递增顺序，对 distanceset 排序；

4) 定义映射 varcategoryCountMap= Map[String, Int]()，使用 categoryCountMap.foreach 算子，统计前 K 个样本的类别。

(8) 将结果输出至 HDFS 或者其他持久化存储系统(如，HBase 等)。

在上述算法步骤(3)中，执行 collect 算子，将分布式 RDD 返回至本地节点的主要原因是：识别过程中，需要计算未知样本与训练集全集的样本之间的距离，因此需要先将训练数据 RDD 存储到本地，在利用 broadcast 广播到集群中的各个节点，才能完成并行的的距离计算。同理，参数 *K* 也需要广播到各节点。在算法步骤(2)、(6)中，执行格式转换，是为了样本数据转换成多元组的形式，有利于距离的计算。

2.3 Spark-KNN 算法的 RDD 数据处理流程

Spark-KNN 算法的执行过程是建立在统一抽象的 RDD 之上的，是通过 RDD 的各类算子进行转换的过程。算法的数据处理流程如图 2 所示。

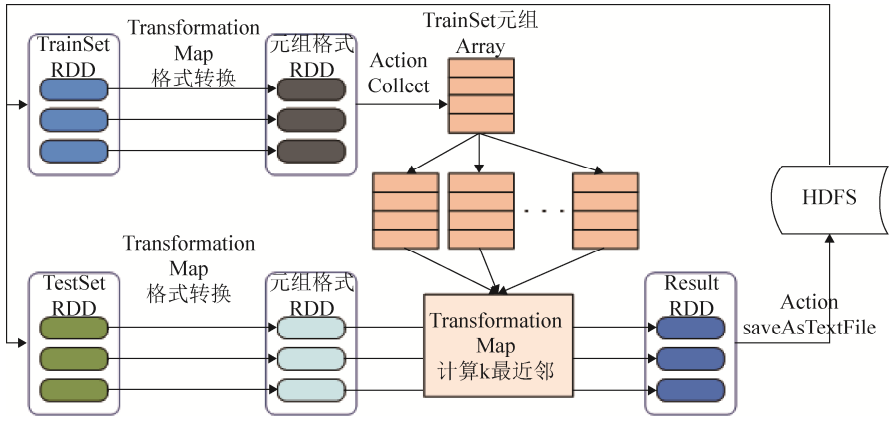


图 2 Spark-KNN 数据处理流程

Fig. 2 Data processing flow of Spark-KNN

https://dc-china-simulation.researchcommons.org/journal/vol30/iss4/32
DOI: 10.16182/j.issn1004731x.joss.201804032

在图 2 中，数据来源于 HDFS，使用 Spark Context 的 textFile()方法读取训练集和测试集文件，并将数据组织为 RDD 的形式。格式转换操作通过 map 算子完成。map 对 RDD 中的每个元素都执行一个指定的函数来产生一个新的 RDD。任何原 RDD 中的元素在新 RDD 中都有且只有一个元素与之对应。Collect 算子是 Action 类型的算子，用于将分布式的 RDD 返回到 Driver 程序所在的节点，以 scala Array 数组形式存储。broadcast 算子是 Action 类型的算子，用于将 Driver 节点上的数据广播到各个 Worker 所在的节点；saveAsTextFile 算子用于将 RDD 存储于 HDFS。

2.4 基于 Hadoop MapReduce 的并行化 KNN 算法 MR-KNN

MapReduce 是目前流行的并行编程框架。为了对比 Spark 和 MapReduce 在实现海量监测数据模式识别的性能，本文也设计实现了基于 Hadoop MapReduce 的并行化 KNN 算法 MR-KNN。

本文设计的算法假设 KNN 的训练集作为缓存文件在每个节点上共享。测试集文件分块存储于 HDFS。Map 过程中，测试集的样本将逐条输入至 map 函数，在 map 函数中完成测试样本和训练样本距离的计算，并对距离进行排序，将距离最短的 K 个训练样本的类别输出至 Reduce。在 Reduce 阶段完成类别的频率统计，并将频率最高的类别作为本次的分类结果，MR-KNN 算法描述如下：

(1) 输入：<key_1, $value_1$>；key_1 是训练样本 ID，$\boldsymbol{value_1}$ 是训练样本值，可以用元组表达 $value_1 = (v_1, v_2, ..., v_N)$；

(2) 输出：<key_3, $value_3$>；key_3 是训练样本 ID，$\boldsymbol{value_3}$ 是训练样本值和类别，可以用元组表达 $value_3 = (v_1, v_2, ..., v_N, C)$；其中，$C$ 表示样本的类别；

(3) Setup 过程：利用 DistributedCache 类(由 Hadoop 提供)，将训练集和参数 K 缓存到各个数据节点的内存；

(4) Map：

计算测试样本和训练样本的距离；

并对距离进行排序，将距离最短的 K 个训练样本的类别输出；

(5) Reduce：

统计类别频率，将样本值和频率最高的类别组织为 $value_3$ 输出；

在上述算法的步骤(3)中，发送训练集合参数 K 只需要执行一次，因此放在 Setup 过程中(该过程只在程序开始时执行一次)实现；而 Map 和 Reduce 会被多次调用。

3 实验与结果分析

3.1 实验环境搭建

在阿里云云计算平台上，使用 E-MapReduce 服务创建了包含 5 台 ECS 服务器的 Spark 集群，部署方式采用目前流行的 Spark on YARN 模式，用于运行所设计的 Spark-KNN 算法。硬件配置如下：

(1) Master 节点(1 个)

带宽：8 M；CPU：4 核；内存：8G；硬盘类型：SSD 云盘；硬盘容量：40G；

(2) Core 节点(4 个)

带宽：8 M；CPU：4 核；内存：8 G；硬盘类型：SSD 云盘；硬盘容量：40 G；

系统软件配置如下：

主版本：EMR 1.0.0

软件信息：hive 1.0.1；ganglia 3.7.2；Spark 1.4.1；yarn 2.6.0；pig 0.14.0；

上述软件部署之后，集群既可以运行 Mapreduce 程序，又可以运行 Spark 程序。其中的 ganglia 3.7.2 程序主要用于集群硬件资源的利用率，依据其提供的 CPU、内存利用率等监控数据，可以调整、优化并行任务配置参数，如根据 CPU 利用率调整 Spark 作业的 number-exector 的数量等，使集群性能充分发挥。

3.2 实验数据

本文以输电线路监测中覆冰绝缘子泄漏电流

Journal of System Simulation, Vol. 30 [2018], Iss. 4, Art. 32

数据模式识别为例，应用所设计的 Spark-KNN 算法进行绝缘子泄漏电流数据的快速模式识别。针对绝缘子泄漏电流特征量提取及应用方面的研究开展得已经非常广泛，本文选取泄漏电流的最大值、进行傅里叶变换后的 50 Hz 幅值、150 Hz 幅值以及 250 Hz 幅值构成 4 维的特征量，用于模式识别。从训练集中选取部分样本特征，如表 1 所示。

表 1 训练集样本
Tab. 1 Samples of the training set

类别	最大值/mA	50 Hz 幅值/mA	150 Hz 幅值/mA	250 Hz 幅值/mA
A	14.893 6	12.470 7	0.108 2	0.101 6
A	18.013 6	14.707 5	0.896 2	0.117 5
A	59.191 9	44.004 0	11.551 1	2.728 6
B	87.625 1	63.240 5	15.729 9	3.748 1
B	92.732 0	68.575 9	17.161 2	4.275 6
C	138.928 7	102.311 6	20.603 1	5.895 2
E	20 781	1 603	348	161

覆冰绝缘子泄漏电流样本的类别[17]描述如表 2 所示。

表 2 覆冰绝缘子泄漏电流样本类别
Tab. 2 Sample category of leakage current of ice covered insulators

类别	类别描述
A	泄漏电流比较小，观察不到放电现象，或者高压端绝缘子钢角处出现蓝紫色局部电晕放电，绝缘子各处出现零星的放电现象。
B	蓝紫色局部放电变成明黄色放电，局部放电明显增多，局部电弧增长，冰层开始小范围脱落，泄漏电流幅值增长迅速。
C	接地端电弧向高压端发展，高压端电弧向上发展的趋势更明显，发展成高温白色电弧，融冰现象加剧，泄漏电流大幅度增大。
D	接地端和高压端白色电弧发展到一定长度，有即将贯穿整串绝缘子的趋势，融冰大量脱落，泄漏幅度值可达 2 000 mA。
E	白色电弧贯穿，发展成闪络。

实验数据来源于人工覆冰实验和实测绝缘子泄漏电流数据，并对数据进行了复制，以模拟产生出大规模的报警或者越限数据。

在数据规模上，本文模拟 600 万个监测量(超过 100 座变电站的监测量规模)的情形。通过设置设备的故障率(0—100%)，模拟由于恶劣天气或设备故障发展阶段等情况下所产生的不同规模的报警数据。短时间内，需要处理的报警数据的规模在 0~600 万条范围内。本次实验中，仅使用了不同规模的泄漏电流数据验证 Spark-KNN 的模式识别性能，暂未考虑对多源异构数据综合使用多种模式识别算法的情况。

实验用数据集包括训练集和测试集，见表 3。

表 3 数据集
Tab. 3 Data set

训练集 ID	样本数量(条)	测试集 ID	故障率/%	样本数量(万条)
T1	50	C1	10	60
T2	500	C2	30	180
T3	1 000	C3	50	300
		C4	80	480
		C5	100	600

3.3 Spark-KNN 性能测试

算法执行时间的横向对比以及并行算法的加速比能够很好的描述算法的性能。设计实验方案，将 Spark-KNN 与 MR-KNN 在相同的硬件集群下进行执行时间的对比分析，并计算了 Spark-KNN 算法的加速比。使用表 3 所示数据集对 Spark-KNN 算法进行性能测试。分别在单机环境下、Hadoop 集群环境下和 Spark 环境下，执行 KNN、MR-KNN 和 Spark-KNN 程序，对比运行时间。其中，MR-KNN 和 Spark-KNN 运行的硬件环境相同。

3.3.1 单机环境下的 KNN 处理性能测试

在单机环境下(4 核 CPU，8 GB 内存，40 GB 高效云磁盘)执行 KNN，算法运行时间随数据规模的变化如图 3 所示。

从图 3 中可以看出，在训练样集为 T1(50 条样本)时，KNN 分类执行均可以控制在 5 分钟以内，但当训练集为 T2(500 条样本)时，KNN 执行时间明显增长，测试集为 C2 时，执行时间为 8.8 分钟；当测试集为 C5(600 万条样本)时，执行时间接近半

https://dc-china-simulation.researchcommons.org/journal/vol30/iss4/32
DOI: 10.16182/j.issn1004731x.joss.201804032

小时，工程实用性差；当训练集选择 T3 时，单机环境下运行时间过长，任务出现“假死”现象，在图 3 中未绘制 T3 曲线。上述实验结果表明，单机环境下无法胜任大规模报警数据的快速模式识别任务。

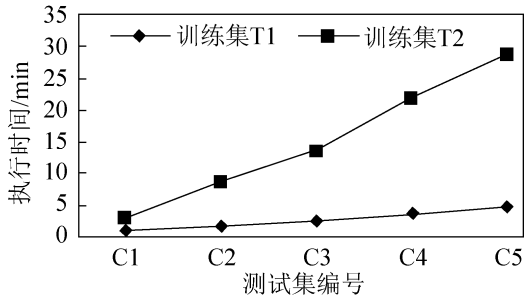
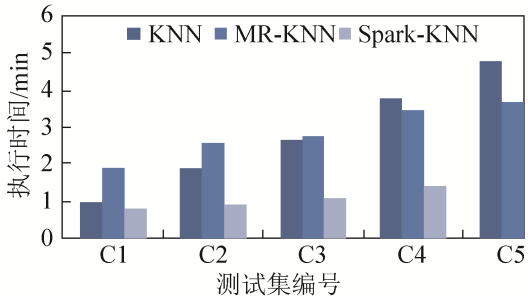


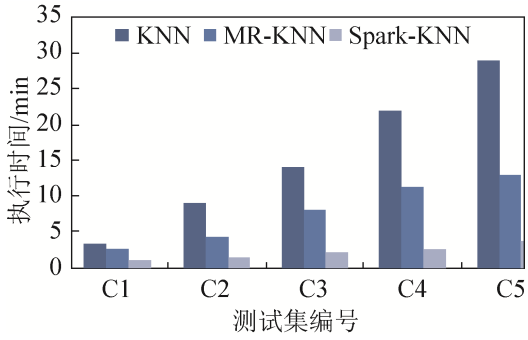
图 3　单机环境下 KNN 算法执行时间
Fig. 3　Execution time of KNN algorithm in a single machine environment

3.3.2 集群环境下并行化 KNN 性能测试

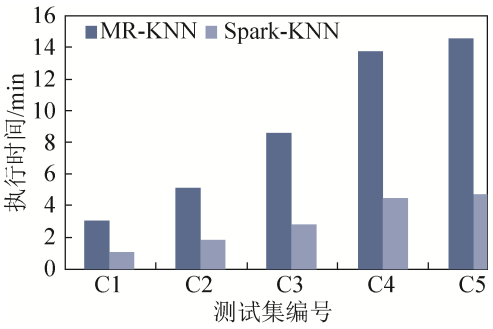
在所搭建的集群上，分别运行 Spark-KNN 和 MR-KNN，对不同数据规模情况下的运行时间进行对比，结果如图 4 所示。图 4(a)使用了训练集 T1，图 4(b)使用了训练集 T2，图 4(c)使用了训练集 T3。测试集均使用了 C1—C5。



(a) 训练集 T1



(b) 训练集 T2



(c) 训练集 T3

图 4　Spark-KNN 和 MR-KNN 的执行时间对比
Fig. 4　Comparison of execution time between Spark-KNN and MR-KNN

从图 4 中可以看出，使用不同规模的训练集，Spark-KNN 性能均优于 MR-KNN。在本次实验中，一次作业执行时，MR-KNN 与 Spark-KNN 执行时间的比值，最大值可以达到 4.8，最小值为 2.3。Spark-KNN 的平均性能是 MR-KNN 的 2.97 倍。

在不同规模的训练集上(T1，T2，T3)，Spark-KNN 的性能均优于单机环境，而 MR-KNN 在数据量较少时(图 4 中，T1C1、T1C2、T1C3)，受磁盘读写、节点间通信开销影响，MR-KNN 执行速度慢于单机。Spark-KNN 高性能的主要原因在于，在程序执行之初，样本数据一次性加载至内存，并在之后的迭代计算中，始终保持在内存中，避免了 MR-KNN 过程中的反复磁盘读写，从而保证了数据的高效处理。

Spark-KNN 在不同数据集上的运行时间变化趋势如图 5 所示。

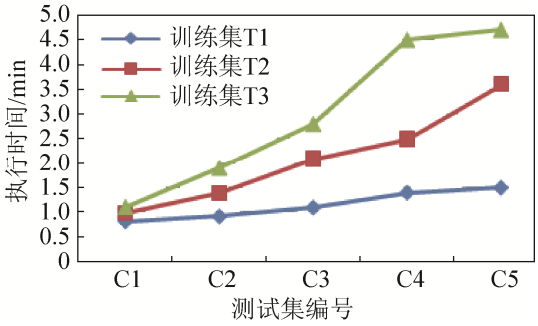


图 5　Spark-KNN 执行时间趋势
Fig. 5 The execution time trend of Spark-KNN

从图 5 中可以看出，随着测试集规模的增长，

Journal of System Simulation, Vol. 30 [2018], Iss. 4, Art. 32

算法运行时间增长缓慢(远低于线性增长)，对于不同规模的训练集，算法整体运行时间平稳，未出现大的波动。

Spark 程序运行时，使用 Spark on YARN 的运行模式，环境参数的配置至关重要，这些参数决定集群是否能够充分发挥其计算性能，会对程序执行性能有很大的影响。本次实验中，在创建 Spark 作业时，配置的参数值和参数说明见表 4。

如果系统硬件配置发生变化，上述作业参数也需要及时调整。另外，还需要根据 ganglia 或者其他硬件监控程序提供的监测参数(cpu、内存利用率等)，适当调整作业参数。

表 4 Spark 作业参数配置
Tab.4 Spark job parameter configuration

参数名称	参数说明	系统默认值	配置值
--executor-memory	Memory per executor	1 GB	2 GB
--driver-memory	Memory for driver	512 MB	1 GB
--num-executors	Number of executors to launch	2	4
--executor-cores	Number of cores per executor	1 in YARN mode	4

3.3.3 Spark-KNN 算法加速比测试

加速比(speedup)是用来衡量并行系统或程序并行化的性能和效果的重要参数，本次实验通过调整集群规模，控制参与计算的 CPU 核心数，重复执行 Spark-KNN 算法，获得执行时间，用以计算 Spark-KNN 的加速比。加速比的计算方法是：同一个任务在单处理器系统和并行处理器系统中运行时间的比率，如公式(1)所示。

$$S_{\text{peedup}} = T_s / T_h \tag{1}$$

其中：T_s 是算法在单个 CPU 核心环境下的执行时间，T_h 是算法在 h 个 CPU 核心环境下时的执行时间。使用训练集(T1，T2)和测试集(C1，C2，C3，C4，C5)进行组合，形成 10 种不同数据量下的加速比，结果如图 6 所示。T1C1 表示使用训练集 T1 和测试集 C1 执行算法，计算执行时间和加速比，其他组合含义相同。

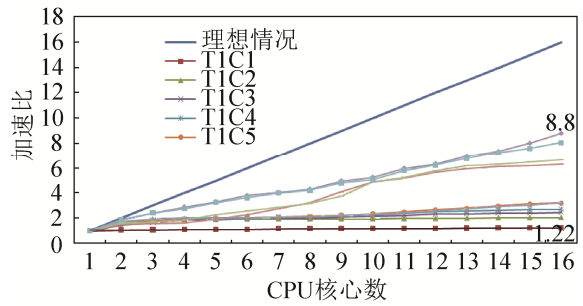


图 6 Spark-KNN 算法加速比
Fig. 6 Acceleration ratio of Spark-KNN algorithm

在图 6 中，算法加速比随数据规模的增长，总体呈现递增趋势。增长最快的是理想加速比曲线。当训练集取 T2，测试集取 C4 时，加速比达到最大(8.8)，加速比最小值为 1.22。从曲线 T1C1 可以看出，曲线基本处于“0 增长”。主要原因是数据量较小，算法加速比迅速达到饱和，即便增加再多的计算节点，计算速度也不会增加。而对于曲线 T2C4、T2C5，可以看到加速比随 CPU 核心数一直在增长。

4 结论

本文研究了基于Spark内存计算技术的电力设备监测大数据实时模式识别方法，设计实现了 Spark-KNN 快速模式识别算法，用于海量绝缘子泄漏电流数据的快速模式识别。在阿里云 E-MapReduce 平台上搭建了实验环境，并完成了 Spark-KNN 和 MR-KNN 算法的性能测试。实验结果表明，Spark-KNN 的平均性能是 MR-KNN 的 2.97 倍，并获得了最高 8.8 倍的加速比，相比 Hadoop MapReduce 更适合执行电力设备监测大数据的实时处理任务。

参考文献:

[1] 周国亮, 朱永利, 王桂兰, 等. 实时大数据处理技术在状态监测领域中的应用[J]. 电工技术学报, 2014, 29(增

https://dc-china-simulation.researchcommons.org/journal/vol30/iss4/32
DOI: 10.16182/j.issn1004731x.joss.201804032

1): 432-437.
Zhou Guoliang, Zhu Yongli, Wang Guilan, et al. Real-Time Big Data Processing Technology Application in the Field of State Monitoring [J]. TRANSACTIONS OF CHINA ELECTROTECHNICAL SOCIETY, 2014, 29(S1): 432-437.

[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM(S0001-0782), 2008, 51(1): 107-113.

[3] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, 2012: 2.

[4] Zaharia M, Chowdhury M, Das T, et al. Fast and interactive analytics over Hadoop data with Spark[J]. USENIX; login, 2012, 37(4): 45-51.

[5] Cover T, Hart, P. Nearest neighbor pattern classification[J]. IEEE Trans. Inf. Theory, 1967, 30(1): 21-27.

[6] 曲朝阳, 朱莉, 张士林. 基于 Hadoop 的广域测量系统数据处理[J]. 电力系统自动化, 2014, 37(4): 92-97.
QU Zhaoyang, ZHU Li, ZHANG Shilin. Data Processing of Hadoop-based Wide Area Measurement System [J]. Automation of Electric Power Systems, 2014, 37(4): 92-97.

[7] Christophe Bisciglia. The smart grid: Hadoop at the tennessee valley authority(TVA)[EB/OL] http://www.cloudera.com/blog/2009/06/smart-grid-hadoop-tennessee valley-authority-tva/

[8] 齐林海, 艾明浩. 一种基于云计算的电压暂降并行计算方法[J]. 中国电机工程学报, 2014, 34(31):5493-5499.
QI Linhai, AI Minghao. A Voltage Sag Parallel Calculation Method Based on Cloud Computing [J]. Proceedings of the CSEE, 2014, 34(31):5493-5499.

[9] 张逸, 杨洪耕, 叶茂清. 基于分布式文件系统的海量电能质量监测数据管理方案[J]. 电力系统自动化, 2014, 38(2): 102-108.
Zhang Yi, YANG Honggeng, YE Maoqing. A Data Management Scheme for Massive Power Quality Monitoring Data Based on Distributed File System [J]. Automation of Electric Power Systems, 2014, 38(2): 102-108.

[10] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. IEEE, 2010: 1-10.

[11] Wang B, Huang S, Qiu J, et al. Parallel online sequential extreme learning machine based on MapReduce[J]. NeuroComputing(S0925-2312), 2015, 149: 224-232.

[12] 王德文, 孙志伟. 电力用户侧大数据分析与并行负荷预测[J]. 中国电机工程学报, 2015 , 35(3): 527-537.
WANG Dewen, SUN Zhiwei. Big Data Analysis and Parallel Load Forecasting of Electric Power User Side[J]. Proceedings of the CSEE, 2015 , 35(3): 527-537.

[13] 曲广龙, 杨洪耕, 张逸. 采用 Map-Reduce 模型的海量电能质量数据交换格式文件快速解析方案[J]. 电网技术, 2014, 38(6): 1705-1711.
QU Guanglong, YANG Honggeng, ZHANG Yi. A Fast Parallel Parsing Scheme for Massive PQDIF Files With Map-Reduce Model [J]. Power System Technology, 2014, 38(6): 1705-1711.

[14] Fadika Z, Govindaraju M, Canon R, et al. Evaluating hadoop for data-intensive scientific operations[C]//2012 IEEE 5th International Conference on Cloud Computing (CLOUD)[C]. IEEE, 2012: 67-74.

[15] 王德文, 刘晓建. 基于 MapReduce 的电力设备并行故障诊断方法[J]. 电力自动化设备, 2014, 34(10): 116-120.
WANG Dewen, LIU Xiaojian. Parallel fault diagnosis based on MapReduce for electric power equipments [J]. Electric Power Automation Equipment, 2014, 34(10): 116-120.

[16] 王远, 陶烨, 袁军, 等. 一种基于 HBase 的智能电网时序大数据处理方法[J]. 系统仿真学报, 2016, 28(3): 559-568.
Wang Yuan, Tao Ye, Yuan Jun, et al. Approach to Process Smart Grid Time-Serial Big Data Base on HBase[J]. Journal of System Simulation, 2016, 28(3): 559-568.

[17] 邵士雯. 超高压线路绝缘子覆冰泄漏电流特性研究[D]. 保定: 华北电力大学, 2012.
Shao Shiwen. Research on Characteristics of Leakage Current of EHV Transmission Line leed Insulator Strings [D]. Baoding: North China Electric Power University, 2012.