DOI: 10.11992/tis.201809007
网络出版地址: http://kns.cnki.net/kcms/detail/23.1538.TP.20181214.1001.002.html

基于 MapReduce 的并行异常检测算法

齐小刚[1], 胡秋秋[1], 刘立芳[2]
（1. 西安电子科技大学 数学与统计学院, 陕西 西安 710071; 2. 西安电子科技大学 计算机学院, 陕西 西安 710071）

摘　要: 为了提高数据挖掘中异常检测算法在数据量增大时的准确度、灵敏度和执行效率, 本文提出了一种基于 MapReduce 框架和 Local Outlier Factor(LOF) 算法的并行异常检测算法 (MR-DLOF)。首先, 将存放在 Hadoop 分布式文件系统 (HDFS) 上的数据集逻辑地切分为多个数据块。然后, 利用 MapReduce 原理将各个数据块中的数据并行处理, 使得每个数据点的 *k*-邻近距离和 LOF 值的计算仅在单个块中执行, 从而提高了算法的执行效率; 同时重新定义了 *k*-邻近距离的概念, 避免了数据集中存在大于或等于 *k* 个重复点而导致局部密度为无穷大的情况。最后, 将 LOF 值较大的数据点合并重新计算其 LOF 值, 从而提高算法准确度和灵敏度。通过真实数据集验证了 MR-DLOF 算法的有效性、高效性和可扩展性。

关键词: 数据挖掘; 异常检测; 局部离群因子; Hadoop; MapReduce; 分布式文件系统; 并行计算; 局部密度
中图分类号: TP311　　**文献标志码:** A　　**文章编号:** 1673−4785(2019)02−0224−07

中文引用格式: 齐小刚, 胡秋秋, 刘立芳. 基于 MapReduce 的并行异常检测算法[J]. 智能系统学报, 2019, 14(2): 224–230.
英文引用格式: QI Xiaogang, HU Qiuqiu, LIU Lifang. Parallel anomaly algorithm based on MapReduce[J]. CAAI transactions on intelligent systems, 2019, 14(2): 224–230.

Parallel anomaly algorithm based on MapReduce

QI Xiaogang[1], HU Qiuqiu[1], LIU Lifang[2]
(1. School of Mathematics and Statistics, Xidian University, Xi'an 710071, China; 2. School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: To improve the accuracy, sensitivity, and efficiency of anomaly detection algorithm in data mining when the amount of data increases, a parallel anomaly detection algorithm (MR-LOF) based on the MapReduce framework and the local outlier factor (LOF) algorithm is proposed in this paper. First, the dataset, stored in the Hadoop distributed file system (HDFS), is logically divided into multiple data blocks. Then, the MapReduce principle is used to process the data in each data block in parallel, so that the *k*-distance and LOF value of each data point is calculated only in a single block. It greatly improves the efficiency of the algorithm. Simultaneously, the concept of *k*-distance is redefined. It avoids the situation where the local density is infinite because more than *k* repeated points exist in the dataset. Finally, the data points whose LOF value is larger than threshold are merged, and the LOF values of combined data are recalculated. This process can effectively improve the accuracy and sensitivity. Experiments with real-world datasets demonstrate the validity, high efficiency, and extendibility of the MR-DLOF algorithm.

Keywords: data mining; anomaly detection; local outlier factor; Hadoop; MapReduce; Distributed File System; parallel computing; local density

收稿日期: 2018−09−04.　**网络出版日期:** 2018−12−18.
基金项目: 国家自然科学基金项目 (61572435, 61472305, 61473222); 教育部-中国移动联合基金项目 (MCM20170103); 复杂电子系统仿真重点实验室基础研究基金项目 (DXZT-JC-ZZ-2015-015); 宁波市自然科学基金项目 (2016A610035, 2017A610119).
通信作者: 胡秋秋. E-mail: huqiuq@yeah.net.

随着当前数据量的增多, 在数据处理和数据分析过程中, 应用有效的数据挖掘技术能够大大提升数据处理和分析的速度, 同时也能够提升数据处理的准确性。其中, 数据挖掘就是从大量

的、不完全的、有噪声的、模糊的及随机的数据集中提取隐含在其中的、事先不知道的但又潜在的有用的信息和知识的过程[1]。异常检测是数据挖掘中的重要任务之一，目的是从给定的数据集中发现异常的数据对象[2]。异常检测又称为离群点检测、偏差检测、孤立点检测等，用于反作弊、故障诊断、金融诈骗等领域。随着移动通信、云计算等技术的快速发展，数据量的日渐增多[3]，传统基于单机内存设计的异常检测算法面临着很大的挑战。因此研究适用于大规模数据的异常检测算法具有重要的应用价值。

近年来，已经出现许多异常检测算法，主要包括两类[4]：有监督性的[5-6]和无监督性的[7-13]。有监督性的异常检测算法在监测异常数据前需要大量的样本进行模型检测，但实际应用中往往无法事先获取大量的训练样本。因此无监督的异常检测算法具有更高的实用价值。

在无监督异常检测算法中，Breunig 等[10]提出的 Local Outlier Factor(LOF) 算法，通过计算每个点的局部异常因子 (LOF 值) 从而判断一个数据对象的异常程度。该算法与其他算法相比，理论简单、适应性较高，并且能有效地检测全局异常和局部异常。然而 LOF 算法是基于局部密度设计的，算法复杂度较高且假设不存在大于或等于 k 个重复点。在这个算法的基础上，Bhatt 等[11]提出了一种改进的 LOF 算法，将 k-distance 修改为 m-distance 以提高算法性能。其中，k-distance 是数据点与其最近的第 k 个数据点之间的距离，而 m-distance 是数据点与其 k 邻域内点距离的平均值。Miao 等[12]提出了一种基于核密度的局部离群因子算法 (KLOF) 来计算每个数据点离群程度。Tang 等[13]引入了相对密度的离群值用来度量数据对象的局部异常，其中数据对象的密度分布是根据数据对象的最近邻来估计的。此外，进一步考虑了反向最近邻和共享最近邻估计对象的密度分布。虽然以上算法一定程度上提高了 LOF 算法的有效性，但其处理数据规模受限于内存容量和数据的复杂性。因此，设计一种既能保证 LOF 算法优点又能高效和有效地处理大量数据的异常检测算法是件有意义的工作。

本文主要针对算法的计算量和算法中重复点对局部异常因子的影响这两个方面对 LOF 异常检测算法进行了深入分析，并根据 Hadoop 作业调度机制和 MapReduce 计算框架，设计了一种基于 MapReduce 和 LOF 思想的并行异常检测算法 (MR-DLOF)。在 MR-DLOF 算法中，为了避免数据集中存在大于或等于 k 个重复点而导致数据点局部密度为无穷大的情况，重新定义了 k-邻近距离。同时为了减少计算量，将整个数据集逻辑地划分为多个数据块，利用 MapReduce 原理并行处理各个数据块中的数据，使得每个数据点的 k-邻近距离和 LOF 值的计算仅在单个块中执行；然后，将每个数据块中 LOF 值较高的数据点合并后进一步计算，得到最终异常数据点，从而提高算法的准确性和灵敏性。

1　Hadoop 技术

Hadoop 是 Apache 软件基金会开发的分布式系统基础架构。其理论基础是 Google 在国际上发表的关于 MapReduce[14]、GFS[15]和 BigTable[16]的三篇论文。Hadoop 由 HDFS、MapReduce、HBase、Hive 和 Zookeeper 等成员组成，其中 HDFS 和 MapReduce 是两个最重要的成员。HDFS 可以实现不同节点、不同类型计算机的数据整合，MapReduce 通过 Map 函数和 Reduce 函数建模来实现可靠的分布式计算。实践证明，大量基于 MapReduce 的并行算法的实现有效地提高了算法的计算能力[17-20]。

1.1　Hadoop MapReduce

Hadoop MapReduce 的实现采用了 Master/Slave 结构。MapReduce 分布式计算框架主要包含两个处理过程：Map 阶段和 Reduce 阶段。Map 阶段的 Map 函数和 Reduce 阶段的 Reduce 函数都由用户根据需求进行自定义。Map 函数主要处理输入数据集并产生中间输出，然后通过某种方式将这些中间输出通过 Reduce 函数组合在一起导出最终结果。此过程均以键值对 (key/value) 形式成对地输入和输出数据，保证了数据的安全性和可靠性。当 Reducer 函数输出最后一个键值对时，MapReduce 任务完成。图 1 为 MapReduce 详细流程。HDFS 首先根据数据大小和块大小对数据进行物理分割，用户可根据需求对数据进行逻辑分割，经过 Map 阶段，对每一块数据执行 Map 任务后，对数据进行排序后进入 Reduce 阶段合并结果，最后将结果写入 HDFS 中。

1.2　Hadoop HDFS

Hadoop Distributed File System (HDFS) 采用 Master/Slave 结构，其作为一个高度容错且易扩展的分布式文件系统，主要包含 NameNode、Secondary NameNode 和 DataNode 三类节点[21]。图 2 详细地描述了 HDFS 结构。

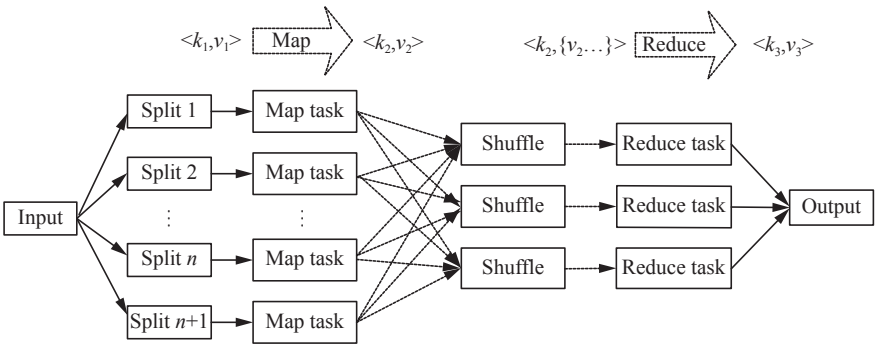

图 1　MapReduce framework 处理过程

Fig. 1　MapReduce framework process

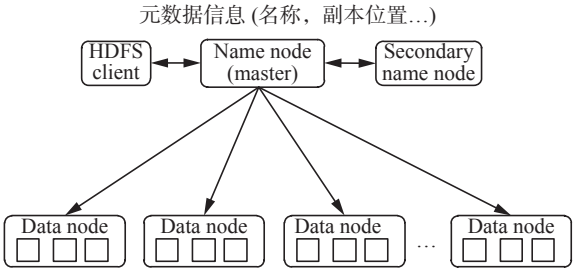

图 2　HDFS 结构

Fig. 2　HDFS structure

NameNode 为主节点且只有一个，负责接收用户的操作请求，记录每个文件数据块在各个 DataNode 上的位置和副本信息。Secondary NameNode 为可选节点，与 NameNode 进行通信，定期保存 HDFS 元数据快照。DataNode 为从节点且可以有多个，主要负责存储数据。同时为了保证数据安全，文件会有多个副本 (默认为 3 个)。DataNode 定期向 NameNode 上报心跳，NameNode 通过心跳响应来控制 DataNode。

2　算法设计

2.1　LOF 算法

LOF 是基于密度的经典算法，其核心部分是关于数据点密度的刻画。由数据点的 k-邻近距离，计算各个数据点的局部可达密度和局部异常因子，根据局部异常因子大小判断数据点的异常程度从而得到异常点。算法的基本概念如下：

1) k-邻近距离 (k-distance): 对于任意正整数 k，点 p 的 k-邻近距离定义为 $k\text{-distance}(p)=d(p,o)$，如果满足以下条件：

① 在样本空间中，至少存在 k 个点 q，使得 $d(p,q)\leqslant d(p,o)$。

② 在样本空间中，至多存在 $k-1$ 个点 q，使得 $d(p,q)<d(p,o)$。

其中，$d(p,o)$ 为点 p 与点 o 之间的距离。

2) k-距离邻域 (k-neighbor): 点 p 的 k-距离邻域 $N_k(p)$ 即与点 p 的之间距离小于 $k\text{-distance}(p)$ 的对象的集合。

3) 可达距离 (reachability distance): 给定参数 k 时，数据点 p 到数据点 o 的可达距离 $\text{reach-dis}(p,o)$ 为数据点 o 的 $k\text{-distance}(o)$ 和数据点 p 与点 o 之间距离的最大值，即

$$\text{reach-dis}_k(p,o)=\max\{k\text{-distance}(o),d(p,o)\}$$

4) 局部可达密度 (local reachability density): 数据点 p 的局部可达密度为它与 k-距离邻域内数据点的平均可达距离的倒数，即

$$lrd_k(p)=\frac{|N_k(p)|}{\sum_{o\in N_k(p)}\text{reach-dis}_k(p,o)} \tag{1}$$

5) 局部异常因子 (local outlier factor): 数据 p 的局部相对密度 (局部异常因子) 为点 p 的邻域内点的局部可达密度和数据点 p 的局部可达密度的比值的平均值，即

$$\text{LOF}_k(p)=\frac{\sum_{o\in N_k(p)}\dfrac{\text{lrd}(o)}{\text{lrd}(p)}}{|N_k(P)|} \tag{2}$$

根据局部异常因子的定义，如果数据点 p 的 LOF 值在 1 附近，表明数据点 p 的局部密度跟它的邻居们差不多；如果数据 p 的 LOF 值小于 1，表明数据点 p 处在一个相对密集的区域，不是一个异常点；如果数据点 p 的 LOF 得分远大于 1，表明数据点 p 跟其他点比较疏远，很可能是一个异常点。

2.2　MR-DLOF 算法

LOF 算法是基于密度的异常检测算法，计算量大，且在 LOF 算法中关于局部可达密度的定义存在假设：不存在大于或等于 k 个重复点。当这样的重复点存在的时候，这些点的平均可达距离为零，局部可达密度就变得无穷，从而影响了算法的有效性。

算法 1　MR-DLOF algorith

输入　$X=x_1,x_2,\cdots,x_N$: data set

k: number of nearest neighbor

N: data block number

θ: threshold for LOF

输出　Abnormal data and LOF values

Get data set which $lof_i > \theta$ from **算法 2** add in XX

Calculate $lof_i > \theta$ of $x_j \in XX$

return Abnormal data and LOF

由于 LOF 算法不足, 本文重新定义了 k-邻近距离的概念, 并结合 MapReduce 框架提出并行异常检测算法。重新定义的 k-邻近距离概念如下。

k-邻近距离 (k-distinct-distance): 对于任意正整数 k, 点 p 的 k-邻近距离定义为 $k-\text{distance}(p) = d(p,o)$, 如果满足以下条件:

1) 在样本空间中, 至少存在 k 个点 q, 使得 $d(p,q) \leqslant d(p,o)$。

2) 在样本空间中, 至多存在 $k-1$ 个点 q, 使得 $0 < d(p,q) < d(p,o)$。

其中, $d(p,o)$ 为点 p 与点 o 之间的距离。

算法 2　Compute $lof_i > \theta$

输入　data set $X = x_1, x_2, \cdots, x_N$

k: number of nearest neighbor

θ: threshold for LOF

N: data block number

输出　data set which $\text{lof}_i > \theta$

Initialize a Hadoop Job

　　Set TaskMapReduce class

Logically divide X into multiple data blocks: $D_1, D_2, \cdots, D_N$.

In the j-th TaskMapReduce

FirstMapper

　　输入　$D = d_1, d_2, \cdots, d_m$

　　输出　$< \text{key}, \text{value} >= < d_i, [(o_k, \text{dis}(d_i, o_k)), k-\text{dis}(d_i)] >$

for each data $d_i, i = 1, 2, \cdots, m$ do

　Calculate $\text{dis}_{ij} = \text{distance}(d_i, d_j), j = 1, \cdots, m$

　　Sort dis_{ij} of d_i

　　for each dis_{ij} of d_i do

　　if $\text{dis}_{ij} \neq 0 \& |k-\text{neighbour}| < k$

　　add d_i and dis_{ij} in k-distinct-neighbor record $(o_k, \text{dis}(d_i, o_k))$

　　end

　　Calculate k-distinct-distance record $k-\text{dis}(d_i)$

end

FirstReducer

　　输入　$< \text{key}, \text{value} >= < d_i, [(o_k, \text{dis}(d_i, o_k)), k-\text{dis}(d_i)] >$

　　输出　$< \text{key}, \text{value} >= < d_i, [(o_k, \text{dis}(d_i, o_k)), k-\text{dis}(d_i)] >$

SecondMapper

　　输入　$< \text{key}, \text{value} >= < d_i, [(o_k, \text{dis}(d_i, o_k)), k-\text{dis}(d_i)] >$

　　输出　$< \text{key}, \text{value} >= < d_i, (o_k, \text{reach}-\text{dis}(d_i, o_k)) >$

for $o_k \in k-\text{distinct}-\text{neighbour}$ do

　　if $k-\text{dis}(o_k) < \text{dis}(d_i, o_k)$

　　$\text{reach}-\text{dis}(d_i, o_k) = \text{dis}(d_i, o_k)$

　　else $\text{reach}-\text{dis}(d_i, o_k) = k-\text{dis}(d_i, o_k)$

end

SecondReducer

　　输入　$< \text{key}, \text{value} >= < d_i, (o_k, \text{reach}-\text{dis}(d_i, o_k)) >$

　　输出　$< \text{key}, \text{value} >= < d_i, \text{lrd}(d_i) >$

for value do

　　$\text{lrd}(d_i) = k / \sum \text{reach}-\text{disk}(d_i, o_k), o_k \in k-\text{distinct}-\text{neighbour}$

end

ThirdMapper

　　输入　$< \text{key}, \text{value} >= < d_i, \text{lrd}(d_i) >$

　　输出　$< \text{key}, \text{value} >= < d_i(\text{lof}(d_i) > \theta), \text{lof}(d_i) >$

for $o_k \in k-\text{distinct}-\text{neighbour}$ do

　　$\text{lof}(d_i) = (\sum \text{lrd}(o_k)/k)/\text{lrd}(d_i), o_k \in k-\text{distinct}-\text{neighbour}$

end

if $\text{lof}(d_i) > \theta$

　　output

ThirdReduce

输入　$< \text{key}, \text{value} >= < d_i(\text{lof}(d_i) > \theta), \text{lof}(d_i) >$

　　输出　$< \text{key}, \text{value} >= < d_i*, \text{lof}(d_i*) >$

for value do

　　Sort $\text{lof}(d_i)$ for d_i and record d_i*

end

本部分主要介绍了 MR-DLOF 算法基本思想和步骤。首先, 将数据集存放在 HDFS 上并将原始数据集逻辑地切分为多个数据块; 然后, 根据 MapReduce 原理并行处理各个数据块中的数据, 使得每个数据点的 k-邻近距离和 LOF 值的计算仅在单个块中执行; 最后将每个数据块中局部异常因子小于设定阈值的点剔除, 并将大于设定阈值的数据点合并成一个新的数据集, 更新其 k-邻近距离和 LOF 值, 从而提高算法的准确度和灵敏

度。Algorihm1 和 Algorihm2 为算法伪代码。

3　实验与分析

实验平台配置：3 台 PC 机 (通过局域网连接)，节点配置为 VMware Workstation Pro 12.0.0 for Windows 下的 CentOS-7，JDK 为 1.8 版本，Hadoop 为 2.7.4 版本。本文所有算法均采用 JAVA 语言实现，eclipse 编译环境。实验环境为基于云平台的 Hadoop 集群，共有 3 个节点：1 个控制节点和 2 个计算节点，控制节点内存为 32 GB，计算节点内存为 8 GB。节点信息如下表 1。

表 1　节点信息

Table 1　Node information

节点名	IP 地址	角色
Master	192.168.0.120	NameNodeResourceManager
Slave0	192.168.0.121	DataNodeNodeManager
Slave1	192.168.0.122	DataNodeNodeManager

实验数据集：为了验证 MR-DLOF 算法的有效性和高效性，本文选用网络入侵数据集 KDD-CUP1999[22]，KDD-CUP1999 数据集中每个连接用 41 个特征和 1 个标签来表述：其中 3 个特征以 CSV 格式写成。这 41 个特征包含 7 个离散变量，34 个连续变量，并且第 20 个变量数据全为 0。

LOF 和 MR-DLOF 算法中采取距离的方法进行计算，由于每个特征属性的度量方法不同，为了避免出现“大数吃小数”的现象，消除属性度量差异对计算结果的影响，需要对数据集进行预处理。本文对除去全为 0 变量和 CSV 格式变量后的 37 个变量进行标准化处理。

3.1　算法的有效性验证

性能衡量标准：异常检测算法对正常数据与异常数据的检测结果如表 2 所示。

表 2　数据检测结果

Table 2　Data test result

结果	被检测为正常	被检测为异常
正常数据	TP(True Positive)	FP(False Positive)
异常数据	FN(False Negative)	TN(True Negative)

准确度：

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{3}$$

灵敏度：即真正辨识率，是正确检测出异常数据的个数与实际异常数据个数之比。

$$\text{Sensitivit} = \frac{\text{TN}}{\text{TN}+\text{FN}} \tag{4}$$

本文主要对比 LOF 算法和 MR-DLOF 算法处理相同规模数据集的准确度和灵敏度，来验证 MR-DLOF 算法的有效性。针对每种规模的数据集，分别从 KDD-CUP1999 标准化处理后的数据库中随机选取 10 组不同的数据集，并使所选取的每种规模数据集中攻击数据 (即异常点) 在该数据集中占比为 1%~2%。使用 LOF 算法和 MR-DLOF 算法分别计算其准确度和灵敏度，并取其平均值作为评价指标，其中设定阈值 $\theta = 1.2$。

由图 3 ~ 4 可知，LOF 和 MR-DLOF 在处理相同数据集时，MR-DLOF 算法在保证其灵敏性的基础上，大大提高了其准确率。

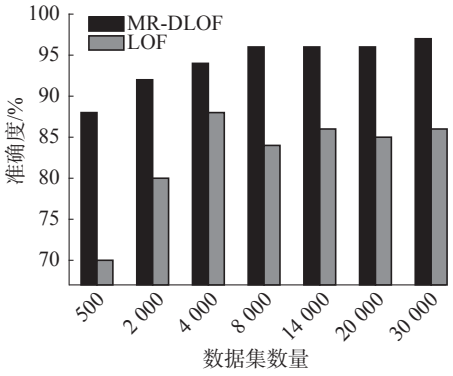

图 3　算法准确度比较

Fig. 3　Accuracy comparison of algorithm

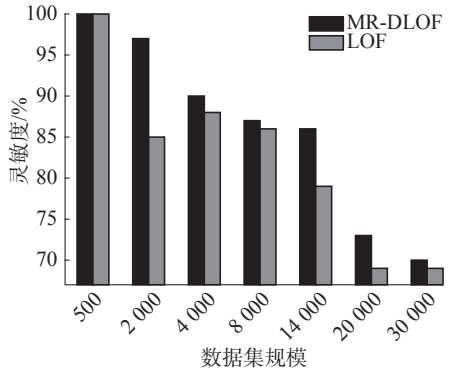

图 4　算法灵敏度比较

Fig. 4　Sensitivity comparison of algorithm

3.2　算法的高效性验证

本文主要对比 LOF 算法和 MR-DLOF 算法处理相同规模数据集的执行时间，来验证 MR-DLOF 算法的执行效率。如图 5 所示，可以看出当数据量比较大时，MR-DLOF 的执行效率明显优于 LOF 算法。数据量少时 LOF 算法执行效率优于 MR-LOF 算法的原因是 Hadoop 调度 Map 任务和 Reduce 任务时需要一定的时间；但当数据量比较大时，将数据分块后，Hadoop 会调度多个 MapReduce 并行执行任务，效率明显增加。

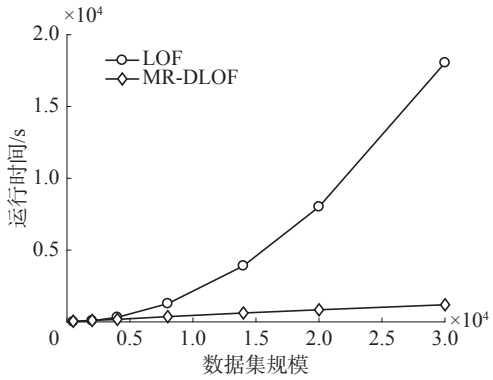

图 5　算法效率比较

Fig. 5　Efficiency comparison of algorithm

3.3　算法的可扩展性验证

为了验证 MR-DLOF 算法的可扩展性, 本文通过扩大数据规模来比较 MR-DLOF 在不同计算节点下的执行效率, 由图 6 可以看出, 在相同数据集规模下, 当集群计算节点增加时, 算法的执行效率提高。因此当数据集增大时, MR-DLOF 算法具有可扩展性, 可通过扩充 Hadoop 集群中计算节点的方法来提高算法的执行效率。

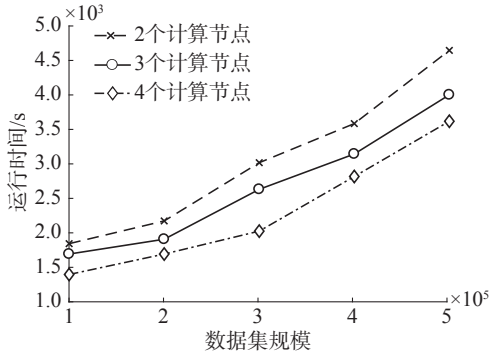

图 6　不同计算节点数量下的执行效率

Fig. 6　Execution efficiency under different calculating node numbers

4　结束语

本文通过分析 LOF 算法的不足, 设计了一种基于 MapReduce 和 LOF 算法的并行异常检测算法。该算法修正了 *k*-邻近距离的概念, 从而避免某些点的可达距离为零、局部可达密度为无穷大的情况, 以提高算法的有效性, 同时, 为了减少计算量, 将分块、利用 MapReduce 框架思想并行化处理各个数据块中的数据点, 大大提高了算法的执行效率。最后, 通过真实数据集验证算法的有效性、高效性和可扩展性。

参考文献:

[1] HAN Jiawei, KAMBER M. Data mining: concepts and techniques[M]. San Francisco: Morgan Kaufmann Publishers Inc., 2006: 1–18.

[2] AGGARWAL C C. Outlier analysis[M]. New York: Springer, 2013: 75–99.

[3] CHEN Feng, DENG Pan, WAN Jiafu, et al. Data mining for the internet of things: literature review and challenges [J]. International journal of distributed sensor networks, 2015, 2015: 431047.

[4] 吴镜锋, 金炜东, 唐鹏. 数据异常的监测技术综述[J]. 计算机科学, 2017, 44(S2): 24–28.
WU Jingfeng, JIN Weidong, TANG Peng. Survey on monitoring techniques for data abnormalities[J]. Computer science, 2017, 44(S2): 24–28.

[5] STEINWART I, HUSH D, SCOVEL C. A classification framework for anomaly detection[J]. The journal of machine learning research, 2005, 6(1): 211–232.

[6] 邓红莉, 杨韬. 一种基于深度学习的异常检测方法[J]. 信息通信, 2015(3): 3–4.
DENG Hongli, YANG Tao. An anomaly detection method based on deep learning[J]. Information and communications, 2015(3): 3–4.

[7] ZHAO Xuanqiang, WANG Guoying, LI Zhixing. Unsupervised network anomaly detection based on abnormality weights and subspace clustering[C]//Proceedings of the 6th International Conference on Information Science and Technology. Dalian, China, 2016: 482–486.

[8] 左进, 陈泽茂. 基于改进 K 均值聚类的异常检测算法[J]. 计算机科学, 2016, 43(8): 258–261.
ZUO Jin, CHEN Zemao. Anomaly detection algorithm based on improved K-means clustering[J]. Computer science, 2016, 43(8): 258–261.

[9] 邹云峰, 张昕, 宋世渊, 等. 基于局部密度的快速离群点检测算法[J]. 计算机应用, 2017, 37(10): 2932–2937.
ZOU Yunfeng, ZHANG Xin, SONG Shiyuan, et al. Fast outlier detection algorithm based on local density[J]. Journal of computer applications, 2017, 37(10): 2932–2937.

[10] BREUNIG M M, KRIEGEL H P, NG R T, et al. LOF: identifying density-based local outliers[C]//Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. Dallas, Texas, USA, 2000: 93–104.

[11] BHATT V, SHARMA K G, RAM A. An enhanced approach for LOF in data mining[C]//Proceedings of 2013 International Conference on Green High Performance Computing. Nagercoil, India, 2013: 1–3.

[12] MIAO Dandan, QIN Xiaowei, WANG Weidong. Anomalous cell detection with kernel density-based local outlier factor[J]. China communications, 2015, 12(9): 64–75.

[13] TANG Bo, HE Haibo. A local density-based approach for

outlier detection[J]. Neurocomputing, 2017, 241: 171–180.

[14] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107–113.

[15] GHEMAWAT S, GOBIOFF H, LEUNG S T. The google file system[C]//Proceedings of the 19th ACM Symposium on Operating Systems Principles. Bolton Landing, NY, USA, 2003: 29–43.

[16] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: a distributed storage system for structured data[C]//Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation. Seattle, WA, 2006: 15.

[17] MU Yashuang, LIU Xiaodong, YANG Zhihao, et al. A parallel C4.5 decision tree algorithm based on MapReduce[J]. Concurrency and computation practice and experience, 2017, 29(8): e4015.

[18] 侯泳旭, 段磊, 秦江龙, 等. 基于 Isolation Forest 的并行化异常探测设计[J]. 计算机工程与科学, 2017, 39(2): 236–244.
HOU Yongxu, DUAN Lei, QIN Jianglong, et al. Parallel anomaly detection based on isolation forest[J]. Computer engineering & science, 2017, 39(2): 236–244.

[19] 叶海琴, 孟彩霞, 王意锋, 等. 一种基于 MapReduce 的频繁模式挖掘算法[J]. 南京理工大学学报, 2018, 42(1): 62–67.
YE Haiqin, MENG Caixia, WANG Yifeng, et al. Frequent pattern mining algorithm based on MapReduce[J]. Journal of Nanjing University of Science and Technology, 2018, 42(1): 62–67.

[20] 刘义, 景宁, 陈荦, 等. MapReduce 框架下基于 R-树的 k-近邻连接算法[J]. 软件学报, 2013, 24(8): 1836–1851.
LIU Yi, JING Ning, CHEN Luo, et al. Algorithm for processing k-nearest join based on R-tree in MapReduce[J]. Journal of software, 2013, 24(8): 1836–1851.

[21] WHITE T. Hadoop: the definitive guide[M]. 4th ed. Gravenstein Highway North: O'reilly Media Inc., 2015: 1–4.

[22] http://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/.

作者简介：

齐小刚，男，1973 年生，教授，博士生导师，主要研究方向为系统建模与故障诊断、网络优化与算法设计。发表学术论文 50 余篇，被 SCI 检索 10 余篇、EI 检索 50 余篇。申请专利 18 项 (授权 9 项)、登记软件著作权 3 项。

胡秋秋，女，1995 年生，硕士研究生，主要研究方向为分布式系统、数据处理与分析。

刘立芳，女，1972 年生，教授，博士，主要研究方向为数据处理与智能计算。发表学术论文 40 余篇，其中被 SCI 检索 9 篇、EI 检索 30 余篇。