(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109086291 A
(43)申请公布日 2018.12.25

(21)申请号 201810590567.7

(22)申请日 2018.06.09

(71)申请人 西安电子科技大学
地址 710071 陕西省西安市太白南路2号西安电子科技大学

(72)发明人 齐小刚 胡秋秋 刘立芳 冯海林 胡绍林

(74)专利代理机构 西安长和专利代理有限公司 61227
代理人 黄伟洪

(51)Int.Cl.
G06F 17/30(2006.01)
G06F 8/30(2018.01)
G06F 11/07(2006.01)

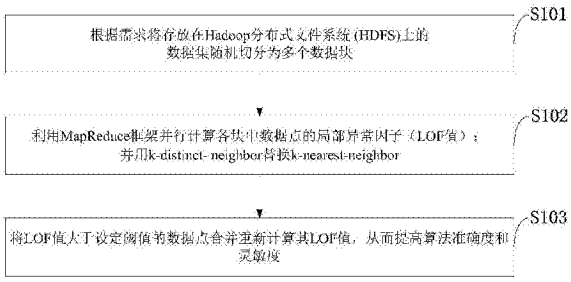
权利要求书1页 说明书9页 附图2页

(54)发明名称

一种基于MapReduce的并行异常检测方法及系统

(57)摘要

本发明属于门适用于特定应用的数字计算或数据处理的设备或方法技术领域，公开了一种基于MapReduce的并行异常检测方法及系统，根据需求将存放在Hadoop分布式文件系统上的数据集随机切分为多个数据块；利用MapReduce框架并行计算各块中数据点的局部异常因子，并用k-distinct-neighbor替换k-nearest-neighbor；将各块中LOF值大于设定阈值的数据点合并重新计算其LOF值。MR-DLOF在处理大量数据时的执行效率明显优于LOF算法。



CN 109086291 A

1.一种基于MapReduce的并行异常检测方法，其特征在于，所述基于MapReduce的并行异常检测方法根据需求将存放在Hadoop分布式文件系统上的数据集随机切分为多个数据块；利用MapReduce框架并行计算各块中数据点的局部异常因子，并用k-distinct-neighbor替换k-nearest-neighbor；将各数据块中LOF值大于设定阈值的数据点合并重新计算LOF值。

2.如权利要求1所述的基于MapReduce的并行异常检测方法，其特征在于，所述基于MapReduce的并行异常检测方法具体包括以下步骤：

首先将数据集存放在HDFS上并根据需求随机的将原始数据集切分为多个数据块；

然后，在每个数据块中，利用LOF思想计算各个点的局部异常因子，其中为了去除大于或等于k个重复点对结果的影响；将k-nearest-neighbor替换为k-distinct-neighbor；

最后将每个数据块中局部异常因子小于设定阈值的点删除，并将大于设定阈值的数据点合并成一个数据集，再次计算这些点的局部异常因子以提高算法的准确度和灵敏度。

3.如权利要求2所述的基于MapReduce的并行异常检测方法，其特征在于，所述基于MapReduce的并行异常检测方法进一步包括以下步骤：

(1)将存储在HDFS上的数据集逻集划分为多个数据块；

(2)基于MapReduce框架，将数据块分配到多个Map中利用LOF算法思想进行并行计算各个数据对象的局部异常因子，将LOF算法中计算的k-nearest-neighbor替换为k-distinct-neighbor；

LOF算法：

输入：样本集合D，正整数k，用于计算第k距离，LOF阈值；

输出：异常数据集；

1)计算每个数据对象与其他数据对象的距离；

2)对距离进行排序，计算第k直接距离以及k领域；

3)计算每个数据对象的可达密度；

4)计算每个数据对象的局部离群因子；

5)将局部离群因子LOF值大于LOF阈值的数据对象作为异常点输出；

(3)将各个数据块中的异常点合并，再次利用LOF算法思想计算这些数据对象的LOF值以提高算法的准确度和灵敏度。

4.一种实现权利要求1所述基于MapReduce的并行异常检测方法的基于MapReduce的并行异常检测系统，其特征在于，所述基于MapReduce的并行异常检测系统包括：

数据集切分模块，用于根据需求将存放在Hadoop分布式文件系统上的数据集随机切分为多个数据块；

并行计算模块，用于利用MapReduce框架并行计算各块中数据点的局部异常因子，并用k-distinct-neighbor替换k-nearest-neighbor；

合并计算模块，用于将各块中LOF值大于设定阈值的数据点合并重新计算LOF值。

5.一种应用权利要求1～3任意一项所述基于MapReduce的并行异常检测方法的移动通信系统。

6.一种应用权利要求1～3任意一项所述基于MapReduce的并行异常检测方法的云计算系统。

一种基于MapReduce的并行异常检测方法及系统

技术领域

[0001] 本发明属于专门适用于特定应用的数字计算或数据处理的设备或方法技术领域，尤其涉及一种基于MapReduce的并行异常检测方法及系统。

背景技术

[0002] 目前，业内常用的现有技术是这样的：随着当前数据的增多，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；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；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；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。近年来，已经出现许多异常检测算法，主要包括两类：有监督性的和无监督性的；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。(1)有监督性的异常检测算法无法事先获取大量的训练样本；(2)无监督异常检测算法处理数据规模受限于内存容量和数据的复杂性。在无监督异常检测算法中，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算法基于局部密度设计，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，一种改进的LOF算法，将k-distance修改为m-distance以提高性能；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-距离是数据点与其k邻域内点距离的平均值。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；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；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，设计一种在证LOF算法优点的基础上又能高效和有效的处理大量数据的异常检测算法是件有意义的工作。

[0003] 综上所述，现有技术存在的问题是：

[0004] (1)有监督性的异常检测算法无法事先获取大量的训练样本，在异常检测数据集中，异常数据会比较稀少，需要利用人工合成的异常数据或现存样本中的少量异常数据作为训练样本，从而会降低异常检测的准确性和有效性。

[0005] (2)无监督异常检测算法处理数据规模受限于内存容量和数据的复杂性，当数据量增大时，算法的运行效率会大大降低。

[0006] 解决上述技术问题的难度和意义：

[0007] 有监督的异常检测算法需要事先获取大量的数据集进行训练，而异常检测中异常

数据比较稀少，需要利用人工合成的异常数据或现存的少量的异常数据作为训练样本，从而降低了异常检测的准确性和有效性。因此无监督异常检测算法具有更高的应用价值。近年来随着移动通信、云计算等技术的快速发展，所产生的数据日益增多，传统的异常检测算法都基于单机设计，且算法复杂度较大，算法处理数据规模受限于内存容量和数据的复杂性，当数据量增大时，算法的运行效率大大降低。因此，研究适用于大量数据的异常检测算法具有重要的意义。Hadoop云计算平台核心包括分布式存储系统和MapReduce编程模型，既能为大量数据提供存储空间，又能为其提供高速的计算能力。因此，基于MapReduce的异常检测算法在处理大量复杂数据时不仅解决了内存容量受限的问题而且大大提高了异常检测的效率。

[0008] 本发明利用Mapreduce框架和Local Outlier Factor算法(无监督算法)思想提出了一种分布式的异常检测算法，解决了算法处理数据规模受限于内存容量和数据的复杂性的问题。

发明内容

[0009] 针对现有技术存在的问题，本发明提供了一种基于MapReduce的并行异常检测方法及系统。

[0010] 本发明是这样实现的，一种基于MapReduce的并行异常检测方法，所述基于MapReduce的并行异常检测方法根据需求将存放在Hadoop分布式文件系统上的数据集随机切分为多个数据块；利用MapReduce框架并行计算各块中数据点的局部异常因子；并用k-distinct-neighbor替换k-nearest-neighbor；将各数据块中LOF值大于设定阈值的数据点合并重新计算LOF值。

[0011] 进一步，所述基于MapReduce的并行异常检测方法具体包括以下步骤：

[0012] 首先将数据集存放在HDFS上并根据需求随机的将原始数据集切分为多个数据块；

[0013] 然后，在每个数据块中，利用LOF思想计算各个点的局部异常因子，其中为了去除大于或等于k个重复点对结果的影响；将k-nearest-neighbor替换为k-distinct-neighbor；

[0014] 最后将每个数据块中局部异常因子小于设定阈值的点删除，并将大于设定阈值的数据点合并成一个数据集，再次计算这些点的局部异常因子以提高算法的准确度和灵敏度。

[0015] 详细步骤：

[0016] 1.将存储在HDFS上的数据集逻集划分为多个数据块；

[0017] 2.基于MapReduce框架，将数据块分配到多个Map中利用LOF算法思想进行并行计算各个数据对象的局部异常因子(LOF值)，其中将LOF算法中计算的k-nearest-neighbor替换为k-distinct-neighbor；

[0018] LOF算法：

[0019] 输入：样本集合D，正整数k(用于计算第k距离)，LOF阈值

[0020] 输出：异常数据集

[0021] 1)计算每个数据对象与其他数据对象的距离

[0022] 2)对距离进行排序，计算第k直接距离以及k领域

[0023] 3)计算每个数据对象的可达密度

[0024] 4)计算每个数据对象的局部离群因子

[0025] 5)将局部离群因子(LOF值)大于LOF阈值的数据对象作为异常点输出

[0026] 3.将各个数据块中的异常点合并,再次利用LOF算法思想计算这些数据对象的LOF值以提高算法的准确度和灵敏度。

[0027] 本发明的另一目的在于提供一种实现所述基于MapReduce的并行异常检测方法的基于MapReduce的并行异常检测系统,所述基于MapReduce的并行异常检测系统包括:

[0028] 数据集切分模块,用于根据需求将存放在Hadoop分布式文件系统上的数据集随机切分为多个数据块;

[0029] 并行计算模块,用于利用MapReduce框架并行计算各块中数据点的局部异常因子;并用k-distinct-neighbor替换k-nearest-neighbor;

[0030] 合并计算模块,用于将LOF值大于设定阈值的数据点合并重新计算LOF值。

[0031] 本发明的另一目的在于提供一种应用所述基于MapReduce的并行异常检测方法的移动通信系统。

[0032] 本发明的另一目的在于提供一种应用所述基于MapReduce的并行异常检测方法的云计算系统。

[0033] 综上所述,本发明的优点及积极效果为:本发明针对计算量和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;其次,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思想的新的异常检测算法(MR-DLOF);在MR-DLOF算法中,将整个数据集分块,并用k-distinct-neighbor替换k-nearest-neighbor,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,利用真实数据集通过一系列仿真实验证实了算法的有效性和高效性。

附图说明

[0034] 图1是本发明实施例提供的基于MapReduce的并行异常检测方法流程图。

[0035] 图2是本发明实施例提供的算法准确性比较示意图。

[0036] 图3是本发明实施例提供的算法灵敏性比较示意图。

[0037] 图4是本发明实施例提供的算法效率比较示意图。

具体实施方式

[0038] 为了使本发明的目的、技术方案及优点更加清楚明白,以下结合实施例,对本发明进行进一步详细说明。应当理解,此处所描述的具体实施例仅仅用以解释本发明,并不用于限定本发明。

[0039] 本发明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,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,根据需求将存放在Hadoop分布式文件系统(HDFS)上的数据集随机切分为多个数据块;然后利用MapReduce框架并行计算各块中数据点的局部异常因子(LOF值);并用k-distinct-neighbor替换k-nearest-neighbor,避免了数据集中存在多于或等于k个重复点而导致局部密度为无穷大的情况;最后将LOF值大于设定阈值的数据点合并重

新计算其LOF值，从而提高算法准确度和灵敏度；实验仿真结果显示，随着数据量的增大，MR-DLOF算法与LOF算法相比，在准确度和执行效率方面具有更好的优势；

[0040] 下面结合附图对本发明的应用原理作进一步的描述。

[0041] 如图1所示，本发明实施例提供的基于MapReduce的并行异常检测方法及系统包括以下步骤：

[0042] S101：根据需求将存放在Hadoop分布式文件系统(HDFS)上的数据集随机切分为多个数据块；

[0043] S102：利用MapReduce框架并行计算各块中数据点的局部异常因子(LOF值)；并用k-distinct-neighbor替换k-nearest-neighbor；

[0044] S103：将各数据块中LOF值大于设定阈值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。

[0045] 下面结合具体实施例对本发明的应用原理作进一步的描述。

[0046] 1、算法设计

[0047] 1.1LOF算法

[0048] 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-nearest-neighbor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；算法的基本概念和流程如下：

[0049] (1)k-邻近距离(k-distance)：对于任意正整数k，在距离数据点p最近的k个点所组成的邻域(k-nearest-neighbor)中，第k个点跟点p之间的距离记为k-distanc(e)p；

[0050] (2)可达距离(rechability distance)：给定参数k时，数据点p到数据点o的可达距离reach-dis(t p,o)为数据点o的k-distance(o)和数据点p与点o之间距离的最大值；即：

[0051] $reach\_dist_k(p,o)=\max\{k\text{-}distance(o),d(p,o)\}$；

[0052] (3)局部可达密度(local rechability density)：数据点p的局部可达密度为它与k-nearest-neighbor内数据点的平均可达距离的倒数；即：

[0053] $$lrd_k(p)=\frac{|N_k(p)|}{\sum_{o\in N_k(p)} reach\_disk_k(p,o)}$$；

[0054] (4)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：

[0055] $$LOF_k(p)=\frac{\sum_{o\in N_k(p)}\frac{lrd(o)}{lrd(p)}}{|N_k(P)|}$$；

[0056] 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。

[0057] 1.2MR-DLOF算法

[0058] 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一个假设：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大。

[0059]

Algorihm1 ： The proposed MR-DLOF algorihm;

Input: $X = x_1, x_2, ..., x_N$: data set
k :number of nearest neighbor
N :data block number
θ :threshold for LOF

Output: Abnormal data and LOF values

1 Get data set which $lof_i > \theta$ from **Algorihm2** add in XX

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

3 return Abnormal data and LOF

[0060] 详细步骤：

[0061] 1.将存储在HDFS上的数据集逻集划分为多个数据块；

[0062] 2.基于MapReduce框架，将数据块分配到多个Map中利用LOF算法思想进行并行计算各个数据对象的局部异常因子(LOF值)，其中将LOF算法中计算的k-nearest-neighbor替换为k-distinct-neighbor；

[0063] LOF算法：

[0064] 输入：样本集合D，正整数k(用于计算第k距离)，LOF阈值；

[0065] 输出：异常数据集；

[0066] 1)计算每个数据对象与其他数据对象的距离；

[0067] 2)对距离进行排序，计算第k直接距离以及k领域；

[0068] 3)计算每个数据对象的可达密度；

[0069] 4)计算每个数据对象的局部离群因子；

[0070] 5)将局部离群因子(LOF值)大于LOF阈值的数据对象作为异常点输出；

[0071] 3.将各个数据块中的异常点合并，再次利用LOF算法思想计算这些数据对象的LOF值以提高算法的准确度和灵敏度。

[0072] 由LOF算法思想以及不足，本发明结合Mapreduce框架提出MR-DLOF异常检测算法；本发明首先将数据集存放在HDFS上并根据需求随机的将原始数据集切分为多个数据块；然后，在每个数据块中，利用LOF思想计算各个点的局部异常因子，其中为了去除大于或等于k个重复点对结果的影响；本发明将k-nearest-neighbor替换为k-distinct-neighbor，从而避免这些点的平均可达距离为零，局部密度为无穷大的情况；最后将每个数据块中局部异常因子小于设定阈值的点删除，并将大于设定阈值的数据点合并成一个数据集，再次计算这些点的局部异常因子以提高算法的准确度和灵敏度；Algorihm1和Algorihm2为算法伪代码。

[0073]

Algorihm2: Compute $lof_i > \theta$

Input: data set $D = d_1, d_2, ..., d_m$
k :number of nearest neighbor
θ :threshold for LOF
N :data block number

Output: data set which $lof_i > \theta$

1 **Initalize a Hadoop Job**
Set askMapReduce class
2 Randomly divide X into multiple data blocks: $block_1, block_2, ..., block_N$.
3 **In the j-th TaskMapReduce**
4 **FirstMapper**
Input: $D = d_1, d_2, ..., d_m$
Output: $< key, value > = < d_i, [(o_k, dis(d_i, o_k)), k - dis(d_i)] >$
5 **for** each data $d_i, i = 1, 2, ..., m$ **do**
6 Calculate $dis_{ij} = distance(d_i, d_j), j = 1, .., m$
7 Sort dis_{ij} of d_i
8 **for** each dis_{ij} of d_i **do**
9 **if** $dis_{ij} \neq 0 \& | k - distinct - neighbour | < k$
10 add d_i and dis_{ij} in k-distinct-neighbour record$(o_k, dis(d_i, o_k))$
11 **end**
12 Calculate k-distance record $k - dis(d_i)$

[0074]

13 **end**

14 **FirstReducer**

Input: $<key,\ value>=\ <d_i,[(o_k,dis(d_i,o_k)),\ k-dis(d_i)]>$

Output: $<key,\ value>=\ <d_i,[(o_k,dis(d_i,o_k)),\ k-dis(d_i)]>$

15 **SecondMapper**

Input: $<key,\ value>=\ <d_i,[(o_k,dis(d_i,o_k)),\ k-dis(d_i)]>$

Output: $<key,\ value>=\ <d_i,(o_k,reach-dis(d_i,o_k))>$

16 **for** $o_k \in k-distinct-neighbour$ **do**

17 **if** $k-dis(o_k)<dis(d_i,o_k)$

18 $reach-dis(d_i,o_k)=dis(d_i,o_k)$

19 else $reach-dis(d_i,o_k)=k-dis(d_i,o_k)$

20 **end**

21 **SecondReducer**

Input: $<key,\ value>=\ <d_i,(o_k,reach-dis(d_i,o_k))>$

Output: $<key,\ value>=\ <d_i,lrd(d_i)>$

22 **for** value **do**

23 $lrd(d_i)=k/\sum reach-disk(d_i,o_k),\ o_k \in k-distinct-neighbour$

24 **end**

25 **ThirdMapper**

Input: $<key,\ value>=\ <d_i,lrd(d_i)>$

Output: $<key,\ value>=\ <d_i(lof(d_i)>\theta),lof(d_i)>$

26 **for** $o_k \in k-distinct-neighbour$ **do**

27 $lof(d_i)=(\sum lrd(o_k)/k)/lrd(d_i),$

$o_k \in k-distinct-neighbour$

28 **end**

29 **if** $lof(d_i)>\theta$

ouput

30 **ThirdReduce**

Input: $<key,\ value>=<d_i(lof(d_i)>\theta),lof(d_i)>$

Output: $<key,\ value>=\ <d_i*,lof(d_i*)>$

31 **for** value **do**

32 Sort $lof(d_i)$ for d_i and record d_i*

33 **end**

[0075] 由于LOF算法具有较强的耦合性，为了充分利用MapReduce框架并行计算功能，该过程定义了三个Map函数和三个Reduce函数。首先将数据集逻辑划分为多个数据块；第一个MapReduce任务中Map函数计算数据块中各个数据对象之间的距离，并计算每个数据对象的k-距离，Reduce函数中不执行任务；第二个MapReduce任务中的Map函数计算数据块中各个数据点的可达距离，Reduce中根据可达距离计算各个数据点可达密度；第三个MapReduce任务中Map函数计算数据块中的各个数据对象的局部离群因子(LOF值)，并将LOF值大于设定

的LOF阈值的数据点输出到Reduce中进行合并和排序。其中计算过程中各个数据块并行计算又保持相互独立。

[0076] 下面结合实验对本发明的应用效果作详细的描述。

[0077] 1、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 12.0.0for Windows下的CentOS-7，JDK为1;8版本，Hadoop为2.7.4版本；本发明所有算法均采用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32G内存，计算节点内存为8G内存；节点信息如下表1。

[0078] 实验数据集：为了验证MR-DLOF算法的有效性和高效性，本发明选用网络入侵数据集KDD-CUP1999，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；41个特征中包含7个离散变量，34个连续变量，并且第20个变量数据全为0。

[0079] 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；本发明对除去全为0变量和CSV格式变量后的37个变量进行标准化处理。

[0080] 表1节点信息

[0081]

节点名	IP 地址	角色
Master	192.168.0.120	NameNode ResourceManager
Slave0	192.168.0.121	DataNode NodeManager
Slave1	192.168.0.122	DataNode NodeManager

[0082] 2、算法的有效性验证

[0083] 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；

[0084] 表2数据检测结果

[0085]

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

[0086] 准确度：$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}*100\%$。

[0087] 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；

[0088] $Sensitivit=\frac{TP}{TP+FN}*100\%$。

[0089] 本发明主要对比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，验证MR-DLOF算法的有效性；针对每种数据集的规模，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θ=1.2;

[0090] 由图2-图3可知,LOF和MR-DLOF在处理相同数据集(N)时,MR-DLOF算法在保证其灵敏性(S)的基础上,大大提高了其准确率(A);

[0091] 3、算法的高效性验证

[0092] 本发明主要对比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;图4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算法原因是Hadoop调度Map任务和Reduce任务时需要一定的时间。

[0093] 本发明通过分析LOF算法的不足:计算量大和假设没有大于或等于k个重复点,设计了一种基于MapReduce和LOF算法的MR-DLOF算法;该算法将LOF算法中的k-nearest-neighbor改成k-distinct-neighbor,从而避免某些点的可达距离为0和局部可达密度为无穷大的情况,以提高算法的有效性,并利用MapReduce框架思想将数据分块,从而将算法过程并行化处理,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和高效性。

[0094] 由于MR-DLOF算法采用LOF算法思想,其异常检测的准确度和灵敏度受限于LOF算法,并且计算效率、准确度和灵敏度均受参数k和分块数量的影响(本发明工作中均由实验对参数k∈[10,35]和逻辑分块数量进行选择);所以下一步工作要在参数的使用和参数值的决定上进行更多的研究,并在资源充足情况下增加集群节点数量研究其可扩展性。

[0095] 以上所述仅为本发明的较佳实施例而已,并不用以限制本发明,凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等,均应包含在本发明的保护范围之内。

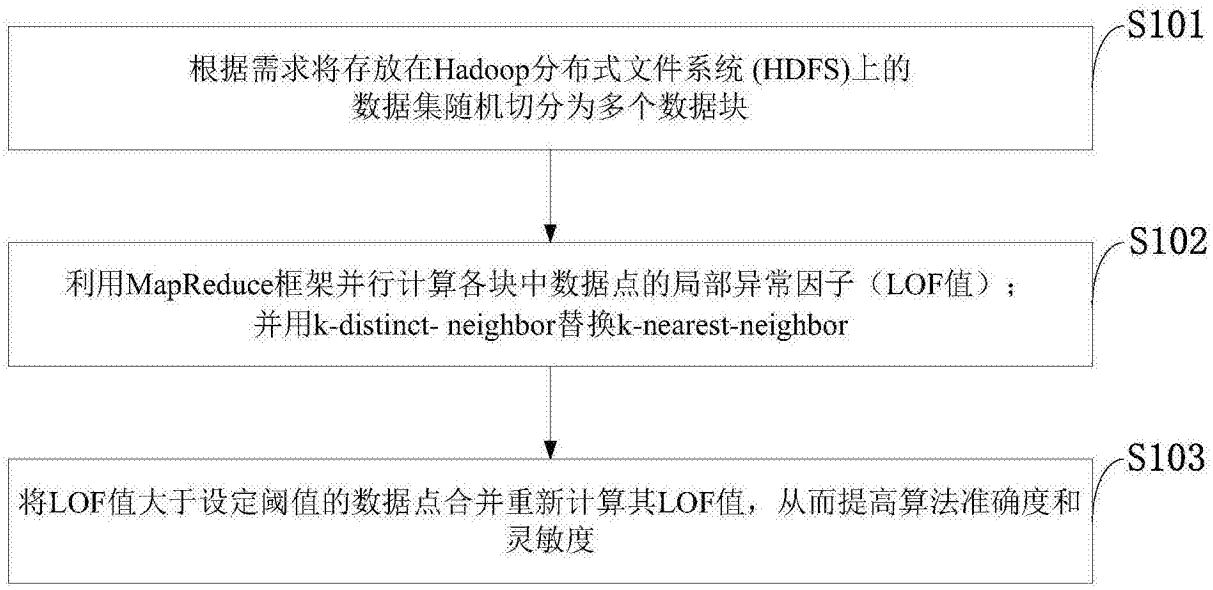

S101
根据需求将存放在Hadoop分布式文件系统 (HDFS)上的
数据集随机切分为多个数据块
S102
利用MapReduce框架并行计算各块中数据点的局部异常因子（LOF值）；
并用k-distinct- neighbor替换k-nearest-neighbor
S103
将LOF值大于设定阈值的数据点合并重新计算其LOF值，从而提高算法准确度和
灵敏度

图1

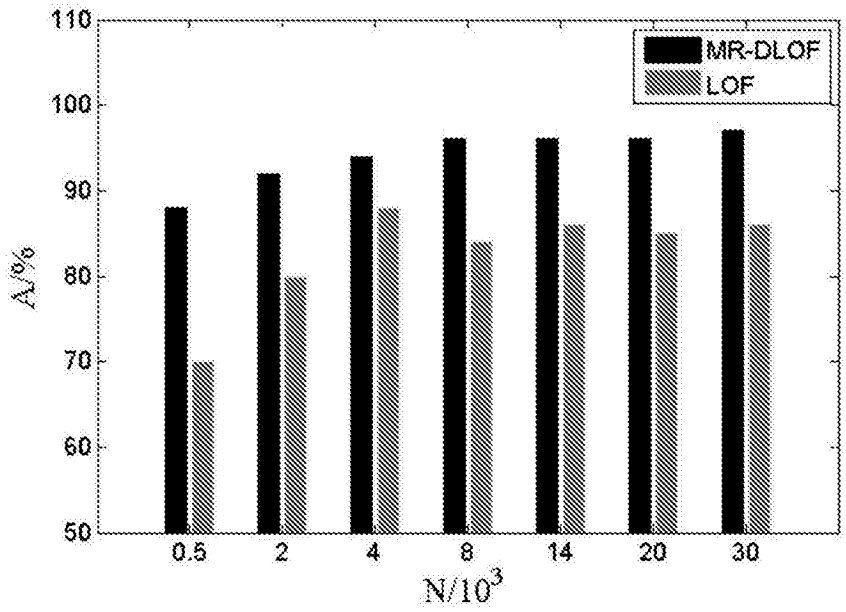

MR-DLOF
LOF
A/%
110
100
90
80
70
60
50
0.5
2
4
8
14
20
30
N/10^3

图2

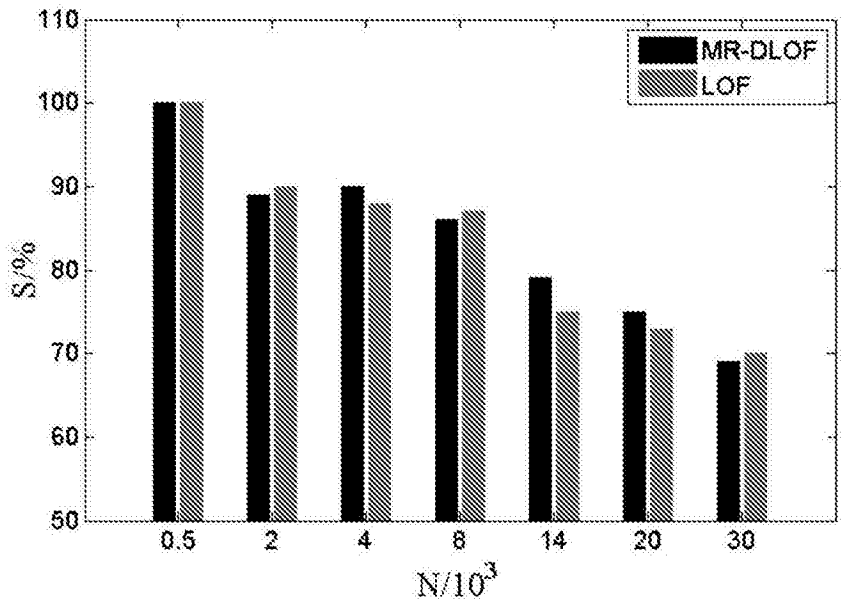

MR-DLOF
LOF
S/%
110
100
90
80
70
60
50
0.5
2
4
8
14
20
30
N/10^3

图3

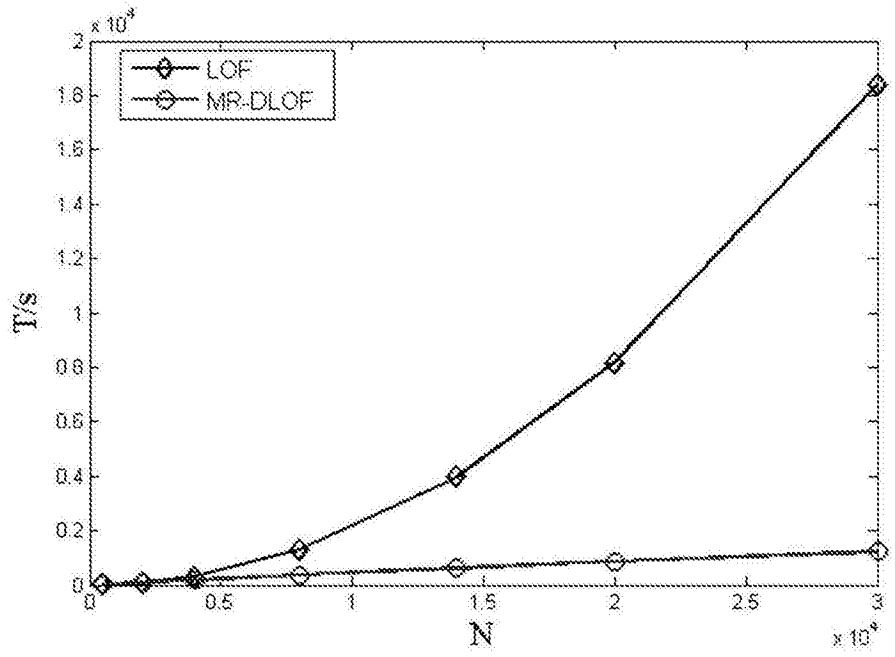

x 10^4
LOF
MR-DLOF
T/s
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.5
1
1.5
2
2.5
3
N
x 10^4

图4