VFDT 算法基于 Storm 平台的实现方案

张发扬,李玲娟,陈 煜
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:以提升流数据的分类效率为目标,研究如何在流数据处理平台 Storm 上实现快速决策树算法-VFDT。设计了 VFDT 基于 Storm 的分布式并行化实现方案,将 VFDT 算法分为建树、分类和评价共三个模块,建树模块负责决策树的初始化和增量建树,分类模块负责对待分类样本进行分类标记,评价模块负责用已标记的样本对 VFDT 决策树进行评价。通过正确设计 Storm 拓扑中的 Spout/Bolt 实现各模块的功能,通过为分类 Bolt 设定多个 Task 来实现分类模块的并行化;用内存数据库 Redis 实现三个模块的有效衔接和决策树的保存;用消息中间件 Kafka 来提高算法对流数据突增的容忍度。基于该方案的 VFDT 算法实现与测试结果表明:在 Storm 集群环境下的 VFDT 算法分类效率相对于单机环境有显著提高,而且合理设定分类 Bolt 的 Task 数可使分类效率进一步提高。

关键词:流数据;快速决策树算法;分布式;并行化;Storm
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2016)09-0192-05
doi:10.3969/j.issn.1673-629X.2016.09.043

Implementation Scheme of VFDT Algorithm on Storm Platform

ZHANG Fa-yang, LI Ling-juan, CHEN Yu
(School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: In order to improve the classification efficiency of the stream data, studies how to implement VFDT algorithm on Storm, a stream data processing platform. A scheme of distributed parallel implementing of VFDT algorithm based on Storm platform is designed. The VFDT algorithm is divided into three modules including building tree module, classification module and evaluation module. The building tree module is responsible for the initializing and incremental building of decision tree, and the classification module for classifying the samples, and the evaluation module for evaluating the VFDT decision tree using the labeled samples. The functions of each module are realized by correctly designing the Spout/Bolt of Storm Topology, and the parallelization of the classification module is implemented by deploying multiple tasks for Classification Bolt. The memory database Redis is used to realize the effective connection of the three modules and the preservation of the decision tree. The message middleware Kafka is used to improve the tolerance of burst stream data. The results of implementing and testing VFDT algorithm based on the proposed scheme show that the classification efficiency of VFDT algorithm under the Storm cluster environment is significantly improved compared with that under the single machine environment, and the classification efficiency can be further improved by reasonably setting the task number in Classification Bolt.

Key words: stream data; Very Fast Decision Tree (VFDT); distribution; parallelization; Storm

0 引 言

20 世纪末,为适应网络监控、入侵检测、情报分析、商业交易管理和分析等应用的要求,数据流技术应运而生[1]。数据流中的数据(即流数据)是有序的、连续的且不断变化的,甚至是无限的[2],不能像传统的静态数据一样将其存储在硬盘或者内存之中,即再次处理这些数据的代价将非常昂贵。

流数据挖掘是指从快速、大量、连续的数据流中挖掘出有价值的信息。数据流具有快速、连续、大量的特点,传统数据挖掘(Data Mining, DM)算法难以对流数据进行有效的挖掘。对于流数据,它的搜集和挖掘过程必须同时进行,且必须以最快的速度从不断到来的数据中挖掘出用户所需要的信息。传统的静态数据挖掘通常能满足数据分析处理的精确性要求,但是,对流

收稿日期:2015-11-13 **修回日期**:2016-03-03 **网络出版时间**:2016-08-23
基金项目:国家自然科学基金资助项目(61302158,61571238);中兴通讯产学研项目
作者简介:张发扬(1990-),男,硕士研究生,CCF 会员,研究方向为流数据挖掘;李玲娟,教授,CCF 会员,研究方向为数据挖掘、信息安全、分布式计算。

数据而言,由于数据收集的时间和处理速度有限,因此得到的挖掘模型是近似结果。流数据挖掘结果的近似性是其不同于传统数据挖掘的一个重要特点。

分类挖掘是数据挖掘的主要任务之一,决策树算法是分类挖掘的一类主流算法。VFDT(Very Fast Decision Tree)算法[3]是经典的流分类算法之一。VFDT 在假设数据流不发生概念漂移的情况下对持续不断的数据流进行增量学习,能够很好地适应流数据的分类。VFDT 算法可以使每条训练样本的处理花费恒定的内存和时间,因此可以有效解决时间和内存的限制问题。该算法通过最初的少量样本生成随时可用的分类器,并可随着训练样本的到来,对原有分类器进行增量更新,不断优化原始决策树,即 VFDT 算法以增量的形式,通过不断地将叶子节点替换为决策节点而生成决策树。

与传统的静态大数据处理平台 Hadoop[4] 不同,Storm 是开源的流数据处理框架[5],能够高效可靠地处理源源不断的数据流。流挖掘算法运行于数据流处理平台是充分发挥算法效力的前提。为此,文中研究如何将 VFDT 算法部署到 Storm 平台上进行分布式并行化实现,以提高 VFDT 算法对流数据的分类效率。

1　VFDT 算法分析

VFDT 算法是通过对 Hoeffding 树改进而实现的。Hoeffding 树是通过不断地将叶子节点转换为内部节点而生成的,其中每个叶节点都存有关于属性值的统计信息,这些统计信息用于计算属性的信息增益。当数据流中一个新的样本到来后,该样本沿着决策树从上到下遍历,树的每个内部节点都对其进行划分测试,根据不同的属性取值,样本进入不同的分枝,最终到达树的叶节点。当数据到达叶节点后,更新该叶节点上的统计信息。如果统计信息的计算结果满足节点分裂条件,则该叶节点变为内部节点,并产生基于该内部节点的子女叶节点。子女叶节点的个数取决于新的内部节点的属性的可能取值数目。

VFDT 算法采用信息熵或者 Gini 指标作为选择分裂属性的标准,以 Hoeffding 不等式作为判定节点分裂的条件。选择 Hoeffding 不等式作为节点分裂条件的目的是确定叶子节点变为内部节点所需要的样本数目,以达到使用尽量少的样本建立准确率较高的决策树的目的。

以 t 作时间戳,x_t 表示 t 时刻到达的数据向量,数据流可表示为 $\{\cdots,x_{t-1},x_t,x_{t+1},\cdots\}$ [6]。VFDT 算法的有关定义如下:

(1)信息增益[7]:叶子节点 l 中存储训练样本集 D 的统计信息,则对于样本集 D 分类所需的期望信息为 $\mathrm{Info}(D)=-\sum_{i=1}^{m}p_i\log_2(p_i)$。其中,$p_i$ 是样本集 D 中任意一条样本属于类 C_i 的概率,$p_i=|C_{i,D}|/|D|$,m 是类别属性的取值个数。对于叶子节点可能的分裂属性 A,设 A 有 v 个取值,则利用属性 A 对样本集 D 进行划分的期望信息为 $\mathrm{Info}_A(D)=-\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times\mathrm{Info}(D_j)$,属性 A 的信息增益为 $\mathrm{Gain}(A)=\mathrm{Info}(D)-\mathrm{Info}_A(D)$。

(2)Hoeffding 边界[8-9]:对一个真值随机变量 $r\in R$,设对 r 取了 n 个独立的观察值,平均值为 $\bar{r}$,其 Hoeffding 约束以 $1-\delta$ 的概率保证变量 r 的真实值与观察值 $\bar{r}$ 之差小于 ε,即:$P(r\geqslant\bar{r}-\varepsilon)=1-\delta$。其中,$\varepsilon=\sqrt{\frac{R^2\ln(1/\delta)}{2n}}$,$r$ 代表信息增益,R 的取值范围是 $\log_2$#Classes(Classes 是类别属性取值的数量)。

(3)主动分裂系数 τ[10]:τ 的作用在于当几个属性的信息增益值 G 几乎相等时,可能需要更多的样本来决定出叶子节点的决策属性,通过设定 τ 值来主动选择属性并实现叶子节点分裂。当满足 $\Delta G<\varepsilon<\tau$ 时,选择 ΔG 中信息增益最大或者次大的属性作为该叶子节点的决策属性。

(4)节点分裂条件:判断节点分裂的有效条件之一是 $(\overline{G_l(X_a)}-\overline{G_l(X_b)})>\varepsilon$ 或者 $\varepsilon<\tau$。$\overline{G_l(X)}$ 指在叶子节点中属性 X 的信息增益,X_a、X_b 分别指具有最大及次大信息增益的属性。

VFDT 算法的建树流程如图 1 所示。

概括地说,一条训练样本是一个 Tuple,即一条流,样本中各属性的元素与初始化阶段抽取的属性信息保持一致,通过分析流的非类别属性与类别属性的关系建立一棵决策树。增量建树过程就是不断将叶子节点转化为内部节点的过程,其中每个叶子节点都将保存有关节点分裂的统计信息。当一个训练样本到达之后,从根节点开始,根据该节点的属性取值进入不同的分支,以此过程进行递归直至到达叶子节点。到达叶节点之后将对叶子节点的统计信息进行更新,当统计值满足计算的阈值时将触发计算各可能属性的信息增益以及 Hoeffding 边界值,若满足节点分裂的条件,则将该叶子节点转化为内部节点,并根据决策属性的取值产生新的叶子节点。

2　Storm 平台

Apache Storm 是由 Twitter 公司开源的分布式实时计算系统。与 Hadoop 的批处理相比,Storm 具有更好的实时性、可拓展性和容错性,能有效地处理流数据,被广泛用于实时分析、在线机器学习等场景[11]。

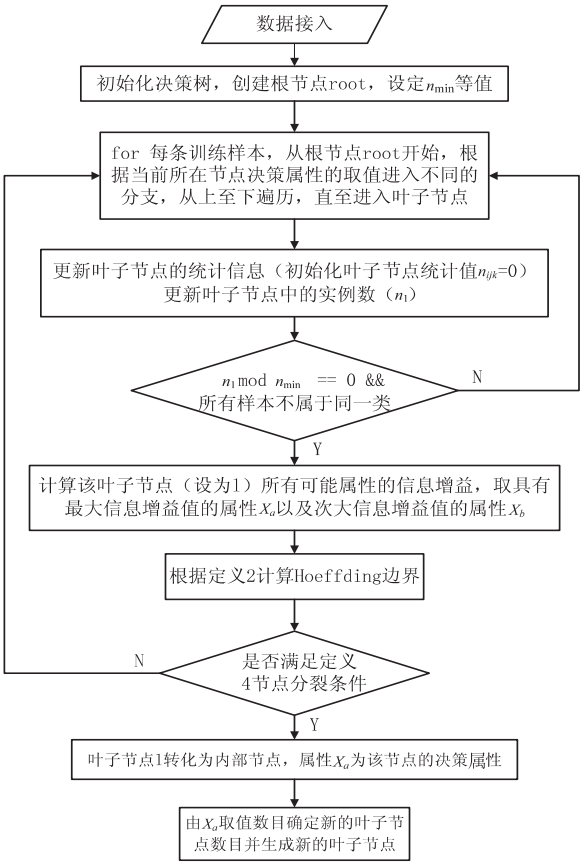

图 1 VFDT 算法的建树流程

在 Storm 内部,数据流是由拓扑(Topology)来处理的。拓扑包含 Spout、数据源以及 Bolt[12]。Bolt 是拓扑的一个重要实体,负责数据流动转换,比如计算、过滤、聚合以及机器学习等。一个拓扑可以有一个或者多个 Bolt 实现数据流的复杂流转。

Storm 集群的基本架构如图 2 所示,主要包括两种节点:主节点 Nimbus(Master Node)以及工作节点 Supervisor(Worker Node)。其中,Nimbus 既负责将代码分发到不同的工作节点,又负责监控集群。在每个工作节点上都会运行一个 Supervisor,负责监听 Nimbus 分配给该节点的工作[13]。每个 Worker 进程执行一个具体的 Topology,Worker 中的执行线程为 Executor,每个 Executor 中又包含一个或者多个 Task,Task 为 Storm 的最小处理单元。一个运行中的 Topology 是由运行在一台或者多台工作节点上的 Worker 来完成具体的业务操作。Nimbus 与 Supervisor 之间的通信由 Zookeeper 来完成。

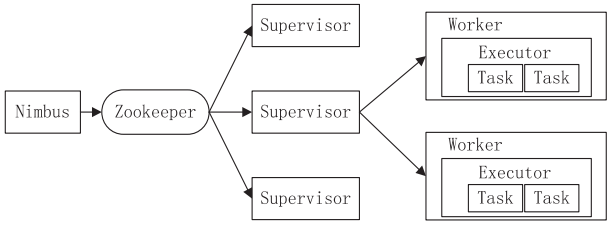

图 2 Storm 集群的基本架构

3 VFDT 算法基于 Storm 的实现方案设计

文中设计的 VFDT 基于 Storm 的分布式并行化实现方案,将 VFDT 算法分为建树、分类和评价共三个模块,建树模块负责决策树的初始化和增量建树,分类模块负责对待分类样本进行分类标记,评价模块负责用已标记样本对 VFDT 决策树进行评价。各模块都有相应的 Topology,如图 3 所示。三个模块之间通过内存数据库 Redis[14]实现衔接,从而形成一个整体的计算框架,Redis 也用于决策树的保存;消息中间件 Kafka[15]用来提高算法对流数据突增情况的容忍度。

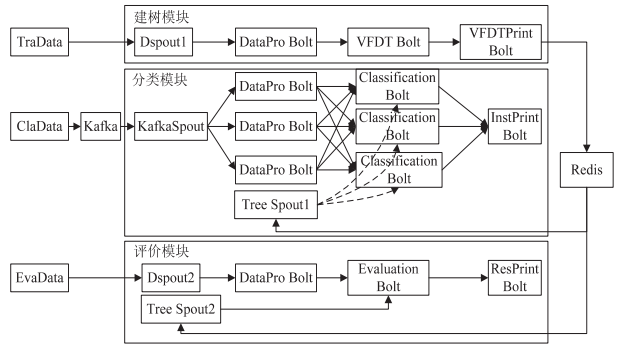

图 3 VFDT 算法在 Storm 平台上的拓扑

图中的 TraData 表示外部数据源,为建树模块提供训练样本;Dspout1 作为建树拓扑的数据源从 TraData 中拉取数据并传递给其他数据处理 Bolt;DataPro Bolt 主要工作是对训练样本进行初始化,并将其转换成算法所需要的类;VFDT Bolt 接收样本,并利用训练样本进行决策树的增量建立;VFDTPrint Bolt 接收增量建立的决策树,并将决策树进行序列化存储到 Redis 数据库中;ClaData 表示外部数据源,为分类模块提供待分类样本;Kafka 是消息中间件,订阅 ClaData 中的样本,同时供分类模块进行消费;KafkaSpout 作为分类拓扑的数据源,接收 Kafka 中的待分类样本,并将样本进行分发;Tree Spout1 表示拓扑的决策树数据源,从 Redis 中实时获取决策树并进行分发;Classification Bolt 将利用 Tree Spout1 传递到决策树对待分类样本进行分类;InstPrint Bolt 主要是对标记好的样本进行存储;EvaData 表示外部数据源,为评价模块提供评价样本;Tree Spout2 的功能与 Tree Spout1 一致;Evaluation Bolt 利用 EvaData 对 Tree Spout2 中的决策树进行评价;ResPrint Bolt 将对评价结果进行存储。

(1)建树模块。

如前所述,建树模块主要负责决策树的初始化以及决策树的增量建立。初始化的过程主要包括数据集属性信息的抽取以及根节点的建立。决策树的增量建立过程包括读入训练样本集和基于图 1 所示的流程建立决策树。如图 3 所示,DSpout1 作为训练样本的数据源,不断向负责数据预处理的 DataPro Bolt 发送训练样

本,DataPro Bolt 将训练样本处理成算法需要的类,并将其传递到负责建树的 VFDT Bolt 中,VFDT Bolt 将调用 VFDT 插入样本的方法实现决策树的动态更新,最后将 VFDT 决策树传递到 VFDTPrint Bolt 中实现决策树的序列化并存储到 Redis 中。

(2)分类模块。

分类模块的主要功能是完成对待分类样本的标记工作。考虑到待分类样本数量大且源源不断地到来,当数据源突然增加时,有可能导致算法在 Storm 上并发度不足而引起异常,文中使用了消息中间件 Kafka 做数据暂存区。Kakfa 具有高性能、高拓展性、分布式及持久性,当数据源突然增加时可以将部分数据持久化至硬盘中去,不至于造成数据的丢失[15]。为保证分类模块的拓扑能够保持较高的数据吞吐量,文中将该模块中的数据预处理 DataPro Bolt 以及对待分类样本进行分类的 Classification Bolt 的 Task 都设置为多个,以提高处理的并发度。

如图 3 所示,DataPro Bolt 使用 Shuffle Grouping(随机分组)的流分组方式从 KafkaSpout 中拉取数据,使得该 Bolt 的多个 Tasks 中的每个 Task 处理的样本数目大致相同。Classification Bolt 同样采用 Shuffle Grouping 的方式使该 Bolt 中每个 Task 能够平均处理数据。为了使该 Bolt 中的每个 Task 可以取到相同的决策树,这部分还采用 All Grouping(广播分组)方式从负责由 Redis 内存数据库中实时获取 VFDT 决策树的 Tree Spout1 中拉取决策树。最后对 Classification Bolt 中标记过的待分类样本采用 Global Grouping(全局分组)的方式发送到 InstancePrint Bolt 中。

其中,Classification Bolt 中利用决策树 VFDTTree 对待分类样本 ClaData 进行分类的伪代码如算法 1 所示。

算法 1:样本分类。

输入:待分类样本 ClaData,决策树 VFDTTree(根节点为 root);
输出:已标记样本 MarkedData。

```
1:if VFDTTree! =null then
2:for instance in ClaData
3:instance 从 VFDTTree 的 root 节点开始,根据节点决策属性的取值进入不同的分支,从上至下进行遍历,直至进入叶子节点 node_l
4:根据 node_l 的统计值获取该节点的类别属性的取值 val
5:输出标记后的样本 MarkedData=instance+val
6:end for
7:end if
```

(3)评价模块。

评价模块的主要功能是利用已标记的评价样本实现对实时传递过来的 VFDT 决策树的评价。为了保证评价样本的实时性,文中采用滑动窗口的方式保存最新的评价数据,每隔一秒触发一次评价,并将评价结果传输至 ResultPrint Bolt 中。

如图 3 所示,Dspout2 作为评价样本的数据源,向 DataPro Bolt 发送数据,经过 DataPro Bolt 处理后发送到负责评价的 Evaluation Bolt 中,在 Evaluation Bolt 中维持一个固定大小的滑动窗口,用于存储最近的 N 条评价数据。每当 Evaluation Bolt 从 Tree Spout2 中拉取最新的决策树后,都利用窗口中的样本对决策树进行一次评价,并将评价结果发送到 ResPrint Bolt 中,实现结果的存储。

4　VFDT 算法基于 Storm 的实现与性能测试

文中分别在单机和集群环境下,用 Java 进行了 VFDT 算法的实现,算法运行环境是:

集群硬件环境:1 个 Nimbus 节点,2 个 Supervisor 节点。

集群软件环境:操作系统为 Centos6.4、JRE1.7.0_13、Zookeeper-3.4.6、Storm0.9.1、Kafka2.8.1、redis-2.4.5。

单机环境:eclipse_4.5.0、JRE1.7.0_13、Windows7、2.13 GHz、4 GB 内存。

目的是借助流处理平台提高 VFDT 算法的效率。为了验证所设计的 VFDT 算法基于 Storm 的实现方案的可行性和有效性,分别测试了单机与集群环境下,基于 Storm 的 VFDT 算法分类模块的吞吐量与分类 Bolt(Classification Bolt)的 Task 线程数(即并行度)的关系,以及相同的 Task 线程数下数据处理时间与数据量的关系。

测试使用的数据集是 KDD CUP 的 Nursery 数据集,属性个数是 8,类别属性取值个数是 5,基本训练样本数量是 12 400,通过复制得到所需量的分类样本。

(1)吞吐量测试。

实验通过复制 Nursery 得到大规模的分类样本。单机与集群环境下,VFDT 分类模块对应于不同分类线程(Task)数的吞吐量如图 4 所示。

可以看出,单机环境下,Task 数为 8 时,吞吐量达到最大,为 35 106.7 条/s,当线程继续增加,吞吐量呈现下降趋势。集群环境下,Task 数为 8 时,吞吐量达到最大,为 88 007.5 条/s,当线程继续增加,吞吐量略呈下降趋势。

(2)数据处理时间测试。

图 5 对比了当单机和集群环境下 Classification Bolt 的 Task 数为 8 时,不同数据量所需的处理时间。单机环境下,随着数据量的增加,处理时长急剧增加;而集群环境下,随着数据量的增加,处理时长缓慢

增加。

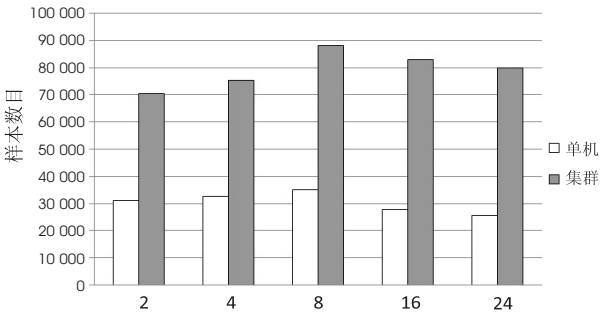

图 4 Classification Bolt 的 Task 数量变化对算法吞吐量的影响

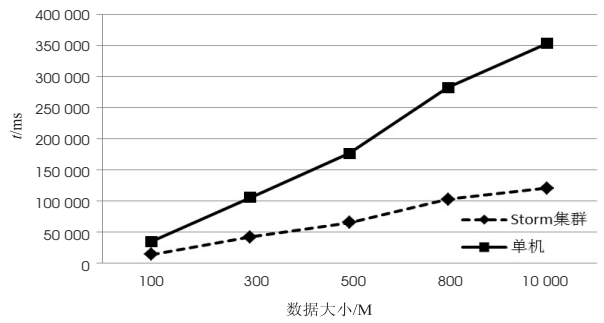

图 5 算法对应于不同数据量的运行时间

(3)测试结果分析。

由图 4 以及图 5 可以看出,基于 Storm 集群的 VFDT 算法在处理规模较大的流式数据时,吞吐量优势明显,对数据量的增加具有较高的承受力。这是由于 Storm 是将 Topology 的各个组件(Spout/Bolt)分配到不同的 Executor 中,并在多个 Worker 中执行的。由图 4 还可以看出,合理设定分类 Bolt 的 Task 数可以最大限度发挥 Storm 的并行处理能力。

5 结束语

文中将经典的流数据分类挖掘算法-VFDT 部署于流数据处理平台 Storm 上,以实现对流数据的分布式并行化分类。所设计的 VFDT 算法基于 Storm 的实现方案按算法功能划分出建树模块、分类模块和评价模块,其中的分类模块以并行化方式运作;通过合理配置 Storm 拓扑和使用 Redis 与 Kafka 提高了实现方案的完整性和可行性。基于该方案的 VFDT 算法实现与性能测试结果说明了方案的正确性和有效性,也说明了基于 Storm 的 VFDT 算法对大规模流数据有良好的适应能力。

参考文献:

[1] 史金成,胡学钢. 数据流挖掘研究[J]. 计算机技术与发展,2007,17(11):11-14.

[2] 顾 伟. 分布式流数据实时计算框架的研究和开发[D]. 杭州:浙江理工大学,2013.

[3] Gama J,Rocha R,Medas P. Accurate decision trees for mining high-speed data streams[C]//Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. [s. l.]:ACM,2003:523-528.

[4] White T. Hadoop:the definitive guide[M]. [s. l.]:O'Reilly Media,Inc. ,2012.

[5] The Apache Foundation. Storm official website[EB/OL]. [2014-04-08]. http:// storm-project. net.

[6] Raahemi B,Zhong W,Liu J. Peer-to-peer traffic identification by mining IP layer data streams using concept-adapting very fast decision tree[C]//Proc of 20th IEEE international conference on tools with artificial intelligence. [s. l.]:IEEE,2008:525-532.

[7] Maron O,Moore A W. Hoeffding races:accelerating model selection search for classification and function approximation[J]. Advances in Neural Information Processing Systems,1993,6(1):59-66.

[8] 王 涛,李舟军,胡小华,等. 一种高效的数据流挖掘增量模糊决策树分类算法[J]. 计算机学报,2007,30(8):1244-1250.

[9] Littlestone N. Learning quickly when irrelevant attributes abound:a new linear-threshold algorithm[J]. Machine Learning,1988,2(4):285-318.

[10] 蒋良孝,蔡之华,刘 钊. 一种基于信息增益的分类规则挖掘算法[J]. 中南大学学报:自然科学版,2003,34(z1):69-71.

[11] Github Inc. Storm Wiki[EB/OL]. [2013-12-07]. https://github. com/nathanmarz/storm/wiki.

[12] Marz N. Storm:distributed and fault-tolerant real time computation[EB/OL]. [2011-10-21]. https://www. infoq. com/presentations/Storm-Introduction.

[13] Petkov V,Gerndt M. Integrating parallel application development with performance analysis in periscope[C]//Proc of IPDPSW. [s. l.]:IEEE,2010:1-8.

[14] 曾泉匀. 基于 Redis 的分布式消息服务的设计与实现[D]. 北京:北京邮电大学,2014.

[15] Kreps J,Narkhede N,Rao J. Kafka:a distributed messaging system for log processing[C]//Proceedings of the NetDB. Athens,Greece:[s. n.],2011:1-7.