单位代码： 10293 密 级：



专 业 学 位 硕 士 论 文



论文题目：

学号

姓名

导师

专业学位类别

类型

专业（领域）

论文提交日期

申请 全日制/非全日制

申请

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

↑

（请填写论文英文题目，Times New Roman 2号字加粗）

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of XXXXXXXX学位类别英文



By

XXXXXX作者英文名字

Supervisor: Prof. XXXXXX导师英文名字

XXX XXXX论文提交日期例如：April 2018

注：（论文定稿时，此页直接删除，不要打印）

专业学位类别英文：

工程硕士：Master of Engineering

工商管理硕士：Master of Business Administration

工程管理硕士：Master of Engineering Management

上一页英文封面中，红色字体的文字打印时直接删除。

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生学号：\_\_\_\_\_\_\_\_\_\_\_\_ 研究生签名：\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_

南京邮电大学学位论文使用授权声明

本人承诺所呈交的学位论文不涉及任何国家秘密，本人及导师为本论文的涉密责任并列第一责任人。

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布（包括刊登）授权南京邮电大学研究生院办理。

非国家秘密类涉密学位论文在解密后适用本授权书。

研究生签名：\_\_\_\_\_\_\_\_\_\_\_\_ 导师签名：\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_

摘要

《摘要正文》×××××××××××××××××××中文摘要（一至两页），英文摘要，二者应基本对应。它是论文内容的高度概括，应说明研究目的、研究方法、成果和结论，要突出本论文的创造性成果或新的见解、用语简洁、准确。论文摘要的关键词3至8个。关键词应为公知公用的词和学术术语，不可采用自造字词和略写、符号等，词组不宜过长。

英文摘要采用第三人称单数语气介绍该学位论文内容，目的是便于其他文摘摘录，因此在写作英文文摘时不宜用第一人称的语气陈述。叙述的基本时态为一般现在时，确实需要强调过去的事情或者已经完成的行为才使用过去时、完成时等其他时态

关键词: ， ， ， ，

Abstract

Abstract …………..

Key words:, , ,

目录

[专用术语注释表 V](#_Toc528056498)

[第一章 绪论 1](#_Toc528056499)

[1.1 研究背景和研究意义 1](#_Toc528056500)

[1.1.1 《章内标题1.1.1》 1](#_Toc528056501)

[1.1.2 《章内标题1.1.2》 1](#_Toc528056502)

[1.2 国内外的研究现状 2](#_Toc528056503)

[1.2.1 《章内标题1.2.1》 2](#_Toc528056504)

[1.2.2 《章内标题1.2.2》 2](#_Toc528056505)

[1.3 研究内容和主要工作 2](#_Toc528056506)

[1.4 论文的结构和安排 2](#_Toc528056507)

[第二章 差分隐私的技术概要 3](#_Toc528056508)

[2.1 差分隐私的相关概念 3](#_Toc528056509)

[2.1.1 基本定义 4](#_Toc528056510)

[2.1.2 噪声机制 4](#_Toc528056511)

[2.2 差分隐私的算法性质 4](#_Toc528056512)

[2.2.1 并行机制 4](#_Toc528056513)

[2.2.2 串行机制 5](#_Toc528056514)

[2.3 差分隐私的实现形式 5](#_Toc528056515)

[2.3.1 数据的噪声扰动 5](#_Toc528056516)

[2.3.2 发布中的信 5](#_Toc528056517)

[2.4 本章小结 5](#_Toc528056518)

[第三章 日志采集完整性和实时性算法 6](#_Toc528056519)

[3.1 研究背景 6](#_Toc528056520)

[3.2 日志采集流程和相关概念 6](#_Toc528056521)

[3.2.1 日志采集框架Flume相关组件 6](#_Toc528056522)

[3.2.2 日志采集的整体流程图和框架 7](#_Toc528056523)

[3.3 日志采集完整性研究 8](#_Toc528056524)

[3.3.1 队列管理算法 8](#_Toc528056525)

[3.3.2 基于AQM的分类器实现 9](#_Toc528056526)

[3.4 日志采集实时性研究 13](#_Toc528056527)

[3.4.1 日志采集实时性策略分析 13](#_Toc528056528)

[3.4.2 日志采集的优化策略 14](#_Toc528056529)

[3.5 仿真实验 16](#_Toc528056530)

[3.5.1 实验结果与分析 16](#_Toc528056531)

[3.6 本章小结 18](#_Toc528056532)

[第四章 基于差分隐私的集成算法 19](#_Toc528056533)

[4.1 分类的集成算法 19](#_Toc528056534)

[4.1.1 集成算法的种类 19](#_Toc528056535)

[4.1.2 集成算法的实现形式 19](#_Toc528056536)

[4.2 基于差分隐私的Adboost算法 19](#_Toc528056537)

[4.2.1 Adboost与差分隐私的结合 19](#_Toc528056538)

[4.2.2 Xgboost与差分隐私的结合 19](#_Toc528056539)

[4.2.3 GBDT与差分隐私的结合 19](#_Toc528056540)

[4.3 基于差分隐私的随机森林 20](#_Toc528056541)

[4.3.1 基于差分隐私的随机采样 20](#_Toc528056542)

[4.3.2 差分隐私的ID3算法构建随机森林 20](#_Toc528056543)

[4.4 仿真实验 20](#_Toc528056544)

[4.4.1 实验结果 20](#_Toc528056545)

[4.4.2 实验分析 20](#_Toc528056546)

[第五章 原型系统的设计与实现 21](#_Toc528056547)

[5.1 差分隐私的引用场景 21](#_Toc528056548)

[5.2 差分隐私算法的模型设计 21](#_Toc528056549)

[5.2.1 日志数据采集 21](#_Toc528056550)

[5.2.2 日志数据分类 21](#_Toc528056551)

[5.3 差分隐私集成算法的模型设计 21](#_Toc528056552)

[5.3.1 系统日志信息采集 21](#_Toc528056553)

[5.3.2 系统日志信息分类 21](#_Toc528056554)

[5.4 实现环境搭建 22](#_Toc528056555)

[5.5 本章小结 22](#_Toc528056556)

[5.5.1 实验结果 22](#_Toc528056557)

[5.5.2 实验结果 22](#_Toc528056558)

[第六章 总结与展望 23](#_Toc528056559)

[参考文献 24](#_Toc528056560)

[附录1 程序清单 25](#_Toc528056561)

[附录2 攻读硕士学位期间撰写的论文 26](#_Toc528056562)

[附录3 攻读硕士学位期间申请的专利 27](#_Toc528056563)

[附录4 攻读硕士学位期间参加的科研项目 28](#_Toc528056564)

[致谢 29](#_Toc528056565)

# 专用术语注释表

**符号说明：**

|  |  |
| --- | --- |
|  | 多普勒扩展 |
|  | 多普勒功率谱 |
|  |  |
|  |  |

**缩略词说明：**

|  |  |  |
| --- | --- | --- |
| ABS | Adaptive Binary Splitting | 自适应二叉树分割 |
| AES | Advanced Encryption Standard | 高级加密标准 |
| AMS | Adaptive Multi-tree Search | 自适应多叉树 |
| AQS | Adaptive Query Splitting | 自适应查询分割树 |
| ARAP | Anonymous RFIDAuthentication Protocol | 匿名RFID认证协议 |
| ASC | Allocated Slot Counter | 分配计数器 |
| BSCTTA | Bi-Slotted Collision Tracking Tree Algorithm | 双时隙碰撞跟踪树算法 |
| I4QT | Improved 4-ary Query Tree | 改进型四叉查询树 |
| IAMS | Improved Adaptive Multi-tree Search | 改进型自适应多叉树 |

本部分内容非强制性要求，如果论文中所用符号不多，可以省略《专用术语注释表》。

# 绪论

* 1. 研究背景和研究意义

《正文》×××××

正文是学位论文的主体。内容可因研究课题的性质不同而有所变化。一般应包括：绪论、文献综述、理论分析、计算方法、实验方法及结果的分析讨论、见解和结论。

正文一律用阿拉伯数字编排页码，页码在底部居中。

以下为正文中参考文献示例，参考文献编码使用右上标标注：

这是一种重要的盲自适应算法，最早是由Godard提出来的[1]。

移动无线信道中的时间色散与频率色散可能产生4种显著效应[2]，这些是由信号、信道及发送速率的特性引起的。

本文采用的是文献[3]中介绍的一种转换方法，通过限制信道的带宽可以得到其对应的均匀抽头延迟模型[4-5]

* + 1. 《章内标题1.1.1》

《正文》×××××

（1）正文内一级序号（首行不缩进）

《正文》×××××

（a）正文内二级序号

（b）正文内二级序号

（2）正文内一级序号（首行不缩进）

《正文》×××××

（a）正文内二级序号

（b）正文内二级序号

* + 1. 《章内标题1.1.2》

《正文》×××××

* 1. 国内外的研究现状

《正文》×××××

* + 1. 《章内标题1.2.1》

《正文》×××××

* + 1. 《章内标题1.2.2》

《正文》×××××

* 1. 研究内容和主要工作
  2. 论文的结构和安排

《正文》×××××

# 差分隐私的技术概要

《正文》×××××

* 1. 差分隐私的相关概念

《正文》×××××

《公式》 （2.1）

《正文》×××××

《公式》 （2.2）

《正文》×××××

图2.1 图的名称

《正文》×××××

表2.1 表的名称

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

《正文》×××××

《正文》×××××



图2.2 图的名称

《正文》×××××

表2.2 表的名称

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

《正文》×××××

《正文》×××××

* + 1. 基本定义

《正文》×××××

* + 1. 噪声机制

《正文》×××××

* 1. 差分隐私的算法性质

《正文》×××××

* + 1. 并行机制

《正文》×××××

* + 1. 串行机制

《正文》×××××

* 1. 差分隐私的实现形式

《正文》×××××

* + 1. 数据的噪声扰动

《正文》×××××

* + 1. 发布中的信

《正文》×××××

* 1. 本章小结

《正文》×××××

# 日志采集完整性和实时性算法

* 1. 研究背景

在云计算大数据快速发展的今天，用户在服务器上进行相关操作，必然会留下各种行为日志，系统日志信息，这些日志信息表面上只是对用户行为的一些记录。但是实际上存在着某种关联，能够反应用户的一些行为。所以对日志信息进行实时而且完整的采集就显得尤为重要。现在比较主流的日志采集系统都是使用一些开源的框架，这些框架可以提供基本的采集功能，但是如果需要达到自己的采集需求，需要做一些定制化的优化和调整。比如说加上一个Kafka的消息队列，或者说自定义消息的拦截器以及采集格式等等，通过这些改动才能够使得采集系统能够满足需求。但是最为重要的还是日志采集的实时性和完整性问题，实时性达不到要求就不能够进行实时处理，等同于线下的批处理，完整性达不到要求数据就存在缺失的问题，直接影响到后续对日志的分析。本文提出了基于Flume采集框架的日志采集完整性和实时性的方法，最大程度地保证采集的完整性和实时性。

* 1. 日志采集流程和相关概念

Flume是一种分布式的，可靠的系统。能够非常高效地将不同来源的日志进行收集和聚合。同时Flume也能进行数据源的传输，外部源可以向Flume发送事件（比如Web服务器），Source接收之后通过Channel来分发事件，Sink来消耗事件，并将其存储到数据库中（比如HDFS），或者将其转发到流中的下一个Flume代理来进行异步运行。

但是很多情况下，我们需要指定数据源，Flume中可以通过RPC机制来指定一个文件。不同类型的文件收集之后，可能需要合并分类之类的操作，此时就必须设置多代理流程或者复用流。但是Source必需是源实例，只有源实例才能指定多个通道，实现数据的合并和分类。定义好流程之后，需要设置每个源，通道，容器的属性，继而按照自己的需求去配置多流代理或者扇出流。这里需要强调的是，源和容器都可以自己制定一个选择通道的策略，这样可以使用不同的通道进行数据传输的分流，最大效率地保证传输的速度。

* + 1. 日志采集框架Flume相关组件

Source是一个可以接受不同数据中心产生数据的组件，可以通过监听端口来获取数据也可以从本地的文件中读取到数据。需要注意的是每个Source都必须至少连接到一个Channel上，对于一个Source将数据复制到多个Channel的情况也是很常见的。关于Channel，一般我们将其当作一个被动的组件来看待。它的主要责任就是充当Source和Sink之间的通道，行为上等价于队列的作用。同样一个Sink只能够读取一个Channel中的内容，但是也有不同的Sink读取同一个Channel的可能，不过这些Sink中只有一个可以读取到Channel中的内容。对于Sink，主要是不间断地从Channel中来读取或者删除事件，Sink将获取到的事件传递到下一个地点，可能是PRCSink也可能是最终的地点。一旦可以确认下一个地点数据的安全，Sink就会从Channel中删除这些数据。此外flume本身并不限制一个agent中各个组件的数量，所以Source可以接受不同目标的数据，最后在处理中，可以设置Channel处理器，拦截器或者Channel选择器来将数据传输到Channel中。拦截器一般来说就是一段正则匹配的代码，可以基于接收到的数据的类型和存储方式来决定哪些数据必须写入到Channel中，哪些数据需要过滤掉一部分再写入Channel。

* + 1. 日志采集的整体流程图和框架

用户在云服务器上进行各种操作，所产生的日志种类繁多，而且格式不一。要将这些日志都采集汇总到一起是一件比较困难的事。不同的日志类型需要采用不用的采集方式，所以需要针对用户行为日志进行针对性的采集。这就需要充分发挥Flume3个组件的功能。具体的功能如下所示：

（1） 利用Source源支持多类型输入流的特性，使用多Source源来满足不同类型的日志。避免出现日志采集由于方式不对导致信息的损坏和丢失。

（2） 使用Channel能够使得Source和Sink运作在不同速率上的特点，合理使用Channel的缓冲特性来协调整体的采集过程

（3）利用Sink支持完全自定义序列化数据的特点，对多Source源采集到的数据进行批量移除后者传输到下一个Agent中。

利用上述特点，设计的流程图如图3.1所示：

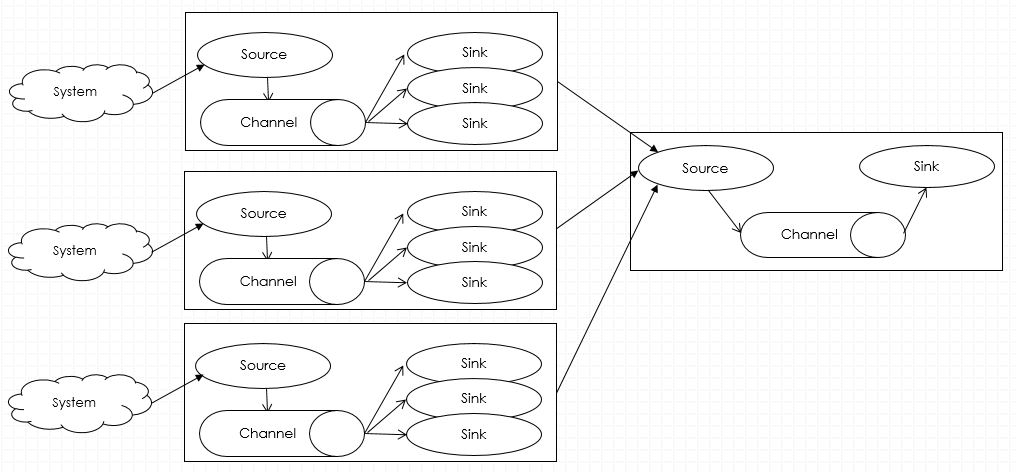


图3.1 日志采集架构图

从图中可以看出，对于不同类型的日志使用多个Source来进行采集，此外Sink从Channel中获取日志数据时使用了多个Sink同时进行，所有的日志采集完成之后没有直接写入到文件中去，而是又接了一个Agent，这样做的好处在于提升采集系统的灵活性，如果需求变化了，只需要修改一下配置文件中的参数就可以改变采集的过程，如果需要多采集到的日志进行过滤也可以在传输到Agent之前做一个预处理，或者不同的日志文件需要保存在不同的地方，以上的架构也可以灵活应对。

* 1. 日志采集完整性研究

Flume中的Channel在采集数据的过程中分为两种类型，一种是Memory Channel，另一种是File Channel。File Channel是磁盘上的缓存机制，每次都将日志信息保存在磁盘中，可以避免由于宕机或者重启导致的日志数据丢失。Memory Channel是使用的内存缓存机制，因为在内存中保存日志数据，所以具有很高的吞吐量。但是内存的大小是有限制的，不能想磁盘那样具备大量的缓存空间，所以当内存被全部占满之后，Channel中的队列将无法继续接受数据，此时如果队列仍然有日志数据传输进来，就会造成日志数据的丢失。由于本文的场景需要的是更高的实时性，所以采用的是Memory Channel的形式，所以需要在数据的完整性方面做进一步的优化，最大程度地保证数据的完整性。

* + 1. 队列管理算法

当数据的采集速率与后续的传输速率不一致的时候，会导致整个采集系统不协调，在缓冲队列中，数据便会存在冗余和丢失的情况，比如说采集速率过快，队列始终是满的或者近似满状态，那么短时间内没有将这些数据存起来，就会造成数据的丢失。从而影响到数据的有效性和完整性。所以需要在Flume采集框架中使用队列管理算法来协调Source和Channel的运行速率。

常见的队列管理算法都是通过采用检查缓冲队列的平均长度，然后根据平均长度来协调采集和传输速率，同时尽量保证队列的平均长度在队长的最小值和最大值之间。具体来说就是通过两个参数来协调。

如果缓冲队列的长度在之间，采集速率在之间，数据的传输速率在之间，当发现一段时间内的平均队列长度在minth附近波动，那么就说明采集过于缓慢，传输过滤快速。就需要增大采集速率，降低传输速率，，，当发现一段时间内的平均队列长度在maxth附近波动时，就需要降低采集速率，增大传输速率，，。这种方式简单效果也不错。但是存在一些问题。比如参数的设定，是，还是。此外，参数或者过大会造成采集和传输速率一直大幅度震荡，影响队列的稳定性。如果采集过程不是很稳定，那么将需要很长一段时间，整个队列才能够平稳下来。

对于常见的队列管理算法，可以通过采集源的数量以及采集源的类型来辅助判断当前队列的状况。进而增加队列平均长度的稳定性。但是这种通过队列长度来进行判断的思路是以队列的内存空间来作为代价。

* + 1. 基于AQM的分类器实现

自动队列管理还是很多的实现方式，虽然概率丢弃是实现AQM的一种有效的策略，其侧重点在概率的计算上和权重的更新方式上。但是丢弃的策略不一定是最优的，是否需要丢弃或者说达到了丢弃的标准实际上是一个目标最优化的问题，通过决策的手段对采集的数据源以及写入的地址数量等进行判断，可以充分发挥这些辅助信息的价值，这样就可以从另一个角度出发去实现队列的管理。分类器的思路由此而来。

但是在进行算法之前，需要将有用的信息进行标量化，同时归一化到0和1之间，这样日志数据的分布就在一个合理的范围内。考虑到不同的队列状态具有不同的特性，使用一个简单的函数来拟合所有的状态是不合理的。所以需要从数据本身的特点出发。

要了解队列的状态，必须通过一些变量来作为判定依据。队列长度是最多被作为队列状态变量的，毋庸置疑，这个变量确实非常的有效，一个是这个变量特别好观测获得，另一个是他与队列的整体协调性有着比较密切的关系。获取到队列的状态信息，可以将队列的长度保持在一个期望的范围内，这就需要有效的协调采集速率和传出速率。一般意义上来说，我们能够利用的信息越多，那么最终所采取的控制措施就越科学。假设一个队列所包含的有效信息有n个，那么就相当于存在一个n维度的状态空间，空间中的每一个点都是队列的一种状态，这样通过算法来将这些点划分成两部分就等价于多维空间的二分类。

但是如果完全凭借队列长度这个指标去判断队列的状况，在某些情况下是可能得到错误的结论。

我们先考虑以下两种情况

第一种：缓冲区为的队列，队列长度的变化率为

第二种：缓冲区为的队列，队列长度的变化率为。

这时候，去判断哪种情况下队列可能会拥堵是很困难的，通过单一的队列长度的因素去判定是不合理，也许平均的队列长度会稍微缓和一点，但是还是很难在很多种情况下得到正确的结果。或许将队列长度的变化率也作为一个衡量的指标，可能得到的结果会准确一点。从另一个角度来看，队列长度的变化率其实是一个主动的队列管理方式，能够使得队列的管理更为主动化。这里也考虑过将采集源的数量等因素也一并加入到判定中去，这样能够综合更多的因素，使得结果更加精确。但是实验发现如果添加其他的因素作为判定条件，并没有提升缓冲队列的效果，有些时候反而会造成精度的扰动，这是因为过多的判定条件使得分类的特征过于复杂，而很多特征往往不是数据的主要方向，影响了日志数据特征的提取，从而降低了队列的判断精度。

根据以上的分析，我们选定队列的及时长度以及队列的变化率做为队列状态的衡量指标。由于整个算法是一个监督式的过程，所以需要通过数据采样来训练算法。我们采取作为数据采集的周期，使用作为队列的长度，作为队列的长度变化速率，那么可以得到一个队列的样本序列。根据以上的定义，那么某一时刻队列的变化速率就可以表示为。

在此基础上，构建一个两种类型的集合，集合一代表数据采集过快，集合二代表数据采集过慢。集合一，集合二，其中表示当前的队列长度和期望长度之间的差值。表示这个差值的变化速率。样本划分好之后，只需要依据队列长度和队列变化速率做一个二分类。考虑到队列长度和队列长度的变化速率这两个指标存在这一定的联系。所以使用向量投影的方式进行特征的提取，最终的算法能够使得集合一和集合二样本的均值尽可能地偏离，同时集合一和集合二内部的方差尽可能的小。

要获取集合的均值和方差，首先要获取集合的中心点，对于集合，如果其样本个数为，其中心点为：

在此基础上通过计算两个元素之间的协方差矩阵来评估两者之间的偏差。对于中心点，可以求得其协方差（去除分母）为：

同理可以求得集合的中心点，以及去除分母的协方差，这里不使用标准的协方差定义是因为协方差适用于最终的损失函数的最优值求解，分母在运算过程中会被消去，这里为了表达式的简洁，所以去掉，对最终的结果不会造成任何影响。

至此，可以定义整个优化过程的损失函数，对于两个集合，我们将其投影到一条直线上，那么集合投影到直线上的值就变为，选取中心点来代表整个合计的分布情况，使用二阶范数来衡量两个集合投影值之间的差值，我们希望这个差值越大越好，这样能够使分类的精确度得到提升。另一方面需要降低集合内的方差，因为集合内部的每一个元素都是一个二元组，所以需要通过协方差的形式来代替通常意义上的方差。使得两个集合在投影上的协方差最小，这里对两者的协方差求一个平均来降低损失函数求解过程中的波动，也就是使最小。综合上述的两个方面，最终的损失函数定义如下：

将具体值代入得到：

一般来说都是最小化损失函数，但是这里的是去求解损失函数的最大值。为了求解损失函数，首先定义，那么可以代表协方差的一个趋势。另外定义，很明显代表了两个集合之间的差异。带入损失函数：

利用广义的瑞利熵定理，即对于非零的向量，以及共轭转置等与自身的矩阵，同时矩阵正定，如果为矩阵的特征值，那么存在一下的不等关系：

这样整个损失函数的最优解就转变成求解一个矩阵的最大特征值的特征向量。回到损失函数中，发现的方向是固定的，为了方便求解，将表示为，带入到矩阵的特征值求解中，得到最终解：

得到投影线之后，对于新的队列长度和队列变化速率，将其投影到特征直线上，根据投影的范围来将其进行划分。也有计算每个类别的概率密度函数，然后将新数据据投影之后，带入到每个类别的概率密度函数之后，得到属于每个类别的概率。这种方式计算繁琐，对于二分类来说，直接根据投影的区间就可以达到很高的分类准确率。

根据上述求解过程，给出整体的伪代码如下所示：

基于分类的队列管理算法

输入：队列的及时信息（队列长度，队列长度变化率），期待的队列长度

样本的采集周期，速率降低权值，速率提升权值，投影分界点。

输出：队列的调整方式

While 数据仍在传输:

If now\_time > last\_time + T：

，

last\_time = now\_time

else:

pass

=集合的样本数，=集合的样本数

求得

，

if 在的上方：

降低采集队列长度，增加传输队列长度

else：

增加采集队列长度，降低传输队列长度

* 1. 日志采集实时性研究

采集过程的实时性是数据采集的核心问题，不能够对数据进行实时的采集会直接影响到整个分析平台的整体效果。如果一个预警系统对日志采集不是实时性的，那么对其的分析处理也必然是线下的方式，这样就不能够依据日志对用户行为做出及时的判断，也就很难达到预警的效果。不难发现，大多数实时处理框架都需要依赖于数据的实时采集，数据的采集过程收到多方面因素的影响，针对不同平台的情况可以进行实时性的优化。比如说Linux中数据的存储地，数据的存储方式的选择，这些都会直接或者间接地影响到数据的采集效率。所以需要从多方面入手，才能够最大效率地提升采集的实时性。

* + 1. 日志采集实时性策略分析

要提升日志采集的实时性，就需要分析影响日志的采集速率的因素。进而选取当前采集系统能够支持的改进或者优化方式。针对一般的采集系统，我们可以从以下的几个方面入手：

1. 日志输出的目的地，有的采集系统偏向于将日志输出到控制台，有的采集系统偏向与将日志输出到文件系统中。这两种方式存在响应时间的差异。
2. 日志的输出格式，日志的输出格式越复杂，系统的压力就越大，在采集的量级很大的情况下，容易对系统的性能产生负担。降低系统的运行效率。
3. 日志信息的级别，日志信息的级别越低，信息量越多，内容也越多，高级别的日志信息往往都是比较重要的信息，而且内容很紧凑，简明。
4. 日志的输出方式，日志的输出方式大体分为同步和异步两种，同步和异步都有各自的优缺点。对采集的效率也有影响。
5. 日志缓存的大小，达到多少日志信息量就进行日志的传输，日志采集是当一批日志达到设定的阈值时，才会进行传输存储。使用最合理的缓存大小，也能够在一定程度上提升日志的采集效率。

根据以上的5点，考虑到Flume采集系统的特性，3个组件都可以进行相应的优化。Flume使用事务的形式进行采集，将各部分功能细化，使得优化和自定义变得不是那么困难，同时还提供了相应的接口函数供调用。可定制性比较强。

* + 1. 日志采集的优化策略

首先考虑Source端的优化策略，Source提供了拦截器的功能，可以使用正则匹配的方式去约束采集到的日志类型和信息。实现方式也比较简单，直接在配置文件中添加相应的正则匹配公式就可以了。当然也可以使用Interceptor接口来实现。

因为本文所需要的日志类型比较多，所以需要使用以上两种方式，使用配置文件中的hostHeader来限定采集的主机Ip，使用timestamp来添加时间戳拦截器，方便后续Sink可以按照时间戳进行分桶。

复杂的正则匹配需要使用接口来实现，因为Flume的拦截器需要在进入Channel之前就完成相应的操作，这样才能够通知后续的组件进行运作，所以Flume在实现基本的拦截器的功能时都没有设计太复杂，毕竟大数据量的拦截处理会影响整体的性能，而轻量级的拦截能够减少无用信息的干扰，提升整体的采集效率。这里通过Interceptor的接口来实现，只需要重写初始化和正则化方法就可以了。

在Channel中能够进行的优化就是在于缓冲区缓存大小的设定，这个值需要根据采集的日志类型来决定，不同类型的日志大小不一致，所以这个需要根据Source端来动态分配。经过实验，设定不同数量的Source分别对日志进行采集，对于30M左右的日志信息使用不同的缓存大小来测试采集时间，得到的结果如图3.2所示：

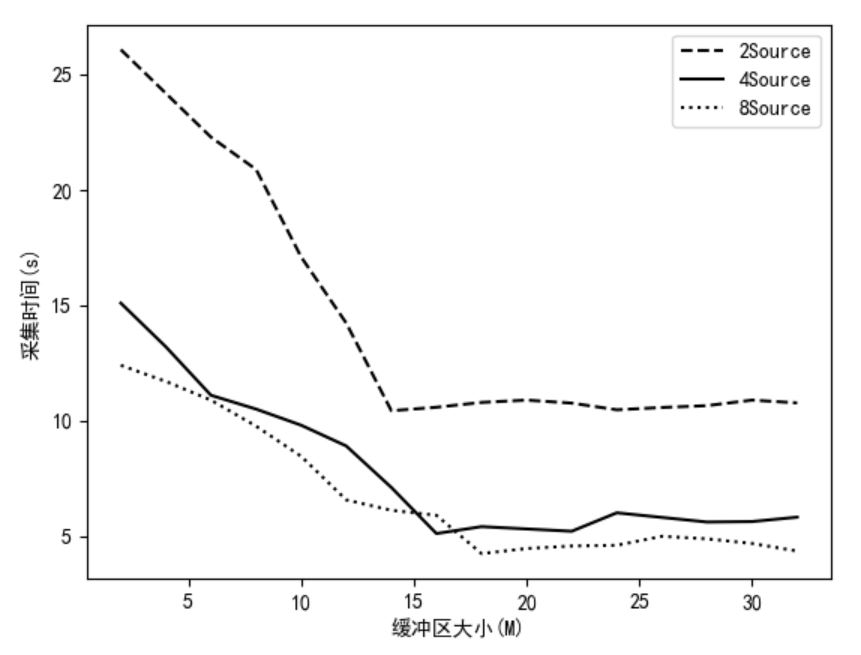


图3.2 缓冲区大小实验图

从上图中可以发现一般来说缓存大小设置为16M是比较合理的选择，此外多个Source对同一个日志进行采集能够提升效率，但是Source过多之后效率提升并不明显，另一方面，可以适当的增加缓存大小来提升采集效率，如果日志内容很多并不建议增加缓存，相应地增大缓存并没有对采集速率有较大的提升。

接着是Sink端的优化策略，Sink端可以进行优化的地方比较多，首先是日志输出的级别，日志的级别一般来说分为INFO，WARN，ERROR，DEBUG，TRACE这几种。对于采集而言，只需要关注前面三个，后面两个是调试和开发人员常用的。INFO是一些业务逻辑或者一些操作的运行信息，目的是为了让高级用户或者系统管理员能够理解当前的运行状态。所以信息比较详细，内容多。WARN日志主要是为了提醒用户当前系统存在一些问题或者潜在的问题，但是没有影响到系统当前的运行状态。ERROR类型的日志表明当前系统存在问题，不能够继续运行，需要立即解决，都是比较直接的错误信息，指明错误的类型和位置。从采集到的日志来看，IFNO的信息量要大于WARN和ERROR的日志信息量。所以在采集过程中可以使用Sink组从Channel中读取数据，这样一方面可以提升采集的效率，因为使用Sink组相当于开启了多线程模型，每个sink都是一个独立的线程，另一方面可以对采集的日志类型进行控制，这里为了保证整个系统的性能，将INFO，WARN，ERROR的采集比例控制在。Sink组的负载均衡可以直接使用Sink自带的round\_robin来实现。

关于日志的输出格式，对采集效率的提升是间接的。日志的输出如果不做修改直接使用默认的格式，也是可以输出的，但是Sink中需要需要根据日志的内容进行分桶输出。这样的话在对不同的日志进行筛选的时候会消耗比较多的时间，但是日志如果都规整话，那么会提升整体的处理效率。这里采用PatternLayout的自定义格式来解决，主要实在时间上的一个格式化，使用的格式。这样会产生如下形式的样例日志信息格式（使用auto日志作为样例）：

Accepted password for root from 192.168.133.1 port 52320

这样的话Sink端的消耗Channel中的日志信息的时候就可以直接根据时间和日志类型做一个直接的区分，输出到指定的地方。

对于日志文件的输出地，一般来说可以输出到控制台或者文件系统。有些监控需要输出到控制台进行认为的监控和审查，有的需要输出到文件中进行后续的处理。对于实时采集来说，直接输出到控制台是很快速的，但是日志信息的价值并没有被挖掘出来。虽然说可以接一个分析系统之后在输出，但是这样涉及到在Flume中添加一个分析系统。这样无疑增加了系统的复杂了，调度也需要消费时间，并没有降低整体的运行时间。所以本文采用输出到文件中的方式，让后续的分析系统直接读取文件中的日志信息进行分析。这样将两个功能区分开，各自独立运行。另一方面输出到文件中相当于做了一个日志的备份，使得整个系统更加完善，能够应对宕机等突发情况。

最后需要考虑日志的输出方式是同步还是异步。同步的方式，每个Sink都会起一个线程从Channel中获取数据，要等到每个线程都成功读取到数据，然后才能输出到文件中去。虽然每个线程之间都是独立运行的，但是需要进行同步操作之后，控制器会发布指令，表明现在可以向文件中进行写入操作。异步的方式则不同，每个Sink产生的线程获取到Channel中的数据之后不用等待其他线程获取完数据的信号，直接可以将日志信息写入到文件系统中。由于每个线程写入文件的大小不一致，所以写入的时间消耗也不一致，这样会导致每一批的日志写入可能出现日志重复的情况。但是省去了线程间的同步操作，本文的日志是写入到文件系统，有的采集系统会将日志写入到数据库中，写入数据库的操作更加消耗时间和内存，线程间的同步开销会被放大。所以，使用异步的操作能够一定程度上提升实时采集的时间消耗，提升采集的整体效率。

* 1. 仿真实验

为了验证采集系统中加入队列管理算法能够保证数据采集完整性，使用Flume去采集Linux系统中指定文件里文本的变化以及文件的增加。测试在各种情况下，文本中日志采集的完整性。此外，为了验证日志采集效率的提升，测试了加入改动和不加入改动的情况下对用户文件操作日志采集所消耗的时间。

* + 1. 实验结果与分析

实验时模拟用户创建文件，并向文件中写入文本所产生的日志，使用Flume以及加入队列管理算法的Flume分别采集日志，最后对比与实际日志之间的匹配度。这里匹配度使用正确匹配的条数占总日志条数的比例。实验中在固定Sink数为6的情况下，分别使用不同的Source源进行日志数据的采集，每种情况分别做100次取平均的匹配度。这样可以模拟Source和Sink在不同的配置下，数据的采集效率。因为在实际的采集过程中可能用户的一种行为产生的日志并不能覆盖所有的Source，也就是说有些Source是没有数据采集的。将Source数量从1到10个情况的结果放到一起进行比对，得到的结果如图3.3所示：

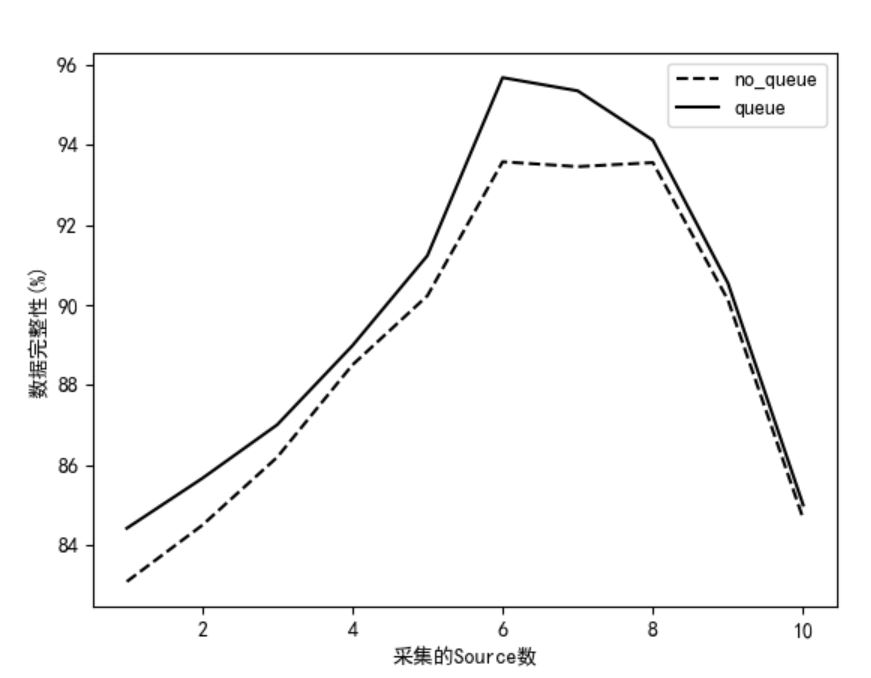


图3.3 队列管理算法实验结果

从图中可以发现，在6Source和6Sink的情况下，数据采集的完整性比较高，加入队列管理算法之后对数据采集的完整性有较大程度的提升。在2Source的情况下数据采集的完整性处于低值，原因是采集速率和传输速率之间的不协调导致的，这时候加入队列管理算法，对完整性的提升还是比较理想。在8Source的情况下，采集的完整性开始降低，原因在于日志的采集速率过快，缓冲队列也无法完全协调，导致日志数据存在较多的丢失。这个时候队列管理算法也不能够提供有效的完整性提升。

关于采集效率的实验。通过Flume自定义配置以及相应接口的重新实现，对用户一定操作所产生的系统日志进行采集，获取整个过程采集所消耗的时间。将标准的Flume配置，自定义配置，加入队列管理的标准配置以及加入队列管理的自定义配置所消耗的时间放到一起做一个对比，Source和Sink的配置还是采用6-6的设定，这样避免了采集源和传输不匹配对实验结果造成的影响。上述的4种状况分别对应关系如表3.1所示：

表3.1 采集配置情况对应表

|  |  |  |  |
| --- | --- | --- | --- |
| Situation1 | Situation2 | Situation3 | Situation4 |
| 标准配置 | 标准配置+队列 | 自定义配置 | 自定义配置+队列 |

使用上述4中配置分别对用户操作所产生的10M，20M，30M日志进行采集，得到最终的实验结果如图3.4所示：

图3.4 数据采集时间实验结果

从图中可以发现不同大小的日志数据使用自定义的配置总是可以降低采集的时间，在数据量比较大的时候效果比较明显，原因在于数据很多的时候，日志的格式种类也会变多，规范日志格式可以降低划分日志的时间，间接地提升了日志采集的效率。另一方面，加入了队列管理算法一定程度上增加了采集的时间，但是增加的幅度不是很大，还是要优于标准的配置所消耗的时间。

* 1. 本章小结

本章首先阐述了日志采集实时性和完整性的意义，然后介绍Flume采集框架的相关组件，并且根据当前需要采集日志的类型和方式设计了一套采集方案。接着针对采集过程中的日志完整性问题，提出了基于缓冲队列长度和队列变化速率的队列管理算法，根据实时的队列信息对其长度进行动态的调整。同时针对采集日志的实时性问题，对日志的输出级别，格式，输出地以及输出方式进行自定义的配置以及接口的重新实现。使用云环境中用户行为所产生的日志进行实验，从不同的维度进行对比，验证了队列管理算法和自定义输出配置的有效性，使得日志采集在完整新和实时性上得到提升。

# 基于神经网络的日志分析算法

《正文》×××××

* 1. 日志分析算法

《正文》×××××

* + 1. 传统的分类算法

《正文》×××××

* + 1. 基于神经网络的分类算法

《正文》×××××

* 1. 基于Densnet的分类实现

《正文》×××××

* + 1. 日志数据预处理

《正文》×××××

* + 1. 网络结构的设计与实现

《正文》×××××

* + 1. 模型训练与优化

《正文》×××××

* 1. 结构化日志信息的分析算法

《正文》×××××

* + 1. 结构化日志的处理

《正文》×××××

* + 1. 基于联级网络的结构化分析算法

《正文》×××××

* 1. 仿真实验

《正文》×××××

* + 1. 实验结果

《正文》×××××

* + 1. 实验分析

《正文》×××××

# 原型系统的设计与实现

《正文》×××××

* 1. 预警系统的应用场景

《正文》×××××

* 1. 预警系统的架构设计
     1. 日志数据采集

《正文》×××××

* + 1. 日志数据分类

《正文》×××××

* 1. 差分隐私集成算法的模型设计

《正文》×××××

* + 1. 系统日志信息采集

《正文》×××××

* + 1. 系统日志信息分类

《正文》×××××

* 1. 实现环境搭建

《正文》×××××

* 1. 本章小结

《正文》×××××

* + 1. 实验结果
    2. 实验结果

《正文》×××××

# 总结与展望

《正文》×××××

《正文》×××××

# 

# 参考文献

[1] Zhang H Y, Zheng L, Cai L. Design and Analysis of Hierarchical Physical Layer Network Coding[J]. IEEE Transactions on Wireless Communications, 2017, 16(12): 7966-7981.

[2] 张贤达, 保铮. 通信信号处理[M]. 北京:国防工业出版社, 2000: 30-50.

[3] Larimore M, Treichler J. Convergence behavior of the constant modulus algorithm[C]. IEEE International Conference on ICASSP,2017:13-16.

**按照“GB/T 7714”格式列参考文献，请删除此行。**

# 附录1 程序清单

第三章 QAM系统CMA算法程序（QAM programme文件夹）

mainQAM\_MSE\_CMAvsCADAMA：QAM系统，画MSE曲线主函数

mainQAM\_SER\_CMAvsCADAMA：QAM系统，画SER曲线主函数

mainDQAM\_SER\_PhaseRotate：差分QAM系统，相位旋转下，画SER曲线主函数

mainDQAM\_SER\_Doppler：差分QAM系统，多普勒频移下，画SER曲线主函数

mainDQAM\_SER\_SNR：差分QAM系统，不同SNR下，画SER曲线主函数

第四章 QPSK系统CMA算法程序（PSK programme文件夹）

mainDPSK\_SER\_PhaseRotate：差分PSK系统，相位旋转下，画SER曲线主函数

mainDPSK\_SER\_Doppler：差分PSK系统，多普勒频移下，画SER曲线主函数

mainDPSK\_SER\_SNR：差分PSK系统，不同SNR下，画SER曲线主函数

FunQPSK：QPSK编码子函数

FunDeQPSK\_b：QPSK解码子函数(输出为比特)

第五章 CMA频域均衡系统程序（FDE programme文件夹）

main\_CMAFDE：CMA频域均衡主函数

FunCAZACSeGen：产生CAZAC序列子函数

Fun8PSK：8PSK调制子函数

FunDe8PSK\_s：8PSK解调 输出符号子函数

FunQPSK1：QPSK调制幅度为1子函数

FunDeQPSK1\_s：QPSK解调 输出符号 幅度为1子函数

FunSDCMA22f：频域均衡(2 2)SDCMA算法子函数

# 附录2 攻读硕士学位期间撰写的论文

[1]Li C, Sun X,Zhou X, et al. Optimal biased association scheme with Networks[J]. IEEE Transactions on Wireless Communications, 2017, 6(8):360-371. （SCI二区：000407686300001）；

[2]Li C, Ji Y. Modeling and analysis of epidemic spreading on community network [J]. Journal of China Universities of Posts & Telecommunications, 2016, 23(5):82-87.（EI：20170303250062）

[3]李春，网络路由技术，电子学报，已录用。

**百度学术，搜索论文，点击“引用”，按照“GB/T 7714”格式 复制出论文引用，后面加上（检索 信息），SCI论文需要写上几区。请删除此行。**

# 附录3 攻读硕士学位期间申请的专利

[1] 作者一，作者二，作者三. 专利名称，专利申请号，专利申请日期，专利授权日期；

[2] 张三，李四. 一种面向代理的安全传输方法，2007062.5，2016.1,2017.10；

[3] 张三. 一种实用的网络路由方法，200845610.5，2017,1,2018.1。

# 附录4 攻读硕士学位期间参加的科研项目

（1）项目类型，项目名称（项目编号）；

（2）国家自然科学基金，基于安全移动代理新一代网络研究(702710456)；

（3）国家自然科学基金，移动代理机制关键技术研究(601732578)。

# 致谢

主要感谢导师和对本论文学术研究有特别贡献的组织或个人：

对提供资助或者支持的基金、合同单位、企业、组织或者个人（基金项目应该包括基金名称、项目名称、项目编号、项目负责人、研究起止年月）；

协助完成研究工作或提供便利的组织或个人。