流式日志收集系统

架构设计说明书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件状态： | 版本 | 作者 | 修订日期 | 描述 |
| [ √ ] 草稿/审批稿  [ ] 正式发布  [ ] 作废  [ ] 保留 | 1.0 | 冯进 | 2016-10-18 | 文档初始化 |
| 1.1 | 冯进 | 2016-10-23 | 修改了归集层日志格式规范定义 |
|  |  |  |  |
|  |  |  |  |

目录

[1. 概述 3](#_Toc464738538)

[2. 软件架构 3](#_Toc464738539)

[2.1 数据视图 3](#_Toc464738540)

[2.2 逻辑架构 4](#_Toc464738541)

[2.3 日志格式规范 5](#_Toc464738542)

[2.4 日志存储规范 6](#_Toc464738543)

[2.5 部署架构图 7](#_Toc464738544)

[2.6 非功能性需求 8](#_Toc464738545)

[2.6.1 可靠性 8](#_Toc464738546)

[2.6.2 可扩展性 8](#_Toc464738547)

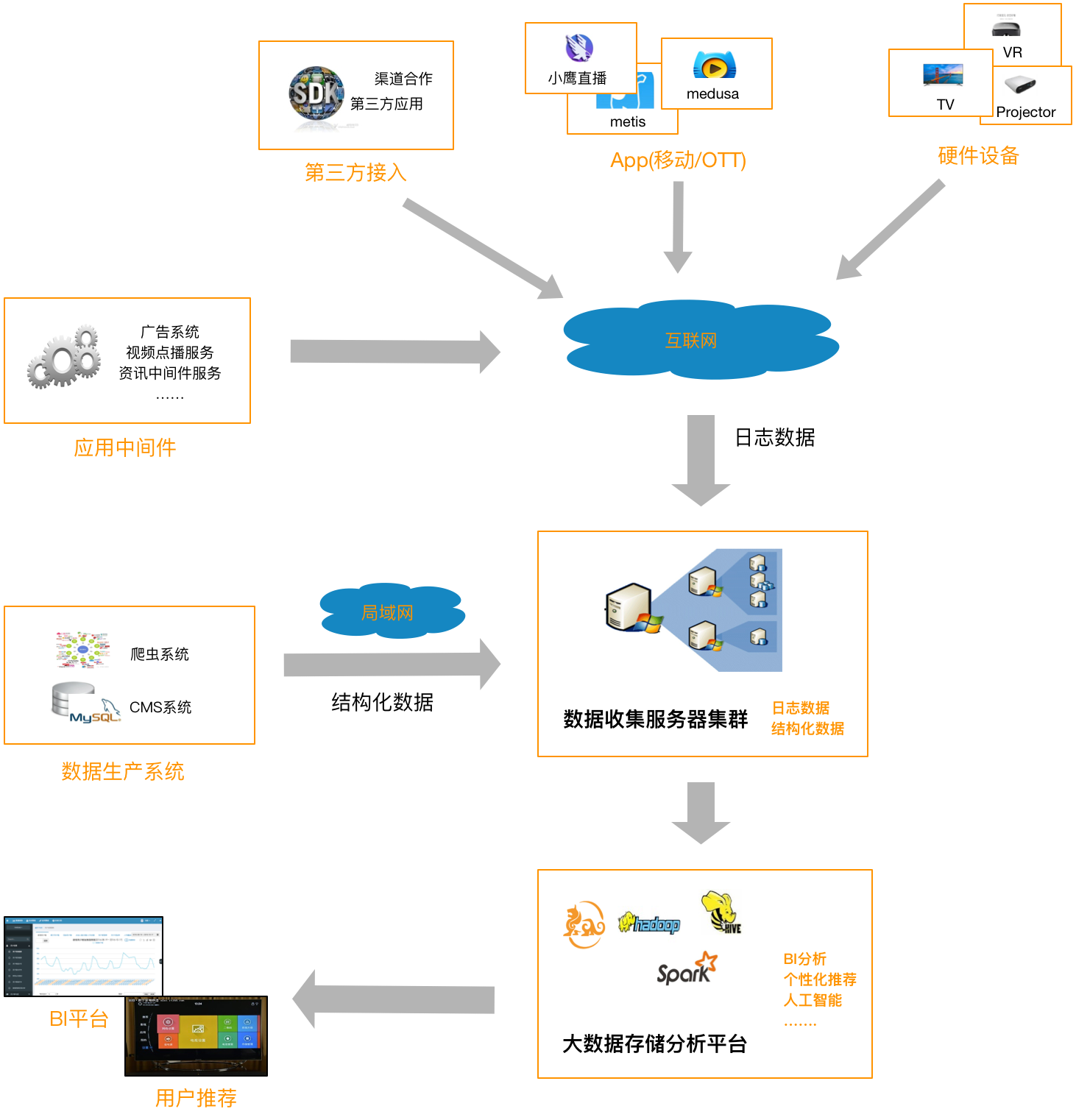
[2.6.3 可维护性 8](#_Toc464738548)

1. **概述**

来自于各个设备和应用系统的日志数据，是大数据平台重要的数据来源，日志数据天然具有时序特性，通过建立一套日志数据流收集系统，并对外提供实时数据流接口，一方面可以满足流式实时或准实时数据分析需求，另一方面离线分析系统通过定期收集流数据，实现数据批处理分析。

本文对日志流收集系统的设计和实现，从逻辑架构、日志格式规范、日志存储规范、日志流时序模型等方面进行分析说明，同时对一些非功能性需求作预案分析。

1. **软件架构**
   1. **数据视图**



**图1**

目前各个终端设备和应用系统产生的日志数据，是大数据平台中最主要的数据来源，日志数据将统一由日志流收集系统导入到大数据平台。目前考虑到的数据接入方如图1所示，图中的结构化数据暂不在日志流收集系统考虑范畴。

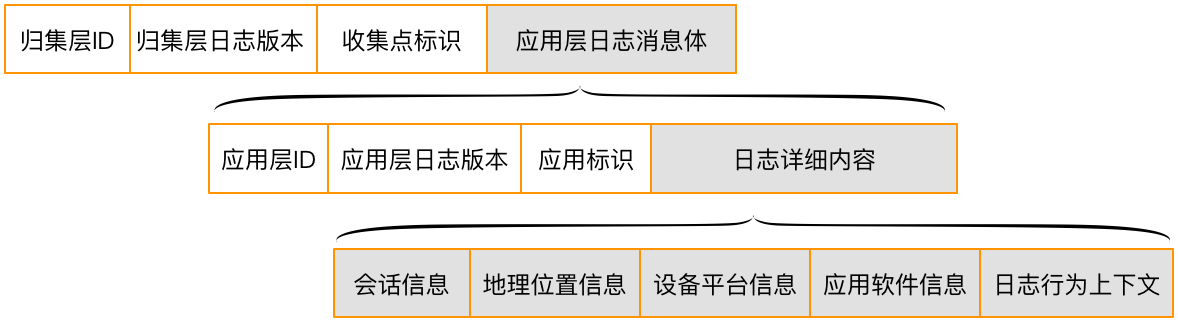
* 1. **逻辑架构**



**图2**

在逻辑层次上，分别为应用系统、日志接入层、日志归集层、日志应用层。

* 应用系统：产生日志数据的各类产品和应用，通过HTTP协议向日志接入层提交日志上传请求。
* 日志接入层：负责接收应用系统通过网络提交的日志上传请求，每一个请求形成一个日志消息体，并按归集层日志格式规范，在本地进行缓冲并持久化。
* 日志归集层：以实时流式的方式收集日志接入层产生的日志消息，并提供流式数据接口供下游系统接入。原则上该层仅负责日志的收集，不对日志数据进行复杂的解码操作。
* 日志应用层：从日志归集层流式数据接口消费数据，对日志数据进行解码后，形成应用层规范化格式消息体，持久化存储并提供流式数据接口，供下游系统接入。
  1. **日志格式规范**



**图3**

根据逻辑架构的分层处理设计，分别定义两层日志格式：归集层格式、应用层格式。

* 归集层日志格式：JSON字符串

|  |  |  |  |
| --- | --- | --- | --- |
| **域** | **字段名** | **约束** | **描述** |
| 归集层ID | msgId | 【必备字段】  全局唯一  255个以内不定长字符串 | 该字段可全局唯一确定一条消息，且实现上应该具有局部自增性 |
| 归集层日志版本 | msgVersion | 【必备字段】  2位定长字符串 | 用于日志消息格式的演进 |
| 收集点标识 | msgSite | 【必备字段】  255个以内不定长字符串 | 日志收集者位置标识符，标明具体是哪台日志接入服务器 |
|  | msgSource | 【必备字段】  255个以内不定长字符串  ngx\_log: nginx服务器日志文件 | 日志收集者类型标识符，标明是何种类型的的日志接入服务器 |
|  | msgFormat | 【必备字段】  255个以内不定长字符串  json：json结构体  tsv：以制表符为字段分隔符 | 应用层日志消息体的序列化格式 |
| 应用层日志消息体 | msgBody | 【必备字段】  有限大小的变长字符串（有限大小的HTTP请求体，外加有限数量的接入层服务器内置变量、部分HTTP头） | 由接入层服务器内置变量、部分HTTP头、HTTP请求体组成，具体内容由应用层决定，一般应该包括如下几个方面的信息：  method：日志提交方式(get/post)  contentType：日志内容格式  receiveTime：接入层服务器接收日志时间（精确到毫秒的时间戳）  realIP：日志提交者的IP地址  url: 日志提交请求的url  body: HTTP请求体 |

* 应用层日志格式：JSON字符串，原则上尽量采用单层结构。在日志应用层需要实现一个统一的应用中间件，将归集层格式日志转化为应用层格式，供其他业务应用使用。下表为用户行为日志的建议规范，非必备字段部分由具体的应用决定。

|  |  |  |  |
| --- | --- | --- | --- |
| **域** | **字段名** | **约束** | **描述** |
| 应用层ID | logId | 【必备字段】  全局唯一  255个以内不定长字符串 | 该字段可全局唯一确定一条消息，且实现上应该具有局部自增性 |
| 应用层日志版本 | logVersion | 【必备字段】  2位定长字符串 | 用于日志消息格式的演进 |
| 应用标识 | appId | 【必备字段】  255个以内不定长字符串 | 标识具体的日志提交者，对应业务中的某个应用系统 |
| 应用软件 | appSeries | 255个以内不定长字符串 | 应用软件系列号 |
|  | appVersion | 255个以内不定长字符串 | 应用软件版本号 |
|  | appBuildInfo | 255个以内不定长字符串 | 应用软件构建信息，如构建序列号、构建日期 |
| 会话信息 | sessionId | 255个以内不定长字符串 | 应用系统用户会话ID |
|  | accountId | 255个以内不定长字符串 | 应用系统用户账号ID，比如付费会员账户ID |
|  | userId | 255个以内不定长字符串 | 应用系统用户ID，唯一标识单个用户 |
| 地理位置 | realIP | IP地址字符串 | 日志提交者IP |
| 设备平台 | productModel | 255个以内不定长字符串 | 硬件设备型号字符串，形式因设备而异 |
|  | sysPlatform | 255个以内不定长字符串 | 操作系统平台字符串 |
| 日志上下文件 | logType | 255个以内不定长字符串 | 日志类型，每个应用自定义的用于区分不同日志行为的字符串 |
|  | receiveTime | 长整数 | 日志接收时间，日志接入服务器接收到日志消息的时间点，精确到毫秒的时间戳 |
|  | happendTime | 长整数 | 日志发送时间，由日志提交者提供，记录下用户操作产生日志的时间点，精确到毫秒的时间戳 |
|  | …… | ….. | 其他日志行为相关的上下文字段 |

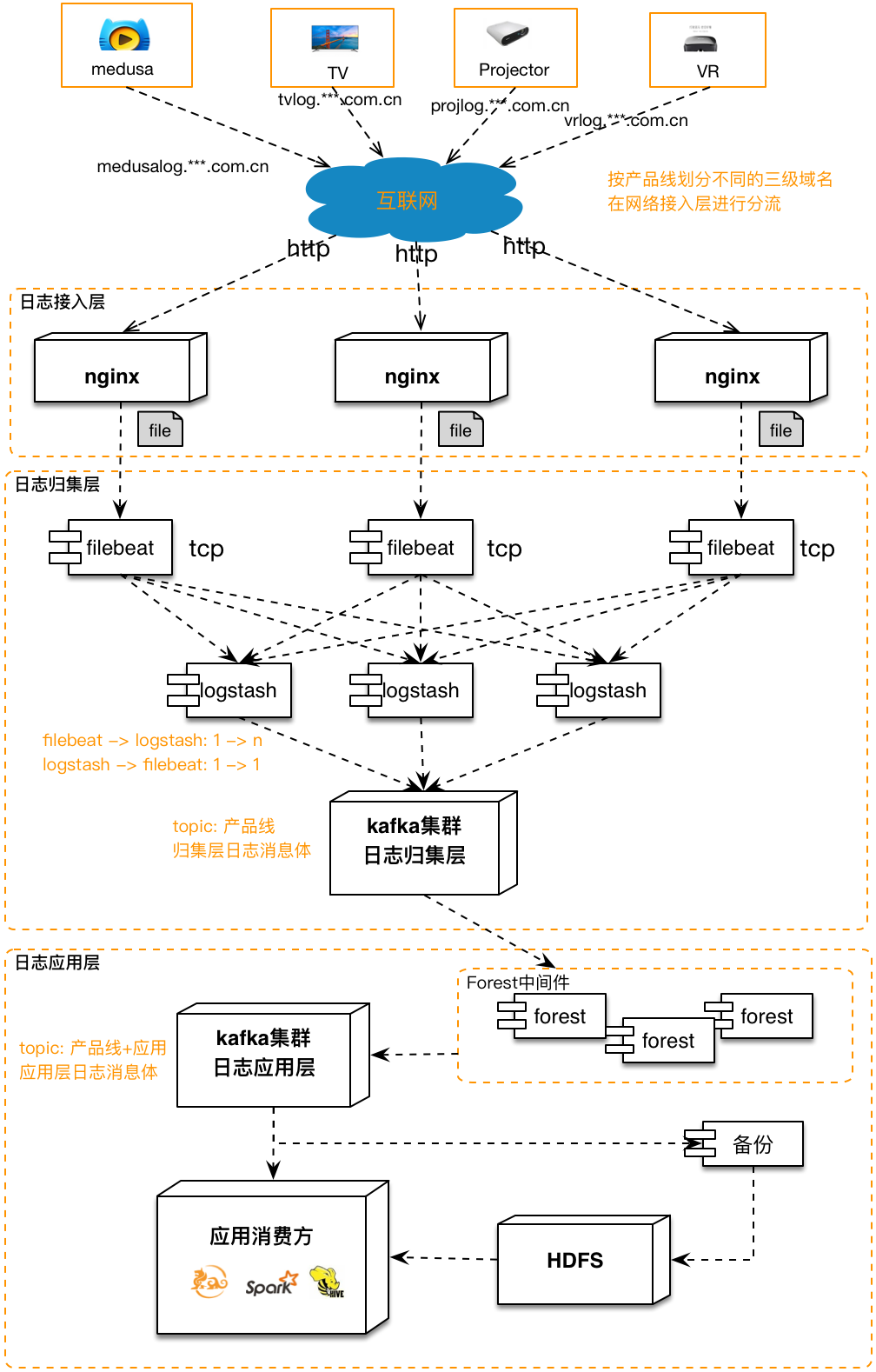
* 1. **日志存储规范**

**图4**

从日志内容存储角度看，整个日志内容需要经过4个层次：日志接入层、日志归集层、应用日志层、日志持久层，基于nginx+kafka+hdfs的技术栈，从数据存储位置、存储格式和存储时长3个方面进行了图4所示的规划。

特别说明的一点：对于日志持久层（HDFS），该层存储的数据一方面作为备份数据，一方面作为批处理数据源，如经过编码转换，产出数据仓库所需的数据。这样日志流收集系统从架构层面，支持后端流式消费和批消费两者模式。

* 1. **部署架构图**

****

**图5**

基于nginx+filebeat+logstask+kafka的技术栈，部署情况如图5所示。nginx为日志接入层核心，产生轮滚日志文件；kafka作为日志归集层和日志应用层核心，日志归集层通过filebeat+logstash实时收集日志接入层产生的日志文件，存储到kafka中；日志应用层通过Forest程序实时从日志归集层kafka消费日志数据，进行转换后存入到应用层kafka，供下游应用使用。

* 1. **非功能性需求**

### 2.6.1 可靠性

采用分层治理的策略，每一层集中关注自己的职责，一方面简化各层的实现，另一方面，分散风险，避免风险扩散。从而提高系统整体的可靠性。

* 日志接入层：主要负责网络流量的接入，目前该层功能还未完善，只是简单的做网络流量的接收落地，未来考虑进一步完善（比如认证机制、安全策略、流量分配机制等等）。
* 日志归集层：主要负责将日志数据从接入层各个节点集中收集，并以流式接口提供给下游应用消费，该层关注的是日志收集过程的及时性和可靠性，不关注具体的日志处理逻辑。归集层在日志应用层和日志接入层之间形成一个缓冲，隔离接入层对日志应用处理带来之间影响。
* 日志应用层：负责日志数据的业务逻辑处理，比如通过Forest程序按产品线+应用进行消息解码，形成应用层规范格式的消息体，并存入kafka集群，供下游应用使用。日志应用层的日志处理发生错误，不影响日志归集层的运作。

另外，在日志接入层和日志应用层都做了日志数据的备份，提高系统数据安全性。

### 2.6.2 可扩展性

在系统中的各个节点，尽量消除单点故障，同时又不引入更多的复杂性，是横向可扩展的基本保障。

* 日志接入层：依据产品线进行三级域名的切分，在网络层进行流量切分，使后端系统能更灵活的进行容量规划。目前采用域名解析到多台nginx服务器的方式，实现接入层在7层网络的横向扩展，未来可考虑引入专业的网络设备或HA软件方案，实现局域网4/7层网络应用的横向扩展。
* 日志归集层：在存储方面的扩展性，由kafka集群来保证；在日志流收集环节，通过filebeat后接多个logstash的方式实现横向扩展。
* 日志应用层：存储方面的扩展性，有kafka集群来保证；应用处理吞吐量方面，可通过kafka的消费群组增加应用部署节点来解决，比如对电视猫日志topic，可部署多个Forest程序，并加入到同一个消费群组中，加快日志处理速度。

### 2.6.3 可维护性

* 减少业务的侵入性：日志流收集系统根本职责是完成线上应用系统日志的收集，并向外提供数据流接口。为了尽可能减少业务上的变更对日志流收集系统产生的影响，我们定义统一的日志格式规范和存储规范，使日志收集过程的处理环节尽量标准化，对日志数据的业务逻辑处理上的差异性，交由具体的业务应用组件来处理。
* 提高可运维性：进行了日志接入层、日志归集层、日志应用层的划分，每个层面有明确的运维边界。做好整个系统的运维，往往需要做一些额外的工作，以下从可运维性角度列出了一下设计考虑：
  1. 日志接入层：来自网络层的日志请求及时落地到日志文件。主要是nginx服务器的运维监控。产生的日志文件定期做好备份，以备后端系统处理失败需要日志重放。
  2. 日志归集层：通过filebeat监控nginx日志文件，logstash连接filebeat，将日志数据导入到kafka。目前logstash、kafka集群有可用的监控工具供运维使用，另外从数据层面上，归集层日志ID可为运维提供恢复点信息。由于归集层日志ID具有局部自增特性，从而可通过一定手段定位到日志接入层原始日志文件的行，然后进行顺序重放。
  3. 日志应用层：应用层日志主要是kafka和Forest程序，另外该层也设计了一层日志数据备份，以应对下游应用程序消费失败需要的日志重放。同样，可通过应用层日志ID确定重放范围。