状态：受控

**密级：机密**

**日志系统2总体设计**

**说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档编号： |  | | |
| 编 写： | 刘协雍 | 编写日期： | 2018/6/13 |
| 审 核： |  | 审核日期： |  |
| 批 准： |  | 批准日期： |  |

**修订状况**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **修订**  **章节** | **修订内容简述** | **修订**  **人员** | **修订**  **日期** | **修订后**  **版本号** | **批准人** |
|  | **升级2.0** | **刘协雍** | **2018/6/14** | **v2** |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目 录**

[状态：受控 1](#_Toc516731457)

[1. 名词解释 3](#_Toc516731458)

[2. 总体说明 3](#_Toc516731459)

[2.1. preface 3](#_Toc516731460)

[2.2. 技术选型 3](#_Toc516731461)

[2.3. 主要技术考虑 3](#_Toc516731462)

[2.3.1. mq采用kafka 3](#_Toc516731463)

[2.3.2. 日志的收集采用filebeat 4](#_Toc516731464)

[~~2.3.3.~~ ~~日志落地采用文件方式~~ 4](#_Toc516731465)

[2.3.4. 日志落地采用HDFS文件方式 4](#_Toc516731466)

[2.3.5. 高并发日志收集 4](#_Toc516731467)

[2.3.6. 集群方案 4](#_Toc516731468)

[2.3.7. 关于系统分离 4](#_Toc516731469)

[2.3.8. 关于横向扩展 4](#_Toc516731470)

[3. 日志收集系统 4](#_Toc516731471)

[3.1. 系统架构图 5](#_Toc516731472)

[3.2. 系统安装 5](#_Toc516731473)

[3.2.1. 日志收集系统规划 5](#_Toc516731474)

[3.2.2. 组件及版本 5](#_Toc516731475)

[3.2.3. 组件下载 5](#_Toc516731476)

[3.2.4. 准备工作（三台服务器分别执行） 6](#_Toc516731477)

[3.2.5. Zookeeper配置 6](#_Toc516731478)

[3.2.6. Kafka 安装（在log2和log3中安装） 8](#_Toc516731479)

[3.2.7. Logstash 安装部署 10](#_Toc516731480)

[3.3. 系统监控 12](#_Toc516731481)

[3.4. 系统维护 12](#_Toc516731482)

[3.4.1. 常用命令 12](#_Toc516731483)

[3.5. 安全 12](#_Toc516731484)

[4. 日志调用接入 12](#_Toc516731485)

[4.1. beats 12](#_Toc516731486)

[4.1.1. 总述 12](#_Toc516731487)

[4.1.2. FileBeat centos7安装 12](#_Toc516731488)

[4.1.3. 配置使用 13](#_Toc516731489)

[4.1.4. 启动运行 13](#_Toc516731490)

[4.2. 高并发日志收集 13](#_Toc516731491)

[5. 日志分析 14](#_Toc516731492)

[5.1. 直通一体化操作及数据日志整合 14](#_Toc516731493)

[5.1.1. 逻辑处理架构图 14](#_Toc516731494)

[5.1.2. 业务处理流程图 14](#_Toc516731495)

[5.1.3. 总体设计考虑 14](#_Toc516731496)

[5.1.4. 详细设计 15](#_Toc516731497)

# 名词解释

|  |  |  |
| --- | --- | --- |
| **名称** | **说明** | **备注** |
| logstash | 日志关键组件 |  |
| kafka | 日志传输管道及缓冲 |  |
| flume | 日志传输组件类似flume |  |

# 总体说明

## preface

日志系统，包括从日志收集系统及日志分析模块。

日志收集系统，包括日志收集客户端、日志管道、日志后端处理

日志分析模块，包括日志分析、整合、计算、报表等

## 技术选型

日志收集采用logstash、kafka实现，日志落地采用HDFS

其中，对于一般系统的日志收集采用，系统原有日志+filebeat =>日志集群

对于高并发的日志收集，采用业务系统+rsyslog+filebeat=>日志集群

日志集群中，队列采用kafka

日志的落地，采用logstash，主要考虑的是它的落地性能

通过WebHDFS方式放入HDFS中

日志的分析采用spark等大数据工具进行分析，具体的分析方法根据不同的业务日志分析要求各有差别，日志分析的大章节中设计了针对不同的系统的分析实例，仅用参考

## 主要技术考虑

### mq采用kafka

几种mq针对现有系统流量都没问题，考虑后期扩展及应用生态，和软件间的配合度，采用kafka

### 日志的收集采用filebeat

filebeat比较于logstash更加轻量，占用系统资源小，也是官方主推的方式，生态也不错。考虑对原有系统的无侵入性，采用这种方案

### ~~日志落地采用文件方式~~

~~考虑方便排查问题及后期扩展性，保留一种基本的文件格式存储所有的数据，当然以后可能会变~~

### 日志落地采用HDFS文件方式

为了数据仓库做准备，采用HDFS方式存放文件~~，并定时进行文件归档，归档方式采用parquet（gzip）方式~~

暂时不考虑归档，后期磁盘不够再使用归档方式

### 高并发日志收集

对于日志压力特别的大的系统，不采用一种通用的http收集方式，让人主动去调用，这样大量的日志会影响整个网络，这种方式网络风险较高。

只能采用本地队列这种方式先落本地文件，采用rsyslog方式，即不会对网络有影响，也能支持高流量，不像log4j等方式对应用系统有io压力

### 集群方案

方案最初采用1个topic，2个partition，2个repication方式，保证高可用性

日志的存储量预订2天以上，最大日志预计不超过10M每条

单个业务系统，配置只走一个partition，保证日志的顺序

落地logstash暂时部署一个，后续存储方案，确定后再进行集群部署

### 关于系统分离

通过 fileds中的system\_id，分离各业务系统的日志落地文件夹

### 关于横向扩展

kafka支持横向扩展，可以添加partition方式，或者添加broker方式

logstash因为需要部署为服务，需要多机部署

### 高可用和可靠性

暂时只有通道和落地文件支持高可用，客户端采集的beats和收集kafka信息的logstash并不支持高可用，需要通过预警系统抓取可用状态。

这些组件短时间允许宕机，宕机后重启可继续服务，保障不丢失数据，宕机时间可配置

信息传递过程中支持信息的高速缓存、并为持久化的。

### 无侵害性

1、日志的采集对业务系统是没有侵入性的，不会占用日志进程

2、日志采集对网络是没有侵害性的

根据请求数量自动平衡处理能力，削峰填谷

通过kafka配置server.properties 的producer、consumer来限制访问的流量

quota.producer.default=10485760

quota.consumer.default=10485760

### 压缩

日志的传输是实时的，不进行压缩

### 关于日志监控

日志的实施告警，可通过实时框架计算，或者接入到其他监控系统

暂不接入

### 日志标准实现

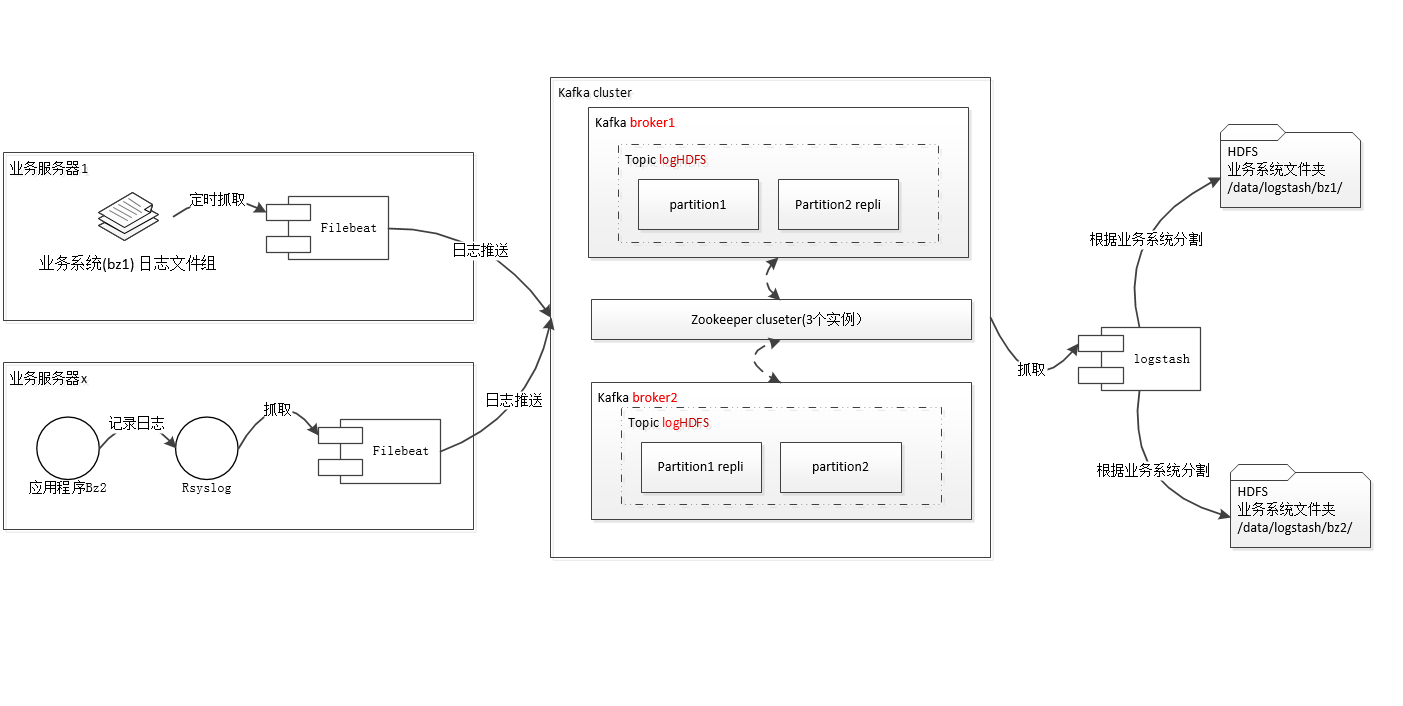
定义一套通用的日志接入标准，要求业务系统按照固定而相对灵活的方式接入，框架实现一套通用的日志分析系统，实现通用的查询、分析、归档等要求

标准实现，包括了用户分析、账号分析、认证方式分析等等各种分析要求（详见日志需求说明）

# 日志收集系统

实现简单的日志收集功能

## 系统架构图



## 系统安装

### 日志收集系统规划

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 主机名称 | IP | 用途 | 实例规格 | 镜像 | 部署组件 |
| log1 | 192.168.12.236 | 日志收集存储机器 | 2核2GB以上、100G硬盘以上 | CentOS-7.3 minimal | logstash、zookeeper |
| log2 | 192.168.12.237 | 日志通道机 | 2核2GB以上、40G硬盘以上 | CentOS-7.3 minimal | kafka、zookeeper |
| log3 | 192.168.12.238 | 日志通道机 | 2核2GB以上、40G硬盘以上 | CentOS-7.3 minimal | kafka、zookeeper |

### 组件及版本

|  |  |  |
| --- | --- | --- |
| 组件 | 中文说明 | 备注 |
| jdk-8u144-linux-x64.rpm | Jdk 1.8 | Jdk 1.8 |
| logstash-5.5.2.tar.gz | logstash | Logstash 从kafka中读取消息 |
| kafka\_2.12-0.11.0.0.tgz | Kafka | kafka |
| zookeeper-3.4.9.tar.gz | zookeeper | Zookeeper 管理kafka集群 |
|  |  |  |

### 组件下载

|  |  |
| --- | --- |
| Jdk 1.8 | <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> |
| logstash-5.5.2.rpm | <https://artifacts.elastic.co/downloads/logstash/logstash-5.5.2.rpm> |
| kafka\_2.11-0.11.0.0.tgz | http://www-eu.apache.org/dist/kafka/0.11.0.0/kafka\_2.12-0.11.0.0.tgz |
| zookeeper-3.4.9.tar.gz | <http://www-eu.apache.org/dist/zookeeper/zookeeper-3.4.9/zookeeper-3.4.9.tar.gz> |
| Scala2.11 |  |

### 准备工作（三台服务器分别执行）

#### 组件准备

yum install -y lrzsz vim wget

#### 下载文件

|  |
| --- |
| mkdir -p /data/pack  cd /data/pack  #上传各安装包到此文件夹下 |

#### 安装jvm

|  |
| --- |
| sudo yum localinstall jdk-8u144-linux-x64.rpm –y |

#### 打开组件所需防火墙端口

firewall-cmd --zone=public --add-port=12888/tcp --permanent

firewall-cmd --zone=public --add-port=13888/tcp --permanent

firewall-cmd --zone=public --add-port=12181/tcp --permanent

firewall-cmd --zone=public --add-port=19092/tcp --permanent

firewall-cmd --reload

### Zookeeper配置

|  |
| --- |
| cd /data/pack  tar -zxvf zookeeper-3.4.9.tar.gz  mv zookeeper-3.4.9 /usr/zookeeper  cd /usr/zookeeper/conf  mkdir -p /data/zookeeper/zkdata  mkdir -p /data/zookeeper/zkdatalog  cp zoo\_sample.cfg zoo.cfg  vim zoo.cfg  #修改配置如下  tickTime=2000  initLimit=10  syncLimit=5  dataDir=/data/zookeeper/zkdata  dataLogDir=/data/zookeeper/zkdatalog  clientPort=12181  server.1=192.168.255.137:12888:13888  server.2=192.168.255.138:12888:13888  server.3=192.168.255.139:12888:13888  #根据需要修改服务器ip |

#### 创建Zookeeper所需要的myid文件,三台服务器分别操作。

|  |
| --- |
| #server1  echo "1" > /data/zookeeper/zkdata/myid  #server2  echo "2" > /data/zookeeper/zkdata/myid  #server3  echo "3" > /data/zookeeper/zkdata/myid |

#### 设置Zookeeper开机启动

|  |
| --- |
| vim /usr/lib/systemd/system/zookeeper.service  #新增以下内容  [Unit]  Description=zookeeper.service  After=network.target  [Service]  Type=forking  Environment=/usr/zookeeper/  ExecStart=/usr/zookeeper/bin/zkServer.sh start  ExecStop=/usr/zookeeper/bin/zkServer.sh stop  ExecReload=/usr/zookeeper/bin/zkServer.sh restart  [Install]  WantedBy=multi-user.target  #保存退出  :wq  systemctl daemon-reload  systemctl enable zookeeper.service |

#### 启动服务

|  |
| --- |
| systemctl start zookeeper.service |

#### 检查服务状态

|  |
| --- |
| /usr/zookeeper/bin/zkServer.sh status |

其中有一台Mode显示leader另外两台显示follower则表示Zookeeper配置正确

### Kafka 安装（在log2和log3中安装）

#### Kafka 软件环境

1. log2和log3两台服务器
2. Jdk 1.8
3. 已经搭建好的zookeeper集群
4. 软件版本kafka\_2.12-0.11.0.0.tgz

|  |
| --- |
| #创建目录  cd /data/pack  mkdir -p /data/kafka/kafkalogs  #解压软件  tar -zxvf kafka\_2.12-0.11.0.0.tgz  mv kafka\_2.12-0.11.0.0 /usr/kafka |

#### Kafka 配置

日志存放时间预计：72 h

日志大小预计：4G

vim /usr/kafka/config/server.properties

编辑config/server.properties文件

|  |
| --- |
| broker.id=0 #每台服务器的broker.id都不能相同  log.dirs=/data/kafka/kafkalogs  listeners=PLAINTEXT://xxx.xxx.xxx.xxx(本机ip):19092  #填写当前服务器ip  advertised.listeners=PLAINTEXT://xxx.xxx.xxx.xxx(本机ip):19092  port=19092  num.recovery.threads.per.data.dir=2  offsets.topic.replication.factor=2  transaction.state.log.replication.factor=2  transaction.state.log.min.isr=2  quota.producer.default=10485760  quota.consumer.default=10485760  #启用删除策略  log.cleanup.policy=delete  log.retention.hours=72  log.retention.bytes=4294967296  message.max.byte=10485760  default.replication.factor=2  replica.fetch.max.bytes=10485760  #设置zookeeper的连接端口  zookeeper.connect=192.168.255.139:12181,192.168.255.140:12181,192.168.255.141:12181 |

#### Kafka开机启动配置

|  |
| --- |
| vim /usr/lib/systemd/system/kafka.service  #新增以下内容  [Unit]  Description=Apache Kafka server (broker)  Documentation=http://kafka.apache.org/documentation.html  After=network.target zookeeper.service  [Service]  Type=forking  Environment=/usr/kafka/  ExecStart=/usr/kafka/bin/kafka-server-start.sh -daemon /usr/kafka/config/server.properties  ExecStop=/usr/kafka/bin/kafka-server-stop.sh  [Install]  WantedBy=multi-user.target  #保存退出  :wq  systemctl daemon-reload  systemctl enable kafka.service |

#### Kafka 服务启动验证

|  |
| --- |
| systemctl start kafka.service |

查看服务是否启动

|  |
| --- |
| [root@dypan139 logs]# jps  2196 QuorumPeerMain  2600 Jps  2539 Kafka |

显示kafka则表示kafka已运行

#### 创建topic

/usr/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.12.236:12181 --replication-factor 2 --partitions 2 --topic logHDFS

查看是否创建成功

/usr/kafka/bin/kafka-topics.sh --list --zookeeper 192.168.12.236:12181

### Logstash 安装部署

#### 安装

部署在log1服务器中

|  |
| --- |
| cd /data/pack  yum localinstall logstash-5.5.2.rpm -y  systemctl daemon-reload  systemctl enable logstash.service  mkdir -p /data/logstash  chmod 777 -R /data/logstash  #新增kafka配置文件  vim /etc/logstash/conf.d/logCenter.conf  #新增以下内容，boostrap\_servers需要改成对应服务器地址  input {  kafka {  bootstrap\_servers => ["192.168.255.139:19092,192.168.255.143:19092"]  topics => "logCenter"  group\_id=> "default"  codec => "json"  }  }  filter {  mutate {  add\_field => {"logpath" => "%{[fields][system\_id]}"}  }  }  output {  stdout {}  file {  path => "/data/logstash/%{logpath}/%{+yyyy-MM-dd}.log"  }  } |

#### 性能优化及参数配置

vim /etc/logstash/logstash.yml

pipeline.workers: 24

pipeline.output.workers: 24

pipeline.batch.size: 10000

pipeline.batch.delay: 10

具体的workers/output.workers数量建议等于CPU数，batch.size/batch.delay根据实际的数据量逐渐增大来测试最优值。

#### 启动及验证

systemctl start logstash.service

ps -ef | grep logstash

## 系统监控

## 系统维护

### 常用命令

/usr/kafka/bin/kafka-run-class.sh kafka.tools.ConsumerOffsetChecker --broker-info --group default --topic logCenter --zookeeper 192.168.12.237:12181

/usr/kafka/bin/kafka-console-consumer.sh --zookeeper localhost:12181 --topic logCenter --from-beginning

## 安全

暂时不加

# 日志调用接入

日志的接入大致分2种情况，一种是高并发收集，一种是beats拉日志2种方式。

高并发收集应对的是日志录入非常频繁情况

beats方式针对的是旧系统改造，日志输出有限情况，对原有系统减小侵入性

高并发情况比较复杂，一般建议采用beats方式

## beats

### 总述

beats是指通过定期执行拉取操作收集日志的方式，官方就提供非常多的拉取方式，详见<https://www.elastic.co/cn/products/beats>

这里推荐FileBeat

### FileBeat centos7安装

下载

<https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-5.5.2-x86_64.rpm>

找到日志机器，上传至 /data/pack/

rpm -vi filebeat-5.5.2-x86\_64.rpm

### 配置使用

vim /etc/filebeat/filebeat.yml

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

filebeat.prospectors:

- input\_type: log

paths:

- /data/log/\*.log

encoding:GB2312

fields:

system\_id: test

multiline:

pattern: '^发'

negate: true

match: after

output.kafka:

hosts: ["192.168.12.237:19092", "192.168.12.238:19092"]

topic: 'logCenter'

key: 'test'

partition.hash:

hash: []

reachable\_only: true

required\_acks: 1

compression: gzip

max\_message\_bytes: 10000000

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

配置说明：

1. multiline 节点只有在需要匹配多行时才需要配置，配置规则略
2. 其中paths 为需要解析的日志路径
3. system\_id为系统名，比如jgt等等，无强制性要求
4. output.kafka.key 每个系统尽量不一致，做好采用与system\_id 一致

### 启动运行

/etc/init.d/filebeat restart

## 高并发日志收集

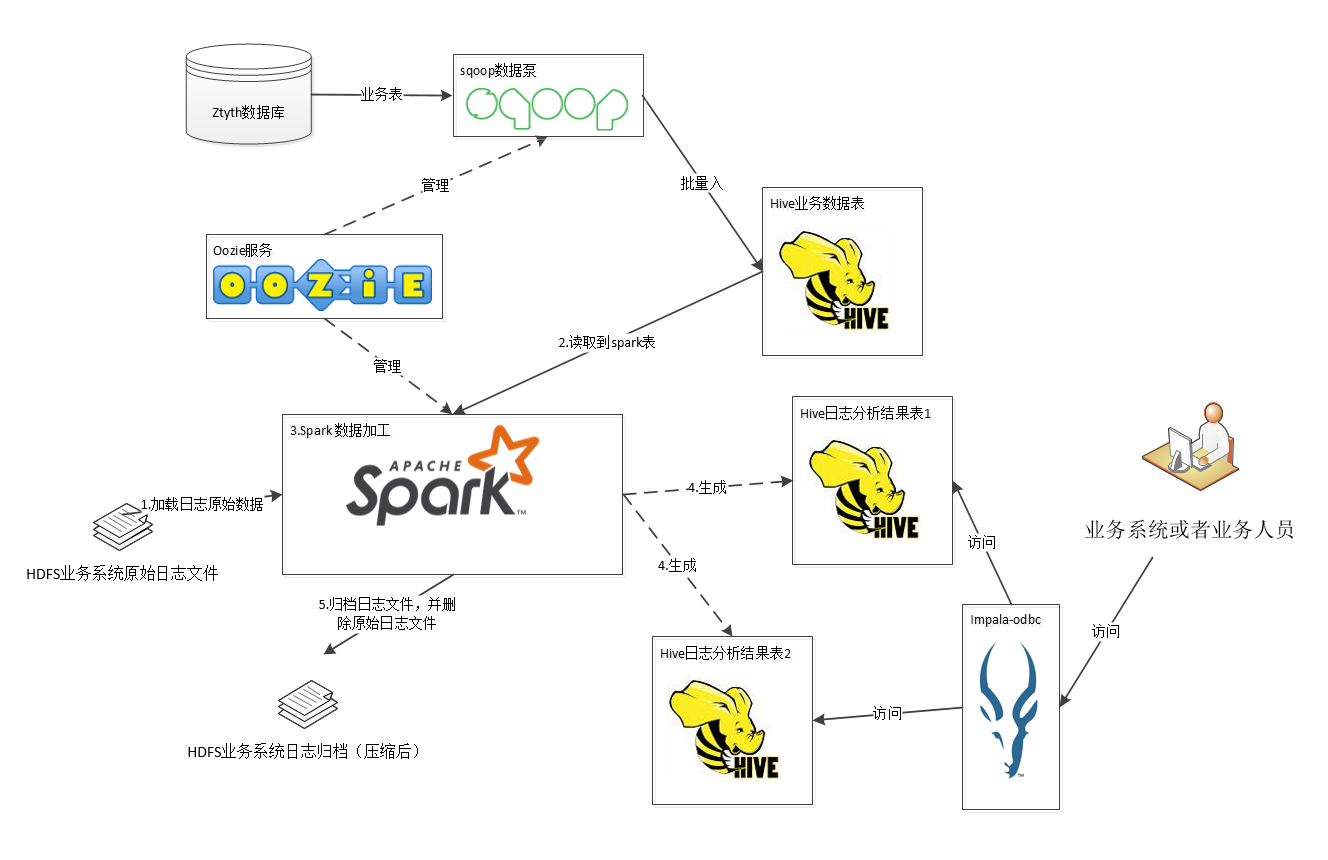
采用rsyslog + filebeat 作为数据收集端，后端一样，详细从略

# 日志具体分析

日志的分析各不相同，下面均为实例，其中一部分是可复用的

## 直通一体化操作及数据日志整合

### 逻辑处理架构图



ps：其中spark可替换为mapreduce等其他方式

### 业务处理流程图

待业务处理逻辑整理出来之后补上

### 总体设计考虑

#### 业务数据从业务库到hive

spark访问业务数据可以直接直连关系型数据库，考虑对业务系统数据库的影响，不使用这种方式。

从业务库数据到hive也存在很多方式，可以全表方式，可以增量方式

考虑对业务库改造小的原则，况且业务数据量也不是太大，通过全表导入方式。

使用sqoop快速迁移数据。

#### **hive保存最后的分析数据**

因为直通一体化查询数据的特殊性（主要是为了查询历史数据），数据体量很大，故采用hive这种数据仓库方式存储。如果是一些统计分析结果，会放到hbase或其他库中。

业务查询访问次数很小，采用impala方式快速查询结果。

hive中的存储采用ORC方式存储，主要考虑压缩比和执行效率

#### 业务处理完之后进行归档

考虑日志文件的存储非常占用硬盘资源，故数据处理之后即进行归档压缩，压缩方式暂定为parquet方式+ gzip压缩，压缩比较高，又利于以后spark和hive读取。

#### 处理引擎选择

按照正式环境配置，从开发效率和执行效率来说暂时spark是有优势的，选用spark作为数据处理的主要引擎。

### 详细设计

## janus日志分析

# 日志标准实现

## 日志消息头

业务系统标识、日志级别（INFO/WARN/ERROR）、服务器标识、记录时间、日志类别，安全证书号

## 日志消息体

具体日志信息内容。