

尚硅谷大数据项目之电商数仓（用户行为采集平台）

（尚硅谷研究院）

版本：V5.0

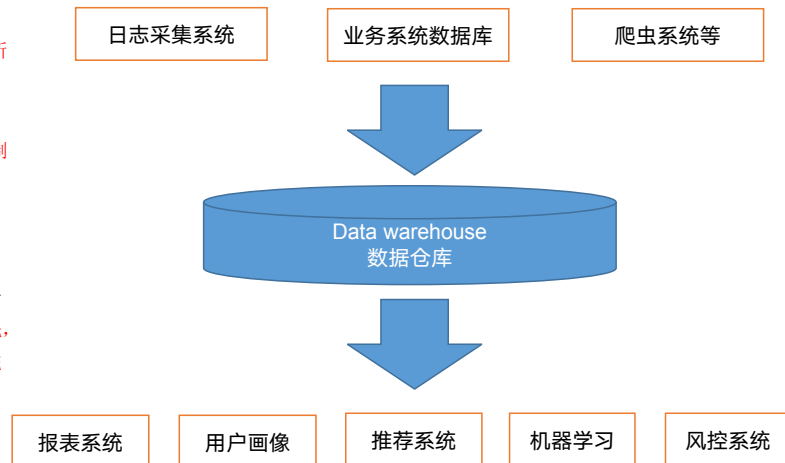
第 1 章 数据仓库概念

数据仓库概念

数据仓库（Data Warehouse），
是企业所有决策制定过程，提供所有系统数据支持的战略集合。

通过对数据仓库中数据的分析，
可以帮助企业，改进业务流程、控制成本、提高产品质量等。

数据仓库，并不是数据的最终目的地，而是为数据最终的目的地做好准备。这些准备包括对数据的：清洗，转义，分类，重组，合并，拆分，统计等等。



让天下没有难学的技术

第2章 项目需求及架构设计

2.1 项目需求分析



项目需求



一、项目需求

- 1、**用户行为**数据采集平台搭建
- 2、**业务**数据采集平台搭建
- 3、**数据仓库**维度建模
- 4、分析，**设备、会员、商品、地区、活动**等电商核心主题，统计的报表指标近100个。完全对比中型公司。
- 5、采用**即席查询**工具，随时进行指标分析
- 6、对**集群性能进行监控**，发生异常需要报警。
- 7、**元数据管理**
- 8、**质量监控**

二、思考题

- 1、项目**技术如何选型**？
- 2、框架版本如何选型（**Apache、CDH、HDP**）
- 3、服务器使用**物理机**还是**云主机**？
- 4、如何确认**集群规模**？（假设每台服务器8T硬盘）

让天下没有难学的技术

2.2 项目框架

2.2.1 技术选型



技术选型

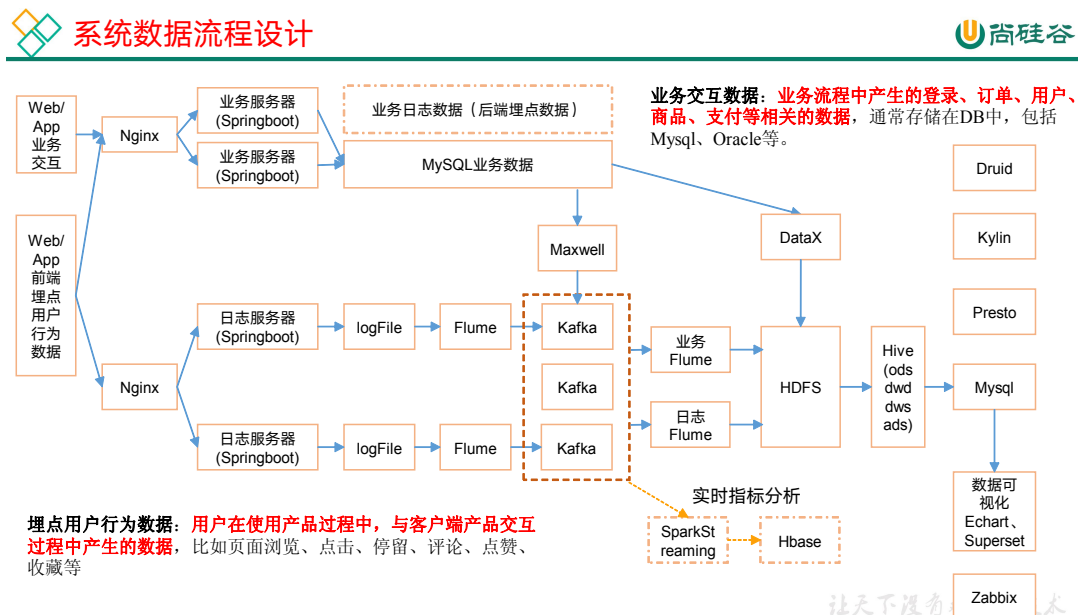


技术选型主要考虑因素：**数据量大小、业务需求、行业内经验、技术成熟度、开发维护成本、总成本预算**

- 数据采集传输：**Flume, DataX, Maxwell, Sqoop, Kafka**
- 数据存储：**MySQL, HDFS, HBase, Redis, MongoDB**
- 数据计算：**Hive, Tez, Spark, Flink, Storm**
- 数据查询：**Presto, Kylin, Impala, Druid**
- 数据可视化：**Superset、QuickBI、DataV**
- 任务调度：**DolphinScheduler、Azkaban、Oozie**
- 集群监控：**Zabbix**
- 元数据管理：**Atlas**

让天下没有难学的技术

2.2.2 系统数据流程设计



2.2.3 框架版本选型



1) 如何选择Apache/CDH/HDP版本？

(1) Apache: 运维麻烦，组件间兼容性需要自己调研。（一般大厂使用，技术实力雄厚，有专业的运维人员）（建议使用）

(2) CDH: 国内使用最多的版本，但CM不开源，今年开始要收费，一个节点1万美金。

(3) HDP: 开源，可以进行二次开发，但是没有CDH稳定，国内使用较少

(1) Apache框架版本

产品	版本
Java	1.8
Hadoop	3.1.3
Hive	3.1.2
Flume	1.9.0
Zookeeper	3.5.7
Kafka	2.4.1
DataX	3.0
Maxwell	1.29.2

注意事项：框架选型尽量不要选择最新的框架，选择最新框架半年前左右的稳定版。

让天下没有难学的技术

2.2.4 服务器选型

服务器选择物理机还是云主机？

1) 物理机：

- 以128G内存，20核物理CPU，40线程，8THDD和2TSSD硬盘，戴尔品牌单台报价4W出头。一般物理机寿命5年左右。
- 需要有专业的运维人员，平均一个月1万。电费也是不少的开销。

2) 云主机：

- 云主机：以阿里云为例，差不多相同配置，每年5W。
- 很多运维工作都由阿里云完成，运维相对较轻松

3) 企业选择

- 金融有钱公司和阿里没有直接冲突的公司选择阿里云
- 中小公司、为了融资上市，选择阿里云，拉倒融资后买物理机。
- 有长期打算，资金比较足，选择物理机。

让天下没有难学的技术

2.2.5 集群资源规划设计

集群规模

- 1) 如何确认集群规模？（假设：每台服务器8T磁盘，128G内存）
 - （1）每天日活跃用户100万，每人一天平均100条： $100万 * 100条 = 1亿条$
 - （2）每条日志1K左右，每天1亿条： $100000000 / 1024 / 1024 = 约100G$
 - （3）半年内不扩容服务器来算： $100G * 180天 = 约18T$
 - （4）保存3副本： $18T * 3 = 54T$
 - （5）预留20%~30%Buf= $54T / 0.7 = 77T$
 - （6）算到这： $约8T * 10台服务器$
- 2) 如果考虑数仓分层？数据采用压缩？需要重新再计算

让天下没有难学的技术

2) 测试集群服务器规划

生产环境集群服务器规划原则：

- 1 消耗内存的分开
- 2 数据传输比较紧密的放在一起 (kafka zookeeper)
- 3 客户端尽量放在同一台服务器上
- 4 有依赖关系的尽量放在同一台服务器上

Master	Master	worker	worker	worker
nn	nn	dn	dn	dn
rm	rm	nm	nm	nm
		JournalNode	JournalNode	JournalNode
		zookeeper	zookeeper	zookeeper
		kafka	kafka	kafka
		hive	hive	hive
		spark	spark	spark
		datax	datax	datax
	oozie			
	hue			
mysql				
supset				
		flume	flume	Flume(2)
		maxwell		

第3章 用户行为日志

3.1 用户行为日志概述

用户行为日志的内容，主要包括用户的各项**行为信息**以及行为所处的**环境信息**。收集这些信息的主要目的是优化产品和为各项分析统计指标提供数据支撑。收集这些信息的手段通常为**埋点**。

目前主流的埋点方式，有**代码埋点（前端/后端）、可视化埋点、全埋点**等。

代码埋点是通过调用埋点 SDK 函数，在需要埋点的业务逻辑功能位置调用接口，上报埋点数据。例如，我们对页面中的某个按钮埋点后，当这个按钮被点击时，可以在这个按钮对应的 OnClick 函数里面调用 SDK 提供的数据发送接口，来发送数据。

可视化埋点只需要研发人员集成采集 SDK，不需要写埋点代码，业务人员就可以通过访问分析平台的“圈选”功能，来“圈”出需要对用户行为进行捕捉的控件，并对该事件进行命名。圈选完毕后，这些配置会同步到各个用户的终端上，由采集 SDK 按照圈选的配置自动进行用户行为数据的采集和发送。

全埋点是通过在产品中嵌入 SDK，前端自动采集页面上的全部用户行为事件，上报埋点数据，相当于做了一个统一的埋点。然后再通过界面配置哪些数据需要在系统里面进行分析。

3.2 用户行为日志内容

本项目收集和分析的用户行为信息主要有**页面浏览记录、动作记录、曝光记录、启动记录和错误记录**。

3.1.1 页面浏览记录

页面浏览记录，记录的是访客对页面的浏览行为，该行为的环境信息主要有用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息及页面信息等。



页面浏览记录



用户信息	包括用户ID、设备ID
时间信息	用户跳入页面的时间
地理位置信息	用户浏览页面时所处的地理位置
设备信息	包括设备品牌、设备型号、设备系统
应用信息	指用户访问的应用信息，例如应用版本
渠道信息	指应用的下载渠道
页面信息	用户浏览的页面相关信息，包括页面ID，页面对象

让天下没有难学的技术

3.1.2 动作记录

动作记录，记录的是用户的业务操作行为，该行为的环境信息主要有用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息 及动作目标对象信息等。



动作记录

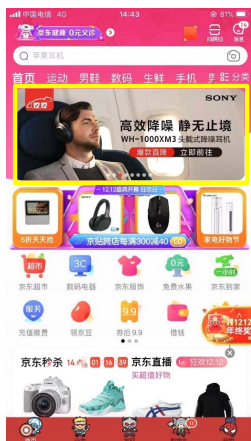


用户信息	包括用户ID、设备ID
时间信息	动作时间
地理位置信息	动作发生时所处的地理位置
设备信息	包括设备品牌、设备型号、设备系统
应用信息	指用户访问的应用信息，例如应用版本
渠道信息	指应用的下载渠道
动作目标信息	动作目标对象相关信息，包括对象类型，对象ID

让天下没有难学的技术

3.1.3 曝光记录

曝光记录，记录的是曝光行为，该行为的环境信息主要有用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息及曝光对象信息等。



用户信息	包括用户ID、设备ID
时间信息	曝光时间
地理位置信息	曝光行为发生时所处的地理位置
设备信息	包括设备品牌、设备型号、设备系统
应用信息	指用户访问的应用信息，例如应用版本
渠道信息	指应用的下载渠道
曝光对象信息	曝光对象相关信息，包括对象类型，对象ID

让天下没有难学的技术

3.1.4 启动记录

启动记录，记录的是用户启动应用的行为，该行为的环境信息主要有用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息、启动类型及开屏广告信息等。



用户信息	包括用户ID、设备ID
时间信息	启动时间
地理位置信息	启动时所处的地理位置
设备信息	包括设备品牌、设备型号、设备系统
应用信息	指用户访问的应用信息，例如应用版本
渠道信息	指应用的下载渠道
启动类型	包括图标和推送
开屏广告信息	包括广告ID等信息

让天下没有难学的技术

3.1.5 错误记录

启动记录，记录的是用户在使用应用过程中的报错行为，该行为的环境信息主要有用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息、以及可能与报错相关的页面信息、动作信息、曝光信息和动作信息。

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：[尚硅谷官网](#)

3.3 用户行为日志格式

我们的日志结构大致可分为两类，一是页面日志，二是启动日志。

3.3.1 页面日志

页面日志，以页面浏览为单位，即一个页面浏览记录，生成一条页面埋点日志。一条完整的页面日志包含，一个页面浏览记录，若干个用户在该页面所做的动作记录，若干个该页面的曝光记录，以及一个在该页面发生的报错记录。除上述行为信息，页面日志还包含了这些行为所处的各种环境信息，包括用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息等。

```
{
  "common": {                                -- 环境信息
    "ar": "230000",                          -- 地区编码
    "ba": "iPhone",                         -- 手机品牌
    "ch": "Appstore",                       -- 渠道
    "is_new": "1",                          -- 是否首日使用，首次使用的当日，该字段
    值为 1，过了 24:00，该字段置为 0。
    "md": "iPhone 8",                       -- 手机型号
    "mid": "YXfhjAYH6As2z9Iq",             -- 设备 id
    "os": "iOS 13.2.9",                     -- 操作系统
    "uid": "485",                           -- 会员 id
    "vc": "v2.1.134"                       -- app 版本号
  },
  "actions": [{                             -- 动作(事件)
    "action_id": "favor_add",               -- 动作 id
    "item": "3",                           -- 目标 id
    "item_type": "sku_id",                  -- 目标类型
    "ts": 1585744376605                     -- 动作时间戳
  }],
  "displays": [{                             -- 曝光
    "displayType": "query",                 -- 曝光类型
    "item": "3",                           -- 曝光对象 id
    "item_type": "sku_id",                  -- 曝光对象类型
    "order": 1,                             -- 出现顺序
    "pos_id": 2                             -- 曝光位置
  },
  {
    "displayType": "promotion",
    "item": "6",
    "item_type": "sku_id",
    "order": 2,
    "pos_id": 1
  },
  {
    "displayType": "promotion",
    "item": "9",
```

```

        "item_type": "sku_id",
        "order": 3,
        "pos_id": 3
    },
    {
        "displayType": "recommend",
        "item": "6",
        "item_type": "sku_id",
        "order": 4,
        "pos_id": 2
    },
    {
        "displayType": "query ",
        "item": "6",
        "item_type": "sku_id",
        "order": 5,
        "pos_id": 1
    }
],
"page": {
    "during_time": 7648,
    "item": "3",
    "item_type": "sku_id",
    "last_page_id": "login",
    "page_id": "good_detail",
    "sourceType": "promotion"
},
"err": {
    "error_code": "1234",
    "msg": "*****"
},
"ts": 1585744374423
}
-- 页面信息
-- 持续时间毫秒
-- 目标 id
-- 目标类型
-- 上页类型
-- 页面 ID
-- 来源类型
--错误
--错误码
--错误信息
--跳入时间戳

```

3.3.2 启动日志

启动日志以启动为单位，及一次启动行为，生成一条启动日志。一条完整的启动日志包括一个启动记录，一个本次启动时的报错记录，以及启动时所处的环境信息，包括用户信息、时间信息、地理位置信息、设备信息、应用信息、渠道信息等。

```

{
  "common": {
    "ar": "370000",
    "ba": "Honor",
    "ch": "wandoujia",
    "is_new": "1",
    "md": "Honor 20s",
    "mid": "eQF5boERMJFOujcp",
    "os": "Android 11.0",
    "uid": "76",
    "vc": "v2.1.134"
  },
  "start": {
    "entry": "icon",
    --icon 手机图标  notice 通知  install 安装

```

后启动

```
"loading_time": 18803,  --启动加载时间
"open_ad_id": 7,        --广告页 ID
"open_ad_ms": 3449,     -- 广告总共播放时间
"open_ad_skip_ms": 1989  -- 用户跳过广告时点
},
"err":{                  --错误
"error_code": "1234",    --错误码
"msg": "*****"         --错误信息
},
"ts": 1585744304000
}
```

3.3 模拟生成用户行为日志

3.3.1 环境准备

1) 服务器准备

CDH 已经安装完成。

3.3.2 模拟数据

1) 模拟器使用说明

（1）上传模拟器

将 application.yml、gmall2020-mock-log-2021-10-10.jar、path.json、logback.xml 上传到 hadoop103 的/opt/module/applog 目录下

① 创建 applog 路径

```
[root@hadoop103 ~]# mkdir /opt/module
[root@hadoop103 module]# mkdir /opt/module/applog
```

② 上传文件

（2）配置文件说明

① application.yml 文件

可以根据需求生成对应日期的用户行为日志。

```
[root@hadoop103 applog]# vim application.yml
```

修改如下内容

```
# 外部配置打开
logging.config: "../logback.xml"
#业务日期
mock.date: "2020-06-14"

#模拟数据发送模式
mock.type: "log"
#mock.type: "http"
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网

```
#mock.type: "kafka"
#http 模式下, 发送的地址
mock.url: "http://localhost:8090/applog"

mock:
  kafka-server: "hdp1:9092,hdp2:9092,hdp3:9092"
  kafka-topic: "ODS_BASE_LOG"

  #启动次数
mock.startup.count: 200
  #设备最大值
mock.max.mid: 1000000
  #会员最大值
mock.max.uid: 1000
  #商品最大值
mock.max.sku-id: 35
  #页面平均访问时间
mock.page.during-time-ms: 20000
  #错误概率 百分比
mock.error.rate: 3
  #每条日志发送延迟 ms
mock.log.sleep: 20
  #商品详情来源 用户查询, 商品推广, 智能推荐, 促销活动
mock.detail.source-type-rate: "40:25:15:20"

#领取购物券概率
mock.if_get_coupon_rate: 75

#购物券最大 id
mock.max.coupon-id: 3

#搜索关键词
mock.search.keyword: "图书,小米,iphone11,电视,口红,ps5,苹果手机,小米盒子"

# 男女浏览商品比重 (35sku)
mock.sku-weight.male:
"10:10:10:10:10:10:10:5:5:5:5:5:10:10:10:10:12:12:12:12:12:5:5:5:5:5:3:3:3:3:3:3:3:3:10:10"
mock.sku-weight.female:
"1:1:1:1:1:1:1:5:5:5:5:5:1:1:1:1:2:2:2:2:2:8:8:8:8:15:15:15:15:15:15:15:15:1:1:1"
```

② path.json

该文件用来配置访问路径, 可以根据需求, 灵活配置用户访问路径。

```
[
{
  "path": ["home", "good_list", "good_detail", "cart", "trade", "payment"],
  "rate": 20
},
{
  "path": ["home", "search", "good_list", "good_detail", "login", "good_detail", "cart", "trade", "payment"],
  "rate": 30
},
{
  "path": ["home", "search", "good_list", "good_detail", "login", "regis
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可百度访问: [尚硅谷官网](#)

```
ter","good_detail","cart","trade","payment"],"rate":20 },
{"path":["home","mine","orders_unpaid","trade","payment"],"rate":
10 },
{"path":["home","mine","orders_unpaid","good_detail","good_spec",
"comment","trade","payment"],"rate":5 },
{"path":["home","mine","orders_unpaid","good_detail","good_spec",
"comment","home"],"rate":5 },
{"path":["home","good_detail"],"rate":20 },
{"path":["home" ],"rate":10 }
]
```

③logback 配置文件

可配置日志生成路径，修改内容如下

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property name="LOG_HOME" value="/opt/module/applog/log" />
  <appender name="console"
class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%msg%n</pattern>
    </encoder>
  </appender>

  <appender name="rollingFile"
class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
<fileNamePattern>${LOG_HOME}/app.%d{yyyy-MM-dd}.log</fileNamePattern>
    </rollingPolicy>
    <encoder>
      <pattern>%msg%n</pattern>
    </encoder>
  </appender>
  <!-- 将某一个包下日志单独打印日志 -->
  <logger name="com.atguigu.gmall2020.mock.log.util.LogUtil"
    level="INFO" additivity="false">
    <appender-ref ref="rollingFile" />
    <appender-ref ref="console" />
  </logger>

  <root level="error" >
    <appender-ref ref="console" />
    <!-- <appender-ref ref="async-rollingFile" /> -->
  </root>
</configuration>
```

2) 生成日志

(1) 进入到/opt/module/applog 路径，执行以下命令

```
[root@hadoop103 applog]# java -jar
gmall2020-mock-log-2021-10-10.jar
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网

(2) 在/opt/module/applog/log 目录下查看生成日志

```
[root@hadoop103 log]# ll
```

3) 集群日志生成脚本

在 hadoop103 的/root 目录下创建 bin 目录，这样脚本可以在服务器的任何目录执行。

```
[root@hadoop103 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.8.0_181-cloudera/bin:/usr/java/jdk1.8.0_181-cloudera/bin:/usr/java/jdk1.8.0_181-cloudera/bin:/root/bin
```

(1) 在/root/bin 目录下创建脚本 lg.sh

```
[root@hadoop103 ~]# mkdir bin
[root@hadoop103 ~]# cd bin/
[root@hadoop103 bin]# vim lg.sh
```

(2) 在脚本中编写如下内容

```
#!/bin/bash
for i in hadoop103 hadoop104; do
    echo "===== $i ====="
    ssh $i "cd /opt/module/applog/; java -jar gmall2020-mock-log-2021-10-10.jar >/dev/null 2>&1 &"
done
```

注：

①/opt/module/applog/为 jar 包及配置文件所在路径

②/dev/null 代表 linux 的空设备文件，所有往这个文件里面写入的内容都会丢失，俗称“黑洞”。

标准输入 0：从键盘获得输入 /proc/self/fd/0

标准输出 1：输出到屏幕（即控制台） /proc/self/fd/1

错误输出 2：输出到屏幕（即控制台） /proc/self/fd/2

(3) 修改脚本执行权限

```
[root@hadoop103 bin]# chmod u+x lg.sh
```

(4) 将 jar 包及配置文件上传至 **hadoop104** 的/opt/module/applog/路径

```
[root@hadoop103 applog]# scp -r /opt/module/ hadoop104:/opt/
```

(5) 启动脚本

```
[root@hadoop103 module]# lg.sh
```

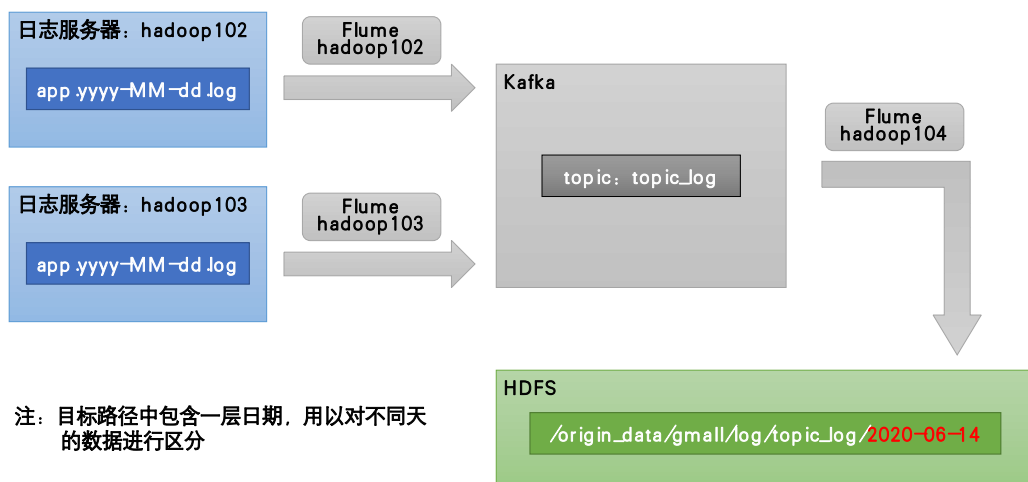
(6) 分别在 hadoop103、hadoop104 的/opt/module/applog/log 目录上查看生成的数据

```
[root@hadoop103 logs]# ls
app.2020-06-14.log
[root@hadoop104 logs]# ls
app.2020-06-14.log
```

第4章 数据采集模块

4.1 数据通道

用户行为日志数据通道



让天下没有难学的技术

4.2 日志采集 Flume

4.2.1 日志采集 Flume 配置概述

按照规划，需要采集的用户行为日志文件分布在 hadoop103，hadoop104 两台日志服务器，故需要在 hadoop103，hadoop104 两台节点配置日志采集 Flume。日志采集 Flume 需要采集日志文件内容，并对日志格式（JSON）进行校验，然后将校验通过的日志发送到 Kafka。

此处可选择 TailDirSource 和 KafkaChannel，并配置日志校验拦截器。

选择 TailDirSource 和 KafkaChannel 的原因如下：

1) TailDirSource

TailDirSource 相比 ExecSource、SpoolingDirectorySource 的优势

TailDirSource：断点续传、多目录。Flume1.6 以前需要自己自定义 Source 记录每次读取文件位置，实现断点续传。

ExecSource 可以实时搜集数据，但是在 Flume 不运行或者 Shell 命令出错的情况下，数据将会丢失。

SpoolingDirectorySource 监控目录，支持断点续传。

2) KafkaChannel

采用 Kafka Channel，省去了 Sink，提高了效率。

日志采集 Flume 关键配置如下：

日志采集Flume关键配置

TailDirSource

#通过正则表达式匹配需要采集的日志文件

a1.sources.r1.filegroups=f1

a1.sources.r1.filegroups.f1=/opt/module/applog/log/app.*

#配置ETL拦截器，对非法数据进行过滤

interceptors=i1

interceptors.i1.type=ETLInterceptor.Builder

KafkaChannel

#指定数据要发往的目标Topic

a1.channels.c1.kafka.topic=topic_log

让天下没有难学的技术

日志采集 Flume，数据被发送到了 Kafka，该 Flume 相当于一个 Kafka 生产者。所以需要进行上述 Kafka 客户端的安全认证。但是此处不需要我们进行手动配置，在开启 Kerberos 后，CM 会自动进行配置。

4.2.2 日志采集 Flume 配置实操

1) 集群规划

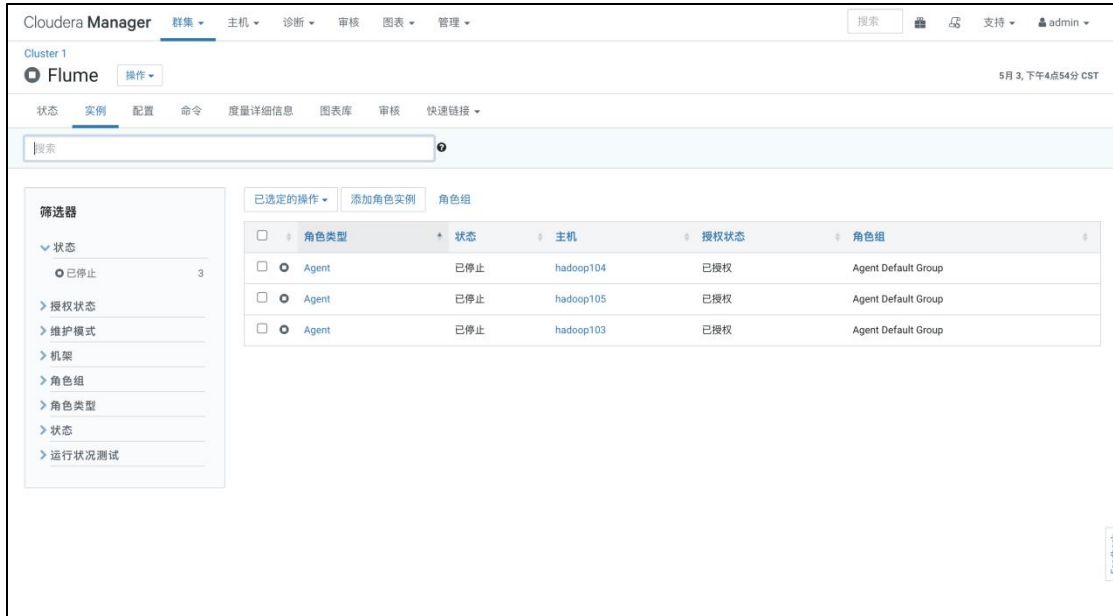
	服务器 hadoop103	服务器 hadoop104	服务器 hadoop105
Flume（日志采集）	Flume	Flume	

2) Flume 具体配置



更多 Java - 大数据 - 前端 - python 人工智能资料下载，可百度访问：尚硅谷官网

(1) 点击实例



Cloudera Manager 群集 主机 诊断 审核 图表 管理 搜索 支持 admin

Cluster 1

Flume 操作 5月3, 下午4点54分 CST

状态 实例 配置 命令 度量详细信息 图表库 审核 快速链接

搜索

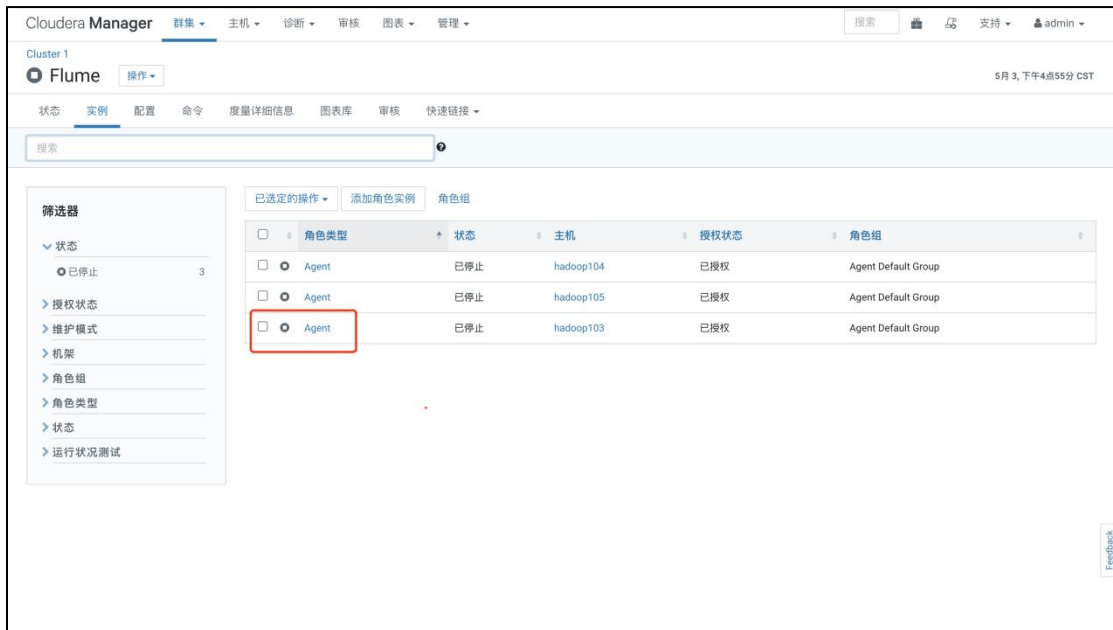
筛选器

- 状态
 - 已停止 3
- 授权状态
- 维护模式
- 机架
- 角色组
- 角色类型
- 状态
- 运行状况测试

已选定的操作 添加角色实例 角色组

角色类型	状态	主机	授权状态	角色组
Agent	已停止	hadoop104	已授权	Agent Default Group
Agent	已停止	hadoop105	已授权	Agent Default Group
Agent	已停止	hadoop103	已授权	Agent Default Group

(2) 点击 hadoop103 的 Agent 选项



Cloudera Manager 群集 主机 诊断 审核 图表 管理 搜索 支持 admin

Cluster 1

Flume 操作 5月3, 下午4点55分 CST

状态 实例 配置 命令 度量详细信息 图表库 审核 快速链接

搜索

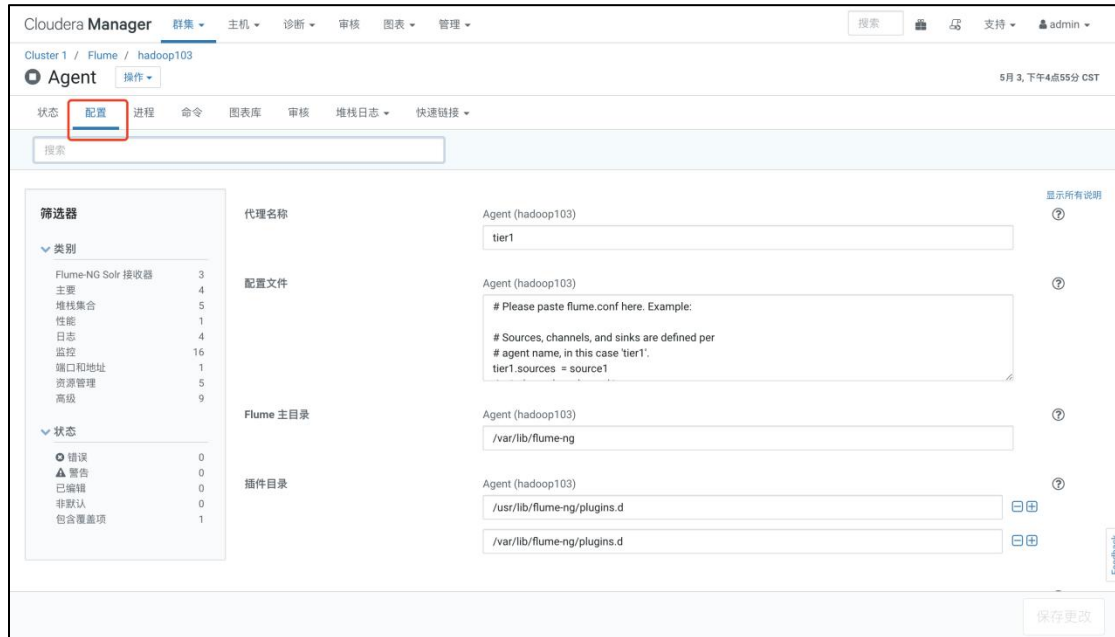
筛选器

- 状态
 - 已停止 3
- 授权状态
- 维护模式
- 机架
- 角色组
- 角色类型
- 状态
- 运行状况测试

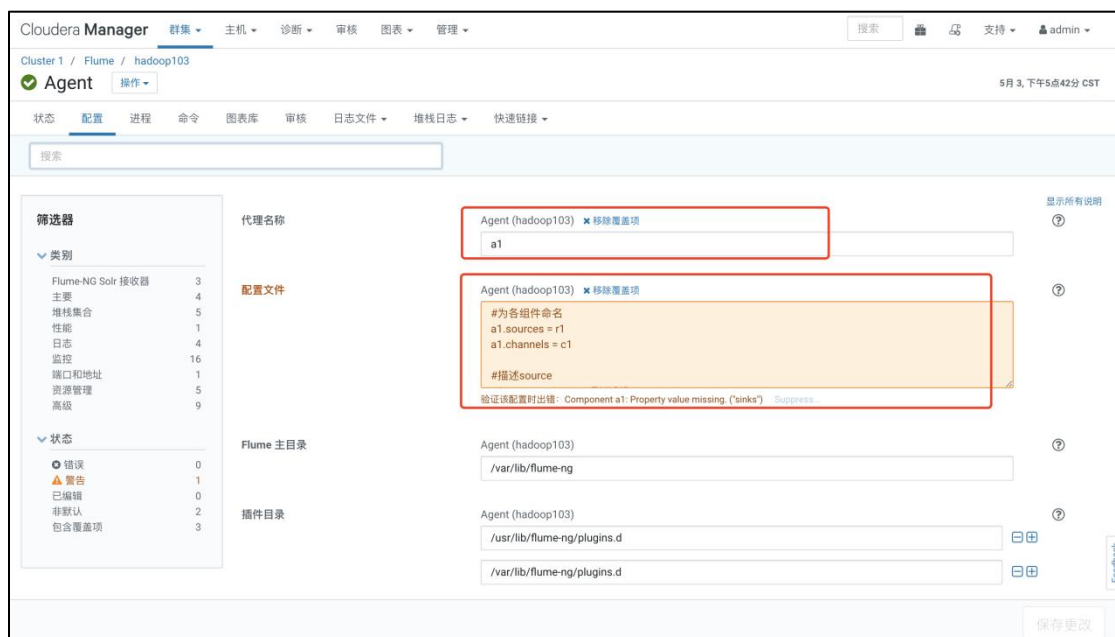
已选定的操作 添加角色实例 角色组

角色类型	状态	主机	授权状态	角色组
Agent	已停止	hadoop104	已授权	Agent Default Group
Agent	已停止	hadoop105	已授权	Agent Default Group
Agent	已停止	hadoop103	已授权	Agent Default Group

(3) 点击配置



(4) 对 Flume Agent 进行具体配置



配置文件内容如下

```
#为各组件命名
a1.sources = r1
a1.channels = c1

#描述 source
a1.sources.r1.type = TAILDIR
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /opt/module/applog/log/app.*
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type =
com.atguigu.flume.interceptor.ETLInterceptor$Builder
```

```
#描述 channel
a1.channels.c1.type = org.apache.flume.channel.kafka.KafkaChannel
a1.channels.c1.kafka.bootstrap.servers =
hadoop103:9092,hadoop104:9092
a1.channels.c1.kafka.topic = topic_log
a1.channels.c1.parseAsFlumeEvent = false

#绑定 source 和 channel 以及 sink 和 channel 的关系
a1.sources.r1.channels = c1
```

(5) 在 hadoop104 上重复相同的操作

3) 编写 Flume 拦截器

(1) 创建 Maven 工程 flume-interceptor

(2) 创建包: com.atguigu.flume.interceptor

(3) 在 pom.xml 文件中添加如下配置

```
<dependencies>
  <dependency>
    <groupId>org.apache.flume</groupId>
    <artifactId>flume-ng-core</artifactId>
    <version>1.9.0</version>
    <scope>provided</scope>
  </dependency>

  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.62</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.3.2</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <configuration>
        <descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
      <executions>
        <execution>
          <id>make-assembly</id>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

```
        </goals>
    </execution>
</executions>
</plugin>
</plugins>
</build>
```

(4) 在 `com.atguigu.flume.interceptor` 包下创建 `JSONUtils` 类

```
package com.atguigu.flume.interceptor;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONException;

public class JSONUtils {
    public static boolean isJSONValidate(String log){
        try {
            JSON.parse(log);
            return true;
        } catch (JSONException e) {
            return false;
        }
    }
}
```

(5) 在 `com.atguigu.flume.interceptor` 包下创建 `ETLInterceptor` 类

```
package com.atguigu.flume.interceptor;

import com.alibaba.fastjson.JSON;
import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.Iterator;
import java.util.List;

public class ETLInterceptor implements Interceptor {

    @Override
    public void initialize() {
    }

    @Override
    public Event intercept(Event event) {

        byte[] body = event.getBody();
        String log = new String(body, StandardCharsets.UTF_8);

        if (JSONUtils.isJSONValidate(log)) {
            return event;
        } else {
            return null;
        }
    }

    @Override
    public List<Event> intercept(List<Event> list) {
```

```

        Iterator<Event> iterator = list.iterator();

        while (iterator.hasNext()){
            Event next = iterator.next();
            if(intercept(next)==null){
                iterator.remove();
            }
        }

        return list;
    }

    public static class Builder implements Interceptor.Builder{

        @Override
        public Interceptor build() {
            return new ETLInterceptor();
        }

        @Override
        public void configure(Context context) {

        }

    }

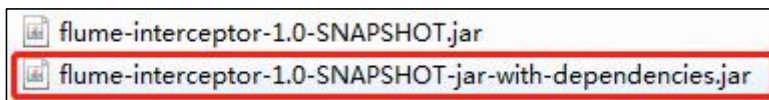
    @Override
    public void close() {

    }

}

```

(6) 打包



(7) 采用 root 用户将 flume-interceptor-1.0-SNAPSHOT.jar 包放入到 hadoop103 的 /opt/cloudera/parcels/CDH/lib/flume-ng/lib 文件夹下面。

(8) 分发 Flume 到 hadoop104

```

[root@hadoop103 lib]$ scp flume-interceptor-1.0-SNAPSHOT-jar-with-dependencies.jar hadoop105:/opt/cloudera/parcels/CDH/lib/flume-ng/lib

```

(9) 重启 flume，查看日志

```

[root@hadoop103 lib]$ tail -f /var/log/flume-ng/flume-cmf-flume-AGENT-hadoop103.log

```

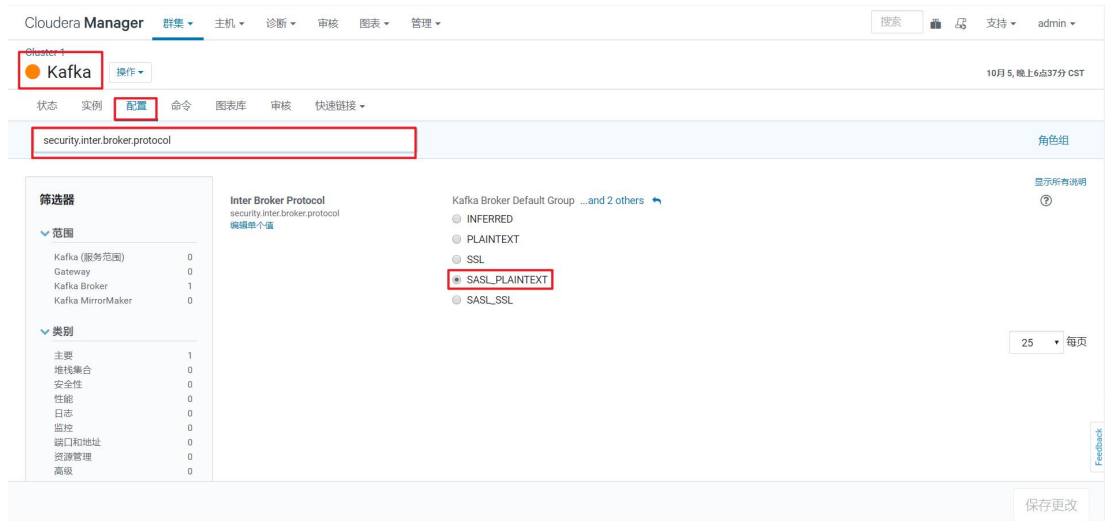
4.2.3 日志采集 Flume 测试

1) 启动 Flume

2) 进行 kafka\kerberos 配置

(1) 修改 Kafka 配置

① 在 Kafka 的配置项搜索 “security.inter.broker.protocol”，设置为 SASL_PLAINTEXT。



(2) 创建 jaas.conf 文件

```
[root@hadoop103 hive]# vim /var/lib/hive/jaas.conf
```

文件内容如下

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useTicketCache=true;
};
```

(3) 创建 consumer.properties 文件

```
[root@hadoop103 conf]# vim /etc/kafka/conf/consumer.properties
```

文件内容如下

```
security.protocol=SASL_PLAINTEXT
sasl.kerberos.service.name=kafka
```

(4) 声明 jass.conf 文件路径

```
[root@hadoop103 conf]# export KAFKA_OPTS="-Djava.security.auth.login.config=/var/lib/hive/jaas.conf"
```

(6) 进行 kerberos 认证

```
[root@hadoop103 ~]# kinit kafka/kafka
```

3) 启动一个 Kafka 的 Console-Consumer

```
[root@hadoop103 ~]# /opt/cloudera/parcels/CDH/bin/kafka-console-consumer
--bootstrap-server hadoop103:9092 --topic topic_log
--consumer.config /etc/kafka/conf/consumer.properties
```

4) 生成模拟数据

```
[root@hadoop103 ~]# lg.sh
```

5) 观察 Kafka 消费者是否能消费到数据

4.3 日志消费 Flume

4.3.1 日志消费 Flume 配置概述

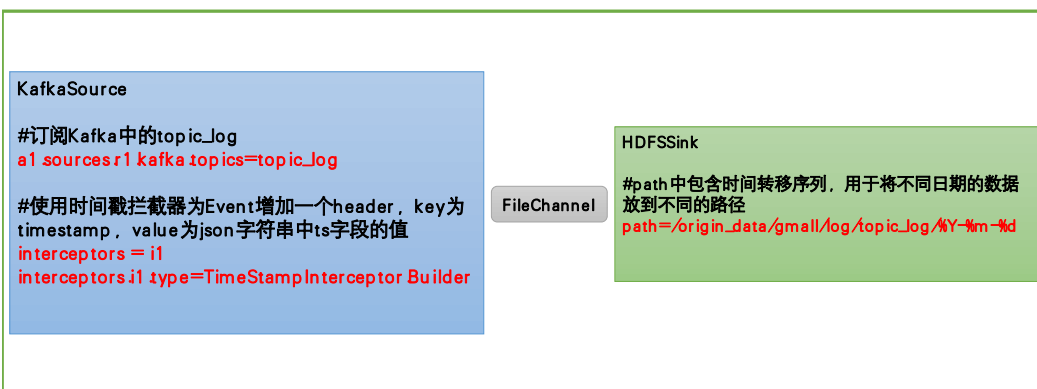
按照规划，该 Flume 需将 Kafka 中 topic_log 的数据发往 HDFS。并且对每天产生的用户行为日志进行区分，将不同天的数据发往 HDFS 不同天的路径。并且消费 Kafka Flume，将数据从 Kafka 传输到 HDFS，该 Flume 相当于一个 Kafka 消费者。所以也需要我们进行上述 Kafka 客户端的安全认证（无需手动认证）。除此之外，我们还需要进行 HDFS 客户端的安全认证，这需要我们手动配置。

此处选择 KafkaSource、FileChannel、HDFSSink。

关键配置如下：



日志消费Flume关键配置



让天下没有难学的技术

（1）生成 hive 用户的 keytab 文件

用户认证的方式有“输入密码”和“使用 keytab 密钥文件”两种方式，此处需使用 keytab 密钥文件进行认证。

```
[root@hadoop101 hive]# kadmin.local -q "xst -norandkey -k /var/lib/hive/hive.keytab
hive/hive@HADOOP.COM"
```

（2）增加读权限

```
chmod +r /var/lib/hive/hive.keytab
```

（3）分发 keytab 文件

```
xsync /var/lib/hive/hive.keytab
```

4.3.2 日志消费 Flume 配置实操

	服务器 hadoop103	服务器 hadoop104	服务器 hadoop105
--	---------------	---------------	---------------

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网

Flume（日志采集）			Flume
-------------	--	--	-------

1) 创建 Flume 配置文件

2) 创建一个文件夹

```
[root@hadoop105 opt]#mkdir /opt/flume
[root@hadoop105 opt]#sudo chown flume:flume /opt/flume/
```

在 CM 管理页面 **hadoop105** 上 Flume 的配置中找到代理名称

a1

3) 配置文件内容如下

```
## 组件
a1.sources=r1
a1.channels=c1
a1.sinks=k1

## source1
a1.sources.r1.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.r1.batchSize = 5000
a1.sources.r1.batchDurationMillis = 2000
a1.sources.r1.kafka.bootstrap.servers =
hadoop103:9092,hadoop104:9092,hadoop105:9092
a1.sources.r1.kafka.topics=topic_log
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type =
com.atguigu.flume.interceptor.TimestampInterceptor$Builder

## channel1
a1.channels.c1.type = file
a1.channels.c1.checkpointDir = /opt/flume/checkpoint/behavior1
a1.channels.c1.dataDirs = /opt/flume/data/behavior1/
a1.channels.c1.maxFileSize = 2146435071
a1.channels.c1.capacity = 1000000
a1.channels.c1.keep-alive = 6

## sink1
a1.sinks.k1.type = hdfs
#a1.sinks.k1.hdfs.proxyUser=hive
a1.sinks.k1.hdfs.kerberosPrincipal=hive/hive@HADOOP.COM
a1.sinks.k1.hdfs.kerberosKeytab=/var/lib/hive/hive.keytab

a1.sinks.k1.hdfs.path = /origin_data/gmall/log/topic_log/%Y-%m-%d
a1.sinks.k1.hdfs.filePrefix = log-
a1.sinks.k1.hdfs.round = false

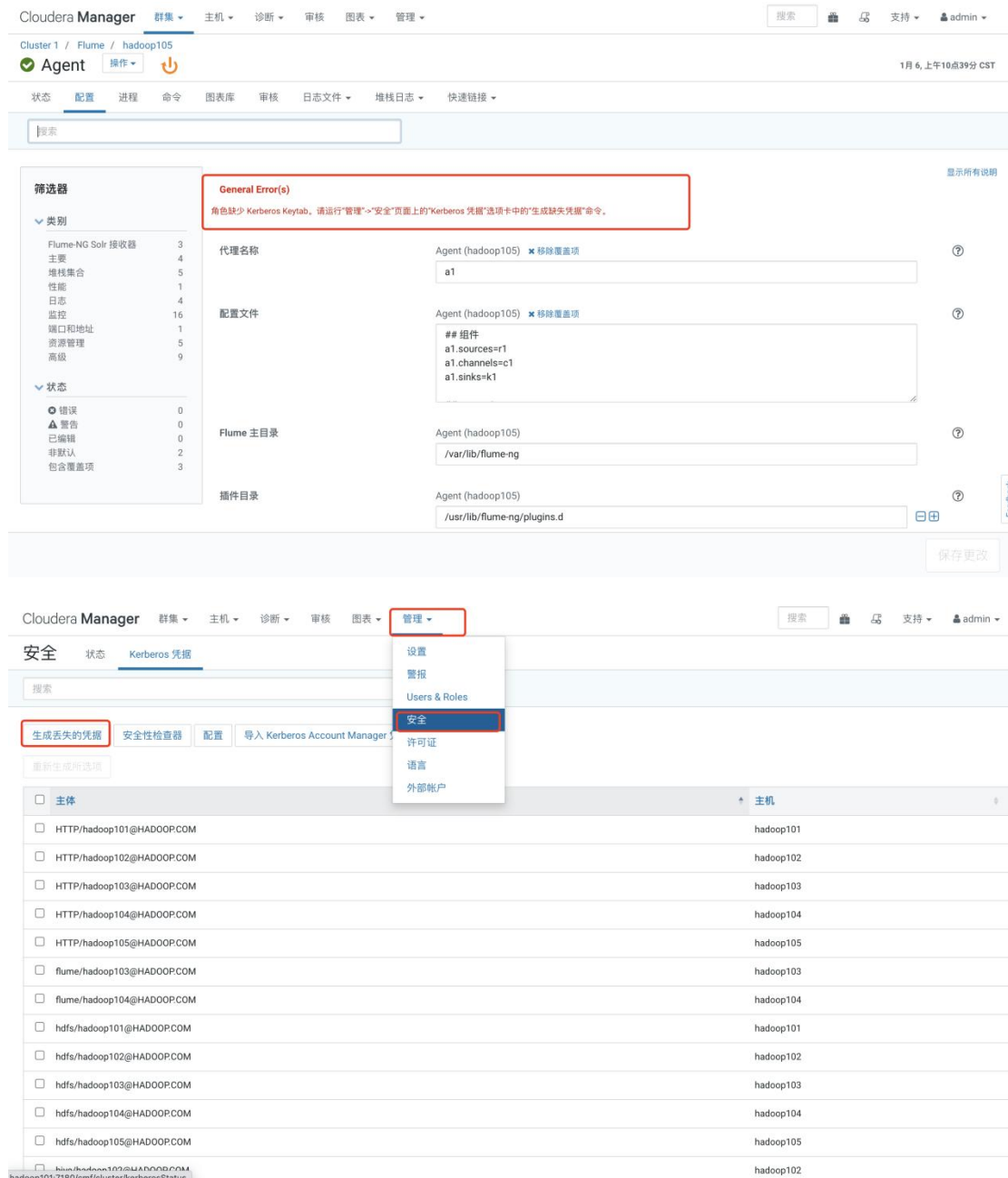
a1.sinks.k1.hdfs.rollInterval = 10
a1.sinks.k1.hdfs.rollSize = 134217728
a1.sinks.k1.hdfs.rollCount = 0

## 控制输出文件是原生文件。
a1.sinks.k1.hdfs.fileType = CompressedStream
a1.sinks.k1.hdfs.codec = gzip
```


拼装

```
a1.sources.r1.channels = c1
a1.sinks.k1.channel= c1
```

注：配置优化



The screenshot shows the Cloudera Manager interface. The top navigation bar includes 'Cluster 1 / Flume / hadoop105' and a search bar. The left sidebar has tabs for 'Agent', '配置', '进程', '命令', '图表库', '审核', '日志文件', '堆栈日志', and '快速链接'. The main content area shows the '配置' (Configuration) page for the 'Agent (hadoop105)'. A red box highlights a 'General Error(s)' message: '角色缺少 Kerberos Keytab, 请运行“管理”->“安全”页面上的“Kerberos 凭据”选项卡中的“生成缺失凭据”命令。' (The role is missing the Kerberos Keytab, please run the "Generate Missing Credentials" command in the "Kerberos Credentials" tab on the "Management" -> "Security" page.). Below the error message, there are fields for '代理名称' (Agent Name), '配置文件' (Configuration File), 'Flume 主目录' (Flume Home), and '插件目录' (Plugin Directory). The '配置文件' field contains the following content:

```
## 组件
a1.sources=r1
a1.channels=c1
a1.sinks=k1
```

The bottom part of the screenshot shows the '安全' (Security) page, specifically the 'Kerberos 凭据' (Kerberos Credentials) tab. A red box highlights the '生成丢失的凭据' (Generate Missing Credentials) button. A dropdown menu is open, showing options: '设置' (Settings), '警报' (Alerts), 'Users & Roles', '安全' (Security), '许可证' (Licenses), '语言' (Languages), and '外部帐户' (External Accounts). The '安全' option is highlighted. Below the dropdown, there is a table with columns '主体' (Principal) and '主机' (Host). The table lists various principals and their corresponding hosts, such as 'HTTP/hadoop101@HADOOP.COM' on 'hadoop101' and 'hdfs/hadoop101@HADOOP.COM' on 'hadoop101'.



1) FileChannel 优化

通过配置 **dataDirs** 指向多个路径，每个路径对应不同的硬盘，增大 Flume 吞吐量。

官方说明如下：

Comma separated list of directories for storing log files. Using multiple directories on separate disks can improve file channel performance

checkpointDir 和 **backupCheckpointDir** 也尽量配置在不同硬盘对应的目录中，保证 checkpoint 坏掉后，可以快速使用 **backupCheckpointDir** 恢复数据

2) HDFS Sink 优化

(1) HDFS 存入大量小文件，有什么影响？

元数据层面：每个小文件都有一份元数据，其中包括文件路径，文件名，所有者，所属组，权限，创建时间等，这些信息都保存在 Namenode 内存中。所以小文件过多，会占用 Namenode 服务器大量内存，影响 Namenode 性能和使用寿命

计算层面：默认情况下 MR 会对每个小文件启用一个 Map 任务计算，非常影响计算性能。同时也影响磁盘寻址时间。

(2) HDFS 小文件处理

官方默认的这三个参数配置写入 HDFS 后会产生小文件，**hdfs.rollInterval**、**hdfs.rollSize**、**hdfs.rollCount**

基于以上 **hdfs.rollInterval=3600**，**hdfs.rollSize=134217728**，**hdfs.rollCount=0** 几个参数综合作用，效果如下：

(1) 文件在达到 128M 时会滚动生成新文件

(2) 文件创建超 3600 秒时会滚动生成新文件

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：[尚硅谷官网](#)

3) 编写 Flume 拦截器

(1) 在 com.atguigu.flume.interceptor 包下创建 TimestampInterceptor 类

```
package com.atguigu.interceptor;

import com.alibaba.fastjson.JSONObject;
import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class TimestampInterceptor implements Interceptor {

    private ArrayList<Event> events = new ArrayList<>();

    @Override
    public void initialize() {

    }

    @Override
    public Event intercept(Event event) {

        Map<String, String> headers = event.getHeaders();
        String log = new String(event.getBody(),
StandardCharsets.UTF_8);

        JSONObject jsonObject = JSONObject.parseObject(log);

        String ts = jsonObject.getString("ts");
        headers.put("timestamp", ts);

        return event;
    }

    @Override
    public List<Event> intercept(List<Event> list) {
        events.clear();
        for (Event event : list) {
            events.add(intercept(event));
        }

        return events;
    }

    @Override
    public void close() {

    }

    public static class Builder implements Interceptor.Builder {
        @Override
        public Interceptor build() {
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：[尚硅谷官网](#)

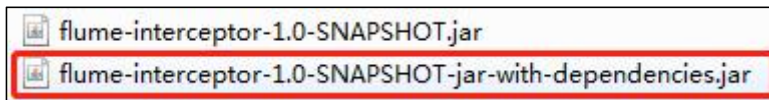
```

        return new TimeStampInterceptor();
    }

    @Override
    public void configure(Context context) {
    }
}
}

```

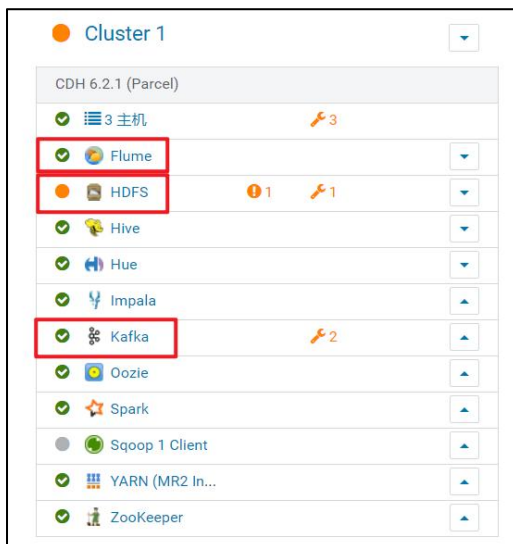
(2) 重新打包



(3) 需要先将打好的包放入到 hadoop105 的/opt/cloudera/parcels/CDH/lib/flume-ng/lib 文件夹下面。

4.2.4 模拟生成日志

1) 确保 Flume、Kafka 等服务正常运行



2) Kerberos 认证

```

[root@hadoop101 ~]# kadmin.local -q "addprinc hdfs/hdfs"
[root@hadoop101 ~]# kinit hdfs/hdfs

```

2) 在 HDFS 创建/origin_data 路径，并修改所有者为 hive

```

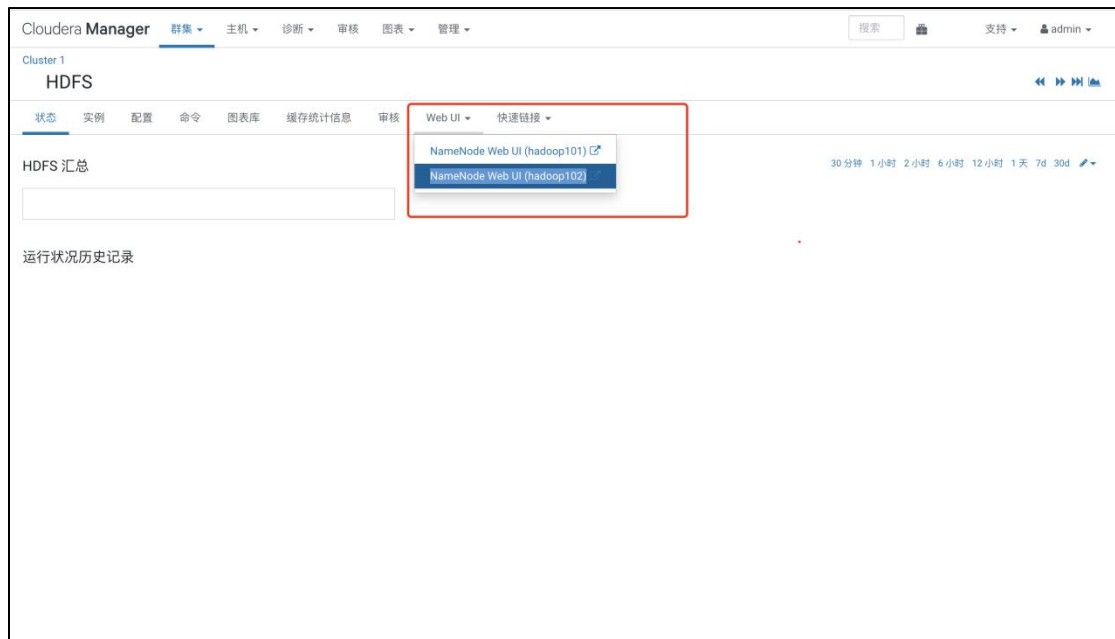
[root@hadoop101 ~]# hadoop fs -mkdir /origin_data
[root@hadoop101 ~]# hadoop fs -chown hive:hive /origin_data

```

3) 调用日志生成脚本

```
lg.sh
```

4) 观察 hdfs 上是否有数据



HadoopOverviewDatanodesDatanode Volume FailuresSnapshotStartup ProgressUtilities

Browse Directory

/origin_data/gmall/log/topic_log/2020-06-14

Go!

Show25entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	flume	supergroup	46.82 KB	May 03 17:40	1	128 MB	log-1651570617335.gz
<input type="checkbox"/>	-rw-r--r--	flume	supergroup	42.85 KB	May 03 17:40	1	128 MB	log-1651570631465.gz
<input type="checkbox"/>	-rw-r--r--	flume	supergroup	38.41 KB	May 03 17:40	1	128 MB	log-1651570641768.gz

Showing 1 to 3 of 3 entries

Previous

1

Next

Hadoop, 2018.

4) 查看 flume 日志

更多 Java -大数据 -前端 -python 人工智能资料下载，可百度访问：尚硅谷官网