基于Kafka-Spark Streaming的数据处理系统及测试

- 搜狗搜索测试部
- 甄丽霞
- 2016/11/1

- **技术知识**：初步学习和了解待测系统涉及到的相关技术知识，了解系统架构。
- **需求分析**：分析掌握原始数据以及待测系统的数据结构、数据特点、数据类型；分析针对不同数据的不同运营架构。
- **技术细节**：掌握系统架构中每个环节的技术细节和具体实现。
- **测试相关**：正确性测试：确认系统架构各环节中，研发根据具体业务需求完成了哪些功能的编码实现。架构测试：从系统架构特点考虑架构自身变化对业务代码的影响。

搜狗搜索

Flume

丰富的数据采集：分布式、可靠、和高可用的海量日志采集、聚合和传输的系统。

Kafka

数据总线：分布式的，高吞吐量、信息分片存储的消息订阅和发布系统。

Spark streaming

流式计算：可扩展、高吞吐、容错的流计算引擎。

数据分析、运营架构

普通全量数据内容：questionId、title、created、 authorId、authorName、detail、知乎站内url、userPictureUrl 等等。

运营流程

- ✓ Fast
- ✓ Instant
- ✓ freq

数据种类

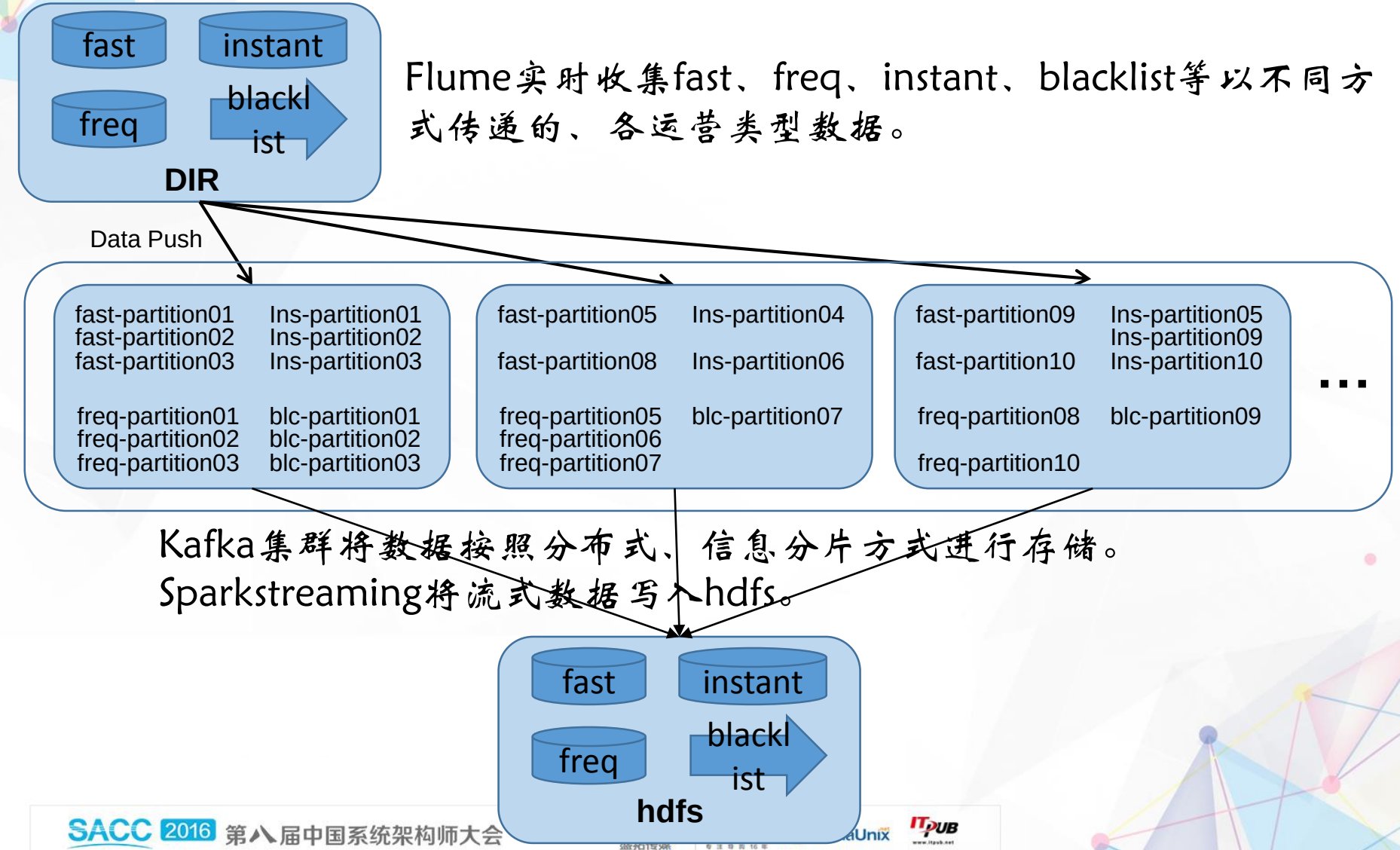
- ✓ Question
- ✓ Answer
- ✓ People
- ✓ Topic

数据推送

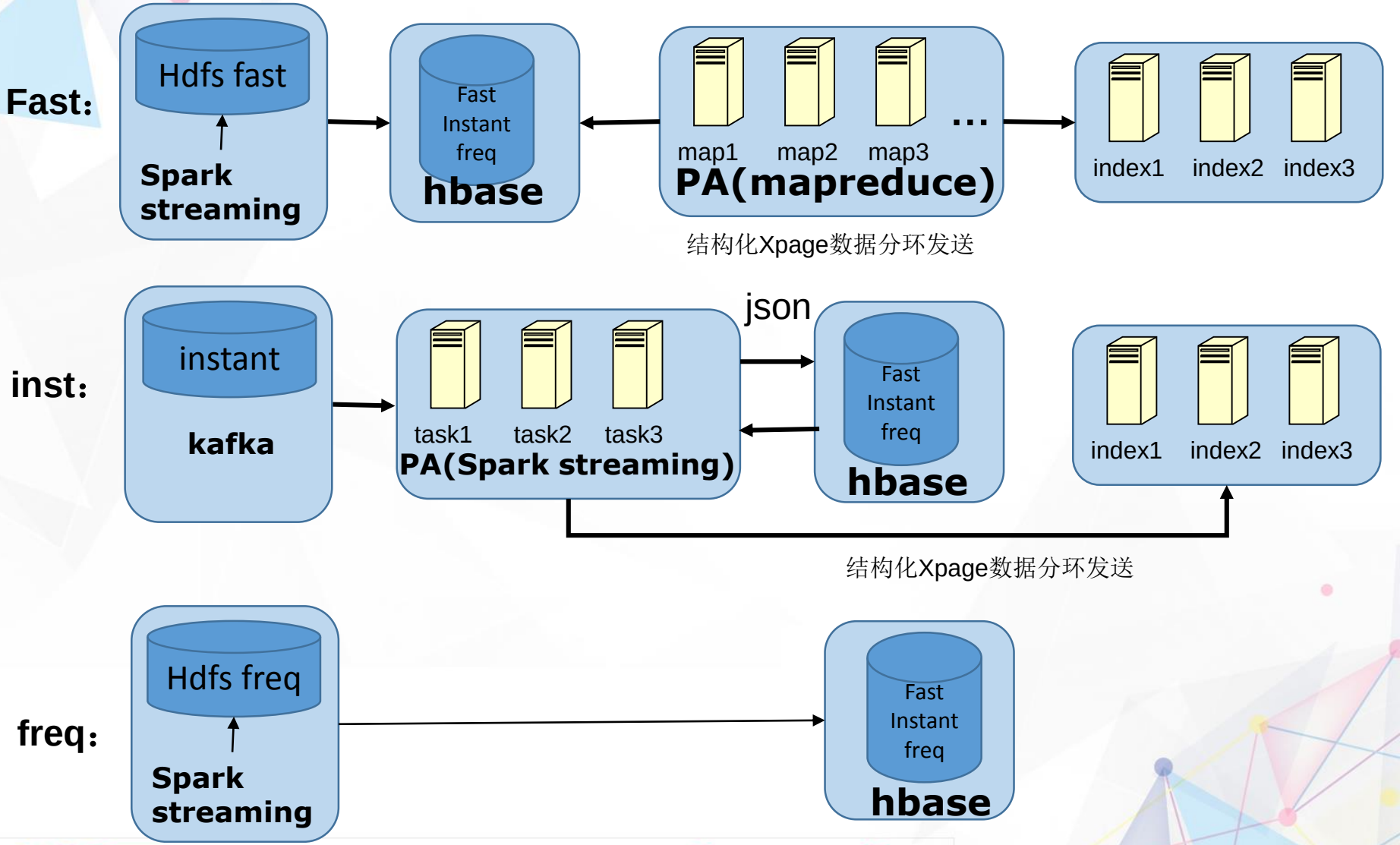
- ✓ 非频繁更新项
- ✓ 频繁更新项

频繁更新的数据内容，包括followerIds、answerMemberIds、answerNumber、followerNumber、 pv、voteUp、 voteDown等。

数据分析、运营架构

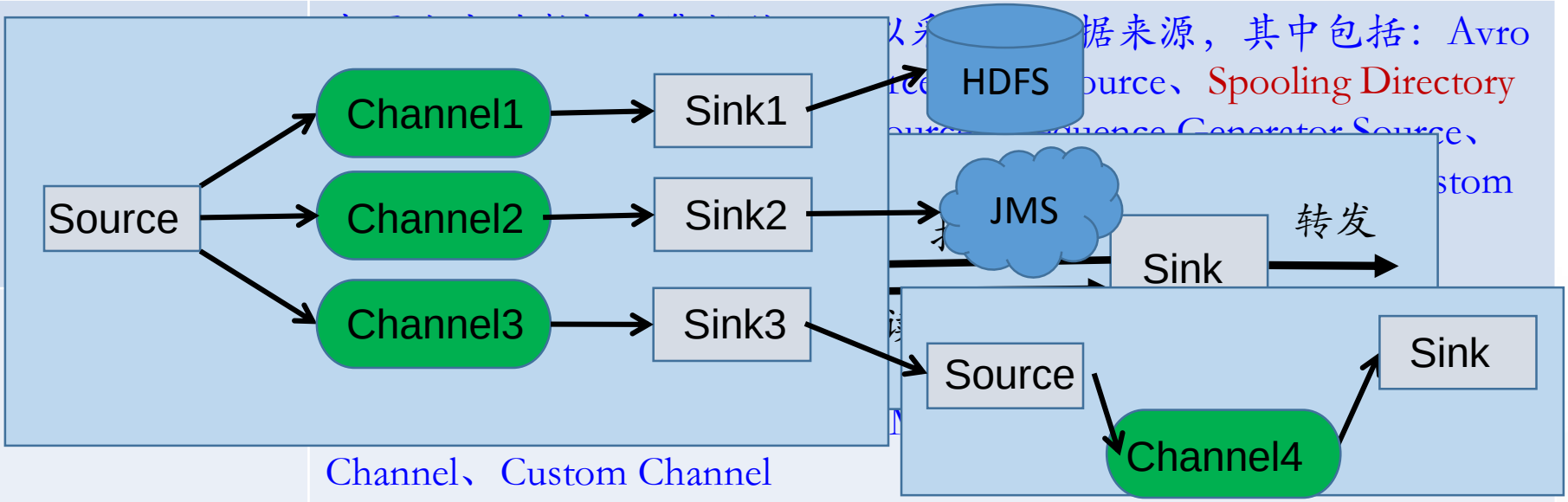

fast
instant
freq
blackl
ist
DIR
Flume实时收集fast、freq、instant、blacklist等以不同方式传递的、各运营类型数据。
Data Push
fast-partition01
fast-partition02
fast-partition03
Ins-partition01
Ins-partition02
Ins-partition03
freq-partition01
freq-partition02
freq-partition03
blc-partition01
blc-partition02
blc-partition03
fast-partition05
fast-partition08
Ins-partition04
Ins-partition06
freq-partition05
freq-partition06
freq-partition07
blc-partition07
fast-partition09
fast-partition10
Ins-partition05
Ins-partition09
Ins-partition10
freq-partition08
freq-partition10
blc-partition09
...
Kafka集群将数据按照分布式、信息分片方式进行存储。
Sparkstreaming将流式数据写入hdfs。
fast
instant
freq
blackl
ist
hdfs

数据分析、运营架构


Fast:
Hdfs fast
Spark streaming
Fast Instant freq
hbase
map1
map2
map3
...
PA(mapreduce)
index1
index2
index3
结构化Xpage数据分环发送
inst:
instant
kafka
task1
task2
task3
PA(Spark streaming)
json
Fast Instant freq
hbase
index1
index2
index3
结构化Xpage数据分环发送
freq:
Hdfs freq
Spark streaming
Fast Instant freq
hbase

Flume主要由3个重要的组件购成：

组件名称	
	据来源，其中包括：Avro ource、Spooling Directory ce、 stom Channel、Custom Channel
Sink	数据出口，数据将通过Sink向外发射，文件系统、数据库、或者提交到远程服务器：HDFS Sink、Hive Sink、Logger Sink、Avro Sink、Thrift Sink、IRC Sink、File Roll Sink、Null Sink、HBase Sink、Async HBase Sink、Morphline Solr Sink、Elastic Search Sink、Kite Dataset Sink、Kafka Sink、Custom Sink。



在知乎中的实际应用：定义source、channel

■ 定义source、channel组件：

```
zhihu-spooldir-kafka-agent.sources  = zhihu-spooldir-source-freq zhihu-spooldir-source-freq3 zhihu-spooldir-source-fast zhihu-spooldir-source-instant
zhihu-spooldir-kafka-agent.channels = zhihu-kafka-channel-freq zhihu-kafka-channel-freq3 zhihu-kafka-channel-fast zhihu-kafka-channel-instant
```

■ 定义zhihu-freq、zhihu-freq3、zhihu-fast、zhihu-instant各组件对应的的配置项

```
# zhihu-freq
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.type = spooldir //注
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.spoolDir = /se
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.fileSuffix
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.del
// 注：deletePolicy这是是否删除读取完毕的文件，默认是"never"，就是不删
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.trackerDir = .flume
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.batchSize = 100 /
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.inputCharset = UT
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.deserializer.maxLi
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.decodeErrorPolicy =
//注：传输的过程中有不可解码的流出现会导致flume停止服务，加上这个配置之后增加flume鲁棒性
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.customSourceCounterType = TimedSourceCounter
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors = timestamp-interceptor static-interceptor circlenumber-interceptor docid-interceptor
//注：在events header中加入key和value的拦截器，包括时间戳、环号、docid
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors.timesta
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors.static-i
//注：静态拦截器，用于在events header中加入一组静态的key和value
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors.static-i
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors.static-i
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interceptors.circl
//注：调用分环算法，在header中加入环号
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.interce
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.in          ptors.docid-i
//注：调用docid算法，在header中加docid
zhihu-spooldir-kafka-agent.sources.zhihu-spooldir-source-freq.channels = zhihu-kafka-channel-freq //注：确认本source对应的channel组件名称
zhihu-spooldir-kafka-agent.channels.zhihu-kafka-channel-freq.type = org.apache.flume.channel.kafka.KafkaChannel //注：定义channel的类型为Kafkachannel。
zhihu-spooldir-kafka-agent.channels.zhihu-kafka-channel-freq.brokerList = 10.1xx.1xx.29:xxxx  //注：设定链接的kafka的ip和端口
zhihu-spooldir-kafka-agent.channels.zhihu-kafka-channel-freq.topic = zhihu-freq  //注：设定本source对应的kafka的topic类型是zhihu-freq
zhihu-spooldir-kafka-agent.channels.zhihu-kafka-channel-freq.zookeeperConnect = 10.1xx.1xx.29:xxxx/kafka
//注：设定需要注册和分配资源的zookeeper的ip和端口
```

Sources.type=spooldir
Sources. spoolDir=/xxx/xxx/
Sources. fileSuffix = .COMPLETED
Sources. Interceptors=circlenm docid
Sources. Channels= zh-kafka-channel-freq

Channels.type=flume.channel.KafkaChannel
Channels. brokerList=ip:port
Channels. topic = zhihu-freq
Channels. zookeeperConnect=ip:port/kafka

在知乎中的实际应用：定义channel、sink

定义channel、sink组件，每个channel都对应一个sink配置：

zhihu-kafka-hdfs-agent.channels = zhihu-kafka-channel-freq-localquery zh
-channel-fast zhihu-kafka-channel-instant zhihu-kafka-channel-blacklist
zhihu-kafka-hdfs-agent.sinks = zhihu-hdfs-sink-freq-localquery zhihu-hdfs
-hdfs-sink-instant zhihu-hdfs-sink-blacklist

定义zhihu-freq-localquery、zhihu-freq、zhihu-freq3、zhihu-fast、zh

zhihu-freq
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-freq.type =
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-freq.br
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-fr pic = zhihu-fr
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-freq.zookeeperConnect
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-freq.groupId = zhihu-freq//注：consumer需要设置其所归属的groupid
zhihu-kafka-hdfs-agent.channels.zhihu-kafka-channel-freq.kafka.fetch.message.max.bytes = 2000000000//注：表示消息的最大大小，单位是字节

zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.type = hdfs
//注：sink组件取出channel队列中的数据，存入相应类型的存储文件系统。这里定义的是存储系统的类型
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.path = hdfs://sss/xxx/xxx/data/zhihu-test/%{pushtype}/sjs_100_29/%Y%m/%Y%m%d
//注：写入hdfs的路径，包含文件系统标识“sss/xxx/xxx/data/”是hdfs的
“%{pushtype}”是在spooldir-kafka的配置文件（sources.zhihu-spooldir
“sjs_100_29”是本测试机的标识；“%Y%m/%Y%m%d”年月的
//注：存储到hdfs里的文件名称：/sss/xxx/xxx/data/zhihu-test/fr
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.w
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.codeC = lzop//注：
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.rollInterval = 120
//注：hdfs sink间隔多长将临时文件滚动成最终目标文件，单位：秒；
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.rollSize = 0
//注：当临时文件达到该大小（单位：bytes）时，滚动成目标文件；如果
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.callTimeout = 120000
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.hdfs.batchSize = 10000//注：每个批次刷新到HDFS上的events数量
zhihu-kafka-hdfs-agent.sinks.zhihu-hdfs-sink-freq.channel = zhihu-kafka-channel-freq//注：该sink对应的channel的名称

Channels.type=flume.channel.KafkaChannel
Channels. brokerList=ip:port
Channels. topic = zhihu-freq
Channels. zookeeperConnect=ip:port/kafka

sinks.type=hdfs
sinks. hdfs.path=/xxx/xxx/
sinks. Channels= zh-kafka-channel-freq

测试相关：Flume

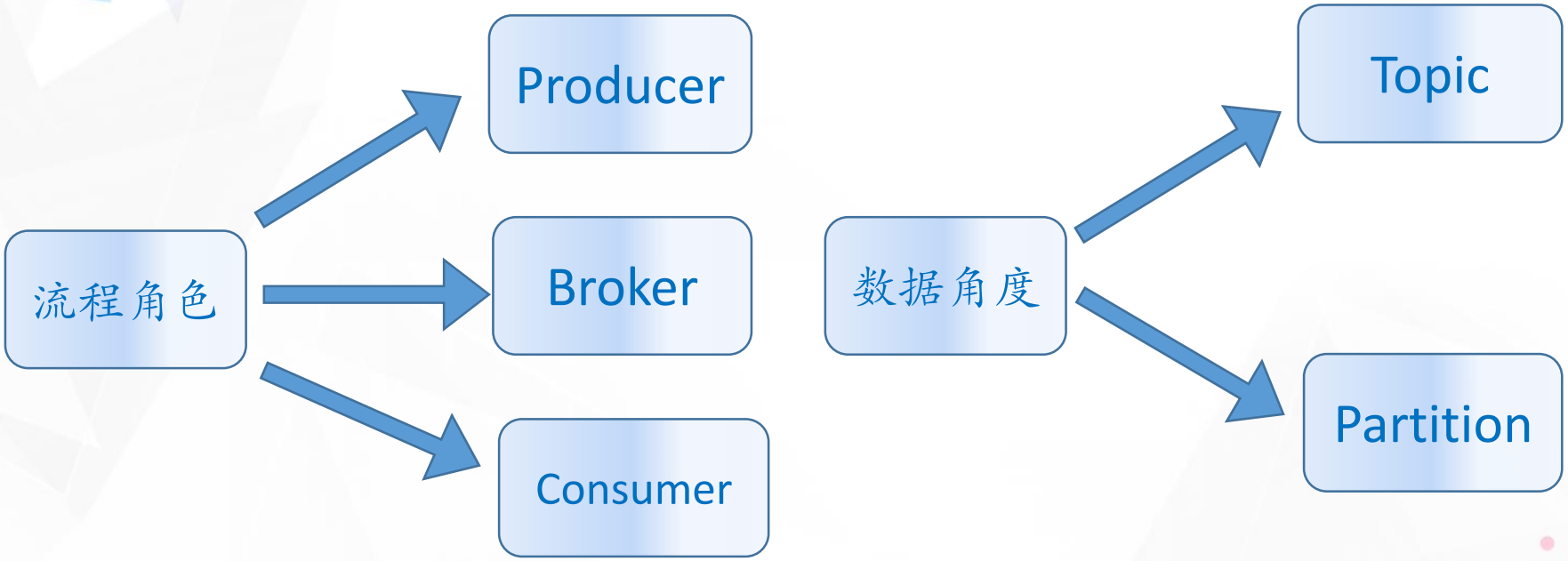
搜狗搜索

关注维度	关注内容
技术特点	➢ 已拷贝到目录下的数据文件不能再打开编辑； ➢ 目录下不能再包含目录；
运维特点	➢ 特定文件名数据文件； ➢ 新文件持续写入； ➢ 文件size不固定； ➢ 不同类型的文件存入不同目录
数据特点	➢ 大文件、小文件 ➢ 所有类型的数据 ➢ 所有类型单条大数据特点，特殊属性数据特点 ➢ 数据完整性、数据正确性

简介：

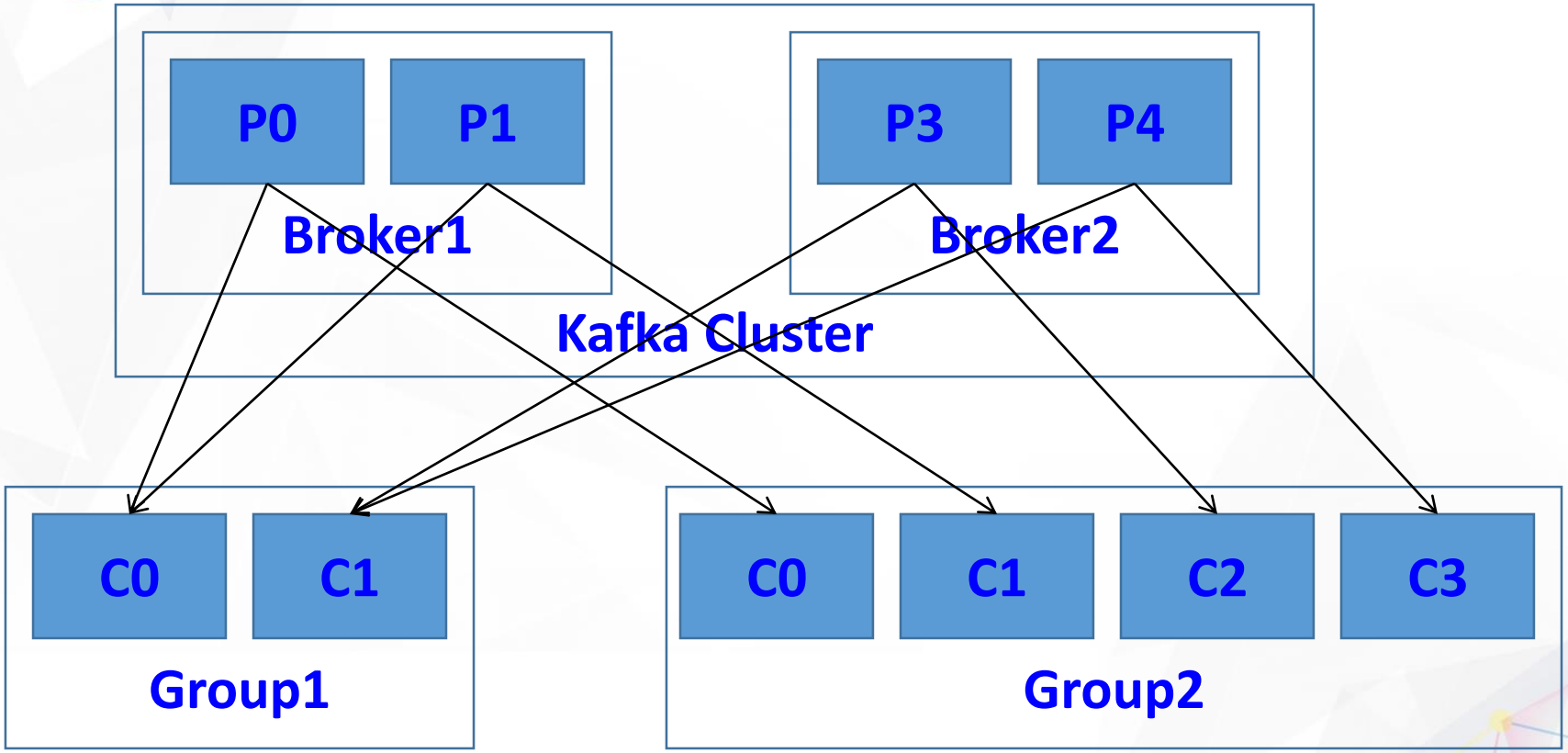
特点	描述
高性能	单节点每秒可处理来自数千节点数百兆数据的读写请求；
可扩展	数据按照partition分布存储在集群中，无需宕机弹性扩容；
持久化	数据在磁盘持久化，并进行多备份；
分布式	分布式设计，保证较强的可用性和容错能力。

Terminology：

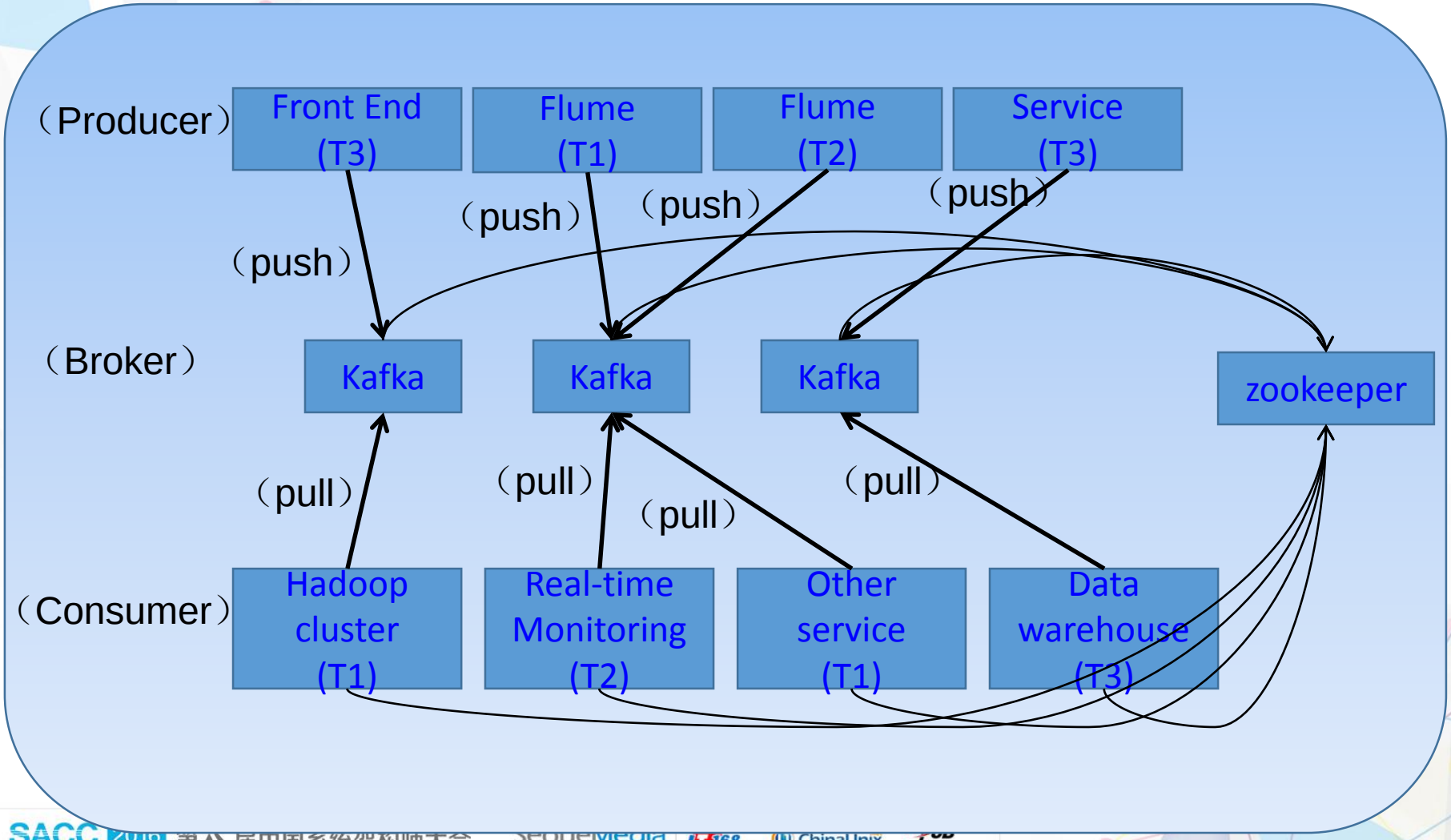


Consumer Group：

P:partition
C:consumer



拓扑结构：


（Producer）
Front End (T3)
Flume (T1)
Flume (T2)
Service (T3)
（push）
（push）
（push）
（push）
（Broker）
Kafka
Kafka
Kafka
zookeeper
（pull）
（pull）
（pull）
（pull）
（Consumer）
Hadoop cluster (T1)
Real-time Monitoring (T2)
Other service (T1)
Data warehouse (T3)

技术细节：Kafka

搜狗搜索

■知乎线上配置的brokerlist：

```
zhihu-spooldir-kafka-agent.channels.zhihu-kafka-channel-freq.brokerList = rsync.broker01.                    d:9092,
rsync.broker02.                    :9092,rsync.                    :9092,rsync.broker04.                    :9092,rs
ync.broker05.                    :9092,rsync.broker06.                    :9092,rsync.broker07.                    :9092,rsyn
c.broker08.                    :9092
```

■知乎线上kafka集群partition存储的情况：

Channels. brokerList=ip1:port, ip2:port, …, ipn:port,

每个Topic分为12个partition，数据3备份，随机均匀分布、存储在8个broker服务器的物理磁盘上。

■测试环境单机kafka的partition存储的情况：

```
[@  data]# pwd
/search/  /kafka/data
[@  data]# ls
recovery-point-offset-checkpoint  zhihu-fast-6   zhihu-freq3-0   zhihu-freq3-8   zhihu-instant-10
replication-offset-checkpoint     zhihu-fast-7   zhihu-freq3-1   zhihu-freq3-9   zhihu-instant-11
zhihu-fast-0                      zhihu-fast-8   zhihu-freq3-10  zhihu-freq-4    zhihu-instant-2
zhihu-fast-1                      zhihu-fast-9   zhihu-freq3-11  zhihu-freq-5    zhihu-instant-3
zhihu-fast-10                     zhihu-freq-0   zhihu-freq3-2   zhihu-freq-6    zhihu-instant-4
zhihu-fast-11                     zhihu-freq-1   zhihu-freq3-3   zhihu-freq-7    zhihu-instant-5
zhihu-fast-2                      zhihu-freq-10  zhihu-freq3-4   zhihu-freq-8    zhihu-instant-6
zhihu-fast-3                      zhihu-freq-11  zhihu-freq3-5   zhihu-freq-9    zhihu-instant-7
zhihu-fast-4                      zhihu-freq-2   zhihu-freq3-6   zhihu-instant-0 zhihu-instant-8
zhihu-fast-5                      zhihu-freq-3   zhihu-freq3-7   zhihu-instant-1 zhihu-instant-9
[@  data]#
```

写入kafka的数据，需要测试的内容：

timestamp

docid

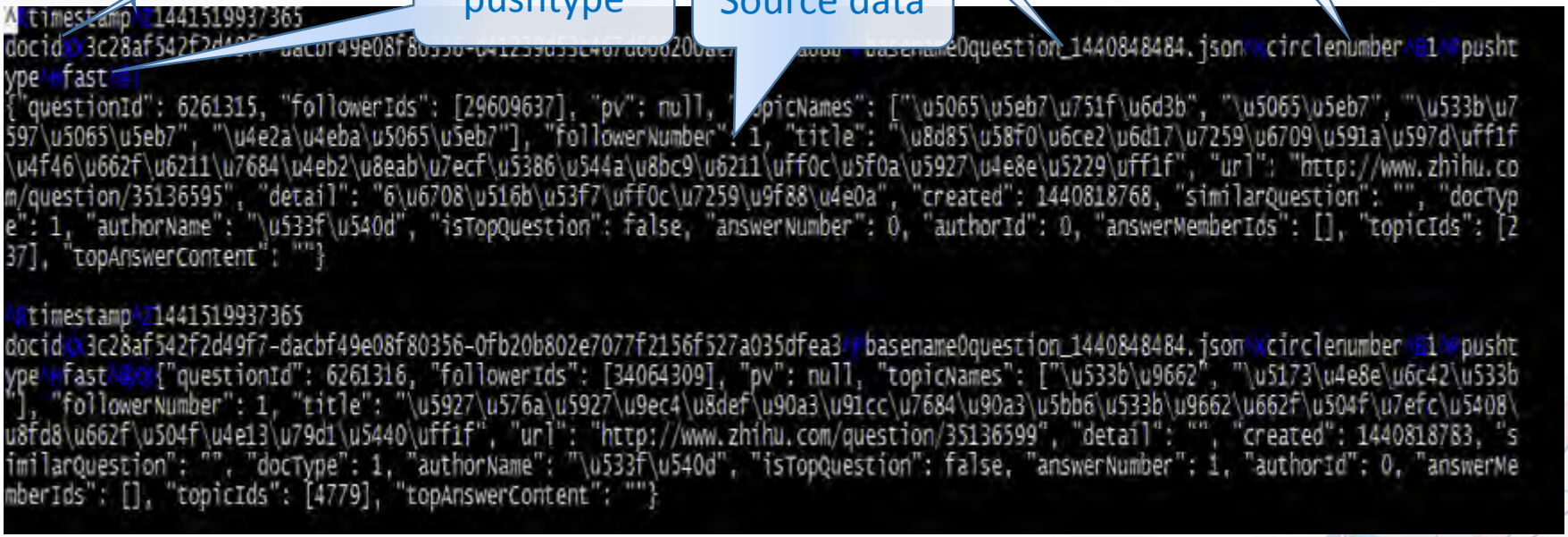
basename

circlenumber

加的header信息，存入的数据stamp、basename是哪个原始文件的数据）、ber（注：该数据类型，伪造各类型数据的url，计算环数，数据总共分8环）、pushtype（注：该数据类型是freq、freq3、instant、fast）

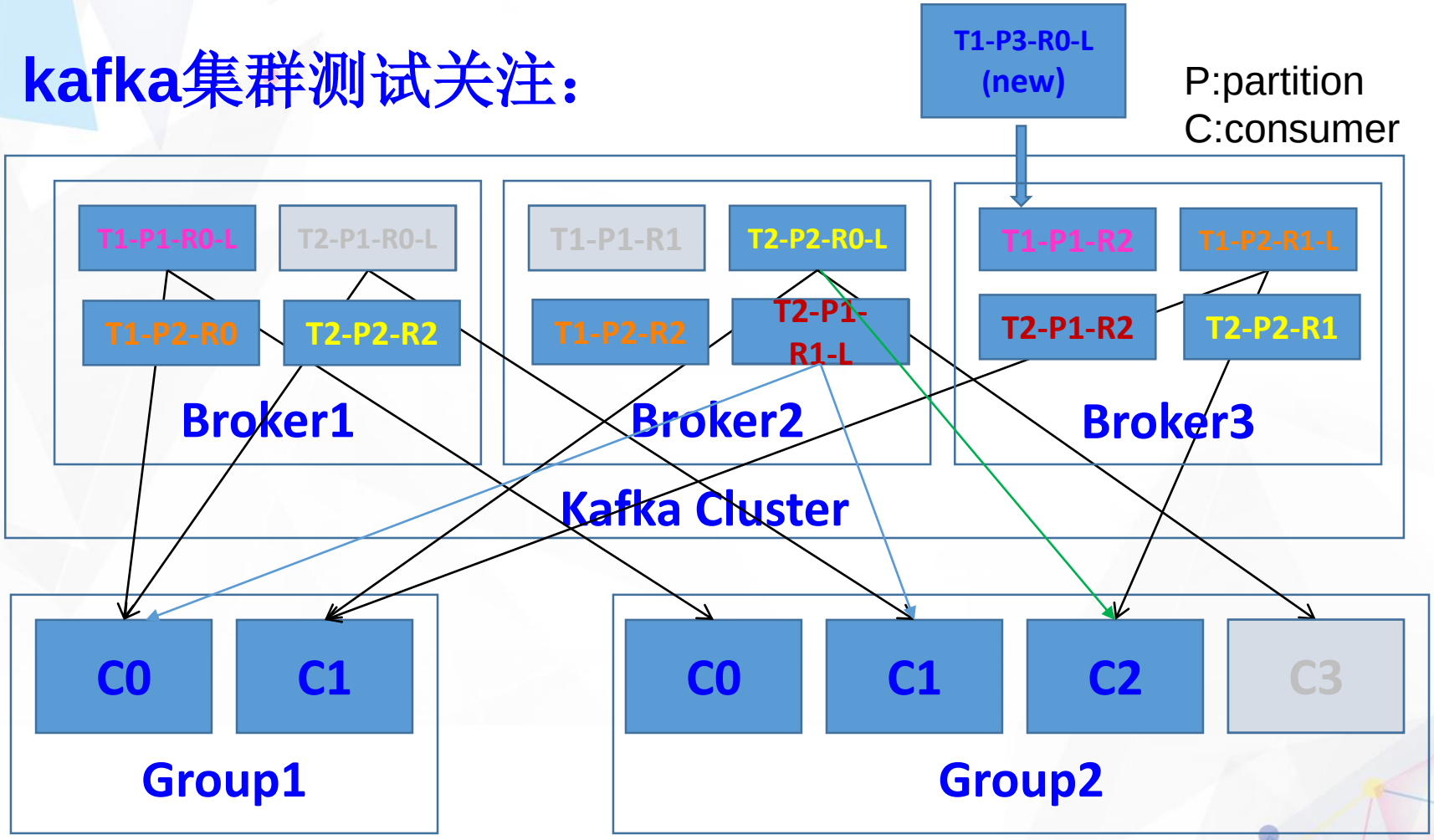
pushtype

Source data



搜狗搜索

kafka集群测试关注：



技术细节：Spark streaming

技术细节：Spark streaming

搜狗搜索

Instant数据流程spark-submit提交任务：

■mainClass=com.sogou.spark.streaming.KafkaStreaming //注：执行[illegible]/xxx/zhihu-kafka-to-hbase/src/main/scala/com/sogou/spark/streaming/KafkaStre[illegible]

■configFile=zhihu-instant-kafka-to-hbase.properties //注：[illegible]读取kafka[illegible]ist，读取kafka的topicid，consumer的Groupid等

■export SPARK_CLASSPATH=.:$confDir:$SPARK_CLASSPATH

■spark-submit \

■ --master yarn-client \//注：以client方式连接到YARN集群，集群的定位由环境变量HADOOP_CONF_DIR定义，该方式driver在client运行。本机需要安装访问相应Hadoop集群的客户端[illegible]的客户端，则sparkstreaming的job就会提交到xxx集群执行。

■ --queue root.default \//注：sparkstream是需要在wapnew集群上占用[illegible]的，这里定义了使用了集群上的哪个队列。

■ --driver-memory 1G \//注：driver memory并不是[illegible]少内存，而是管理多少内存。换言之就是为当前应用分配了多少内存

■ --num-executors 5 \//注：在yarn集群上启动5个进程进行数据处理，[illegible]余进程进行数据处理

■ --executor-memory 1G \//注：executor [illegible]是每个节点上占用的内存。每一个节点可使用内存。executor memory受限于driver memory。当executor memory设置特别大，[illegible]内存依然很小，数据达到一定程度就会爆栈。一般先设置DM（driver memory），随后根据集[illegible]来设置EM（executor memory）。

■ --class $mainClass \

■ --jars $confDir/$configFile,$confDir/hbase-site.xml \//注：Driver依赖的第三方jar包，这里添加了分发数据的index的配置以及回填hbase的地址

■ --conf spark.app.name=zhihu-instant-kafka-to-hbase \//注：App[illegible]，你可以在 Spark 内置 UI 上看到它

■ --conf spark.streaming.receiver.maxRate=80 \

■$libDir/zhihu-kafka-to-hbase-assembly-1.0.jar $configFile

ConfigFile：Kafka相关配置文件，brokerlist、topicid、groupid

Job提交到Yarn-client集群

运行的集群队列

集群中启动数据处理的进程数

引入第三方jar包等

测试关注点：

1、对架构的测试：执行测试时出现过instant流程在源
送到集群后，执行抛异常的情况：在集群上的数
然后把数据序列化之后发送给剩余的进
作even比较弱，所以通过yarn执行的时候涉及到数据分
flume的代码需要对even函数进行修改；但本机运行的时候不涉及数据分发所以能够跑通。

2、对业务功能的正确性测试：

1）对于知乎instantPA解析xpage、xslt内容解析正确性

2）针对不同运营流程数据（fast
中不存在fast数据，但先收到了freq数据时；库中已存在
instant数据时；库中已存在fast、instant数据，又收到

3）针对不同type的数据进行的测试：answer类数据不能
包含该answer对应的question数据时，再收到该question
要将hbase该question对应的所有数据全部取出来（question的fast、instant、freq数据+answer的instant、freq数据）再进行完整内容的解析，且保证新收到的answer数据能够写到hbase库中所属question的对应answer列中。

本机数据处理的执行流程与分布式数据处理的执行流程不同，有些函数需要了解并支持分布式数据处理特点。

不同业务流程涉及不同的测试内容，通过分析业务流程各阶段的处理过程、数据发送和接收流程对数据的操作的影响，进行有针对性的测试。

技术细节：Map reduce

搜狗搜索

知乎fast流程：定期启动MapReduce，扫hbase库，给下游模块发送数据。


```
hadoop jar bin/QDBLikeHBasePortal-0.0.4-SNAPSHOT.jar -
Dmapreduce.map.memory.mb=2000 -Dmapreduce.map.java.opts=-Xmx1800m -
Dmapreduce.job.queuename=root.default -
Dmapreduce.task.classpath.user.precedence=true -libjars bin/json-lib-
2.2.3.jar,bin/ezmorph-1.0.4.jar ./zhihu.xml &>zhihu.log_$mydate &
```

了解MapReduce框架结构中数据处理流程，从代码设计角度提醒研发注意。

测试经验总结：

1）知乎PA解析增加繁简转换功能，研发同学一开始使用的是c++代码，需要在hadoop上运行的MapReduce框架加载C++编译出来的.SO文件，根据之前测试和学习的经验感觉通过指定export LD_LIBRARY_PATH的方法不符合MapReduce执行架构的特点，与平台组研发同学沟通后确认这种方法在运营上确实不方便，hadoop框架里jar包的依赖包的路径是java.library.path=/usr/lib/hadoop/lib/native目录下，这个目录最好不要有外部程序，以免污染hadoop自身运行环境。

2）研发组同学经过协商后使用了前端java代码的繁简转换模块，但需要加载外部data目录的数据文件。根据hadoop的机制，需要修改fast、instant流程的执行脚本，增加外部数据上传的参数，hadoop将data文件存储在hdfs上，再分发到Map执行服务器，这个方法也会增加运营成本。跟研发沟通后，又与前端研发沟通确认可以直接把data文件打到jar包里。

结论：此时基本达到了修改代码尽量不增加运营成本和运营风险的要求。

THANKS

SequeMedia 盛拓传媒 | IT168 专注导购16年 | ChinaUnix | ITPUB