链滴

企业级数据仓库构建（四）：数据仓库项目实战

作者：fc13240

原文链接：https://ld246.com/article/1600157254015

来源网站：链滴

总结

1）数仓概念总结

【1】数据仓库的输入数据源和输出系统分别是什么？

输入系统：埋点产生的用户行为数据、JavaEE 后台产生的业务数据
输出系统：报表系统、用户画像系统、推荐系统

2）项目需求及架构总结

【1】集群规模计算

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

【2】框架版本选型

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，但其实对中、小公司使用来说没有影响
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

【3】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。

3）数据采集模块总结

【1】Linux&Shell 相关总结

1. Linux 常用高级命令

序号	命令	命令解释
1	top	查看内存
2	df -h	查看磁盘存储情况
3	iotop	查看磁盘 IO 读写(yum install iotop 安装）
4	iotop -o	直接查看比较高的磁盘读写程序
5	netstat -tunlp | grep 端口号	查看端口占用情况
6	uptime	查看报告系统运行时长及平均负载
7	ps aux	查看进程

2. Shell 常用工具

awk、sed、cut、sort

【2】Hadoop 相关总结

1. Hadoop 默认不支持 LZO 压缩，如果需要支持 LZO 压缩，需要添加 jar 包，并在 hadoop 的 cores-site.xml 文件中添加相关压缩配置。需要掌握让 LZO 文件支持切片

2. Hadoop 常用端口号，50070,8088,19888,8020

3. Hadoop 配置文件以及简单的 Hadoop 集群搭建。8 个配置文件

4. HDFS 读流程和写流程

5. MapReduce 的 Shuffle 过程及 Hadoop 优化（包括：压缩、小文件、集群优化）
6. Yarn 的 Job 提交流程
7. Yarn 的默认调度器、调度器分类、以及他们之间的区别
8. HDFS 存储多目录
9. Hadoop 参数调优

10）项目经验之基准测试

【3】Zookeeper 相关总结

1. 选举机制
半数机制，安装奇数台
10 台服务器几台：3 台
20 台服务器几台：5 台
100 台服务器几台：11 台
不是越多越好，也不是越少越好。 如果多，通信时间长，效率低；如果太少，可靠性差
2. 常用命令
ls、get、create

【4】Flume 相关总结

1）Flume 组成，Put 事务，Take 事务
Source 到 Channel 是 Put 事务
Channel 到 Sink 是 Take 事务

Taildir Source：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

File Channel：数据存储在磁盘，宕机数据可以保存。但是传输速率慢。适合对数据传输可靠性要求高的场景，比如，金融行业

Memory Channel：数据存储在内存中，宕机数据丢失。传输速率快。适合对数据传输可靠性要求不高的场景，比如，普通的日志数据

Kafka Channel：减少了 Flume 的 Sink 阶段，提高了传输效率

2）Flume 拦截器
（1）拦截器注意事项
项目中自定义了：ETL 拦截器和区分类型拦截器。
采用两个拦截器的优缺点：优点，模块化开发和可移植性；缺点，性能会低一些

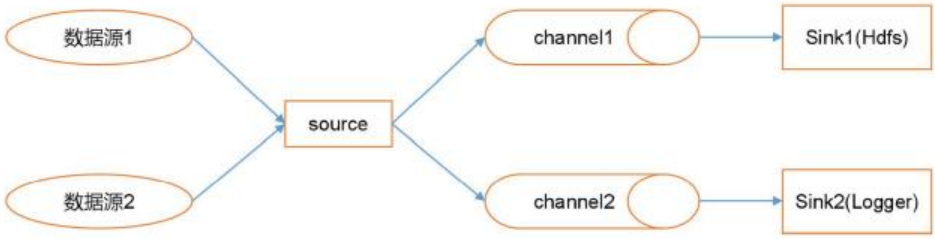
（2）自定义拦截器步骤
a）实现 Interceptor

b）重写四个方法

- initialize 初始化
- public Event intercept(Event event) 处理单个 Event
- public List intercept(List events) 处理多个 Event，在这个方法中调用 Event intercept(Event event)
- close 方法

c）静态内部类，实现 Interceptor.Builder

3）Flume Channel 选择器



Channel Selectors，可以让不同的项目日志通过不同的Channel到不同的Sink中去。官方文档上Channel Selectors 有两种类型:Replicating Channel Selector (default)和 Multiplexing Channel Selector

这两种Selector的区别是:Replicating 会将source过来的events发往所有channel,而Multiplexing可以选择该发往哪些Channel。

4）Flume 监控器

Ganglia

5）Flume 采集数据会丢失吗?

不会，Channel 存储可以存储在 File 中，数据传输自身有事务

6）Flume 内存

开发中在 flume-env.sh 中设置 JVM heap 为 4G 或更高，部署在单独的服务器上（4 核 8 线程 16G 内存)

-Xmx 与-Xms 最好设置一致，减少内存抖动带来的性能影响，如果设置不一致容易导致频繁 fullgc

-Xms 表示 JVM Heap(堆内存)最小尺寸，初始分配；-Xmx 表示 JVM Heap(堆内存)最大允许的尺寸，按需分配。如果不设置一致，容易在初始化时，由于内存不够，频繁触发 fullgc

7）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 peformance

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

8）Sink：HDFS Sink 小文件处理

（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

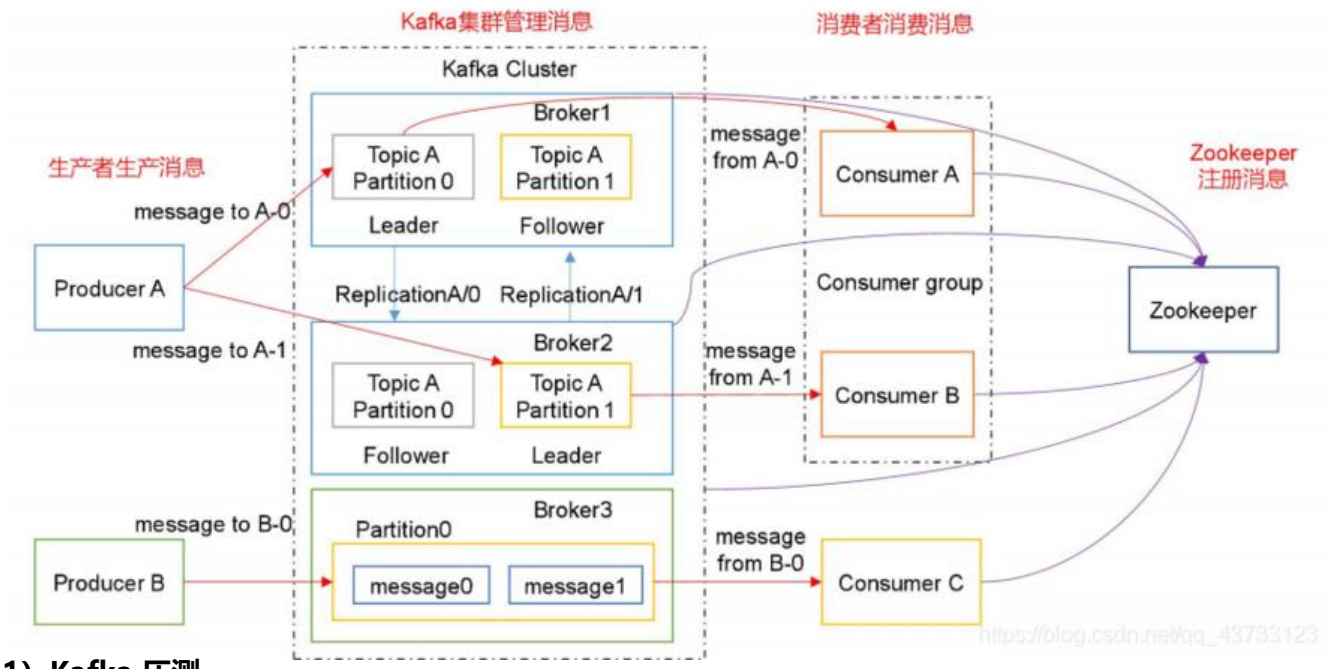
官方默认的这三个参数配置写入 HDFS 后会产生小文件，hdfs.rollInterval、hdfs.rollSize、hdfs.rollCount

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

【5】Kafka 相关总结



1）Kafka 压测

Kafka 官方自带压力测试脚本（kafka-consumer-perf-test.sh、kafka-producer-perf-test.sh）。Kafka 压测时，可以查看到哪个地方出现了瓶颈（CPU，内存，网络 IO）。一般都是网

络 IO 达到瓶颈

2）Kafka 的机器数量

Kafka 机器数量=2*（峰值生产速度*副本数/100）+1

3）Kafka 的日志保存时间

3 天

4）Kafka 的硬盘大小

每天的数据量*3 天

5）Kafka 监控

公司自己开发的监控器

开源的监控器：KafkaManager、KafkaMonitor

6）Kakfa 分区数

（1）创建一个只有 1 个分区的 topic

（2）测试这个 topic 的 producer 吞吐量和 consumer 吞吐量。

（3）假设他们的值分别是 Tp 和 Tc，单位可以是 MB/s。

（4）然后假设总的目标吞吐量是 Tt，那么分区数=Tt / min（Tp，Tc）

例如：producer 吞吐量=10m/s；consumer 吞吐量=50m/s，期望吞吐量 100m/s；

分区数=100 / 10 =10 分区

分区数一般设置为：3-10 个

7）副本数设定

一般我们设置成 2 个或 3 个，很多企业设置为 2 个

8）多少个 Topic

通常情况：多少个日志类型就多少个 Topic。也有对日志类型进行合并的

9）Kafka 丢不丢数据

Ack=0，producer 不等待 kafka broker 的 ack，一直生产数据

Ack=1，leader 数据落盘就发送 ack，producer 收到 ack 才继续生产数据

Ack=-1，ISR 中的所有副本数据罗盘才发送 ack，producer 收到 ack 才继续生产数据

10）Kafka 的 ISR 副本同步队列

ISR（In-Sync Replicas），副本同步队列。ISR 中包括 Leader 和 Follower。如果 Leader 进程挂掉，会在 ISR 队列中选择一个服务作为新的 Leader。有 replica.lag.max.messages（延迟条数）和 replica.lag.time.max.ms（延迟时间）两个参数决定一台服务是否可以加入 ISR 副本队列，在 0.10 版本移除了 replica.lag.max.messages 参数，防止服务频繁的进去队列。

任意一个维度超过阈值都会把 Follower 剔除出 ISR，存入 OSR（Outof-Sync Replicas）列表，新加入的 Follower 也会先存放在 OSR 中

11）Kafka 分区分配

Range 和 RoundRobin

12）Kafka 中数据量计算

每天总数据量 100g，每天产生 1 亿条日志， 10000 万/24/60/60=1150 条/每秒钟

平均每秒钟：1150 条

低谷每秒钟：400 条

高峰每秒钟：1150 条*（2-20 倍）=2300 条-23000 条

每条日志大小：0.5k-2k（取 1k）

每秒多少数据量：2.0M-20MB

13） Kafka 挂掉

（1）Flume 记录

（2）日志有记录

（3）短期没事

14）Kafka 消息数据积压，Kafka 消费能力不足怎么处理?

（1）如果是 Kafka 消费能力不足，则可以考虑增加 Topic 的分区数，并且同时提升消费组的消费者数量，消费者数=分区数。（两者缺一不可）

（2）如果是下游的数据处理不及时：提高每批次拉取的数量。批次拉取数据过少（拉取数据/处理时间<生产速度），使处理的数据小于生产的数据，也会造成数据积压

15）Kafka 幂等性

Kafka0.11 版本引入了幂等性，幂等性配合 at least once 语义可以实现 exactly once 语义。但只能保证单次会话的幂等。

16）Kafka 事务

Kafka0.11 版本引入 Kafka 的事务机制，其可以保证生产者发往多个分区的一批数据的原子性。

原文：

【项目】数仓项目（四）