法律声明

- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：*http://www.ibeifeng.com/*

消息订阅框架KAFKA

高吞吐量的分布式发布订阅消息系统

主讲人：Gerry

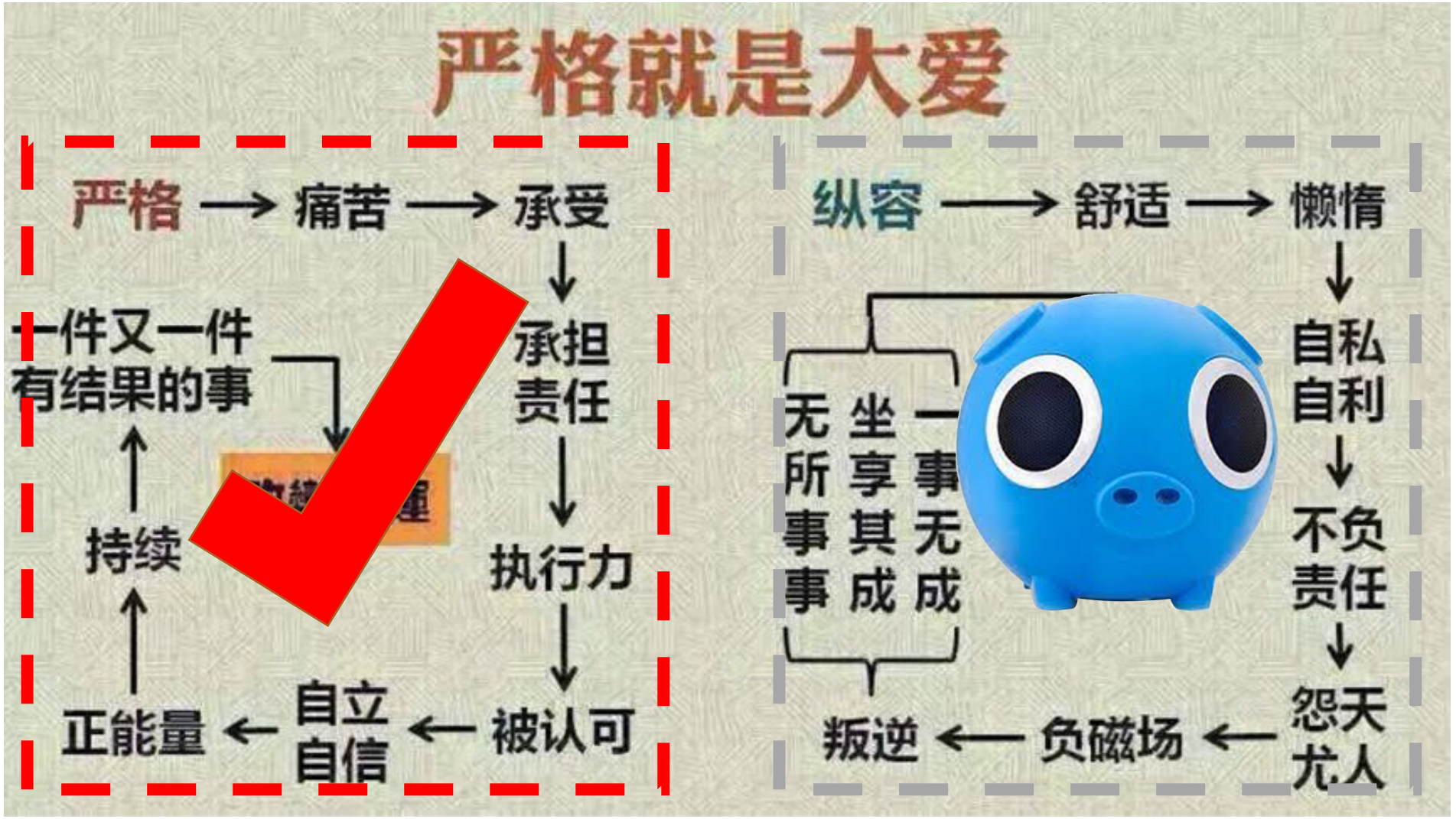
上海育创网络科技有限公司

北风网 IBEIFENG.COM

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

严格是大爱


严格就是大爱
严格 → 痛苦 → 承受
承担责任
执行力
被认可
自立自信
正能量
持续
一件又一件有结果的事
纵容 → 舒适 → 懒惰
自私自利
不负责任
怨天尤人
负磁场
叛逆
一事无成
坐享其成
无所事事

寄语

做别人不愿做的事，

做别人不敢做的事，

做别人做不到的事。

学习内容与目标

- Kafka初识
- Kafka功能架构
- Kafka重要概念
- Kafka安装部署与测试
- Kafka Producer讲解
- Kafka Consumer讲解
- Kafka与Flume集成
- Kafka与Log4j集成
- Kafka集群监控

What is Kafka™?

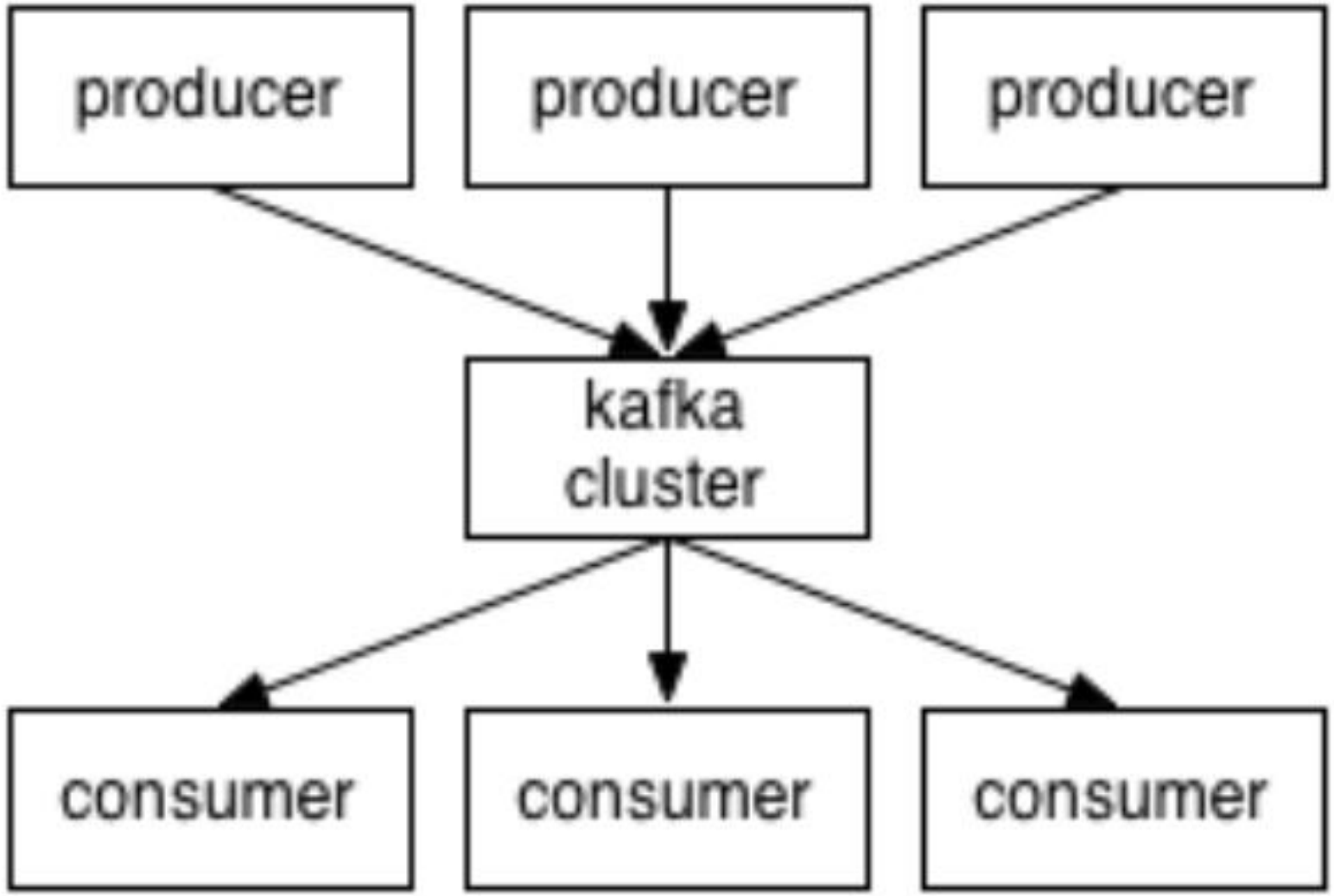
- It lets you **publish** and **subscribe** to streams of **records**. In this respect it is similar to a **message queue** or enterprise **messaging system**
- It lets you **store** streams of **records** in a **fault-tolerant** way
- It lets you **process** streams of records as they **occur**

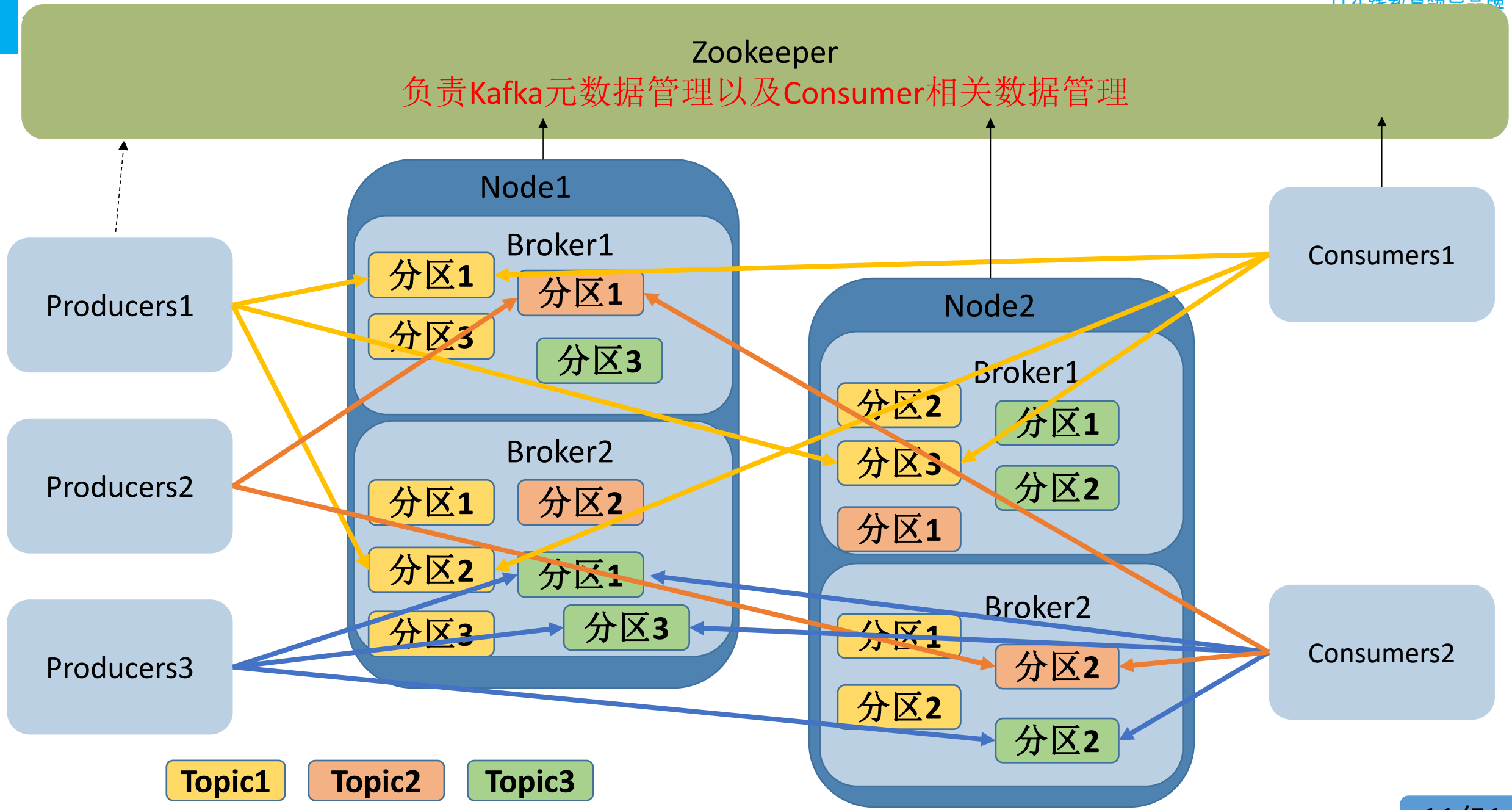
What Kafka can do & Kafka features?

- Kafka™ is a **distributed**, **partitioned**, **replicated** commit log service
- Kafka™ is used for building **real-time data pipelines** apps.
- **Features**:
  - **Horizontally Scalable**：水平可扩展(扩展性)
  - **Fault-tolerant**：容错(容错性&可用性&可靠性)
  - **Fast**：快速
  - **Distributed**：分布式

Kafka适用场景

- 由于Kafka存在高容错、高扩展、分布式等特性，Kafka主要应用场景如下：
  - 消息系统
  - 日志收集系统
  - Metrics监控系统


producer
producer
producer
kafka
cluster
consumer
consumer
consumer

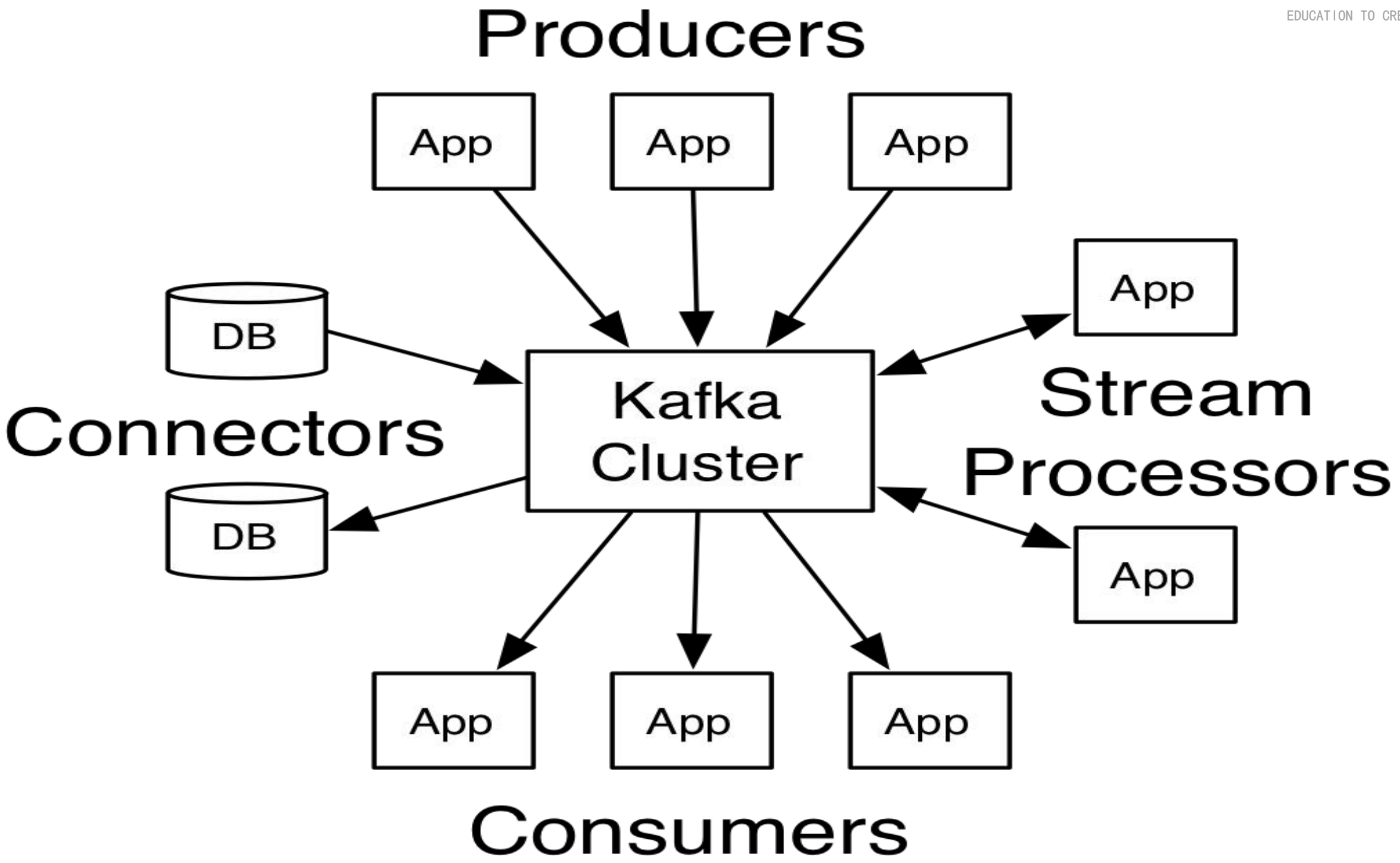

Zookeeper
负责Kafka元数据管理以及Consumer相关数据管理
Producers1
Producers2
Producers3
Node1
Broker1
分区1
分区3
分区1
分区3
Broker2
分区1
分区2
分区2
分区1
分区3
分区3
Node2
Broker1
分区2
分区1
分区3
分区2
分区1
Broker2
分区1
分区2
分区2
分区2
Consumers1
Consumers2
Topic1
Topic2
Topic3

Kafka基本信息术语

- **Message**(消息)：传递的数据对象，主要由四部分构成：offset(偏移量)、key、value、timestamp(插入时间)
- **Broker**(代理者)：Kafka集群中的机器/服务被成为broker， 是一个物理概念。
- **Topic**(主题)：维护Kafka上的消息类型被称为Topic，是一个逻辑概念。
- **Partition**(分区)：具体维护Kafka上的消息数据的最小单位，一个Topic可以包含多个分区；Partition特性：**ordered** & **immutable**。(在数据的产生和消费过程中，不需要关注数据具体存储的Partition在那个Broker上，只需要指定Topic即可，由Kafka负责将数据和对应的Partition关联上)
- **Producer**(生产者)：负责将数据发送到Kafka对应Topic的进程
- **Consumer**(消费者)：负责从对应Topic获取数据的进程
- **Consumer Group**(消费者组)：每个consumer都属于一个特定的group组，一个group组可以包含多个consumer，但一个组中只会有一个consumer消费数据。

Kafka0.10.x版本新特性

- Kafka不仅仅可以作为消息系统，还可以作为流式数据处理平台和数据存储平台，已经完全成为一个分布式的流式数据平台，在已有的概念之上，新增了两个新的功能概念，分别是：
  - Streams：处理Kafka上流式数据的一个模块/API，类似于Storm、SpearkStreaming等流式处理平台
  - Connector：提供可重用的生产者、消费者，可以将Kafka中的数据持久化到存储系统或者从存储系统中读入数据，比较类似Flume的数据收集功能


Producers
App
App
App
DB
DB
Connectors
Kafka
Cluster
App
Stream
Processors
App
App
App
App
Consumers

Kafka安装介绍

- Kafka是由LinkedIn公司开发的，之后贡献给Apache基金会，成为Apache的一个顶级项目，开发语言为Scala。提供了各种不同语言的API，具体参考Kafka的cwiki页面；
- 安装方式主要由三种，分别是：单机、伪分布式、完全分布式；其中伪分布式和完全分布式基本一样
- 安装步骤：
  - 安装JAVA和Scala
  - 安装Zookeeper
  - 安装Kafka

Kafka安装(伪分布式)

- 下载安装包、解压并配置环境变量KAFKA_HOME
- 修改配置文件${KAFKA_HOME}/conf/server.properties。如果是伪分布式，那么需要在的单台机器上copy多个server.properties文件；如果是完全分布式，那么需要将修改好的KAFKA完全copy到其他机器上
- 启动Kafka服务，启动命令如下(伪分布式)：
  - ${KAFKA_HOME}/bin/kafka-server-start.sh xxx/server0.properties
  - ${KAFKA_HOME}/bin/kafka-server-start.sh xxx/server1.properties
- 关闭服务使用${KAFKA_HOME}/bin/kafka-server-stop.sh进行操作

Kafka安装配置项(一)

```
[beifeng@hadoop-senior02 config]$ ls
consumer.properties   server1.properties   test-log4j.properties
log4j.properties      server2.properties   tools-log4j.properties
producer.properties   server3.properties   zookeeper.properties
server0.properties    server.properties
[beifeng@hadoop-senior02 config]$
```

伪分布式情况下，server配置信息

```
20 # The id of the broker. This must be set to a unique integer for
   each broker.
21 broker.id=0
22
23 ############################# Socket Server Settings
   #############################
24
25 # The port the socket server listens on
26 port=9092
27
28 # Hostname the broker will bind to. If not set, the server will bind
   to all interfaces
29 host.name=hadoop-senior02.ibeifeng.com
30
```

服务器唯一标识符

服务器监听端口号

服务器监听主机名或者IP地址

Kafka安装配置项(二)

```
58 # A comma seperated list of directories under which to store log files
59 log.dirs=/opt/cdh-5.3.6/kafka_single/data
60
```

Kafka数据存储本地磁盘路径

```
114 # Zookeeper connection string (see zookeeper docs for details).
115 # This is a comma separated host:port pairs, each corresponding to a
    zk
116 # server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
117 # You can also append an optional chroot string to the urls to
    specify the
118 # root directory for all kafka znodes.
119 zookeeper.connect=hadoop-senior02.ibeifeng.com:2181/kafka
120
121 # Timeout in ms for connecting to zookeeper
122 zookeeper.connection.timeout.ms=6000
123
```

Kafka元数据管理ZOOKEEPER配置信息

Kafka基本操作

- 创建Topic
- 列出Topic
- 查看Topic信息
- 修改Topic
- 启动Kafka自带Producer和Consumer进行数据测试

Kafka发送消息格式

- 一个Kafka的Message由一个固定长度的**header**和一个变长的消息体**body**组成
- **header**部分由一个字节的**magic**(文件格式)和四个字节的**CRC32**(用于判断body消息体是否正常)构成。当magic的值为1的时候，会在magic和crc32之间多一个字节的数据：**attributes**(保存一些相关属性，比如是否压缩、压缩格式等等)；如果magic的值为0，那么不存在attributes属性
- **body**是由N个字节构成的一个消息体，包含了具体的key/value消息
- **备注：每个版本的Kafka消息格式是不一样的**

Kafka Log消息格式(一)

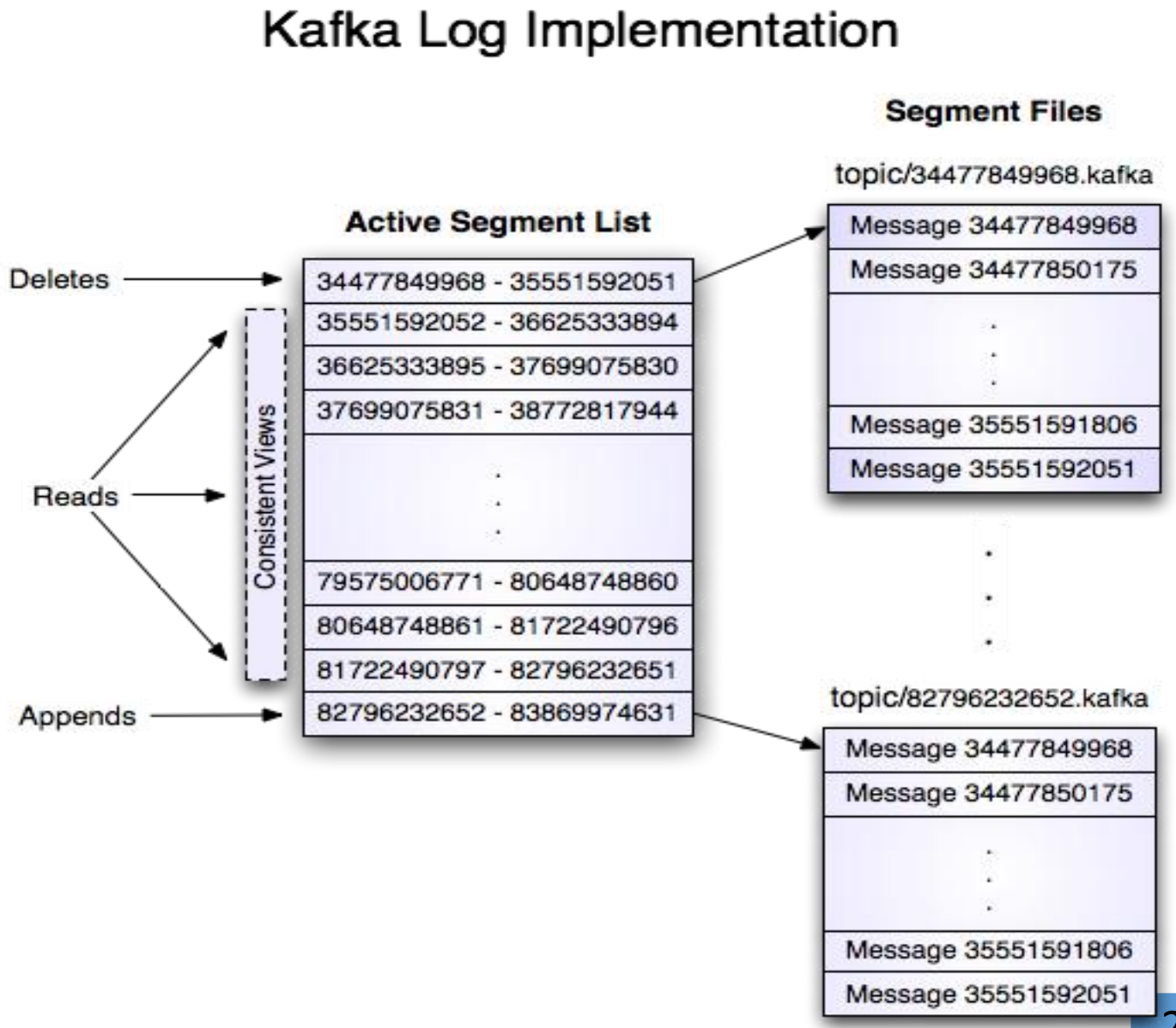
- 存储在磁盘的日志采用不同于Producer发送的消息格式，每个日志文件都是一个“log entries”序列，每一个log entry包含一个四字节整型数(message长度，值为1+4+N)，一个字节的magic，四个字节的CRC32值，最终是N个字节的消息数据。每条消息都有一个当前Partition下唯一的64位offset，指定该消息的起始下标位置，存储消息格式如下：

```
On-disk format of a message

message length : 4 bytes (value: 1+4+n)
"magic" value  : 1 byte
crc            : 4 bytes
payload        : n bytes
```

Kafka Log消息格式(二)

- 这个“log entries”并非由一个文件构成，而是分成多个segment file(日志文件，存储具体的消息记录)和一个索引文件(存储每个segment文件的offset偏移量范围)。 结构如右图所示：

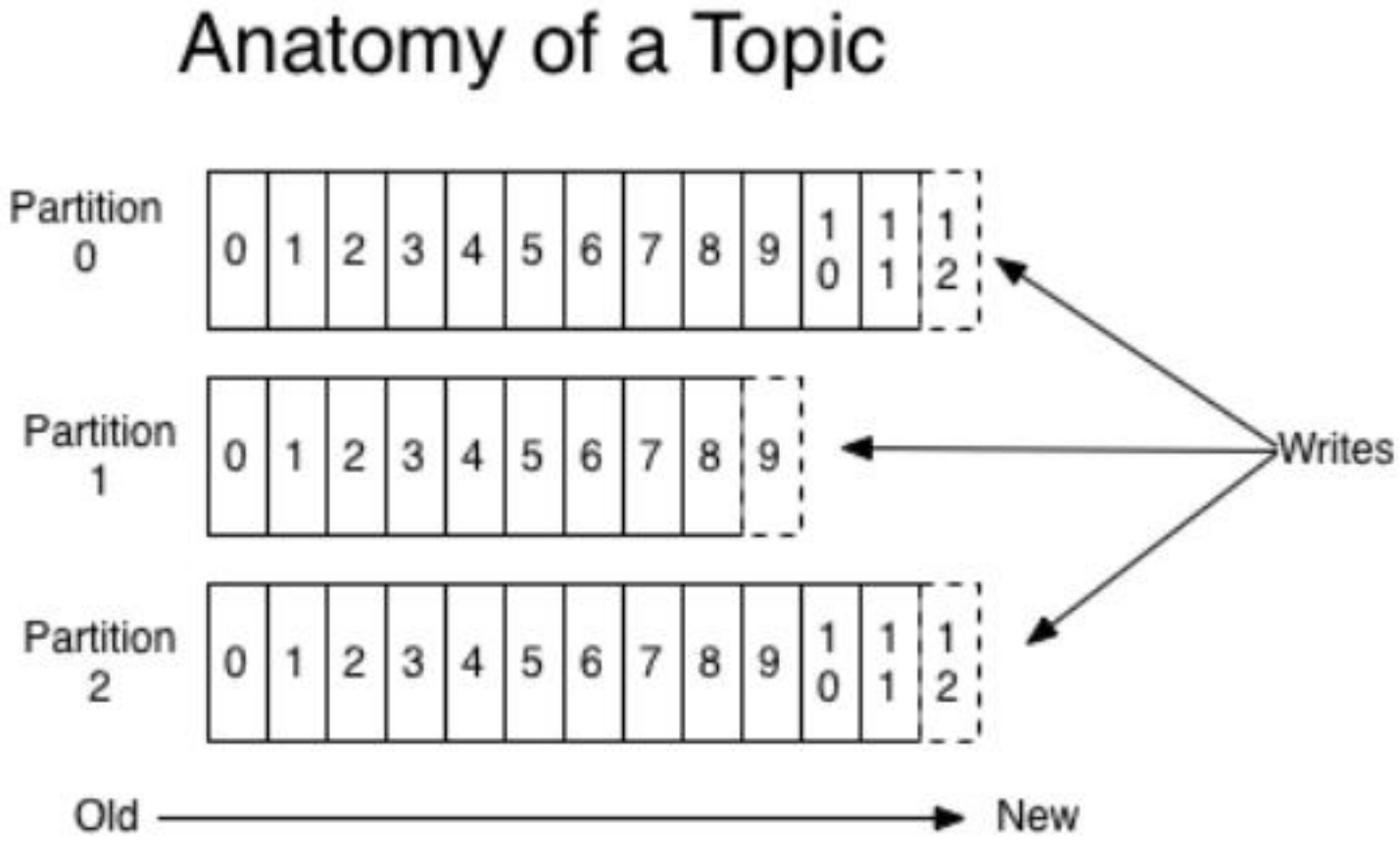


Kafka消息存储机制(一)

- 一个Topic分为多个Partition来进行数据管理，一个Partition中的数据是有序、不可变的，使用偏移量(offset)唯一标识一条数据，是一个long类型的数据
- Partition接收到producer发送过来数据后，会产生一个递增的offset偏移量数据，同时将数据保存到本地的磁盘文件中(文件内容追加的方式写入数据)；Partition中的数据存活时间超过参数值(log.retention.{ms,minutes,hours}，默认7天)的时候进行删除(默认)
- Consumer根据offset消费对应Topic的Partition中的数据(也就是每个Consumer消费的每个Topic的Partition都拥有自己的offset偏移量)
- **注意**：Kafka的数据消费是**顺序读写**的，磁盘的顺序读写速度(600MB/sec)比随机读写速度(100k/sec)快

Kafka消息存储机制(二)

Offset	0	1	2	3	4	5	6	7	8	9	10
Key	K1	K2	K1	K1	K3	K2	K4	K5	K5	K2	K6
Value	V1	V2	V3	V4	V5	V6	V7	V8	V9	V 10	V 11



Kafka分布式机制

- 一个Topic中的所有数据分布式的存储在kafka集群的所有机器(broker)上，以分区(partition)的的形式进行数据存储；每个分区允许存在**备份数据/备份分区**(存储在同一kafka集群的其它broker上的分区)
- 每个数据分区在Kafka集群中存在一个broker节点上的分区叫做**leader**，存储在其它broker上的备份分区叫做**followers**；只有leader节点负责该分区的数据读写操作，followers节点作为leader节点的热备节点，从leader节点备份数据；当leader节点挂掉的时候，followers节点中会有一个节点变成leader节点，重新提供服务
- Kafka集群的Partition的leader和followers切换依赖Zookeeper

Kafka消息产生/收集机制

- Kafka集群中由producer负责数据的产生，并发送到对应的Topic；Producer通过**push**的方式将数据发送到对应Topic的分区
- Producer发送到Topic的数据是有key/value键值对组成的，Kafka根据key的不同的值决定数据发送到不同的Partition，默认采用**Hash**的机制发送数据到对应Topic的不同Partition中，配置参数为{partitioner.class}
- Producer发送数据的方式分为**sync**(同步)和**async**(异步)两种，默认为同步方式，由参数{producer.type}决定；当发送模式为异步发送的时候，Producer提供重试机制，默认失败重试发送3次

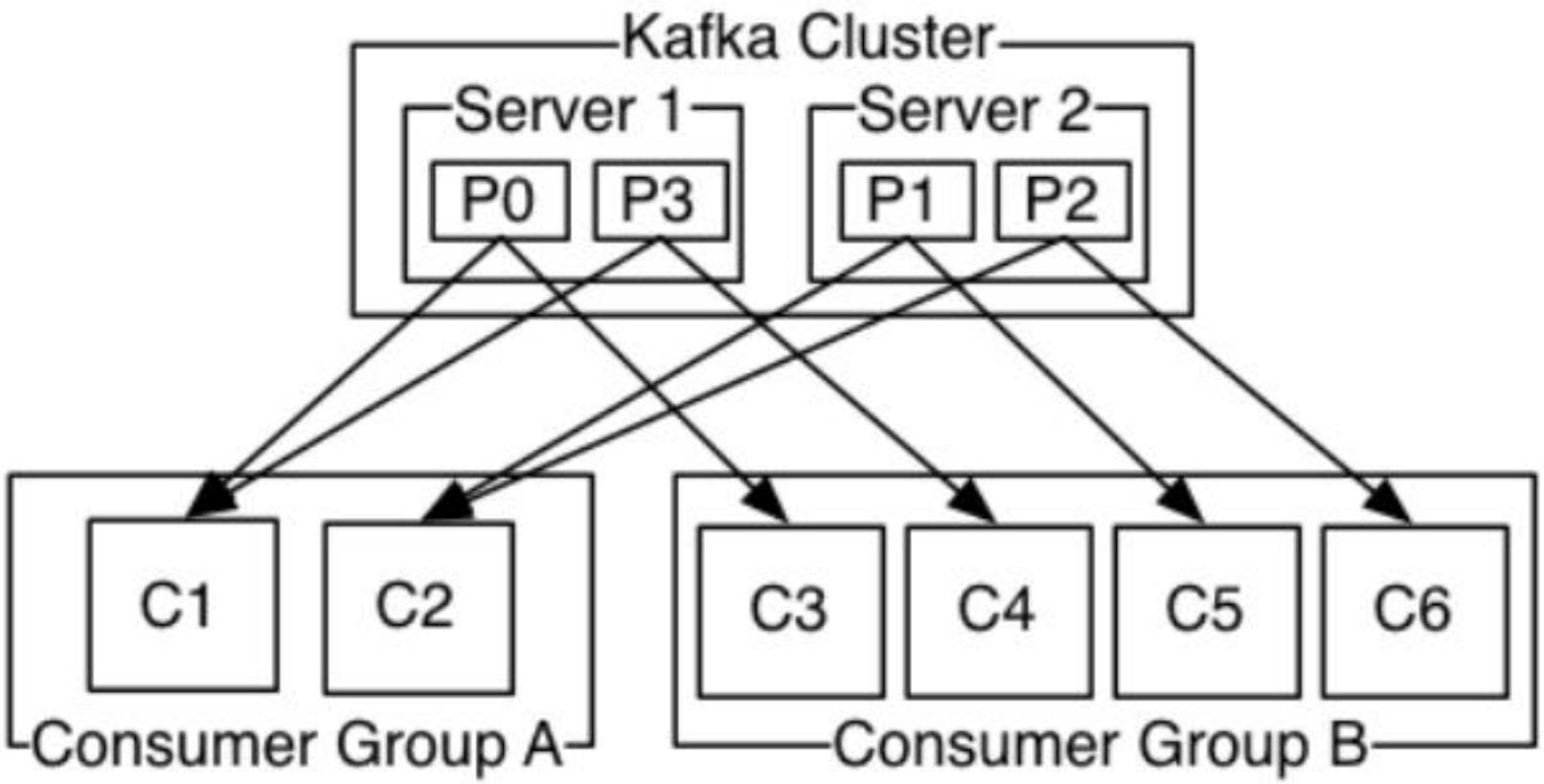
Kafka消息消费机制(一)

- Kafka有两种模式消费数据：**队列**和**发布订阅**；在队列模式下，一条数据只会发送给customer group中的一个customer进行消费；在发布订阅模式下，一条数据会发送给多个customer进行消费
- Kafka的Customer基于offset对kafka中的数据进行消费，对于一个customer group中的所有customer共享一个offset偏移量
- Kafka中通过控制Customer的参数{group.id}来决定kafka是什么数据消费模式，如果所有消费者的该参数值是相同的，那么此时的kafka就是类似于队列模式，数据只会发送到一个customer，此时类似于负载均衡；否则就是发布订阅模式

Kafka消息消费机制(二)

- Kafka的数据是按照分区进行排序的(插入的顺序)，也就是每个分区中的数据是有序的。在Consumer进行数据消费的时候，也是对分区的数据进行有序的消费的，但是不保证所有数据的有序性(多个分区之间)
- **Consumer Rebalance**：当一个consumer group组中的消费者数量和对应Topic的分区数量一致的时候，此时一个Consumer消费一个Partition的数据；如果不一致，那么可能出现一个Consumer消费多个Partition的数据或者不消费数据的情况，这个机制是根据Consumer和Partition的数量动态变化的
- Consumer通过**poll**的方式主动从Kafka集群中获取数据

Kafka消息消费机制(三)



Kafka Replication

- Kafka的**Replication**指的是Partition的复制，一个Partition的所有分区中只有一个分区是leader节点，其它分区是follower节点。
- Replication对Kafka的吞吐率有一定的影响，但是极大的增强了可用性
- Follower节点会定时的从leader节点上获取增量数据，一个活跃的follower节点必须满足一下两个条件：
  - 所有的节点必须维护和zookeeper的连接(通过zk的heartbeat实现)
  - follower必须能够及时的将leader上的writing复制过来，不能“落后太多”；“落后太多”由参数{replica.lag.time.max.ms}和{replica.lag.max.messages}决定

Kafka Leader Election

- Kafka提供了一个in-sync replicas(ISR)来确保Kafka的Leader选举，ISR是一个保存分区node的集合，如果一个node宕机了或数据“落后太多”，leader会将该node节点从ISR中移除，只有ISR中的follower节点才有可能成为leader节点
- Leader节点的切换基于Zookeeper的Watcher机制，当leader节点宕机的时候，其他ISR中的follower节点会竞争的在zk中创建一个文件目录(只会有一个follower节点创建成功)，创建成功的follower节点成为leader节点

Message Delivery Semantics

- **MessageDeliverySemantics**是消息系统中数据传输的可靠性保证的一个定义，主要分为三种类型：
  - At most once（最多一次）：消息可能会丢失，但是不可能重复发送
  - At least once（最少一次）：消息不可能丢失，但是可能重复发送
  - Exactly once（仅仅一次）：消息只发送一次，但不存在消息的丢失
- Kafka的Producer通过参数{request.required.acks}来定义确定Producer和Broker之间是那种消息传递类型
- Kafka的数据是分区存储的，每个分区中的数据是按照进入kafka的时间进行排序的，这样不需要为每条数据存储一个元数据(是否消费)，只需要为每个Consumer记录一个对应分区数据消费的最高标记位，Kafka中叫做“偏移量”(offset)

Why Kafka is Fast?

- 消息集(message set)
- 二进制传输
- 顺序读取磁盘
- “零”拷贝
- 端到端数据压缩

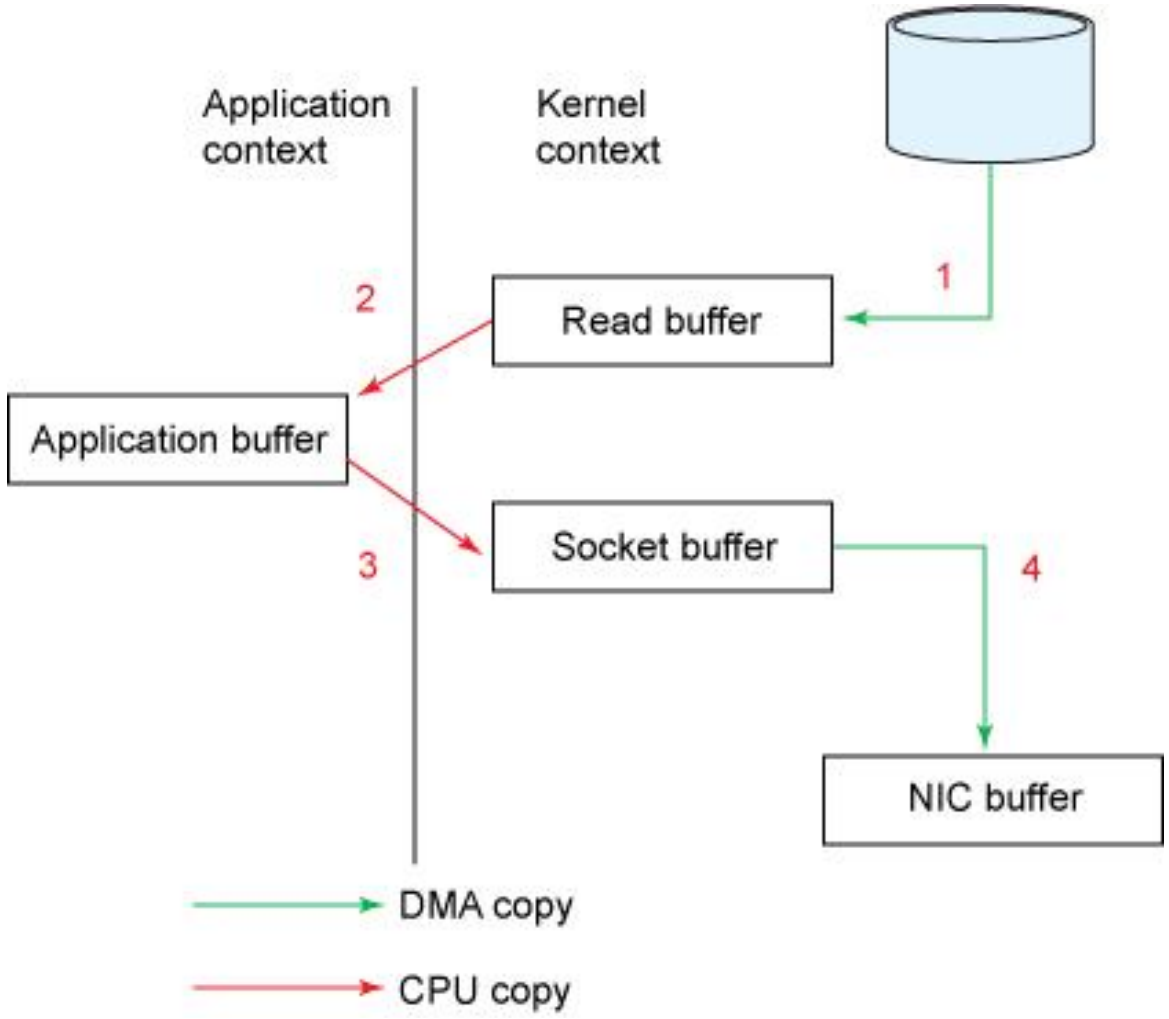
Why Kafka is Fast?

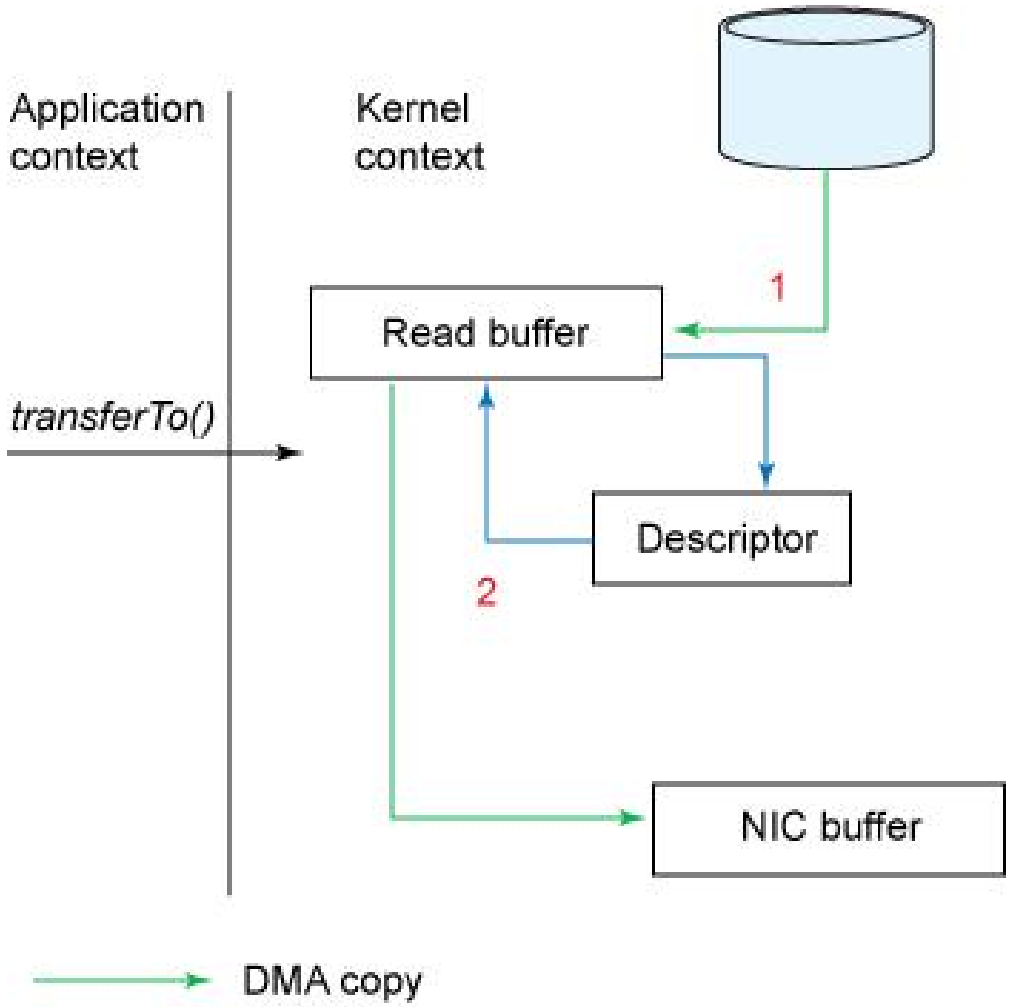
- **消息集(message set)**：Producer可以将多条消息一次发送给Kafka集群，Kafka可以一次将所有的数据追加到文件中，减少磁盘零碎的磁盘IO；同时consumer也可以一次性的请求一个数据集的数据
- **二进制传输**：同时消息在传递过程中是基于二进制进行传递的，不需要进行反序列化，在高负载的情况下，对性能是有一定的提升的
- **顺序读写磁盘**：Kafka的所有数据操作都是基于文件操作的，而操作文件的方式都是顺序读写，而顺序读写磁盘的速度会比随机读写快6000倍左右

Why Kafka is Fast?

- **“零”拷贝**：在Kafka服务中，数据发送到consumer的过程中采用的是“零拷贝”，比普通的读写文件方式减少了两次操作，速度能够提高50%
- **端到端数据压缩**：Producer可以需要发送的数据/数据集进行压缩后发送到Kafka集群，Kafka集群直接将数据保存到文件，然后Consumer消费数据的时候，将压缩后的数据获取到，进行解压缩操作。在性能瓶颈是网络带宽的情况下，非常有效。默认情况下，kafka支持gzip和snappy压缩

“零”拷贝


Application context
Kernel context
1
2
Read buffer
Application buffer
3
Socket buffer
4
NIC buffer
DMA copy
CPU copy


Application context
Kernel context
1
Read buffer
transferTo()
Descriptor
2
NIC buffer
DMA copy

Kafka API

- Kafka分别提供了基于Java和Scala的API，由于Kafka不仅仅只是在大数据中使用到，所以Kafka的JavaAPI应用的比较多。基于Maven进行Kafka的开发，Kafka Maven依赖如下：

```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka_2.10</artifactId>
    <version>${kafka.version}</version>
</dependency>
```

Kafka Producer涉及到的配置信息(一)

参数名称	默认参数值	备注
metadata.broker.list		指定kafka服务器监听的主机名和端口号列表，不同服务器之间使用“,”进行分割
request.required.acks	0	指定producer需要等待broker返回数据成功接收标识；0表示不等待，1表示等待一个broker返回结果，-1表示等待所有broker返回结果
request.timeout.ms	10000	当acks参数配置的时候，指定producer等待连接过期的时间毫米数
producer.type	sync	指定producer发送数据的方式是异步(async)还是同步(sync)
serializer.class	kafka.serializer.DefaultEncoder	指定producer发送数据的时候数据/消息编码器，即将消息转换为byte数组的编码器
key.serializer.class		指定producer发送数据的时候key类型的数据编码器，默认使用${serializer.class}给定的值
partitioner.class	kafka.producer.DefaultPartitioner	指定producer发送数据的数据分区器，默认采用hash进行数据分区操作；该参数的主要功能是：决定数据到底发送到那一个分区中

Kafka Producer涉及到的配置信息(二)

参数名称	默认参数值	备注
compression.codec	none	给定发送数据是否进行压缩设置，默认不进行压缩；参数可选：none、gzip、snappy
message.send.max.retries	3	指定数据发送失败，重试次数，默认3次
retry.backoff.ms	100	在数据重新发送过程中，producer会刷新topic的元数据信息(leader信息)，由于topic元数据的变化需要一点点时间，故该参数指定的值主要用于在producer刷新元数据之前的等待时间
topic.metadata.refresh.interval.ms	600000	给定producer中topic元数据周期性刷新的间隔时间，默认10分钟；当该参数给定的值为**负数**的时候，topic元数据的刷新只有在**发送数据失败**后进行刷新；当该参数给定为**0**的时候，**每次**发送数据后都进行元数据**刷新**(不推荐)；注意：元数据的刷新是在发送数据后触发的，如果永远不发送数据，那么元数据不会被刷新
queue.buffering.max.ms	5000	当数据传输方式是**async**(异步)的时候，指定数据在producer端停留的**最长时间**，该参数对于数据吞吐量有一定的影响，当时会增加数据的延迟性
queue.buffering.max.messages	10000	当数据传输方式为async(异步)的时候，指定producer端最多允许临时保存的**最大数据量**，当数据量超过该值的时候，发送一次数据

Kafka Producer涉及到的配置信息(三)

参数名称	默认参数值	备注
queue.enqueue.timeout.ms	-1	当数据发送方式为**async**(异步)，而且等待队列数据填充满的时候{**queue.buffering.max.messages**}，一条新的数据过来，最大阻塞时间；设置为**0**表示，**不阻塞**，当队列满的时候，直接将新数据删除(不发送)；当设置为**正数**的时候，表示等待给定毫秒数后，进行重试操作，失败则数据删除(不发送)；设置为**-1**表示**一直等待**，直到队列允许添加数据
batch.num.messages	200	当数据发送方式为**async**(异步)的时候，producer一个批次发送的数据条数；当producer中的数据量达到该参数${**batch.num.messages**}的设置值或者数据停留时间超过参数${**queue.buffering.max.ms**}的时候，触发producer发送数据的动作(实际发送数据量可能不超过该参数值)
send.buffer.bytes	102400	指定producer端数据缓存区大小，默认值为：10KB

Kafka Producer开发参考页面：

http://kafka.apache.org/082/documentation.html#producerapi
http://kafka.apache.org/081/documentation.html#producerconfigs
http://kafka.apache.org/081/documentation.html#apidesign
http://kafka.apache.org/081/documentation.html#producerapi

Kafka Producer API

- Kafka的Producer API主要提供下列三个方法：
  - public void send(KeyedMessage<K,V> message)

    发送单条数据到Kafka集群
  - public void send(List<KeyedMessage<K,V>> messages)

    发送多条数据(数据集)到Kafka集群
  - public void close()

    关闭Kafka连接资源
- 案例：使用Java语言实现一个Kafka Producer程序并测试

Kafka Consumer涉及到的配置信息(一)

参数名称	默认参数值	备注
group.id		Consumer的group id值，如果多个Consumer的group id的值一样，那么表示这多个Consumer属于同一个group组
zookeeper.connect		Kafka元数据Zookeeper存储的url，和配置文件中的参数一样
consumer.id		消费者id字符串，如果不给定的话，默认自动产生一个随机id
socket.timeout.ms	30000	Consumer连接超时时间，实际超时时间是socket.timeout.ms + max.fetch.wait
socket.receive.buffer.bytes	65536	接收数据的缓冲区大小，默认64kb
fetch.message.max.bytes	1048576	指定每个分区每次获取数据的最大字节数，一般该参数要求比message允许的最大字节数要大，否则可能出现producer产生的数据consumer没法消费
num.consumer.fetchers	1	Consumer获取数据的线程数量
auto.commit.enable	true	是否自动提交offset偏移量，默认为true(自动提交)
auto.commit.interval.ms	60000	自动提交offset偏移量的间隔时间

Kafka Consumer涉及到的配置信息(二)

参数名称	默认参数值	备注
rebalance.max.retries	4	当一个新的Consumer添加到Consumer Group的时候，会触发数据消费的rebalance操作；rebalance操作可能会失败，该参数的主要作用是设置rebalance的最大重试次数
fetch.min.bytes	1	一个请求最少返回记录大小，当一个请求中的返回数据大小达到该参数的设置值后，记录数据返回到consumer中
fetch.wait.max.ms	100	一个请求等待数据返回的最大停留时间
rebalance.backoff.ms	2000	rebalance重试过程中的间隔时间
auto.offset.reset	largest	指定consumer消费kafka数据的时候offset初始值是啥，可选参数：largest和smallest；smallest指该consumer的消费offset是当前kafka数据中的最小偏移量；largest指该consumer的消费offset是当前kafka数据中的最大偏移量
consumer.timeout.ms	-1	给定当consumer多久时间没有消费数据后，抛出异常；-1表示不抛出异常
zookeeper.session.timeout.ms	6000	zk会话时间
zookeeper.connection.timeout.ms	6000	连接zk过期时间

Kafka Consumer API

- Kafka提供了两种Consumer API，分别是：**High Level Consumer API** 和 **Lower Level Consumer API**(**Simple Consumer API**)
- High Level Consumer API：将底层具体获取数据、更新offset、设置偏移量等操作屏蔽掉，直接操作数据流的处理工作。优点是：操作简单；缺点：可操作性太差，无法按照自己的业务场景选择处理方式。(类：ConsumerConnector)
- Lower Level Consumer API：通过直接操作底层API获取数据的方式获取Kafka中的数据，需要自行给定分区、偏移量等属性。优点：可操作性强；缺点：代码相对而言比较复杂。(类：SimpleConsumer)
- 案例：使用Java语言开发一个Kafka Consumer程序并测试

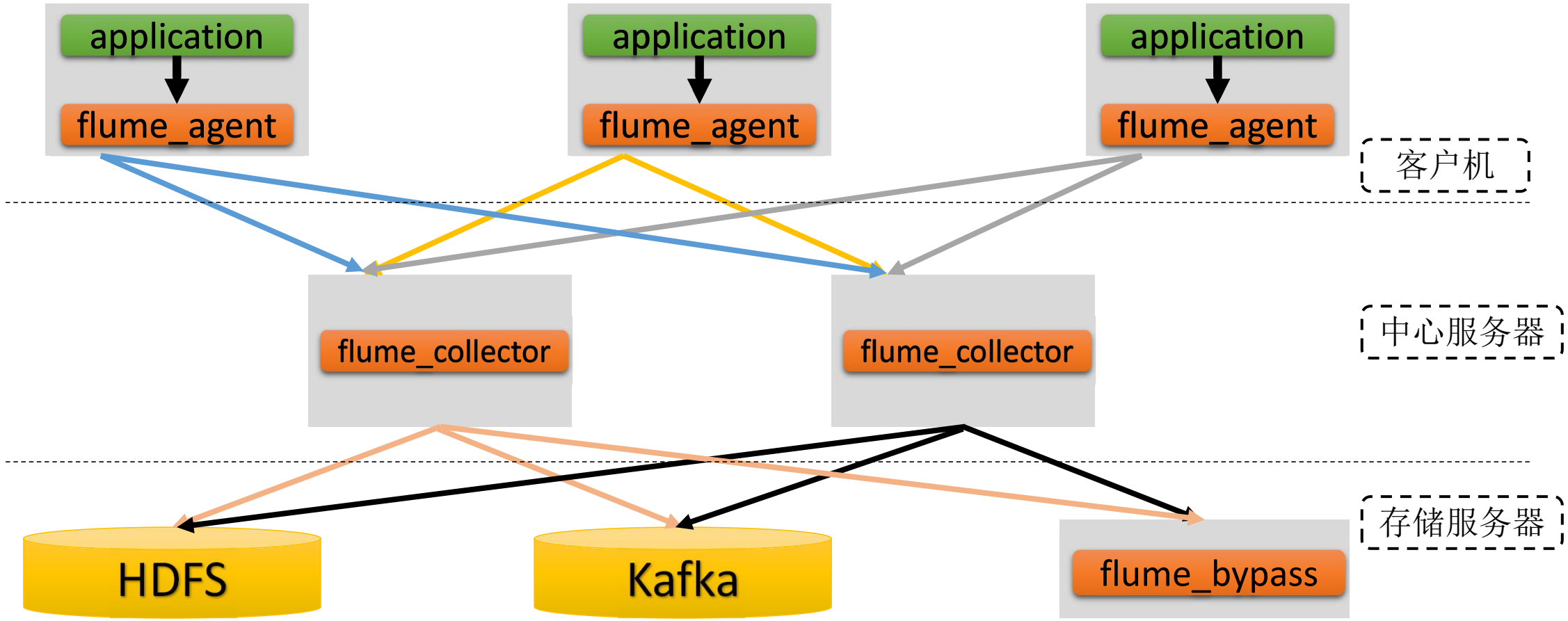
Kafka和Flume集成(一)

- 区别：
  - Flume(Apache日志收集系统)，主要功能就是收集同步数据源的数据，并将数据保存到持久化系统中，适合数据来源比较广，数据收集结构比较固定的场景
  - Kafka(Apache分布式消息系统)，主要是作为一个中间件系统的方式存在，适合高吞吐量和负载的情况，可以作为业务系统中的缓存、消息通知系统、数据收集等场景
- 为什么还需要Flume作为日志收集系统？
  - 主要原因：Kafka需要进行一些额外的开发，Flume可以直接使用Sink将数据保存
  - 次要原因：对于不需要高并发的业务场景，Flume足够使用，而且Flume对于机器的性能要求低于Kafka的。

Kafka和Flume集成(二)

- 案例一：Flume收集文本文件中的数据，并将数据发送到Kafka(公司常用)
  - 在Flume中使用KafkaSink将数据发送到Kafka中
- 案例二：Flume收集Kafka中的数据，并将数据日志打印到控制台
  - 在Flume中使用KafkaSource收集Kafka中的数据

Kafka和Flume集成(三)



Kafka与Log4j集成(一)

- 很多应用中会使用Log4j记录日志，如何将Log4j的日志输入到Kafka中，一般情况下有两种实现方式：
  - 方式一：使用Flume等日志收集工具将Log4j的日志上传到Kafka中；这种方式比较慢，而且依赖于其他日志收集组件
  - 方式二：直接在Log4j中将日志数据直接通过Kafka的日志Appender传输到Kafka的对应Topic中；这种方式比较快捷，只需要配置项目的log4j.properties文件即可达到日志收集的功能

Kafka与Log4j集成(二)

```
log4j.rootLogger=INFO,console,KAFKA

## appender KAFKA
log4j.appender.KAFKA=kafka.producer.KafkaLog4jAppender
log4j.appender.KAFKA.topic=test
log4j.appender.KAFKA.brokerList=hadoop-senior01.ibeifeng.com:9092
log4j.appender.KAFKA.compressionType=none
log4j.appender.KAFKA.syncSend=true
log4j.appender.KAFKA.layout=org.apache.log4j.PatternLayout
log4j.appender.KAFKA.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}
 %-5p %c{1}:%L %% - %m%n
```

Kafka集群管理&监控

- 在实际生产环境中，需要对Kafka集群进行管理和监控，方便了解当前Kafka中的数据情况，比如：数据的产生量以及消费量等；Kafka提供了基于JMX的元数据管理接口，所以可以基于Kafka的JMX接口进行定制化的Kafka Web监控管理项目的开发；另外市面上有一些以及实现了Kafka集群管理的框架或者软件，常用监控软件如下：
  - Kafka Web Console：监控信息比较全面，但是存在Bug；源码：https://github.com/claudemamo/kafka-web-console
  - Kafka Manager：雅虎开源，偏向Kafka集群管理，可以实现监控；源码：https://github.com/yahoo/kafka-manager
  - KafkaOffsetMonitor：以jar文件的形式运行，部署简单，只有监控功能；源码：https://github.com/quantifind/KafkaOffsetMonitor

THANK YOU

上海育创网络科技有限公司