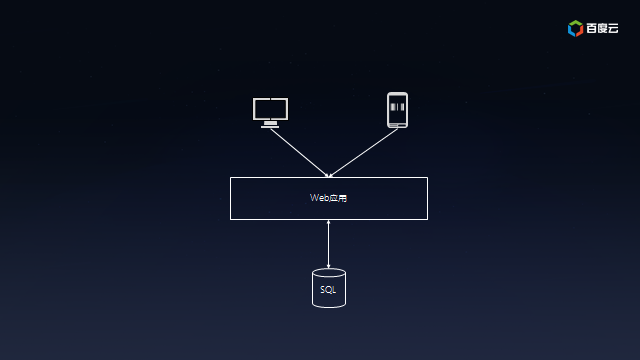
Kafka背景

假设你意气风发，要开发新一代的互联网应用，以期在互联网事业中一展宏图。借助云计算，很容易开发出如下原型系统：

1. Web应用：部署在云服务器上，为个人电脑或者移动用户提供的访问体验。
2. SQL数据库：为Web应用提供数据持久化以及数据查询。



这套架构简洁而高效，很快便能够部署到百度云等云计算平台，以便快速推向市场。互联网不就是讲究小步快跑嘛！

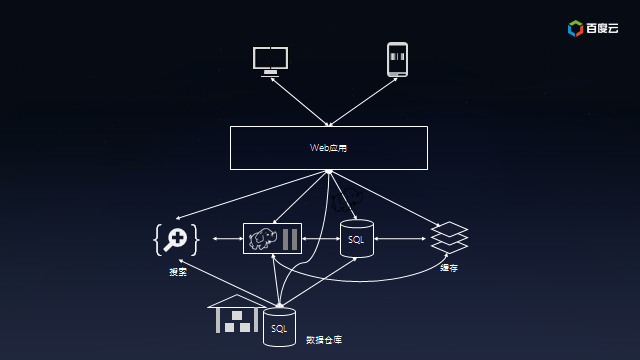
好景不长。

随着用户的迅速增长，所有的访问都直接通过SQL数据库使得它不堪重负，不得不加上缓存服务以降低SQL数据库的荷载；

为了理解用户行为，开始收集日志并保存到Hadoop上离线处理，同时把日志放在全文检索系统中以便快速定位问题；

由于需要给投资方看业务状况，也需要把数据汇总到数据仓库中以便提供交互式报表。

此时的系统的架构已经盘根错节了，考虑将来还会加入实时模块以及外部数据交互，真是痛并快乐着……

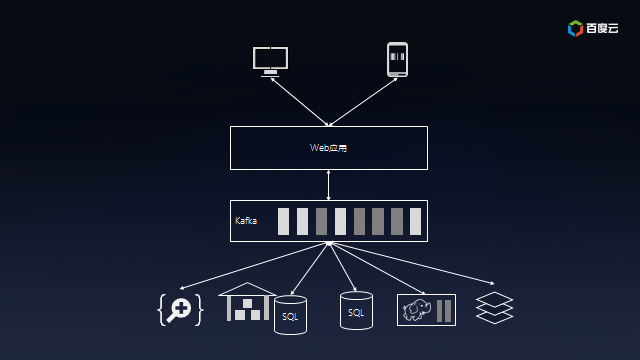


这时候，应该跑慢一些，让灵魂跟上来。

本质上，这是一个数据集成问题。没有任何一个系统能够解决所有的事情，所以业务数据根据不同用途存而放在不同的系统，比如归档、分析、搜索、缓存等。数据冗余本身没有任何问题，但是不同系统之间像意大利面条一样复杂的数据同步却是挑战。

这时候就轮到Kafka出场了。

Kafka可以让合适的数据以合适的形式出现在合适的地方。Kafka的做法是提供消息队列，让生产者单往队列的末尾添加数据，让多个消费者从队列里面依次读取数据然后自行处理。之前连接的复杂度是O(N^2)，而现在降低到O(N)，扩展起来方便多了：



在Kafka的帮助下，你的互联网应用终于能够支撑飞速增长的业务，成为下一个BAT指日可待。

Kafka简介

Kafka最早由LinkedIn开发，如今已经成为Apache基金会顶级项目，被Walmart、Netflix、PayPal、Uber、eBay等采用。

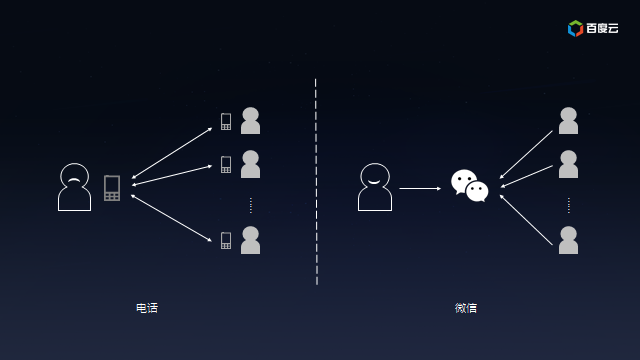
在国内互联网圈子中Kafka也是大数据圈子中必备的社交货币，茶余饭后指点江山说起卡夫卡，联想到Franz Kafka是文青，联想到Apache Kafka的才是大数据工程师。

本质上Kafka是分布式的流数据平台，因为以下特性而著名：

1. 提供Pub/Sub方式的海量消息处理。
2. 以高容错的方式存储海量数据流。
3. 保证数据流的顺序。

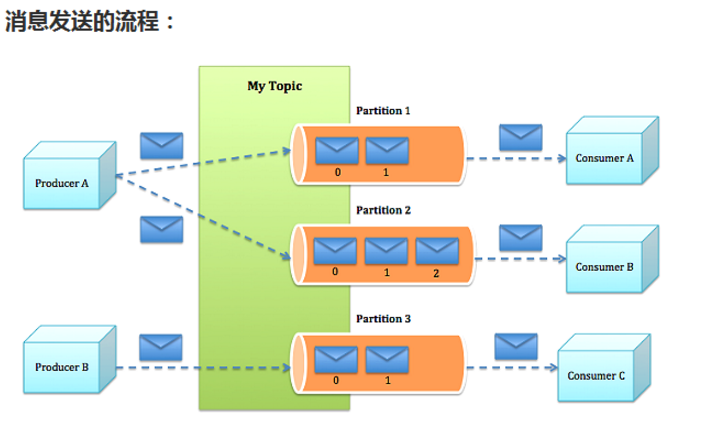
说起Pub/Sub，熟悉企业应用集成（Enterprise Application Integration，EAI）的朋友不会陌生，它是一种处理消息的范式，消息的发布者（Pub）只需要指定消息的类别，而不需要与订阅者（Sub）打交道。订阅者对一个或多个类别表达兴趣，于是只接收感兴趣的消息，而不需要知道什么样的发布者发布的消息。这种发布者和订阅者的解耦可以给应用带来更好的可扩展性。

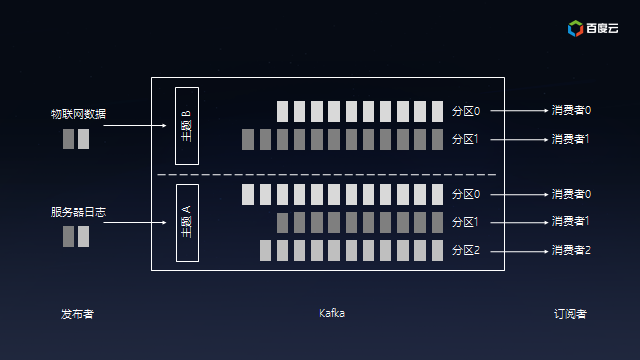
打个比方，你的公司业务蓬勃发展，前后开发了多个互联网应用都需要做市场推广。一种方法是通过电话向客户推广产品新特性，不但需要找到每个互联网应用所对应的客户名单，还要挨个电话联系，可是，客户不一定有空接电话或者已经在通讯中了，长长的客户名单也会让你头疼不已；另一种方法是为每个互联网应用创建一个微信公众号（用公众号分割推广信息），让客户订阅后推送产品新特性的信息（客户不用关心到底谁发的），客户有空的时候看一下（客户不用立等答复你），信息量太大的话就再加个同事订阅（水平扩展客户处理能力），有机会还可以介绍给其他潜在用户订阅（你也不必特意通知新客户）。很明显，电话这种同步消息交换的方式很容易产生瓶颈，而微信公众号这类异步消息交换的方式客户再多也不用担心。



Kafka提供的Pub/Sub就是典型的异步消息交换，用户可以为服务器日志或者物联网设备创建不同主题（Topic），之后数据可以源源不断地发送到各个主题，后端数据仓库、流式分析或者全文检索等对接特定主题，服务器或者物联网设备是无需关心的。

同时，Kafka可以将主题划分为多个分区（Partition），会根据分区规则选择把消息存储到哪个分区中，只要如果分区规则设置的合理，那么所有的消息将会被均匀的分布到不同的分区中，这样就实现了负载均衡和水平扩展。另外，多个订阅者可以从一个或者多个分区中同时消费数据，以支撑海量数据处理能力：

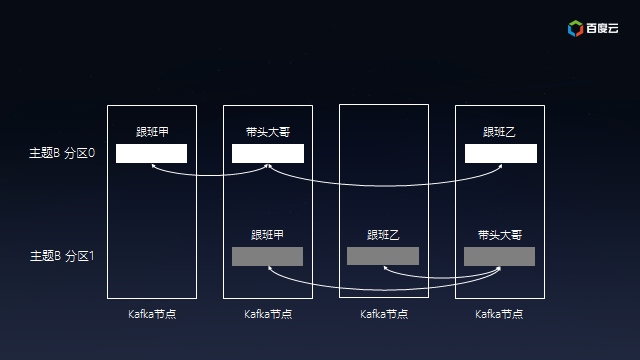




Kafka的设计也是源自生活，好比是为公路运输，不同的起始点和目的地需要修不同高速公路（主题），高速公路上可以提供多条车道（分区），流量大的公路多修几条车道保证畅通，流量小的公路少修几条车道避免浪费。收费站好比消费者，车多的时候多开几个一起收费避免堵在路上，车少的时候开几个让汽车并道就好了，嗯……

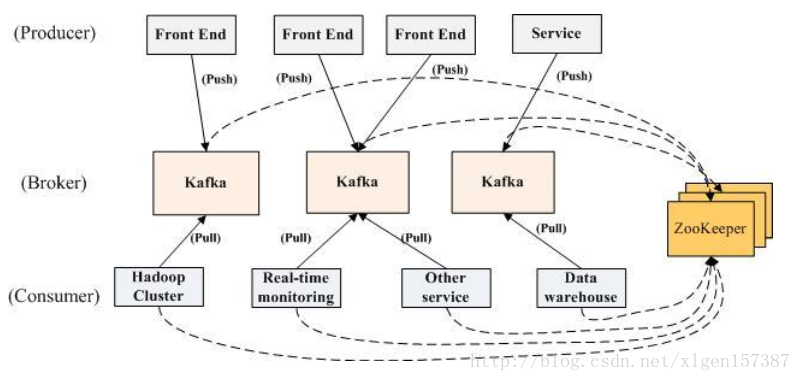
顺便说一句，由于**消息是以追加到分区中的，多个分区顺序写磁盘的总效率要比随机写内存还要高（引用Apache Kafka – A High Throughput Distributed Messaging System的观点），是Kafka高吞吐率的重要保证之一。**

为了保证数据的可靠性，Kafka会给每个分区找一个节点当带头大哥（Leader），以及若干个节点当随从（Follower）。消息写入分区时，带头大哥除了自己复制一份外还会复制到多个随从。如果随从挂了，Kafka会再找一个随从从带头大哥那里同步历史消息；如果带头大哥挂了，随从中会选举出新一任的带头大哥，继续笑傲江湖。



最后，每个发布者发送到Kafka分区中的消息是确保顺序的，订阅者可以依赖这个承诺进行后续处理。

Kafka工作图



Kafka设计

A吞吐量

高吞吐是kafka需要实现的核心目标之一，为此kafka做了以下一些设计：

1. 数据磁盘持久化：

消息不在内存中cache，直接写入到磁盘，充分利用磁盘的顺序读写性能

直接使用linux 文件系统的cache，来高效缓存数据。

1. zero-copy：减少IO操作步骤

传统的数据发送需要发送4次上下文切换，采用sendfile系统调用之后，数据直接在内核态交换，系统上下文切换减少 为2次。根据测试结果，可以提高60%的数据发送性能。Zero-Copy详细的技术细节可以参考：https://www.ibm.com /developerworks/linux/library/j-zerocopy/

1. 数据批量发送
2. 数据压缩
3. Topic划分为多个partition，提高parallelism
4. 显示分布式

即所有的producer、broker和consumer都会有多个，均为分布式的。Producer和broker之间没有负 载均衡机制。broker和consumer之间利用zookeeper进行负载均衡。所有broker和consumer都会在zookeeper中进 行注册，且zookeeper会保存他们的一些元数据信息。如果某个broker和consumer发生了变化，所有其他的broker和 consumer都会得到通知。

B：负载均衡

1. producer根据用户指定的算法，将消息发送到指定的partition
2. 存在多个partiiton，每个partition有自己的replica，每个replica分布在不同的Broker节点上
3. 多个partition需要选取出lead partition，lead partition负责读写，并由zookeeper负责fail over
4. 通过zookeeper管理broker与consumer的动态加入与离开

Kafka应用场景

## 1.消息队列

比起大多数的消息系统来说，Kafka有更好的吞吐量，内置的分区，冗余及容错性，这让Kafka成为了一个很好的大规模消息处理应用的解决方案。 消息系统一般吞吐量相对较低，但是需要更小的端到端延时，并尝尝依赖于Kafka提供的强大的持久性保障。在这个领域，Kafka足以媲美传统消息系统， 如ActiveMR或RabbitMQ。

## 2.行为跟踪

Kafka的另一个应用场景是跟踪用户浏览页面、搜索及其他行为，以发布-订阅的模式实时记录到对应的topic里。那么这些结果被订阅者拿到后，就可以做进一步的实时处理，或实时监控，或放到hadoop/离线数据仓库里处理。

## 3.元信息监控

作为操作记录的监控模块来使用，即汇集记录一些操作信息，可以理解为运维性质的数据监控吧。

## 4.日志收集

日志收集方面，其实开源产品有很多，包括Scribe、Apache Flume。很多人使用Kafka代替日志聚合（log aggregation）。日志聚合一般来说是从服务器上收集日志文件，然后放到一个集中的位置（文件服务器或HDFS）进行处理。然而Kafka忽略掉 文件的细节，将其更清晰地抽象成一个个日志或事件的消息流。这就让Kafka处理过程延迟更低，更容易支持多数据源和分布式数据处理。比起以日志为中心的 系统比如Scribe或者Flume来说，Kafka提供同样高效的性能和因为复制导致的更高的耐用性保证，以及更低的端到端延迟。

## 5.流处理

这个场景可能比较多，也很好理解。保存收集流数据，以提供之后对接的Storm或其他流式计算框架进行处理。很多用户会将那些从原始topic来的 数据进行阶段性处理，汇总，扩充或者以其他的方式转换到新的topic下再继续后面的处理。例如一个文章推荐的处理流程，可能是先从RSS数据源中抓取文 章的内容，然后将其丢入一个叫做“文章”的topic中；后续操作可能是需要对这个内容进行清理，比如回复正常数据或者删除重复数据，最后再将内容匹配的 结果返还给用户。这就在一个独立的topic之外，产生了一系列的实时数据处理的流程。Strom和Samza是非常著名的实现这种类型数据转换的框架。

## 6.事件源

事件源是一种应用程序设计的方式，该方式的状态转移被记录为按时间顺序排序的记录序列。Kafka可以存储大量的日志数据，这使得它成为一个对这种方式的应用来说绝佳的后台。比如动态汇总（News feed）。

## 7.持久性日志（commit log）

Kafka可以为一种外部的持久性日志的分布式系统提供服务。这种日志可以在节点间备份数据，并为故障节点数据回复提供一种重新同步的机制。Kafka中日志压缩功能为这种用法提供了条件。在这种用法中，Kafka类似于Apache BookKeeper项目。

百度Kafka

Kafka优点种种，但是要把Kafka用好并不容易。开源免费是好事情，但是如何能够保证24×7的运维保障业务稳定运行是个大问题。同时，初期业务量小的时候，闲置Kafka集群又能造成很大的浪费。

针对以上问题，百度云天算大数据平台推出了百度Kafka服务。大体上，百度Kafka是社区版本的多租户全托管服务，与自行运维Kafka集群相比，有以下增强：

1. 开箱即用：可以直接创建主题并使用Kafka服务，专注业务而不用花费精力去安装、部署、配置、调试和维护集群。
2. 低廉价格：只需为使用的资源而不是虚拟主机付费，同时支持动态扩容。
3. 数据安全：支持SSL加密，保证数据在传输的过程中不被窃听或者篡改。
4. 可靠耐用：独特的服务高可用性以及数据高可靠性设计。

秉承开源开放的宗旨，百度Kafka与社区的Kafka高度兼容，迁移成本极低且不用担心被供应商绑定。

Kafka部署

## 1：下载：

curl -L -O http://mirrors.cnnic.cn/apache/kafka/0.9.0.0/kafka\_2.10-0.9.0.0.tgz

## 2：解压：

tar zxvf kafka\_2.10-0.9.0.0.tgz

目录说明

* /bin 操作kafka的可执行脚本，还包含windows下脚本
* /config 配置文件所在目录
* /libs 依赖库目录
* /logs 日志数据目录，目录kafka把server端日志分为5种类型，分为:server,request,state，log-cleaner，controller

## 3：配置zookeeper

编辑kafka的配置文件config/server.properties

kafka最为重要三个配置依次为：broker.id、log.dir、zookeeper.connect

配置文件说明：<https://blog.csdn.net/lizhitao/article/details/25667831>

## 4：启动zookeeper

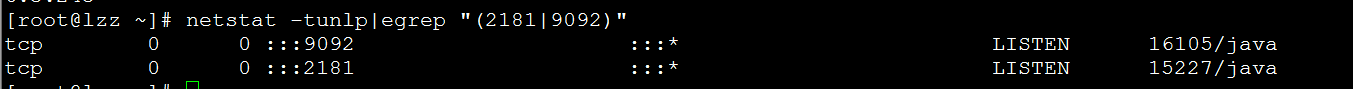
进入zookeeper的bin目录：./zkServer.sh start &

## 5：启动kafka

bin/kafka-server-start.sh config/server.properties &

**注意：启动命令后面带参数config/server.properties，只是启动了一个kafka节点。**

检测2181与9092端口



Kafka占用的端口为9092，进程id为16105

Zookeeper占用的端口为2181，进程id为15227

## 6：创建并查看主题topic

**创建topic：**

bin/kafka-topics.sh --create --zookeeper 192.168.0.7:2181 --replication-factor 1 --partition 1 --topic test

这条命令会创建一个名为test的topic，有1个分区，每个分区需分配1个副本。

Kafka controller部分主要做下面这些事情：1. 创建分区；2. 创建副本；3. 为每个分区选举leader、ISR；4.更新各种缓存

**查看topic：**

bin/kafka-topics.sh --list --zookeeper localhost:2181

## 7：单机连通性测试

A：启动生产者

bin/kafka-console-producer.sh --broker-list 192.168.0.7:9092 --topic test

启动后，随便输入一个字符串。

B：启动消费者

bin/kafka-console-consumer.sh --zookeeper 192.168.0.7:2181 --topic test --from-beginning

就可以看到刚才生产者输入的信息

**注意：**

producer，指定的Socket(192.168.0.7:+9092),生产者的消息发往kafka，也即是broker

consumer, 指定的Socket(192.168.0.7:+2181),消费者的消息来自zookeeper（协调转发）

## 8：搭建集群(伪集群)

刚才只是启动了单个broker，现在启动有3个broker组成的集群，这些broker节点也都是在本机上。

### （1）为每个broker提供配置文件

config/server0.properties

config/server1.properties

config/server2.properties

broker.id为集群中唯一的标注一个节点，因为在同一个机器上，所以必须指定不同的端口和日志文件，避免数据被覆盖。

注意：配置项broker.id、port、listeners要区别开

broker.id=0

port=9092

listeners=PLAINTEXT://:9092

broker.id=1

port=9093

listeners=PLAINTEXT://:9093

broker.id=2

port=9094

listeners=PLAINTEXT://:9094

### （2）启动多个broker

bin/kafka-server-start.sh config/server0.properties & #启动broker0

bin/kafka-server-start.sh config/server1.properties & #启动broker1

bin/kafka-server-start.sh config/server2.properties & #启动broker2

### （3）发送接收消息

bin/kafka-console-producer.sh --topic topic\_1 --broker-list 192.168.1.181:9092,192.168.1.181:9093,192.168.1.181:9094

bin/kafka-console-consumer.sh --topic topic\_1 --zookeeper 192.168.1.181:2181 --from-beginning

此时producer将topic发布到了3个broker中，现在就有点分布式的概念了。

Kafka监控

为了简化开发者和服务工程师维护Kafka集群的工作，yahoo构建了一个叫做Kafka管理器的基于Web工具，叫做 Kafka Manager。这个管理工具可以很容易地发现分布在集群中的哪些topic分布不均匀，或者是分区在整个集群分布不均匀的的情况。它支持管理多个集群、选择副本、副本重新分配以及创建Topic。同时，这个管理工具也是一个非常好的可以快速浏览这个集群的工具，有如下功能：

1. 管理多个kafka集群
2. 便捷的检查kafka集群状态(topics,brokers,备份分布情况,分区分布情况)
3. 选择你要运行的副本
4. 基于当前分区状况进行
5. 可以选择topic配置并创建topic(0.8.1.1和0.8.2的配置不同)
6. 删除topic(只支持0.8.2以上的版本并且要在broker配置中设置delete.topic.enable=true)
7. Topic list会指明哪些topic被删除（在0.8.2以上版本适用）
8. 为已存在的topic增加分区
9. 为已存在的topic更新配置
10. 在多个topic上批量重分区
11. 在多个topic上批量重分区(可选partition broker位置)