美团网
meituan.com

Kafka & Mafka技术分享及讨论

李志涛
基础架构组

美团网
meituan.com

1 Kafka设计关键点

2 Mafka设计与实现

3 Mafka & Kafka不同

为什么设计Kafka

美团网 meituan.com

- 面临问题
  - 上游每天产生海量日志数据(访问日志，投票、评分)需要分析和处理
  - 只需要简单pub/sub模型，用于一对多消费、服务解耦等场景
  - 传统消息系统吞吐量低、性能差、堆积能力弱无法承载海量数据
- 设计目标
  - 分布式设计
  - pub/sub模型
  - 集群无限扩展，保证海量消息堆积能力
  - 高可用
  - 可扩展
  - 高吞吐量，在高峰时段可以支持每秒数以百万计的消息传递

发展历程

美团网 meituan.com

- Kafka-0.6.x
  - 只是开发出分布式日志/消息系统雏形，功能不完善，仅能适合对数据可靠性要求不高，日志收集的场景(例如：海量日志数据收集、分析)
- Kafka-0.7.x
  - 新增feature(基于时间清除历史数据、按照时间滚动分段文件、日志分类)
  - bugfix & Improvement
- Kafka-0.8.x
  - 具有里程碑意义的一个版本，加入多副本机制容灾，再也不只是适合日志收集的场景了，Kafka在可靠性方面做了很多改进，已经可以用在企业级应用。
- Kafka-0.9.x
  - 新增安全性模块，客户端访问读写权限控制
  - 重构Connect模块
  - 重新设计Consumer

应用领域

美团网 meituan.com

- Kafka + Flume + Storm在线计算
- flume + Kafka + hdfs用于MR离线计算
- Kafka + Spark用于数据挖掘和机器学习
- Apache Kylin(Hadoop + Spark + Kafka + HBase)大数据OLAP引擎
- Kafka纯消息系统
- …

美团网
meituan.com

基于Kafka-0.8.2.1版本分享

本次分享客户端基于Scala语言版本

关键特色

美团网 meituan.com

- 高吞吐、高性能
  - 比传统的MQ快得多，100W+QPS
- 分布式设计 & 可扩展性好
  - 能够对消息进行数据分片，简称分区
  - 基于分区实现数据迁移
- 高容量
  - 使用硬盘替代内存存储
  - 多目录支持(可以同时挂载多个硬盘)
- 读写负载均衡
  - 读写负载均衡(压力分散到集群中每个Broker上)

关键角色

- Producer:向Kafka发布消息的实例
- Consumer:从Kafka中订阅Topic的实例
- Broker：Kafka集群中每个实例称作Broker，由id唯一标示，负责消息存储、转发
- Controller：每个集群中会选举一个Broker作为Controller，它负责执行分区、副本分配、replica leader选举，调度数据复制和迁移。

关键术语

- Topic：Kafka维护消息的种类，每一类消息由Topic标识
- Partition：对Topic中消息水平切分，至少1 partition/每个topic，Partition内消息有序，多个Parition消息无序
- Consumer Group：同一个Consumer Group中的Consumers,Kafka将相应Topic中的一条消息只能被一个Consumer消费，用多个Consumer Group实现多播，一条消息被多个Consumer Group消费
- Replica：将Partition复制，每一份叫做一个Replica
- **Replica Leader**：每一个Partition都有一个“Leader”负责Partition上所有的读写操作
- **Replica Follower**：每一个Partition都有0个或多个 Follower，只负责同步Leader的数据

关键术语

- **LogEndOffset**：表示每个Replica的log最后一条Message，有可能为脏数据(在page cache中)
- **Isr**：全称In-Sync Replicas，是Replicas的一个子集，由leader维护isr列表，Follower从Leader同步数据有一些延迟(包括了延迟时间和延迟条数两个维度)，任意一个超过阈值都会把该Follower踢出Isr。
- **Osr**：全称OutOf-Sync Replicas，新Follower或从Isr列表中剔除放入Osr列表中
- **Replicas**：Isr + Osr
- minIsr：如果isr.size小于minIsr.size写入不可用，目的是保证replicas数据一致性，牺牲可用性。
- **highWatermark**：每个replica都有highWatermark，leader和follower各自负责更新自己的highWatermark状态。

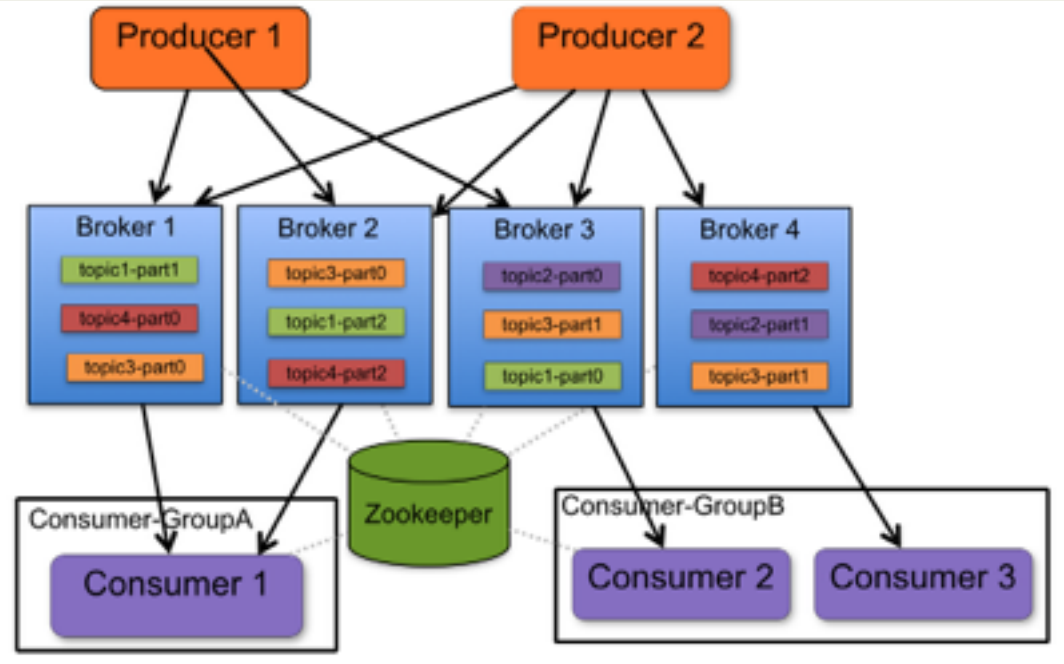
highWatermark的作用

- 每当follower从leader同步数据时
  - **leader更新Isr中highWatermark**，Isr中最小logEndOffset设置为highWatermark
  - **follower更新自身的highWatermark**，follower logEndOffset与Isr中highWatermark比较取最小值为follower的highWatermark
  - **当Osr中follower追赶上leader**，条件达到follower的logEndOffset >= leaderHighWatermark，follower会加入或重新加入Isr列表。

highWatermark的作用

- 作用
  - highWatermark > Consumer fetchOffset， Consumer才能消费到消息。
  - 决定follower是否加入Isr。

Kafka物理架构



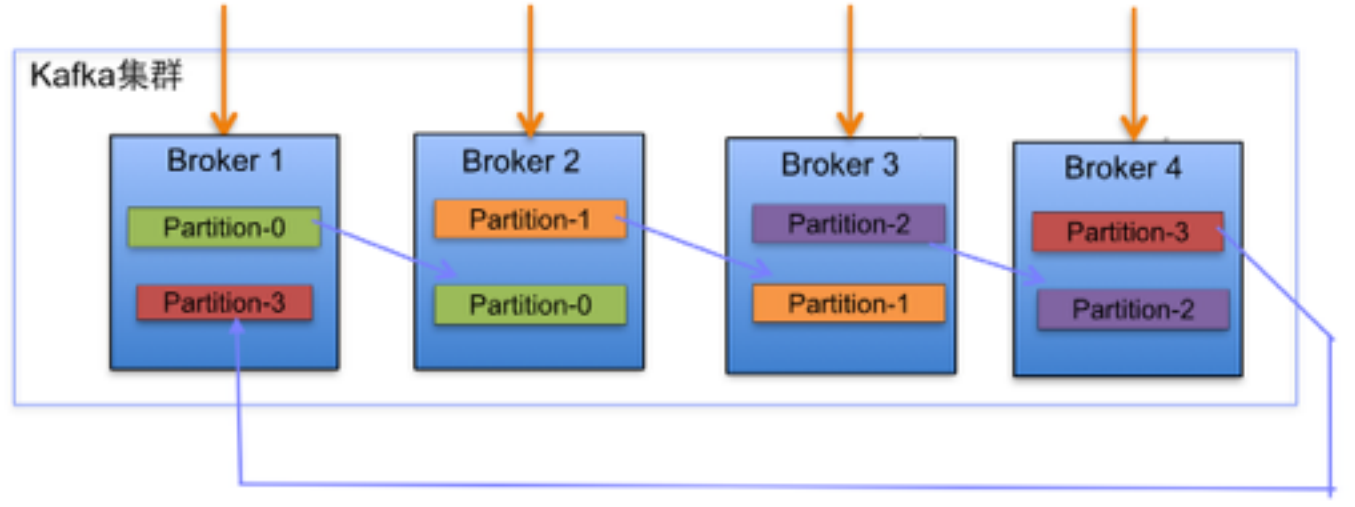
- zk为Kafka提供了稳定的服务和failover机制，存储元数据，订阅关系等。
- Broker为无状态，Partitions均衡散列到多个Broker上，集群能在多个Broker中fail-over和load-balance，每个集群中会选举出一个Broker来担任Controller，负责Partitions中replicas的Leader选举，调度replicas的数据迁移和复制。

Kafka物理架构-续

美团网 meituan.com

- Producer
  - 并不依赖zk
  - 启动前硬编码方式配置一个Brokers列表
  - 只有发送消息时，才会与目标Paritions所在Broker建立连接
  - 多个Partitions分布在一个Broker连接复用
  - 随机选择一个Broker建立连接，定期更新指定Topic元数据
- Consumer
  - high level api
  - low level api

Partition和Replica分布策略



如图左边，以一个Kafka集群中4个Broker举例，创建1个topic包含4个Partition，2 Replica

随机分配一个Broker作为起点

- Partitions和Replicas分布算法如下：
  - 将所有N Broker和待分配的i个Partition排序.
  - 将第i个Partition分配到第(i mod n)个Broker上.
  - 将第i个Partition的第j个Replica分配到第((i + j) mod n)个Broker上.

高吞吐量考虑

美团网 meituan.com

- Producer
  - 同步批量发送消息
  - 基于配置策略，消息缓存本地队列，然后异步批量发送出去
- Consumer
  - 异步批量拉取数据放入本地缓存队列
  - 按Consumer Group批量Commit Offsets更新消费状态
- Broker
  - 无状态设计，对Broker负担小
  - Broker的内部处理流水线化，分为多个阶段来进行(SEDA)
  - 利用sendfile系统调用，zero-copy减少Copy次数，批量化传输数据
  - 消息持久化，充分利用磁盘顺序读写和操作系统page cache优势，批量化flush磁盘。
  - follower从leader异步批量复制数据，这种复制方式极大的提高了吞吐率。

Controller与Broker关系

- Controller职责
  - 每个集群中会选举一个Broker作为Controller，负责与集群内每个Broker建立连接(包括Controller自身)并做交互。
- Controller向Broker发送三种请求:
  - 选举出新的replica leader通知所有Brokers
  - 新建或停止replica(**删除或停止复制replica**)通知目标Brokers
  - 向所有Brokers发送增量的元数据更新请求，元数据内容包含:Controller选举轮次、Partition对应的replicas、replica leader、replica选举轮次
- Broker职责
  - Broker负责维护replica leader中Isr列表，并把Isr变化通知给Controller

Partition中Leader选举

- 以下例子中，初始创建1 Topic 10 Partition 3 replicas，红色框中都为replica leader，其他都为replica follower。

broker-0	broker-1	broker-2	broker-3	broker-4	
p0	p1	p2	p3	p4	(1st replica)
p5	p6	p7	p8	p9	(1st replica)
p4	p0	p1	p2	p3	(2nd replica)
p8	p9	p5	p6	p7	(2nd replica)
p3	p4	p0	p1	p2	(3nd replica)
p7	p8	p9	p5	p6	(3nd replica)

- 选举Partition leader原则为Isr中第一个replica被选为leader

- **自动partition leader rebalance触发条件**：计算出Partitions中本应该属于某个Broker作为leader，自己却不是数量/Partitions中本应该属于某个Broker为leader的数量 ＞比例(默认10%)
- **partition leader必须从Isr中选出**，如果某个Partition Leader所在Broker挂了，Isr又为空，是无法选举出replica leader的，此时Partition读写为不可用状态

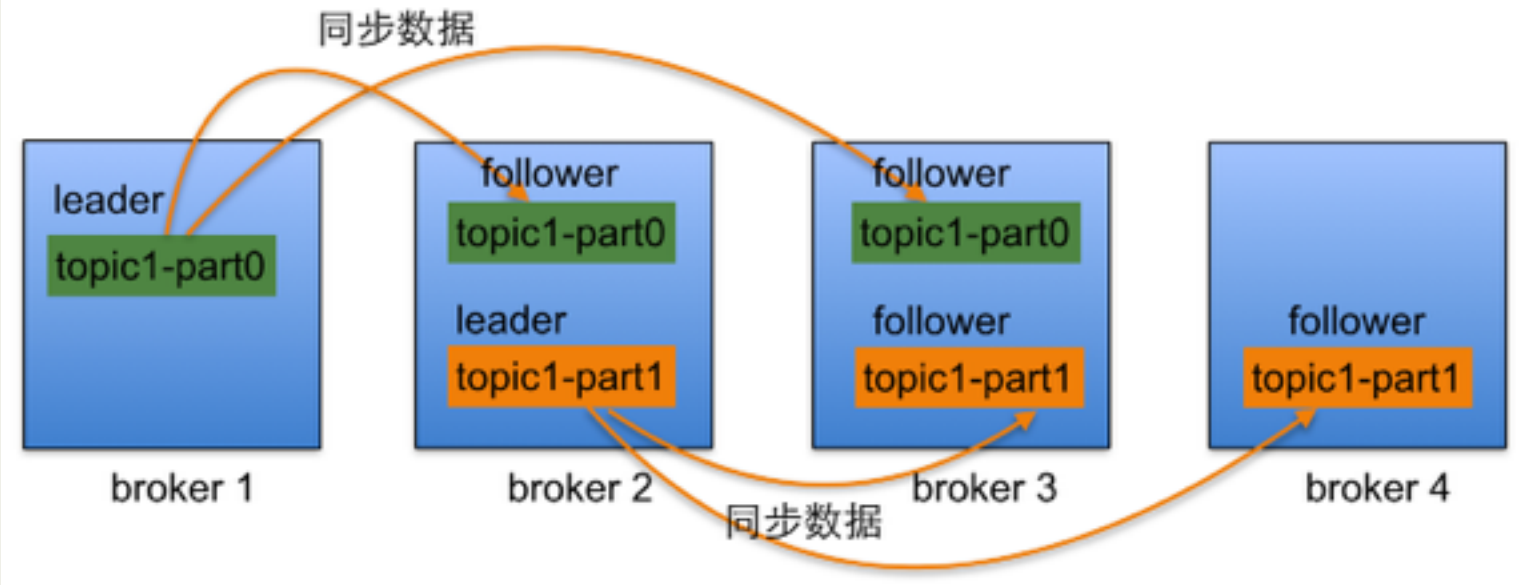
Isr扩张/收缩

- Isr扩张
  - 非Isr中follower从leader同步数据时，follower的**LogEndOffset** >= **leaderHighWatermark**，leader就会把follower加入Isr中
- Isr收缩
  - ISR expiration线程定期执行，Follower从Leader同步数据有一些延迟(包括了**延迟时间**和**延迟条数**两个维度)，任意一个超过阈值都会把该Follower踢出Isr。

Isr与MinIsr关系

- Isr
  - 决定能否选出leader，如果Isr.size < 2，leader挂了，无法选出Partition的leader，Patition读写为不可用状态
- MinIsr
  - 决定消息是否会丢失，ack = -1、

Replica复制机制



- Follower异步从Leader复制数据
  - 吞吐率量高
  - 复制性能好
  - 降低了读取磁盘IO次数
  - all replicas ack延迟性高
  - 如没有限流措施，极端情况网卡容量打满

Replicas数据迁移

broker-0	broker-1	broker-2	broker-3	broker-4	
p0	p1	p2	p3	p4	(1st replica)
p5	p6	p7	p8	p9	(1st replica)
p4	p0	p1	p2	p3	(2nd replica)
p8	p9	p5	p6	p7	(2nd replica)
p3	p4	p0	p1	p2	(3nd replica)
p7	p8	p9	p5	p6	(3nd replica)

Replicas迁移过程如下：

- 创建新的replicas列表
  - 用户执行工具，在zk上创建新的replicas列表
- Controller感知replicas变化
  - Controller从zk上wather到replicas变化

Replicas数据迁移-续

Replicas迁移过程如下：

- Controller发送新建New Replica请求
  - Controller向目标Brokers发送RPC请求，Broker在本地创建新replica，加入replicas
  - follower开始从leader同步数据，Controller等待直到follower追赶上leader加入Isr，Broker通过zk协调通知Controller
- Controller发送停止Old Replica请求
  - Controller向目标Brokers发送停止old replicas请求
  - Broker删除本地replicas
- 停止Old Replica解释：
  - 如果为Broker Shutdown，则为停止复制replica请求
  - 如果是迁移replicas，则为删除replica请求

刷盘策略 & 恢复(checkpoint)

- 同步刷盘
- 异步刷盘
  - 二者的区别在于是写完PageCache直接返回，还是刷盘后返回。
- Checkpoint机制的作用
  - 保证数据的一致性，将脏数据写入到硬盘，保证page cache和硬盘上的数据是一样的.
  - 缩短Broker恢复的时间，也避免数据损坏，consumer无法消费
- Checkpoint工作过程
  - checkpoint线程定期将所有topic-parition的EndLogOffset更新到RecoveryPointCheckpointFile文件中。

Producer HA机制

美团网 meituan.com

- 同步模式
  - Producer会在尝试重新发送message.send.max.retries（默认值为3）次后抛出Exception
  - 用户捕捉到异常，停止发送会阻塞，继续发送消息会丢失。
  - 每次retry都会更新元数据，发送到最新replica leader上。
- 异步模式
  - 发送3次失败后，用户无法捕捉异常，只能通过日志定位问题了。
- 为什么会发送失败
  - 要么只有一个replica
  - 要么虽有多个replicas，当时follower不在isr中，无法选举出leader
- Producer发送消息到Broker如何容错?
  - 按Partition粒度返回错误信息

Producer Ack机制

- ack机制为Producer粒度，通过设置request.required.acks支持：
  - acks = 0，发了不需要返回ack，无论成功与否。
  - acks = 1，写完leader replica就返回，这样会有数据丢失的风险，如果leader的数据没有来得及同步，leader挂了，那么会丢失数据；
  - acks = -1，isr = replicas - Osr都成功后，才能返回，消息不会丢失；这种纯同步写的延迟会比较高

Consumer API

美团网 meituan.com

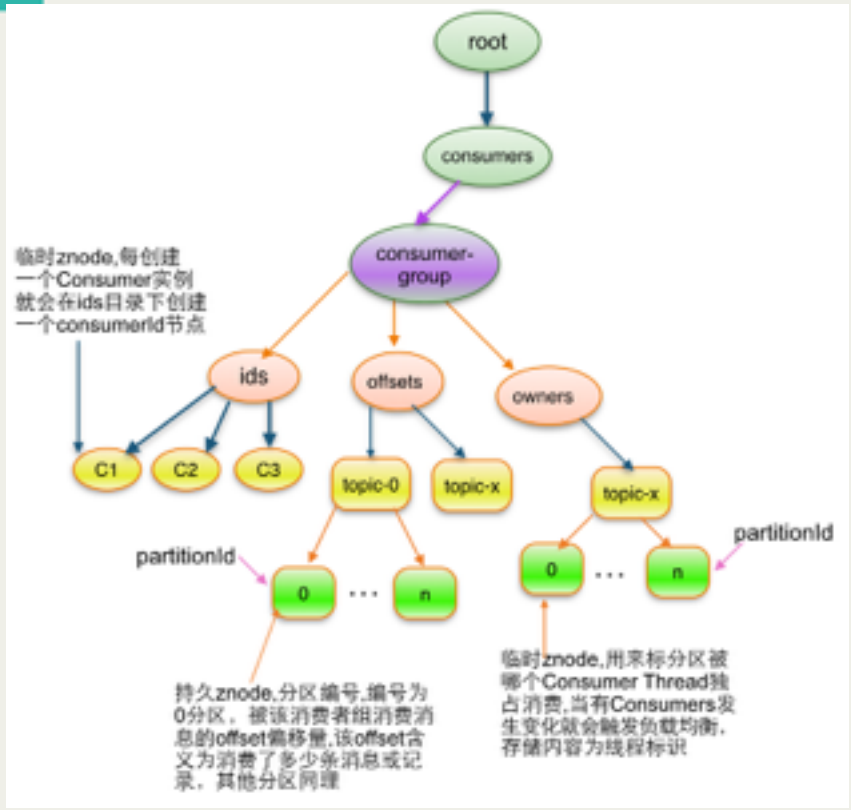
- 低级API
  - 使用者自己管理和维护消费状态(offset)
  - 自己维护replica leader HA
  - 自己维护集群中Broker List变化
  - 需要手动指定partition消费
  - 使用复杂，维护困难，很少使用
- 高级API
  - 基于zk协调，自动负载均衡
  - 支持自动replica leader HA
  - 支持自动维护消费状态(offset)
  - 使用起来简单、方便，但API灵活性欠缺，目前广泛使用。

Consumer Ack机制

美团网 meituan.com

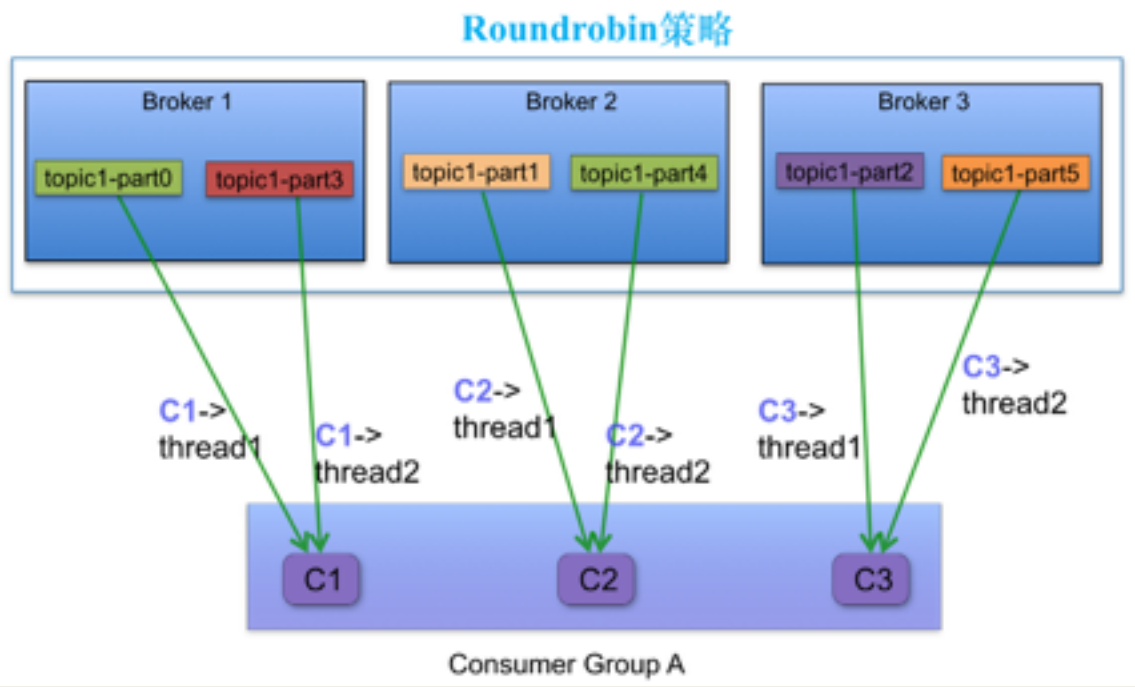
- 消费状态存储zk
  - 手动commit offsets更新消费状态
  - 异步线程定期自动commit offsets更新消费状态

Consumer rebalance



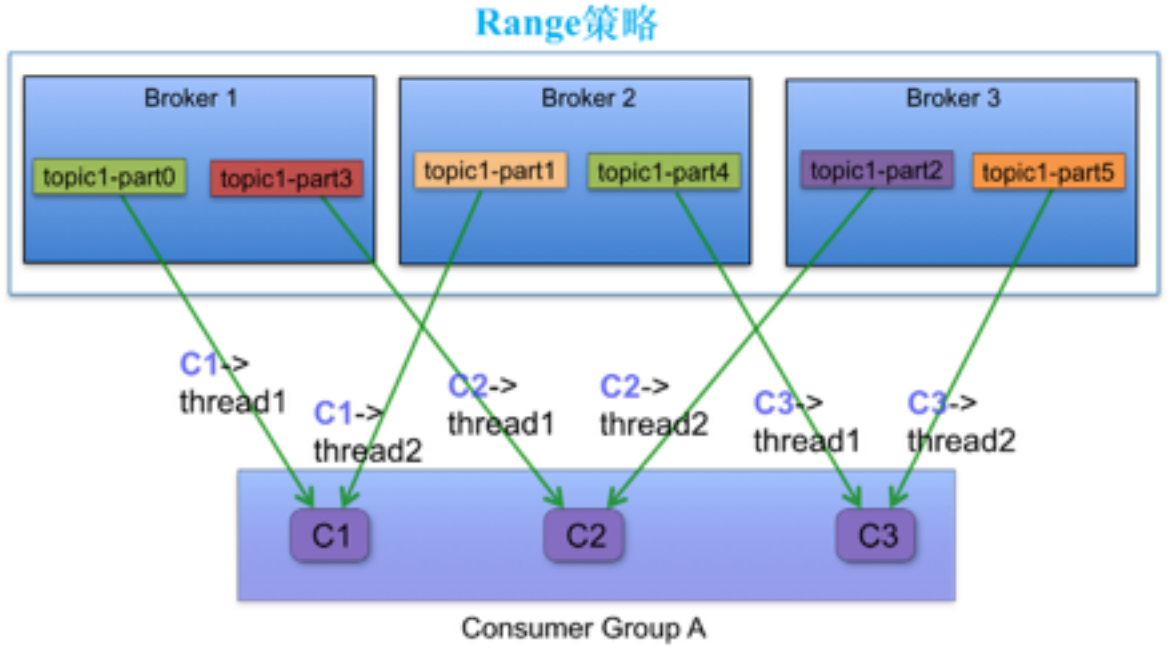
- 触发Consumer rebalance条件
  - Consumer group下Consumers数量发生变化
  - Topic中partitions和replicas变化
  - zk会话超时过期
- Partitions分配策略
  - Roundrobin策略
  - Range策略

Roundrobin策略



- Partition分配器列出所有可用Partitions和所有可用consumer线程。
- Round-robin的方式将Partition分配给Consumer线程。
- 如果所有Consumer的订阅是相同的，那么Partitions会被均匀分配。
- 所有Consumer线程占有的Partition数量相差不会超过1

Range策略



- Range分配器是基于Topic的。
- 对于每个Topic，以数字顺序列出所有Partitions，以词典序列出所有Consumer线程。
- 将Partitions数除以Consumer线程数，来决定分给每个Consumer多少Partition。
- 分配不均匀，前几个Consumers会多分1个Partitoin。

美团网
meituan.com

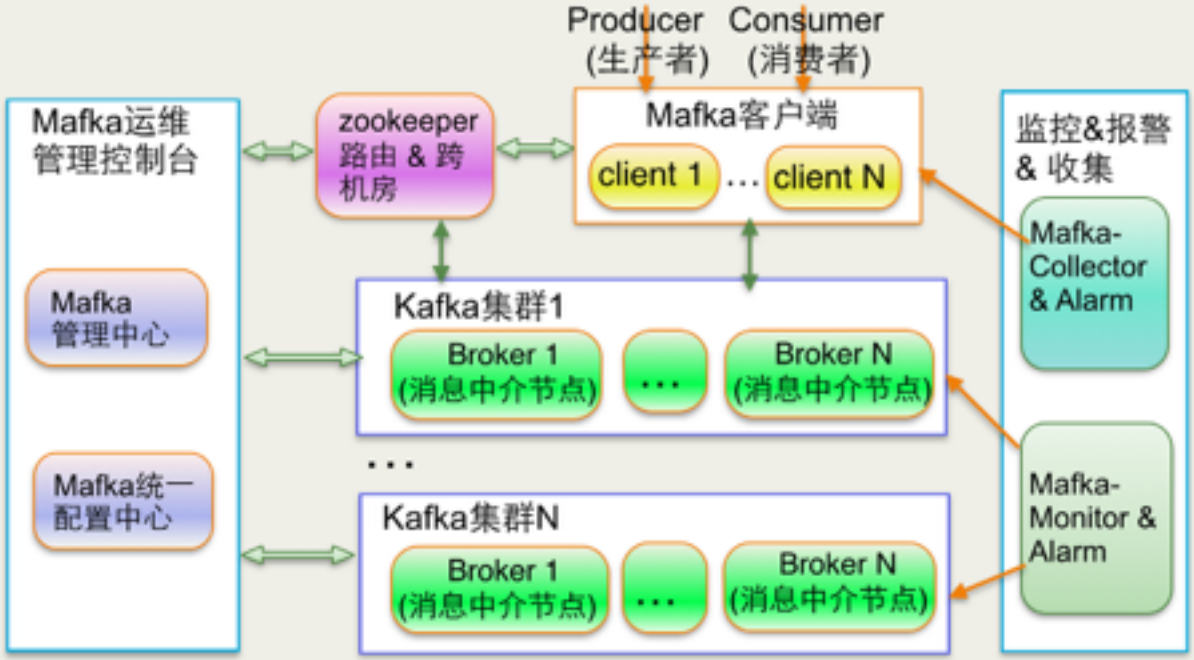
美团网
meituan.com

1 Kafka设计关键点

2 Mafka设计与实现

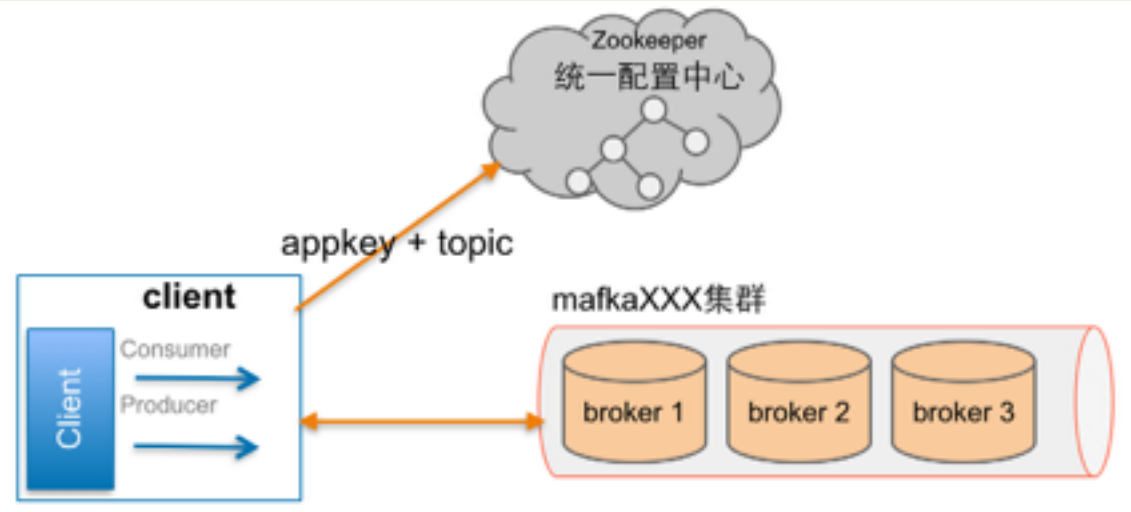
3 Mafka & Kafka不同

Mafka物理架构



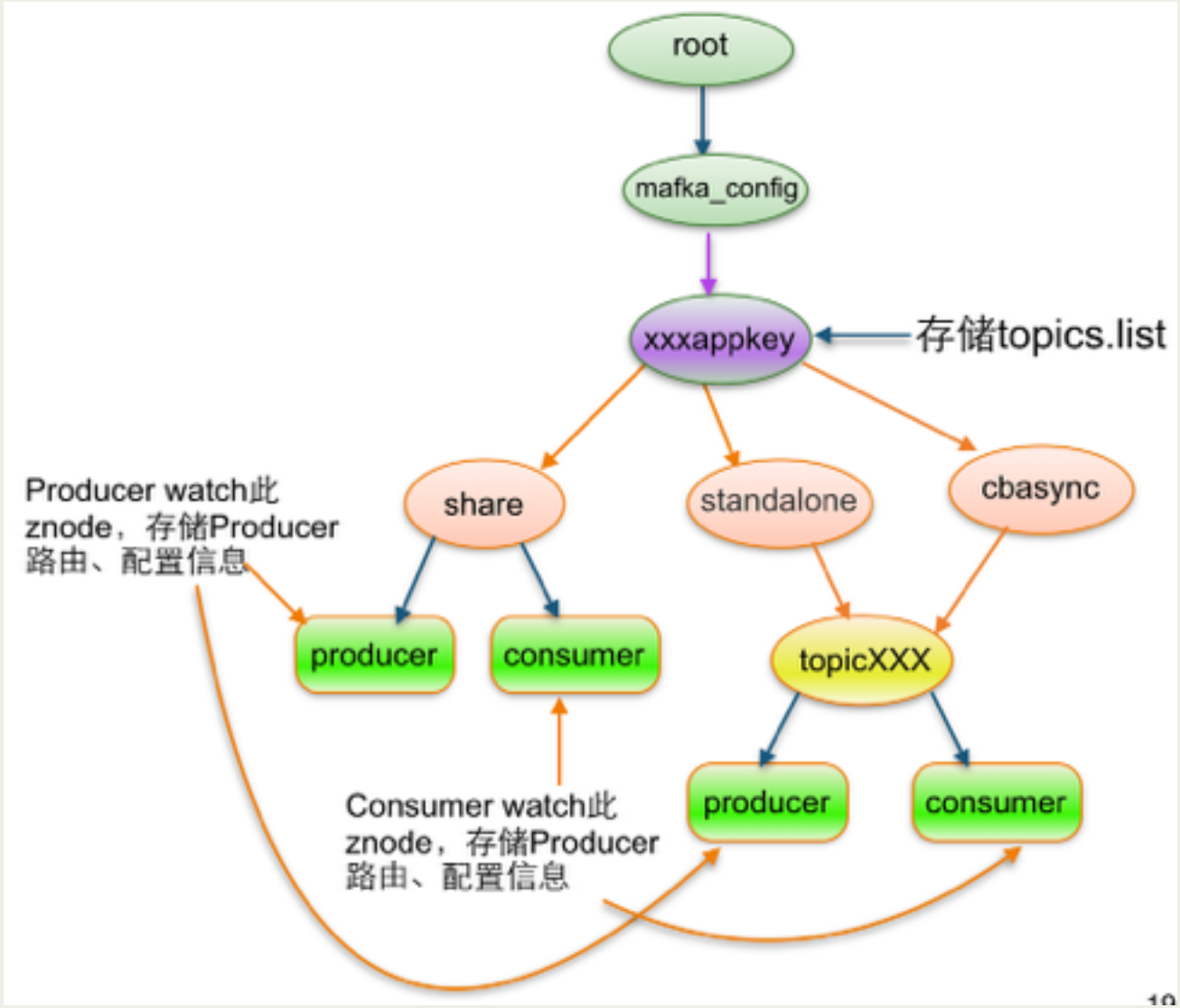
- Mafka管理中心：负责集群服务节点管理，主题/订阅管理和信息查看等。
- Mafka统一配置中心：统一配置中心管理平台(实际是zk的管理工具)。
- Mafk客户端：提供简单易用客户端,负责向Kafka集群发送和接受消息.
- Mafka监控和报警：用于监控和报警，定制规则.
- Kafka集群：由多个Broker组成的Kafka集群用于存储和转发消息
- Zookeepr：**在Kafka基础上新增职责，存储统一配置中心元数据**。

Mafka客户端设计



- Producer和Consumer通过appkey + topic唯一性访问统一配置中心获取路由以及自身配置信息。
- Appkey是什么，是用于Mafka客户端路由、读取自身配置信息以及权限访问，他就是在zk下面创建/xxxx/mafka_config/xxxappkey节点
- Appkey包含topic列表，只有Appkey下列表包含相应topic，客户端才能访问

Mafka配置中心设计



- Mafka配置中心在zk上数据结构

美团网
meituan.com

1 Kafka设计关键点

2 Mafka设计与实现

3 Mafka & Kafka不同

Mafka & Kafka不同

美团网 meituan.com

Mafka相对于Kafka特有的一些功能

- Makfa客户端
  - 提供一个使用友好，开箱即用客户端
  - 统一配置中心来实现在线修改客户端配置
  - 支持多机房本地访问
  - 为了支持async + batch + callback，集成了Kafka java语言客户端
  - 能实时了解客户端使用版本，方便升级。
  - Producer增加了zk依赖，**原生Kafka的Producer不依赖zk**
  - Producer watcher集群中目标Topic的partitions变化，数据迅速发送到新增的Partitions
  - Producer watcher集群总Broker.list变化，某个或些Brokers掉线迅速从发送partitions列表中摘除(指的是落在该Broker的partitions)
  - 支持基于配置的消费失败重新投递

Mafka & Kafka不同

美团网 meituan.com

Mafka相对于kafka特有的一些功能

- Kafka scala语言客户端功能增强
  - 发送消息，支持Partition间失效转移
  - 基于高级API扩展功能，保证每条消息至少消费一次
  - 基于高级API扩展功能，支持consumer指定partition消费
- Mafka管理中心
  - 主题/订阅管理
  - 监控管理(消息堆积、Broker变化、性能指标等)
  - 用户管理(用户权限管理、用户管理)
  - 运维工具(服务节点管理、zk管理，集群信息管理、数据迁移工具)
- Mafka监控系统
  - 消息堆积监控、Brokers存活监控等

Mafka不足

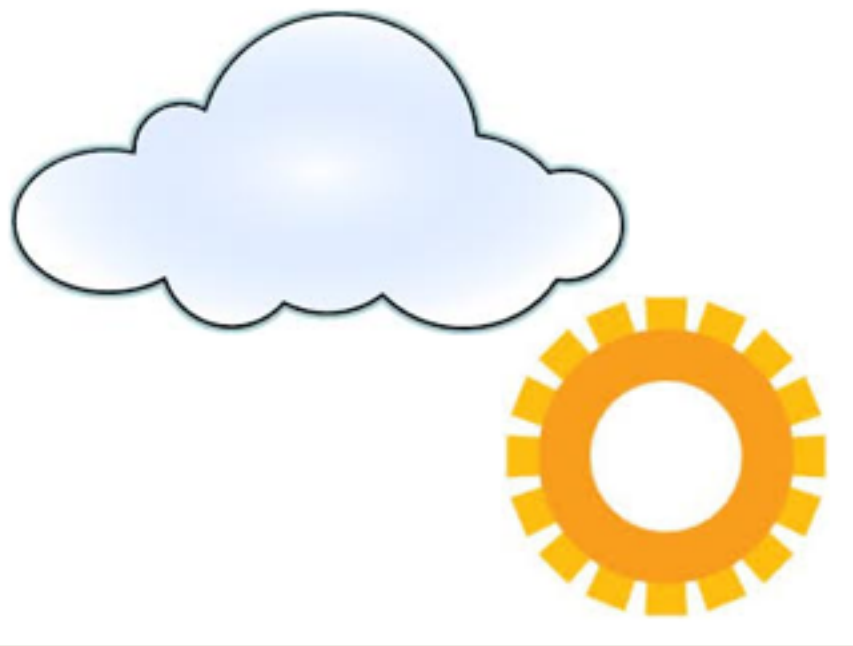
美团网 meituan.com

- Mafka客户端
  - 接口需要收敛(1，2接口弃用)
- Mafka管理中心
  - 操作体验不够优化
  - 页面显示性能较差(与UI框架有关，dom为动态创建)
  - UI扩展性差，添加交互稍复杂功能比较困难
  - 参考开源UI框架重构Mafka管理中心
- Mafka项目git仓库
  - 项目需要梳理
  - 项目依赖关系优化

QA

美团网
meituan.com

Thank you!
Any Quest?



美团网
meituan.com

谢谢！