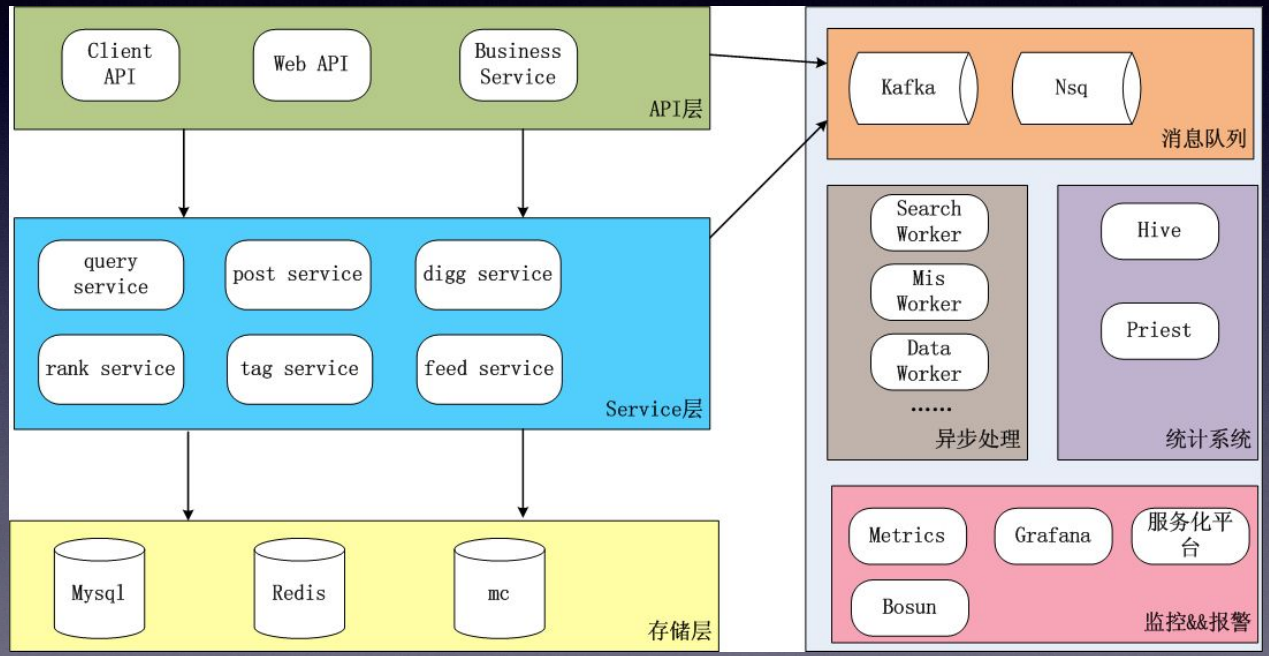
分布式消息队列介绍

付建双

概要

- 为什么使用消息队列
- Nsq
- Kafka
- TMQ框架介绍
- 实现简介

场景 - 问答业务


Client API
Web API
Business Service
API层
query service
post service
digg service
rank service
tag service
feed service
Service层
Mysql
Redis
mc
存储层
Kafka
Nsq
消息队列
Search Worker
Mis Worker
Data Worker
.....
异步处理
Hive
Priest
统计系统
Metrics
Grafana
服务化平台
Bosun
监控&&报警

场景 - 问答业务

- 用户有如下操作：
  - 在问答app提出一个问题
  - 自己收藏这个问题
  - 其他人回答这个问题
- 要求：对该问题的操作要有序处理
- 消费：会有审核、运营、推荐、搜索等多个消费方
- 消费时出错怎么办？

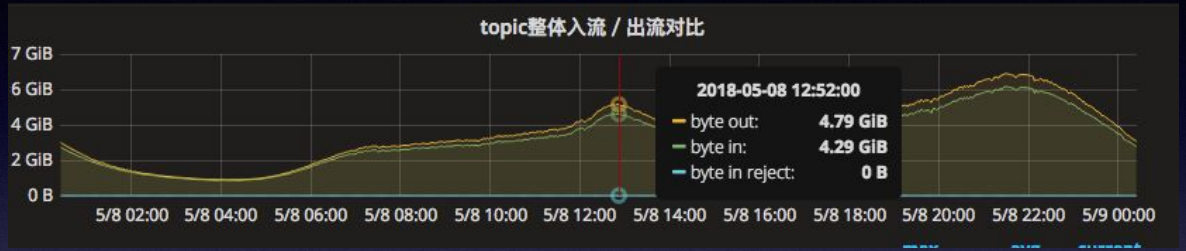
场景 - 推送

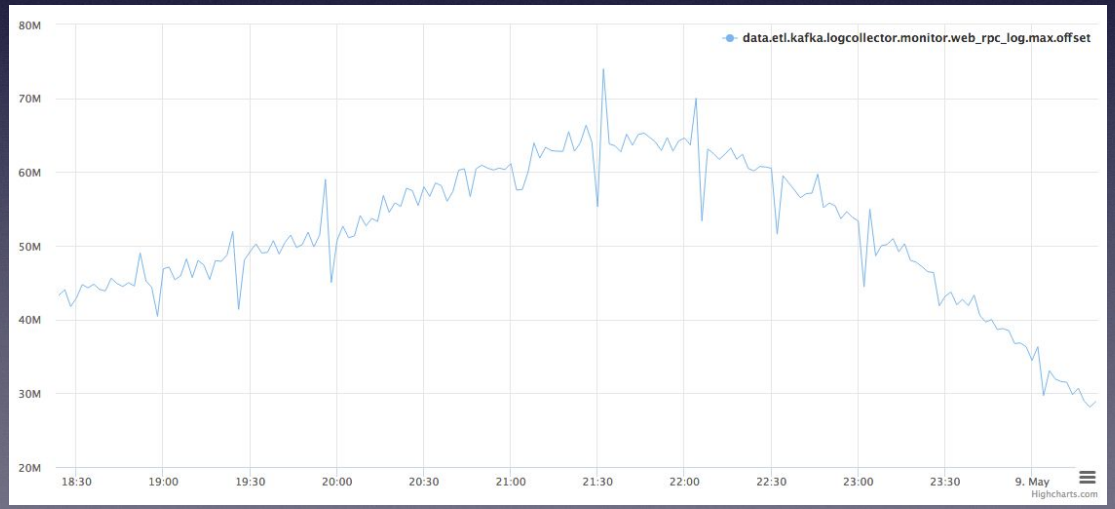
- 推送数据的突发流量

场景 - 买火车票

- 节假日抢票困难，12306服务经常不可用
- 解决措施之一：排队

场景 - 服务日志处理


topic整体入流 / 出流对比
7 GiB
6 GiB
4 GiB
2 GiB
0 B
2018-05-08 12:52:00
byte out: 4.79 GiB
byte in: 4.29 GiB
byte in reject: 0 B
5/8 02:00
5/8 04:00
5/8 06:00
5/8 08:00
5/8 10:00
5/8 12:00
5/8 14:00
5/8 16:00
5/8 18:00
5/8 20:00
5/8 22:00
5/9 00:00


data.etl.kafka.logcollector.monitor.web_rpc_log.max.offset
80M
70M
60M
50M
40M
30M
20M
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30
9. May
Highcharts.com

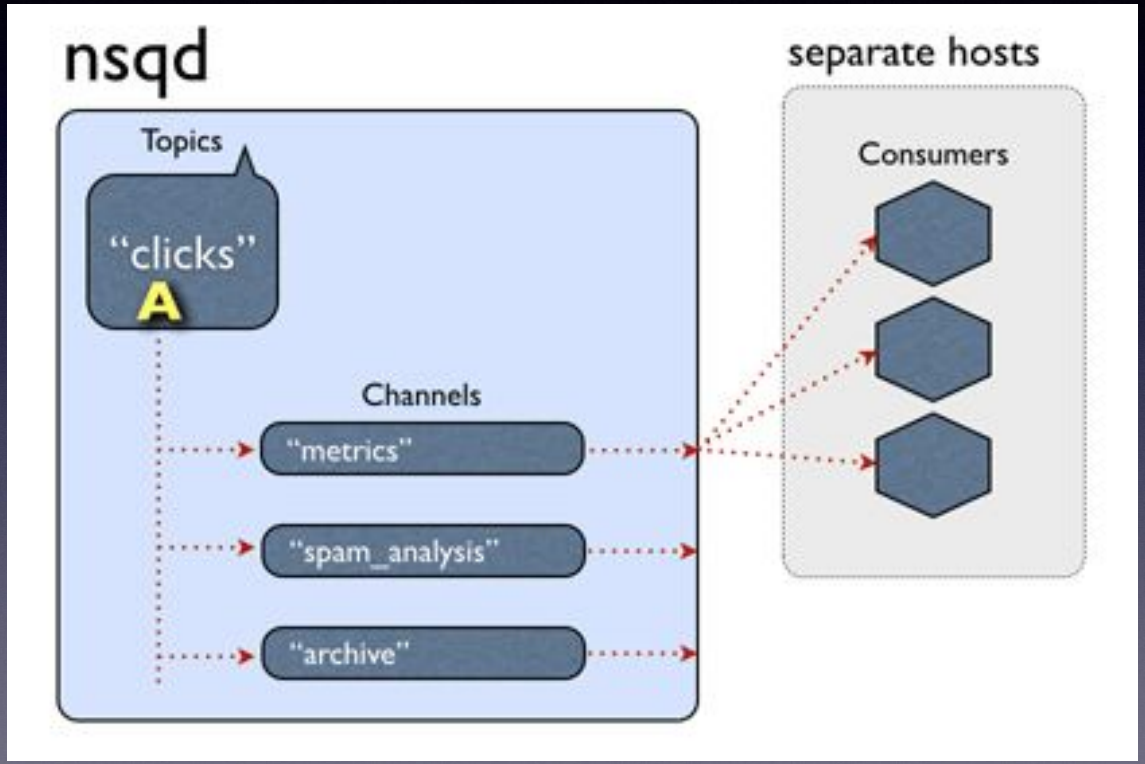
需要一个消息队列

- 解耦：上游关注通知，而不关注下游处理。需要约定好消息格式
- 缓冲：应对突发流量
- 广播：一条消息，可以被多个下游分别处理
- 异步：发送消息后处理消息不需要马上进行
- 冗余：消息可以被持久化，可以回溯

公司常用消息队列

- Nsq：模型简单、性能一般、功能丰富
- Kafka：模型复杂、高吞吐、大数据生态

Nsq


nsqd
separate hosts
Topics
"clicks"
A
Channels
"metrics"
"spam_analysis"
"archive"
Consumers

Nsq

- producer：生产者，通常通过http发送消息
- consumer：消费者，nsqd把消息推送过来
- nsqd：server端主要组件
- topic：一个队列的逻辑概念
- channel：一个topic的消息可被多个channel消费

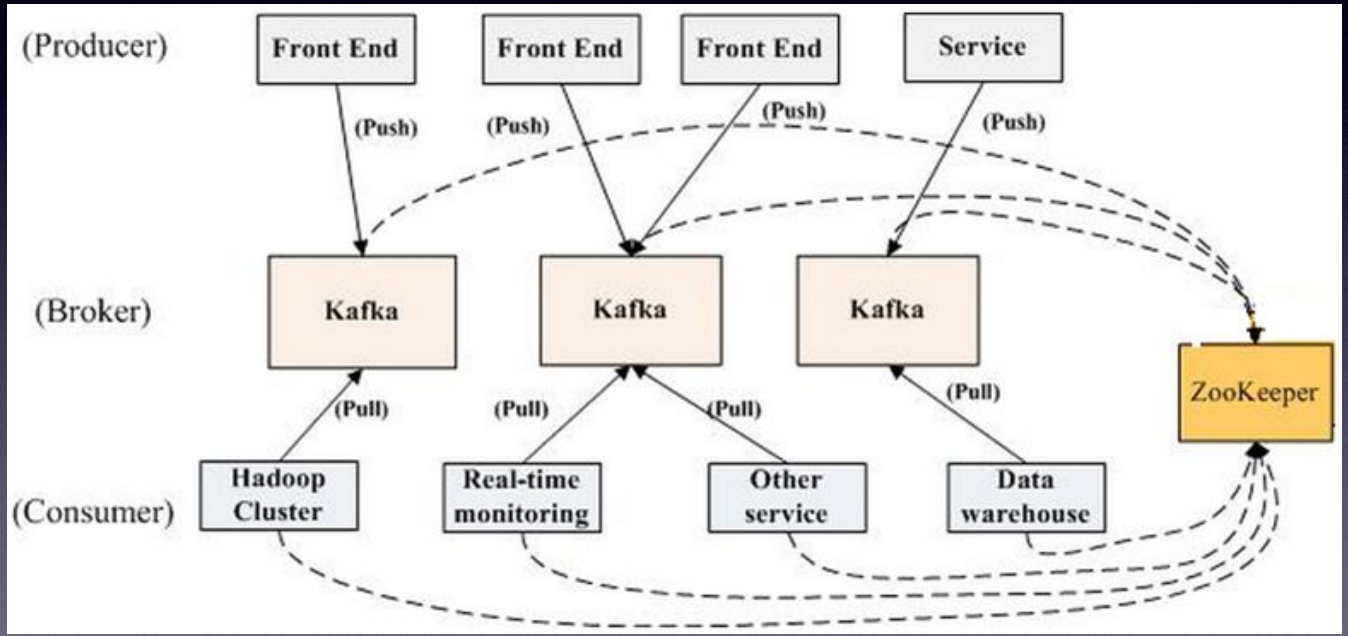
Nsq主要特性

- 消息没有被持久化
- 消息至少被投递一次(不严格)
- 支持requeue，延时消费
- 消息投递是无序的(为什么？)
- 消费者最终会发现并连接到所有nsqd

特性足够吗？

	消息不丢	投递保证	顺序保证	可回溯	requeue defer功能	吞吐	生态
Nsq	nsqd故障时会丢	at least once	无序	否	是	一般	无
Kafka	持久化，Replica	at least once at most once exactly once	有序	是	否	高	Hadoop

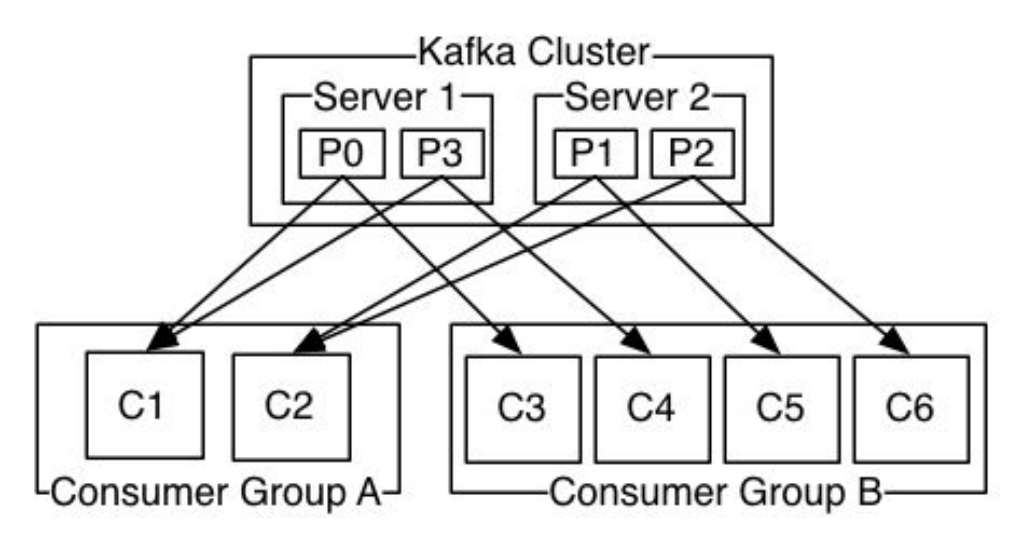
Kafka整体结构


(Producer)
Front End
Front End
Front End
Service
(Push)
(Push)
(Push)
(Push)
(Broker)
Kafka
Kafka
Kafka
ZooKeeper
(Pull)
(Pull)
(Pull)
(Pull)
(Consumer)
Hadoop Cluster
Real-time monitoring
Other service
Data warehouse

Kafka角色

- Producer：消息发布者
- Consumer：消息消费者，从Kafka broker读取消息
- Broker：Kafka集群服务端一个服务器
- Topic：队列的逻辑抽象
- Partition：每个Topic包含一个或多个Partition，发送消息的时候要选择一个partition来发，单个partition内消息被消费时有序
- Consumer Group：一个topic的消息可以被多个group消费，每个group消费一份完整的消息

Consumer


Kafka Cluster
Server 1
Server 2
P0
P3
P1
P2
C1
C2
C3
C4
C5
C6
Consumer Group A
Consumer Group B

Consumer

- 推拉模型：
  - Kafka是拉模型，吞吐略高，实时性偏低
  - Nsq是推模型，nsqd需要考虑consumer负载能力
- offset提交：zk or broker or abase
- auto rebalance

TMQ框架

- 封装了kafka和nsq的produer和consumer，有python和go两个语言的实现
- 根据场景选择不同底层MQ
- log、metrics、初始化等对标RPC框架
- 监控报警
- 精心调节的默认参数和动态配置功能
- 调用链搜索

TMQ使用-Go producer

- 获取代码：go get -u code.byted.org/tmq/tmq
- 参考示例：go get -u code.byted.org/tmq/examples
- 初始化config
- 初始化producer
- 发送数据

TMQ使用-Go producer

```go
var producer *tmq.TmqProducer

func ProducerExample() {
        // 服务的PSM
        psm := "toutiao.webarch.test_service"
        topic := "tmq_test"
        // 在NSQ平台(http://nsq.byted.org)申请topic时对应的集群
        cluster := "test_lf"

        config := tmq.NewProducerConfig()
        config.MqCluster = cluster
        // NSQ生产者HTTP请求的timeout默认值是150ms，这应该能满足绝大部分应用。
        // 如果你想自定义timeout值，解注释下面这行代码，并把timeout的值指定成你想的值。
        // config.Nsq.Timeout = time.Millisecond * 150

        var err error
        producer, err = tmq.NewNsqProducer(psm, topic, config)
        if err != nil {
                fmt.Fprintf(os.Stderr, "create producer error: %v\n", err)
                os.Exit(-1)
        }

        // 如果服务使用了Kite、TMQ、Ginex等框架，直接把ctx传递给Send函数即可，
        // 对于没有使用框架的服务，使用context.Background()作为ctx传递给Send函数
        ctx := context.Background()
        err := producer.Send(ctx, []byte("byte array"))
}
```

TMQ使用-Go Consumer

- 获取工具：go get -u code.byted.org/tmq/tmqtool
- 生成代码：tmqtool new --type mqserver --pre code.byted.org/namespace/repo --psm toutiao.test.consumer --topic test --mqtype nsq
- 在TMQ平台注册consumer信息
- 在MessageHandler中填写消费时的业务逻辑

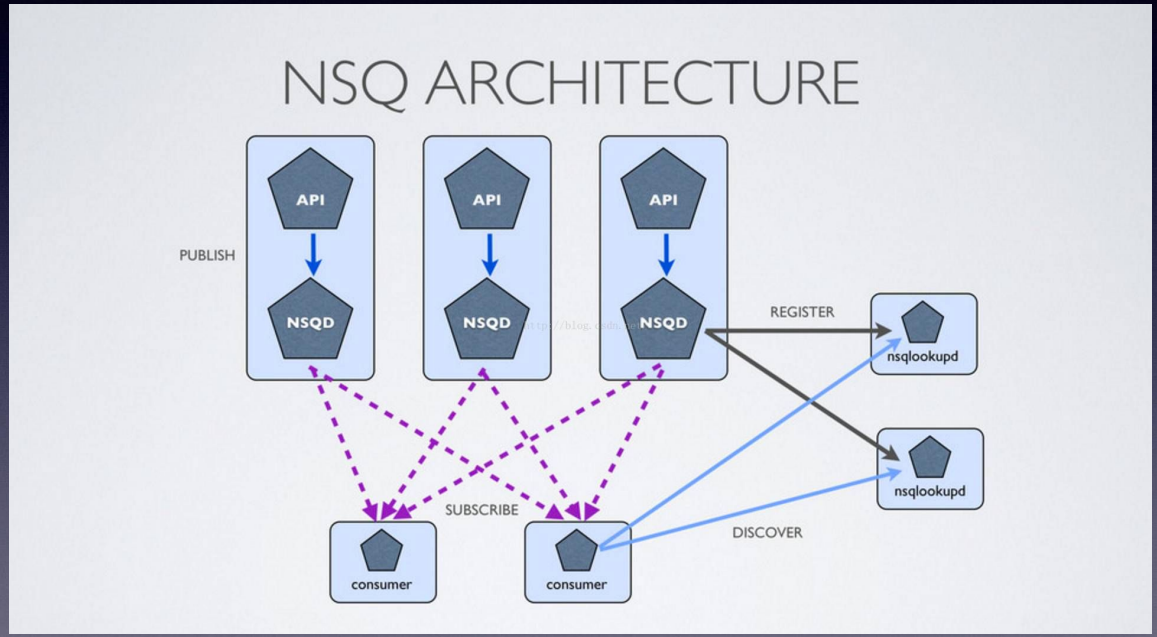
TMQ平台

- 地址：tmq.byted.org
- 注册topic和consumer信息
- 监控
- 报警
- 动态配置

实现简介

- Nsq组件和结构
- Kafka consumer auto rebalance
- Kafka high performance

Nsq架构


NSQ ARCHITECTURE
PUBLISH
API
NSQD
API
NSQD
API
NSQD
REGISTER
nsqlookupd
nsqlookupd
SUBSCRIBE
DISCOVER
consumer
consumer

Nsq组件

- nsqd
  disk queue
- nsqlookupd
- nsqadmin

Kafka Auto Rebalance

- Why？

- Watch zookeeper?
  Herd effect
  Split Brain

- Coordinator
  1. join group
  2. partition assign by consumer
  3. metadata version
  4. heart beat

Kafka High Performance

- batch send
- compression
- ISR
- write append only
- page cache
- zero copy

思考

- 如何做到写DB和发消息的事务性
- 如何让nsq尽量不丢消息
- 如何做到exactly-once

Thanks