**防火墙**

**防火墙**

HTML端、API接口

前端架构

分布式配置

分布式服务1

应用程序

本地缓存

统一数据访问模块

分布式服务2

消息队列

B应用服务器

A应用服务器

分布式缓存服务器

远程分布式缓存

**RPC**

分布式文件服务器

文件

**防火墙**

CDN服务器

反向代理服务器

分布式数据库服务器

数据库

负载均衡服务器

**防火墙**

实时同步配置信息

代码与数据同步更新

分布式发布

日志

获取数据做缓存

数据

搜索引擎服务器

数据采集与监控

1. **前端架构（Vue.js，API，WebSocket）**

减少HTTP请求

浏览器缓存

启用压缩

CSS、JavaScript加载

Cookie传输

1. **CDN服务器**

CDN加速

CDN服务器配置

CDN服务器优化

1. **反向代理服务器（Nginx）**

反向代理的原理

操作实现

1. **负载均衡服务器**

实现方式

HTTP重定向

DNS域名解析

反向代理

IP负载均衡

数据链路层负载均衡

均衡算法

轮询（Round Robin， RR）

加权轮询（Weighted Round Robin， WRR）

随机（Random）

最少链接（Least Connections）

源地址散列（Source Hashing）

1. **应用服务器**

演化过程

业务拆分

分层

分割

分布式

集群

性能瓶颈

解决方案

1. **消息队列服务器（RabbitMQ、Kafka、ActiveMQ、RocketMQ、Redis）**

作用

解耦

异步

削峰/限流

带来的问题

可用性降低

数据丢失

一致性问题

流行技术

| **对比方向** | **概要** |
| --- | --- |
| 吞吐量 | 万级的 ActiveMQ 和 RabbitMQ 的吞吐量（ActiveMQ 的性能最差）要比十万级甚至是百万级的 RocketMQ 和 Kafka 低一个数量级。 |
| 可用性 | 都可以实现高可用。ActiveMQ 和 RabbitMQ 都是基于主从架构实现高可用性。RocketMQ 基于分布式架构。 kafka 也是分布式的，一个数据多个副本，少数机器宕机，不会丢失数据，不会导致不可用 |
| 时效性 | RabbitMQ 基于Erlang开发，所以并发能力很强，性能极其好，延时很低，达到微秒级。其他三个都是 ms 级。 |
| 功能支持 | 除了 Kafka，其他三个功能都较为完备。 Kafka 功能较为简单，主要支持简单的MQ功能，在大数据领域的实时计算以及日志采集被大规模使用，是事实上的标准 |
| 消息丢失 | ActiveMQ 和 RabbitMQ 丢失的可能性非常低， RocketMQ 和 Kafka 理论上不会丢失。 |

ActiveMQ 的社区算是比较成熟，但是较目前来说，ActiveMQ 的性能比较差，而且版本迭代很慢，不推荐使用。

RabbitMQ 在吞吐量方面虽然稍逊于 Kafka 和 RocketMQ ，但是由于它基于 erlang 开发，所以并发能力很强，性能极其好，延时很低，达到微秒级。但是也因为 RabbitMQ 基于Erlang开发，所以国内很少有公司有实力做erlang源码级别的研究和定制。如果业务场景对并发量要求不是太高（十万级、百万级），那这四种消息队列中，RabbitMQ 一定是你的首选。如果是大数据领域的实时计算、日志采集等场景，用 Kafka 是业内标准的，绝对没问题，社区活跃度很高，绝对不会黄，何况几乎是全世界这个领域的事实性规范。

RocketMQ 阿里出品，Java 系开源项目，源代码我们可以直接阅读，然后可以定制自己公司的MQ，并且 RocketMQ 有阿里巴巴的实际业务场景的实战考验。RocketMQ 社区活跃度相对较为一般，不过也还可以，文档相对来说简单一些，然后接口这块不是按照标准 JMS 规范走的有些系统要迁移需要修改大量代码。还有就是阿里出台的技术，你得做好这个技术万一被抛弃，社区黄掉的风险，那如果你们公司有技术实力我觉得用RocketMQ 挺好的

kafka 的特点其实很明显，就是仅仅提供较少的核心功能，但是提供超高的吞吐量，ms 级的延迟，极高的可用性以及可靠性，而且分布式可以任意扩展。同时 kafka 最好是支撑较少的 topic 数量即可，保证其超高吞吐量。kafka 唯一的一点劣势是有可能消息重复消费，那么对数据准确性会造成极其轻微的影响，在大数据领域中以及日志采集中，这点轻微影响可以忽略这个特性天然适合大数据实时计算以及日志收集。

**redis与rabbitmq**

**可靠性**

redis ：没有相应的机制保证消息的可靠消费，如果发布者发布一条消息，而没有对应的订阅者的话，这条消息将丢失，不会存在内存中；

rabbitmq：具有消息消费确认机制，如果发布一条消息，还没有消费者消费该队列，那么这条消息将一直存放在队列中，直到有消费者消费了该条消息，以此可以保证消息的可靠消费，那么rabbitmq的消息是如何存储的呢？（后续更新）；

**实时性**

redis:实时性高，redis作为高效的缓存服务器，所有数据都存在内存中，所以它具有更高的实时性

**消费者负载均衡**

rabbitmq队列可以被多个消费者同时监控消费，但是每一条消息只能被消费一次，由于rabbitmq的消费确认机制，因此它能够根据消费者的消费能力而调整它的负载；

redis发布订阅模式，一个队列可以被多个消费者同时订阅，当有消息到达时，会将该消息依次发送给每个订阅者，她是一种消息的广播形式，redis本身不做消费者的负载均衡，因此消费效率存在瓶颈；

**持久性**

redis：redis的持久化是针对于整个redis缓存的内容，它有RDB和AOF两种持久化方式（redis持久化方式，后续更新），可以将整个redis实例持久化到磁盘，以此来做数据备份，防止异常情况下导致数据丢失。

rabbitmq：队列，每条消息都可以选择性持久化，持久化粒度更小，更灵活；

**队列监控**

rabbitmq实现了后台监控平台，可以在该平台上看到所有创建的队列的详细情况，良好的后台管理平台可以方面我们更好的使用；

redis没有所谓的监控平台。

**总结**

redis：轻量级，低延迟，高并发，低可靠性；

rabbitmq：重量级，高可靠，异步，不保证实时；

1. **分布式服务（HSF、Dubbo、Hprose、swoft、docker）**

实现方式

微服务

技术方案

1. **分布式缓存（Redis、Memcached、MongoDB）**

优缺点分析

NoSQL产品及使用场景

Session服务器

问题分析

一致性hash

缓存穿透

缓存雪崩

1. **分布式文件服务器（腾讯COS、七牛云存储、阿里OSS）**

快速加载与压缩

市场各大产品使用

1. **分布式数据库（MySQL、Oracle、MyCat、**360 Atlas**）**

MySQL使用基础

优化方案

分布式集群实施

读写分离

主从同步

中间件

配置

分片规则

分布式事务

2PC

3PC

TCC补偿事务

Saga事务

MQ消息队列+最终一致性

1. **搜索引擎服务器（ElasticSearch、Solr、sphinx、opensearch）**

实现原理

全文搜索引擎搭建

1. **分布式配置**

分布式服务器管理

实时原则

案例实现

1. **分布式测试发布（Jenkins）**

版本工具使用

自动化测试

预发布验证

代码控制

自动化发布

Jenkins自动化持续发布工具，用于持续、自动的构建/测试软件项目、监控外部任务的运行。

灰度发布

1. **数据采集与监控系统**

浏览数据采集

JS脚本采集

服务器业务数据采集

日志和业务数据

服务器性能数据采集

系统负载

内存使用率

硬盘容量

网卡流量

CPU使用率

系统监控

监控系统：Zabbix、Open-Falcon、Prometheus。

自动化运维

系统报警

阈值异常报警

1. **安全性设计**

Web攻击

XSS攻击

注入攻击

CSPF攻击

其他攻击和漏洞

Web应用防火墙

网站安全漏洞扫描

数据安全

信息加密技术

单向散列加密

对称加密

非对称加密

密钥安全管理

信息过滤与反垃圾

文本匹配

分类算法

黑名单