高并发专题

- 并发编程-> 异步编程
- 分布式消息中间件 -> 异步设计
- 分布式存储 -> I/O
- 分布式架构(服务治理) ->
- 性能优化
    - JVM
    - tomcat
    - mysql
    - 算法

站在全局的视角去思考高并发设计

用户数量100w -> 1000W

- 架构设计
- 人员的招聘
- 人员管理
- 目标推进
- 。。。。

1000W个用户的问题分解

tps/QPS/并发数/RT/DAU

200Wqps

2/8定律

每天大部分用户集中在5个小时以内访问

8000W的pv 在5个小时内集中访问

4500个请求/s *4 =18000个请求/s（峰值） qps

并发数量是多少?

qps*rt值

一个请求平均处理时间是3s = 54000

有哪些因素影响到并发数量

Tomcat

- 针对Tomcat的优化
    - 连接数量(socket.accept)

        - 192.168.221.128:8080
        - Socket/File： Can't open so many file
        - fd
        - 连接数量是对内存的占用
        - 一个连接(4096) 8k 100W（服务器能够支撑的并发连接数量）
- 如何在应用层面去支撑更多的链接数量(NIO机制)
- 线程池

```
max-threads  ; //cpu  200默认的最大工作线程数量
max-connections:  //10000默认  NIO模式下;  ARP :
```

> http://tomcat.apache.org/tomcat-8.0-doc/config/http.html

JVM

虚拟机

- gc算法
- 内存空间

> 空间不足的情况
>
> - 频繁GC
> - new object() , 内存溢出

Order order=new Order(); N ->

服务器的数量评估

一台服务器能支撑多少qps =500， 18000 = 36台服务器.

频繁访问的页面或者服务去增加服务器.

RT值

业务逻辑的处理本质上就是IO操作;

- 查询数据库(磁盘IO)
    - 数据库层面的基本优化
- 访问磁盘（磁盘IO）
    - 交易对账的时候. 文本的解析和存储
    - 磁盘的页缓存（异步刷盘、同步算盘）
    - 顺序读写
    - 零拷贝
    - SSD（固态）- HDD （机械磁盘）
    - mmap(内存映射，减少拷贝次数)
    - 异步化（最有效的武器）
- 内存运算（内存IO）
    - 内存缓存
- （远程通信）网络通信（网络IO）

    - 池化技术（一次性创建多个连接）
    - 长连接（）
    - 同网段通信
    - 减少网络通信的场景
- 异步化设计
    - 线程的设计
    - MQ的使用
- 数据结构算法

应用架构层面

采用微服务架构