# JavaVip高级课程大纲

# 1 高并发框架和分布式技术

## 1.1 NIO、Netty

### 1.1.1 和Andriod联合，实时推送

### 1.1.2和前端联合,websocket,IM聊天

### 1.1.3.netty原理

## 1.2 分布式RPC

### 1.2.1 zookeeper

#### 1.2.1.1 Zookeeper基本原理

##### 1.2.1.1.1 数据模型

##### 1.2.1.1.2 重要概念

###### 1.2.1.1.2.1 zNode

###### 1.2.1.1.2.2 Session

###### 1.2.1.1.2.3 Watcher

##### 1.2.1.1.3 zookeeper特性

###### 1.2.1.1.3.1 读、写（更新）模式

###### 1.2.1.1.3.2 WALt和Snapshot

###### 1.2.1.1.3.3 FIFO

###### 1.2.1.1.3.4 Linearizability

#### 1.2.1.2 Zookeeper Client Api

##### 1.2.1.2.1 create(path、data、flags)

##### 1.2.1.2.2 delete(path、version)

##### 1.2.1.2.3 exist(path、watch)

##### 1.2.1.2.4 getData(path、watch)

#### 1.2.1.3 Zookeeper 典型应用场景

##### 1.2.1.3.1 数据发布与订阅

##### 1.2.1.3.2 负载均衡

##### 1.2.1.3.3 命名服务

### 1.2.2 Dobbu

#### 1.2.2.1 Dobbu 原理

#### 1.2.2.2 Dobbu 层次

1.2.2.2.1 协议层

1.2.2.2.2 序列化/反序列化

1.2.2.2.3 通讯层

1.2.2.2.1 数据绑定

#### 1.2.2.3 Dobbu 使用

##### 1.2.2.3.1 spring的集成

##### 1.2.2.3.2 OSGI的支持

## 1.3 缓存

### 1.3.1 NO-SQL

#### 1.3.1.1 redis

##### 1.3.1.1.1 redis原理剖析

1.3.1.1.1.1 一致性哈希

##### 1.3.1.1.2 redis 主从模式

###### 1.3.1.1.1.2.1 一主一从

###### 1.3.1.1.1.2.1 一主多从

##### 1.3.1.1.3 redis常用命令

###### 1.3.1.1.3.1 set

###### 1.3.1.1.3.2 get

###### 1.3.1.1.3.3 lpush

###### 1.3.1.1.3.4 lrange

###### 1.3.1.1.3.5 hmget

###### 1.3.1.1.3.6 hmset

##### 1.3.1.1.4 redis 客户端

###### 1.3.1.1.4.1 jedis

###### 1.3.1.1.4.2 jedispool

###### 1.3.1.1.4.3 jedisCluster

###### 1.3.1.1.4.4 jedisCluster+springMVC整合

##### 1.3.1.1.5 redis 持久化

###### 1.3.1.1.5.1 RDB

###### 1.3.1.1.5.2 AOF

#### 1.3.1.2 manggoDB

##### 1.3.1.2.1 manggoDB原理

##### 1.3.1.2.2 动态查询

##### 1.3.1.2.3 索引

##### 1.3.1.2.4 manggoDB客户端Robomongo

##### 1.3.1.2.4 manggoDB常用命令

###### 1.3.1.2.4.1 db.foo.find()

###### 1.3.1.2.4.2 show.dbs

###### 1.3.1.2.4.3 db.dropDatabase()

###### 1.3.1.2.4.4 db.userInfo.find({age:{$gt 22}})

###### 1.3.1.2.4.5 db.userinfo.save

###### 1.3.1.2.4.2 等等。。。。

### 1.3.2 memcached

#### 1.3.2.1 缓存并发

#### 1.3.2.1 缓存漂移

## 1.4 消息中间件activeMQ

### 1.4.1 JMS规范

#### 1.4.1.1 点对点模型

##### 1.4.1.1.1 消息队列

##### 1.4.1.1.2 发送者

##### 1.4.1.1.3 接收者

#### 1.4.1.2 发布订阅模型

##### 1.4.1.2.1 主题

##### 1.4.1.2.2 发送者

##### 1.4.1.2.3 订阅者

### 1.4.2 activeMQ

## 1.5 多线程

### 1.5.1 线程基础

### 1.5.2 原子操作类

### 1.5.3 Lock、Condition、和AQS分析

#### 1.5.3.1 Lock接口

#### 1.5.3.2显示锁

#### 1.5.3.3读写锁

#### 1.5.3.4 Condition接口

#### 1.5.3.5 LockSupport工具类

#### 1.5.3.6 \*\*\*\*\*\*\*\*\*\*\*

### 1.5.4 并发工具类和并发容器

#### 1.5.4.1 容器

#### 1.5.4.2 Fock/\*\*\*\*

#### 1.5.4.3 并发工具类

#### 1.5.4.4 \*\*\*\*分析

### 1.5.5 线程池和Excutor框架

#### 1.5.5.1 线程池ThreadPoolExcutor

#### 1.5.5.2 Excutor框架

### 1.5.6 线程安全

### 1.5.7 实现原理和JVM-java内存模型

## 1.6 SSO

### 1.6.1 session跨域场景

### 1.6.2 spring-session

### 1.6.3 sso框架

## 1.7 高并发分流技术

### 1.7.1 lvs

### 1.7.2 apache

### 1.7.3 nginx

#### 1.7.3.1 nginx模块工作原理

##### 1.7.3.1.1 Handles（处理器模块）

##### 1.7.3.1.2 Filters（过滤器模块）

##### 1.7.3.1.3 Proxies（代理类模块）

#### 1.7.3.2 master进程

#### 1.7.3.3 nginx进程模型

##### 1.7.3.3.1 master进程

##### 1.7.3.3.2 worker进程

#### 1.7.3.4 Handles(处理器模块)

#### 1.7.3.5 Filters(过滤器模块)

#### 1.7.3.6 Proxies(代理类模块)

#### 1.7.3.7 worker进程

#### 1.7.3.8 负载均衡配置

#### 1.7.3.9 Ngnix配置详情

##### 1.7.3.9.1 负载均衡配置

##### 1.7.3.9.2 动静分离配置

##### 1.7.3.9.3 其它配置

#### 1.7.3.10 动静分离配置

#### 1.7.3.11 其它配置

## 1.8 数据库水平切分技巧

### 1.8.1 分库及db路由

#### 1.8.1.1 单点集群（Group）

#### 1.8.1.2 负载均衡策略(LB)

#### 1.8.1.3 读写分离

### 1.8.2 分表

#### 1.8.2.1 分表规则

#### 1.8.1.2 分表增删改查

### 1.8.3 MYSQL集群实践

## 1.9 系统监控

### 1.9.1 链路监控

Spring-cloud-sleuth+ zipkin

### 1.9.2 日志监控

### 1.9.3 代码性能监控

# 2 应用框架

## 2.1 springCloud

### 2.1.1 服务注册与发现Eureka

### 2.1.2 负载均衡器Ribbon

### 2.1.3 RPC工具Feign

### 2.1.4 熔断器 hystrix

### 2.1.5 网关 zuul

### 2.1.6 分布式配置中心spring cloud config

### 2.1.7 Docker

#### 2.1.7.1 docker构建微服务

#### 2.1.7.2 docker介绍

#### 2.1.7.3 docker安装

#### 2.1.7.4 docker常用命令

#### 2.1.7.5 docker私有仓库

#### 2.1.7.6 集群swarm

#### 2.1.7.7 集群部署springcloud微服务

## 2.2 springboot

### 2.1.1 读取配置信息

#### 2.1.1.1 @Configuration

#### [2.1.1.2 @ComponentScan](mailto:2.1.1.2@componentScan)

#### 2.1.1.3 @PropertySource

#### 2.1.1.4 @Value

### 2.1.2 spring event

#### 2.1.2.1 ApplicationListener

#### 2.1.1.2 ApplicationEvent

#### 2.1.1.3定义config启动

### 2.1.3 spring Aware

### 2.1.4 文件上传

## 2.2 Spring

### 2.2.1 MVC

#### 2.2.1.1 原理介绍

#### 2.2.1.2 controller调用

#### 2.2.1.3参数绑定

#### 2.2.1.4 messageConverter

#### 2.2.1.5 展示优化

##### 2.2.1.5.1 模块引擎

##### 2.2.1.5.2 动态页面的静态化处理

### 2.2.2 AOP

#### 2.2.2.1 aop原理

#### 2.2.2.2 java动态代理

#### 2.2.2.3 Cglib代理

#### 2.2.1.4 aop的坑

##### 2.2.1.4.1调用问题

##### 2.2.1.4.2接口还是类

### 2.2.3 事务处理

#### 2.2.3.1 声明式事务

#### 2.2.3.2 事务的传播属性

#### 2.2.3.3 事务的实现方式

#### 2.2.3.4 spring的连接处理

### 2.2.4 FactoryBean

##### 2.2.4.1 接口类的实例化

##### 2.2.4.2 FactoryBean的初始化

### 2.2.5 spring源码解读

## 2.3 持久化

### 2.3.1 Mybatis

#### 2.3.1.1 搭建

#### 2.3.1.2 分页实现

#### 2.3.1.3 代码生成

#### 2.3.1.4 自定义语句的实现

### 2.3.2 代码分层

### 2.3.3 Mybatis源码导读

#### 2.3.3.1 手写实现Mybatis

#### 2.3.3.2 ognl

#### 2.3.3.3 sqlsource

#### 2.3.3.4 cache

#### 2.3.3.5 DataSource

#### 2.3.3.6 Transaction

#### 2.3.3.7 resultHandle

#### 2.3.3.8 executor

#### 2.3.3.9 Sqlsession

# 3 搜索引擎

## 3.1 Lucene

### 3.1.1 Lucene组成及原理

### 3.1.2 构建索引

### 3.1.3 Lucene过程分析

### 3.1.4 高级搜索

### 3.1.5 扩展应用

### 3.1.6 管理及性能调优

## 3.2 solr

### 3.2.1 建立索引

### 3.2.2 文本分析

### 3.2.3 多语言搜索

### 3.2.4 分组&合并域

### 3.2.5 solr云

## 3.3 ElasticSearch

### 3.3.1 ES搜索

#### 3.3.1.1 结构化搜索

#### 3.3.1.2 全文搜索

#### 3.3.1.3 分布式搜索

### 3.3.2 索引&映射

### 3.3.3 分布式CRUD

### 3.3.4 索引管理

### 3.3.5 分片

### 3.3.6 搜索优化

# 4 性能优化

## 4.1 JVM优化

### 4.1.1 JVM原理剖析

### 4.1.2 JVM内存大小设置

#### 4.1.2.1 每个线程栈大小

#### 4.1.2.2 设置JVM最大堆内存

#### 4.1.2.3 设置年轻代大小

#### 4.1.2.4 设置持久代大小

### 4.1.3 垃圾回收器选择

#### 4.1.3.1 串行收集器

#### 4.1.3.2 并行收集器(吞吐量优先)

#### 4.1.3.3并发收集器(响应时间优先)

### 4.1.4 JVM服务参数调优实践

## 4.2 JAVA程序性能优化

#### 4.2.1 略

## 4.3 数据库优化

### 4.3.1 mysql优化

#### 4.3.1.1 省略

### 4.3.2 oracle优化

#### 4.3.2.1 oracle访问数据的存取方式

##### 4.3.2.1.1全表扫描

##### 4.3.2.1.2通过Rowid的表存取

##### 4.3.2.1.3索引扫描

#### 4.3.2.2 表之间的连接

##### 4.3.2.2.1 排序—合并连接

##### 4.3.2.2.2 嵌套循环

##### 4.3.2.2.3 哈希连接

# 5 工程化管理

## 5.1 maven

### 5.1.1 maven打包

### 5.1.2 jar依赖

### 5.1.3 jar冲突解决

### 5.1.4 jar发布

## 5.2 nexus

### 5.2.1搭建

### 5.2.2上传

### 5.2.3仓库管理

## 5.3 jenkins

### 5.3.1 自动化发布

### 5.3.2 自动化执行

## 5.4 git

### 5.4.1冲突的解决

### 5.4.2 提交过滤

### 5.4.3 git常用命令

### 5.4.4 git协同开发

### 5.4.5 GltHub介绍

### 5.4.6搭建本地仓库