JVM：技术点有启动参数的配置，垃圾回收器的选择、运行时的内存变化情况、运行时编译的汇编码查看

根据服务器的运行状况选择一个适合的jvm启动参数

JVM内存结构：method Area 、Heap、JVM Stacks、PC Register 、Native Method Stacks

JVM的垃圾回收期：应用于Heap中：SerialGC、ParallelGC、CMSGC、G1GC

内存模型 运行时数据区

本地方法栈

虚拟机栈

方法区

堆

程序计数器

执行引擎

本地库接口

本地方法库

NIO:同步非阻塞的I/O模型

三大核心部分：Channel管道，Buffer缓冲区，Selector传统的IO基于字节流和字符流进行操作

而NIO基于Channel和Buffer缓冲区进行操作，数据总是通过通道读取到缓冲区中，或者从缓冲区写入通道

Selector选择区 用于监听多个通道的时间（比如：链接打开，数据到达）。因此 单个线程 可以监控多个数据通道

简单的 管道示意图：

Channel

Channel

Client

Server

Buffer

Buffer

并发编程：技术点：乐观锁（volatile）在高并发环境下使用可能造成多次的循环判断，悲观锁过多的使用可能造成代码的

执行效率下降，因为会堵塞线程；降低锁的锁粒度，避免过多的锁粒度造成的线程压力过大，原子操作的实现

synchronized的应用。

Spring和ORM框架：技术点：SSM/SSH框架结构 spring+soringMVC+Mybatis/spring+springMVC+Hibernate

aop切面编程、ioc容器注入（DI）、注解注入和xml注入、一站式的模式可以在上面

使用开源框架和优秀的第三方库

事务：利用orm技术可以轻松的达到事务的控制：

M部分 SSM框架 S部分

Web

WebSocket Servlet

Web Portlet

Data Access/Integration

JDBC ORM

OXM JMS

Transactions

Massaging

Instrumentation

Aspects

AOP

代码容器

SpEL

Context

Core

Beans

Test

MySql数据库和SQL：

技术点：

MySql的事务：总结成ACID 原子，隔离性->一致性 / 持久性 mysql采用默认提交 的方式只要不适用start transaction 那么每个语句都会被当成一个事务 自动提交

锁：粒度：行级锁，表级锁

读写锁：排他锁（X锁）加了X锁其他不可以加任何锁，共享锁（S锁）加了s 锁其他可以加s锁不可以加X锁

意向锁：代表事务有意向操作这个表，意向锁可以更加容易支持多粒度封锁

分库分表：

技术点：

垂直分库：根据分的业务不同，将不同的数据属性分成不同的数据库，每个库可以在不同的服务器

垂直分表：将表中的数据类型拆分，减少单表的数据

水平分库：将数据库的内容分成不同的数据库中

水平分表：将单个表内容分成不同的的表单中 加快查找速度：

分库分表中间件：sharedingspere

RPC和微服务：

技术点：

rpc的网络实现方式：TCP\HTTP\UDP-----TCP支持保持连接，所以不需要每次调用都要三次调用

http需要每次调用都进行三次连接 安全系数更加高 对外更加有优势

rpc最主要的作用是进行服务调用、

微服务的作用是为了进行多团队配合开发， 是一个大型组织开发且维护的一个很好的架构方式

工作原理

注册

Registry

提交

Provider

提醒

启

动

反射

Consumer

Container

计数

Monitor

简单的rpc传输原理

客户端<---->Filter层<---->线程池<---->序列化<---

|

|

v

网络链接

^

|

|

服务器<---->Filter层<---->线程池<---->序列化<---

分布式缓存：

常用到的缓存中间件：Redis

缓存技术问题：缓存雪崩：缓存map突然消失，大量的请求达到服务导致突然大 量的sql

请求范文数据库，导致缓存雪崩

缓存击穿：在突然间一个m ap失效，一个大量的map打入请求 中，导致缓存被击穿

找不到合适的map返回，造成DB范文量加大，压力增大

缓存穿透：DB中不存在数据，每次都打入不存在的数据，一般属于 恶意攻击

淘汰策略：FIFO\LRU(久未使用)\LFU（最近最少使用）

分布式消息队列：

功能：应用解耦：将每一个模块的消息响应写入消息队列中，这样每一个模块就不必须等待模块响应成功

异步消息：将一些异步处理的信息移交给mq进行队列提交，将加快速度

流量削峰,实现高可用:高性能，：在前端加入消息队列，减少后端访问的人数，提出先进入消息队列的人

可伸缩的最终一致性架构

常见：rocketMQ、ActiveMQ、RabbitMq、ZeroMq、KafkaMQ

PUB公布者/SUB单点发送模式：

发布者

消息管理者

消费者

订阅

转发

发布

Kafka的 基础概念：Kafka通过Zookeeper管理集群配置，选举Leader，以及在Consumer Group发生变化时进行Rebalance。Producer使用push模式将消息发布到Broker，Consumer使用Pull模式从Broker订阅并消费消息。

kafka对外使用topic的概念，生产者往topic里写消息，消费者读消息。为了做到水平扩展，一个topic实际是由多个partition组成的，遇到瓶颈时，可以通过增加partition的数量来进行横向扩容,单个parition内是保证消息有序

RabbitMq的Exchange核心

Client2

Client1

Queue2

Queue1

Exchangge2

Exchangge1

Server2

Server1