1. [分布式缓存Redis+Memcached经典面试题和答案](http://youzhixueyuan.com/redis-memcached-interview-questions-and-answers.html)
2. **Redis相比memcached有哪些优势？**

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

1. **Redis支持哪几种数据类型？**

String、List、Set、Sorted Set、hashes

1. **Redis集群方案应该怎么做？都有哪些方案？**

1.twemproxy，大概概念是，它类似于一个代理方式，使用方法和普通redis无任何区别，设置好它下属的多个redis实例后，使用时在本需要连接redis的地方改为连接twemproxy，它会以一个代理的身份接收请求并使用一致性hash算法，将请求转接到具体redis，将结果再返回twemproxy。使用方式简便(相对redis只需修改连接端口)，对旧项目扩展的首选。 问题：twemproxy自身单端口实例的压力，使用一致性hash后，对redis节点数量改变时候的计算值的改变，数据无法自动移动到新的节点。

1. **Redis回收使用的是什么算法？**

LRU算法

1. **为什么要做Redis分区？**

分区可以让Redis管理更大的内存，Redis将可以使用所有机器的内存。如果没有分区，你最多只能使用一台机器的内存。分区使Redis的计算能力通过简单地增加计算机得到成倍提升,Redis的网络带宽也会随着计算机和网卡的增加而成倍增长。

1. **Redis的内存占用情况怎么样？**

给你举个例子： 100万个键值对（键是0到999999值是字符串“hello  
world”）在我的32位的Mac笔记本上 用了100MB。同样的数据放到一个key里只需要16MB， 这是因为键值有一个很大的开销。 在Memcached上执行也是类似的结果，但是相对Redis的开销要小一点点，因为Redis会记录类型信息引用计数等等。

1. **Memcached服务特点及工作原理是什么？**

a、完全基于内存缓存的

b、节点之间相互独立

c、C/S模式架构，C语言编写，总共2000行代码。

d、异步Ｉ/O 模型，使用libevent作为事件通知机制。

e、被缓存的数据以key/value键值对形式存在的。

f、全部数据存放于内存中，无持久性存储的设计，重启服务器，内存里的数据会丢失。

g、当内存中缓存的数据容量达到启动时设定的内存值时，就自动使用LRU算法删除过期的缓存数据。

h、可以对存储的数据设置过期时间，这样过期后的数据自动被清除，服务本身不会监控过期，而是在访问的时候查看key的时间戳,判断是否过期。

j、memcache会对设定的内存进行分块，再把块分组，然后再提供服务。

1. **如何实现集群中的session共享存储？**

Session是运行在一台服务器上的，所有的访问都会到达我们的唯一服务器上，这样我们可以根据客户端传来的sessionID，来获取session，或在对应Session不存在的情况下（session 生命周期到了/用户第一次登录），创建一个新的Session；但是，如果我们在集群环境下，假设我们有两台服务器A，B，用户的请求会由Nginx服务器进行转发（别的方案也是同理），用户登录时，Nginx将请求转发至服务器A上，A创建了新的session，并将SessionID返回给客户端，用户在浏览其他页面时，客户端验证登录状态，Nginx将请求转发至服务器B，由于B上并没有对应客户端发来sessionId的session，所以会重新创建一个新的session，并且再将这个新的sessionID返回给客户端，这样，我们可以想象一下，用户每一次操作都有1/2的概率进行再次的登录，这样不仅对用户体验特别差，还会让服务器上的session激增，加大服务器的运行压力。

为了解决集群环境下的seesion共享问题，共有4种解决方案：

1.粘性session  
粘性session是指Ngnix每次都将同一用户的所有请求转发至同一台服务器上，即将用户与服务器绑定。  
2.服务器session复制  
即每次session发生变化时，创建或者修改，就广播给所有集群中的服务器，使所有的服务器上的session相同。  
3.session共享  
缓存session，使用redis， memcached。  
4.session持久化

将session存储至数据库中，像操作数据一样才做session。

1. **memcached与redis的区别？**

1、Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。而memcache只支持简单数据类型，需要客户端自己处理复杂对象

2、Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用（PS：持久化在rdb、aof）。

3、由于Memcache没有持久化机制，因此宕机所有缓存数据失效。Redis配置为持久化，宕机重启后，将自动加载宕机时刻的数据到缓存系统中。具有更好的灾备机制。

4、Memcache可以使用Magent在客户端进行一致性hash做分布式。Redis支持在服务器端做分布式（PS:Twemproxy/Codis/Redis-cluster多种分布式实现方式）

5、Memcached的简单限制就是键（key）和Value的限制。最大键长为250个字符。可以接受的储存数据不能超过1MB（可修改配置文件变大），因为这是典型slab 的最大值，不适合虚拟机使用。而Redis的Key长度支持到512k。

6、Redis使用的是单线程模型，保证了数据按顺序提交。Memcache需要使用cas保证数据一致性。CAS（Check and Set）是一个确保并发一致性的机制，属于“乐观锁”范畴；原理很简单：拿版本号，操作，对比版本号，如果一致就操作，不一致就放弃任何操作

cpu利用。由于Redis只使用单核，而Memcached可以使用多核，所以平均每一个核上Redis在存储小数据时比Memcached性能更 高。而在100k以上的数据中，Memcached性能要高于Redis 。

7、memcache内存管理：使用Slab Allocation。原理相当简单，预先分配一系列大小固定的组，然后根据数据大小选择最合适的块存储。避免了内存碎片。（缺点：不能变长，浪费了一定空间）memcached默认情况下下一个slab的最大值为前一个的1.25倍。

8、redis内存管理： Redis通过定义一个数组来记录所有的内存分配情况， Redis采用的是包装的malloc/free，相较于Memcached的内存 管理方法来说，要简单很多。由于malloc 首先以链表的方式搜索已管理的内存中可用的空间分配，导致内存碎片比较多

[java架构师](http://youzhixueyuan.com/tag/java%e6%9e%b6%e6%9e%84%e5%b8%88)最全技能图谱上篇，包含：数结构算法、java进阶、web开发、框架与工具四大技能图谱。 下篇将包含大数据以及性能、设计模式、UML、中间件、分布式集群、负载均衡、通讯协议、架构设计等技术图谱等章节

1. [史上最全java架构师技能图谱（上）](http://youzhixueyuan.com/architect-skills-map-1.html)

[Java](http://youzhixueyuan.com/series/java)  [mikechen](http://youzhixueyuan.com/author/mikechen)  1年前 (2017-09-09)  365次浏览  [1个评论](http://youzhixueyuan.com/architect-skills-map-1.html#comments)  扫描二维码

[java架构师](http://youzhixueyuan.com/tag/java%e6%9e%b6%e6%9e%84%e5%b8%88)最全技能图谱上篇，包含：数结构算法、java进阶、web开发、框架与工具四大技能图谱。 下篇将包含大数据以及性能、设计模式、UML、中间件、分布式集群、负载均衡、通讯协议、架构设计等技术图谱等章节

本文作者，陈睿 优知学院创始人，前百度研发经理，携程定制旅游CTO。

优知学院是IT人在线进阶站,帮助IT人升职加薪，导师来自于BAT等一线互联网公司总监。提供系统的互联网产品技术进阶干货资料和课程，以及定期的线下实战活动。

1. **一：数据结构算法**



**算法分析**

时间复杂度和空间复杂度

**算法思想**

递推、递归、穷举、贪心、分治、动态规划、迭代、分枝界限

**数据结构**

数组、链表、堆、栈、队列、Hash表、二叉树等

**算法**

排序

经典排序：插入排序、冒泡排序、快排（分划交换排序）、直接选择排序、堆排序、合并排序等

查找

经典查找：顺序查找、二分查找、二叉排序树查找

**高级算法**

贪婪

回溯

剪枝

动态规划

**大数据算法**

hash分桶

统计

1. **二：Java进阶**



**java编程基础：**

对象和类 、基本数据类型 、变量类型、运算符、循环分支结构、数组、正则表达式等

**集合**



总的说来，Java API中所用的集合类，都是实现了Collection接口，他的一个类继承结构如下：

Collection<–List<–Vector

Collection<–List<–ArrayList

Collection<–List<–LinkedList

Collection<–Set<–HashSet

Collection<–Set<–HashSet<–LinkedHashSet

Collection<–Set<–SortedSet<–TreeSet



**面向对象高级知识**



类、对象、继承、构造函数、封装、接口、抽象类、多态、重写、this static关键字、类与对象的关系

**异常处理**

异常类类图：throwable exception error RuntimeException

异常处理机制

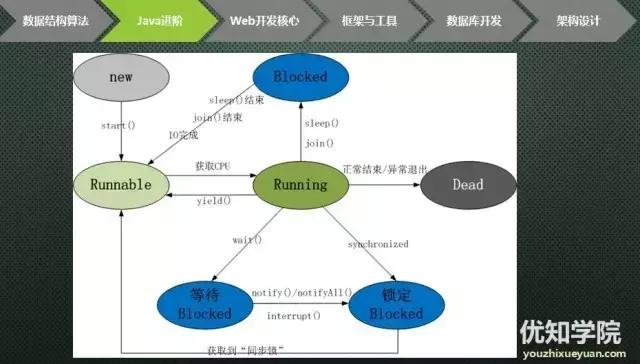
如何定义和使用异常

运行时异常和受检查异常区别

运行时错误

java异常处理的原则和技巧

**多线程**



概念与原理

创建于启动

线程的生命周期及五种基本状态

线程交互

死锁

调度合并

调度让步

调度休眠

同步方法

同步块

同步与锁

线程池

阻塞队列

**IO/NIO**

同步阻塞 同步非阻塞 异步IO

**反射**

**序列化**

**泛型**

**网络编程**

**高级特性**

**JVM**

**运行时数据区**：方法区、虚拟机栈、本地方法栈、堆、程序计算器

**GC算法：**

内参回收三要素：什么内容需要回收、什么时候回收、如何回收

并发与执行

引用计数算法

根搜索算法

垃圾回收算法：标记-清楚算法 复制算法 标记-整理算法 分代手机算法

垃圾收集器：新生代、老年代收集器

**溢出**

java堆溢出

方法区溢出

outofmemoryerror

虚拟机栈和本地方法栈溢出

直接内容溢出

1. **三：Web开发核心**



**HTML JS CSS**

html js css语法基础

Js css框架

Html开发工具

JS和CSS调试工具

**模板引擎**

jsp

velocity

freemarker

**Java web**

容器：tomcat jetty等

热部署插件：run-jetty-run

cookie session使用和区别

fliter和listener的启动和步骤

身份验证

单点登录原理以及实现

**web核心**

事物JTA

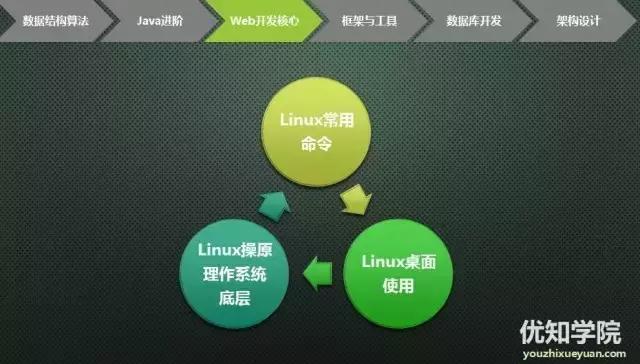
JMX

安全：JCCA/JAAS

通信:JNDI/JMS

SSI技术

**linux**



常用命令以及操作系统原理等

**线上故障处理和分析**



**性能工具**

visualVM Jprofiler JMeter等

**线上故障**

线程数超标

访问超时

长事务

CPU超标

内存超标

**开发工具使用**



**web开发调试**

firebug

Web Developer

JavaScript Debugger

IETester

Yslow

**构建工具**

maven Grails

maven私服 nexus

**版本控制**

git svn

**java调试工具**

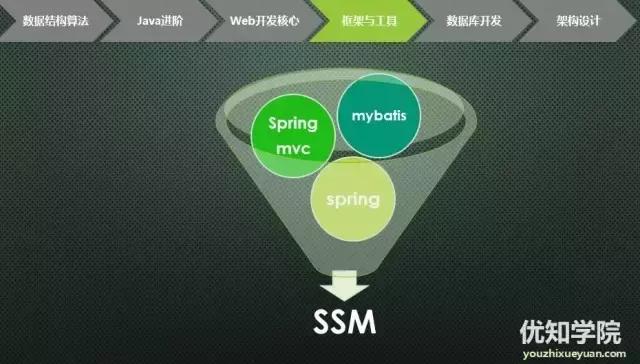
JCover

Junit

Jtest

以及大量的eclipse插件，eg：findbugs等

**开发框架**



SSH：struts2+spring+hibernate

SSM:springmvc+spring+mybatis

**阿里开源框架**



1. **史上最全java架构师技能图谱（下）**

**架构师进阶六大要求**



程序设计要求

WEB开发要求

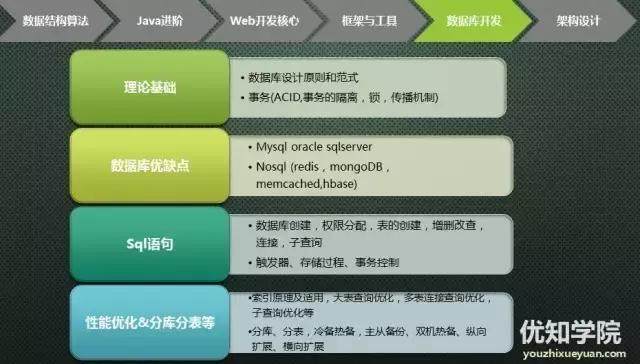
架构设计要求

运维能力要求

操作系统要求

产品业务要求

数据库技能



理论基础

数据库设计原则和范式

事务(ACID,事务的隔离，锁，传播机制)

数据库优缺点

Mysql oracle sqlserver

Nosql (redis，mongoDB，memcached,hbase)

Sql语句

数据库创建，权限分配，表的创建，增删改查，连接，子查询

触发器、存储过程、事务控制

性能优化&分库分表等

索引原理及适用，大表查询优化，多表连接查询优化，子查询优化等

分库、分表，冷备热备，主从备份、双机热备、纵向扩展、横向扩展

**架构设计**



设计模式与UML建模

Java设计模式

JAVA与UML建模

面向服务和资源架构

SOA/SCA/ESB/OSGI/EAI，微服务架构

面向资源架构：ROA/REST

高性能和负载均衡

性能调优

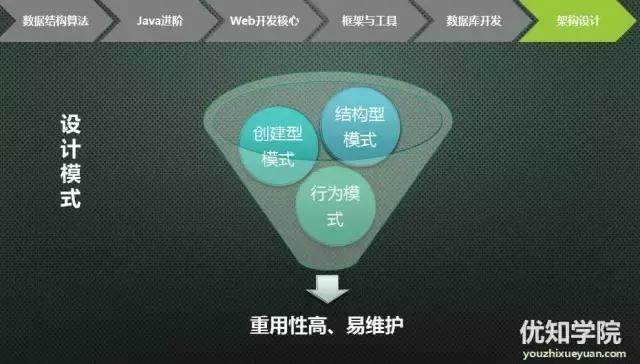
大型网站负载均衡、系统调优等

大型网站架构

集群

架构设计 容灾

设计模式



结构型模式

创建型模式

行为模式等

重用性高

易维护

**技术架构能力**



缓存搭建

CDN基站搭建

消息队列搭建

数据库拆分

工程代码拆分

SOA服务

分布式文件

搜索引擎

Sql和Nosql混搭

运维系统

监控系统

机房容灾

**架构前瞻性**



业务和技术的把控能力

技术挑战

预判业务

淘宝店铺装修项目大数据实战



如果你对java架构有兴趣可以参加优知学院的java进阶实战，java进阶群：629511325

淘宝店铺装修大数据实战

业务场景：涉及淘宝大数据业务的访问，以及大量的读写业务场景。

技术场景：涉及分布式场景(硬软件集群和负载均衡等)、dubbo(SOA)、消息队列(ActiveMQ)、分布式缓存Redis、分布式存储(TFS)等技术场景。

架构设计

详细技术方案

数据库设计

表结构搭建

搭建SSM框架

数据库以及开发环境搭建

搭建分布式缓存和分布式文件存储系统

编码阶段