一.计算机网络面经
1.多层网络结构
2.各个层的作用 以及其中的某些协议
3.TCP三次握手和四次挥手
4.TCP和UDP协议的区别
5.TCP协议保证可靠性
6.ARQ协议
7.网页显示的过程涉及到的协议
8.HTTP的一小部分状态码（无聊的背面经）
9.HTTP协议中的有关内容
10.还有很多的内容，可以等到后面再看，现在这样回忆了一边计网的知识
二.linux基础知识
三.数据库
1.InnoDB和MyISAM的相同与区别
2.注解：事务性 ACID 包含并发数据库的各种问题 脏读、脏写、幻读、不可重复读
3.字符集及校对规则
4.索引
5.事务
6.大表优化
7.Redis数据库

一.计算机网络面经

1.多层网络结构

OSI TCP/IP网络层次结构，从下到上描述

OSI：物理层、数据链路层、网络层（包的分发，路由建立等）、传输层、会话层、表示层（数据加密、压缩等内容）、应用层

TCP/IP：网络接口层、网络层（IP协议）、传输层（TCP UDP协议）、应用层

为了简化问题，一般使用五层网络结构：物理层、数据链路层、网络层、传输层、应用层

2.各个层的作用 以及其中的某些协议

传输层：两台主机之间的通信提供通用的数据传输服务，TCP UDP均是传输层协议。

网络层：数据传输时，会经过很多子网和数据链路，网络层是为了选择合适的网络路由和交换节点，会把用户的数据封装为数据包发送，IP分组也叫IP数据报。

数据链路层：两个节点之间传输数据时，需要将IP数据报划分为数据帧，从而在两台主机之间的链路上传递。这里有差错控制，（我们的计网主要学习的内容就是来自于这里）

3.TCP三次握手和四次挥手

这个是面试经常问的内容，需要背诵下来。

三次握手的目的：双方确认自己与对方的发送和接收是正常的。

也就是二者均有一个互相的来回过程，保证了通信的可靠性。

四次挥手的细节：

一共有四次?

客户端 FIN 服务器 关闭客户端到服务器的通道（应该还带有一个序号才对）

服务器收到FIN 返回ACK 确认序号为收到序号加一 （确定通道关闭）

服务器发送一个FIN 关闭服务器到客户端的连接

客户端发送ACK返回，并且把确认序号加一

4.TCP和UDP协议的区别

UDP：无连接，不可靠，远方主机在收到UDP报文后，不需要返回内容。一般用于即时性要求比较高的情况。

TCP：面向连接，一般用于准确性要求比较高的情况。

5.TCP协议保证可靠性

包排序并且编号、校验和（保持首部和数据部分的校验和）、丢弃重复数据、流量控制（大小可变的滑动窗口机制）、拥塞控制（慢开始、拥塞避免、乘法减小、快重传、快恢复）

拥塞出现以后，直接把ssthresh的数值（慢开始门限）设置为当前拥塞窗口的一半，并且慢开始。

如果遇到快重传，也设置为一半，但是窗口从门限开始，进行拥塞避免算法。

接收到一个不按顺序的包，会发一个重复确认，如果出现了三个重复确认，它会重传指定的数据段，并且进行快恢复算法。

ARQ协议：每次发送一个包，就停下来等待下一个包。

超时重传：发出TCP包，但是没有收到ACK回应，超过定时器以后，会将这个包重传。

这里是有很多机制的。

6.ARQ协议

等待式ARQ协议和连续ARQ协议。

确认重传+超时机制

等待式就是发送一个就在那里等着ACK，速度很慢。

连续式ARQ 涉及到了回退N帧的情况，倘若出错，会回退N帧重传，效率不好说是高效还是低效。一般都涉及到一个发送窗口的内容。

7.网页显示的过程涉及到的协议

域名-DNS变为ip地址-发送http请求-发挥html响应-浏览器解析并且展示

ARP协议专门用于将IP地址转换为MAC物理地址，是数据链路层通信的目标地址。

8.HTTP的一小部分状态码（无聊的背面经）

1XX 信息性状态码 接受的请求正在处理

2XX 成功 请求正常处理完毕

2XX 重定向 需要执行附加的操作，以此来完成请求

4XX 客户端错误码 服务器无法处理请求，请求有误

5XX 服务器错误码 服务器处理合法请求时，出现错误

9.HTTP协议中的有关内容

TCP 生成报文段，将HTTP请求分割成多个报文段

IP：分为IP数据报

数据链路层：数据帧按照物理MAC地址进行处理。

HTTP长连接、短连接

HTTP短连接：在1.0中默认使用短连接，每一次HTTP操作都会建立一次HTTP连接。

HTTP长连接：Connection:keep-alive

这一个连接不会关闭，会在使用过程中一直保持着。

本质上是TCP的长连接和短连接。我们看TCP是否会关闭连接来区分是否为长连接。

HTTP保存状态的方法：HTTP是一种不保持状态的协议，不对请求和响应之间的通信状态进行保持，我们使用Session机制来弥补不保持状态的问题。

服务器给特定的用户在redis数据库创建特定的session，可以保持一段时间的活性，当这段时间过了以后，就会销毁session。

大部分情况下，在cookie中附加一个session id来跟踪。

若cookie被禁用，url重写把 session id直接加在url路径后面的。

cookie和session的关联和区别？

cookie存储在客户端（浏览器端），保存用户的基本信息。一般的网站在登录时，会存放一个token在cookie中，下次登陆时，直接搜索即可。每次登录后，这个token都会更新一下。

session是为了应对http无状态缺陷的，通过服务端记录用户的状态，专用于弥补无状态协议没有身份认证的问题。

session一般存储在云服务器上面，session的安全性更高。

token一般是加密的？

URL：Uniform Resource Identifier 统一资源定位符，专门用于定位资源的location

URI：Uniform Resource Location 统一资源标识符，唯一标识一个资源

URL是一种具体的URI

HTTP 使用端口80

HTTPS 使用端口443 运行在SSL/TLS协议上方，具有一定的安全性。

10.还有很多的内容，可以等到后面再看，现在这样回忆了一边计网的知识

二.linux基础知识

这个面试会考吗？暂时是不太清楚的，linux基础知识看一下总没有坏处。

操作系统内核：文件管理、虚拟内存、设备I/O等知识。

Linux命令：

pwd 显示当前所在目录

grep 字符串 文件 --color 在文件中搜索对应的字符串

ps -ef / ps -aux 查看当前系统正在运行的进程

ps -aux | grep redis 查看当前名字含有redis的进程

kill -9 进程id 强制杀死进程

netstat -an 查看当前的端口使用情况

三.数据库

主要包括有两种数据库，MySQL和Redis，这两个常用的东西我的了解都不多，为了防止暴雷，是时候认真看一看了。

只会写一些简单的SQL语句，现在来看是不太行的。

1.InnoDB和MyISAM的相同与区别

在以前，MyISAM是MySQL的数据库引擎，性能极佳，但是不支持事务和行级锁，最大的缺陷是崩溃以后无法恢复。

5.5版本之后，MySQL引入了InnoDB，事务性数据库引擎。

2.注解：事务性 ACID 包含并发数据库的各种问题 脏读、脏写、幻读、不可重复读

事务是把整个处理流程抽象为一个统一的整体，多个操作要么同时成功，要么同时失败。也就是数据库处理事务中的多条语句时，需要保证操作的有效性，不会出现事务错误的情况。

事务具有几个特性：

Atomicity 原子性，操作无法被中断，要么全部成功，要么全部失败

Consistency 一致性，数据不会出现别的内容，两方内容要互相对应

isolation 隔离性，没有真正决定时，绝对不会真正执行。

durability 持久性，发生了数据库崩溃、停电等故障时，再次重启后，能够完美恢复之前的故障。

事务会把数据库从一种一致的状态转换为另一种一致的状态。transaction，事务处理

InnoDB 仅支持事务和行级锁。

行级锁大幅度提高了并行操作的速度，但是，这里的行级锁当且仅当WHERE的主键是有效的，非主键的WHERE仍然会在INSERT/UPDATE等操作时将整个表锁定。

innoDB对于事务的支持：

redo日志和undo日志

undo日志：记录事务开始前的原始数据内容 不要做==回滚

undo日志存放的空间叫回滚段，存储在表空间里面。

对于update和delete会存放数据的旧记录，对于insert会存放新数据的行的标记，当事务需要回滚时，可以利用undo日志进行处理。也就是反向的update即可。

redo日志：重做日志，记录的是事务运行时，对数据页的改动。

某一个sql修改了一行记录，redo日志会记录这一行数据所在的物理页以及修改的内容。对于各种索引他都会记录内容。（记录的是数据在物理页面上的直接改动，本身采用顺序读写的方式，效率是很高的）

crash-safe WLA技术 Write-Ahead Logging

当事务提交时，必须先把事务的所有日志写入到redo里面，进行持久化，事务的commit才算完成了。

redo是故障恢复和前滚。

undo是事务回滚，恢复事务开始前的快照。

对于事务来说，一致性是最为重要的属性，是需要重点关照的。

事务的原子性实现：所有操作写入日志，如果一部分操作成功，另一部分失败。那么只需要回溯日志，将已经操作成功的东西撤销，即可完成全部操作失败的目的。

隔离性：使用加锁的方式，这里的锁就和操作系统里面多线程共享变量的锁是一样的。

互斥锁影响效率，有了共享锁和排它锁。

排它锁：读写不并行，写写不并行

共享锁：读读可以并行

mvcc：快照解决读写并发问题

写发生时，数据克隆一份。写任务操作克隆好的数据，读仍然读旧的原始数据。

数据库读写的几个问题：

脏写：A和B更新同一条数据，A更新为A，B更新为B。而由于A是先更新的，undo日志记录的仍然是更新之前的值，倘若A事务失败回滚，就会把值恢复为原来值。

对于B来说，它的更新就不见了，明明写了一个数据，但是过了一会，这个数据不见了。

脏读：A更新，B使用A的数据操作，但是A又回滚了，B直接傻眼了。

不可重复读：事务A在读写期间，事务B提交了对某个数据的修改，事务A会读到不同的数据。

都是由于一个事务取更新或者查询了另一个还没有提交的事务的数据，此时一旦发生回滚，查询到的数据就不见了。

此时我们想到了

幻读：在读取时，用一个事务一样的SQL语句多次查询，每次查询都发现内容发生了变化。

但这种变化难道不应该是正常的吗？每次查询都发现了一些别的内容。

这里的多种加锁不加锁的内容，想要解决这些问题，写一定要加锁，保证无法被处理，就可以处理掉脏问题。

innodb解决幻读：

记录锁：锁定记录，无法被别的事务读取、修改。

间隙锁：一条事务处理一个范围内的数据时，触发间隙锁，不可以被别的事务读取、修改。

这一点总是恒定的。

3.字符集及校对规则

字符集从二进制编码到某类字符符号的映射。 编码方式?

校对规则是某种字符集下的排序规则。

4.索引

索引的主要数据结构有BTree和哈希索引。

哈希索引：低层数据结构是哈希表，当大多数需求为单条记录查询时，使用哈希表。其余场景推荐使用BTree。

B+树 结构可以大致上了解一下？问到了就说不会就行了。

5.事务

事务是逻辑上的一组操作，要么都执行，要么都不执行。

MySQL InnoDB支持 可重读，有可能引发幻读。

6.大表优化

单表记录过度，数据库的CRUD性能明显下降。

1.限定数据范围，查询语句限制数据范围。

2.读写分离，主库负责写，从库负责读

3.垂直分区，将数据表拆分成多个不同的内容哦。

4.水平分区，把一张表分割成多张相同字段的表，水平拆分最好分成多个数据库。

这里涉及到了很多锁的东西，大致能了解一点东西即可，学到了一点点并发数据库的内容也算。

7.Redis数据库

这个就是把数据存储在内存里，当做缓存数据库来处理，最大程度上提高系统的运行效率。

多加了一个缓存数据库的内容，这个数据库的信息是复杂的。