

面试问题一：

1:自我介绍

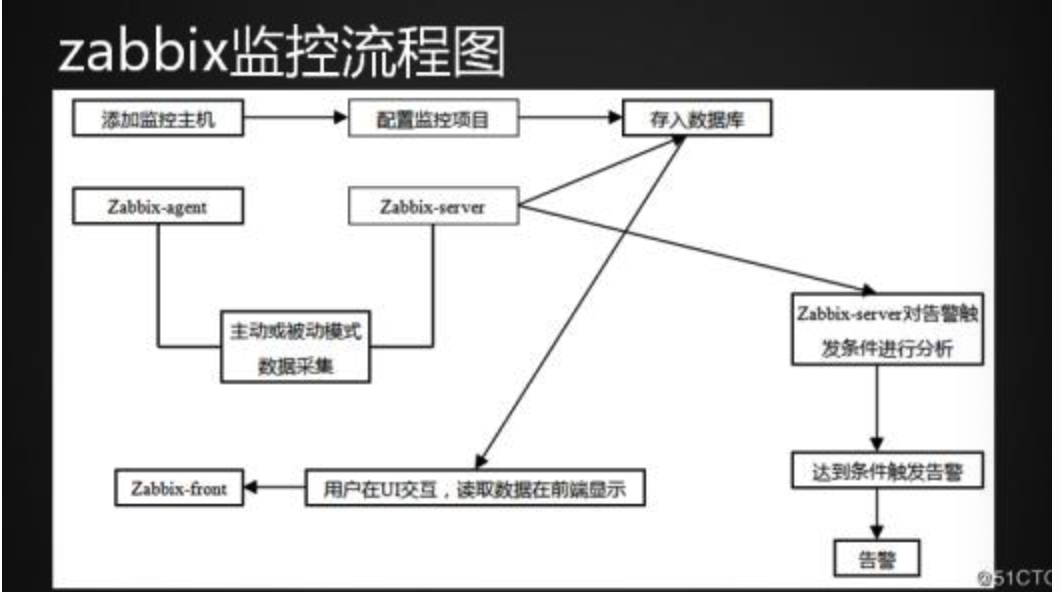
2:工作内容

3:你有用过哪些监控

Zabbix+granfra

Promithu+granafa

4:zabbix是怎么实现监控这个流程的



监控步骤：

一：首先要将监控的主机添加到主机组，可以采用自动发现的方式

二：添加应用集（aplication),可以采用默认的，也可以自己创建

三：添加监控项，可以使用自带的监控项，也可以自定义

四：告警粗发器配置

五：配置告警动作

5:zabbix是怎么实现报警的

自定义的监控项是不会触发报警的。

1:首先是配置好报警媒介类型，方式可以采用邮件，叮叮，微信

2:创建触发器规则（当监控项达到阀值时，便会执行动作）

3:创建动作（一般是执行脚本实现报警）

6:zabbix的数据收集，日志收集，你是用的什么

ELK负责日志收集管理

7:ELK是一套什么东西

ELK是一套解决方案

Elasticsearch:负责日志检索和储存

Logstash:负责日志的收集和分析，处理

Kibana:负责日志的可视化

8:当磁盘满了的时候，你是怎么获取到的这个信息

通过在被监控的机器安装zabbix客户端，收集到磁盘的使用量数据，发送给服务端，在zabbix的web页面上，可以配置磁盘的监控项可以设定多种阀值触发报警，例如80一般严重，90严重，95灾难。

9:docker,k8s了解吗

docker有在测试环境上用过，k8s当时企业还没用上

10:docker怎么查看容器的信息

当时答的是docker ps(这是是获取运行的容器信息）

获取所有的容器信息采用的是docker log

11:当一个服务崩溃了，你会怎么去排查

当某一个服务突然崩溃了，首先查系统日志，根据日志查看崩溃的原因，再逐一排查

一般先检查下是否是内存不够的问题

12:python中装饰器，生成器，迭代器解释一下

1:装饰器其实是闭包的一个典型应用，简单来说在不改变原函数代码基础上，在执行前后进行定制操作。最大的好处是解决代码冗余问题。主要应用于路由，缓存，内置用户登陆

2：生成器是一种特殊的迭代器，内部主要使用了yiled方法，主要用于创建大量数据的列表，随取随用，大量节省了内存，用yiled返回结果

3:迭代器 它主要使用了\_\_next\_\_和\_\_iter\_\_方法，迭代器不会一次性把所有元素加载到内存，而是需要的时候使用return返回结果

13:python算法知道哪些，讲一个你熟悉的

快速排序，堆排序，希尔排序

堆排序：如果是从小到大排列，首先创建一个大顶堆，保证每个父节点的值都比左右节点的值大

将大顶堆的根结点与堆数组中的最后一个元素进行交换，交换后，大顶堆的根结点存放的就是堆数组中的最后一个元素，大顶堆的根结点中存储的原始的最大值被移走了；接着，将剩下的n-1个元素重新调整后，构造成一个新的大顶堆，重复上面的步骤，被移动的元素就构成了一个有序的数据

14：python中线程池和进程池的区别

主要还是线程和进程的区别。

* 进程是资源分配的基本单位，线程是CPU执行和调度的基本单位；
* CPU上真正执行的是线程，线程比进程轻量，其切换和调度代价比进程要小；
* 通信/同步方式不一样
  + 进程：
    - 通信方式：管道，FIFO，消息队列，信号，共享内存，socket，stream流；
    - 同步方式：PV信号量，管程
  + 线程：
    - 同步方式：互斥锁，递归锁，条件变量，信号量
    - 通信方式：位于同一进程的线程共享进程资源，因此线程间没有类似于进程间用于数据传递的通信方式，线程间的通信主要是用于线程同步。
* 一个进程可以包含多个线程，线程之间共享进程的资源（文件描述符、全局变量、堆空间等），寄存器变量和栈空间等是线程私有的；

线程池：

先启动若干数量的线程，并让这些线程都处于睡眠状态，当客户端有一个新请求时，就会唤醒线程池中的某一个睡眠线程，让它来处理客户端的这个请求，当处理完这个请求后，线程又处于睡眠状态。

进程池：

当需要创建的子进程数量不多时，可以直接利用multiprocessing中的Process动态成生多个进程，但如果是上百甚至上千个目标，手动的去创建进程的工作量巨大，此时就可以用到multiprocessing模块提供的Pool方法。

初始化Pool时，可以指定一个最大进程数，当有新的请求提交到Pool中时，如果池还没有满，那么就会创建一个新的进程用来执行该请求；但如果池中的进程数已经达到指定的最大值，那么该请求就会等待，直到池中有进程结束，才会创建新的进程来执行

15：你在哪里用到了线程池，为什么要用线程池，结合你实际用到的场景说一下

场景：服务端的程序经常会遇到来自客户端传入的短小的任务，执行时间短暂，工作内容单一，并且需要快速的返回结果，如果服务器每次接收一个任务创建一个线程，然后进行执行，这在任务少的时候不错，如果有成千上万的任务递交给服务器，服务器将会创建成千上万的线程，创建成千上万的线程将会带给服务器巨大的压力，大量线程频繁的切换上下文将增加系统的负载和大量资源的开销，同时创建的线程也不方便管理，通过线程池技术可以很好的解决这个问题，预先创建若干线程，并且将线程的创建控制权交给线程池，重复使用线程，减小了服务器的压力，降低了资源的开销和使用，降低了系统的负载，提高相应速度，方便管理线程进行统一的分配调优和监控。

CPU密集型：程序需要占用CPU进行大量的运算和数据处理；

I/O密集型：程序中需要频繁的进行I/O操作；例如网络中socket数据传输和读取等；

由于python多线程并不是并行执行，因此较适合与I/O密集型程序，多进程并行执行适用于CPU密集型程序；

16:全局锁解释一下，为什么会有全局锁

提起python多线程就不得不提一下GIL(Global Interpreter Lock 全局解释器锁)，这是目前占统治地位的python解释器CPython中为了保证数据安全所实现的一种锁。不管进程中有多少线程，只有拿到了GIL锁的线程才可以在CPU上运行，即时是多核处理器。对一个进程而言，不管有多少线程，任一时刻，只会有一个线程在执行。对于CPU密集型的线程，其效率不仅仅不高，反而有可能比较低。python多线程比较适用于IO密集型的程序。对于的确需要并行运行的程序，可以考虑多进程。

多线程对锁的争夺，CPU对线程的调度，线程之间的切换等均会有时间开销。

17:django的中间件你有用过哪一些

https://www.cnblogs.com/fu-yong/p/8780850.html

18:diaogo的优化有没有做过，前端怎么优化，后端怎么优化

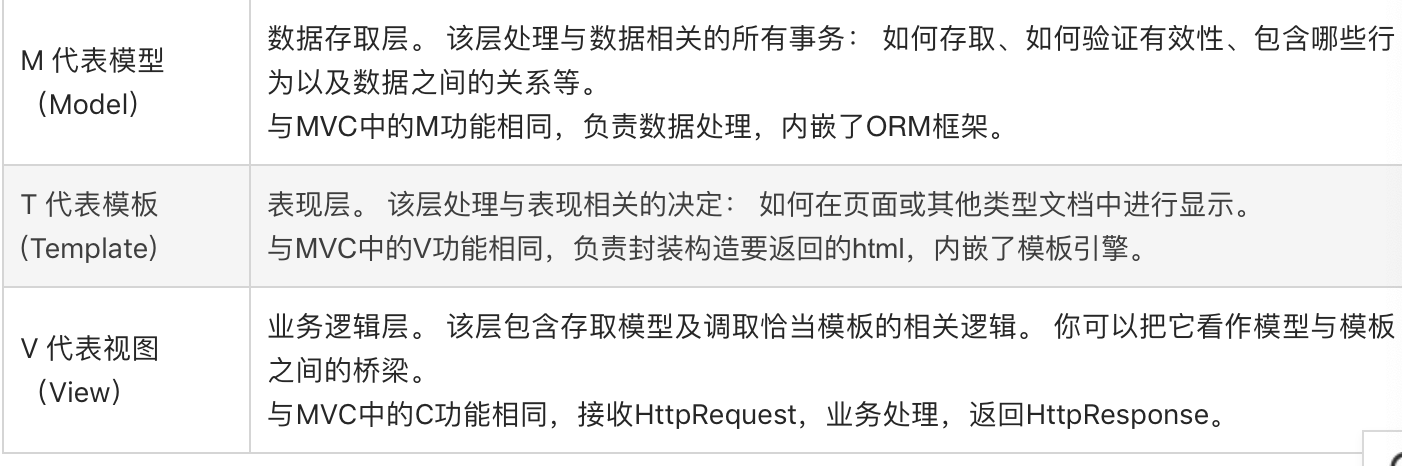
<https://www.cnblogs.com/yunxintryyoubest/p/11028691.html>

https://www.cnblogs.com/yunxintryyoubest/p/11028691.html

19:用户发出一个请求，django是怎么接收的

1. wsgi：创建socket服务端，用于接收用户请求并对请求进行初次封装  
2. 中间件：对所有请求到来之前，响应之前定制一些操作  
3. 路由： 匹配路由，在url和视图函数对应关系中，根据当前请求url找到相应的函数  
4. 视图函数：执行视图函数，业务处理【通过ORM去数据库中获取数据，再去拿到模板，然后将数据和模板进行渲染】  
5. 在经过中间件  
6. 通过wsgi将响应返回给用户

20:解释一下MTV



21:你用django做了一个什么系统平台

CMDB二次开发，运维平台二次开发，监控平台二次开发。

其中包括自动化服务部署（包括nginx,tomcat,zabbix,mysql,redis等等）

业务数据整合显示

服务器查询

权限管理

22:linuxd的内核优化有没有做过，具体是优化什么

当然有啊，主要是在linux的两个文件里做修改，一个sysctl.confl.一个是limits.conf

主要优化的是tcp连接数，以及限制最多打开的文件和进程数

23:前端的框架你有用过哪些？有没有用过vue

Bootstrap,没有用过vue，但是正在学

24:tcp和udp的区别

1 TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接

2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付

Tcp通过校验和，重传控制，序号标识，滑动窗口、确认应答实现可靠传输。如丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。

3、UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

4.每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5、TCP对系统资源要求较多，UDP对系统资源要求较少。

25:tcp三次握手和四次挥手的区别

省略

26:四次挥手中为什么会有time\_wait，且比三次握手要多一次

* 原因一：保证TCP全双工连接的可靠释放
* 原因二：为了使就得数据包在网络中因过期而失效

确保数据能够完整传输。

当被动方收到主动方的FIN报文通知时，它仅仅表示主动方没有数据再发送给被动方了。

但未必被动方所有的数据都完整的发送给了主动方，所以被动方不会马上关闭SOCKET,它可能还需要发送一些数据给主动方后，

再发送FIN报文给主动方，告诉主动方同意关闭连接，所以这里的ACK报文和FIN报文多数情况下都是分开发送的。

因为在断开连接的过程中，服务端不会立马断开连接，需要客户端发

https://blog.csdn.net/weixin\_41966991/article/details/81264095

27：工作过程中有经常遇到过哪些故障

网络类型： TCP mysql主从延迟

磁盘IO： 存储

业务： 运维工程师 处理业务服务器里面的 IO wait

自己开发的CMDB等 ： 数据的读取和写入 IO mysql 读写分离 、 缓存

集群架构：

代码Bug

业务上：如果代码出问题。CI、CD 突然 代码回滚出问题？ 平台？ ansible puppet

自己写的CMDB、运维平台等等

1. 初期 使用人员不多
2. 平台越来成熟、 怎么优化
3. 安全？ 平台权限、 是否可以外网访问？ 内部的平台 》》操作生产服务器

比如页面访问太慢。支付页面弹不出来，mysql主从复制延迟？内存溢出

28:CICD流程讲一下，并且使用ansible部署更新了多少台服务器

主要利用到了gitlab，jenkins和ansible

首先开发者上传代码到gitlab，

通过配置好触发式构建的方式，jenkins会自动拉取代码

之后会进行流水线样式的操作，编译构建，失败的话通知运维，成功的话通过编写pupple脚本

执行anbile自动化更新操作

29:jenkins主要做了什么

jenkins主要做了代码编译，构建，执行pupple脚本等操作

30:你做的监控平台监控了哪些指标

服务器性能

例如cpu负载

内存，磁盘，网卡等等

Nginx,mysql,redis等等其中的重要参数

以及一些业务中需要监控的参数