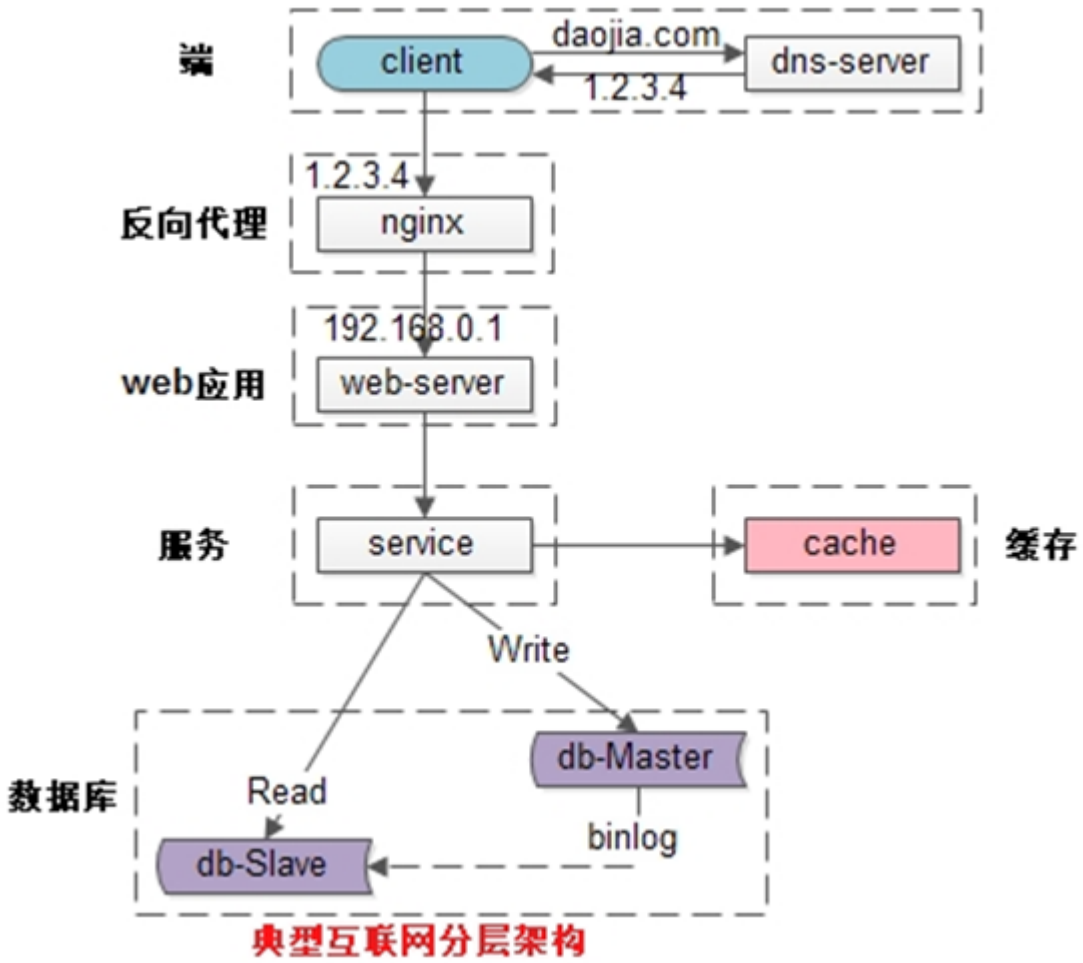
- 
端
client
daojia.com
dns-server
1.2.3.4
1.2.3.4
反向代理
nginx
192.168.0.1
web应用
web-server
服务
service
cache
缓存
Write
db-Master
数据库
Read
binlog
db-Slave
典型互联网分层架构

Django 本身提供了 runserver，为什么不能用来部署?

2018年12月09日 09:59:44 无敌.. 阅读数：757

runserver 方法是调试 Django 时经常用到的运行方式，它使用 Django 自带的

WSGI Server 运行，主要在测试和开发中使用，并且 runserver 开启的方式也是单进程 。

uWSGI 是一个 Web 服务器，它实现了 WSGI 协议、uwsgi、http 等协议。注意 uwsgi 是一种通信协议，而 uWSGI 是实现 uwsgi 协议和 WSGI 协议的 Web 服务器。uWSGI 具有超快的性能、低内存占用和多 app 管理等优点，并且搭配着 Nginx

就是一个生产环境了，能够将用户访问请求与应用 app 隔离开，实现真正的部署 。相比来讲，支持的并发量更高，方便管理多进程，发挥多核的优势，提升性能。

思考：从客户端访问一个数据流程，要经过哪些环节，我们可以思考在每一个环节都想办法做一些优化，加强网站的性能。可以以flask项目访问流程为例，或者Django项目的访问流程为例来参考。

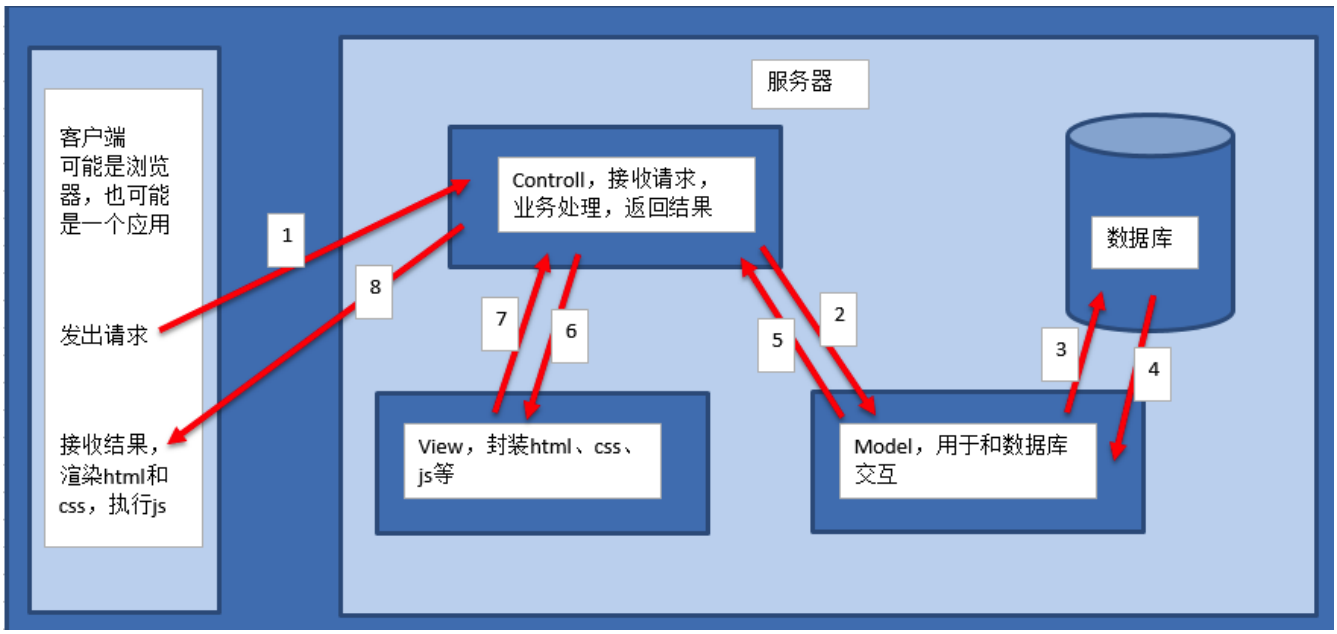
微服务

https://www.cnblogs.com/wintersun/p/6219259.html

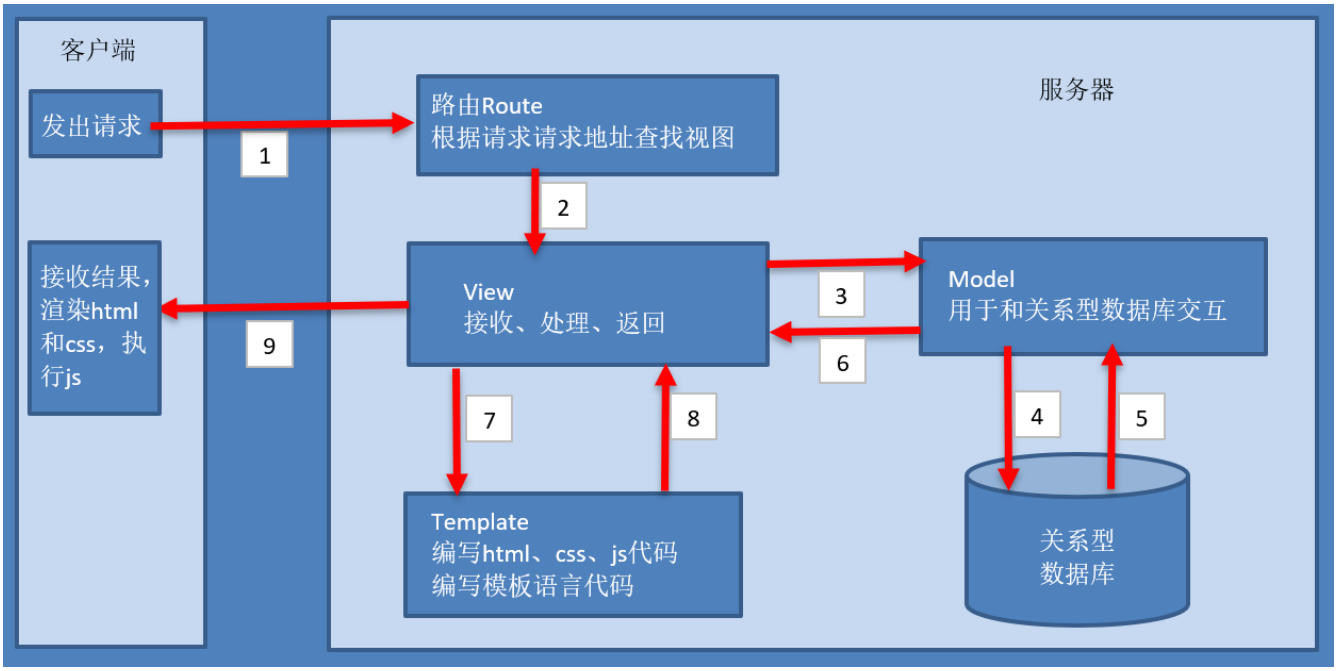
MySQL数据库引擎:

https://www.cnblogs.com/xujishou/p/6343431.html

MVC:



MVT:

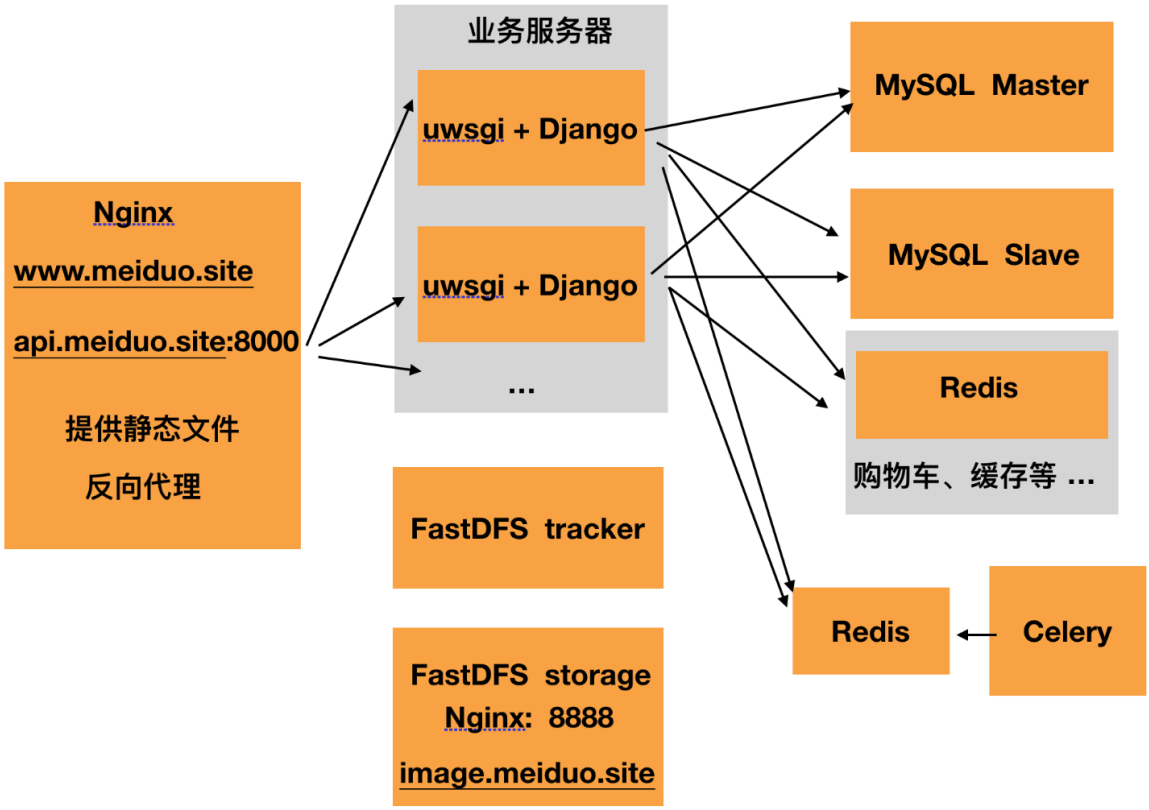


主要的工作是实现产品的原型, 没有太多的考虑架构, 使用Django来快速实现功能, DB的表结构设计好之后, 抽象出功能View, 由于产品设计也很不完善, 后端需要很多的预留设计, 避免产品逻辑的变更带来整个表结构的变动, 在这个阶段代码上最重要的是确定适合团队的代码规范, 代码检查规则.

Nginx:

Nginx负责负载均衡, 分发流量到多个Django服务

数据不经常改变的页面，做静态化，Nginx直接返回静态页面，不用请求数据库。



- 减少数据库的访问:
  - 页面静态化
  - 缓存
- celery异步任务
  - 重要知识点
  - celery工作原理（生产者消费者模式）
  - RabbitMQ或Redis可以作为中间人
  - celery的使用场景。

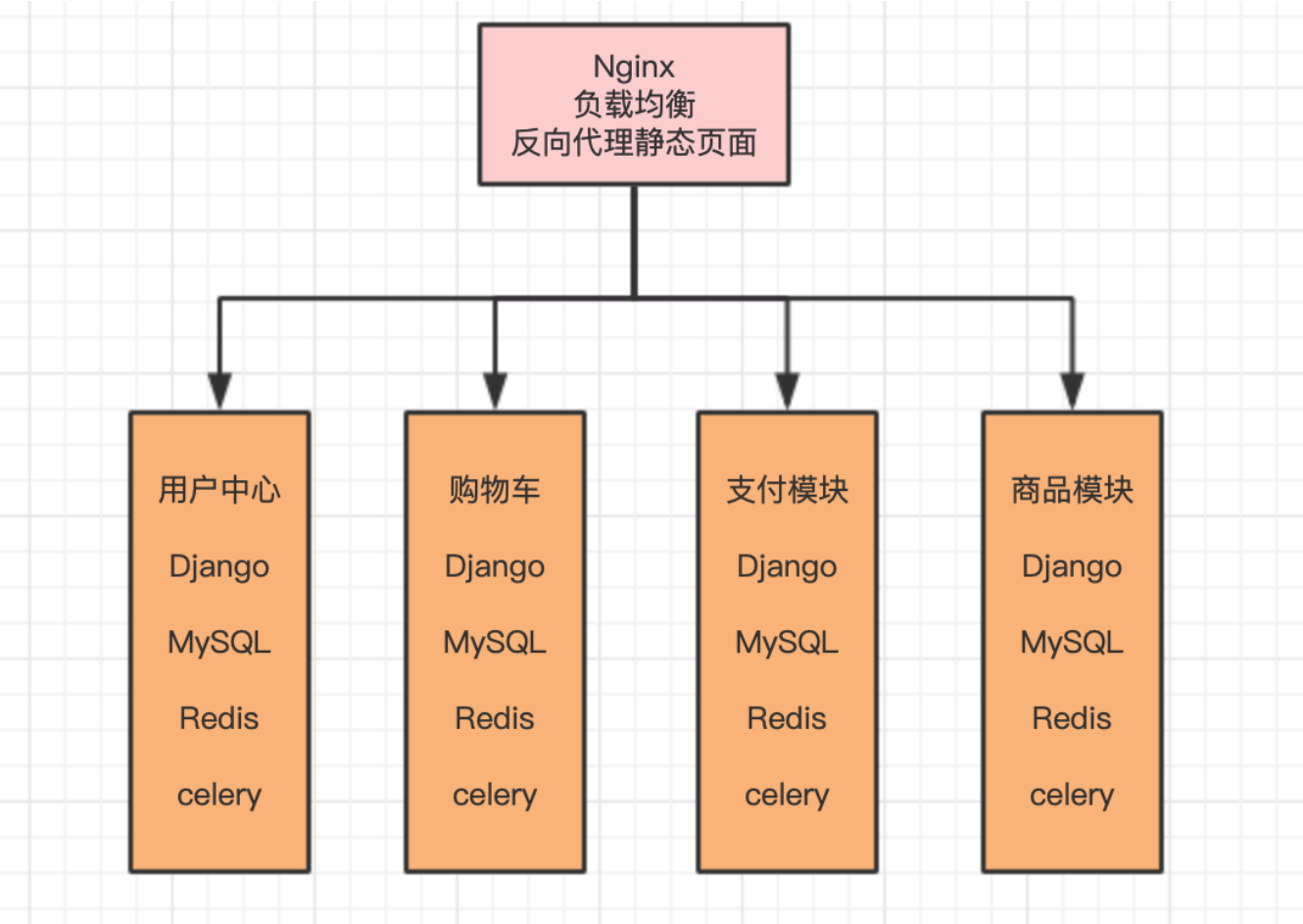
应用拆分:

为什么:

随着后端团队的壮大, 分给每个同事的需求也越来越细, 如果继续在一个工程里面开发所有的代码, 维护起来的代价太高

问题:

1. 服务之间的调用采用http的方式, 并且要求有依赖的服务主机配置hosts指向被调用的地址, 这样带来的维护上的不方便.
2. 以及在调用链的过程中没有重试, 错误处理, 限流等等的策略, 导致服务可用性差. 随着业务拆分, 继续使用Nginx维护配置非常麻烦, 经常因为修改Nginx的配置引发调用错误.
3. 每一个服务都有一个完整的认证过程, 认证又依赖于用户中心的数据库, 修改认证时需要重新发布多个服务.



mongodb使用场景和优缺点

使用场景

1 网站数据

2 缓存

3 大尺寸，低价值的数据

4 高伸缩性的场景

5 用于对象及json数据的存储

6 一般数据存储

7 把mongodb当作MySQL的一个view，view是将表数据整合成业务数据的关键

优点

1 弱一致性，更能保证用户的访问速度

2 文档结构的存储方式，能够更便捷的获取数据

3 内置GridFS，高效存储二进制大对象

4 支持复制集，主备，互为主备，自动分片等特性

5 动态查询

6 全索引支持，扩展到内部对象和内嵌数组

缺点

1 不支持事务

2 占用空间过大

3 维护工具不够成熟