send\_file。不用经过用户空间。

worker线程的数量。最好设置为和CPU核心数量一样。

epoll是异步非阻塞的，改进只能监控1024个缺陷，可以监控不限数量的描述符，

并且不用轮询,每当描述符就绪，采用系统回调函数将fd放入，效率更高。

。

配置过虚拟域名。

IO多路复用。多个描述符的IO操作都在一个线程并发交替的顺序完成。

location.

server。

log\_format配置在http。

tcp\_nopush。gzip。

nginx有很多模块，还可以配置请求限制，访问控制啥的，有些不记得了。

可以作为静态资源web服务器，代理，七层负载均衡，rewrite.

进程是一个“执行中的程序”，在CPU上运行的时候。是程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位。

线程是进程中的一个实体，是被系统独立调度和分派的基本单位。

进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。

进程是个容器，线程才是容器里面真正执行的执行序列。线程可以在一个进程内共享内存。

进程就是包换上下文切换的程序执行时间总和 = CPU加载上下文+CPU执行+CPU保存上下文

进程和线程都是一个时间段的描述，是CPU工作时间段的描述，不过是粒度大小不同。

线程是共享了进程的上下文环境的更为细小的CPU时间段。

协程是用户态的轻量级线程，基本没有内核切换的开销，上下文切换非常快。（通过将寄存器上下文和栈保存在其他地方）。

curl\_multi\_init

消息队列

swoole

浏览器缓存，gzip压缩，减少http请求（合并CSS，合并javascript,合并图片），合理布局页面（CSS在最上面，JS放最下面），减少COOKIE传输（静态资源用独立域名）。

CDN。

异步按需加载。

建立独立的图片服务器。（突破连接数，省COOKIE）。

反向代理（保护安全，保护隐私，缓存静态资源，七层负载均衡）。

Nginx作为静态资源，网页静态化，动静分离。反向代理，http服务器，较少作为负载均衡服务器。

本地缓存，分布式缓存（一致性哈希）。二八定律。（缓存是第一考虑的手段）。

缓存通过LRU算法对访问数据筛选出来的。

缓存穿透（恶意请求不存在的数据导致访问数据库，简单解决手段是在不存在的数据缓存起来）。

防盗链.

opcode缓存。

页面静态化。

主从复制，读写分离，数据分库。

分库分表分区。

分区：避免跨分区查询。分区查询最好在WHERE从句中包含分区键。具有主键或者唯一索引的表，主键或唯一索引必须是分区键的一部分。

启动MYSQL查询缓存。query\_cache\_type。

使用Memcache。分布式内存对象缓存系统。在内存里维护一个统一的巨大的hash表。

消息队列（异步，生产者消费者模式）应用：削锋，解耦，日志处理，消息通讯

负载均衡。（LVS四层，nginx七层。几种原理：

HTTP进行302重定向。

通过DNS域名解析重定向。

反向代理服务器Nginx也可以提供负载均衡（七层负载均衡，容易成为瓶颈）。

IP负载均衡（客户端返回的时候，两种方式：第一，SNAT改变数据包的源地址，设为自身的IP。相应就会回到负载均衡服务器。

第二，自己本身就是网关，所有数据都会经过自己.)ps:这种方法受制于负载具横网卡带宽。

通过修改MAC地址负载均衡。也叫DR（直接路由）方式。

最大的优点就是，数据是直接返回客户端的，不经过负载均衡服务器。PS：需要配置所有集群的虚拟IP和负载均衡服务器一致。这也是LVS的应用原理。

负载均衡算法有五种：RR,WRR,Random，Least Connections,Source Hashing(共享session的一种办法).

集群(用REDIS分布式缓存共享session。其他：session复制同步,session绑定（根据应用层的Cookie判断用户，然后发到同一台机器），COOKIE记录SESSION，用数据库)。

分布式服务。

SSD，RAID。

安全：XSS分为反射型和持久型。防护：转义，httponly。

SQL注入：转义，参数绑定预编译。

CSRF：表单TOEKN，验证码，refer.

文件上传。

Memcached一般使用TCP协议通信，其序列化协议是基于文本的自定义协议。命令是以命令关键字开头，后面是一组命令操作数，如读取数据get<key>。

rabc有四张表。

一张记录了角色和权限（角色是权限的集合）auth\_item。

角色权限关联表：auth\_item\_child

还有一张用户角色表，角色又关联着user\_id。

就是账号对应的角色。auth\_assignnment。

规则表（自定义的额外规则）auth\_rule。对应角色权限表的rule\_name字段。

返回前端必要的数据。

多表关联（多对多(belongsToMany)，一对多(hasMany)，一对一(hasOne,belongsTo(外键在自己的表上))），隐藏字段。

**项目特色：AOP思想：分离异常层。验证层抽离。四层。RestFul API**

**存七牛（独立域名图片服务器）。购物车用依赖缓存以及普通缓存。**

**查询商品用查询缓存。分类菜单缓存。**

**发送邮件消息队列。Elasticsearch全文检索。**

**猜想：库存量减少的问题。用Redis保存30分钟。**

**采用redis进行计数（评论数，鉴定数，浏览数，点赞数等，redis操作都是原子性）。**

**将用户好友封死关注等存在集合当中，求共同好友就是求交集。**

**用sorted set作排行榜**

**Bcript加密，加盐加密。**

**模拟负载均衡，主从复制，读写分离，双主热备，心跳检测。分库分表。**

分离异常层。验证层抽离。model分离。增删改查。

购物车用依赖缓存以及普通缓存。

将表中的基本信息和详细信息分开来。冷热数据分离。

商品使用查询缓存。

库存量减少的问题。用Redis保存30分钟。

分成四层，使得Controller层逻辑清晰，让人一看就明白这一步要干什么。

将复杂的操作隐藏在Service层和model层。

配置化。

IP转换为正整数。

图片放七牛。

第三方登录

详细信息和基本信息分开

先拿到appid和appsecret 携带回调地址和appid去让客户端访问，客户端获得code以后给我，我通过code和appid和appsecret获得AccessToken

然后再用accessToken拿去换取openid。唯一标识客户。最后可以用appid和accessToken和openid去调用接口获取信息。

http动词操作资源。

基于资源。

轻量级，无状态，用json描述数据。

主从复制，双主热备

三个库的userid如果用默认的主键会冲突，可以用序列解决。当然mycat有自带的解决方案。

代理模式实现数据库的主从自动切换。

观察者模式实现数据更新时间的一系列更新操作。

装饰器模式实现权限验证，模板渲染，JSON串化。

PHP使用ArrayAccess实现配置文件的加载。

系统吞吐量和系统并发数与响应时间的关系。

并发数低，响应时间快，吞吐量也小。

并发数变高，响应时间变慢，吞吐量变高。

并发数太高，响应时间更慢，吞吐量变小。

Redis和其他比较

redis在2.0版本后加入了自己的VM特性，突破物理内存的限制。（操作系统的页太大，KEY还保存在内存中）。

memcache可以修改最大可用内存，采用LRU算法

（从item list尾部开始淘汰，在列表中查找一个引用计数为0的item，并且释放掉，因为memcached会将刚访问过的放在item的头部）。

mongoDB适合大数据量的存储，依赖操作系统VM作内存管理，吃内存也比较厉害，服务不要和别的服务在一起。

redis支持主从复制。

redis事务支持比较简单，只能保持连续执行，是将命令放入队列当中一条条执行。

redis支持（快照、AOF）：依赖快照进行持久化，AOF增强了可靠性的同时，对性能有所影响。

aof是把增加、修改数据的命令通过write函数追加到文件尾部(默认是appendonly)。

Redis重启时读取该文件中的所有命令并且执行，从而把数据写入内存中。

内存快照是将内存中的数据以快照方式写入二进制文件中，默认文件名为dump.rdb。

Explain :

Id为一组数字或者NO

数字表示操作顺序，为NO就是UNION操作后产生。

ID值相同，执行顺序由上至下。

ID值越大，越先被执行。

MYSQL会调整关联顺序。

Select Type

SIMPLE 不包含子查询或者union操作

PRIMARY 查询中如果包含任何子查询，那么最外层的查询则被标记为PRIMARY

SUBQUERY SELECT列表中的子查询

DEPENDENT SUBQUERY 依赖外部结果的子查询。

UNION Union操作的第二个或是之后的查询值为union

DEPENDENT UNION 当UNION作为子查询时候，第二或者第二额后的查询的select type.

UNION RESULT UNION产生的结果集

DERIVED 出现在FROM子句中的子查询。

TABLE列

数据是由哪个表输出的，数据行所在的表的名称。

unionM,N 由ID为MN查询UNION产生的结果集。

<derived=N>/<subqueryN>ID为N的查询产生的结果。

PARTITIONS 列

对于分区表，显示查询的分区ID。

对于非分区表。显示为NULL。

# 

