系统设计基础

原作者github: https://github.com/CyC2018/Interview-Notebook

PDF制作github: https://github.com/sjsdfg/Interview-Notebook-PDF

一、性能

性能指标

1. 响应时间

指从某个请求发出到接收到响应消耗的时间。

在对响应时间进行测试时，通常采用重复请求方式，然后计算平均响应时间。

2. 吞吐量

指系统在单位时间内可以处理的请求数量，通常使用每秒的请求数来衡量。

3. 并发用户数

指系统能同时处理的并发用户请求数量。

在没有并发存在的系统中，请求被顺序执行，此时响应时间为吞吐量的倒数。例如系统支持的吞吐量为 100 req/s，那么平均响应时间应该为 0.01s。

目前的大型系统都支持多线程来处理并发请求，多线程能够提高吞吐量以及缩短响应时间，主要有两个原因：

- 多 CPU
- IO 等待时间

使用 IO 多路复用等方式，系统在等待一个 IO 操作完成的这段时间内不需要被阻塞，可以去处理其它请求。通过将这个等待时间利用起来，使得 CPU 利用率大大提高。

并发用户数不是越高越好，因为如果并发用户数太高，系统来不及处理这么多的请求，会使得过多的请求需要等待，那么响应时间就会大大提高。

性能优化

1. 集群

将多台服务器组成集群，使用负载均衡将请求转发到集群中，避免单一服务器的负载压力过大导致性能降低。

2. 缓存

缓存能够提高性能的原因如下：

- 缓存数据通常位于内存等介质中，这种介质对于读操作特别快；
- 缓存数据可以位于靠近用户的地理位置上；
- 可以将计算结果进行缓存，从而避免重复计算。

3. 异步

某些流程可以将操作转换为消息，将消息发送到消息队列之后立即返回，之后这个操作会被异步处理。

二、伸缩性

指不断向集群中添加服务器来缓解不断上升的用户并发访问压力和不断增长的数据存储需求。

伸缩性与性能

如果系统存在性能问题，那么单个用户的请求总是很慢的；

如果系统存在伸缩性问题，那么单个用户的请求可能会很快，但是在并发数很高的情况下系统会很慢。

实现伸缩性

应用服务器只要不具有状态，那么就可以很容易地通过负载均衡器向集群中添加新的服务器。

关系型数据库的伸缩性通过 Sharding 来实现，将数据按一定的规则分布到不同的节点上，从而解决单台存储服务器存储空间限制。

对于非关系型数据库，它们天生就是为海量数据而诞生，对伸缩性的支持特别好。

三、扩展性

指的是添加新功能时对现有系统的其它应用无影响，这就要求不同应用具备低耦合的特点。

实现可扩展主要有两种方式：

- 使用消息队列进行解耦，应用之间通过消息传递的方式进行通信；
- 使用分布式服务将业务和可复用的服务分离开来，业务使用分布式服务器框架调用可复用的服务。新增的产品可以用过调用可复用的服务来实现业务逻辑，对其它产品没有影响。

四、可用性

冗余

保证高可用的主要手段是使用冗余，当某个服务器故障时就请求其它服务器。

应用服务器的冗余比较容易实现，只要保证应用服务器不具有状态，那么某个应用服务器故障

时，负载均衡器将该应用服务器原先的用户请求转发到另一个应用服务器上不会对用户有任何影响。

存储服务器的冗余需要使用主从复制来实现，当主服务器故障时，需要提升从服务器为主服务器，这个过程称为切换。

监控

对 CPU、内存、磁盘、网络等系统负载信息进行监控，当某个数据达到一定阈值时通知运维人员，从而在系统发生故障之前及时发现问题。

服务降级

服务器降级是系统为了应对大量的请求，主动关闭部分功能，从而保证核心功能可用。

五、安全性

要求系统的应对各种攻击手段时能够有可靠的应对措施。
