Runoob:

一致性hash在扩缩容时只有一部分映射关系发生变化

熔断分为实例熔断和服务熔断

连接多路复用（Connection Multiplexing）和连接池（Connection Pooling）是两种优化网络通信性能和资源使用的技术，尽管它们的目的相似，但在实现方式和应用场景上存在一些关键的区别：

连接多路复用

定义：连接多路复用指的是在单个网络连接上同时传输多个独立的消息流的技术。它允许多个请求和响应在同一时间通过同一个TCP连接并行进行，而不需要为每个请求/响应对建立新的连接。HTTP/2是实现连接多路复用的一个典型例子，它可以在单个连接上同时处理多个HTTP请求和响应。

优点：减少了因建立和关闭连接所需的时间和资源消耗，减轻了服务器压力，降低了延迟，并提高了传输效率。

应用场景：特别适合于需要高效率和高性能的客户端和服务器通信，如现代web服务。

连接池

定义：连接池是预先创建并维护一组网络连接的技术，这些连接可以被多个客户端共享和重用。当一个客户端需要与服务器通信

什么是一致性hash

ChatGPT

一致性哈希（Consistent Hashing）是一种特别设计的哈希算法，最初为了解决分布式系统中的动态扩缩容问题而提出。在传统的哈希算法中，当哈希表的大小发生变化（例如，添加或删除存储节点）时，几乎所有的键值对都需要重新映射到新的节点上，这在分布式系统中会导致大量的数据迁移，进而引起高昂的成本和显著的性能下降。一致性哈希算法通过特定的方式组织哈希表，极大地减少了因节点变化导致的数据重映射数量，从而优化了系统的扩展性和稳定性。

工作原理

环形空间：一致性哈希将哈希值组织在一个环形的空间里，哈希空间通常是一个固定范围的整数集合，例如0到2^32-1。这个环被称为哈希环。

数据映射：每当添加一个数据项时，会计算它的哈希值，并将该数据项放置到哈希环上对应哈希值的位置。

节点映射：同样地，系统中的每个节点（例如缓存服务器或数据库服务器）也会被映射到哈希环上的某个点上。

数据定位：为了找到一个特定数据项应该被存储在哪个节点上，算法计算该数据项的哈希值，然后沿着哈希环顺时针寻找，直到遇到的第一个节点，那个节点就是该数据项应该存储的位置。

特点和优势

动态伸缩性：当系统中添加或移除节点时，只有一小部分数据需要重新映射到新的节点上，大多数数据可以保持不变。

负载均衡：通过适当的策略，如虚拟节点技术，一致性哈希可以保证系统中的数据分布相对均衡，避免某些节点过载而其他节点空闲的情况。

高可用性：在面对节点故障时，一致性哈希通过将故障节点上的数据快速重新映射到其他节点上，能够提高系统的整体可用性。

一致性哈希广泛应用于分布式缓存、分布式存储系统、负载均衡器和CDN等领域，成为构建可扩展、高效和稳定的分布式系统的关键技术之一。

重试雪崩（Retry Avalanche）是一个分布式系统术语，用来描述在系统部分失败时，由于客户端或服务的重试机制不当设置导致的问题。具体来说，当服务因为某种原因变得响应缓慢或者失败时，客户端通常会尝试重试请求。如果重试没有适当地控制，比如所有客户端在短时间内连续快速重试，那么这些重复的请求可能会迅速积累，形成巨大的流量，这就是“重试雪崩”。

这种情况下，原本只是暂时的或者局部的服务性能下降，可能会被放大成全面的系统崩溃，因为重试请求大量增加了服务器的负载，导致服务器资源（如CPU、内存、带宽等）迅速耗尽。此外，系统恢复正常的时间也会因为这些额外的负载而大幅延长。

重试雪崩的主要问题包括：

请求堆积：客户端连续重试导致请求在服务器端堆积，增加了服务器的负载。

延迟增加：大量的重试请求可能导致系统延迟进一步增加，影响用户体验。

服务恶化：服务器可能完全处理不过来这

合并部署：尽量将流量分配到同一台物理机上

