分布式程序设计

First 什么是分布式系统

**分布式系统是一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统。**

为什么需要分布式系统？

那就要讲到软件系统的发展历史，在以前，最流行的是集中式系统，前端资源文件、后端服务、数据库 and so on ，这些都部署在一台机器上，这台机器承担了整个系统的所有计算任务。

但是随着系统的扩大，流量的增加，集中式系统的问题就暴露了：扩容困难，故障后系统不可用等等问题

所以出现了分布式系统，例如后端服务和数据库部署在不同的机器上，后端服务通过网络访问数据库服务。数据库服务也可以部署在多台机器上，组成数据库集群。

但是分布式系统同样也有问题，例如保证服务之间正确通信，如何保证数据一致性、并发问题如何解决、如何负载均衡等等

现在有很多成熟的解决方案和工具，例如

可以使用nginx来做反向代理，实现负载均衡。

可以使用redis做缓存来提升查询性能。

可以使用主从同步来解决数据一致性

可以使用分布式锁来解决并发问题

可以使用kubernetes来管理部署服务实例，提升服务可用性

分布式系统中有一个经典理论叫CAP原则CAP Principle

指的是在一个分布式系统中，[一致性](https://baike.baidu.com/item/%E4%B8%80%E8%87%B4%E6%80%A7/9840083" \t "_blank)（Consistency）、[可用性](https://baike.baidu.com/item/%E5%8F%AF%E7%94%A8%E6%80%A7/109628" \t "_blank)（Availability）、[分区容错性](https://baike.baidu.com/item/%E5%88%86%E5%8C%BA%E5%AE%B9%E9%94%99%E6%80%A7/23734073" \t "_blank)（Partition tolerance）。CAP 原则指的是，这三个[要素](https://baike.baidu.com/item/%E8%A6%81%E7%B4%A0/5261200" \t "_blank)最多只能同时实现两点，不可能三者兼顾。

用一个情景来举例。

We have a backend service,this backend service is deployed on three machines

And this service need storage some data at local

这些数据在服务运行时会不断的被使用，被更新，数据的正确性会影响服务的正常运行。

C是保证机器这些数据一直一致，A是保证用户在任何时间使用服务，至少有一台机器提供服务，也就是机器不能全部死掉，P是保证如果有一个机器死了，其他机器需要正常运行，不受影响

那么如果 CA ，那么三台机器的数据必须保证同步，如果有一台机器死掉，那么其他两台机器就无法了解另一台机器数据是否改变，是否正确。计算可能会出错，so P is broken。

如果AP，如果一个机器死了，其他不能受影响，那么就必须牺牲C，数据必须放弃一致性，才能保证其他2个机器正常使用

如果CP，那么在有机器死掉的时候，必须把所有机器全部下线，停止服务，这样可以保证CP，计算永远不会出问题，但是牺牲了A。

我们无法同时满足CAP，但是我们可以通过技术手段尽量去减少CAP带来的影响。