分布式系统概念

起由

数据的分布式

单机数据服务

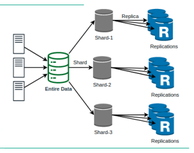
在一个项目的初期，请求量以及存储数据容量都不多，此时一台Web Server + 一台 DB Server完全能够满足业务需求

再往后，项目访问量逐步增多，一台Web Server无法处理当前数量的请求，故添加多台Web Server

多机数据服务

项目用户规模持续扩大，数据量持续增多，数据库已成为新的瓶颈，此时进行读写分离

数据量出现爆炸式增长，以达到单机无法处理的程度，此时需对数据进行分片。对数据进行分片后，为了应对节点故障(网络、磁盘等)，同时又需要对分片数据进行数据冗余，通常称之为副本



分布式系统问题总览

网络通信

节点和节点之间的可靠、高效通信是分布式系统首要解决的问题

TCP、UDP协议如何选取，选择协议后又如何构建通信协议，是使用已有的HTTP协议，还是自定义通信协议，均需要根据实际情况进行抉择

数据复制

数据复制采用何种手段进行，日志格式如何，同步复制还是异步复制，数据副本数量如何确定，如何更快速地进行日志复制，出现复制延迟时该如何操作...

数据分区

当单机无法承载完整数据时，必然需要对数据进行分区，那么分区规则该如何确定，分区后的数据索引如何建立，新增节点后是否需要对数据进行迁移，迁移成本如何...

分布式计算

如何充分得利用节点中的CPU与内存资源对原有计算任务进行加速，如何处理节点失效而导致的任务失败...

分布式系统的目的就在于通过增加机器的方式来应对快速增长的用户量、请求量以及数据量，并且能够达到每增加一倍的节点，系统吞吐量也能够达到一倍，甚至更多

名词释义

数据副本

在分布式系统中，所有的节点,所有的数据块，其地位都是平等的，副本是指数据的拷贝(Copy)，而不是主副

数据副本简单来说就是一份数据被拷贝成了多份，散布于不同的节点之上

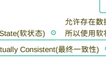
以内存和CPU缓存为例，数据首先被写入内存，而后CPU进行读取，为了加快数据的后续读取速度，CPU将数据在自身的多级缓存(L1,L2,L3)中缓存一份。那么此时数据就有两份，一份存在于内存，一份存在于CPU多级缓存中

一致性

一致性建立在副本这一概念之上，如果集群中只有一个节点，那么自然也就没有一致性(集群范围)问题一说

分布式系统中的一致性有时为了避免和数据库事务的ACID中的C冲突，Google Spanner提出了External Consistency，即外部一致性的概念

同样以内存和CPU缓存为例，由于数据目前存在2个副本，那么当更新其中一个副本数据时，如何使得另外一个数据副本也进行更新，是一致性需要探讨的问题



在数据写入后的复制过程中若出现了复制故障，导致最新的数据未能同步至其它副本节点中，此时我们会说它们的数据不具有一致性

容错

一个系统可能会出现各种各样的故障，比如机房断电，电缆被挖断，磁盘故障，那么能预料并应对这些故障的系统特性可称为容错

以无状态的Web Server为例，将其部署于多个节点之上，当其中某一个节点失效时，仍然能对外提供相应的服务

分区

分区一词在分布式系统中存在多种释义，例如数据分区，网络分区

数据分区是指将全量的数据按照某种规格进行划分，使得数据均匀地分布在所有集群节点中，并且每一条数据仅存在于一个分区之上

同络分区更多的是指一种常见的集群网络状况。在正常情况下，集群内所有的节点间应该是能够进行相互通信的，但是由于网络设备的故障，导致集群网络分裂成了多个独立的组

CAP理论

内容

Consistency(一致性)

数据在多个副本内保持严格的一致性

所谓严格一致性，即当数据在主节点成功修改后，于任何从节点进行数据读取时均能读取到这一修改

Availability(可用性)

每一次请求均能获取到正确的响应

Partition Tolerance(分区容错性)

当发生网络分区时，即因某些故障导致集群节点被割裂，仍然能对外提供满足一致性和可用性的服务

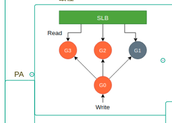
释义

Network Partition这件事情，我们认为是无法避免的，因为世界上没有挖不断的光纤，没有不会犯错的人员

正因为如此，所以目前追求的是PA或者是PC，CAP无法全部满足

PA

PA严格意义上来说，并不是舍弃了一致性，而是舍弃了强一致性，即舍去了Linearizability这一一致性



如图所示，假设G1节点与G0节点的网络因故障断开，导致数据同步意外停止。此时G1数据内容落后于G0主节点

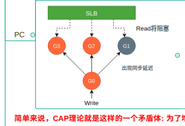
PA保证的是可用性，即不会停止整个集群的工作，而是让SLB继续请求G1节点数据，使用若一致性来保持集群服务可用性

的确，我们可以通过各种各样的监控手段以及单节点停机机制来尽可能地缩短用户读取到不一致数据的时间，但是从理论上来说，仍然不是强一致性

PC

PC严格意义上来说，也不是放弃了可用性，而是放弃了强可用性，即在任何时刻都能在约定的时间内返回成功的结果

PC中的C表示Linearizability级别的强一致性，只要有一个节点同步出现延迟，整个集群服务均需要等待该同步过程完毕，才能对外提供服务



此时的强一致性是牺牲可用性作为代价实现的，其原因在于如果要保持强一致性，就必须等待所有节点数据同步完成，而在这个过程中，集群对外提供的服务是停滞的

简单来说，CAP理论就是这样的一个矛盾体: 为了增强集群的分区容错性，将数据复制到更多的节点上，但是此时数据的复制就会变多，强一致性很难保证。如果要保持强一致性，那么更新所有节点数据的时间也就会变长，此时可用性又会降低

BASE理论

在CAP理论中，我们认为P必须得到满足(因为分区必然存在)，所以在一致性和可用性中进行二选一。但是，分区不会每时每刻都在进行，也不是永远不会恢复，在绝大部分时刻，一致性与可用性均能得到满足

内容

Basically Available(基本可用性)

基本可用性其实就是可用性的降级，部分用户的请求无法满足，但绝大部分的用户请求能够得到满足

例如正常情况下同站P99的响应时间为100ms，在发生某些故障后，P99的时间为200ms，但仍然能对外提供服务，称为基本可用性

Soft State(软状态)

允许存在数据复制的延时，在出现延时到解决延时的这一段时间内，有一个中间状态，在该状态下，数据并不是严格的一致。

Eventually Consistent(最终一致性)

所以使用软状态依次表示

从软状态中恢复成完全一致的最终结果