链滴

# 解决一致性问题的模式和思路之酸碱平衡理论

作者：AutisticV5

原文链接：https://ld246.com/article/1577352298914

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180518.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h3 id="酸碱平衡理论">酸碱平衡理论</h3>
<ol>
<li>
ACID(酸)
关系型数据库天生用于解决具有复杂事务场景的问题，完全满足 ACID 的特性。
ACID 指如下内容
<ul>
<li>
A：Atommicity，原子性
</li>
<li>
C：Consistency，一致性
</li>
<li>
I：Isolation，隔离性
</li>
<li>
D：Durability，持久性。
</li>
</ul>
具有 ACID 特性的数据库支持强一致性，强一致性代表数据库本身不会出现不一致，每个事务都原子的，或者成功或者失败，事物间是隔离的，互相完全不受影响，而且最终状态是持久罗盘的。因，数据库会从一个明确的状态过渡到另外一个明确的状态，中间的临时状态是不会出现的，如果出现会及时地自动修复，因此是强一致的。3 个典型的关系型数据库 Oracle、MySQL、DB2 都能保证强致性，通常是通过多版本控制协议（MVCC）来实现的。
如果在为交易的相关系统做技术选型，则交易的存储应该只考虑关系型数据库，对于核心系统，果需要较好的性能，则可以考虑使用更强悍的硬件，这种向上扩展（升级硬件）虽然成本较高，却是简单、有效的。另外，NoSQL 完全不适合交易场景，主要用来做数据分析、ETL、报表、数据挖掘、荐、日志处理、调用链跟踪等非核心交易场景。
</li>
<li>
CAP(帽子原理)
由于对系统或者数据进行了拆分，我们的系统不再是单机系统，而是分布式系统，针对分布式系的 CAP 原理包含如下三个元素。
<ul>
<li>
C：Consistency，一致性。在分布式系统中的所有数据备份，在同一时刻具有同样的值，所有点在同一时刻读取的数据都是最新的数据副本。
</li>
<li>
A：Availability，可用性，好的响应性能。完全的可用性指的是在任何故障模型下，服务都会在限的时间内处理完成并进行响应。
</li>
<li>
P：Partition tolerance，分区容忍性。尽管网络上有部分消息丢失，但系统仍然可继续工作。</
>
</li>
</ul>
CAP 原理证明，任何分布式系统只可同时满足以上两点，无法三者兼顾。由于关系型数据库是节点无复制的，因此不具有分区容忍性，但是具有一致性和可用性，而分布式的服务化系统都需要满分区容忍性，那么我们必须在一致性和可用性之间进行权衡。如果在网络上有消息丢失，也就是出现网络分区，则复制操作可能会被延后，如果这时我们的使用方等待复制完成再返回，则可能导致在有

时间内无法返回，就失去了可用性，而如果使用方不等待复制完成，而在主分片写完后直接返回，则有了可用性，但是失去了一致性。
</li>
<li>
BASE(碱)
BASE 思想解决了 CAP 提出的分布式系统的一致性和可用性不可兼得的问题。BASE 是“碱”的思，ACID 是“酸”的意思，基于这两个名词提出了“酸碱平衡”的理论，简单来说就是在不同的场下，可以分别利用 ACID 和 BASE 来解决分布式服务化系统的一致性问题
BASE 思想和 ACID 原理截然不同，它满足 CAP 原理，通过牺牲强一致性获得可用性，一般应于服务化系统的应用层或者大数据处理系统中，通过达到最终一致性来尽量满足业务的绝大多数需求
BASE 模型包含如下三个元素。
<ul>
<li>
BA：Basically Available，基本可用。
</li>
<li>
S：Soft State，软状态，状态可以在一段时间内不同步。
</li>
<li>
E：Eventually Consisient，最终一致，在一定的时间窗口内，最终数据达成一致即可。
</li>
</ul>
软状态是实现 BASE 思想的方法，基本可用可最终一致是目标。以 BASE 思想实现的系统由于不证强一致性，所以系统在处理请求的过程中可以存在短暂的不一致，在短暂的不一致的时间窗口内，求处理处于临时状态中，系统在进行每步操作时，通过记录每个临时状态，在系统出现故障时可以从些中间状态继续处理未完成的请求或者退回到原始状态，最终达到一致状态。
在实际应用中，上面这个过程通常是通过持久化执行任务的状态和环境信息，一旦出现问题，则时任务会捞取未执行完的任务，继续执行未执行完的任务，知道执行完成，或者取消已经完成的部分作并回到原始状态。这个方法在任务完成每个阶段时，都要更新数据库中任务的状态，这在大规模、并发系统中不会有太好的性能，一种更好的方法是用 Write-Ahead-Log（写前日志），这和数据库的 BinLog（操作日志）相似，在进行每个操作步骤时，都先写入日志，如果操作遇到问题而停止，则可读取日志并按照步骤进行恢复，继续执行未完成的工作，最后达到一致的状态。写前日志可以利用机硬盘的追加写来达到较好的性能，然后这是一种专业化的实现方式，多数业务系统还是使用数据库记的字段来记录任务的执行状态，也就是记录中间的“软状态”。一个任务的状态流转一般可以通过数库的行级锁来实现，这比使用写前日志实现更简单，快捷。
</li>
<li>
酸碱平衡总结
<ul>
<li>
向上扩展（强悍的硬件）并运行专业的关系型数据库，能够保证强一致性，能用向上扩展解决的题都不是问题。
</li>
<li>
如果向上扩展的成本很高，则可以对廉价硬件运行的开源关系型数据库进行水平伸缩和分片，将关数据分到数据库的同一个片上，仍然能够使用关系型数据库保证事务。
</li>
<li>
如果业务规则限制，无法将相关数据分到同一个分片上，就需要实现最终一致性，在记录事务的状态（中间状态、临时状态）时若出现不一致，则可以通过系统自动化或者人工干预来修复不一致的题。
</li>
</ul>

```
</li>
</ol>
```

原文链接：解决一致性问题的模式和思路之酸碱平衡理论