CAP理论

**1.CAP概述**

CAP理论是由EricBrewer教授提出的，在设计和部署分布式应用的时候，存在三个核心的系统需求，这个三个需求之间存在一定的特殊关系。三个需求如下：

C: **Consistency** 一致性

A: **Availability** 可用性

P:**Partition Tolerance**分区容错性

CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

**2.CAP定义**

(1)C: **Consistency** 一致性

一致性又称为原子性或者事务性。表示一个事务的操作是不可分割的，要不然这个事务完成，要不然这个事务不完成，不会出现这个事务完成了一半这样的情况。这种事务的原子性使得数据具有一致性。

我们通常情况下在数据库中存在的脏数据就属于数据没有具有一致性的表现。而在分布式系统中，经常出现的一个数据不具有一致性的情况是读写数据时缺乏一致性。比如两个节点数据冗余，第一个节点有一个写操作，数据更新以后没有有效的使得第二个节点更新数据，在读取第二个节点的时候就会出现不一致的问题出现。

传统的ACID数据库是很少存在一致性问题的，因为数据的单点原因，数据的存取又具有良好的事务性，不会出现读写的不一致。

(2)A: **Availability** 可用性

         好的可用性主要是指系统能够很好的为用户服务，不出现用户操作失败或者访问超时等用户体验不好的情况。可用性通常情况下可用性和分布式数据冗余，负载均衡等有着很大的关联。

(3)P:**Partition Tolerance**分区容错性

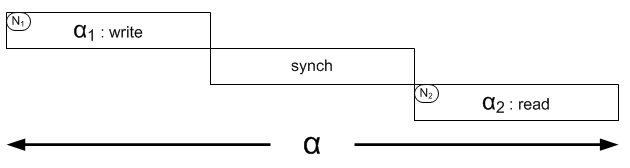
         分区容错性和扩展性紧密相关。在分布式应用中，可能因为一些分布式的原因导致系统无法正常运转。好的分区容错性要求能够使应用虽然是一个分布式系统，而看上去却好像是在一个可以运转正常的整体。比如现在的分布式系统中有某一个或者几个机器宕掉了，其他剩下的机器还能够正常运转满足系统需求，这样就具有好的分区容错性。

**3.CAP理论的意义**

         随着互联网应用的飞速发展，数据量与日俱增，传统的ACID数据库已经不能满足如此大的海量数据存储了。这个时候需要设计出好的分布式数据存储方式。而这些分布式数据存储方式受到CAP理论的约束，不可能达到高一致性，高可用性，高分区容错性的完美设计。所以我们在设计的时候要懂得取舍，重点关注对应用需求来说比较重要的，而放弃不重要的，在CAP这三者之间进行取舍，设计出贴合应用的存储方案。

         目前众多的分布式数据系统通过降低一致性来换取可用性。下面是一个简单的例子：

两个节点数据冗余，第一个节点先有一个写操作，第二个节点后有一个读操作。下面的图中a是整个过程，要具有一致性的话需要等待a1进行write，然后同步到a2，然后a2再进行write，只有整个事务完成以后，a2才能够进行read。但是这样的话使得整个系统的可用性下降，a2一直阻塞在那里等待a1同步到a2。这个时候如果对一致性要求不高的话，a2可以不等待a1数据对于a2的写同步，直接读取，这样虽然此时的读写不具有一致性，但是在后面可以通过异步的方式使得a1和a2的数据最终一致，达到最终一致性。



**4.BASE理论**

BASE理论是CAP理论结合实际的产物。 BASE(**B**asically **A**vailable, **S**oft-state,**E**ventuallyconsistent)英文中有碱的意思，这个正好和ACID的酸的意义相对，很有意思。BASE恰好和ACID是相对的，BASE要求牺牲高一致性，获得可用性或可靠性。

**5.CAP之间的取舍**

满足一致性，可用性的系统，通常在可扩展性上不太强大：

·        Traditional RDBMSs like Postgres,MySQL, etc (relational)

·        Vertica (column-oriented)

·        Aster Data (relational)

·        Greenplum (relational)

满足一致性，分区容忍必的系统，通常性能不是特别高:

·        BigTable (column-oriented/tabular)

·        Hypertable (column-oriented/tabular)

·        HBase (column-oriented/tabular)

·        MongoDB (document-oriented)

·        Terrastore (document-oriented)

·        Redis (key-value)

·        Scalaris (key-value)

·        MemcacheDB (key-value)

·        Berkeley DB (key-value)

满足可用性，分区容忍性的系统，通常可能对一致性要求低一些:

·        Dynamo (key-value)

·        Voldemort (key-value)

·        Tokyo Cabinet (key-value)

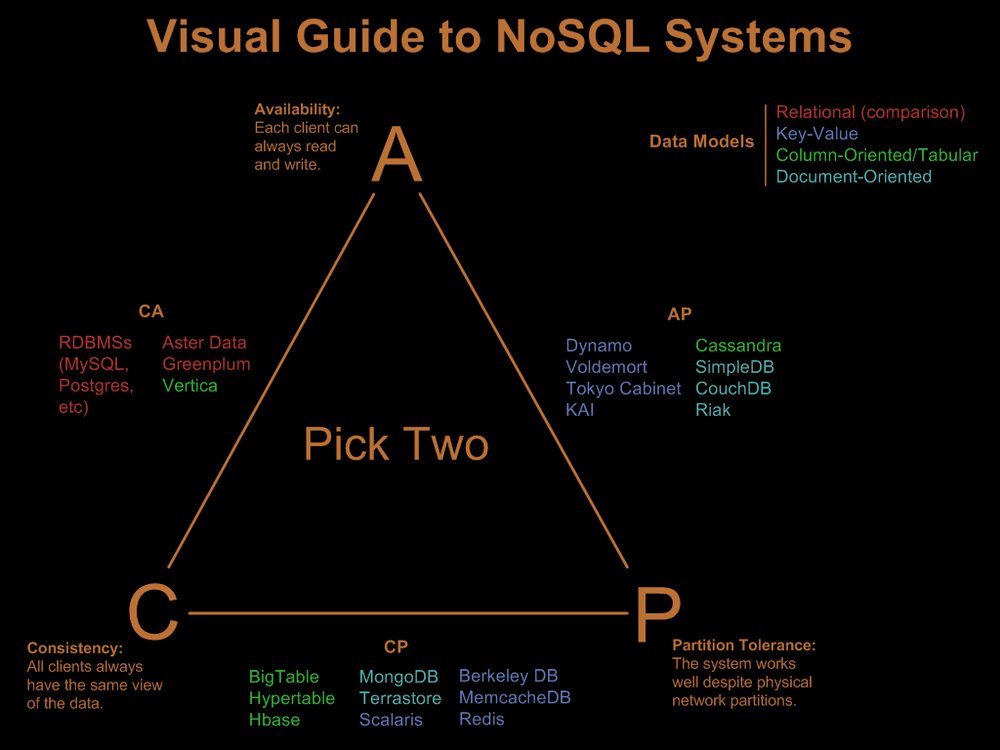
·        KAI (key-value)

·        Cassandra (column-oriented/tabular)

·        CouchDB (document-oriented)

·        SimpleDB (document-oriented)

·        Riak (document-oriented)



**6.CAP的反对声音**

Guy Pardon写了一篇文章“A CAP Solution (Proving Brewer Wrong)”来反对CAP理论。他提出了一个同时满足CAP的解决方案来反对Brewer的三者只能取其二的说法。

他设计的系统如下：

(1)程序如果能够读取数据库的话读取数据库，如果不能的话可以使用缓存代替。

(2)所有的读取操作使用版本号或者其他可以使用乐观锁的机制。

(3)客户端的所有更新操作全部放在队列中顺序处理。更新操作中要包括该更新的读取操作时的版本信息。

(4)当分区数量足够少的时候，可以处理队列中的更新操作。比较简单的方式是建立一个跨越所有分布式副本的事务，对每个副本进行更新操作（其他方式比如quorum等等也可以）。如果该更新的读取操作时的版本信息不是当前数据库中数据的版本信息，则将失败返回给客户端，否则返回成功。

(5)数据库操作结果（确认或者取消）通过异步的方式发送到客户端，可以通过邮件，消息队列或者其他异步方式。

该系统符合CAP如下：

符合C（高一致性）：读取的数据都是基于快照的，而且错误的更新操作不会执行。

符合A（高可用性）：读取和更新都会返回数据。

符合P（高分区容错性）：允许网络或者节点出错。

该设计是符合BASE理论的。

参考资料：

<http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>

<http://www.sigma.me/2011/06/13/NoSQL-CAP-Theorem.html>

<http://blog.nosqlfan.com/html/1112.html>

<http://guysblogspot.blogspot.com/2008/09/cap-solution-proving-brewer-wrong.html>