Paxos是一个分布式选举算法，主要用于保证分布式存储中副本（或者状态）的一致性。

Paxos解决问题使用的是少数服从多数的思想，只要在所有节点中，有一半以上的节点以上同意了某个决定，则认为系统达到了一致，并且按照Paxos原则，最终理论上也达到了一致，不会再改变。这样的话，客户端不必与所有服务器通信，选择与大部分通信即可；也无需服务器都全部处于工作状态，有一些服务器挂掉，只有保证半数以上存活着，整个过程也能持续下去。

在Paxos算法中，有三种角色：提议者 (Proposer)，决策者 (Acceptor)，和最终决策学习者 (Learner)，以及一个概念：提案(Proposal)

Paxos算法通过一个决议分为两个阶段：

第一阶段：Prepare阶段：

(a) Proposer选择一个提案编号N，然后向半数以上的Acceptor发送编号为N的Prepare请求。

(b) 如果一个Acceptor收到一个编号为N的Prepare请求，且N大于该Acceptor已经响应过的所有Prepare请求的编号，那么它就会将它已经接受过的编号最大的提案（如果有的话）作为响应反馈给Proposer，同时该Acceptor承诺不再接受任何编号小于N的提案。

第二阶段：Accept阶段：

1. 如果Proposer收到半数以上Acceptor对其发出的编号为 N 的Prepare请求的响应，那么它就会发送一个Accept请求给半数以上的Acceptor。
2. 如果Acceptor收到一个针对编号为 N 的提案的Accept请求，只要该Acceptor没有对编号大于N的Prepare请求做出过响应，它就接受该提案。

Paxos算法一致性的体现：

(1)每个提案者在提出提案时都会首先获取到一个具有全局唯一性的、递增的提案

号N， 即在整个集群中是唯一的编号N，然后将该编号赋予其要提出的提案。

(2)每个表决者在accept某提案后，会将该提案的编号N记录在本地，这样每个表决

者中保存的已经被accept的提案中会存在一个编号最大的提案，其编号假设为maxN。每个表决者仅会accept编号大于自己本地maxN的提案。

(3)在众多提案中最终只能有一个提案被选定。

(4)一旦一个提案被选定，则其它服务器会主动同步(Learn)该提案到本地。

(5)没有提案被提出则不会有提案被选定。