17 以太坊中的交易树和收据树

所有的交易会组织成一颗交易树，和比特币类似的。

每个交易完成之后还有一个收据，会组织成一个收据树。增加收据树的结构是为了更好的服务智能合约

状态树和交易树都是MPT - Merkle Paritia Trie，而不是比特币中的普通的Merkle Tree。这样的话以太坊中的三种树都是同一种数据类型，比较好维护，而且可以支持查找功能。

交易树和收据树是只把当前区块的交易信息组织起来，而状态树是将所有的账户信息组织成一个状态树。多个区块的状态树是共享节点的，在新的区块中，状态树只需要新建修改过的节点就可以。而交易数和收据树每个区块都是各自独立的。

交易树和收据树的主要是为了提供merkle proof。除此之外，以太坊中还支持一些比较复杂的查找功能，比如说查过去十天里和某个智能合约达成的所有交易。这个是没有办法通过扫描所有的交易信息来获得的，因为时间复杂度太高，并且轻节点没有所有交易信息的列表，没有办法查询。

为了实现这个功能，以太坊中引入了bloom filter，这个数据结构可以高效的查找某个指定的元素是否存在于某个指定的集合中。我们可以对于任何的一个集合而言，我们可以计算出来一个digest。首先初始一个向量，然后将所有的位数都设置成0，然后将集合中的每个元素，依次通过哈希函数H算出来一个index，然后将向量中对应index位置的数从0改成1，最后的向量就会成为原集合的摘要。这个时候我们再想判断一个元素是否在这个集合中，只需要对这个元素取哈希值，然后去摘要向量里面看，如果是1的话就在这个集合里，如果是0的话就不在这个集合里。但这个的问题是，一旦发生哈希碰撞的话，就可能出现false positive。也就是说这个元素本身不在集合里面，但是因为和别的在集合里的元素发生了哈希碰撞，导致最后查出来的数字也是1。但是bloom filter只会出现错报，不会出现漏报。也就说如果这个元素在集合里面，那么查出来一定是1，但这个元素不在集合里面，也有可能查出来是1。这种简单的bloom filter，是没有办法进行删除操作的，因为可能两个元素哈希碰撞映射到了同一个位置。

每个交易结束之后会有一个收据，收据里会有一个bloom filter，记录一些地址之类的关键信息。然后每一个发布的区块在块头里会有一个bloom filter，是这个区块里的所有bloom filter的并集。这个时候当你查找某个事件的时候，你可以现在块头里找，如果块头里有，再去看具体的，如果块头没有，那就可以直接过滤掉。这样bloom filter的好处就是可以快速过滤掉一些无关的节点。

状态树，交易树，收据树的根哈希值都是保存在块头里的。

以太坊的运行过程是一个交易驱动的状态机。transaction driven state machine。状态就是所有账户的状态，状态转移就是交易。比特币也可以看成一个状态机，状态是UTXO，状态转移也是交易。这些状态机的共同点是状态的转移必须是确定性的。

以太坊和比特币一样，创建账户的时候是不需要通知其他的节点，只有收到钱的时候才会需要在状态树中插入一个新的节点。

如果把状态树设计的和交易树和收据树一样，都是在区块中只保存这个区块中发生改变的，那么查找一个账户的状态就非常的不方便了。而且比如说要给一个新建的账户转账的话A -> B，需要查看B的状态，而此时需要一直回溯到创世纪块才会知道B是一个新的账户，非常的不方便，