读写集语义

本文档讨论有关读写集语义实现的详细信息。

交易模拟和读写集

> 背书节点 在模拟交易期间，会为交易准备一个读写集。读集 包含了模拟期间交易读取的键和键的版本的列表。写集 包含了交易写入键（可以与读取集中的键重叠）的新值。如果交易是删除一个键，该键就会被增加一个删除标识（在新值的位置）。

如果交易多次向同一个键写入数据，只有最后写入的数据会记录下来。同样，如果交易读取一个键的值，就会返回这个键的已提交状态的值，即使读取之前在同一个交易中更新了键值。换句话说，不支持“读你所写”的语义。

就像前面所说的，键的版本只记录在读集中；写集只包含键和交易设置的键的最新值。

版本的实现有很多种。版本设计的基本需求是，键不能有重复的版本号。例如单调递增的数字。在目前的实现中，我们使用交易所在的区块高度来作为交易中所有修改的键的版本号。这样区块中交易的高度通过一个元组来表示（txNumber 是区块中交易的高度）。这种方式比递增的数字有更多好处，主要有，它可以让其他组件比如状态数据库、交易模拟和验证有更多的设计选择。

下边是为模拟一个交易所准备的读写集示例。为了简化说明，我们使用了一个递增的数字来表示版本。

```
<TxReadWriteSet>
  <NsReadWriteSet name="chaincode1">
    <read-set>
      <read key="K1", version="1">
      <read key="K2", version="1">
    </read-set>
    <write-set>
      <write key="K1", value="V1">
      <write key="K3", value="V2">
      <write key="K4", isDelete="true">
    </write-set>
  </NsReadWriteSet>
<TxReadWriteSet>
```

另外，如果交易在模拟中执行的是一个范围查询，范围查询和它的结果都会被记录在读写集的 查询信息（query-info） 中。

交易验证和使用读写集更新世界状态

提交节点 使用读写集中的读集来验证交易，使用写集来更新受影响的键的版本和值。

在验证阶段，如果读集中键的版本和世界状态中键的版本一致就认为该交易是 有效的 ，这里我们假设所有之前 有效 的交易（同一个区块中该交易之前的交易）都会被提交（*提交状态*）。当读写集中包含一个或多个查询信息（query-info）时，需要执行额外的验证。

这种额外的验证需要确保在根据查询信息获得的结果的超集（多个范围的合并）中没有插入、删除或者更新键。换句话说，如果我们在模拟执行交易期间重新执行任何一个范围，我们应该得到相同的结果。这个检查保证了如果交易在提交期间出了虚项，该交易就会被标记为无效的。这种检查只存在于范围查询中（例如链码中的 `GetStateByRange` 方法）其他查询中没有实现（例如链码中的 `GetQueryResult` 方法）。其他查询仍会存在出现虚项的风险，我们应该只在不向排序服务提交的只读交易中使用查询，除非应用程序能保证模拟的结果和验证/提交时的结果一致。

如果交易通过了有效性验证，提交节点就会根据写集更新世界状态。在更新阶段，会根据写集更新世界状态中对应的键的值。然后，世界状态中键的版本会更新到最新的版本。

模拟和验证示例

本章节通过示例场景帮助你理解读写集语义。在本例中，`k` 表示键，在世界状态中表示一个元组 `(k,ver,val)`，`ver` 是键 `k` 的版本，`val` 是值。

现在假设有五个交易 `T1, T2, T3, T4 和 T5`，所有的交易模拟都基于同一个世界状态的快照。下边的步骤展示了世界状态和模拟这些交易时的读写活动。

```
World state: (k1,1,v1), (k2,1,v2), (k3,1,v3), (k4,1,v4), (k5,1,v5)
T1 -> Write(k1, v1'), Write(k2, v2')
T2 -> Read(k1), Write(k3, v3')
T3 -> Write(k2, v2'')
T4 -> Write(k2, v2'''), read(k2)
T5 -> Write(k6, v6'), read(k5)
```

现在，假设这些交易的顺序是从 T1 到 T5（他们可以在同一个区块，也可以在不同区块）

1. `T1` 通过了验证，因为它没有执行任何读操作。然后世界状态中的键 `k1` 和 `k2` 被更新为 `(k1,2,v1'), (k2,2,v2')`
2. `T2` 没有通过验证，因为它读了键 `k1`，但是交易 `T1` 改变了 `k1`
3. `T3` 通过了验证，因为它没有执行任何读操作。然后世界状态中的键 `k2` 被更新为 `(k2,3,v2'')`
4. `T4` 没有通过验证，因为它读了键 `k2`，但是交易 `T1` 改变了 `k2`
5. `T5` 通过了验证，因为它读了键 `k5`，但是 `k5` 没有被其他任何交易改变

Note: 不支持有多个读写集的交易。