Distributed Systems HW5

21307289 刘森元

Prob.1

> Leader selection在分布式系统中具有重要的用途，主要用于容错，即当主节点失效后能够从备份节点中选择新的leader，但是新选择的leader需要得到其他节点的认同。主流的leader选择算法有：Bully、Ring算法，但不限于这些算法，调研以下软件，简述这些软件所采用的选举算法：Zoo keeper、Redis、MongoDB、Cassandra。

1. ZooKeeper：基于ZAB协议的选举算法。ZAB协议基于Paxos算法，通过选举一个称为Leader的节点来处理所有的写操作。当Leader节点失效时，ZooKeeper会进行新的Leader选举。
2. Redis：基于Raft协议的选举算法。Raft是一种分布式一致性算法，它通过选举一个称为Leader的节点来处理所有的写操作。当Leader节点失效时，Redis会进行新的Leader选举。
3. MongoDB：基于Raft协议的选举算法。类似于Redis，MongoDB通过选举一个称为Primary的节点来处理所有的写操作。当Primary节点失效时，MongoDB会进行新的Primary选举。
4. Cassandra：基于Gossip协议的选举算法。Gossip协议是一种去中心化的选举算法，它通过节点之间的相互通信来达成一致。当节点检测到Leader节点失效时，Cassandra会进行新的Leader选举。

Prob.2

> 什么是分布式系统一致性？主要有哪几种一致性模型？分别用在哪些场景中？

分布式系统一致性是指在分布式环境下，多个节点之间的数据副本保持一致的特性。

主要的一致性模型包括：

1. 强一致性（Strong Consistency）：在强一致性模型下，所有节点在任何时间点都能够看到相同的数据副本，即数据的读写操作是按照顺序一致的。
2. 弱一致性（Weak Consistency）：在弱一致性模型下，不同节点之间的数据副本可能存在一定的延迟或不一致，但最终会达到一致状态。
3. 最终一致性（Eventual Consistency）：最终一致性模型是弱一致性模型的一种特例，它保证在没有新的更新操作发生时，所有节点最终会达到一致的状态。

Prob.3

> 你会选择哪一种一致性来实现股票市场，请解释原因

选择强一致性模型。

在股票市场中，数据的一致性对于交易的准确性和公平性至关重要。强一致性模型可以确保所有交易节点在任何时间点都能够看到相同的数据副本，从而避免了不一致的情况。这对于确保交易的一致性和可靠性非常重要，尤其是在高频交易和大宗交易等场景下。

然而，强一致性模型可能会对系统的可用性和性能产生一定的影响。可能需要采用一些技术手段来提高系统的容错性和并发处理能力，以确保强一致性的同时满足系统的可用性和性能要求。

Prob.4

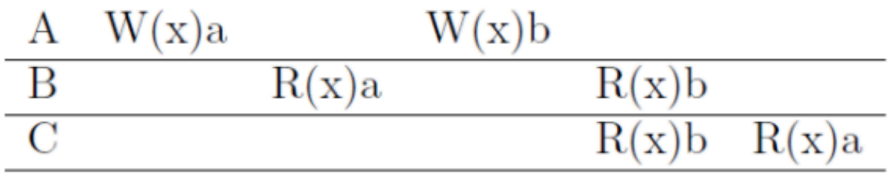
> 请描述一个用于显示刚被更新的Web页面的写读一致性的简单实现。

使用版本号或时间戳来跟踪页面的更新。

1. 在Web页面的后端，创建一个全局的版本号或时间戳变量，用于表示页面的当前版本。
2. 当页面被更新时，增加版本号或更新时间戳。
3. 在前端，使用Ajax或其他技术定期向后端发送请求，以检查页面的当前版本。
4. 在前端，将当前页面的版本号或时间戳与后端返回的版本号或时间戳进行比较。
5. 如果前端检测到后端的版本号或时间戳与当前页面的版本号或时间戳不一致，说明页面已经被更新。
6. 在前端，根据需要重新加载或更新页面内容，以确保显示最新的版本。

Prob.5

> 根据以数据为中心的一致性模型，请回答以下几个问题：
>
> a. 什么是严格一致性？为什么在分布式系统中很难实现？
>
> b. 举例说明顺序一致性。
>
> c. 以下数据存储是否是顺序一致性？给出解释并修改。
>
> d. 根据下图给出一个满足因果一致性的例子。



a.

严格一致性是指在分布式系统中，所有的操作都按照其在系统中发生的顺序对所有节点可见。对于任何两个操作，要么按照其在系统中的顺序执行，要么按照相反的顺序执行。

因为分布式系统的特点包括多个节点、网络延迟和可能的节点故障。这些因素导致了消息传递的不确定性和并发操作的可能性，使得在所有节点上实现严格一致性变得困难。

b．顺序一致性是指在分布式系统中，所有的操作都按照其在每个节点上的执行顺序进行排序，但不要求在所有节点上的执行顺序一致。

例如，如果一个节点执行了操作A，然后执行了操作B，那么顺序一致性要求其他节点看到的操作顺序也是A在B之前。但是，不同节点之间的操作顺序可以不同。

c．不是顺序一致性。

原因是节点C在执行R(x)b之前执行了R(x)a，但是在节点B上执行的顺序是R(x)a在R(x)b之后。

为了满足顺序一致性，我们需要调整节点C的操作顺序，使得R(x)b在R(x)a之前执行。

修改后的数据存储如下：

```
| A | W(x)a |       | W(x)b |       |       |
| B |       | R(x)a |       | R(x)b |       |
| C |       | R(x)b |       | R(x)a |       |
```

d．满足因果一致性的例子如下：

```
| A | W(x)a |       | W(x)b |       |       |
| B |       | R(x)a |       | R(x)b |       |
| C |       |       |       | R(x)a | R(x)b |
```

1. 当事件A发生时，事件W(x)a也发生。
2. 当事件W(x)a发生时，事件R(x)a也发生。
3. 当事件R(x)a发生时，事件R(x)b也发生。

这个例子中，每个事件都有一个明确的因果关系，符合因果一致性的要求。