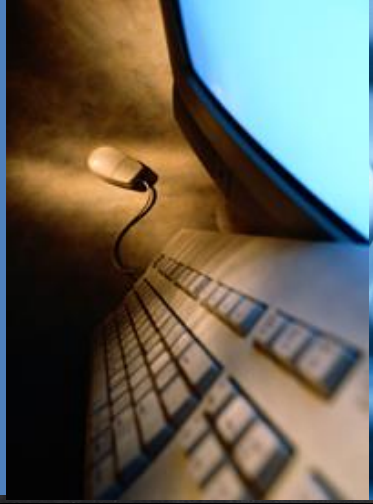



分布式系统
Distributed Systems

陈鹏飞
数据科学与计算机学院
chchenpf7@mail.sysu.edu.cn
办公室：超算5楼529d
主页：http://sdcs.sysu.edu.cn/node/3747



第七讲 — 一致性和复制

引言

CAP原则又称CAP定理，指的是在一个分布式系统中，Consistency（一致性）、Availability（可用性）、Partition tolerance（分区容错性），三者不可得兼

大纲

1. 背景知识
2. 以数据为中心的一致性模型
3. 以客户为中心的一致性模型
4. 复制管理
5. 一致性协议



1 背景知识

进行复制的原因

性能　　可用性

- **主要问题**

为了保证复制的一致性，我们通常需要确保所有冲突的操作无论在任何地方都要按照相同的顺序执行。

- **操作冲突：**

读写冲突（Read-write conflict），读操作和写操作并发执行；

写写冲突（Write-write conflict），两个并发的写操作；

- **问题**　　最终一致性

当发生冲突操作时，保证全局一致性是非常困难的，代价很高，一个折中的解决方案是：减弱一致性需求，避免全局一致性；



2

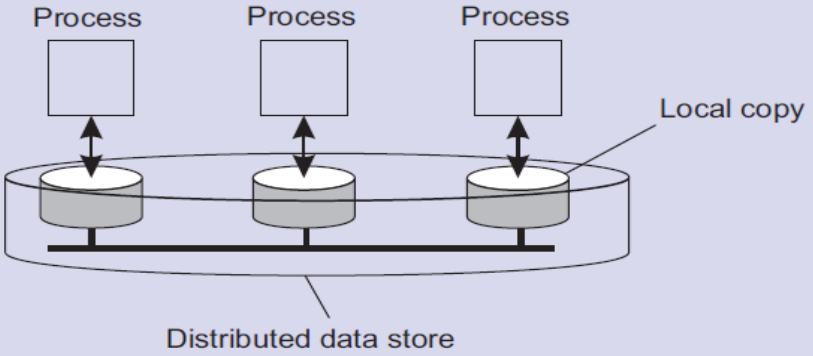
以数据为中心的一致性模型

以数据为中心的一致性模型

➢ 一致性模型

- 实质上是进程和数据存储之间的一个约定。即如果进程统一遵守某些规则，那么数据存储将正常运行。数据存储精确规约了当出现并发操作时读写操作的返回结果。

➢ 关键点



连续一致性

- **也可以定义为“一致性程度”**
  - 数值偏差：不同副本之间的数值可能不同；
  - 新旧偏差：不同副本之间的“新旧”不同；
  - 顺序偏差：不同副本更新操作的**顺序和数量**可能不同（即不同更新方法）；
- **Conit**
  - 一致性单元=> 一致性可度量的单元；

Conit

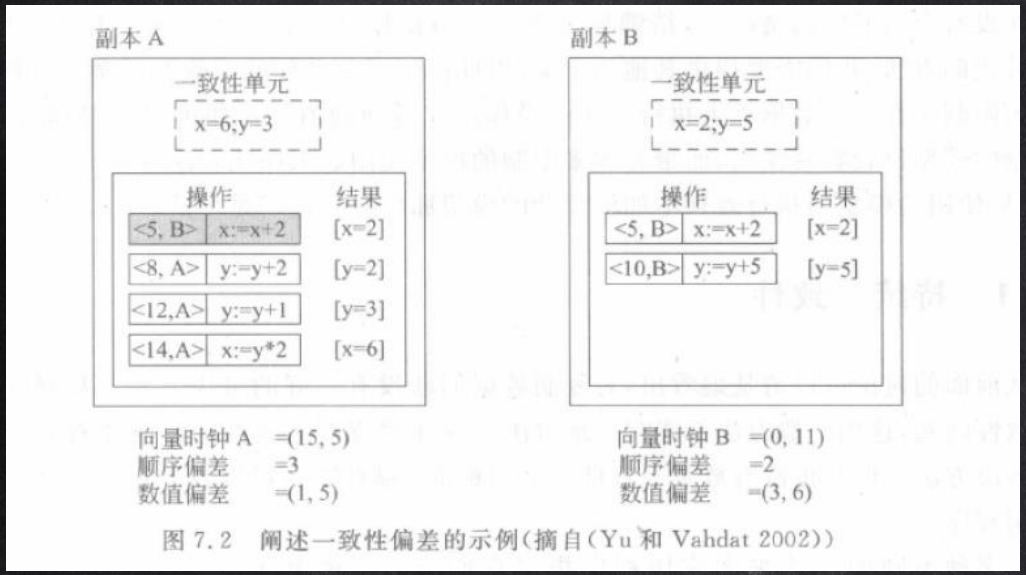


图 7.2 阐述一致性偏差的示例(摘自(Yu 和 Vahdat 2002))

- 该Conit包含 x, y;
- 每一个副本包含一个向量时钟；
- A 接收 B的操作那么A会将该操作持久化（不允许回滚）；

Conit

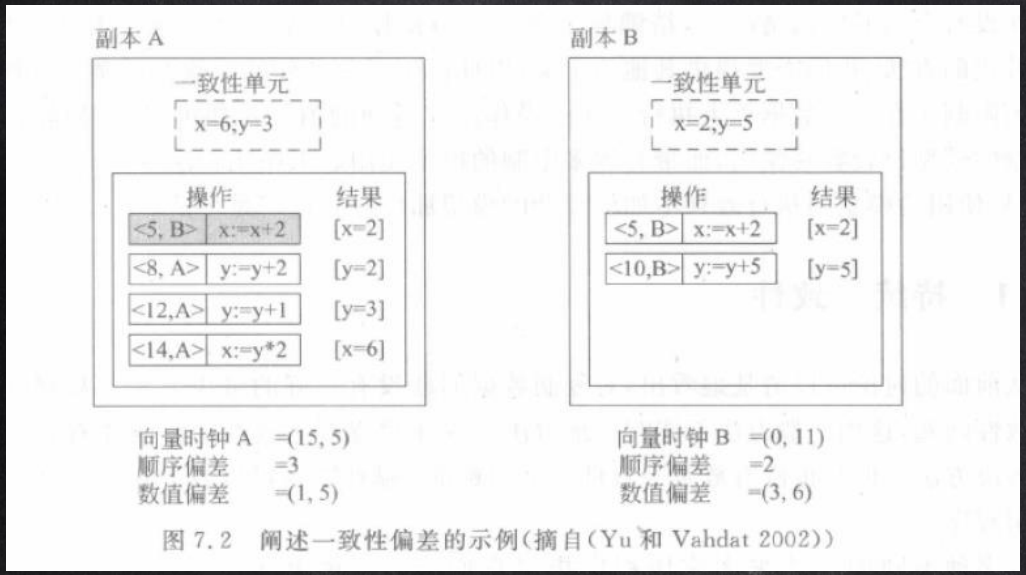


图 7.2 阐述一致性偏差的示例(摘自(Yu 和 Vahdat 2002))

- 数值偏差的构成（未接收到的更新次数，偏差权重）
- 偏差权重= A已提交的值（x,y），与还没有收到来自B的操作产生的结果之间的最大差分；

Conit的粒度

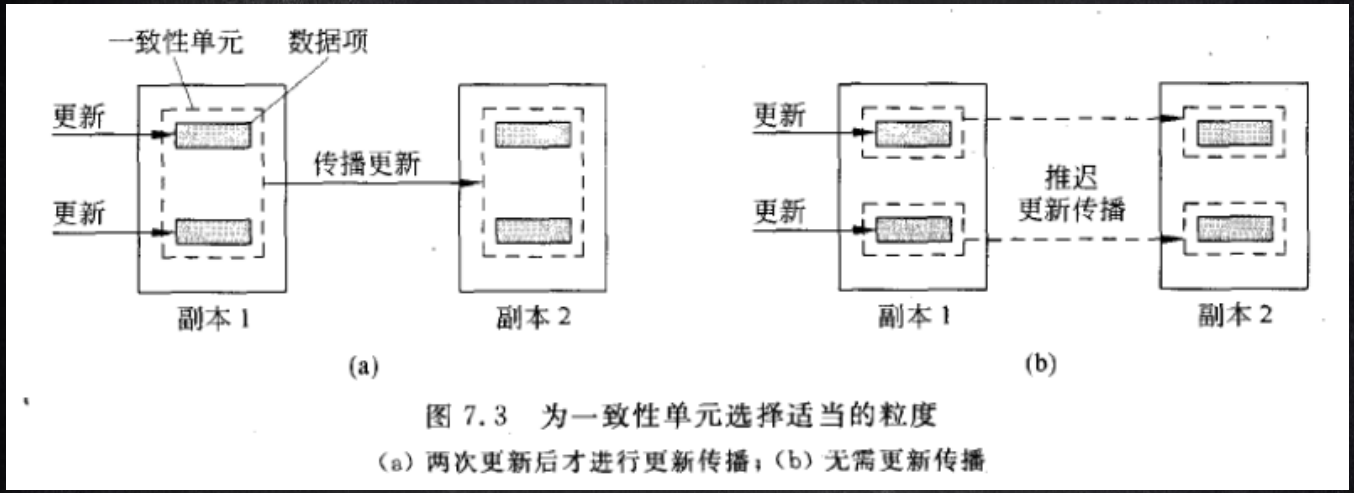


图 7.3 为一致性单元选择适当的粒度

(a) 两次更新后才进行更新传播；(b) 无需更新传播

需要在“粗粒度”和“细粒度”之间权衡

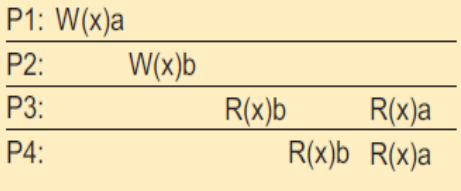
顺序一致性

- 定义

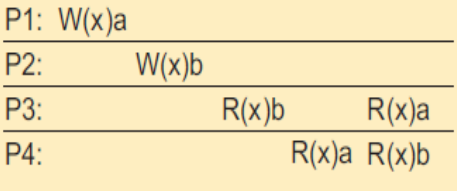
重要的以数据为中心的一致性模型。 任何执行结果都是相同的，就好像所有进程对数据存储的读写操作是按照某种序列顺序执行的，并且每个进程的操作按照程序所定制的顺序出现在这个序列中。

进程可以看到所有进程的写操作，但是只能看到自己的读操作

- 顺序一致的数据存储（a）和非顺序一致的数据存储



(a)



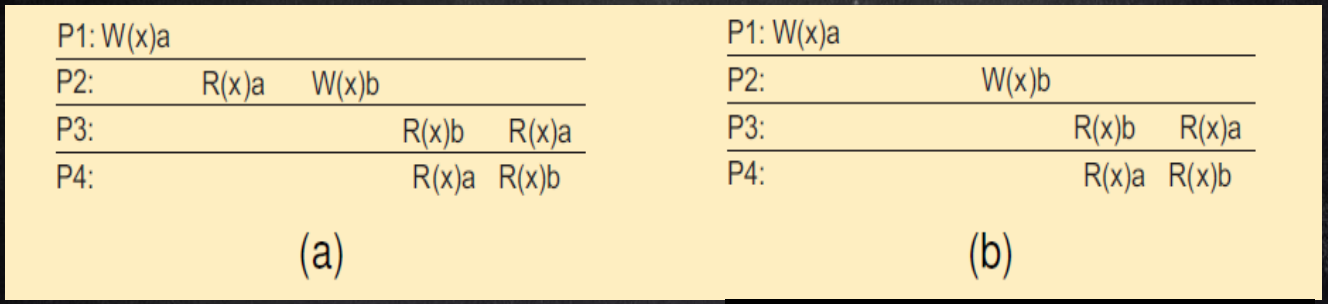
(b)

因果一致性（Causal Consistency）

➢ 定义

一种弱化的顺序一致性模型。 所有的进程必须以相同的顺序看到具有潜在因果关系的写操作。不同机器上可以以不同的顺序看到并发写操作。

➢ 不符合因果一致性 （a） 符合因果一致性



顺序一致性却不满足？

分组操作

- 定义

- 在一个进程对被保护的共享数据的所有更新操作执行完之前，不允许另一个进程执行对同步化变量的获取访问；
- 如果一个进程对某个同步化变量正进行互斥模式访问，那么其他进程就不能拥有该同步化变量，即使是非互斥模式也不行；
- 某个进程对某个同步化变量的互斥模式访问完成后，除非该变量的拥有者执行完操作，否则，任何其他进程对该变量的下一个非互斥模式访问也是不允许的；

分组操作

➢ 解释

- **条件1：**在一个进程对被保护的共享数据的所有更新操作执行完之前，不允许另一个进程执行对同步化变量的获取访问；

- **解释：**当一个进程获得拥有权后，这种拥有权直到所有被保护的数据都已更新为止、换句话说：对被保护数据的所有远程修改都是可见的。

分组操作

➢ 解释

- **条件2：**如果一个进程对某个同步化变量正进行互斥模式访问，那么其他进程就不能拥有该同步化变量，即使是非互斥模式也不行；

- **解释：**在更新一个共享数据项之前，进程必须以互斥模式进入临界区，以确保不会有其他进程试图同时更新该共享数据。

分组操作

➢ 解释

- **条件3**：某个进程对某个同步化变量的互斥模式访问完成后，除非该变量的拥有者执行完操作，否则，任何其他进程对该变量的下一个非互斥模式访问也是不允许的；
- **解释**：如果一个进程要以非互斥模式进入临界区，必须首先与该同步化变量的拥有者进行协商，确保临界区获得了被保护共享数据的最新副本。

分组操作

- 一个入口一致性的实例

P1:	L(x) W(x)a L(y) W(y)b U(x) U(y)
P2:	L(x) R(x)a R(y) NIL
P3:	L(y) R(y)b

- 观察

入口一致性暗示我们需要（显式或者隐式）加锁或者解锁数据；

- 问题

如何让这种一致性对于开发人员更加透明？

中间件



3 以客户为中心的一致性模型

背景

- 以数据为中心的一致性模型

  读写并重的数据存储；

- 以客户为中心的一致性模型

  读多写少，提供弱一致性即最终一致性；（举例）

- 最终一致性

  如果在很长一段时间内没有更新操作，那么所有的副本将逐渐地成为一致的。

移动用户的一致性

➢ 样例

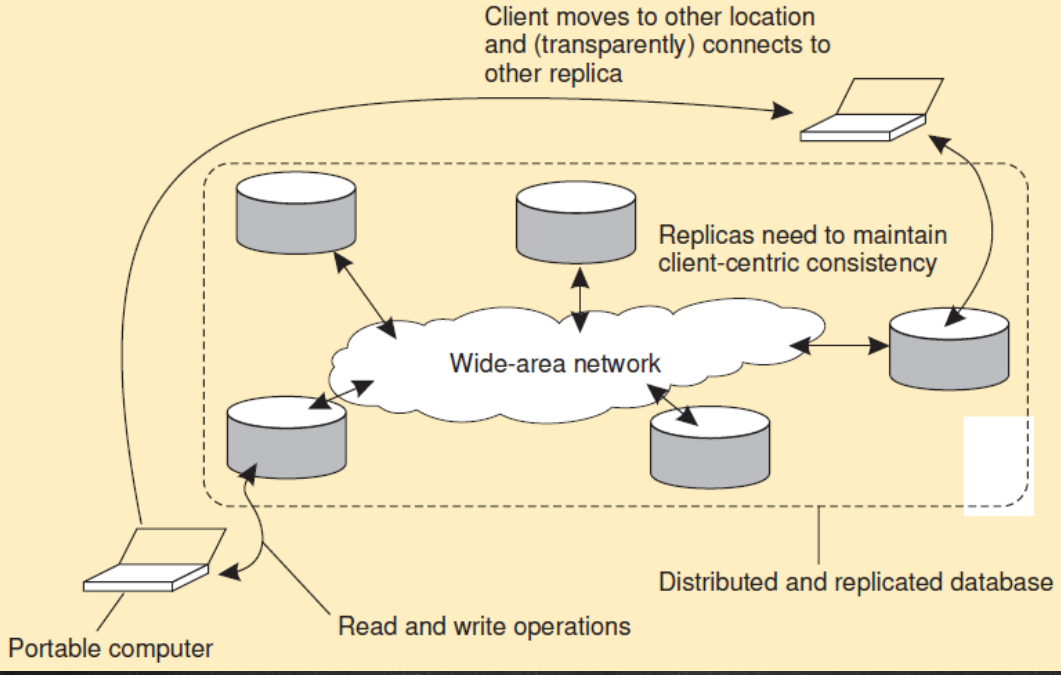
考虑一个可以通过笔记本访问的分布式数据库，假定你的笔记本可以作为前端访问数据库；

- 在地点 A，你访问数据库并进行读写操作；
- 在地点 B 你继续原来的工作，但是除非你访问位于A上的服务器，否则会检测到不一致：
  - 你在A处的更新还没有传播到B；
  - 你可能访问到比A处更新的数据；
  - 你在B处的更新可能最终于A处的更新发生冲突；

基本架构

以客户为中心的一致性

The principle of a mobile user accessing different replicas of a distributed database



单调读

➢ 定义

如果一个进程读取数据项 x 的值，那么该进程对 x 执行的任何后续操作将总是得到第一次读取的那个值或更新的值；

➢ 单调一致性

单调一致性的数据存储（a）和不提供单调一致性的数据存储(b)

L1:	$W_1(x_1)$	$R_1(x_1)$
L2:	$W_2(x_1;x_2)$	$R_1(x_2)$

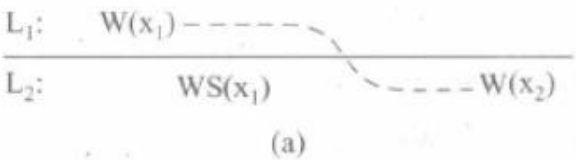
L1:	$W_1(x_1)$	$R_1(x_1)$
L2:	$W_2(x_1\|x_2)$	$R_1(x_2)$

- $W_1(x_2)$ is the write operation by process P_1 that leads to version x_2 of x
- $W_1(x_i;x_j)$ indicates P_1 produces version x_j based on a previous version x_i.
- $W_1(x_i|x_j)$ indicates P_1 produces version x_j concurrently to version x_i.

单调写

- 定义

一个进程对数据项 x 执行的写操作必须在该进程对 x 执行任何后续写操作之前完成；



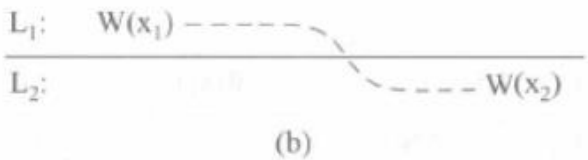


图 7.13 单一进程 P 对同一数据存储的两个不同本地副本所执行的写操作

(a) 提供单调写一致性的数据存储；(b) 不提供单调写一致性的数据存储