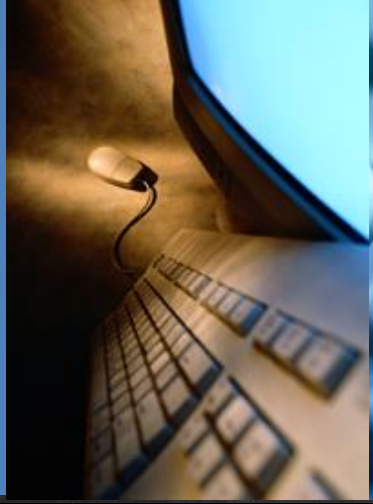



分布式系统
Distributed Systems

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

单调写

- **举例-1**

  - 在服务器 S2上更新程序，并且保证程序编译、链接所依赖的组件已经在S2 存在；

- **举例-2**

  - 代码版本管理程序， 在任何地方按照正确的顺序维护副本文件的版本；

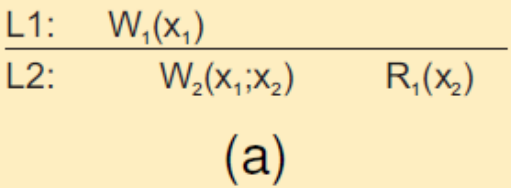
读写一致性（Read your writes）

- 定义

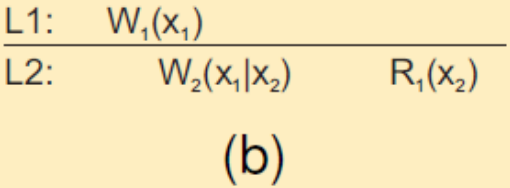
一个进程对数据项 **x** 执行一次写操作的结果总是会被该进程对 **x** 执行的后续读操作看见。

也就是说，一个写操作总是在同一进程执行的后续读操作之前完成，而不管这个后续读操作发生在什么位置

- 保证读写一致性的数据存储（a）和非读写一致性的数据存储



(a)



(b)

读写一致性例子

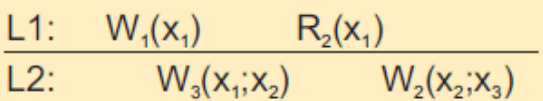
- 更新Web页面，并且保证Web浏览器能够展示最新的版本的数据，而不是缓存的内容；

写读一致性（Writes follow reads）

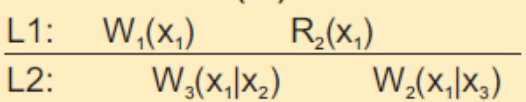
- 定义

同一个进程对数据项 **x** 执行的读操作之后的写操作，保证发生在与 **x** 读取值相同或比之更新的值上。即更新是作为前一个读操作的结果传播的。 进程对数据项 **x** 所执行的任何后续的写操作都会在 **x** 的副本上执行，而该副本是用该进程最近读取的值更新的。

- 写读一致性的数据存储（**a**）和非写读一致性的存储（**b**）



(a)



(b)

副本管理

- **背景**

对于任何支持副本的分布式系统来说，关键的问题是决定何处、何时、由谁来负责副本以及以何种机制来保持副本的一致性。 副本放置问题：副本服务器的放置问题和内容放置问题。

副本服务器放置

➢ 本质

从 N 个可能的位置中找出 K 个最佳的位置；

- 假定已放置了 k 个服务器，需要从 N-k个服务器中选择一个最佳的服务器，与所有的客户端之间的距离最小。 计算复杂度过高；
- 选择第 K 大的自治系统 ，然后在含有最大数量的连接的路由器上放置一台服务器，一次类推。计算复杂度高；
- 假定在一个 d 维的几何空间中放置服务器， 节点之间的距离反映了延迟。 把整个空间划分为多个单元，选择 K 个密度最大的单元放置副本服务器。 计算复杂度比较低；

内容放置

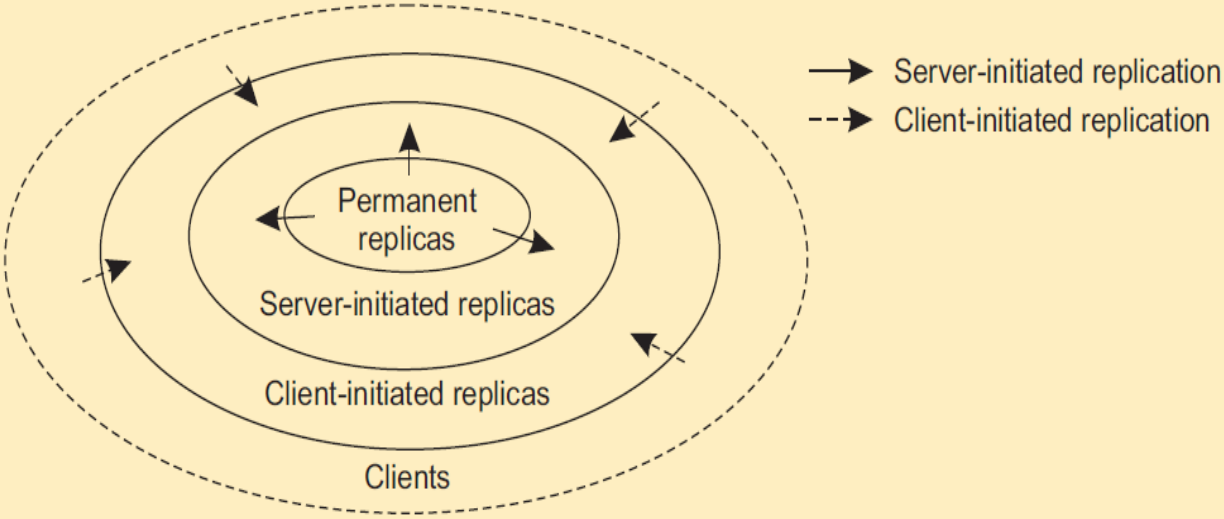
- **区分三种不同的进程**

一个进程能够维护对象或者数据的副本：

- 永久副本：进程/机器持久存储副本数据；
- 服务器启动的副本：进程可以动态的持有副本数据，该副本是在初始化数据存储的所有者时创建的；
- 客户端启动的副本：当客户端初始化时创建的副本；

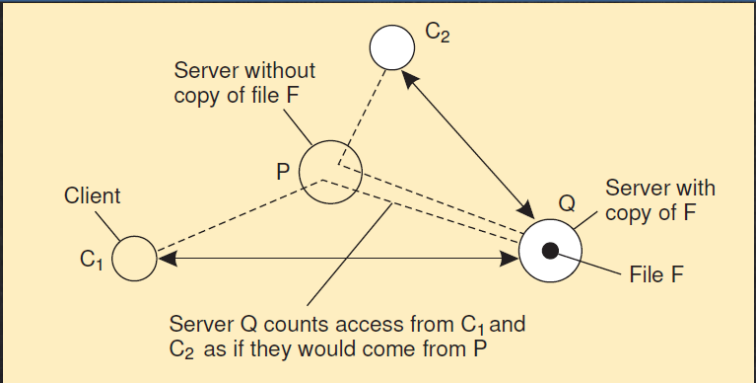
内容放置

- 不同类型的副本逻辑地组织成三个同心环



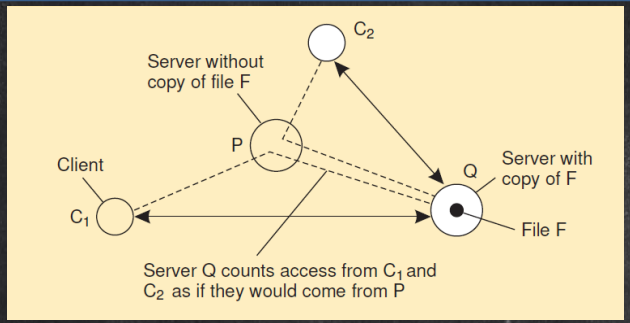
服务器初始化的副本

- **动态复制主要考虑的问题**
  - 复制可能是为了减轻一台服务器的负载而进行的；
  - 一台服务器上制定的文件可能被转移或复制到对这些文件提出很多请求的客户附近的服务器。
- **对来自不同客户端的请求计数**



服务器初始化的副本

- 对来自不同客户端的请求计数



一个进程能够维护对象或者数据的副本：

- 记录每个文件的访问次数，并且将其合并为来自最靠近客户端服务器的请求；
- 如果请求数量低于阈值D => 删除文件；
- 如果请求数量超过阈值R=>复制文件；
- 如果请求数量在 D 和 R之间 => 移动文件；

内容分发

➢ 状态与操作

一个重要的设计问题是要实际传播哪些信息：

- 只传播更新的通知（常用于缓存）；
- 将数据从一个副本传送到另一个副本（被动复制）；
- 把更新操作传播到其他副本（主动复制）；

➢ 注意

没有哪一个方法是最佳的选择，高度依赖于可用的网络带宽和副本上的读写比率；

内容分发：客户/服务器系统

➢ **在多客户端、单服务器系统中，基于Push和基于Pull的协议的比较**

- 基于Push的更新：服务器初始化的方法，不需要其他副本请求更新，这些更新就被传播到那些副本那里；
- 基于Pull的更新：客户端初始化的方法，客户端请求的更新；

Issue	Push-based	Pull-based
1:	List of client caches	None
2:	Update (and possibly fetch update)	Poll and update
3:	Immediate (or fetch-update time)	Fetch-update time

1: State at server
2: Messages to be exchanged
3: Response time at the client

内容分发

➢ 观察

利用租用（lease）的方式在Pull和Push之间动态切换。租用是服务器所作的承诺，在指定的时间内把更新推给客户。租用到期改为pull方式。

➢ 租用失效的时间依赖于系统的行为（自适应租用）

- Age-based leases: An object that hasn't changed for a long time, will not change in the near future, so provide a long-lasting lease
- Renewal-frequency based leases: The more often a client requests a specific object, the longer the expiration time for that client (for that object) will be
- State-based leases: The more loaded a server is, the shorter the expiration times become

一致性协议

- 定义

  一致性协议描述了特定的一致性模型的实现。

- 持续一致性： 限定数值偏差

- Every server S_i has a log, denoted as L_i.
- Consider a data item x and let $val(W)$ denote the numerical change in its value after a write operation W. Assume that

$$\forall W : val(W) > 0$$

- W is initially forwarded to one of the N replicas, denoted as $origin(W)$. $TW[i,j]$ are the writes executed by server S_i that originated from S_j:

$$TW[i,j] = \sum\{val(W) | origin(W) = S_j \ \& \ W \in L_i\}$$

持续一致性： 限定数值偏差

Note

Actual value $v(t)$ of x:

$$v(t) = v_{init} + \sum_{k=1}^{N} TW[k,k]$$

value v_i of x at server S_i:

$$v_i = v_{init} + \sum_{k=1}^{N} TW[i,k]$$

持续一致性： 限定数值偏差

Problem

We need to ensure that $v(t) - v_i < \delta_i$ for every server S_i.

Approach

Let every server S_k maintain a view $TW_k[i,j]$ of what it believes is the value of $TW[i,j]$. This information can be gossiped when an update is propagated.

Note

$$0 \leq TW_k[i,j] \leq TW[i,j] \leq TW[j,j]$$

持续一致性： 限定数值偏差

- 核心思想

整个思想是，当服务器 S_k 知道 S_i 与提交给 S_k 的更新操作步调不一致时，它就把写操作从其日志中转发给 S_i。该转发操作可以有效地把 S_k 的视图 $TW_k[i, k]$ 往 $TW[i, k]$ 靠近，使得偏差 $(TW[i, k]-TW_k[i, k])$ 更小。尤其是当应用程序提交一个新的写操作时，S_k 会把其视图往 $TW[i, k]$ 推进，这将使得 $(TW[i, k]-TW_k[i, k])$ 大于 $\delta_i/(N-1)$。本章后面有一个练习，证明这种推进可以确保 $v(t)-v_i \leqslant \delta_i$。

持续一致性：限定复制的新旧程度偏差

➢ 核心思想

- 让服务器 S_k 保持实时向量时钟 RVC_k，其中RVC_k[i] = T(i) 为到时间 T(i)时， S_k看到了已提交给S_i的所有写操作；

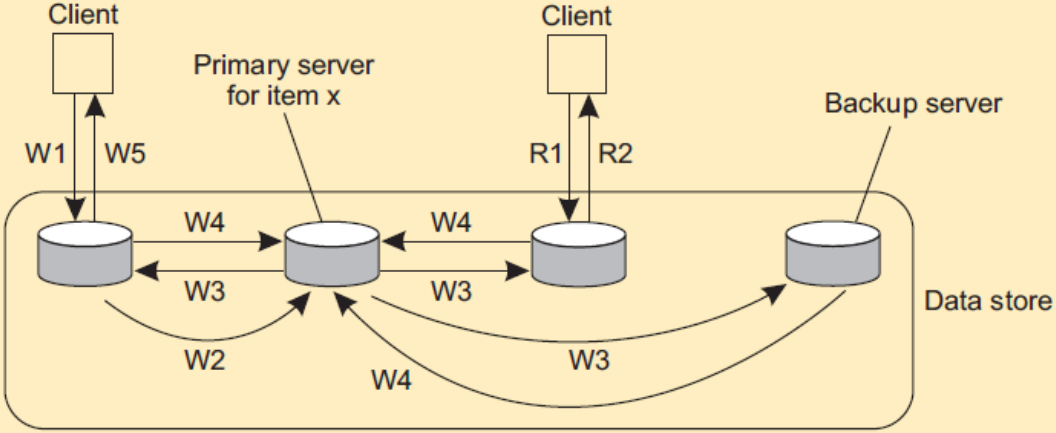
- 只要服务器 S_k 通知 T[k]-RVC_k[i] 将超出指定的界限，那么它就开始拉入来自S_i的时间戳晚于RVC_k[i]的写操作；

持续一致性：限定顺序偏差

- 暂时写操作的本地队列；
- 当本地队列的长度超过制定的最大长度时，服务器不再接受任何新提交的写操作，而是按照相应的顺序提交写操作；

基于主备份的协议

- 远程写协议



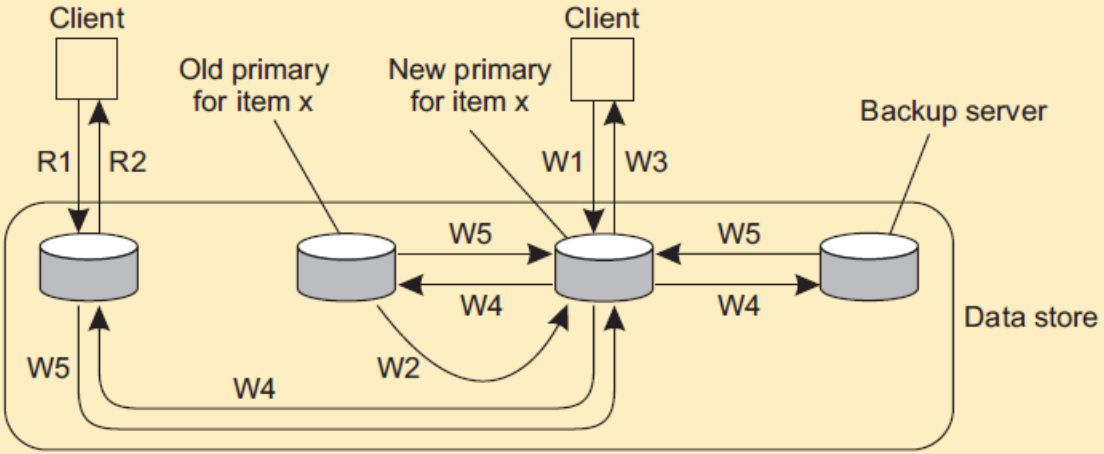
W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

传统的分布式数据库和文件系统需要较高的容错性，副本位于LAN

基于主备份的协议

- 本地写协议



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

主要应用于离线模式下的移动计算机，在断线之前传递相关文件；

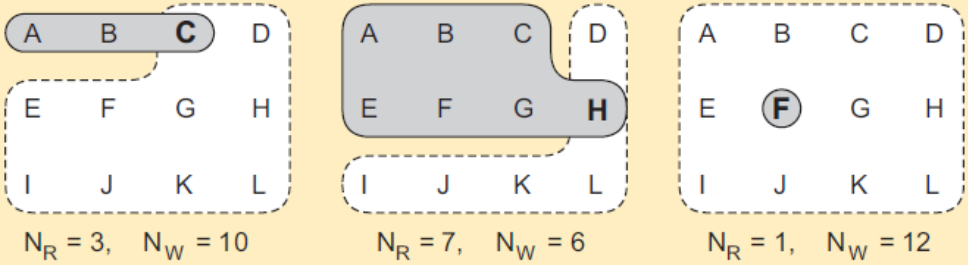
复制的写协议

(1) $N_R + N_W > N$;
(2) $N_w > N/2$。

Quorum-based protocols 基于团体的协议

Ensure that each operation is carried out in such a way that a majority vote is established: distinguish read quorum and write quorum

Three examples of the voting algorithm. (a) A correct choice of read and write set. (b) A choice that may lead to write-write conflicts. (c) A correct choice, known as ROWA (read one, write all)



作业

1. 请描述一个用于显示刚更新的Web页面的写读一致性的简单实现。

2. 简述使用Lamport逻辑时钟的全序多播不能扩展的原因。

3. 请实现一个支持多播的RPC的简单系统。假设系统有多个复制的服务器，每个客户可以通过RPC与一个服务器通信。但是处理复制时，客户需要向每一个副本发送一个 RPC请求。设计客户程序，使得客户好像只往应用程序发送单个RPC。假设复制的目的是为了提高性能，而那些服务器可能容易出故障（程序实现）。

谢谢！