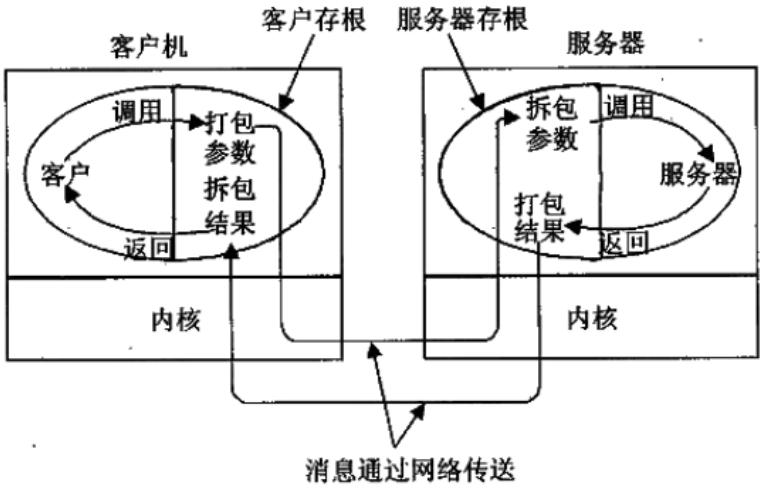
总结

1.分布式系统目标

PPT1：第二节

2.RPC 过程

PPT：分布式通信管理， 第二节



1. 客户过程以普通方式调用相应的客户存根
2. 客户存根建立消息并激活内核陷阱
3. 内核将消息发送到远程内核
4. 远程内核将消息送到服务器存根
5. 服务器存根取出消息中的参数后调用服务器的过程
6. 服务器完成工作后将结果返回值服务器存根
7. 服务器存根将它打包并激活内核陷阱
8. 远程内核将消息发送至客户端内核
9. 客户端内核将消息交给客户存根

10. 客户存根从消息中取出结果返回给客户

效果：讲客户过程对客户存根发出的本地调用转换成对服务器过程的本地调用，而客户和服务器都不会意识到有中间步骤的存在。

3.时间戳向量

PPT：分布式通信管理 第四节

时间戳向量：设组内有 n 个进程，每个进程设置一个 n 元向量 $V[n]=\{V[1],V[2],...,V[n]\}$

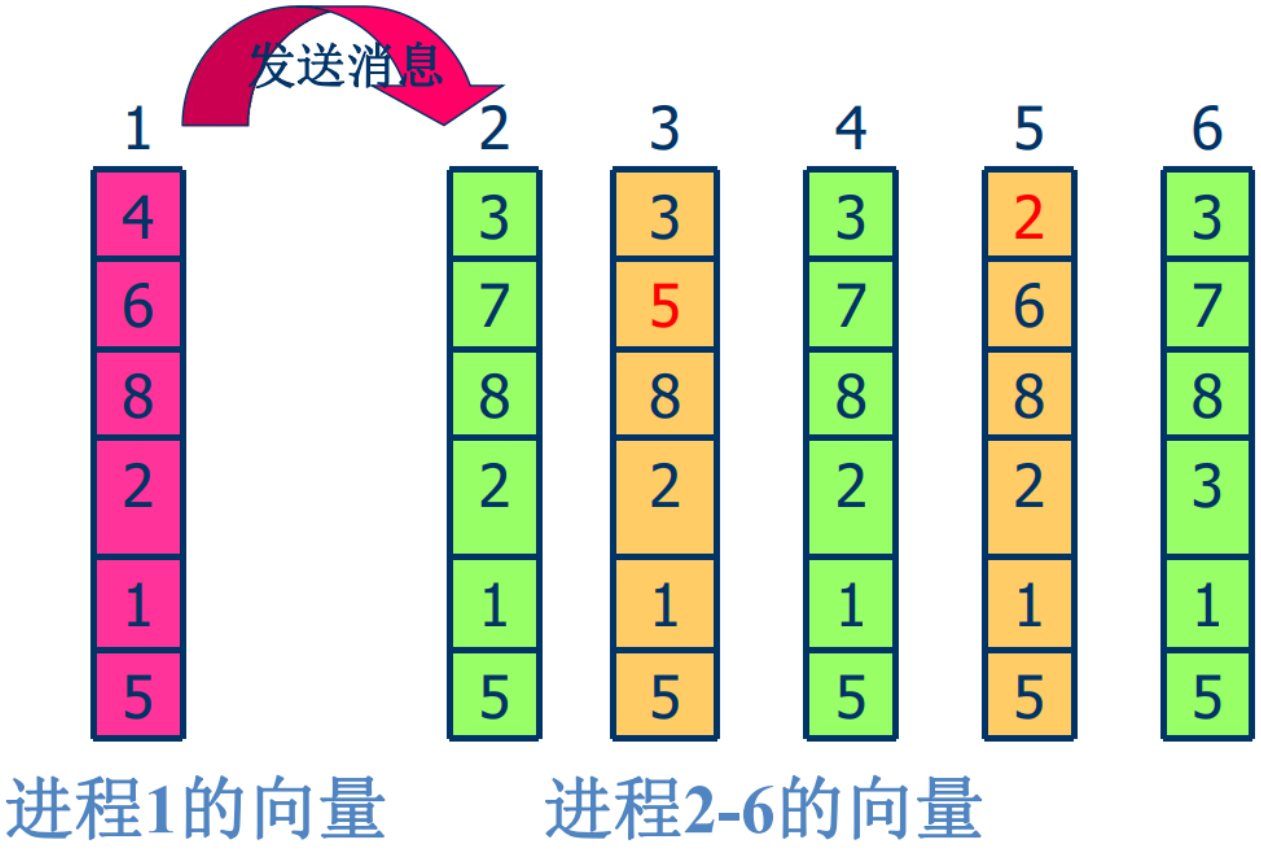
发送者：修改向量： $V[j]=V[j]+1;$ （第 j 个进程）

接收者：

- 接受条件： (1) $L[j]=V[j]-1$；(2) $L[i]\geq V[i](i\neq j)$，其中 $L[j]$ 表示本地向量的第 j 个维度
- 修改向量： $L[j]=V[j]$ 如果可以接收

其中， V_i 是发送方向量种的第 i 位， L_i 是接收方向量中的第 i 位，假设消息由进程 j 发送。

举例 1：对于下列表格表示的进程，由进程 1 向其他的进程发送消息，此时刻，哪些进程需要等待，哪些进行可以接收消息。



对于此题，需要分别分析两个条件是否满足。

对于条件（1）：消息由进程 1 发送，则首先需要满足 $V_1 = L_1 + 1$

也就是对于每个进程的第一维数据（4，3，3，3，2，3）而言，4=3+1，所以可以看出进程 5 不能接收消息，需要等待，其余进程均满足条件（1）。

对于条件（2）：消息由进程 1 发送，则需要满足 $i \neq j$（即所有进程排除第一维数据），有 $V_i \leq L_i$

所以对于 1–6 这几个进程而言，排除第一维数据后，其余的数据都需要比进程 1 的各维数据相等或更大，所以可以看出对于进程 3 而言 5<6，所以进程 3 不能接收消息，需要等待。

综上两个条件，进程 3，5 需要等待，其余进程均可同时满足条件（1）（2），所以可以接收进程 1 发送的消息。

示例 2：下面表格中分别是进程 1 到进程 6 中消息的时间戳向量，在满足因果关系的前提下，试分析说明进程 2 中当前的消息 m 在哪些进程中能够递交，在哪些进程中暂时不能递交。

1		2		3		4		5		6

7		7		7		5		7		7
5		7		6		6		6		6
5		5		5		5		5		5
4		4		4		4		3		4
3		3		3		3		3		3
1		1		1		1		1		1

就是进程 1，4，5 需要等待；3，6 可以接收消息。

4.转发指针

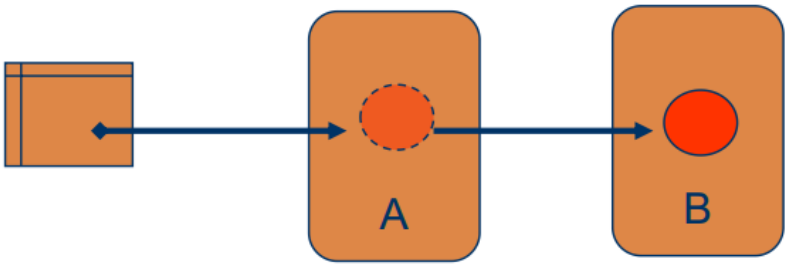
PPT：分布式命名管理 第二节

当实体从 A 移动到 B 之后，在 A 上设置一个指向 B 的引用

优点：客户可利用传统的命名服务

缺点：间址链可能会很长、链的中间节点需要维护转发信息、链容易断

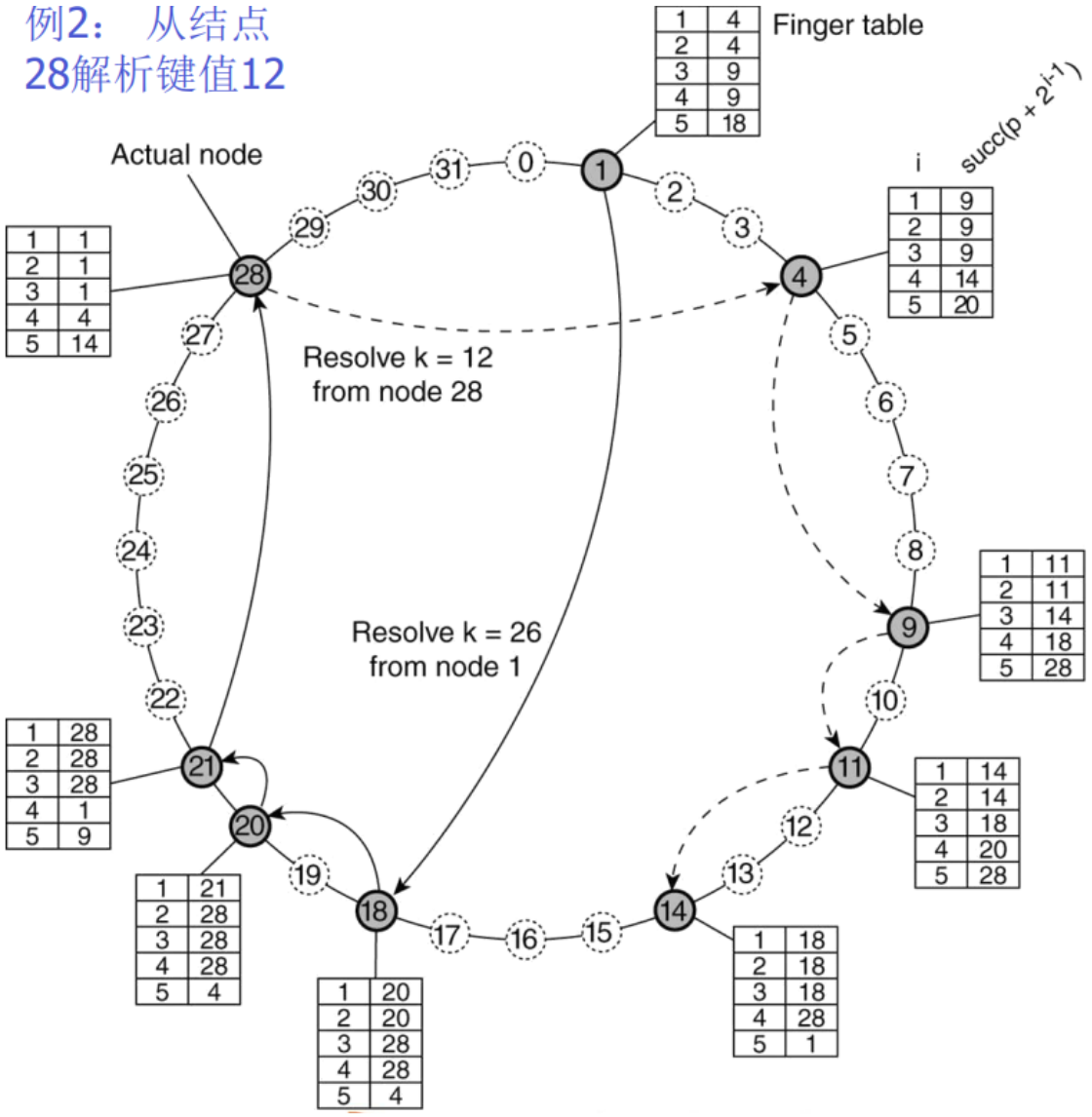
目标：限制链的长度，保证链的鲁棒性



5.DHT

PPT：分布式命名管理 第二节

例2： 从结点28解析键值12



		1	4	9	11	14	18	20	21	28
1	1	4	9	11	14	18	20	21	28	1
2	2	4	9	11	14	18	20	28	28	1
3	4	9	9	14	18	18	28	28	28	1
4	8	9	14	18	20	28	28	28	1	4
5	16	18	20	28	28	1	4	4	9	14

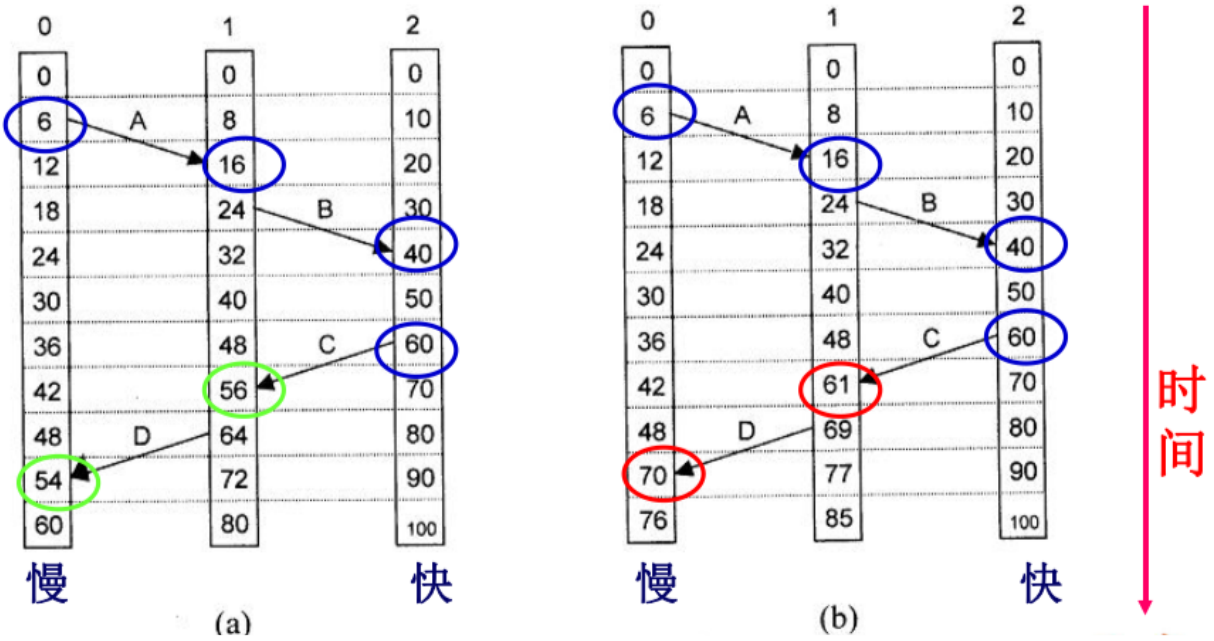
6.lamport 算法

PPT：分布式同步控制 第二节

校正算法

- $a \rightarrow b$
- if $C(b) < C(a)$, then $C(b) = C(a) + 1$

三个进程，各有自己的局部时钟，他们速率不同；通过 Lamport 算法，校正时钟



校正算法：

1. P_i 在执行一个事件之前， P_i 执行 $C_i \leftarrow C_i + 1$
2. P_i 在发送消息 m 给 P_j 时，时间戳 $ts(m) \leftarrow C_i$
3. P_j 接收到消息 m 后， $C_j \leftarrow max\{C_j, ts(m)\}$

7.选举算法

PPT：分布式同步控制 第四节

8.以数据为中心的一致性模型

8.1 连续一致性

连续一致性（continuous consistency）

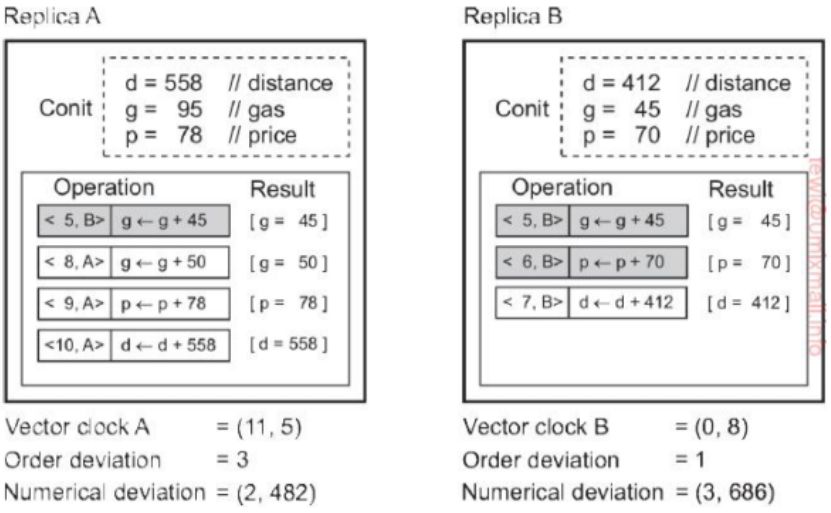
- （数值误差，次序误差，陈旧度）
- **数值误差**：未传播的写操作的权重值
- **次序误差**：临时写操作的个数
- **陈旧度**：写操作传播的延迟(只要一个副本不太旧，就可以容忍它提供旧的数据。例如，天气报告）

一致性单元（conit）：受控的数据集

顺序偏差即在本地还没有提交的更新操作的个数

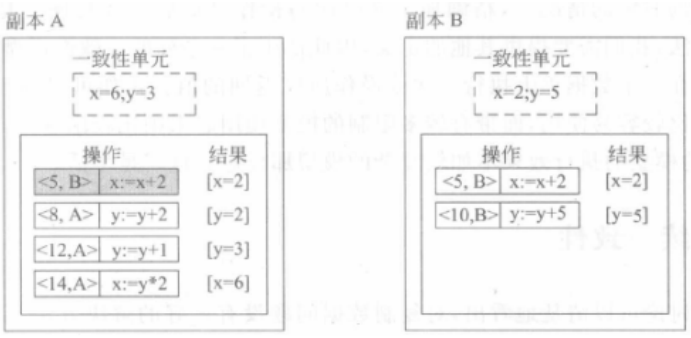
数值偏差（x,y）： x 表示其他副本已经做了， 但是本地副本还没看见的操作数量；Y 表示这些操作带来的数值变化

举例：conit(x, y)



<5, B>：5 为时间戳，B 为从 B 传过来

示例 1：如下图所示，为持续一致性的示意图。在副本 A 和副本 B 中有含数据项 x 和 y，这两个变量都假设初始化为 0。操作表格中阴影部分表示为持久化（永久性）的操作，不能回滚；其余操作表示暂时的更新操作。请采用向量时钟、顺序误差和数值误差三个指标描述两个副本之间的差异。（其中数值误差用各数据项的差分和表示）



(1) 向量时钟：

- 对于副本 A 而言，由于副本 A 和副本 B 中有含数据项 x 和 y，这两个变量都假设初始化为 0，副本 A 最后的操作是<14, A>，所以 A 的向量时钟： A = （15, 5）
- 对于副本 B 而言，由于副本 A 和副本 B 中有含数据项 x 和 y，这两个变量都假设初始化为 0，副本 A 最后的操作是<10, B>，所以 B 的向量时钟：B = （0, 11）

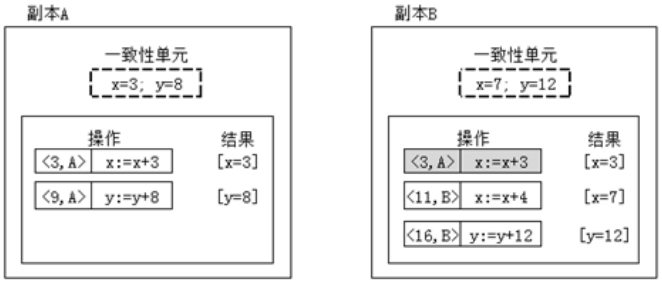
(2) 顺序偏差

- 对于副本 A 而言，阴影部分表示为持久化（永久性）的操作，不能回滚，而 A 中不是阴影部分的有三个，即<8, A>, <12, A>, <14, A>，所以 A 中有三个暂存的操作，即顺序偏差为 3
- 对于副本 B 而言，阴影部分表示为持久化（永久性）的操作，不能回滚，而 B 中不是阴影部分的有二个，即<5, B>, <10, B>，所以 B 中有二个暂存的操作，即顺序偏差为 2

(3) 数值偏差

- 对于副本 A 而言，A 已经收到了 B 的<5, B>，但是还未看到<10, B>，所以数值偏差=1（即还有一个没有收到）；而此时 A 中已提交的值是（x, y）=(2, 0)，假设收到 B 的<10，B>之后，y=5；所以副本 A 的数值偏差=（1，5）
- 对于副本 B 而言，B 还未看到 A 中的<8, A>，<12, A>, <14, A>，所以数值偏差=3（即还有三个没有收到）；而此时 B 中已提交的值为（x，y）=（0，0），假设收到 A 的<8, A>，<12, A>, <14, A>之后，x=(1+2)*2，所以偏差的权重= 6；所以副本 B 的数值偏差=（3，6）

示例 2：如下图所示，为持续一致性的示意图。在副本 A 和副本 B 中有含数据项 x 和 y，这两个变量都假设初始化为 0。操作表格中阴影部分表示为持久化（永久性）的操作，不能回滚；其余操作表示暂时的更新操作。请采用向量时钟、顺序误差和数值误差三个指标描述两个副本之间的差异。（其中数值误差用各数据项的差分和表示）



A：向量时钟：(10, 0)；顺序偏差：2 ； 数值偏差：(2, 16)

B：向量时钟：(3, 17)；顺序偏差：2 ； 数值偏差：(1, 8)

8.2 一致性模型

- **严格一致性**：对数据项 x 的读操作返回的值为最近写入 x 的值
- **顺序一致性**：所有进程执行的结果，等同于它们的操作按某种顺序在数据仓上执行的结果。每个进程的操作都按照程序规定的顺序。

9.容错性的基本概念

分布式系统设计中的一个重要目标，是以这样的方式构建系统：它可以从部分失效中自动恢复，而且不会严重的影响整体性能。特别是，当故障发生时，分布式系统应该在进行恢复的同时继续以可接受的方式进行操作，也就是说，它应该能容忍错误，在发生错误时某种程度上可以继续操作。

第 8 章 分布式系统之容错 – 简书 (jianshu.com)

10.lamport 递归算法

PPT：分布式容错管理 第二节