第3章

网格计算和云计算

网络计算模式

- 课程主要内容
  - 概述
  - 企业计算
  - 网格计算和云计算
  - P2P网络、CDN网络和物联网
  - 社会计算

数据一致性理论

CAP理论

其中字母“C”、“A”、“P”分别带表以下特征：

✦强一致性（**C**onsistency）。系统在执行某项操作后仍然处于一致状态。在分布式系统中，更新操作执行成功后所有用户都应该读取到最新的值，这样的系统被认为具有强一致性。

✦可用性（**A**vailability）。每一个操作总是能够在**一定的时间**内**返回结果**。

✦分区容错性（**P**artition Tolerance）。分区容错性可以理解为系统在存在网络分区的情况下仍然可以接受请求（满足一致性和可用性）。

数据一致性理论

CAP理论

✦CAP理论是在分布式环境中设计和部署系统时需要考虑的三个重要的系统需求。根据CAP理论，数据共享系统**只能满足这三个特性中的两个**，而不能同时满足三个条件。因此系统必须在这三个特性之间做出权衡。

✦如何理解?

数据一致性理论

CAP理论

✦在网络分区场景下，如发生网络故障，分区G1发送的数据消息不能传达到另一分区G2，数据将处于不一致状态，不能满足一致性要求。采用如阻塞、加锁、集中控制等技术来保证数据的一致性，但同步操作必定要消耗一定时间，必然会影响到系统的可用性和分区容错性，尤其在网络规模较大的时候。

序号	选择	特点	例子
1	C、A	两阶段提交、缓存验证协议	传统数据库
2	C、P	悲观加锁	分布式加锁
3	A、P	冲突处理、乐观	DNS

数据一致性理论

CAP理论

序号	选择	特点	例子
1	C、A	两阶段提交、缓存验证协议	传统数据库
2	C、P	悲观加锁	分布式加锁
3	A、P	冲突处理、乐观	DNS

✦放弃P：如果想避免分区容错性问题的发生，一种做法就是将数据放到一台机器上。虽然无法100%地保证系统不会出错，但不会碰到由分区带来的负面效果。当然，这个选择会严重影响系统的可扩展性。

数据一致性理论

CAP理论

序号	选择	特点	例子
1	C、A	两阶段提交、缓存验证协议	传统数据库
2	C、P	悲观加锁	分布式加锁
3	A、P	冲突处理、乐观	DNS

✦放弃A：相对于放弃“分区容错性”，其方面就是放弃可用性。一旦遇到分区容错故障，那么受到影响的服务需要等待数据一致，因此等待期间系统无法对外提供服务。

数据一致性理论

CAP理论

序号	选择	特点	例子
1	C、A	两阶段提交、缓存验证协议	传统数据库
2	C、P	悲观加锁	分布式加锁
3	A、P	冲突处理、乐观	DNS

✦放弃C：这里所说的放弃一致性，并是不完全放弃数据的一致性，而是放弃**数据的强一致性**，而保留**数据的最终一致性**。以网络购物为例，对只剩最后一件库存商品，如果同时接收了两份订单，较晚的订单将被告知商品售罄。

数据一致性理论

最终一致性模型

✦分布式系统一般通过复制数据来提高系统的可靠性和容错性，并且将数据的不同副本存放在不同的机器上，由于维护数据副本一致性代价很高，许多系统采用弱一致性来提高性能。不同的一致性模型：

✦强一致性

✦弱一致性

✦最终一致性

数据一致性理论

最终一致性模型

✦强一致性：要求无论更新操作在哪个数据副本上执行，之后所有的读操作都要获得最新的数据。对于单副本数据来说，读写操作在同一数据上执行，容易保证强一致性。对于多副本数据来说，需要使用分布式事物协议（如Paxos）。

✦弱一致性：在这种一致性下，用户读到某一操作对系统特定数据的更新需要一段时间。这段时间成为“不一致性窗口”。

数据一致性理论

最终一致性模型

✦最终一致性：是弱一致性的一种特例，在这种一致性下系统保证用户最终能够读取到某操作对系统特定数据的更新。

✦此种情况下，“不一致性窗口”的大小依赖于交互延迟、系统负载，以及复制技术中replica的个数（可以理解为master/slave模式中，slave的个数）。

✦DNS系统是在最终一致性方面最出名的系统，当更新一个域名的IP以后，根据配置策略以及缓存控制策略的不同，最终所有客户都会看见最新值。

数据一致性理论

最终一致性模型

- 最终一致性模型根据其提供的保证可以划分为多个模型：
- 因果一致性（Causal Consistency）：假如有相互独立的A、B、C三个进程对数据进行操作。进程A对某数据进行更新后并将该操作通知给B，那么B接下来的读操作能够读到A更新的数据值。但是由于A没有将该操作通知给C，那么系统将不保证C一定能够读取到A更新的数据。

数据一致性理论

最终一致性模型

✦最终一致性模型根据其提供的保证可以划分为多个模型：

✦读自写一致性（Read Your Own Writes Consistency）：用户更新某个数据后，读取该数据时能够获取其更新后的值，而其他用户读取该数据时则不能保证读取到最新值。

✦会话一致性（Session Consistency）：指读自写一致性被限制在一个会话范围内，也就是说提交更新操作的用户在同一个会话里读取该数据时能够保证数据是最新的。

数据一致性理论

最终一致性模型

- 最终一致性模型根据其提供的保证可以划分为多个模型：
- 单调读一致性（Monotonic Read Consistency）：指用户读取某个数据值，后续操作不会读取到该数据更早版本的值。
- 时间轴一致性（Timeline Consistency）：要求数据的所有副本以相同顺序执行所有更新操作，也称为单调写一致性（Monotonic Write Consistency）。

数据一致性理论

ACID 与 BASE

CAP理论指出一致性、可用性和分区容错性不能同时满足。对于数据不断增长的系统（如社会计算、网络服务的系统），他们对可用性及分区容错性的要求高于强一致性，并且很难满足事务所要求的ACID特性，因此BASE理论被提出。

数据一致性理论

ACID 与 BASE

事务是用户定义的一个**数据库操作序列**，这些操作要么全不做，要么全做，是一个不可分割的单位，ACID是事务所具有的特性。

✦原子性（Atomicity）：事务中的操作要么都做，要么都不做。

✦一致性（Consistency）：系统必须始终处于强一致状态下。

✦隔离性（Isolation）：一个事务的执行不能被其他事务所干扰。

✦持续性（Durability）：一个已提交的事务对数据库中数据的改变是永久性的。

ACID特性是传统关系型数据库中事务管理的重要任务，也是**恢复和并发控制的基本单位。**

数据一致性理论

ACID 与 BASE

BASE方法通过**牺牲一致性和孤立性**来**提高可用性和系统性能**，其中BASE分别代表：

✦基本可（**B**asically **A**vailability）：系统基本能够运行、一直提供服务。

✦软状态（**S**oft-state）：系统不要求保持强一致性。

✦最终一致性（**E**ventually consistency）：系统需要在某一刻后达到一致性要求。

数据一致性理论

ACID 与 BASE

ACID和BASE的比较：

ACID	BASE
强一致性	弱一致性
隔离性	可用性优先
采用悲观、保守的方法	采用乐观方法
难以变化	适应变化、更简单、更快

数据一致性理论

数据一致性实现技术

分布式系统在不同节点的数据采用什么技术保证一致性，取决于应用对系统一致性的需求，在关系型数据管理系统中一般会采用悲观的方法（如加锁），这些方法代价比较高、对系统性能影响较大，而在一些强调性能的系统中则会采用乐观的方法。

- Quorum系统NWR策略
- 两阶段提交协议
- 时间戳策略
- PAXOS算法
- …

数据一致性理论

Quorum系统NRW策略

对于数据在不同副本中的一致性，此策略采用类似于Quorum系统的一致性协议实现。这个协议有三个关键值N、R、W。

- N表示数据所有副本数。
- R表示完成读操作所需要读取的最小副本数，即一次读操作所需参与的最小节点数目。
- W表示完成写操作所需要写入的最小副本数，即一次写操作所需参与的最小节点数目。

该策略中，**只需要保证R+W>N，就可以保证强一致性。**

数据一致性理论

Quorum系统NRW策略

✦R+W>N会产生类似Quorum的效果。该模型中的读（写）延迟由最慢的读（写）副本决定，为了获得较高的性能和较小的延迟，R和W的和可能小于N，这时系统不能保证读操作能获取最新的数据。

✦如果R+W>N，那么分布式系统提供强一致性保证，因为读取数据的节点和被同步写入的节点是有重叠的。

✦如果R+W<=N，这时执行读取和写入操作的节点可能没有重叠，系统只能保证最终一致性。副本到达一致的时间则依赖于系统异步更新的实现方式。

数据一致性理论

Quorum系统NRW策略

- R和W的设置直接影响系统的性能、扩展性和一致性。
- 如果W设置为1，则一个副本完成更改就可以返回给用户；如果R设置为1，只要有一个副本被读取就可以完成读操作。然后通过异步的机制更新剩余的N-W个副本。
- R和W的值如果**较小会影响一致性，较大则会影响性能**，因此对这两个值得设置需要权衡。

数据一致性理论

Quorum系统NRW策略

几种特殊情况的设置：

✦当W=1，R=N时，系统对写操作有较高的要求，但读操作会比较慢，若N个节点中有节点发生故障，那么读操作将不能完成。

✦当R=1，W=N时，系统要求读操作高性能、高可用，但写操作性能较低，用户需要大量读操作的系统，若N个节点中有节点发生故障，那么写操作将无法完成。

✦当R=W=Q（Q=N/2+1）时，系统在读写性能之间取得了平衡，兼顾性能和可用性。亚马逊Dynamo系统默认设置为N=3，W=2，R=2。

数据一致性理论

两阶段提交协议

两阶段提交协议（Two Phase Commit Protocol，2PC协议）可以保证数据的强一致性，许多分布式关系型数据库管理系统采用此协议完成分布式事务。

✦两阶段提交协议系统一般包含两类节点：一类为**协调者**，通常一个系统中只有一个；另一类为事务**参与者**，一般包含多个，在数据存储系统中可以理解为数据副本的个数。

数据一致性理论

两阶段提交协议——前提条件

- 协议中假设每个节点都会记录写前日志（Write-ahead log）并持久保存，即节点发生故障日志也不会丢失。
- 协议中还假设节点不会发生永久性故障，而且任意两个节点都可以相互通信。

数据一致性理论

☙两阶段提交协议——执行过程

分两个阶段：

✦**阶段1：请求阶段（或称表决阶段）**：协调者将通知事务参与者准备提交或取消事务，然后进入表决过程。在表决过程中，参与者将告知协调者自己的决策：同意（事务参与者本地作业执行成功）或取消（本地作业执行发生故障）。

✦**阶段2：提交阶段**：协调者将基于第一阶段的投票结果进行决策：提交或取消。当且仅当所有参与者都同意提交，事务协调者才通知所有参与者提交事务，否则协调者将通知所有参与者取消事务。参与者在接收到协调者发来的消息后将执行相应操作。

数据一致性理论

☙两阶段提交协议——缺点

- ✦此协议最大缺点在于它是通过**阻塞**完成的协议。节点在等待消息的时候处于阻塞状态，节点中的其他进程则需要等待阻塞的进程释放资源。
- ✦如果协调者发生永久故障?
- ✦如果参与者发生永久故障?

数据一致性理论

☙两阶段提交协议——变种协议

为解决实际问题（如，大部分参与者在阶段1同意提交事务，则协调者在阶段2通知所有参与者同意提交），存在变种协议：

- ✦树形（递归）两阶段提交协议
- ✦动态两阶段提交协议

数据一致性理论

☙时间戳策略

✦基于时间戳，可为每一份数据附加一个时间戳标记，在进行数据版本比较或数据同步的时候只需要比较其时间戳就可以区分它们的版本。

✦问题解决！？

数据一致性理论

☙时间戳策略

✦基于时间戳，可为每一份数据附加一个时间戳标记，在进行数据版本比较或数据同步的时候只需要比较其时间戳就可以区分它们的版本。

✦在分布式系统中会出现什么状况?

数据一致性理论

☙时间戳策略

- ✦在分布式系统中会出现什么状况?
- ✦不同节点之间的物理时钟可能会有偏差。如何处理?
- ✦设置全局时钟进行时间同步。但全局时钟开销过大，尤其是大规模分布式系统，影响系统效率。
- ✦若全局时钟宕机，系统无法工作。
- ✦全局时钟成为系统效率和可用性的瓶颈。怎么办?

改进时间戳，不依赖单个机器，也不依赖物理时钟同步。该时间戳为逻辑上的时钟。

数据一致性理论

☙时间戳策略——逻辑时间戳

逻辑时间戳用来确定分布式系统中事件的**先后关系**。假设发送或接收进程中的一个事件，分布式系统中事件的先后关系可用“->”来表示，例如：若事件a发生在事件b之前，则a->b。该关系满足：

✦如果事件a和事件b是同一进程中的事件，并且a在b之前发生，那么a->b。

✦如果事件a是某消息发送方进程中的事件，事件b是该消息接收方进程中接收该消息的事件，那么a->b。

✦对于事件a、事件b和事件c，如果有a->b和b->c，那么a->c。

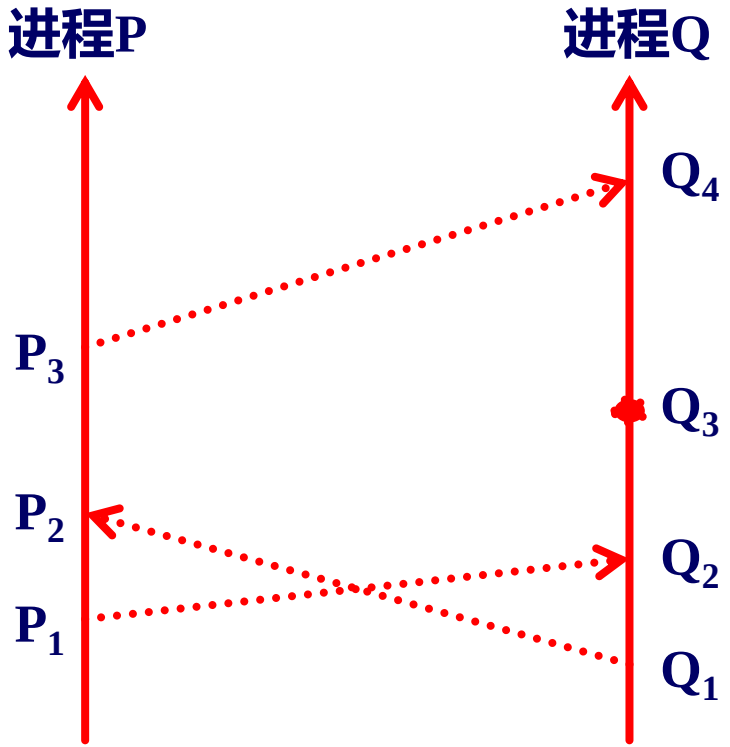
✦如果两个不同的事件a和事件b，既不能得出a->b也不能得出b->a，那么逻辑上认为事件a和事件b同时发生。

数据一致性理论

☙时间戳策略——逻辑时间戳

例1：事件P_1、事件P_2

在同一进程P中，事件P_2发生在事件P_1之后，因此P_1 -> P_2。

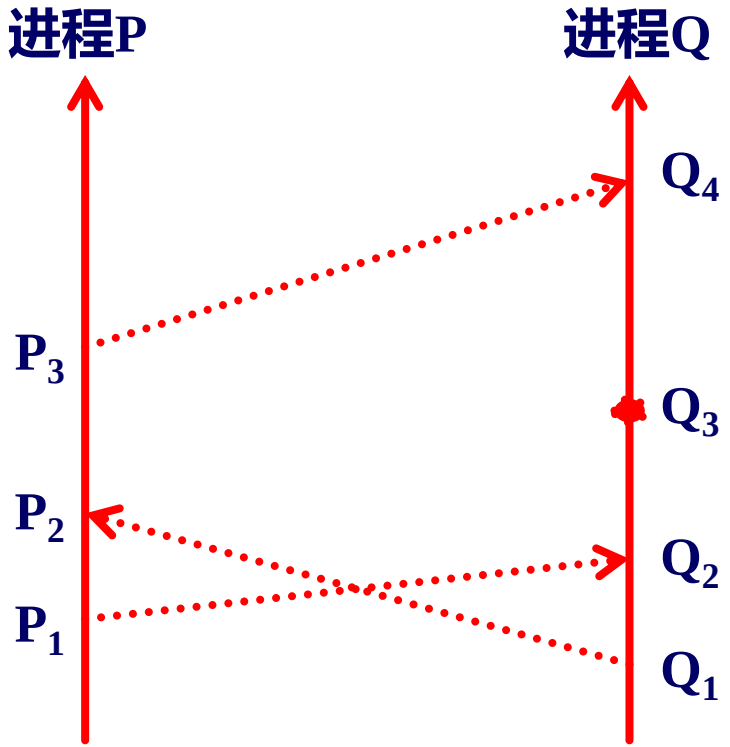


数据一致性理论

☙时间戳策略——逻辑时间戳

例2：事件Q_1、事件P_3

对于事件Q_1和事件P_3，由于存在从Q_1到P_2的消息传递，因此Q_1->P_2，同时在同一进程P中，我们知道P_2 ->P_3，因此，根据该模型，Q_1 ->P_3。

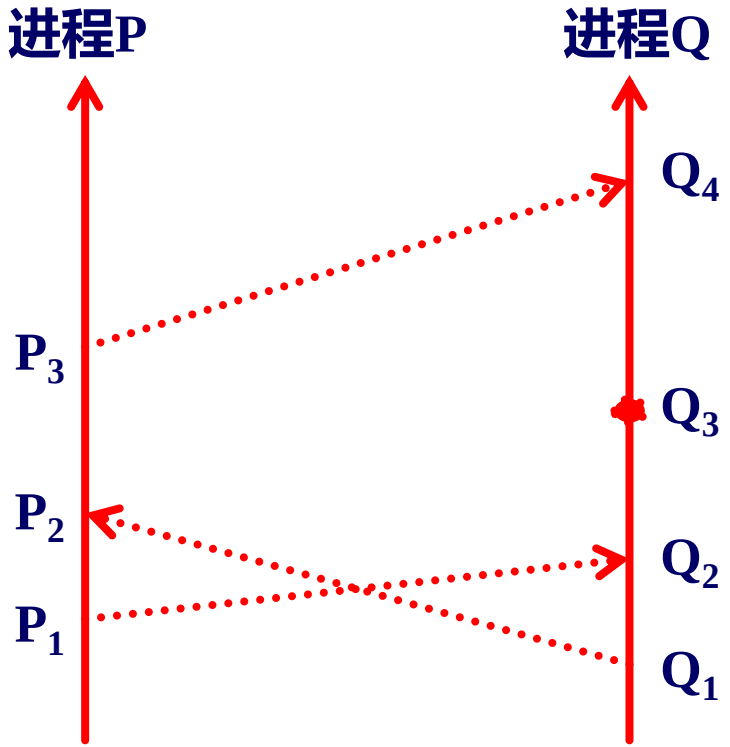


数据一致性理论

☙时间戳策略——逻辑时间戳

例2：事件P_3、事件Q_3

在逻辑上同时发生。



数据一致性理论

时间戳策略——逻辑时钟

将时钟引入系统：

✦为每个进程P_i定义一个时钟C_i，该时钟能够为任意一个事件a分配一个时钟值：C_i（a）。

基于此，可给出下面两个时钟**限制条件**：

✦**C1**：如果事件a和事件b是同一个进程P_i中的事件，并且a在b之前发生，那么C_i（a）<C_i（b）。

✦**C2**：如果a为进程P_i上某消息发送事件，b为进程P_j上该消息接收事件，那么C_i（a）<C_j（b）。

数据一致性理论

时间戳策略——应用

若进程为分布式系统中的节点，每个节点i均包含一个时钟C_i，系统中包含两类事件：一类为节点上数据更新；另一类为节点之间的消息通信（或数据同步）。

如何满足C1和C2这两个时钟**限制条件**：

✦**C1**：如果事件a和事件b是同一个进程P_i中的事件，并且a在b之前发生，那么C_i（a）<C_i（b）。

✦**C2**：如果a为进程P_i上某消息发送事件，b为进程P_j上该消息接收事件，那么C_i（a）<C_j（b）。

数据一致性理论

☙时间戳策略——应用

若进程为分布式系统中的节点，每个节点i均包含一个时钟C_i，系统中包含两类事件：一类为节点上数据更新；另一类为节点之间的消息通信（或数据同步）。

对于条件C1，系统实现需要满足:

✦**IR1**：对于同一节点上任意的连续事件来说，该节点上的时钟只需要保证较晚发生事件的时钟大于较早发生事件的时钟即可。

数据一致性理论

☙时间戳策略——应用

若进程为分布式系统中的节点，每个节点i均包含一个时钟C_i，系统中包含两类事件：一类为节点上数据更新；另一类为节点之间的消息通信（或数据同步）。

对于条件C2，系统实现需要满足:

✦当节点发送消息m时，该消息需要同时携带发送时刻在该节点产生的时间戳。在接收方收到消息m后，接收方节点所产生的时间戳要大于m所携带的时间戳。

✦此条件是否充分?

数据一致性理论

☙时间戳策略——应用

对于条件C2，考虑如下场景:

✦假设某节点A向节点B发送消息m，在发送消息时刻节点A的本地时间为15:33:30，那么m所携带的时间戳可能为T_{m_a}=153330。节点B在接收到m后，可能设置该事件的时间戳T_{m_b}为15340。假如机器之间存在时间误差，此时B节点的系统时间为15:50:05，这将引起逻辑上的错误，因为在153400到155005之间可能有其他事件发生，这些事件本来早于接收消息m事件，却被错误的分配了比153400更高的时间戳。

数据一致性理论

☙时间戳策略——应用

若进程为分布式系统中的节点，每个节点i均包含一个时钟C_i，系统中包含两类事件：一类为节点上数据更新；另一类为节点之间的消息通信（或数据同步）。

因此，对于条件C2，系统实现需要满足:

✦**IR2**：（a）如果事件a代表节点N_i发送消息m，那么消息m将携带时间戳T_m，且$T_m=C_i$（a）；（b）当节点N_j接收到消息m后，节点将设置该事件的时钟C_j大于或等于该节点上一事件的时钟，且大于或等于T_m。

数据一致性理论

Paxos算法

Paxos算法常用于具有较高容错性的分布式系统，其核心就是一致性算法，该算法解决的主要问题是**一个分布式系统如何就某个值（决议）达成一致。**

✦分布式系统Hadoop中的Zookeeper为Paxos算法的开源实现。

数据一致性理论

Paxos算法——问题

假设许多节点都可以提交决议（value），一致性算法需要保证在所有被提出的决议中只有一个能被接受。某个协议一旦被选择，其他节点能够知道所选择的决议。一致性的保证需要满足以下条件：

- 决议只有被提出后才能被选择。
- 算法的执行实例中只能选择一个决议。
- 决议只有被选中后才能让其他节点（learner）知道。

数据一致性理论

Paxos算法

Paxos算法中有三种角色：

- 提议者（proposer）：提出决议。
- 批准者（acceptor）：负责批准决议。
- 学习者（learner）：主要学习决议。

在具体实现中，一个节点可以担当多个角色。

数据一致性理论

Paxos算法

假设节点之间通过发送信息进行通信，这里使用异步、非拜占庭模型。在该模型中：

- 节点以任意的速度进行操作，可能因为故障而停止，也可以重新启动。并且节点所选择的决议不会因为重启等故障消失。
- 消息可以延迟发送、多次发送或丢失，但不会被篡改（即不存在拜占庭问题）。

数据一致性理论

Paxos算法——选择决议

选择决议的最简单方法就是采用唯一批准者（如，批准接收到的第一条决议）。

但是，一旦批准者发生故障，将无法继续后续过程，故需采用多个批准者的方法，此时提供规则如下：

✦**R：当决议被大多数批准者所批准，则表明该决议被批准。**

✦如果一个批准者最多只能批准一个决议，“**大多数**”设置为超过半数即可，保证系统只有一个决议被批准。

数据一致性理论

Paxos算法——选择决议

- **R：当决议被大多数批准者所批准，则表明该决议被批准。**
- 假如只有某一个提议者提出了一个决议，如果我们希望该决议被接受，则需要：
- **P1：一个批准者必须批准它所接收到的第一个决议。**
- 这会产生问题：同一时刻不同的提议者可能会发出不同的决议，每个批准者都批准了一份决议，但没有哪一个决议达到半数以上。

数据一致性理论

Paxos算法——选择决议

在一个系统中，一个批准者应该需要被设置为可以批准多个决议。以数据存储为例，分布式系统中不可能只存一份数据。为区分批准者所批准的决议，每个决议都分配有**版本号**和**提议值**。

因此，某个决议被**选定**是指拥有该提议值的决议被大多数批准者批准。

数据一致性理论

Paxos算法——选择决议

（1）决议只有被proposers提出后才能批准

（2）每次只批准一个决议

（3）只有决议确定被批准后learners才能获取这个决议

保证数据的一致性

数据一致性理论

Paxos算法——选择决议

系统约束条件

P1表明一个批准者可以收到多个决议，为区分，对每个决议进行编号，后到的决议编号（版本号）大于先到的决议编号；约束条件P1不完备！

P2：一旦某个决议得到通过，之后通过的决议必须和该决议保持一致

P2a和P1是有矛盾的！

批准的决议必须是v

P1和P2b保证条件（2），彼此之间不存在矛盾。但是P2b很难通过一种技术手段来实现它，因此提出了一个蕴涵P2b的约束P2c

P2c：如果一个编号为*n*的决议具有值v，那么存在一个“多数派”，要么它们中没有谁批准过编号小于*n*的决议，要么它们进行的最近一次批准具有值v。

数据一致性理论

Paxos算法——选择决议

结合提议者和批准者两方面的操作，Paxos算法将包括两个阶段：

✦阶段1：准备阶段：

✦提议者选择一个恰当的版本号n，并发送一个版本号n的“准备请求”到一个批准者的大多数集。

✦如果某个批准者接收到该“准备请求”，并且该请求的版本号n大于该批准者之前所响应过的任意“准备请求”，那么此批准者将向该提议者承诺不会再响应任何版本号低于n的决议，并告知提议者它曾经批准过的决议的最高版本号。

数据一致性理论

Paxos算法——选择决议

结合提议者和批准者两方面的操作，Paxos算法将包括两个阶段：

✦阶段2：批准阶段：

✦如果提议者接收到来自大多数集的对于其关于版本号为n的“准备请求”的响应信息，那么他将向该大多数集发送关于“版本号为n、值为v的决议”的“批准请求”。如果该响应不为空，其中v的值为该响应信息中最高版本号决议的值；如果该响应信息为空，提议者可以自主指定新的值。

✦如果某批准者接收到了关于版本号为n的决议的“批准请求”，那么除非它曾响应过版本号大于n的决议的“准备请求”，否则它将批准该决议。

数据一致性理论

Paxos算法——学习选择决议

学习者需要找到被大多数集批准的决议：

✦每当批准者批准一个决议，就将其发送给每一个学习者。

✦批准者将批准的决议信息发送给某几个学习者，再由学习者转发其他学习者。

✦学习者也可以是提议者。

数据一致性理论

Paxos算法——其他问题

系统可能出现这样的情况：有两个不同的提议者不断尝试提出更高版本的决议，导致任何一个决议都不会被批准（Paxos算法在第二阶段失败）。

✦针对此情况：系统需要选出某个提议者（要求此提议者能和一个大多数集进行有效的通信）作为唯一的提议者来发出决议。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况1**：A提议爬长城时间定为下周三，并且无人与其竞争。

✦阶段1：A向所有人发出一个决议，包含版本号和内容，其格式定义为：[内容]:[版本号]。那么，该决议为“下周三：2012年3月2日，12:00”。所有人员发出该“准备请求”。B、C、D和E在收到该“准备请求”之后向A回复同意该决议。

✦阶段2：A在收到B、C、D回复后，发现自己的决议被大多数人接受，决议被批准，向所有人广播该决议。

✦结果：A的决议被批准，爬山时间定为下周三。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况2**： A提议爬长城时间定为下周三，同时E提议爬长城时间定为下周五，并且A、E在同一时刻发起决议（即版本号相同）。由于某种原因，A只能与B、C通信，而不能与D、E通信。同时，E只能与C、D通信，而不能与A、B通信。

✦阶段1：A向所有人发出决议“下周三：2012年3月2日，12:00”，同时E向所有人发出决议“下周五：2012年3月2日，12:00”。B在收到A发来的决议后，由于还未接收过任何决议，因此向A回复同意该决议；同样，D也同意E的决议。那么最终决议版本将取决于C的决定。（待续）

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况2**:

✦阶段1：（续）假如A的决议首先被C所接收，此时，C还未收到过任何决议，那么A的决议将被批准。当E的决议到达时，由于C已经批准了相同版本号的A决议，那么C将拒绝E的决议，并回复E：“已经有相同版本号的决议被提出”。E在接收到该回复后将放弃自己的决议。

✦阶段2：A在收到大多数集（B，C）的批准回复后，将再次向大多数集发送“批准请求”，此时没人竞争，A的决议将被批准。

✦结果：A的提议被批准，爬山时间定为下周三。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况3**： A提议爬长城时间定为下周三，同时E提议爬长城时间定为下周五，并且A、E在不同时刻发起决议（即版本号不同）。由于某种原因，A只能与B、C通信，而不能与D、E通信。同时，E只能与C、D通信，而不能与A、B通信。

✦阶段1：A向所有人发出决议“下周三：2012年3月2日，12:00”，半小时候E向所有人发出决议“下周五：2012年3月2日，12:30”。与情况2类似，B、D均能批准A、E的决议。那么关键在于C批准了谁的决议。（待续）

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦情况3：

✦阶段1：（续）假如A的决议优先到达C处，那么C在看到A的决议后将首先批准A的决议。之后，在C看到E发来的决议后，发现E决议的版本号高，则向E回复同意该决议。

✦阶段2：此时A和E的决议同时得到了一个系统中大多数集的同意。下面A和E都能够进入批准阶段，并向系统的大多数集发送“批准请求”。C在接收到发自A的批准请求时，将会告诉A已经有更高版本的决议被E提出。那么A将取消自己的决议。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况3：**

✦结果：E的决议被批准，爬山时间定为下周五。

✦分析：假如在第一阶段C首先接收到来自E的请求？

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况4**： A提议爬长城时间定为下周三，同时E提议爬长城时间定为下周五，并且A、E在同一时刻发起决议（即版本号相同）。由于某种原因D失去了联系，并且A、E均能与B、C进行正常通信，但A、E不能通信。

✦阶段1： A向所有人发出决议“下周三：2012年3月2日，12:00”，同时E向所有人发出决议“下周五：2012年3月2日，12:00”。假如B、C接收决议的顺序按如下序列进行，我们来分析最终的结果。

（待续）

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况4**:

✦阶段1：（续）（1）B接收到来自A的决议，那么B批准A的决议，并回复A；（2）C接收到来自E的决议，那么C批准E的决议，并回复E；（3）B接收到来自E的决议，并发现该决议与批准的A决议的版本号相同，故拒绝E的决议，并回复E已经有相同版本号的决议被提出；（4）C接收到来自A的决议，并发现该决议与批准的E决议的版本号相同，故拒绝A的决议，并回复A已经有相同版本号的决议被提出。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况4**:

✦阶段2：由于A和E都没有收到一个大多数集的回复，那么A、E都不能进入阶段2，此时没有任何版本的协议被批准。

✦结果：没有任何决议被批准。

✦分析：前面的序列如果变化?

✦应对措施：此时系统需要能够选出某个提议者作为唯一的提议者来发出提议。在Zookeeper中正是使用的这种策略（Leader Election）。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况5**： A提议爬长城时间定为下周三，同时E提议爬长城时间定为下周五，并且A、E在不同时刻发起决议（即版本号不同）。由于某种原因D失去了联系，A、E均能与B、C通信，但A、B不能通信。

✦阶段1：A向所有人发出决议“下周三：2012年3月2日，12:00”，半小时候E向所有人发出决议“下周五：2012年3月2日，12:30”。假如B、C按情况4所诉的顺序处理来自A、E的决议，那么我们来分析一下最终结果。（待续）

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况5**:

✦阶段1：（续）（1）B接收到来自A的决议，那么B批准A的决议，并回复A；（2）C接收到来自E的决议，那么C批准E的决议，并回复E；（3）B接收到来自E的决议，并发现该决议的版本号比批准的A决议的版本号高，故回复E的决议；（4） C接收到来自A的决议，并发现该决议的版本号比批准的E决议的版本号低，故拒绝A的决议，并回复A已经有更高版本号的决议被提出。

数据一致性理论

Paxos算法——实例

A、B、C、D、E五个人使用Paxos算法约定时间去爬长城。

✦**情况5**:

✦阶段2：最终A将只收到来自B的回复，不能构成大多数集；而E最终收到B、C的回复，从而构成大多数集，这样E将最终进入到“批准请求”阶段，最终E的决议被批准。

✦结果：E的决议被批准，爬山时间定为下周五。