ACID--CAP--BASE理论

BASE：basically available基本可用, soft state软状态, eventually consistent最终一致性。

最终一致性5个变种：causal consistency因果关系，

Read your writes 读己之所写

Session consistency会话一致性

Monotonic read consistency单调读一致性

Monotonic write consistency单调写一致性

最终一致性在关系型数据库中体现：主从复制，同步时是强一致性，，异步时存在延迟，即最终一致性。

二阶段提交协议

优点：原理简单，容易实现

缺点：同步阻塞，单点问题，脑裂（数据不一致），太过保守的容错机制

三阶段提交协议：CanCommit, PreCommit, Do Commit

Paxos算法

 数据的一致性协议有：

        1.  两阶段提交协议  2PC

        2.  三阶段提交协议  3PC

        3.  RWN协议

        4.  raft协议

        5.  Paxos协议

分布式系统的两个重要协议：

1. Paxos选举协议：Paxos 协议用于多个节点之间达成一致，往往用于实现总控节点选举。关于Paxos可以参考我的另外文章[从分布式数据复制一致性问题到paxos算法的理解（第一部分）](http://kaimingwan.com/post/fen-bu-shi/cong-fen-bu-shi-shu-ju-fu-zhi-zhi-xing-wen-ti-dao-paxossuan-fa-de-li-jie-di-bu-fen) <http://kaimingwan.com/post/fen-bu-shi/cong-fen-bu-shi-shu-ju-fu-zhi-zhi-xing-wen-ti-dao-paxossuan-fa-de-li-jie-di-bu-fen> 和 [从分布式数据复制一致性问题到paxos算法的理解（第二部分）](http://kaimingwan.com/post/fen-bu-shi/cong-fen-bu-shi-shu-ju-fu-zhi-zhi-xing-wen-ti-dao-paxossuan-fa-de-li-jie-di-er-bu-fen) <http://kaimingwan.com/post/fen-bu-shi/cong-fen-bu-shi-shu-ju-fu-zhi-zhi-xing-wen-ti-dao-paxossuan-fa-de-li-jie-di-er-bu-fen>

2. 两阶段提交协议(2PC): 保证跨多个节点操作的原子性，这些操作要么全部成功，要么全部失败。

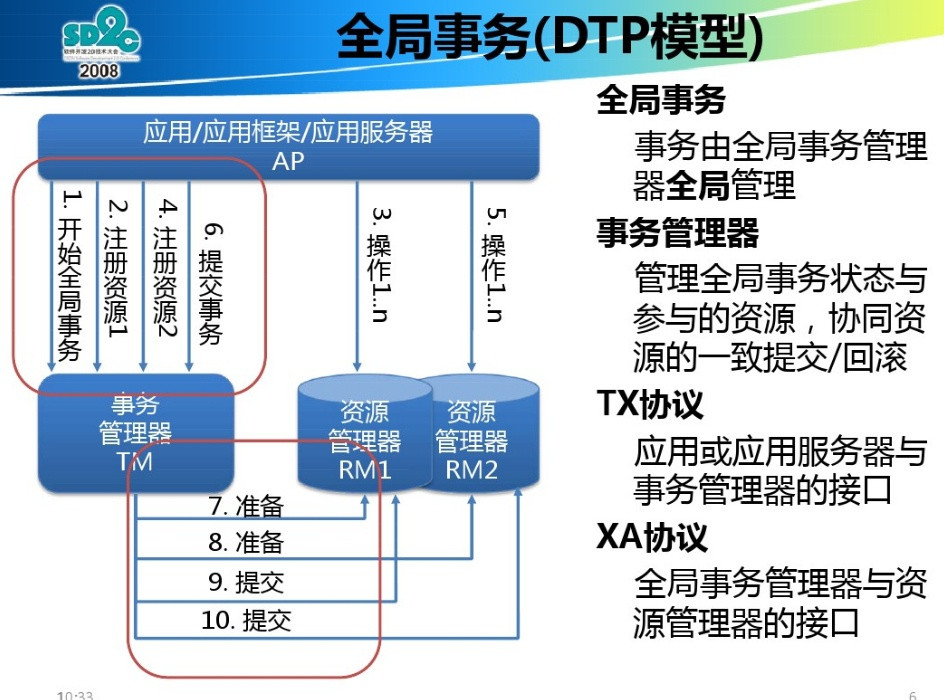
# 5. DTP事务模型

X/Open Distributed Transaction Processing (DTP) 事务模型实现了XA接口。XA事务指的是实现了XA接口符合DTP事务模型的事务。  
DTP事务模型是X/Open 这个组织定义的一套分布式事务的标准，也就是了定义了规范和API接口，由这个厂商进行具体的实现。  
DTP事务模型主要用于分布式事务

XA接口规范定义了以下几个组件：

1. 应用程序(AP):也就是应用程序，可以理解为使用DTP的程序 2.资源管理器(RM):资源管理器，这里可以理解为一个DBMS系统，或者消息服务器管理系统，应用程序通过资源管理器对资源进行控制。资源必须实现XA定义的接口 3.事务管理器(TM):事务管理器，负责协调和管理事务，提供给AP应用程序编程接口以及管理资源管理器

AP、RM、TM三者直接互相的协作关系用下图表示：



可以看到，其中TM 和 RM 通过XA接口进行双向通信。AP和TM、RM的通信接口规范没有约定，可以自己实现。

DTP定义的事务概念：

1. 事务：一个事务是一个完整的工作单元，由多个独立的计算任务组成，这多个任务在逻辑上是原子的。
2. 全局事务：对于一次性操作多个资源管理器的事务，就是全局事务
3. 分支事务：在全局事务中，某一个资源管理器有自己独立的任务，这些任务的集合作为这个资源管理器的分支任务

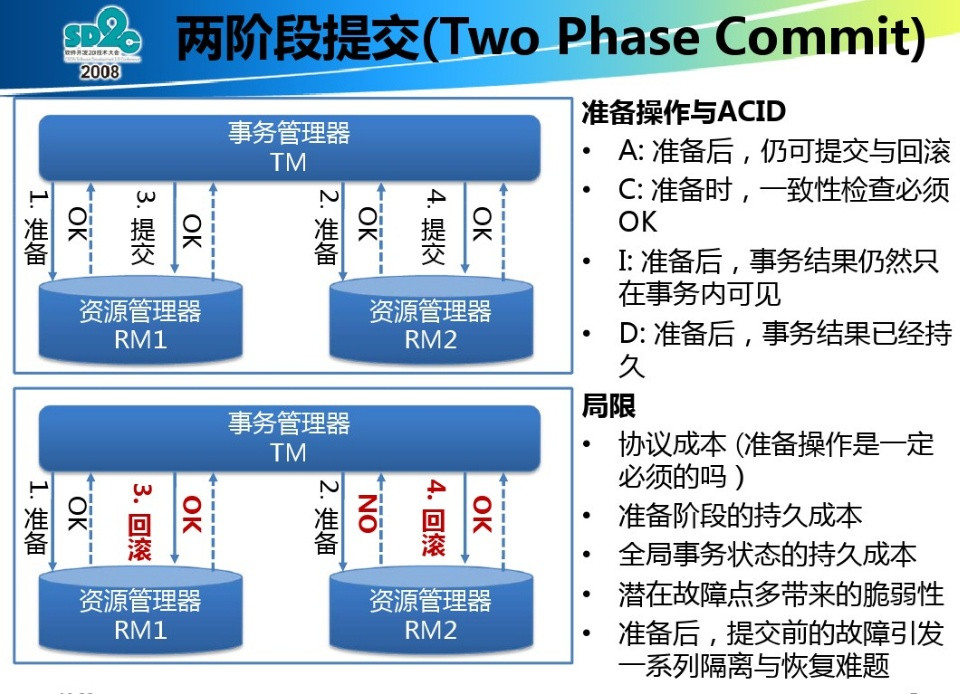
控制线程：用来表示一个工作线程，主要是关联AP,TM,RM三者的一个线程，也就是事务上下文环境。简单的说，就是需要标识一个全局事务以及分支事务的关系。

很多数据库都实现了DTP事务模型，用于支持分布式事务。

## 5.1 两阶段提交(2PC)

两阶段提交是DTP事务模型的一种实现方式。一个TM控制多个RM，协调资源。

两阶段主要指的是：  
第一阶段（准备阶段）：事务管理器通知资源管理器准备分支事务，资源管理器告之事务管理器准备结果  
第二阶段(提交阶段)：事务管理器通知资源管理器提交分支事务，资源管理器告之事务管理器结果

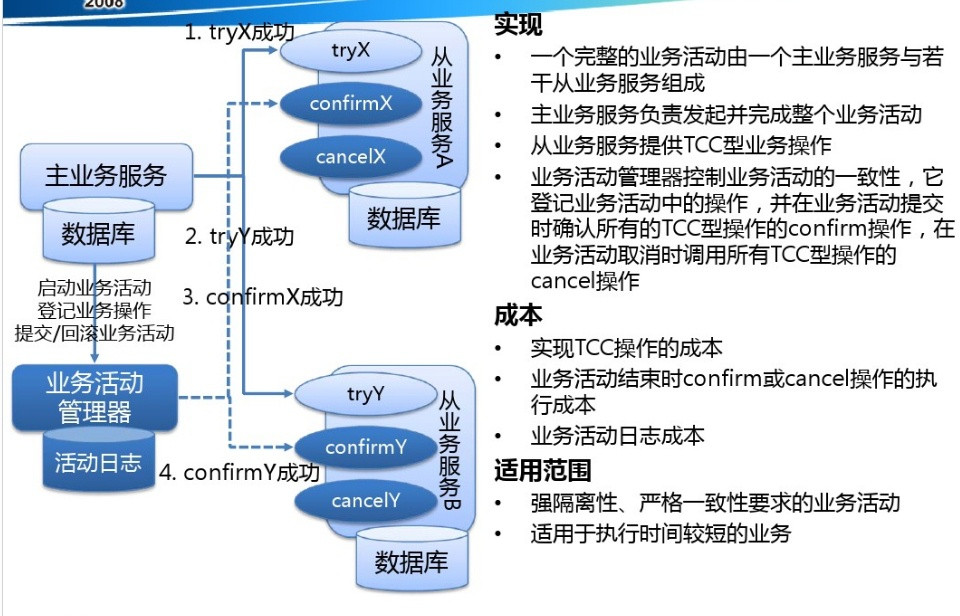


## 5.2 TCC

TCC是一种基于事务补偿（属于事后控制的策略，进行回滚）的模式，注意区别2PC

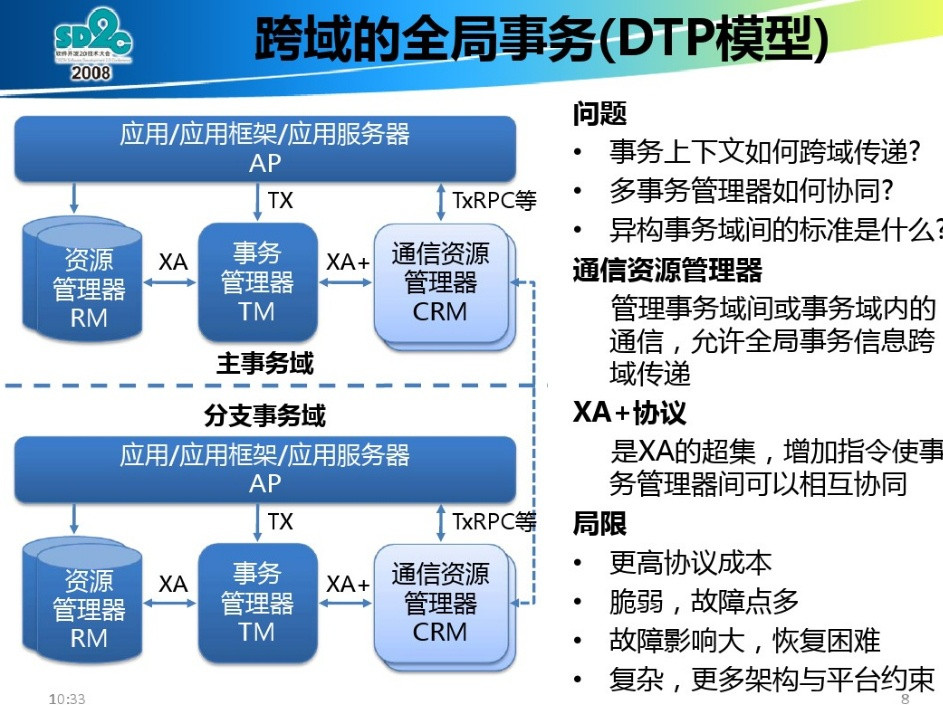


TCC的执行过程



# 6 跨域的DTP模型

需要RPC调用，有一些额外的问题。需要引入通信资源管理器。具体看下图：



集群伸缩性：分为应用服务器集群伸缩性和数据服务器集群伸缩性。

而数据服务器集群伸缩性又包含缓存数据服务器集群和存储数据服务器集群。

应用服务器集群的伸缩性设计：

事实上，大型网站总是部分使用DNS域名解析，利用域名解析作为第一级负载均衡手段，即域名解析得到的一组服务器并不是实际提供Web服务的物理服务器，而是同样提供负载均衡服务的内部服务器，这组内部负载均衡服务器再进行负载均衡，将请求分发到真实的web服务器上。

**负载均衡技术：**

1. HTTP重定向负载均衡：现实中并不多见，利用了一个专门的重定向服务器

优点：简单；

缺点：两次访问，性能差，重定向服务器自身的处理能力存在瓶颈。

1. DNS域名解析负载均衡：利用了DNS服务器

优点：工作移交给DNS，省了网站方管理重定向服务器的麻烦；支持基于地理位置的请求分发，DNS服务器会返回一个距离用户最近的IP地址给用户，加快用户访问速度，改善性能；

缺点：网站方不能做更强管理的控制，DNS多级解析给下线服务器更改带来延迟。并且DNS负载均衡的控制权在域名服务商那里，网站无法做更多改善和管理。

1. 反向代理负载均衡：

优点：与反向代理服务器功能集成在一起，部署简单

缺点：反向代理服务器是所有请求和响应的中转站，存在性能瓶颈

1. IP负载均衡：在网络层通过修改请求目标地址进行负载均衡，请求到专门的负载均衡服务器，服务器在操作系统内核进程获取网络数据包，根据负载均衡算法计算得到一台真实的web服务器ip地址，然后修改数据包目的ip地址，不需要通过用户进程处理。得到响应后，服务器再将数据包 源地址改回原来的ip地址。这个过程需要进行地址转换（真实处理的web服务器要发送数据给负载均衡服务器）

优点：在内核中完成分发，性能好

缺点：请求和响应都经过它，存在瓶颈，例如网卡宽带不好。

1. 数据链路层负载均衡：又叫做直接路由方式（DR）；指在通信协议的数据链路层修改MAC地址进行负载均衡。三角传输模式，负载均衡服务器只修改MAC地址，不修改IP地址，因此真实的WEB服务器处理完后可以直接发送响应到用户浏览器，避免负载均衡服务器网卡带宽成为瓶颈。

优点：

缺点：

**使用三角传输模式的链路层负载均衡是目前大型网站使用最广的一种负载均衡手段。在Linux平台上最好的链路层负载均衡开源产品是LVS（Linux Virtual Server）。**

**负载均衡算法：**

1. 轮询 Round Robin, RR
2. 加权轮询 Weighted Round Robin, WRR：根据服务器硬件性能加权
3. 随机 Random
4. 最少连接 Least Connections：记录每个服务器连接数（请求数），新请求连到最少连接的机器上。
5. 源地址散列 Source Hashing：根据请求来源IP地址进行Hash计算，得到应用服务器，这样来自同一个IP地址的请求总在用同一个服务器上处理，请求上下文信息可以存储在这台机器上，在一个会话周期内重复使用，从而实现会话粘滞。