5.2 两阶段提交协议

两阶段提交协议是分布式系统中广泛采用的请求原子性保证协议。本节介绍其内容，以作为 BWMMS 协议的对比参照。两阶段提交协议主体包括 1 个提交协同者(Coordinator, 图中简写为 C)和若干个提交参与者(Participant,图中简写为 P)，协议过程如图 5.1 所示：

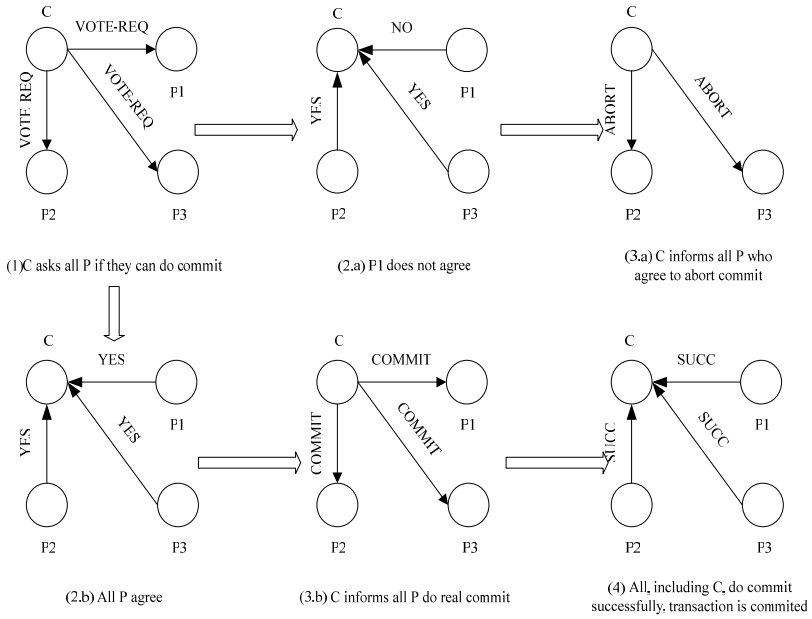

图 5.1 两阶段提交协议

如图 5.1 所示，两阶段提交协议的过程为：

1. 图 5.1（1），提交协同者向所有参与者发出 VOTE-REQ 消息，要求所有参与者投票此次更新结果是否提交。
2. 提交参与者收到 VOTE-REQ 消息后，它向提交协同者返回自己的投票结果，“YES”或“NO”（图 5.1（2.a））。“NO”表示本参与者决定放弃本次提交操作。
3. 提交协同者收集所有参与者的投票结果。如果所有参与者和自己的投票结果都是“YES”（图 5.1（2.b）），它将向所有参与者发送 COMMIT 消息（图 5.1（3.b））。否则，提交协同者做出“放弃本次提交”的决定，并向所有投出“YES”结果的参与者发送 ABORT 消息（图 5.1（3.a）），结束本次提交协议。
4. 所有投出“YES”结果的参与者等待协同者发来的“COMMIT”或者“ABORT”消息。在收到提交协同者的消息后，它根据收到的消息进行相应的处理（图 5.1（4）），然后结束本次提交协议。

从图 5.1 的两阶段提交协议过程可知，将“VOTE-REQ”和“YES/NO”，“COMMIT”和“SUCC/FAIL”分别当作 1 对消息，在涉及到 N 个（N>=2）服务器的操作中，成功提交需要 2(N-1)对消息和 N 次更新设备。不能成功提交时，其最小开销是(N-1)个都投否决票时的(N-1)对消息，最大开销是仅有 1 个提交参与者投否决票，需要(2N-3)对消息。同时，分布式日志技术要求日志内容的同步更新。

5.3 BWMMS 的跨服务器请求原子性保证协议

两阶段提交协议需要相对较多的消息通信和设备更新，还需要能够保证持久存储的

分布式日志技术提供支持。涉及到 4 个服务器(N=4)的文件移动操作，在结果成功执行的情况下，它需要 6 对消息通信和 4 次更新设备的操作。在不能成功提交时，其消息对数目范围是[3，5]。在系统规模非常大的情况下，两阶段提交协议的开销比较大。

为减少请求需要的消息数量和设备更新次数，简化系统的错误恢复过程，BWMMS 利用集中共享存储架构和对称服务器结构提供的灵活性，通过改变元数据宿主，完成元数据在服务器间的迁移，将多服务器协同处理转变成单个服务器集中处理，并借助本地文件系统技术提供请求事务性保证。

5.3.1 目标跨服务器请求

根据第 3.1 节的元数据访问协议，在存储服务与请求服务分离后，需要保证其原子性的跨服务器请求主要包括文件创建、删除和移动等。

文件创建请求向文件系统名字空间添加新的文件。它首先创建一个代表新文件的索引节点，然后在父目录中添加指向该文件的目录项。它要求新创建的文件索引节点先于父目录数据块写回设备，防止故障带来错误的索引节点引用。

文件移动请求将文件（旧文件）从一个目录移动（源目录）到另一个目录（目标目录），同时可能将文件重命名为新的文件名字（新文件）。它需要在目标目录中添加目录项指向新文件，然后从源目录中删除指向旧文件的目录项。如果目标目录中原来存在与新文件名字相同的文件(同名文件),还需要更改目标目录中原来指向同名文件的目录项，修改同名文件的索引节点。文件移动操作最多将更改 4 个元数据，是文件系统名字空间最复杂的元数据请求。

文件删除请求从父目录删除对文件的引用。删除完成后，不能再通过父目录引用文件的索引节点。它要求父目录数据块先于文件索引节点更新到存储设备。

5.3.2 元数据迁移的可行性

如前面所述，BWMMS 的集中共享虚拟存储架构、全共享的存储资源使用方式、文件系统服务的模块化和用户访问统计特征等,使得 BWMMS 通过元数据迁移保证文件系统一致性成为可能。具体包括：

1. BWMMS 基于集中共享虚拟存储模型，系统所有的元数据服务器共享线性逻辑元数据资源空间。对称的服务器结构保证元数据服务器访问同一个逻辑元数据资源的成本相同，元数据请求可以由任意元数据服务器处理，不会有明显的性能影响。
2. 在获得元数据的宿主权限后，每个元数据服务器都可以利用元数据，进行任意的元数据请求，元数据在服务器间的迁移不会影响系统的功能。
3. 分布式文件系统元数据的存储服务和请求服务分离。存储资源可以在任何服务器上释放给存储服务，不会引起存储资源管理的异常。

4. 已有研究表明，可能成为跨服务器请求的文件创建、删除和移动请求所占比例不大，不超过 5%，如表 5.1 所示[Gibson1997]。在元数据请求分布策略决策下，跨服务器请求的比例将更小[Hendricks2006]。已有的两阶段提交协议等传统的请求原子性保证协议，对系统的性能和错误恢复能力的影响极大，有必要探讨轻量级协议的可能性。

表 5.1 跨服务器请求统计特征

NFS			
Operation	Description	Percent	Quantity(*10^6)
DirReadWrite	Creation of files and directories, file renaming, links, etc.	0.7	0.21
DeleteWrite	Deletion of files and directories	0.1	0.04
AFS			
CreateFile	Create a new file in the AFS server namespace	2.1	2.0
Rename	Move a file in the AFS server namespace from one location to another location	1.9	1.7
RemoveFile	Delete a file stored on AFS server	1.5	1.4
Others	Operations includes ACL manipulations, symlinks, directory create and deletion,lock management, volume management,etc.	3.4	3.2

5.3.3 元数据迁移对象

元数据宿主改变首先需要明确迁移的对象和迁移的粒度。目录子树分区法和哈希法的迁移对象是持久存储的元数据和动态的文件锁信息等。它以构成文件系统名字空间目录子树的一组文件作为迁移对象，需要将整个目录子树迁移。相对目录子树分区法更甚的是，哈希法需要迁移的元数据范围不可控，文件系统名字空间的结构将迁移范围放大，直至目录树的叶结点。

BWMMS 的元数据迁移对象是单个索引节点及其关联的元数据块，仅迁移需要持久存储的文件系统元数据, 不迁移文件锁信息等动态的元数据。如果在迁移时元数据存在关联的动态元数据信息，它首先阻止新的动态元数据信息的产生，同时根据迁移策略进行判断，等待或者要求释放已有的动态元数据信息。当没有动态元数据信息存在时，元数据迁移才进行。

5.3.4 迁移协议数据格式

MS 在元数据迁移的数据格式中包括元数据迁移原因，以完成元数据迁移请求与其他元数据请求的同步。元数据请求的同步控制将在下一章讨论。BS 将元数据请求迁移内容转发给被迁移元数据当前的宿主 MS。元数据迁移的请求参数格式如表 5.2 所示。

表 5.2 元数据迁移协议格式

```
enum migrate_reason {
        /*硬连接的目标文件*/
         MGR_LINK = 0,
        /*符号连接的目标文件*/
```

```
        MGR_UNLINK,
        /*移动文件的目标父目录*/
        MGR_RENAME_TDIR,
        /*被移动文件*/
        MGR_RENAME_OINO,
        /*移动文件的目标文件*/
        MGR_RENAME_NINO
};
typedef enum migrate_reason migrate_reason;
struct migrate_request {
        eino_t ino;
        u32 generation;
        u32 version;
        migrate_reason reason;
};

enum migrate_result {
        MGP_OK = 0,
        MGP_DELETED,
        MGP_NOTEMPTY,
        MGP_BUSY,
        MGP_NOTSUPP
};
typedef enum migrate_result migrate_result;
struct migrate_reply {
        eino_t ino;
        u32 generation;
        u32 version;
        migrate_reason reason;
        migrate_result result;
};
```

5.3.5 迁移协议时序图

元数据迁移包括文件索引节点内容的迁移和元数据分布信息的更改。元数据迁移完成元数据从源宿主移动到目标宿主的工作，元数据分布信息改变完成相关服务器记录的元数据当前分布信息的更改。元数据迁移过程必须正确更改元数据的分布信息，避免系统出现“同一个活跃元数据具有多个宿主”的情形。

为降低元数据迁移协议的复杂度，避免复杂的错误恢复协议，元数据迁移请求通过 BS 转发、元数据内容通过 SN 中转。源 MS 将元数据内容写回到 SN，目标 MS 从 SN 读回元数据的内容。在迁移过程中，源 MS、BS 和目标 MS 根据自己的处理结果，更改元数据分布信息。元数据迁移协议过程如图 5.2 所示，假定被迁移元数据当前分布在 MS2。

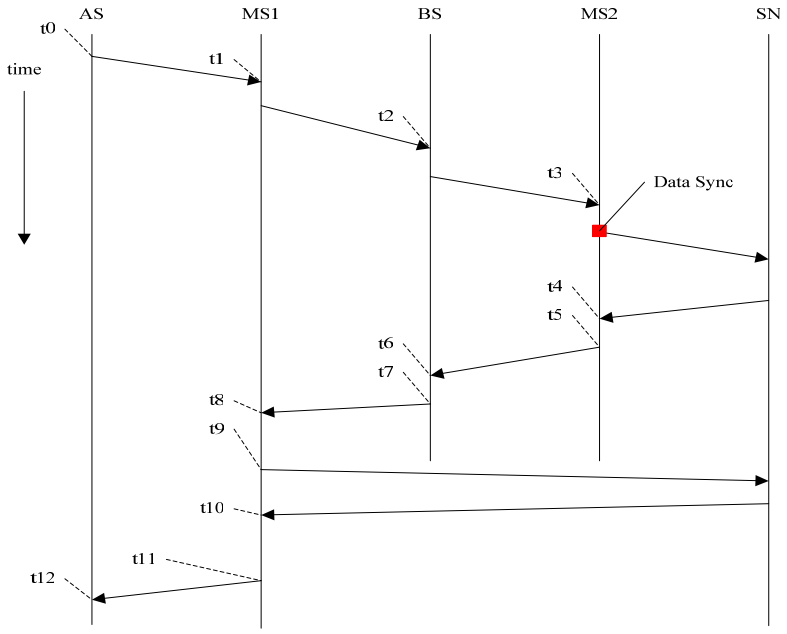

图 5.2 元数据迁移协议时序图

元数据迁移协议过程描述如下：

1. AS 向 MS1 发出硬连接、删除或者移动文件可能的分布式元数据请求;
2. MS1 通过自己的元数据分布信息缓存查找元数据分布信息，获得结果 (foo，not-on-me)。MS1 向 BS 请求获得该元数据的宿主权限；
3. BS 检查 foo 当前的宿主服务器信息，获得（foo, MS2）。BS 以 MS1 的名义要求 MS2 释放索引节点 foo 的宿主权限；
4. MS2 在能够完成权限释放的情况下，将 foo 对应的文件索引节点和元数据块更新到存储设备 SN。foo 的元数据更新完成后，MS2 更改自己维护的元数据分布记录（foo, MS2）→ (foo, MS1)，并返回结果 BS；
5. BS 更改元数据分布记录（foo, MS2）→ (foo, MS1)，赋给 MS1 宿主权限。返回结果给 MS1；
6. MS1 更改元数据分布记录（foo, not-on-me）→ (foo, onme)。然后从 SN 读取索引节点 foo。最后，MS1 按照请求的语义，完成元数据请求的处理。返回结果给 AS。

5.3.6 迁移协议的影响分析

BWMMS 的元数据迁移协议对系统的影响包括两个方面：

1）对元数据请求分布的逻辑性的影响。元数据分布策略的逻辑性考虑，将目录 bar 下的活跃文件 foo1，foo2，……，尽量与目录分布在一起，所有需要同时访问 bar 和文件的请求，如 lookup 等，将由单个元数据服务器完成。当目录 bar 从原来的服务器 MS1 迁移到新的服务器 MS2 后，可以由单个服务器完成处理的 lookup 等请求，将变成需要 MS2 和 MS1 协同才能处理。

2）对元数据服务器负载公平性的影响。在进行元数据请求分布决策时，各个元数据服务器的负载作为决策参数进行考虑。在迁移发生前，各个服务器的负载相对平衡。迁移发生后，所有原来由 MS1 处理的目录关联的元数据请求，将转移到 MS2 处理。这导致 MS1 的负载降低，而 MS2 的负载将升高，导致服务器间的负载不平衡。