

LECTURE FRANGIPANI

1. 背景知识

- Petal: 为上层提供虚拟磁盘服务
 - 可以通过复制提升可用性（可选，通常为2个），使用Chain Replication实现
 - 寻址空间为 2^{64} Byte
 - 支持快照
 - 不支持协调操作，不支持多客户端共享存储
 - 对上提供的接口类似磁盘的硬件接口，而非文件系统接口
- 一致性语义：
 - Andrew File System (AFS)会话语义
 - 写操作，对于打开同一文件的其他用户不可见
 - 一旦文件关闭，写操作仅对后来打开的会话可见，已打开的文件实例不反映变化
 - Unix file system (UFS)
 - 写操作对其他打开同一文件的用户是立即可见的
 - 允许多用户在读写时共享文件指针
 - 不可变共享文件语义
 - 一旦一个文件由创建者声明为共享，他就不能被修改。
 - 不可变文件：名称不可重用、内容不可改变
- 元数据的定义：Frangipani将元数据定义为除了文件内容本身之外的 所有的 在盘的 数据结构
- 粘性（sticky）锁：一个server持有的锁只有在其他server发起持有请求时才会被释放
- 崩溃一致性（Crash Consistency）

2. 概述

- Frangipani是一个分布式**文件系统**，由两层组成：
 - 下层为Petal：用于提供可拓展、高可用、自动管理的虚拟磁盘
 - 上层：多台机器，运行同样的Frangipani文件系统在共享的Petal之上，使用锁服务、WAL（write - ahead log）保证一致性。

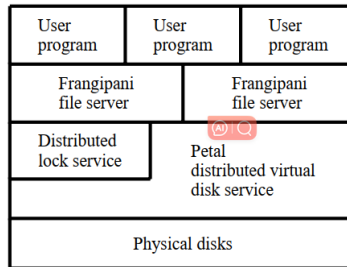


Figure 1: Frangipani layering. Several interchangeable Frangipani servers provide access to one set of files on one Petal virtual disk.

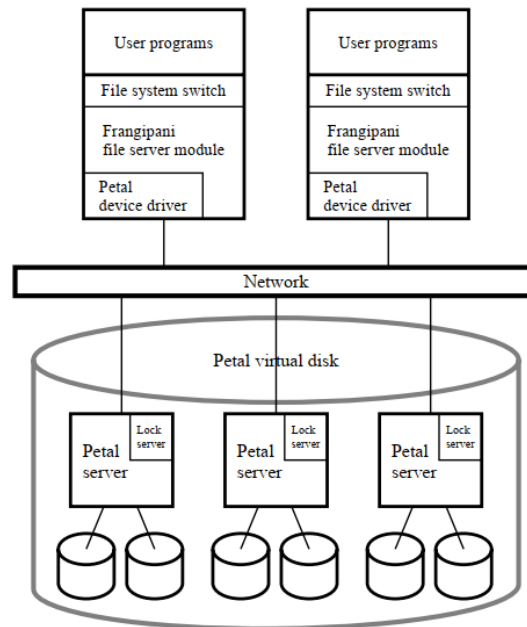


Figure 2: Frangipani structure. In one typical Frangipani configuration, some machines run user programs and the Frangipani file server module; others run Petal and the distributed lock service. In other configurations, the same machines may play both roles.

总体结构如上架构图所示，两图需要结合起来看。图二中左上方和右上方分别在一台机器中。图二中锁服务可以独立于Petal、Frangipani而单独存在，也可以集成到Frangipani中，实现的锁机制为读者写者锁。而Petal服务由多个Petal服务器组成，多个Petal服务器共同为上层提供一个虚拟磁盘。

文件系统可以通过增加Frangipani file server来实现横向拓展。Frangipani还具有负载均衡的功能。其运行在操作系统内核之中。Frangipani file server相互之间也不需要通信

- Frangipani使用场景：小范围相互信任的计算机（Frangipani几乎没有安全机制）。
- 文件相关的操作必须由Frangipani进行，Petal不应该参与。这个要求导致了Frangipani需要额外的CPU和SDRAM开销。
- Frangipani的一些特性：

- 对于同一文件，所有用户看到的视图都是一致的
- 可以增加机器来拓展Frangipani的吞吐量和容量，这一过程不会：（1）影响现存其他服务器的配置信息 （2）中断其他服务器执行
- 管理员可以随意添加上层的用户，而不用考虑这些用户和磁盘的对应关系
- 管理员可以对整个文件系统进行完整且一致的备份（backup），这一过程不需要关闭文件系统
- Frangipani可以容错并恢复，而不需要人工干预
- Frangipani的缺陷：
 - 日志可能会被写两次（Frangipani自身一次，Pental一次）
 - 在放置文件时不能使用磁盘位置信息（因为Petal是虚拟磁盘）
 - 锁粒度过大，只能封锁整个文件，而不能只封锁一个块
- Frangipani

3. 系统结构

3.1. 组成部分

- 共享文件的一致性语义为：UFS
- 应用程序使用标准的操作系统调用接口来使用Frangipani
- 本地Unix文件系统语义（?????）：
 - 写数据时，数据会被保存在操作系统内核的缓冲区中
 - 只有在调用了fsync或sync之后，才能保证缓冲区中对应的数据写入磁盘（即调用fsync/sync之后才能保证持久化）
 - log（存储元数据变化）一定会被即时持久化。**每个Frangipani file server 都有自己的不同的 log（在盘上）**
 - **Frangipani在此基础上的改动：**元数据会包括文件的最近访问时间，Frangipani维护的是一个近似的时间，用于减少log持久化带来的开销

3.2. 安全性问题

- 安全性担忧：
 - Frangipani所有的机器都必须有可靠的操作系统
 - Lock Server、Petal Server和Frangipani File Server在相互通信时必须验证身份
 - Frangipani File Server在向Petal Server验证身份时，不能以OS用户的身份进行
 - 要防止某些用户在网络上窃听
- 解决方案：
 - 阻止普通用户在某机器上引导新的操作系统内核
 - 私有网络
- 管理域之外的访问：概述中两张图的所有机器必须都在管理域之内。若管理域之外的机器需要访问文件，则需要先链接有Frangipani File Server的机器，而后再由Frangipani File Server访问文件，而不能直接访问。如下图：

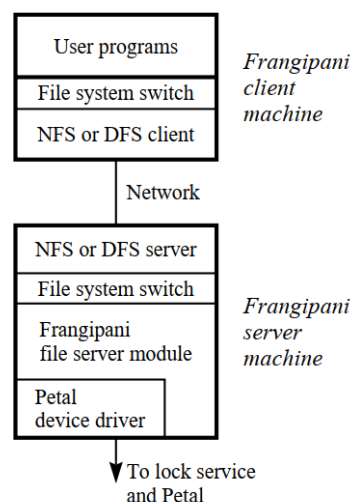


Figure 3: Client/server configuration. A Frangipani server can provide file access not only for the local machine, but also for remote client machines that connect via standard network file system protocols.

4. 磁盘布局

- 在Petal上，仅当虚拟地址被写入数据时，才会将物理磁盘空间提交给虚拟地址；同时，Petal也有一个decommit原语，可以释放虚拟地址对应的物理磁盘空间
- Petal一个块为 $64KB$ 。用更大的块大小是为了减少元数据的大小。每个块的地址范围： $[a \cdot 2^{16}, (a + 1) \cdot 2^{16})$ ，因此应用程序存储数据应当注意不能使用过于稀疏的存储空间，以避

免过多内碎片。

- 磁盘布局：

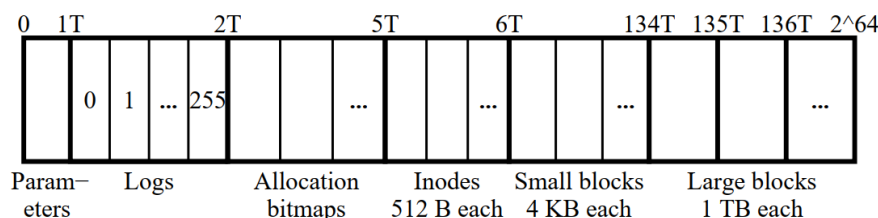


Figure 4: Disk layout. Frangipani takes advantage of Petal's large, sparse disk address space to simplify its data structures. Each server has its own log and its own blocks of allocation bitmap space.

- 第一区：存储共享配置变量和内务信息（housekeeping information），最大 $1TB$ ，通常只会用到若干 KB
 - 第二区：日志区。这一区域被划分为了256个日志。日志区总大小为 $1TB$ ，日志的数目是可以调整的。注意前文提到log是每个Frangipani File Server私有的，因此划分的目的就是为了给每个Frangipani File Server提供日志，在划分成256个日志的情况下，也就意味着最多支持256个Frangipani File Server。
 - 第三区：存储**位示图（见os文件系统）**，位示图存储了哪些块是空闲的。这一部分中，每一台Frangipani File Server拥有独占使用的一些块，这个块由这台Frangipani File Server上锁。当一台Frangipani File Server发现自己当前占有的块中（是位示图对应的块），所有的块都已经被使用了，则会在第三区再找一块未上锁的块，给它上锁，独占使用。第三区 $3TB$ 大小
 - 第四区：inode区。详情见os文件系统。每个文件都有一个inode，存储了文件的元数据（时间戳、文件位置指针等）。符号链接会直接把指向的文件位置存在inode中（对于小文件都是这么处理的，为了充分利用过大的inode大小）。inode区为 $1TB$ 大小
 - 每个inode大小为 $512B$ ，这一大小与磁盘的块大小相同。这一设计能够避免竞争。当两个服务器想要获取位于同一个块内的inode时就可能发生这种竞争。
 - 第五区：小数据块区。一个小数据块为 $4KB$ ，整个小数据块区大小为 $128TB$ 。当文件小于等于 $64KB$ （也就是16个小数据块）时，会被存放在小数据区中，若超过了则超出部分会放在大数据区中。
 - 第六区：大数据块区。一个大数据块为 $1TB$ ，整个大数据块区大小为 $2^{64} - 134TB$ 大小
- 由于文件系统的块大小过大，因此可能会比块小的文件系统有更多的内碎片。

- inode结点太大，可能会造成浪费。因此对于很小的文件，可以直接存在inode之中（例如符号链接/软连接）
- Frangipani的人为限制：
 - 大文件最多不超过 2^{24} 个，所谓的大文件是指大于 $64KB$ 的文件
 - 一个文件最大为 $64KB + 1TB$

5. 日志与恢复

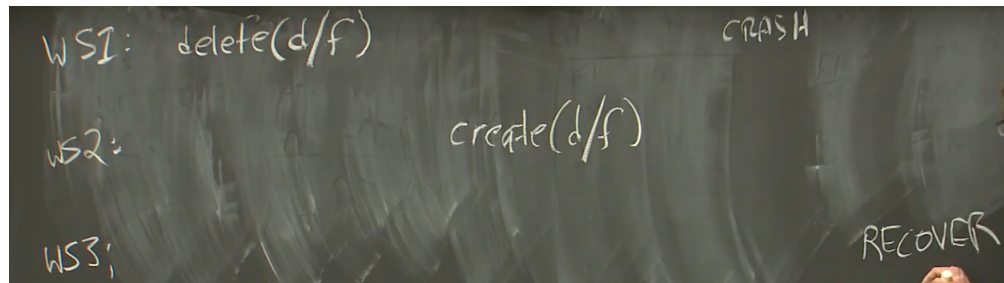
- log存的是元数据，而不是用户数据。在发生故障后，用户数据并不能保证一致！（从用户的角度来看），**所谓的元数据是指：目录、inode与位示图**
- 修改元数据流程：
 1. 先创建一条日志记录，描述元数据的变化
 2. 将这条日志记录插入自己的日志中
 3. 日志被定期写入Petal
 4. 修改实际的元数据
 5. 实际元数据会被系统定期写回
- 日志大小上限为 $128KB$ （究竟是日志大小还是日志条目？），一个日志会被分成2个 $64KB$ 大小的段，分别存在两个物理硬盘上。分配给每个日志条目的空间使用环形缓冲区的方式进行组织。当缓冲区满了，Frangipani就会把最旧的25%释放。回收了的缓冲区中会存在一些日志条目，通常情况下是已经被写回磁盘了，若没有写回磁盘，则会先写回磁盘然后再进行回收。
- 日志条目大小约为 $80 \sim 128 Bytes$ ，因此一个日志最多有 $1000 - 1600$ 条日志条目。在两次sync/fsync操作之间，日志可能会被填满。
- Frangipani File Server故障的发现：
 - 某Server的client发现server故障
 - 锁服务要求Server释放锁时未得到Server的回应
- 故障恢复的过程：
 1. **恢复守护程序（recovery demon）** 获得崩溃的Frangipani File Server的日志和锁

2. 从日志的开头开始按顺序检查

3. 完成未完成的日志条目

4. 释放锁，清空日志

- 上述过程不影响其他server的执行
- 崩溃的server可以选择是否继续执行（当然，重启时，其log是被清空了的）
- 在Petal正常工作的情况下，Frangipani可以容忍任意Frangipani File Server的故障。
- 如何寻找日志的结尾：在日志区（第二区），块大小为512Byte，Frangipani为第二区的每个块赋一个单调递增的序列号。当发现某个块的序列号比前一个块的序列号小时，则说明到达log末尾。
- 日志更新的相关要求：
 - 不同Frangipani File Server对同一数据的写入必须是串行化的。这点是通过锁机制实现的，锁机制要求：只有在脏数据被写回之后，锁才能易主。写回操作既可以通过锁的持有者进行的，也可能是通过**恢复守护程序**进行的。这样就保证了同一时间，最多只有一个server的log中包含了没写回的操作。（如果允许多个server的log中包含没写回的操作，则这两个server故障之后，这些操作的顺序是非确定的了）
 - 恢复守护程序只会应用满足下面条件的日志条目：这些日志条目必须是故障的server加锁之后加入自己的日志的。注意守护程序会仍然拥有这些日志的锁。上面的条件是通过满足其充分条件来间接满足的：恢复守护程序从不会重演那些已经完成了的更新。
 - 为了达到这一点，每一个元数据块都有一个单调递增的编号。当修改元数据时，log中会附上元数据修改前后的变化以及新的版本号。
 - 在恢复时，恢复守护程序会比对元数据块的版本号，若其小于log中的版本号，则说明需要更新，反之则不需要更新。
 - 注意，一个元数据块如果在后续分配给了用户数据，则在进行恢复时，读到对应的log条目时可能会出现问题。因此Frangipani采用了一个非常简单粗暴的做法：回收的元数据块只能再次用于元数据
 - 一个例子说明为什么需要版本号：



在这个例子中，WS1先调用了delete，在delete成功删除文件f之后，崩溃。随后WS2又通过create调用创建了同名文件f，由于WS1崩溃，锁服务器令WS3重演WS1的日志，而后又重演了delete命令，把WS2创建的f删除。在有版本号的情况下，WS3启动的恢复守护程序，会比对日志中的版本号和实际数据的版本号，而后发现实际数据版本号更新，则不会进行重演。

- 同一时间只能有一个恢复守护程序对相应的server进行恢复。这一点是通过锁服务器为恢复守护程序上锁来实现的
- Frangipani在进行恢复时，不能直接恢复！必须要比对版本号！
- 虚拟磁盘中，一个扇区有两个备份。Frangipani会保证这两个扇区状态是一致的。如果一个扇区发生了冗余校验错误，则Petal可以自动恢复它。
- *log的作用：并非用于提供高层次的语义，而是为了加快更新元数据的速度、加快恢复的速度（否则就需要unix系统提供的fsck）。
- 当只有一个Frangipani File Server写数据，没有其他server读数据时，写数据的server拥有惟一的修改的数据。若其崩溃则修改数据会丢失。为了减少这一情况发生的损失，Frangipani会每隔30s刷盘
- 关于Recover时锁的问题：
 - 若恢复时，原始Frangipani File Server已经释放了锁，则恢复守护程序在比对版本号之后会发现当前数据的版本号大于等于日志版本号，因此不会重演，避免二次执行。
 - 若恢复时，原始Frangipani File Server还未释放锁，则进行恢复的Frangipani File Server获取原始Server的锁

6. 同步与cache一致性

- 读写锁：多读者，单写者。当Frangipani File Server A 拥有了某个块的锁时，此时Server B 也发出了要占有锁的请求，则A会被要求释放锁或者降级锁（将写锁变成读锁）。
 - **没有锁就不能有cache** 在释放读锁时，必须要求拥有读锁的Frangipani File Server 将自己的cache条目标记为无效。（释放写锁则需要刷盘）

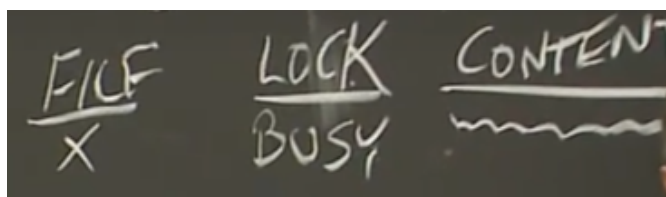
- 拥有写锁时，允许读操作和写操作。仅当拥有写锁时，允许Frangipani File Server的cache和磁盘不一致。因此，当写锁拥有者被要求释放或降级时，必须将脏数据写回；当写锁拥有者被要求释放时，必须也要讲cache条目标记为无效，当被要求降级时，则可以保留cache条目（此时变为读锁了）
- 文件系统的盘上结构将磁盘划分为若干段（segment），这些段可以整体上锁。为了保证正确共享数据，必须要求磁盘上的同一扇区只能归属于同一个区（见前文）。
 - 一个日志是一个单一的段
 - 位示图被分为若干个段
 - 未分配的inode和数据块是被 位示图中的 段的锁 所保护的
 - 文件、目录和软连接都是单一的一个段
- 单操作多锁场景：有些更新操作需要原子化地更新盘上的数据结构，因此需要获得来自多个段的锁。为了实现原子化和避免死锁，操作如下：
 - 决定需要哪些锁，按照锁的inode地址对锁进行排序，而后再按顺序使用锁（预防死锁方法中的有序资源分配法）
 - 检查在第一阶段释放锁的数据是否被修改，若有，则释放锁，回到第一阶段；若无，则持有锁，完成操作，释放锁

7. 锁服务

Frangipani只需要锁服务提供通用的一些功能即可，并且不希望锁服务成为整个文件系统的瓶颈，因此锁服务有很多实现方法。**注意在讨论锁服务时，client指的是Frangipani File Server 而不是用户程序！Server指的是锁服务器！**

- 处理client 故障：使用租约（lease）
 - 租约时间为30秒，租约在client第一次联系server时，Server会给client一个租约，client必须在租约到期之前更新租约，否则server会认为client故障
 - 在网络故障的情况下，尽管client没有崩溃，client可能无法发送租约。这种情况下，client（也就是Frangipani File Server）会放弃所有的锁，丢弃掉整个cache，标志一个内部的标记。后续来自用户应用程序的请求到来时，会报错。这种情况下，必须卸载文件系统来修复错误
- 锁服务结构：整个锁服务由若干台Lock Server以及在Client（也就是Frangipani File Server）上的职员（clerk）模块组成。（锁服务 = 锁服务器 + 职员模块）。

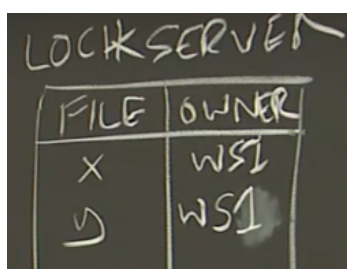
- 每台Frangipani File Server上有一个Clerk! 每个server上可以安装多个Frangipanni文件系统。但是一台机器只有一个clerk
- 锁是通过**表格 (table)** 来进行组织的。**每个Frangipani文件系统 (而不是Frangipani File Server!)** 都会有一张表格。这个表格的名字是ASCII字符。表中存储了锁相关的信息, 锁以64位整数的方式进行命名。Frangipani File Server中的锁表格内容如下:



FILE	LOCK	CONTENT
X	BUSY	~~~~~

- 文件标志符 (64位整数)
- 锁状态: busy/idle, 分别表示Frangipani File Server是否正在使用对应的数据
- 文件的内容

Lock Server中的表格内容如下:



LOCKSERVER	
FILE	OWNER
X	WS1
Y	WS1

- 文件标志符 (64位整数)
 - 锁的持有者
- 当一个文件系统被挂载时, 它会调用Clerk, 而后Clerk就会打开属于该文件系统的表格。当打开成功时, 锁服务器就会给Clerk返回一个租约标志符, 随后Frangipani File Server与锁服务器的通信一直需要使用该租约标志符。当Frangipani File Server上的Frangipani文件系统卸载时, Clerk会关闭对应的表格。
- Clerk和Lock Server之间通过异步消息 (asynchronous message) 而不是RPC通信。
 - 通信可能会传递的操作包括:
 - request
 - grant
 - revoke

- release

其中，grant 和 revoke 是Lock Server发向Clerk的消息，request和release是Clerk向Lock Server发送消息。而锁升级和锁降级是通过上述四种操作的组合来完成的。

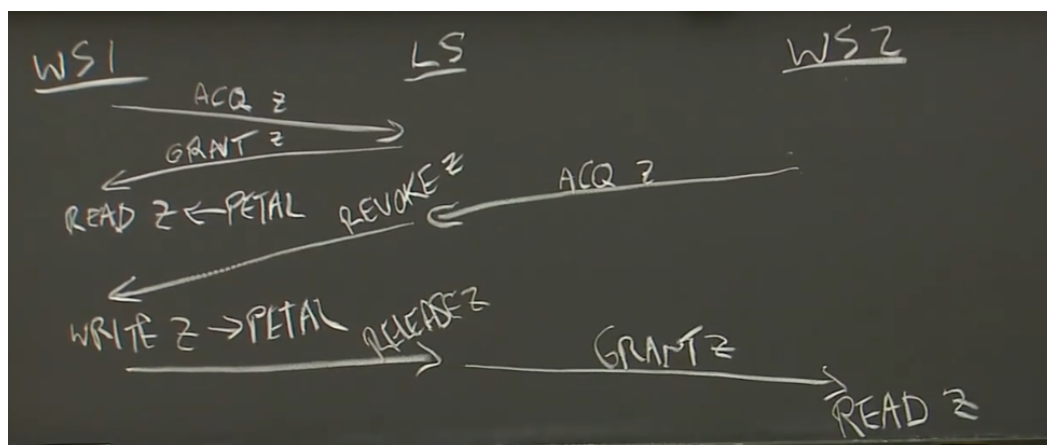
Revoke操作的具体流程：

1. 将日志写入条目（Moris认为是整个日志）
2. 将脏数据写回磁盘
3. 向Lock Server发送Release消息

- 锁服务器之间通过多数投票一致性算法 + 心跳信息 来完成容错和故障检测机制。
- Frangipani的锁是有粘性的（sticky），而锁机制本身需要在Lock Server和Clerk上都占用内存。因此为了避免占用过多的内存，对于超过1小时未使用的锁，会自动释放。
- 整个锁服务的全局信息是通过Paxos算法来进行复制并保持一致的。这里重用了原本用于Petal的Paxos算法。全局信息包括：
 - Lock Server列表
 - 不同Lock负责的区域
 - 已经打开但没关闭表格的Clerk

这些全局信息可以用于：达成共识、在锁服务器之间重新分配锁、故障恢复等

- 一个Tricky的情况：某台Frangipani File Server和锁服务失去联系。因此Frangipani File Server仍然认为自己租约没有过期，而此时锁服务已经把锁给了新的Frangipani File Server，而后原来的Server给Petal发送写请求，由于Petal不检查租约，因此Petal可能会写入不拥有锁的server的数据。可以通过给写请求打上时间戳来解决这一问题，或是将锁服务与Petal结合起来，让Petal拒绝过期了的请求
- 例子：其中WS1占有和释放的是写锁



7.1. 锁的重新分配

- 所有的锁被分成了大约100个组。每次分配锁时不是一个一个分配的，而是按组分配，可以提高效率。
- 锁在如下情况时会被重新分配：
 - 恢复崩溃的锁服务器
 - 通过其他服务器补偿崩溃锁服务器
 - 有服务器加入或离开锁服务器集群
 并且会保证：
 - 每个锁服务器提供的锁是均衡的
 - 最小化锁的重新分配
 - 一个锁最多由一个锁服务器提供
- 锁重新分配的流程：
 - 原持有锁的锁服务器在它们的内部状态中去掉这些锁
 - 后持有锁的服务器获得锁，并且联系那些 表格中有相关锁 的clerk，（后持有所的服务器）从它们（clerk）那里获得用于恢复的信息，并且通知clerk锁的提供者发生了变化
- Frangipani File Server恢复：此时Frangipani File Server A崩溃，日志中可能存在未写回的更新操作
 - Frangipani File Server A的租约到期（期间不释放锁！）
 - Lock Server联系另一台Frangipani File Server B（期间不释放锁！）

- Lock Server会给A的log上锁，该锁由B独占，同时B也继承来自A的锁，A的log 也有租期
- B完成恢复，这一过程见前文日志部分关于Frangipani File Server故障恢复的部分
- B释放所有的锁

之所以要给A的log上锁并给租期是为了保证，如果B恢复失败，Lock Server可以找另一台Frangipani File Server C 来恢复A的操作

8. 网络分区

- 锁服务发生网络分区时，只要大多数Lock Server正常工作则能正常工作
- Petal发生网络分区时，只要大多数Petal Server正常工作则能正常工作，但是分区的另外小部分Petal Server的数据将无法访问
- Frangipani File Server 与锁服务发生网络分区时，锁服务会认为相关Frangipani File Server崩溃；Frangipani File Server与Petal发生网络分区时，将无法访问相应的数据。当Frangipani File Server发现网络分区现象时，会拒绝上层用户代码的请求，直到网络分区恢复且文件系统重新挂载

9. 增删Frangipani File Server

- 增：直接增加即可，而后告知新的Server使用哪个Petal Virtual Server，以及如何与lock Server通信即可。
- 删：可以直接关机（因为会自动恢复）。当然最好是先刷盘，释放锁，再关机。

10. 备份

- Petal支持快照，使用写时复制
- 快照可以使用恢复程序进行恢复。但是这样会比较麻烦，一种解决方法为先由快照程序获得所有的锁，此时原来占有锁的Frangipani File Server会刷盘放锁，而后快照程序完成快照操作之后则释放锁，原有Frangipani File Server正常工作；另一种方法是将快照挂载为Frangipani的卷，但是这种方式文件只读。

11. 原子性

文件系统中有些操作是需要多个步骤的（例如创建文件，需要涉及目录以及inode的改变），需要保证这些步骤执行的原子性。因此Frangipani使用了分布式事务。

具体步骤为：

1. 获得 **所有** 的锁
2. 进行更新
3. 刷盘
4. 释放锁

12. 相关论文

- Petal 24
- 密码学、安全性机制等 13 37
- 最早的双层文件系统设计：UFS 4
- 大地址空间 8