淘宝网
Taobao.com

叠合式文件系统—— overlayfs

内核组
三百 <sanbai@taobao.com>

淘宝网 Taobao.com

Overlayfs 提纲

- **linux** 文件系统的一些基础知识
- 怎么发现 overlayfs（源起）
- Linux 上 vfs 的原理
- overlayfs 的原理

淘宝网 Taobao.com

基础知识

- Linux 的 page cache
  - read/write
  - mmap
- 常见疑问：好多内存都被 cache 占了，我想手工释放它
  - 答：不需要， linux 系统在需要内存时，自己会释放 cache
- 常见疑问：我想让 cache 一直保留不被收回，怎么做？
  - 答： mmap 后 mlock

基础知识

- 文件系统的“快照”（ snapshot ）
  - 多个 inode 指向相同的物理磁盘块
  - 需要引用计数以记录一个磁盘块被几个 inode 使用了

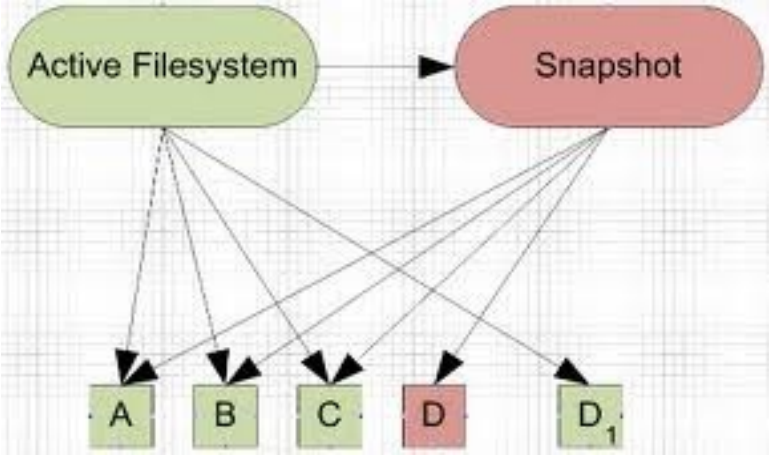


Image from http://sannasanalyst.blogspot.com/2010/06/netapp-fractional-reserve.html

淘宝网
Taobao.com

Overlayfs 提纲

- linux 文件系统的一些基础知识
- 怎么发现 **overlayfs**（源起）
- Linux 上 vfs 的原理
- overlayfs 的原理

淘宝网
Taobao.com

源起

- 需求：在一个 linux 系统上，跑多个不同应用，这些应用由不同的运维人员来操作，为了避免互相干扰，希望运维只能看见自己的文件，而看不见别的应用的文件信息
- 常见解决方案： linux 系统上装多个虚拟机
- 缺陷：多个应用公用的一些动态链接库（比如 libc.so ）和配置文件（比如 hosts.conf ）就复制了多份。例如：原先一个系统在运行时系统文件占了 500M 的 cache ，那么现在装了 4 台虚拟机，就一共要占 2G 的 cache 。
- 平台的层次越深，这种浪费就越大， c/c++ 程序只是用一些系统的 .so ， java 程序依靠 jvm ，需要的公用库就更多了。

淘宝网 Taobao.com

源起

- 我们的最初方案：把 linux 系统装到 ext4 上，然后做 4 个 snapshot（"快照"），这 4 个 snapshot 分别 mount 到 4 个目录，4 个运维人员 chroot 到这 4 个目录里，然后就自己维护自己的，干扰不到别人的文件。（我们最初以为一个文件和它的快照是共享 cache 的）
- 疑问：直接把系统常用的动态链接库做 4 个软链接出来，给 4 个运维用不就行了？
- 答：第一，动态链接库以及各种系统文件很多，不可能一一做软链接；第二，也是关键的一点，如果其中一个运维人员错误操作——例如覆盖写了某个系统文件，那么其他的运维将会受影响，因为软链接实际指向的是同一个实际文件。

淘宝网 Taobao.com

源起

- ext4 snapshot 方案的纰漏：一个文件和它的 snapshot 各自有自己的 page cache，并非共享！
- 现有 linux 的支持快照的文件系统（ ext2/ext3/ext4、 btrfs、 ocfs2 ），它们的快照在 mount 以后，就是个“新的”文件系统、“新的” inode，新的 inode 就有自己“新的” page cache
- 我们还考虑过 link-ref（多个文件共享一样的磁盘块），但问题一样：不同的 inode 无法共享一套 page cache

淘宝网 Taobao.com

源起

- 最终方案：overlayfs
- overlayfs 能将两个目录“合并”，比如两个目录：
  - dir1/ 目录里有
    ./fire
    ./water
  - dir2/ 目录里有
    ./apple
    ./banana

mount -t overlayfs overlayfs -olowerdir=/dir1,upperdir=/dir2 /test1/ （我们不修改 dir1 目录里的文件）以后

- test1/ 目录里将会有
  ./fire
  ./water
  ./apple
  ./banana

其中 /test1/fire 和 /dir1/fire 其实是“同一个文件”，用的是一套 page cache

淘宝网 Taobao.com

生产系统的做法

- 1. 做一个base目录，比如/base，里面存放了所有系统文件，类似/etc、 /bin/、 /lib/等
- 2. 准备4个目录来存储文件，比如/store1,/store2,/store3,/store4
- 3. 准备4个空目录（仅仅作为名字）比如 /system1、 /system2、 /system3、 /system4
- 4. mount

```
mount -t overlayfs overlayfs -olowerdir=/base,upperdir=/store1 /system1/
mount -t overlayfs overlayfs -olowerdir=/base,upperdir=/store2 /system2/
mount -t overlayfs overlayfs -olowerdir=/base,upperdir=/store3 /system3/
mount -t overlayfs overlayfs -olowerdir=/base,upperdir=/store4 /system4/
```

运维人员执行：

cd /system2/

chroot .

后，就拥有自己“独立”的系统了。系统文件的page cache是共享的。

淘宝网 Taobao.com

Overlayfs 概览

- 由 SUSE 的 Miklos Szeredi 开发，实现非常简洁
- 设计用途
  - 各个 linux 发行版的 livecd（以前的方案是 union mount ）
  - 很多嵌入式设备的“恢复原厂设置”功能
  - “virtualized systems built on a common base filesystem”

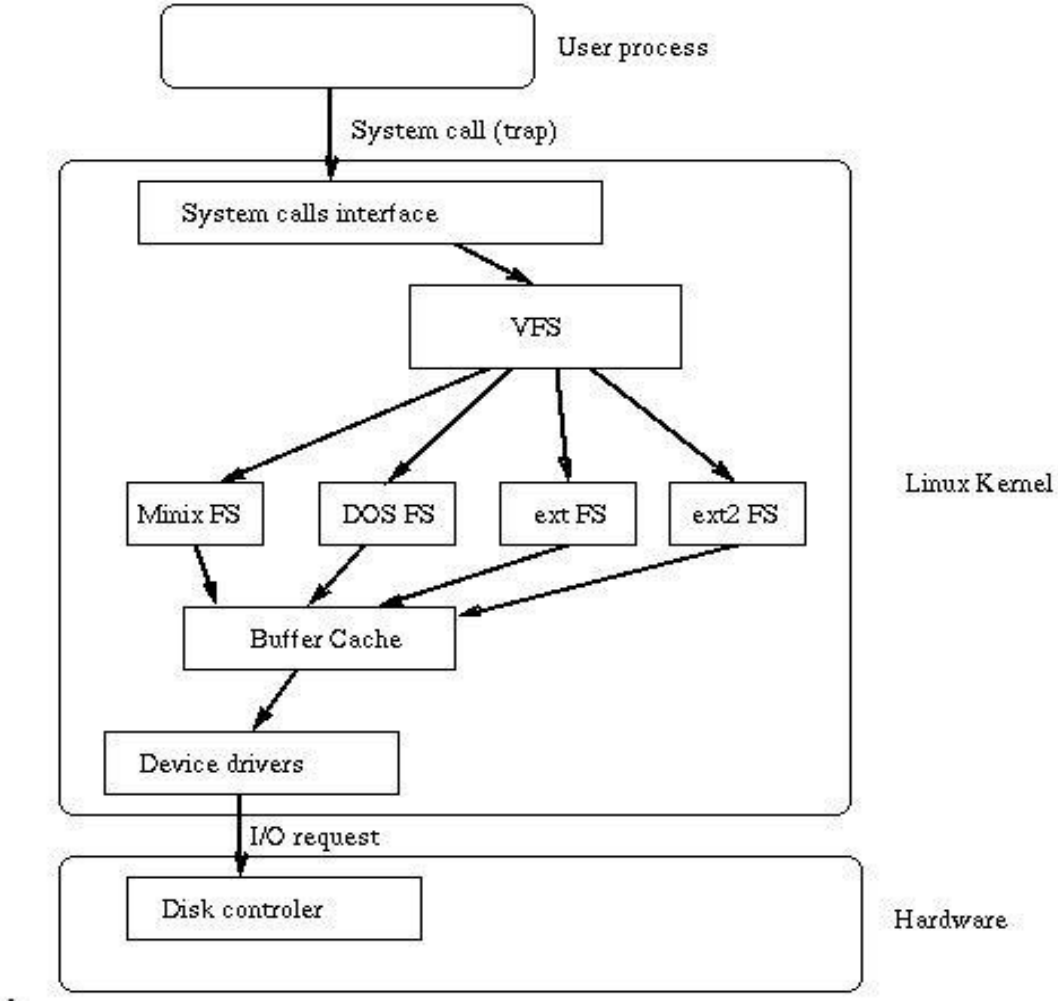
淘宝网
Taobao.com

Overlayfs 提纲

- linux 文件系统的一些基础知识
- 怎么发现 overlayfs（源起）
- **Linux** 上 **vfs** 的原理
- overlayfs 的原理

vfs 原理

- Virtual File System 是一种软件机制，将不同类型的文件系统统一为一套接口

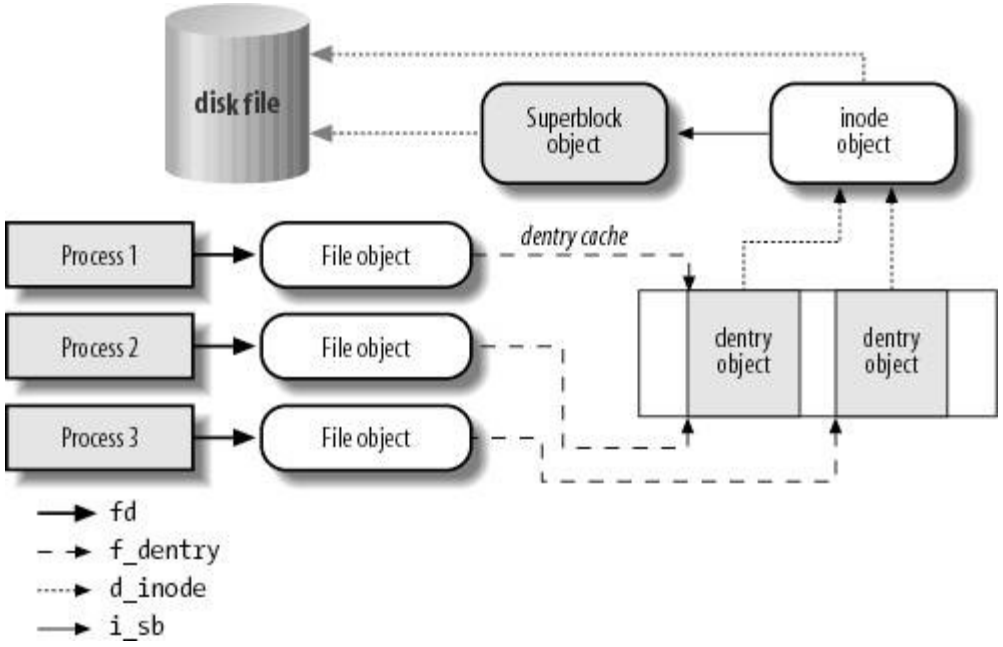


http://space.itpub.net/?uid-26110315-action-viewspace-itemid-719438

淘宝网 Taobao.com

vfs 原理

- VFS在内存中要为（一部分）真实的物理文件建立一套内存视图



此图来自《Understanding Linux Kernel （3rd version）》，被无数的网站引用，但它确实很能说明问题

淘宝网 Taobao.com

vfs 原理

- 普遍的疑惑：inode和dentry有什么区别，为什么会有这两个内核对象？
- “文件“这个词包含了两层意思：
  - 这个”文件“所在的位置，即与兄弟姐妹父母亲的关系
  - 这个”文件“自身的属性，即大小、访问权限等信息
- inode和dentry都是内存里的对象，其中，inode对应一个在磁盘里的真实文件（仅限磁盘文件系统），而dentry没有对应的磁盘对象
- inode主要描述了文件的自身属性
  - 创建时间、被修改时间
  - 操作权限
  - 该怎么操作（inode_operations）
  - 拿到inode对象并不能直接告诉你文件在哪个目录下
- dentry主要描述了文件（相对于其它文件和目录）的位置
  - 父目录是谁
  - 如果自己是目录，那下面有哪些文件
  - 该怎么处理位置关系（dentry_operations）

淘宝网 Taobao.com

vfs 原理

```
struct inode {
        struct hlist_node       i_hash;
        struct list_head        i_list;         /* backing dev IO list */
        struct list_head        i_sb_list;
        struct list_head        i_dentry;
        unsigned long           i_ino;
        atomic_t                i_count;
        unsigned int            i_nlink;
        uid_t                   i_uid;
        gid_t                   i_gid;
        dev_t                   i_rdev;
        u64                     i_version;
        loff_t                  i_size;
#ifdef __NEED_I_SIZE_ORDERED
        seqcount_t              i_size_seqcount;
#endif
        struct timespec         i_atime;
        struct timespec         i_mtime;
        struct timespec         i_ctime;
        blkcnt_t                i_blocks;
        unsigned int            i_blkbits;
        unsigned short          i_bytes;
        umode_t                 i_mode;
        spinlock_t              i_lock; /* i_blocks, i_bytes, maybe i_size */
        struct mutex            i_mutex;
        struct rw_semaphore     i_alloc_sem;
        const struct inode_operations   *i_op;
```

主要描述文件自有属性

对于磁盘文件系统，其中的i_atime、 i_mtime、 i_blocks等都是从磁盘里读出来的

淘宝网 Taobao.com

vfs 原理

```
struct inode_operations {
        int (*create) (struct inode *,struct dentry *,int, struct nameidata *);
        struct dentry * (*lookup) (struct inode *,struct dentry *, struct nameidata *);
        int (*link) (struct dentry *,struct inode *,struct dentry *);
        int (*unlink) (struct inode *,struct dentry *);
        int (*symlink) (struct inode *,struct dentry *,const char *);
        int (*mkdir) (struct inode *,struct dentry *,int);
        int (*rmdir) (struct inode *,struct dentry *);
        int (*mknod) (struct inode *,struct dentry *,int,dev_t);
        int (*rename) (struct inode *, struct dentry *,
                        struct inode *, struct dentry *);
        int (*readlink) (struct dentry *, char __user *,int);
        void * (*follow_link) (struct dentry *, struct nameidata *);
        void (*put_link) (struct dentry *, struct nameidata *, void *);
        void (*truncate) (struct inode *);
        int (*permission) (struct inode *, int);
        int (*check_acl)(struct inode *, int);
        int (*setattr) (struct dentry *, struct iattr *);
        int (*getattr) (struct vfsmount *mnt, struct dentry *, struct kstat *);
        int (*setxattr) (struct dentry *, const char *,const void *,size_t,int);
        ssize_t (*getxattr) (struct dentry *, const char *, void *, size_t);
        ssize_t (*listxattr) (struct dentry *, char *, size_t);
        int (*removexattr) (struct dentry *, const char *);
        void (*truncate_range)(struct inode *, loff_t, loff_t);
        long (*fallocate)(struct inode *inode, int mode, loff_t offset,
                          loff_t len);
        int (*fiemap)(struct inode *, struct fiemap_extent_info *, u64 start,
                      u64 len);
};
```

提供文件属性的修改接口：如何创建、如何添加链接、如何截断、如何删除等

对于磁盘文件系统，这些接口的具体实现都变为对磁盘内容的操作

淘宝网 Taobao.com

vfs 原理

```
struct dentry {
        atomic_t d_count;
        unsigned int d_flags;           /* protected by d_lock */
        spinlock_t d_lock;              /* per dentry lock */
        int d_mounted;                  /* obsolete, ->d_flags is now used for this */
        struct inode *d_inode;

        struct hlist_node d_hash;       /* lookup hash list */
        struct dentry *d_parent;        /* parent directory */
        struct qstr d_name;

        struct list_head d_lru;         /* LRU list */

        union {
                struct list_head d_child;       /* child of parent list */
                struct rcu_head d_rcu;
        } d_u;
        struct list_head d_subdirs;     /* our children */
        struct list_head d_alias;       /* inode alias list */
        unsigned long d_time;           /* used by d_revalidate */
        const struct dentry_operations *d_op;
        struct super_block *d_sb;       /* The root of the dentry tree */
        void *d_fsdata;                 /* fs-specific data */

        unsigned char d_iname[DNAME_INLINE_LEN_MIN];    /* small names */
};
```

主要描述文件的位置关系

父目录是谁、有哪些子文件（如果自己是目录）

淘宝网 Taobao.com

vfs 原理

```
struct dentry_operations {
        int (*d_revalidate)(struct dentry *, struct nameidata *);
        int (*d_hash) (struct dentry *, struct qstr *);
        int (*d_compare) (struct dentry *, struct qstr *, struct qstr *);
        int (*d_delete)(struct dentry *);
        void (*d_release)(struct dentry *);
        void (*d_iput)(struct dentry *, struct inode *);
        char *(*d_dname)(struct dentry *, char *, int);
#ifndef __GENKSYMS__
        struct vfsmount *(*d_automount)(struct path *);
        int (*d_manage)(struct dentry *, bool);
#endif
};
```

提供了文件位置关系处理的接口：

怎么计算hash、怎样比较文件名

淘宝网 Taobao.com

vfs 原理

- 问：文件的删除、创建等都是 inode_operations 接口提供，那文件的 read/write/mmap 呢？
- 答： struct inode 里有一个“ struct file_operations i_fop”, 这个 file_operations 就是负责文件自身操作的接口

淘宝网 Taobao.com

vfs 原理

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
        ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, loff_t, loff_t, int datasync);
        ......
};
```

淘宝网 Taobao.com

vfs 原理

- 问：文件的删除、创建等都是 inode_operations 接口提供，那文件的 read/write/mmap 呢？
- 答： struct inode 里有一个“ struct file_operations i_fop”, 这个 file_operations 就是负责文件自身操作的接口

淘宝网 Taobao.com

vfs 原理

- 问：当我们读一个文件时（先open，后read），发生了什么？（为了避免牵扯太多，我假设文件对象已经都在内存中，也就是说近期已经被访问过了）
- 先查找文件，沿着路径在每一级目录里找对应name的dentry
  - 调用d_hash根据name等计算hash值（d_hash是可以文件系统开发者自己实现的接口）
  - 根据hash值在hash表里找到对应的桶（linux给有一张hash表“dentry_hashtable”，整个系统的dentry都放在里面）
  - 这个桶后面有一串dentry（都是相同hash值的），再用d_compare去一个个对比是不是要找的那个name
- 找到dentry后再调用 __dentry_open
  - dentry里有指针指向对应的inode
  - f->f_op = fops_get(inode->i_fop); 把inode里的i_fop赋值给struct file 对象
  - 从此，对这个文件的read/write/mmap都由file_operations接口负责了

淘宝网 Taobao.com

vfs 原理

- vfs 提供了高度的灵活性：对文件属性的操作、对文件位置的操作、对文件数据的操作都是可以由开发者自己实现的（这也是一种类似面向对象的框架）
- 不同的文件系统通过不同的 inode_operations 和 file_operations 实现（当然，还有别的 operations ）来完成自己的特性。即使是 windows 的 fat、 ntfs 和 plan9 的文件系统都已经移植到了 linux

淘宝网 Taobao.com

Overlayfs 提纲

- linux 文件系统的一些基础知识
- 怎么发现 overlayfs（源起）
- Linux 上 vfs 的原理
- **overlayfs** 的原理

淘宝网 Taobao.com

Overlayfs 的原理

- VFS 的高度灵活性已经给我们很多启发了
- 既然对一个文件的操作是可以任意实现的，那就完全可以把对文件 A 的 read 直接转为对另一个文件 B 的 read
- 这就是 overlayfs 的根本：把对 overlayfs 文件系统里一个文件的操作，转为对 lowerdir 里对应文件的操作

淘宝网 Taobao.com

Overlayfs 的原理

```
static struct file *ovl_open(struct dentry *dentry, struct file *file,
                             const struct cred *cred)
{
        int err;
        struct path realpath;
        enum ovl_path_type type;

        type = ovl_path_real(dentry, &realpath);    /* overlayfs在dentry的d_fsdata成员里放了文件对应的lowerdir和
                                                    upperdir的信息，此处便可以找到该文件对应的lowerdir的文件了*/
        ......
        return vfs_open(&realpath, file, cred);      /* 把这个lowerdir对应的文件的path传给vfs_open，这样，被真正
                                                     open的就是lowerdir对应的“下层文件”了，之后的read/mmap都是
                                                     调用的该文件的file_operation */
}
```

淘宝网 Taobao.com

Overlayfs 的原理

```
static int ovl_readdir(struct file *file, void *buf, filldir_t filler)
{
        ......
        if (!od->is_cached) {
                struct ovl_readdir_data rdd = { .list = &od->cache };
                res = ovl_dir_read_merged(&upperpath, &lowerpath, &rdd);        /*把上下两层目录的内容都读出来，合并，放入rdd这个数据结构 */
                if (res) {
                        ovl_cache_free(rdd.list);
                        return res;
                }
                ......
                ovl_seek_cursor(od, file->f_pos);                               /* 将od->curser指向od->cache里对应偏移的entry，
        }                                                                         这个entry可以理解为类似该目录下面文件的dentry */

        while (od->cursor.next != &od->cache) {                                 /* rdd数据结构里的list成员，实际是指向od->cache的，
                ......                                                            所以移动od->cursor就是在沿着rdd->list找到所有dentry */
                p = list_entry(od->cursor.next, struct ovl_cache_entry, l_node);
                off = file->f_pos;
                if (!p->is_whiteout) {
                        over = filler(buf, p->name, p->len, off, p->ino,
                                      p->type);
                        if (over)
                                break;
                }
                file->f_pos++;
                list_move(&od->cursor, &p->l_node);
        }
        return 0;                                                               /* 于是乎，两层目录下的文件名都读出来放到一起了 */
}
```

淘宝网 Taobao.com

Overlayfs 的原理

- 问：如果上下两层目录里有相同名字的文件，怎么处理？
- 答：上面的“盖住”下面的，显示的是上层目录的文件
- 问：下层目录是 readonly 的，那如果它对应的 overlayfs 里的文件能被修改吗？
- 答：一旦被修改， overlayfs 实际是把它拷贝到上层目录，然后修改之

淘宝网 Taobao.com

Overlayfs 原理

- overlayfs 的特性：将上下两层目录合并为一个文件系统，“下层”目录是只读的，所以修改都是对”上层“的修改
- 在这个新文件系统里
  - 新创建的文件实际是创建在“上层”目录的
  - 删除文件时：
    - 如果文件来自“下层”目录，则隐藏它（看上去就像删除了一样）
    - 如果文件来自“上层”目录，则直接删除即可
  - 写一个文件时：
    - 如果文件来自“下层”，则拷贝其到上层目录，然后写这个”上层“的新文件
    - 如果文件来自”上层“，直接写即可

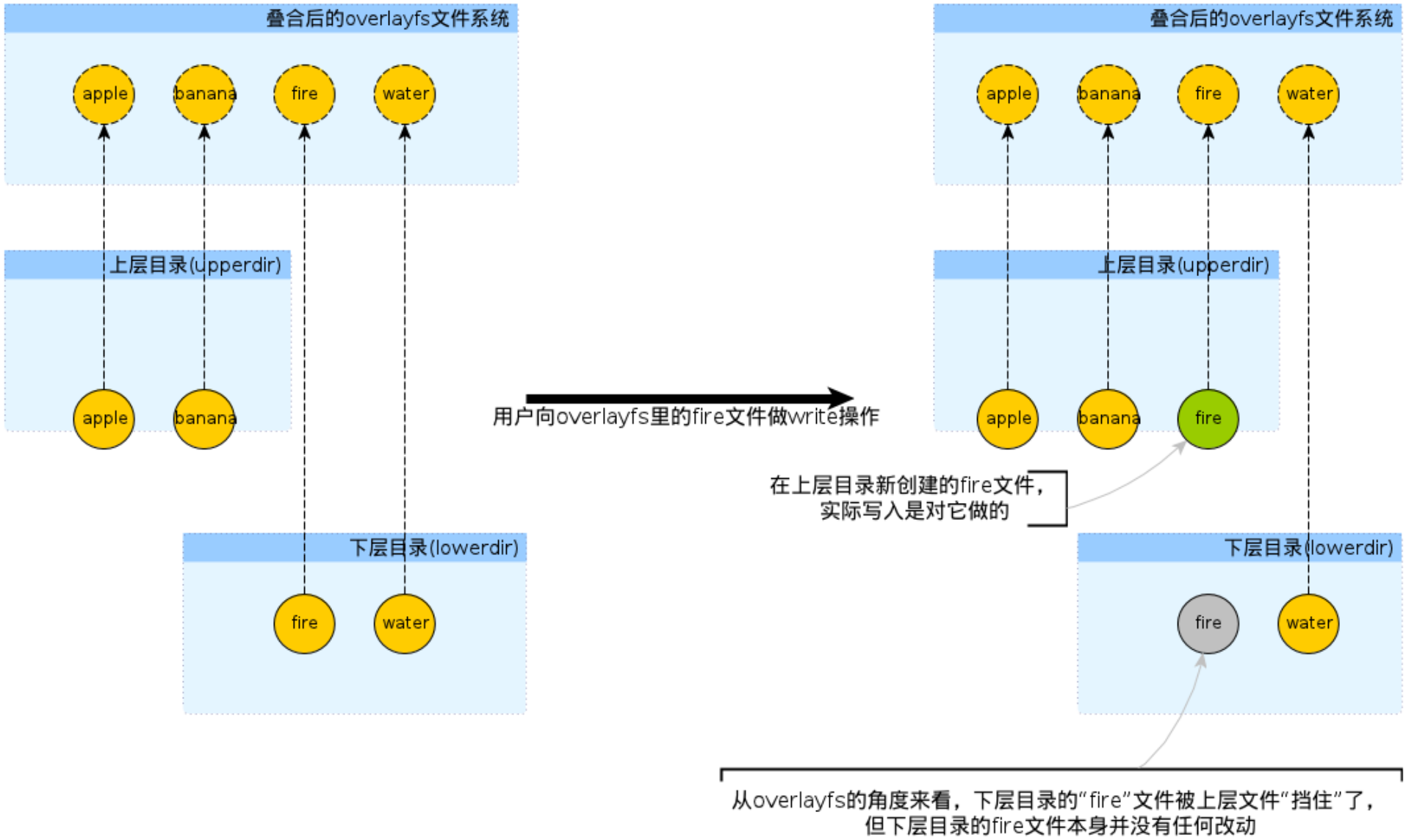
淘宝网 Taobao.com

Overlayfs 的原理

```
static struct file *ovl_open(struct dentry *dentry, struct file *file,
                             const struct cred *cred)
{
        int err;
        struct path realpath;
        enum ovl_path_type type;

        type = ovl_path_real(dentry, &realpath);
        if (ovl_open_need_copy_up(file->f_flags, type, realpath.dentry)) {
                if (file->f_flags & O_TRUNC)
                        err = ovl_copy_up_truncate(dentry, 0);
                else
                        err = ovl_copy_up(dentry);          /* 如果需要，是要拷贝下层的文件的 */
                if (err)
                        return ERR_PTR(err);

                ovl_path_upper(dentry, &realpath);
        }

        return vfs_open(&realpath, file, cred);
}
```

淘宝网
Taobao.com

向 Overlayfs 写入



淘宝网 Taobao.com

Overlayfs 的原理总结

- 有 VFS 的高度灵活性，才有 overlayfs 的简洁实现
  - 上下合并
  - 同名遮盖
  - 写时拷贝

（如果是一个很大的文件，则写时拷贝是比较费时的，但毕竟只需要一次拷贝，下次就不需要了）

Overlayfs 现状

- 2011 年 6 月， Miklos 曾询问社区 overlayfs 是否可以并入 upstream
  - Andrew Morton 表示怀疑，觉得这种需求最好由 fuse 来实现：如果因为 overlayfs 简单就把它合进内核的话，那么“ Not merging it would be even smaller and simpler”
  - Linus：“ So Andrew, I think that arguing that something _can_ be done with fuse, and thus _should_ be done with fuse is just ridiculous.” （参考 http://lwn.net/Articles/447650/ ）
- 层叠型文件系统会进 kernel ，但，是 overlayfs 还是 uni-mount 进，抑或都进，目前还不明朗
- 目前 overlayfs 在开发上更活跃

Q & A
Thanks!