FUSE (Filesystem in USErspace) on Open Solaris

Sunil Subramanya
DTC Intelligent Storage Consortium (DISC)
University of Minnesota

Topics

- What is FUSE
- General Information
- How FUSE works
- How does Application use FUSE
- User space and kernel communication
- Fuse Message Format
- FUSE on Open Solaris Project
  - Phase 1 Design
  - Phase 2 - Mmap() design

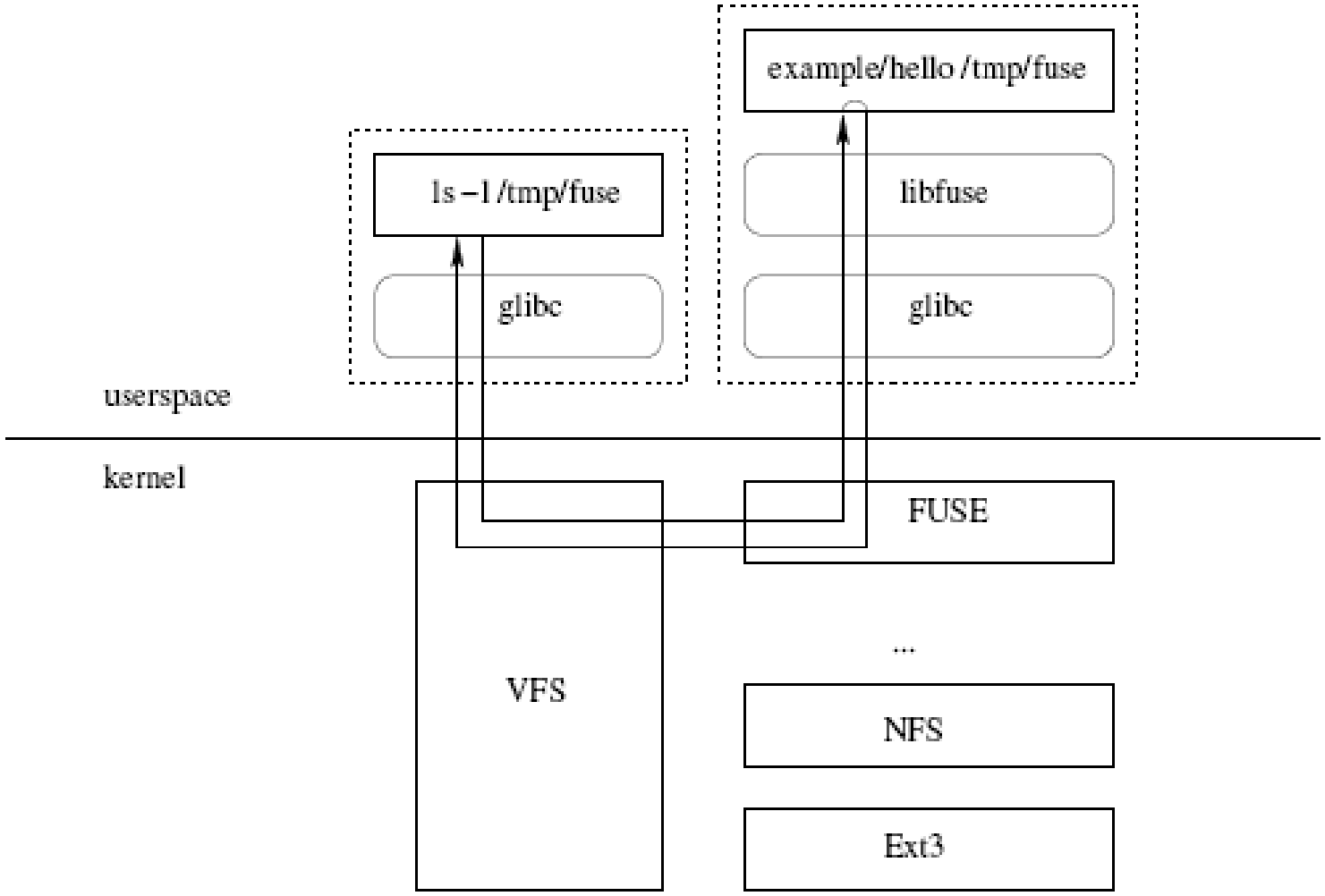
What is FUSE

- FUSE is a **framework** which allows implementing file systems in the user space.
- It consists of 2 components
  - User space library (dynamically linkable) which is used by file system implementers.
  - Kernel module which binds to the VFS and redirects calls to the fuse library.
- Example FS using Fuse: encfs, cvsfs, sshfs, gmailfs, zfs, …

General Info

- Developed originally for Linux, currently part of Linux Kernel 2.6.14.
- Later ported to FreeBSD and Mac OS X (quite recently done).
- Fuse Library remains the same across Operating Systems, ensures file system applications need no rework.

How FUSE works



http://fuse.sourceforge.net/

How FUSE works contd…

- FUSE Kernel module will register with VFS.
- Application implementing file system at user space will use FUSE library and provide following:
  - Specify the mount point in the current file system path to where its file system will be mounted.
  - Expose methods that do all the necessary file handling such as:
    - Creating directory
    - Reading directory
    - Reading file
    - Writing file etc…

How FUSE works contd…

- Fuse Library invokes mount system call at the provided mount point specifying file system type as fuse - ensures mounting of FS App.
- Any file system call is passed by kernel to VFS, which passes the request to Fuse Kernel Module for calls at the mount point.
- Fuse Kernel Module passes the call on to Fuse Library, which invokes appropriate FS App. registered method.

How does Application use FUSE

- Develop application specific implementation for each of the necessary file system methods.
- The interface for these methods is defined by FUSE library.
- Invoke the entry point in the FUSE library, providing function pointers of the implemented methods.

User space and kernel communication

- Fuse works by passing of System calls (and data) from kernel module to fuse lib and data from fuse lib to kernel module.
- This communication uses device file descriptor(/dev/fuse), opened during mounting.
- So Fuse kernel module registers itself as a character device driver.
- Fuse Lib. makes a blocking read call in to fuse kernel module.
- For any incoming request from VFS, kernel module unblocks read thread and sends VFS request info.

User space and kernel communication

- Fuse Lib returns data provided by FS App. by making a write call on kernel module.
- Fuse Kernel module returns back received data to VFS.
- Performance Issue ?
  - Can redundant copy be avoided ?

Fuse Message format

- Every request sent to the fuse library (2.5.3) has the following header:

```
struct fuse_in_header {
        __u32   len;
        __u32   opcode;
        __u64   unique;
        __u64   nodeid;
        __u32   uid;
        __u32   gid;
        __u32   pid;
        __u32   padding;
};
```

- Each operation has its own unique payload structure.

Fuse Message format

- Every response from the fuse library (2.5.3) has the following header:

```
struct fuse_out_header {
        __u32       len;
        __s32       error;
        __u64       unique;
};
```

- unique identifies the request this response associates to.

Non-privileged mounts

- Mounting Fuse file System doesn’t require super user privilege.
- So the user’s FS runs with the user’s privilege.
- Separate Mount utility does the job of mounting.

Fuse on Open Solaris

- How it started?
  - Internal Project as part of Sun's Excellence in Engineering Mentoring program
  - Requested by a big Sun customer, Marketing Development Engineering (MDE) team worked on porting the FreeBSD version.
  - MDE got the preliminary version of this port done.
  - We took over from them and are reworking for ultimate **integration into Solaris**.

Fuse on Open Solaris

- We have requests for Fuse on Solaris to support:
  - Davfs2: allows user to mount webdav server as a local filesystem.
  - EncFs: Encrypted File System.

  Available on Linux, need to port it to Solaris.

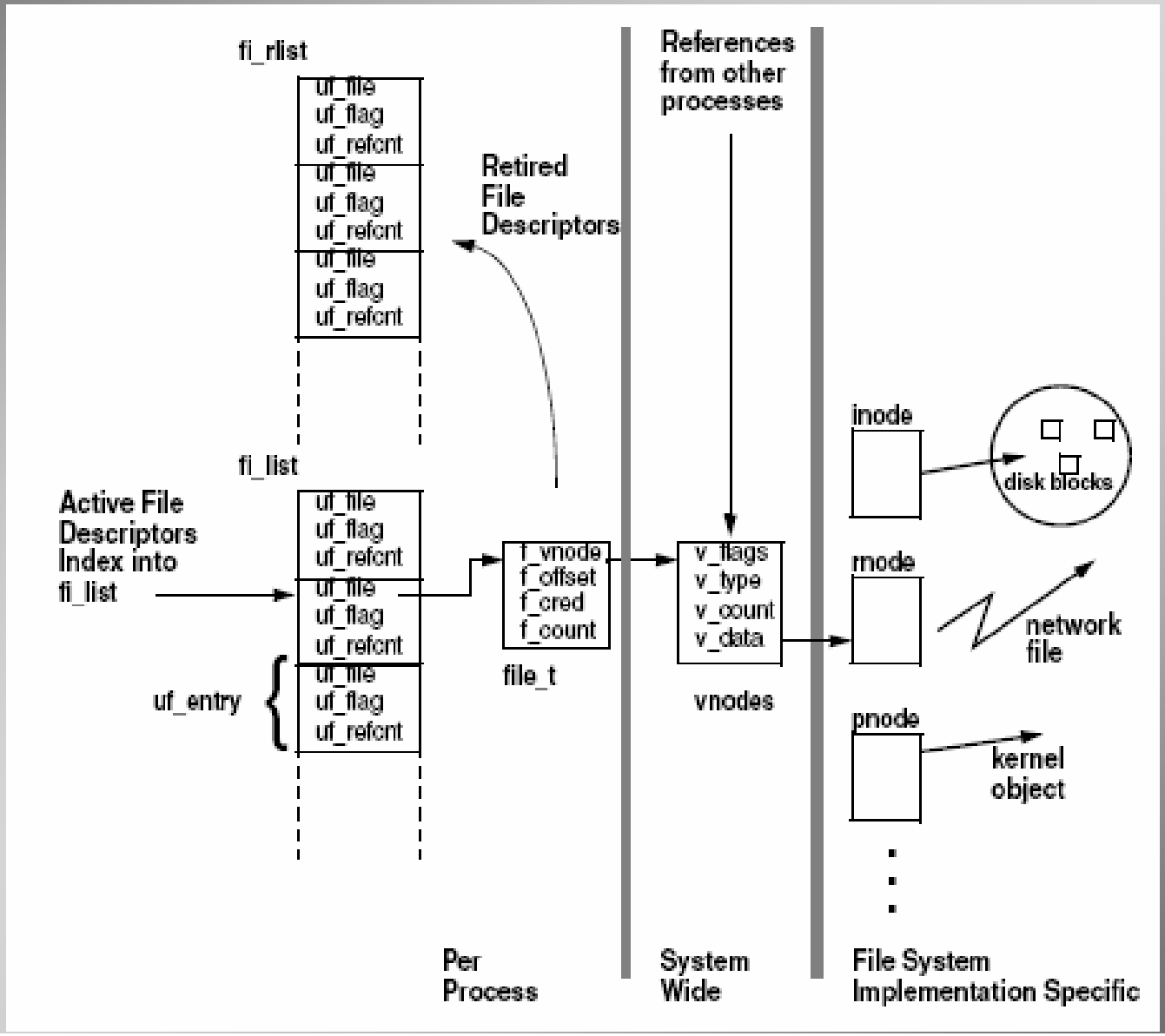
FUSE on Open Solaris

- Phase1:
  - To port existing Fuse implementation from FreeBSD to Open Solaris.
  - Current work.
    - Sunil – Porting kernel module.
    - Mark – Working on the mount utility.
  - Implemented few vnode ops, few more to be done.
  - Other work to be done include – porting fuse library, implement necessary locks and kernel handling for more than one FS mount on Solaris.

Fuse on Open Solaris Design

- Solaris VFS is Vnode based.
- Each opened file will have only one vnode.
- There usually will be many file_t structures pointing to a single vnode.

Design

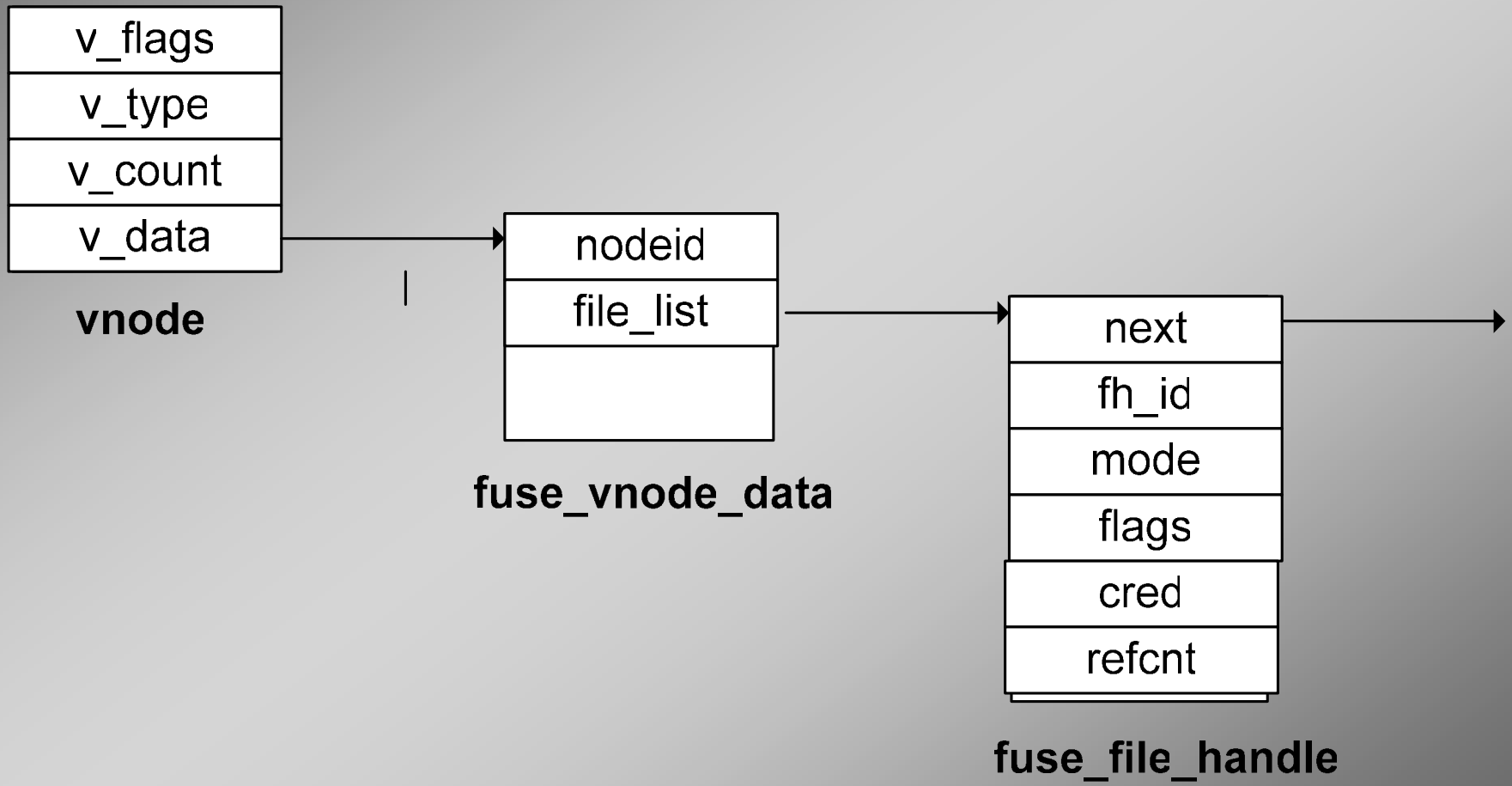


Solaris Internals Book Second Edition

Design…

- Fuse Library is very file* centric (Linux way of dealing with files).
- We need to keep track of all open files associated with each vnode.
- So each vnode is associated with a list of file handles

Design…


v_flags
v_type
v_count
v_data
vnode
nodeid
file_list
fuse_vnode_data
next
fh_id
mode
flags
cred
refcnt
fuse_file_handle

Design…

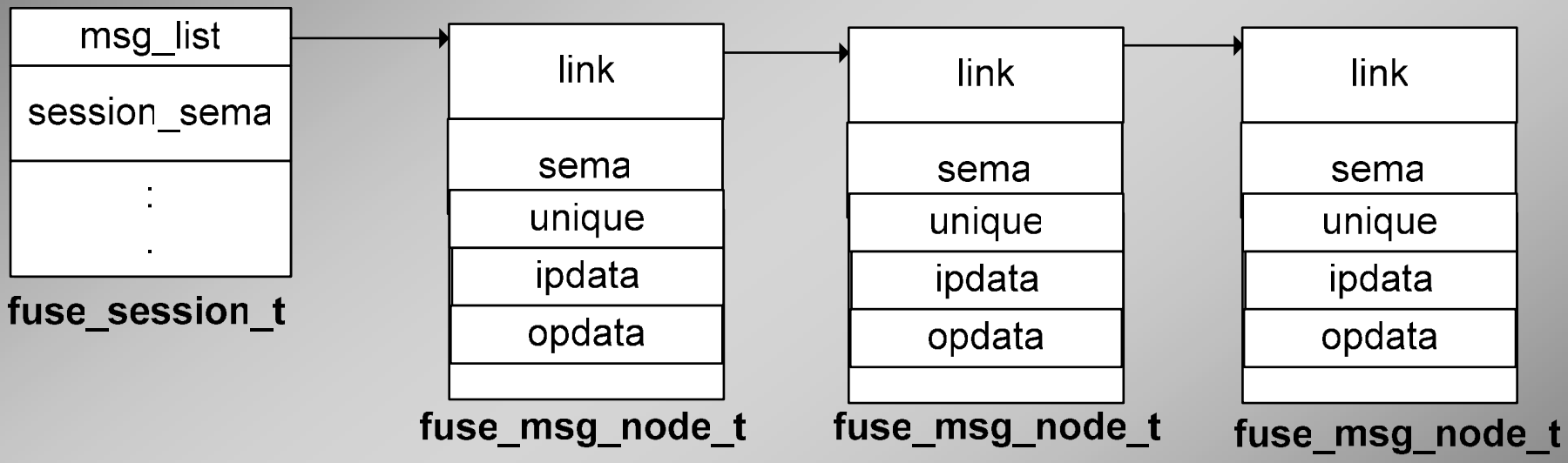
- Each mounted FS App. will have a fuse_session data structure
- AVL Tree is used as a way to retrieve vnode from nodeid

```
typedef struct fuse_session
{
        kmutex_t     sesssion_mutx;
        ksema_t      session_sema;   /* devops read sleeps over it */
        list_t       msg_list;       /* message awaiting service rest here */
        avl_tree_t   avl_cache;      /* used to retrieve vnode from nodeid */
        uint32_t     minor;
        uint32_t     state;
        uint64_t     unique;         /* message identfier used between library and */
                                     /* kernel module */
        cred_t      *usercred;     /* Credentials passed by fuse library */
        uint32_t     max_write;     /* Max Write value passed by fuse library */
                                    /*during mount */
} fuse_session_t;
```

Design …

- Every request received from VFS is represented as a message (fuse_msg_node)
- Every message is identified by a unique 64 bit value, passed to fuse library.
- Contains fuse header and payload to be sent to fuse library.
- Queued in a list maintained per session

Design …



- Each request is inserted at the tail of the list
- Devops read processes requests from the front of the list

Code Snippet

```
fuse_fs_readlink(…)
{
    …
    setup_msg_onlyheader(msgp, FUSE_READLINK,
            FUSE_GET_UNIQUE(sep), VNODE_TO_NODEID(vp), cred_p);

    /* Queue request to be sent to fuse library */
    if ((err = fuse_queue_request_wait(sep, msgp))) {
            DEBUG(" Obtained err=%d for FUSE_READLINK request \n", err);
            goto out;
    }

    err = msgp->opdata.fouth->error;
    if (!err) {
            uiomove(msgp->opdata.outdata, msgp->opdata.outsize, UIO_READ,
                uiop);
    }
    …
}
```

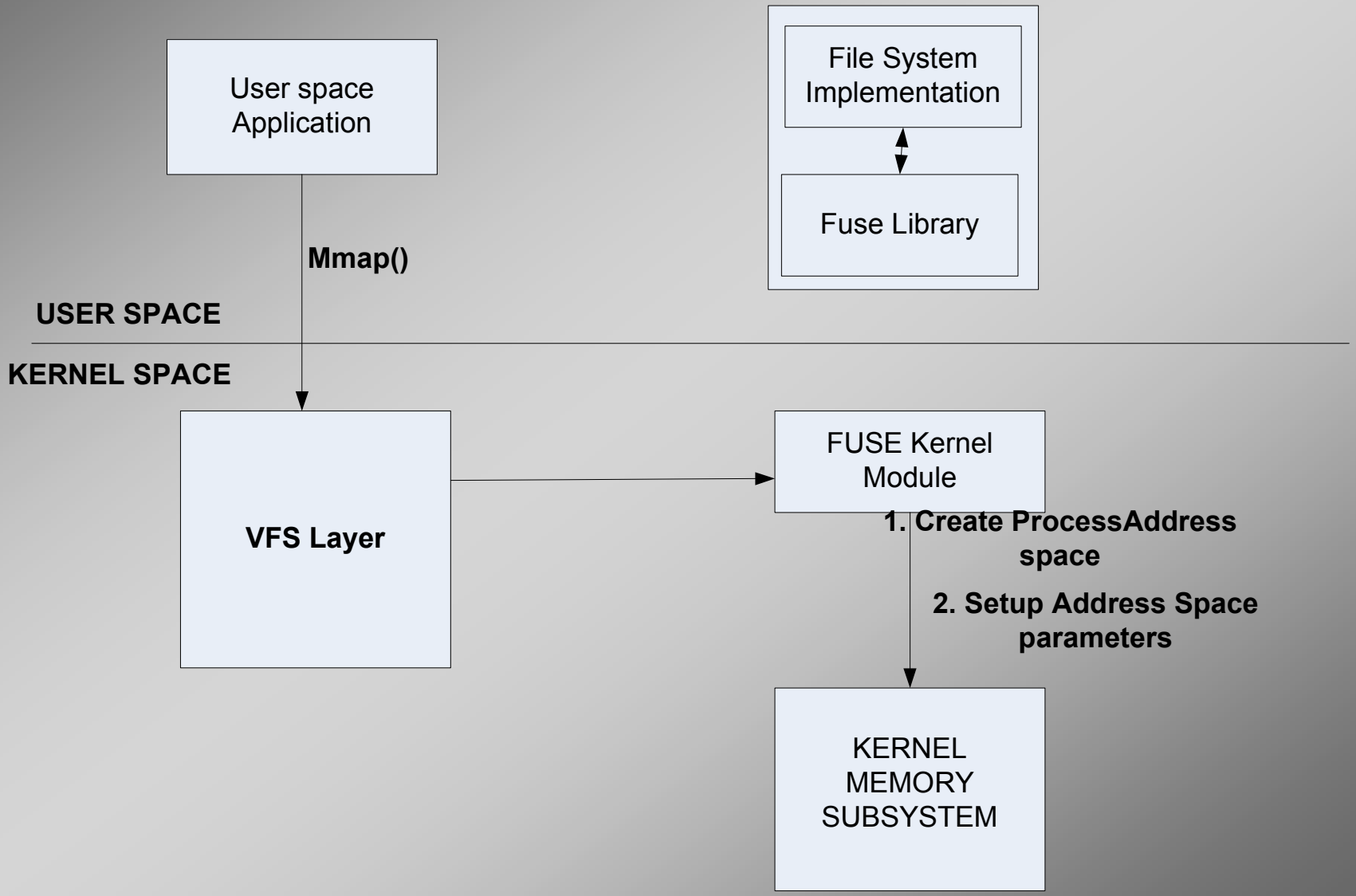
FUSE on Open Solaris

- Phase 2: New Features not in current Fuse implementation (fuse 2.5.3):
  - Implementation of Mmap functionality
  - Provide record locking, ACL support
  - Address performance issues related to FUSE message passing mechanism.
  - Provide inode persistence in the kernel,
    - current fuse doesn’t do any persistence.
    - So any FS App. crash means all information lost

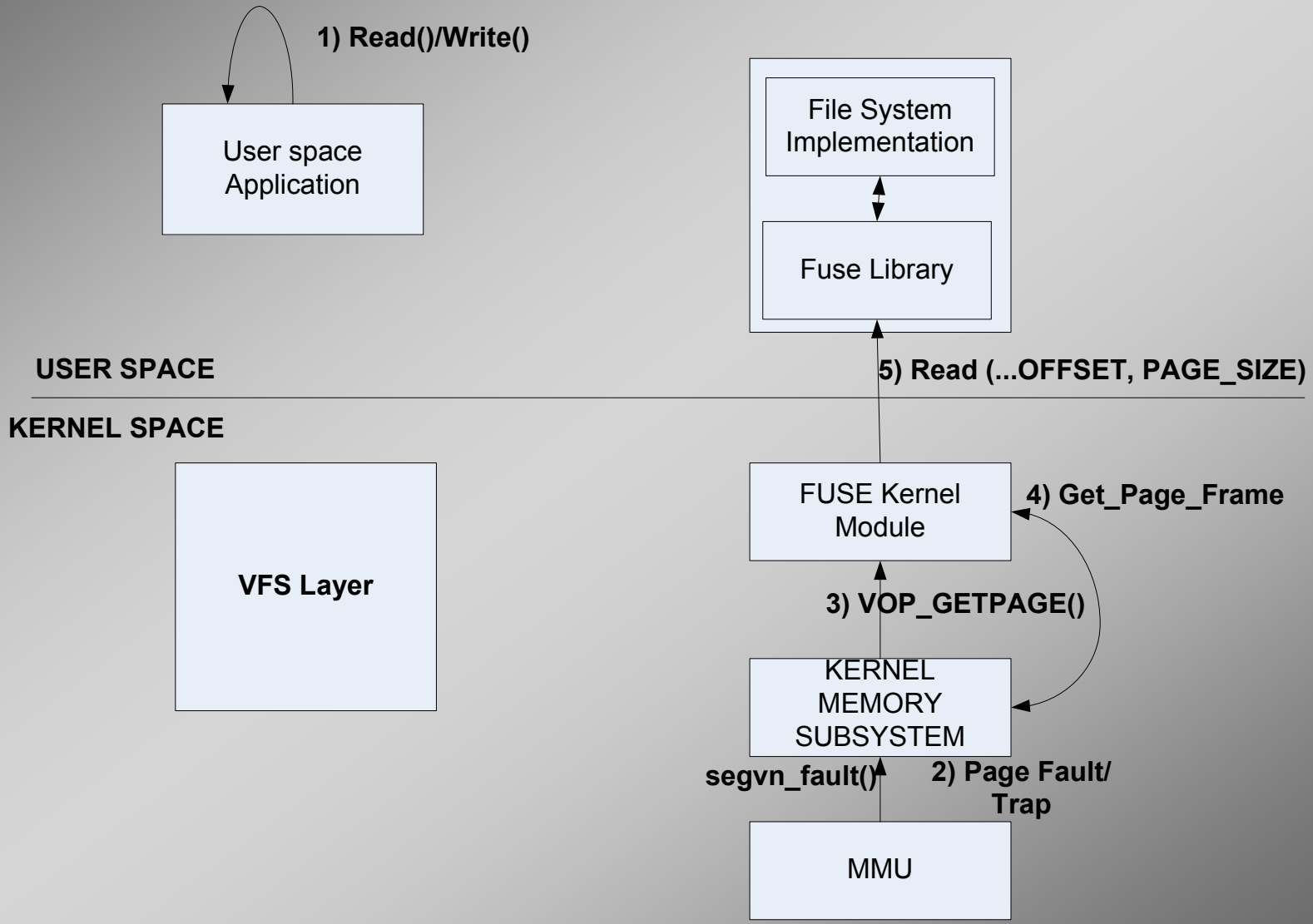
Fuse Mmap (Initial attempt…)

- File system can be mapped to a process address space using mmap() system call.
- Once a mapping is established, read/write of memory corresponds to file data.
- This is performed transparently by the kernel using file system support.

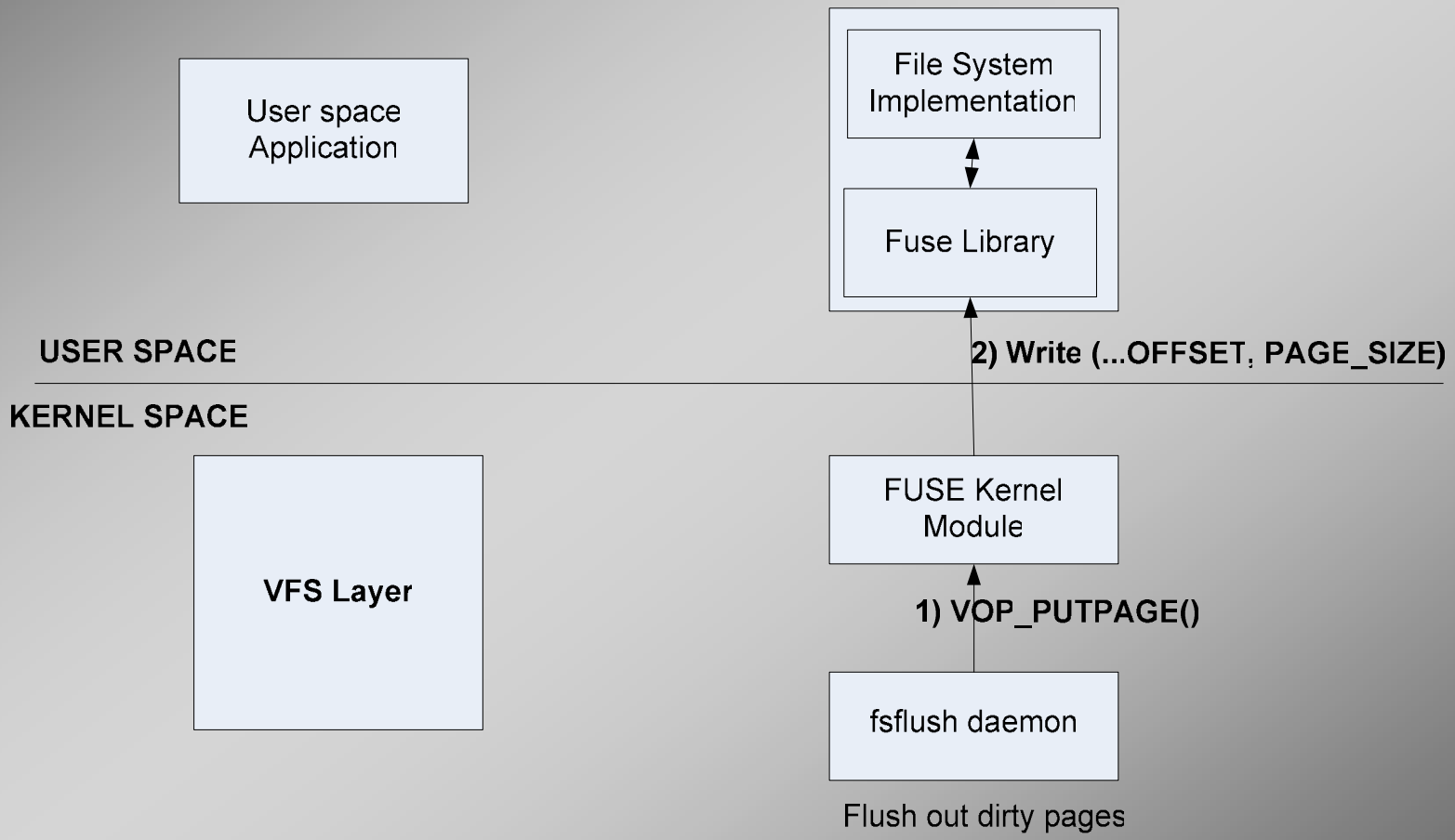
Solaris Fuse mmap() contd…



Solaris Fuse mmap() contd…



Solaris Fuse mmap() contd…



Fuse Project on opensolaris

- http://opensolaris.org/os/project/fuse/

- Code base hosted through mercurial
- Download using:
  - hg clone ssh://anon@hg.opensolaris.org/hg/fuse/libfuse
  - hg clone ssh://anon@hg.opensolaris.org/hg/fuse/fusefs

So why use Fuse?

Advantages:

- Flexibility – easy to code in user space than kernel space.
- File system implementation not tied to any particular OS.
- Doesn’t require Kernel recompile.
- Non-privileged mounts possible.

Disadvantage:

- Performance - redundant data copy, many user space/kernel space switches

Related Work at U of M

- CoreFS:
  - basic network file system built on top of FUSE.
  - http://www.cs.umn.edu/research/sclab/coreFS.html
- Group Key Management for Secure Global File Sharing over CoreFS
  - Provides end to end encryption and makes cryptographic operations transparent to users.

Questions?

Email: subram@cs.umn.edu