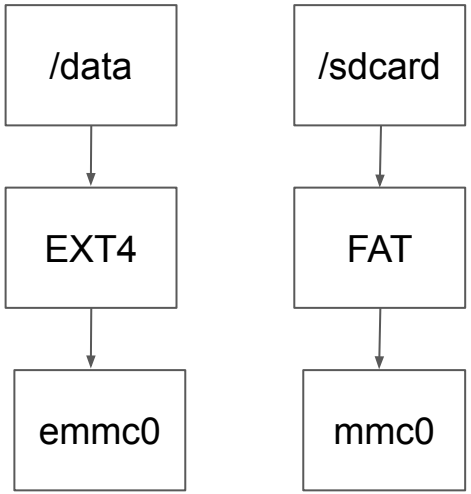
Eliminating WrapFS hackery in Android with ExtFUSE

Ashish Bijlani
PhD student, Georgia Tech

Talk title adopted from a related LWN article: https://lwn.net/Articles/718640/

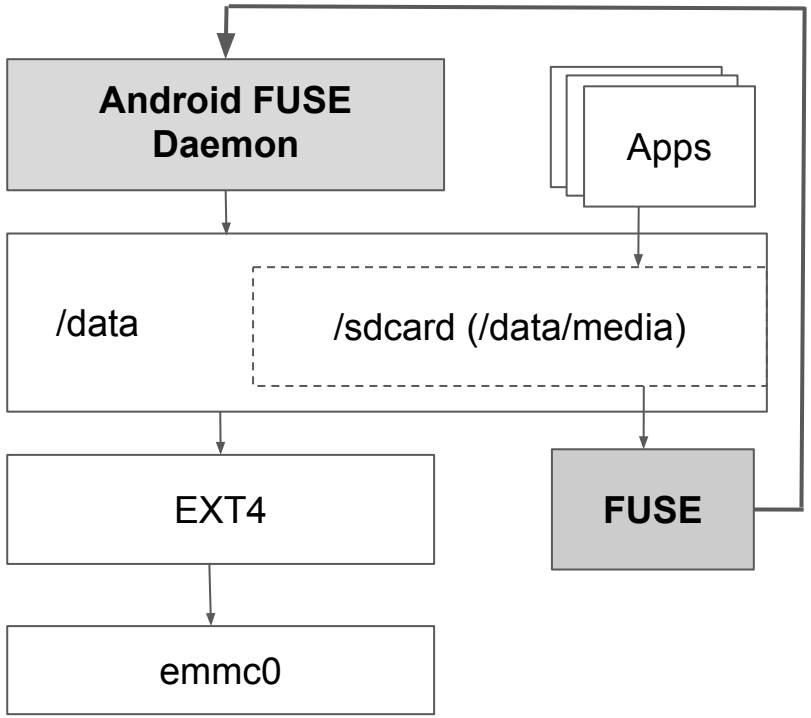
Some ~~Millions and millions of~~ years ago...

- Android devices had micro SD card slots
- Mounted on ‘/sdcard’ with FAT
- Hosts OBB files, pics, videos, etc.

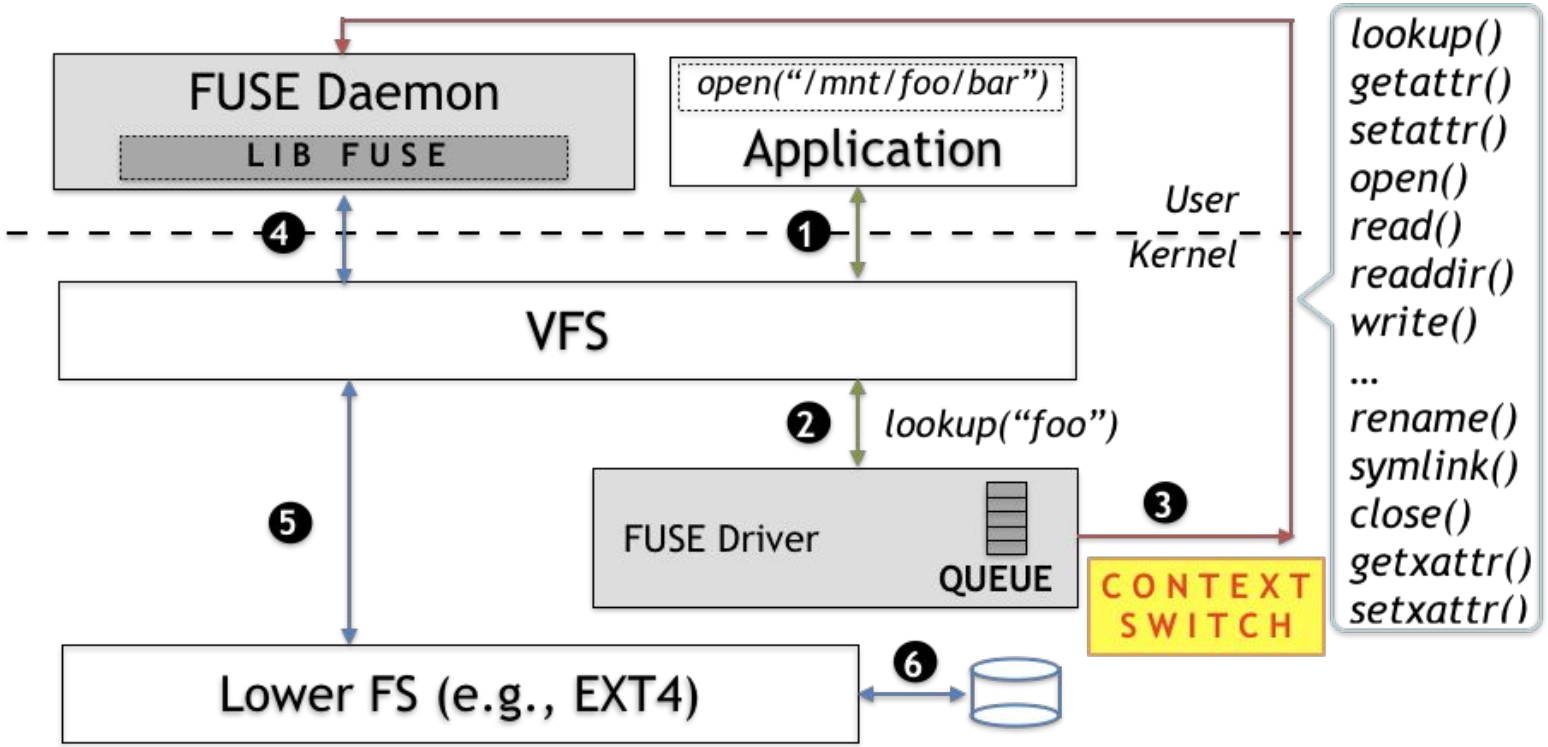


FUSE Emulated SD Card

- Mount ‘/data/media’ as ‘/sdcard’
- Manage ‘/sdcard’ using **FUSE**
  - FAT emulation
  - Custom perm checks
- **User space FS**
  - Use third-party libraries
  - Easy to debug/maintain
  - High performance overhead



File systems in User space (FUSE)

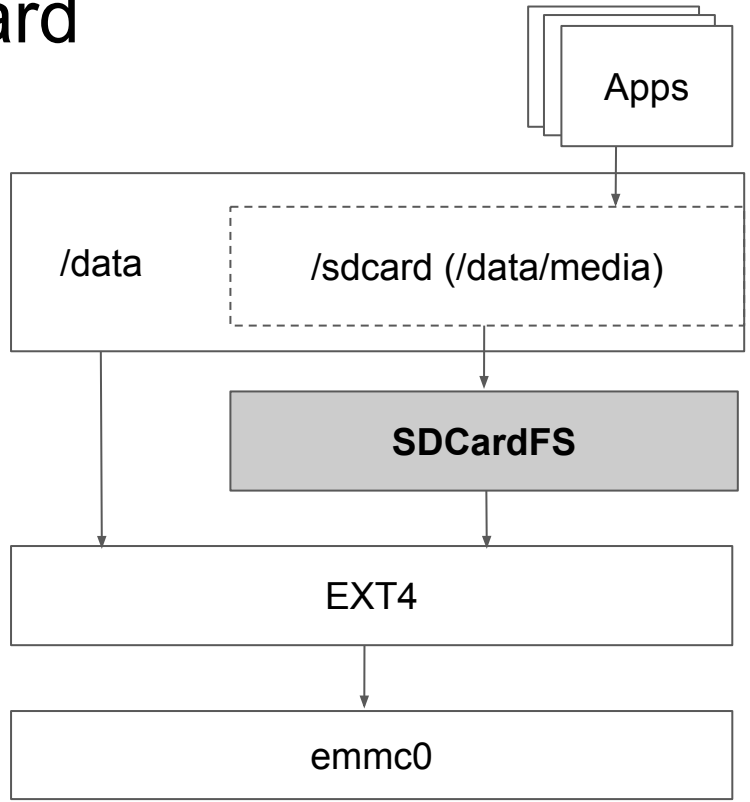


SDCardFS Emulated SD Card

- Manage ‘/sdcard’ w/ **SDCardFS**

- **In-Kernel FS**
  - Based on WrapFS
  - Stackable functionality

- **Disadvantages**
  - Cannot use existing third-party libs
  - Debug, out-of-tree maintenance
  - Bloated TCB

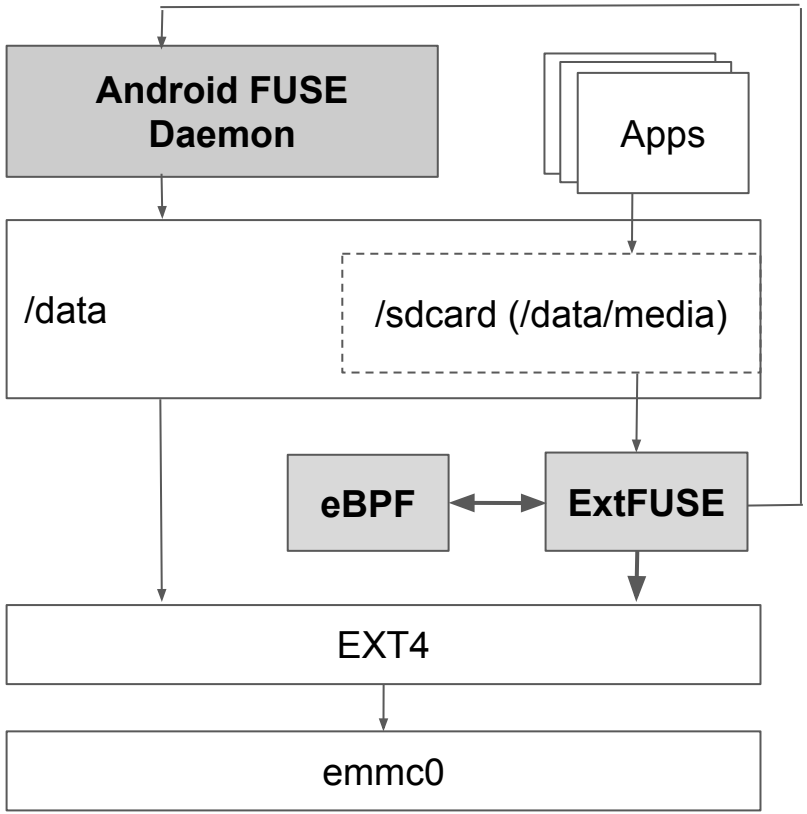


Proposal : ExtFUSE Emulated SD Card

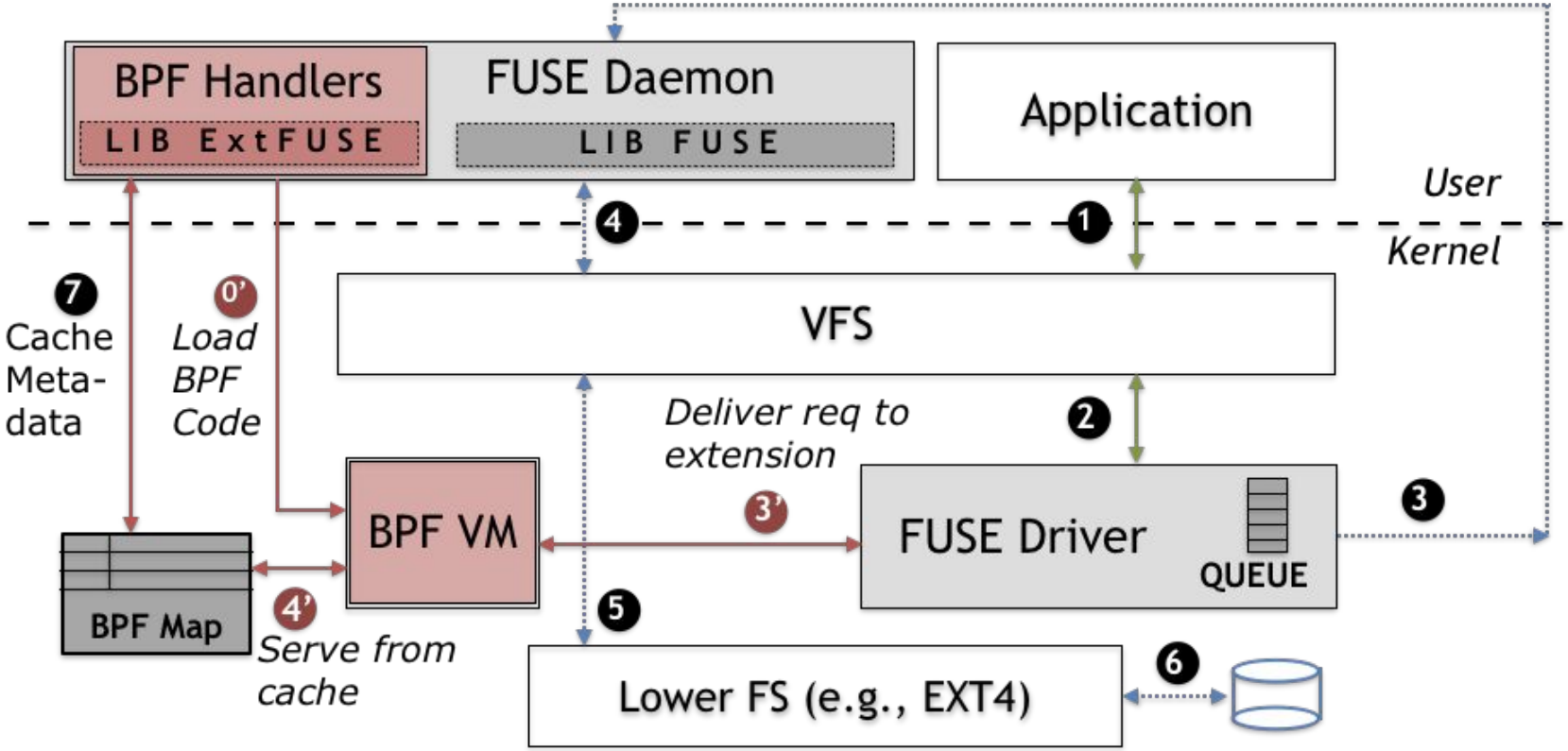
- Manage ‘/sdcard’ w/ **ExtFUSE**

- Complex FS handlers in user space
- eBPF-based “thin” handlers in krnl

- **Advantages**
  - Near-native performance
  - Better system reliability
  - Easy to debug and maintain



ExtFUSE: Extension Framework for FUSE



ExtFUSE: Direct I/O Passthrough Example

```
void handle_open(fuse_req_t req, fuse_ino_t ino,
        const struct fuse_open_in *in) {
    /* file represented by @ino inode num */
    struct fuse_open_out out; char path[PATH_MAX];
    int len, fd = open_file(ino, in->flags, path, &out);
    if (fd > 0){
+       /* install fd in inode map for pasthru */
+       imap_key_t key = out->fh;
+       imap_val_t val = fd; /* lower fd */
+       extfuse_insert_imap(&key, &val);
}   }
```

ExtFUSE: Direct I/O Passthrough Example

```
int read_extension(extfuse_req_t req, fuse_ino_t ino,
        const struct fuse_read_in *in) {
    /* lookup in inode map, passthrough if exists */
    imap_key_t key = in->fh;
    if (!extfuse_lookup_imap(&key)) return UPCALL;


    return PASSTHRU; /* forward req to lower FS */
}
```

Direct I/O passthrough performance

1GB RAM HiKey 620 board, running 4.9 Android kernel.

App Name	OBB Size	Latency (Default)	Latency (Passthru)
Disney Palace Pets 5.1	374 MB	2235 ms	1766 ms
Dead Effect 4	1.1 GB	8895 ms	4579 ms (90% improvement)

All FUSE READ and WRITE requests were directly forwarded to lower EXT4FS, no context switching.

Thank You!

- Related resources
  - Open Source Summit ’18 Slides
  - Linux Plumbers Conference ’18 talk
  - USENIX ATC Paper
  - Project page

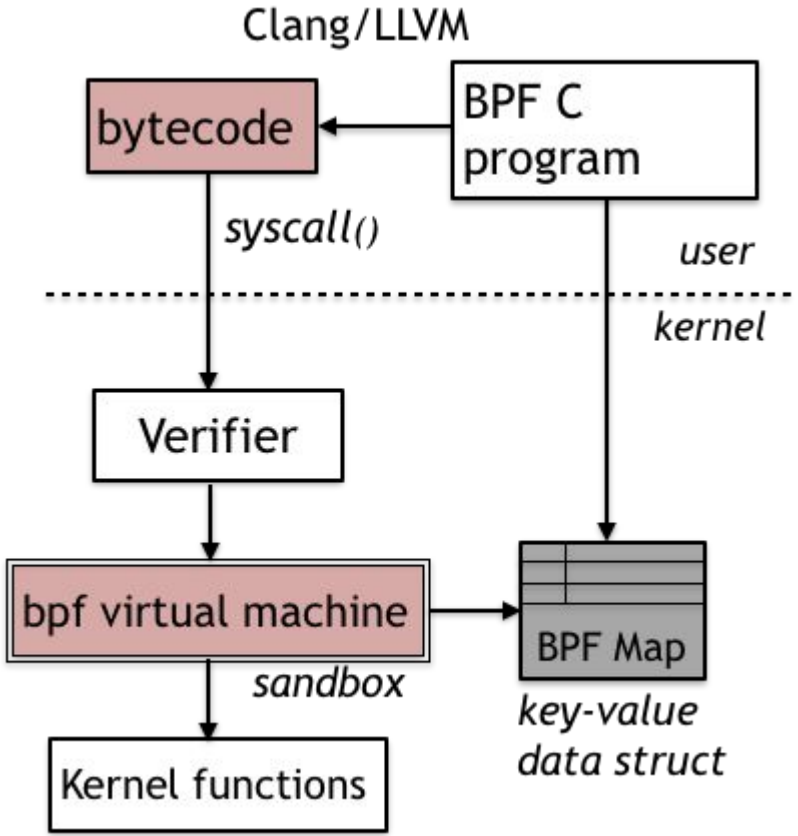
ashish.bijlani@gatech.edu

https://www.linkedin.com/in/ashishbijlani/

Backup Slides

eBPF Overview

- Pseudo machine architecture
- Evolved as a generic kernel extension framework
- Code is verified, loaded into kernel, executed under virtual machine runtime
- Shared BPF maps with user space



ExtFUSE: Custom Perm Checks Example

```
bool check_caller_access_to_name(int64_t key, const char *name) {
    /* define a shmap for hosting permissions */
    int *val = extfuse_lookup_shmap(&key);
    /* Always block security-sensitive files at root */
    if (!val || *val == PERM_ROOT) return false;
    /* special reserved files */
    if (!strncasecmp(name, "autorun.inf", 11) ||
        !strncasecmp(name, ".android_secure", 15) ||
        !strncasecmp(name, "android_secure", 14))
        return false;
    return true;
}
```

Similarly, we can enforce custom RO/RW perms in krnl.

ExtFUSE: Metadata Cache Example

```
// getattr() kernel extension - cache attrs
int getattr(struct extfuse_args *args) {
  u32 key = bpf_extfuse_read(args, PARAM0);
  u64 *val = bpf_map_lookup_elem(map, &key);
  if (val) bpf_extfuse_write(args, PARAM0, val);
}

// setattr() kernel extension - invalidate attrs
int setattr(struct extfuse_args *args) {
  u32 key = bpf_extfuse_read(args, PARAM0);
  if (val) bpf_map_delete_elem(map, &key);
}
```

We can cache READDIR, GETATTR, GETXATTR, SYMLINKS, and LOOKUP replies in kernel.