PORTING THE HAMMER FILE SYSTEM TO LINUX

June 10, 2009

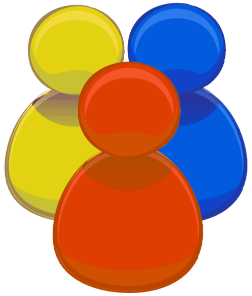
Daniel Lorch

Outline

1. Motivation
2. A Hammer File System Walkthrough
3. Tool Evaluation
4. Porting Work
5. Demo
6. Questions

1. Motivation

Number of DragonFly BSD Users



„A few“

Number of Linux Users

„Millions“

more users ⇨ more peer reviewers

2. Hammer File System Walkthrough (1/3)

- □ Fine Grained History Retention

```
# echo Hello > test
# echo World >> test
# hammer history test
test 000000010061aac0 clean {
00000001007a1520 23-Mar-2009 20:04:11
00000001007a1580 23-Mar-2009 20:04:43
}
# cat test@@0x00000001007a1520
Hello
# cat test@@0x00000001007a1580
Hello
World
```

2. Hammer File System Walkthrough (2/3)

- File System Snapshots
  - Same mechanism as for files: Append transaction id to directory name
  - „hammer snapshot“ command conveniently creates these softlinks

```
# hammer snapshot /mnt /mnt/snap
/mnt/snap
# ls -l snap
lrwxr-xr-x 1 root wheel 25 Mar 23 20:07 snap -> /mnt/
@@0x00000001007a15c0
# ls snap/
test
```

2. Hammer File System Walkthrough (3/3)

- Master-Slave Replication
  - Single Master, Multiple Slaves
  - Pairing via unique „uuid“
  - Replication initiated manually with „hammer mirror-copy“ command; peers can be remote (via SSH)
  - Incremental mirroring: since transaction ids are strictly incremental, only need to negotiate on range to transmit

3. Tool Evaluation (1/2)

□ „So, you want to write a kernel module. You know C, you've written a few normal programs to run as processes, and now you want to get to where the real action is, to where a single wild pointer can wipe out your file system and a core dump means a reboot.”

– Peter Jay Salzman, The Linux Kernel Module Programming Guide

⇨ decided to use a virtualization software

3. Tool Evaluation (2/2)

- Tried VMWare with „guest debug monitor feature“, but couldn‘t load debug symbols on my Mac
- Tried User-Mode-Linux ⇨ Good!
  - It’s even part of the standard Linux kernel

```
./vmlinux ubda=../Slackware-12.2-root_fs
ubdb=../hammerdisk.raw
```

- Tried DragonFly‘s vKernels ⇨ Good!
  - It’s even part of standard DragonFly BSD

```
./kernel -m 64m -r ../rootimg.01 -r /home/
hammerdisk.raw –n1
```

4. Porting Work (1/3)

- „Cowboy-style“ programming:
  1. Add a source file
  2. Fix errors
  3. Goto 1
- Looked at this screen for weeks:

```
$ make ARCH=um 2>&1 | grep 'error: ' | sed -e 's/.*error: //g' | sort
| uniq
'EFTYPE' undeclared (first use in this function)
'FREAD' undeclared (first use in this function)
'FSCRED' undeclared (first use in this function)
'FWRITE' undeclared (first use in this function)
'LK_EXCLUSIVE' undeclared (first use in this function)
...
```

4. Porting Work (2/3)

- Idea: Wrapper Files
  - For compiler errors, add missing definition to dfly_wrap.h, then used it like this:

```
#include "dfly_wrap.h"
#include "dfly/vfs/hammer/hammer_prune.c"
```

  - For linker errors, add stub function to dfly_wrap.c causing kernel panic:

```
int nlookup(struct nlookupdata *nd) {
    panic("nlookup");
}
```

- Result:
  - 14 out of 18 source files re-used without modification

4. Porting Work (3/3)

- Finally, executed kernel and fixed kernel panics, one after the other
- Some stubs are still there..
  - ..don‘t run file system on a live system (yet)

Demo

Questions?