LA-UR-21-27920

SquashFS & FUSE for Better HPC Containers

HPC Showcase 2021

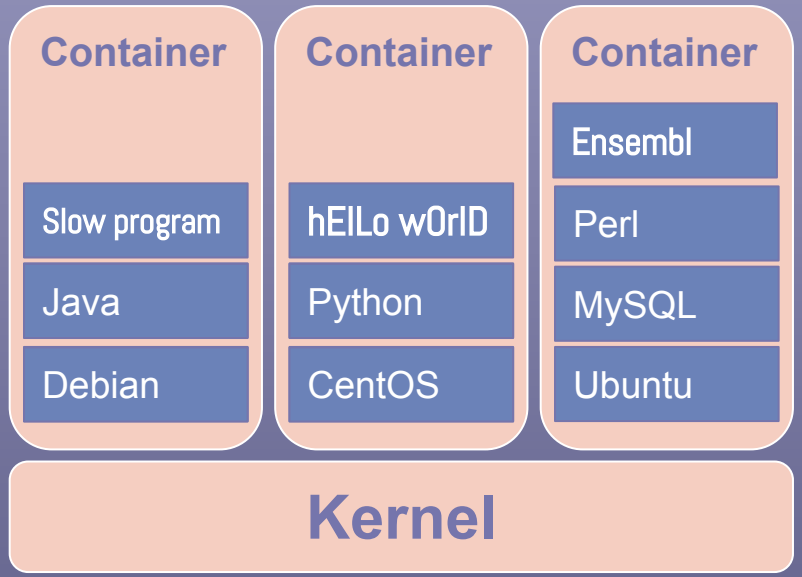
Megan Phinney
Iowa State University
BS Computer Engineering 2022

Anna Chernikov
NC State University
BS Computer Science 2020
University of Arizona
PhD Student

Los Alamos NATIONAL LABORATORY EST. 1943

Containers in HPC

- What are Containers?
  - Contains Application, Software Stack, and OS
  - Can be moved between different machines
- Why Containers in HPC?
  - Hides Complex Dependencies
  - Lightweight
  - Portable
  - Easy Deployment
  - Isolated Environment



What is Charliecloud?

- A Container Runtime developed at LANL specifically for HPC

Why Charliecloud?

- Light-weight
- Fully Unprivileged
- Better choice for HPC

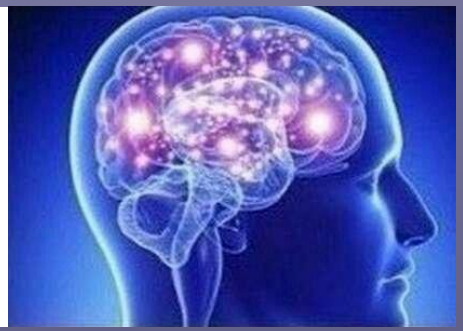
Create a more user-friendly SquashFS workflow for Charliecloud

Typical Tarball Workflow

	Unpack	Mount	Run	Unmount
Tarball Workflow	`ch-tar2dir`		`ch-run`	

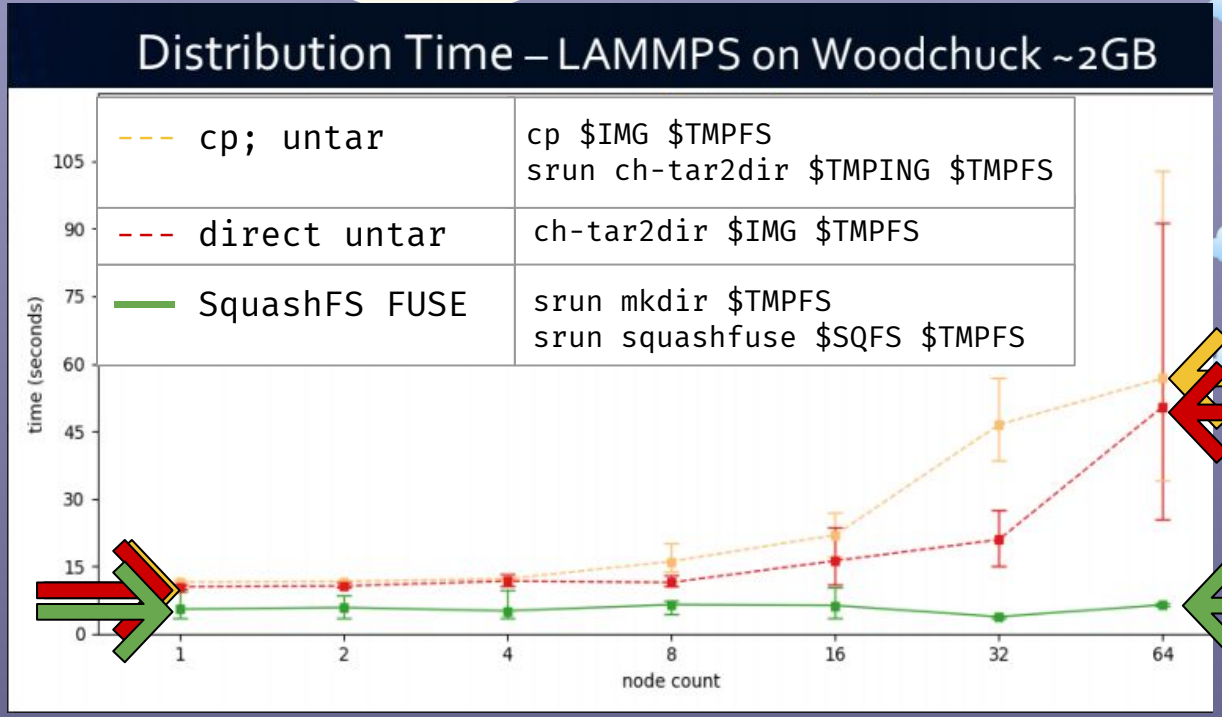
Too Slow Distribution time

Takes too much memory



Why Squash?

- Faster image distribution times
  - time for images to unpack on nodes
- SquashFS scales better than tarballs
- Better choice for HPC



Anaya, Cutshaw, Goff; “Evaluating Container Image Distribution Methods for HPC Using Charliecloud”, Supercomputer Institute HPC Showcase 2018

What is a SquashFS File?

- Compressed read-only filesystem
- Like tarball but mountable
- SquashFUSE enables mounting by unprivileged users

FUSE: Filesystem in Userspace

High Level

Low Level

Typical SquashFS Workflow

	Unpack	Mount	Run	Unmount
Old SquashFS Workflow		`ch-mount`	`ch-run`	`ch-umount`

No automatic clean up mechanism

Doesn't play well with srun (Slurm)

3 User Commands

Our New SquashFS Workflow

	Unpack	Mount	Run	Unmount
New SquashFS Workflow		ch-run		

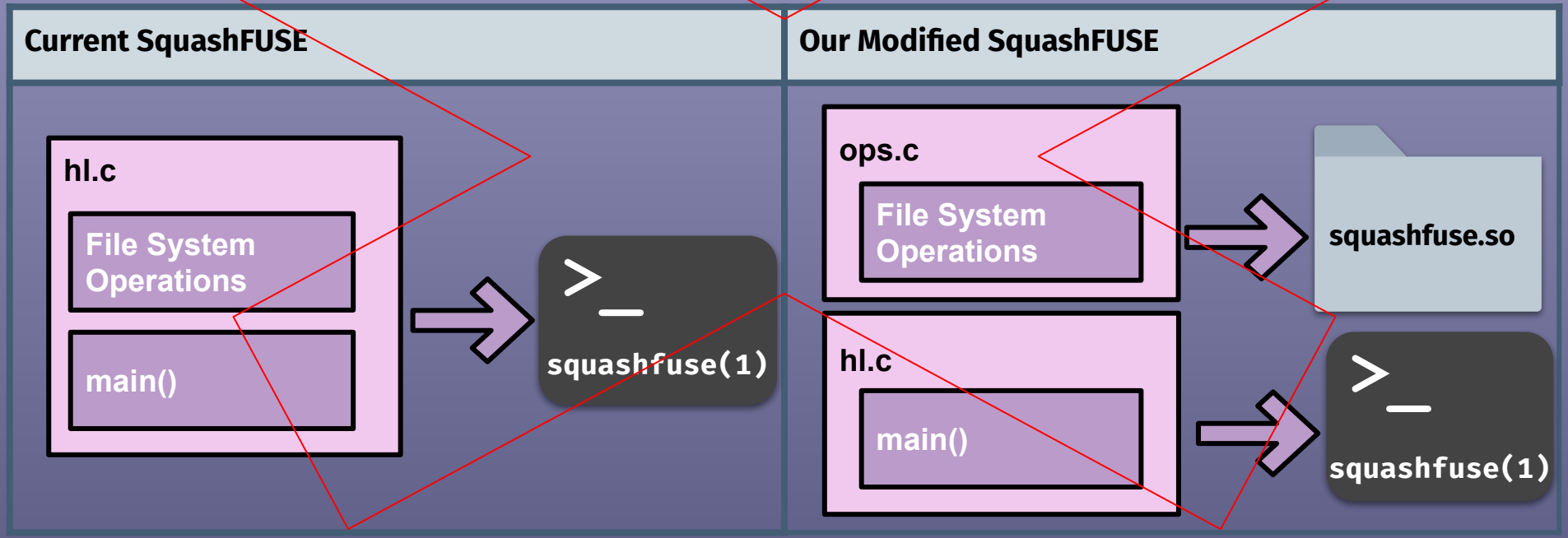
Automatically cleans up user mounts

Works with srun

Only 1 User Command

Moved SquashFUSE File System Operations to Shared Library

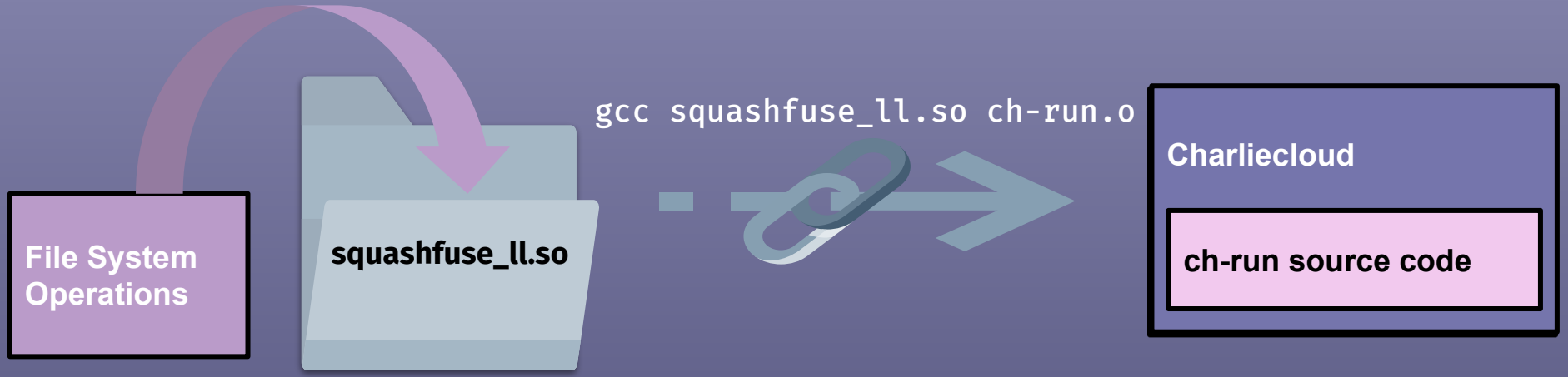
SquashFUSE file system operations are made accessible to ch-run via our new shared library

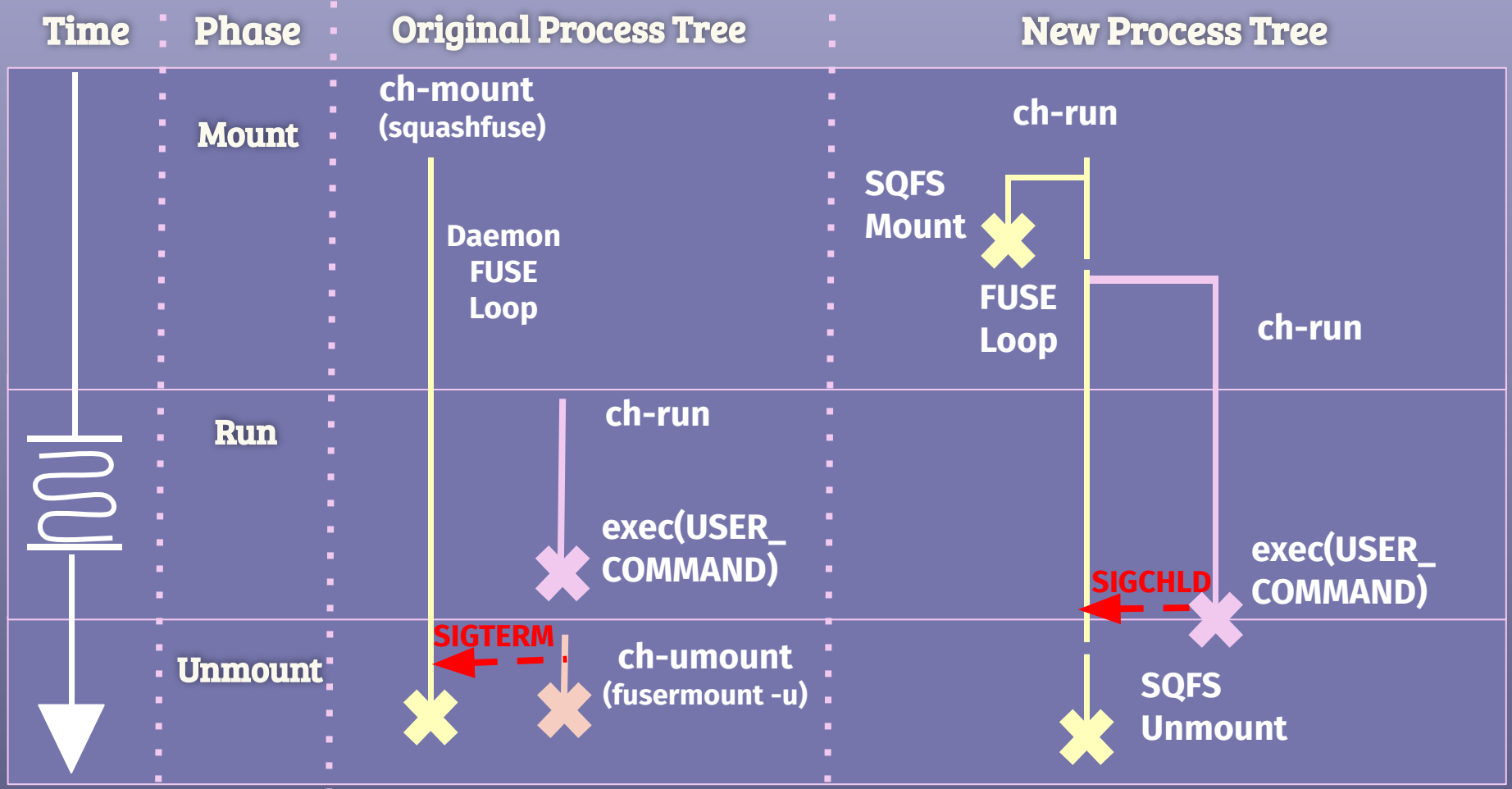


Linked SquashFUSE Libraries to ch-run

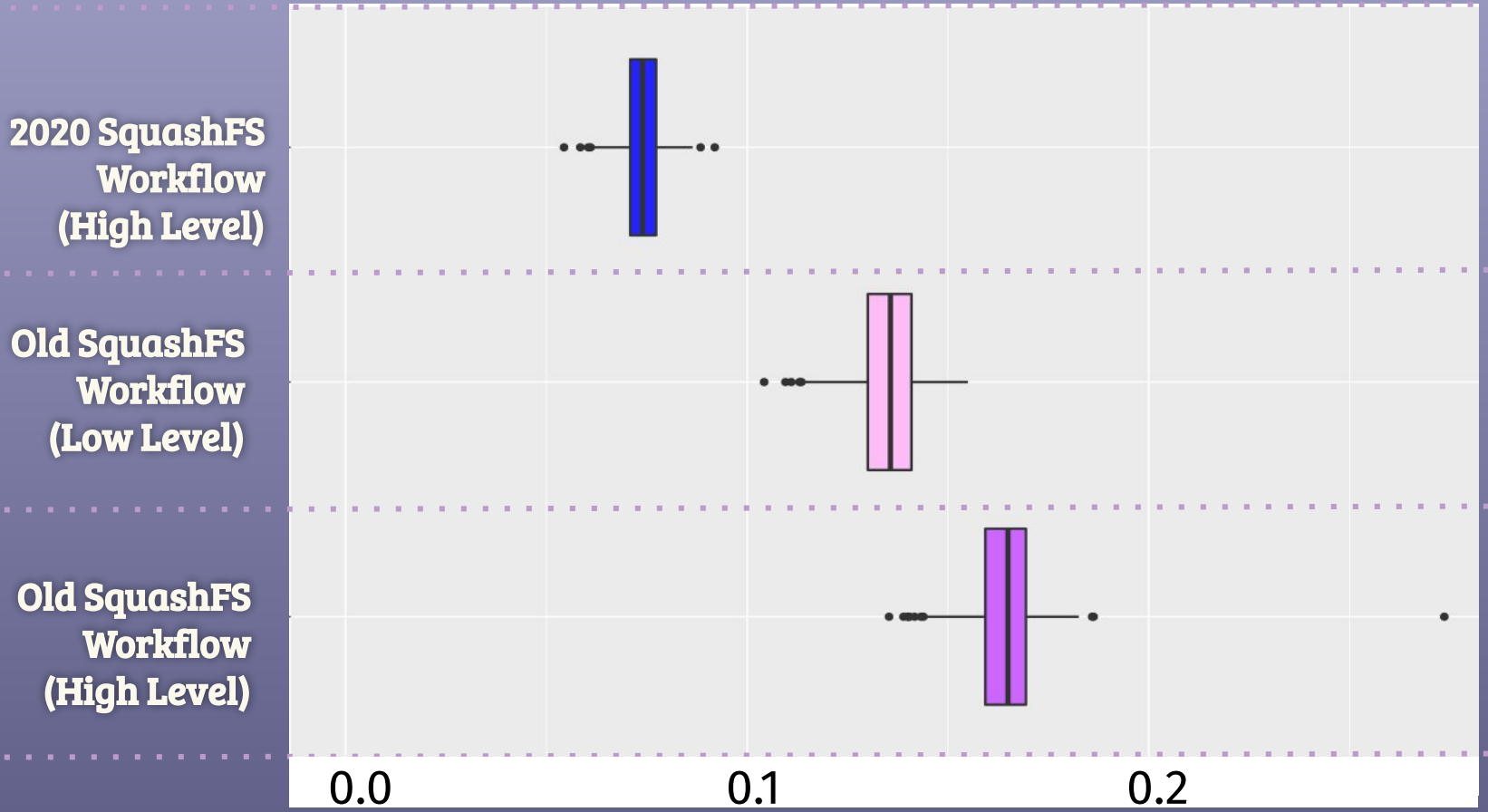
All Fuse File System operations in ch-run are referenced from SquashFUSE libraries:

- **Mount**
- **Unmount**
- **Reads**




Time
Phase
Original Process Tree
New Process Tree
Mount
Run
Unmount
ch-mount
(squashfuse)
Daemon
FUSE
Loop
ch-run
exec(USER_
COMMAND)
SIGTERM
ch-umount
(fusermount -u)
ch-run
SQFS
Mount
FUSE
Loop
ch-run
SIGCHLD
exec(USER_
COMMAND)
SQFS
Unmount

Total Workflow Time



Demo

Tarball Output

```
$ ch-tar2dir /var/tmp/hello.tar.gz /var/tmp

creating new image /var/tmp/hello
/var/tmp/hello unpack ok

$ ch-run /var/tmp/hello -- echo hello
hello
```

User Output

```
$ ch-run /var/tmp/tar/00_tiny.sqfs -- echo hello
hello
```

```
$ ch-run -vv /var/tmp/tar/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
ch-run[24720]: verbosity: 3 (ch-run.c:183)
ch-run[24720]: newroot: /var/tmp/vm-user.ch/mnt (ch-run.c:184)
ch-run[24720]: container uid: 1000 (ch-run.c:185)
ch-run[24720]: container gid: 1000 (ch-run.c:186)
ch-run[24720]: join: 0 0 (null) 0 (ch-run.c:187)
ch-run[24720]: private /tmp: 0 (ch-run.c:189)
setup_namespaces 405: uids=1000,1000,1000, gids=1000,1000,1000 + 4,24,27,30,46,108,1000
setup_namespaces 407: uids=65534,65534,65534, gids=65534,65534,65534 +
65534,65534,65534,65534,65534,65534,65534
setup_namespaces 422: uids=1000,1000,1000, gids=65534,65534,65534 +
65534,65534,65534,65534,65534,65534,65534
setup_namespaces 430: uids=1000,1000,1000, gids=1000,1000,1000 + 65534,65534,65534,65534,65534,65534,1000
enter_udss 168: uids=1000,1000,1000, gids=1000,1000,1000 + 65534,65534,65534,65534,65534,65534,1000
run_user_command 362: uids=1000,1000,1000, gids=1000,1000,1000 + 65534,65534,65534,65534,65534,65534,1000
ch-run[24756]: exec: "echo" "hello"
hello
ch-run[24720]: unmounting: /var/tmp/vm-user.ch/mnt (ch_fuse.c:77)
```

```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
```

1. Is it a directory?
   a. Run original workflow
2. Is it a file?
3. Can we open and read the file?
4. What is the magic number?

Magic Number:

- First 4 bytes in a file
- Used to identify what type of file it is

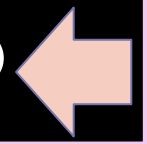
```
hexdump -n 48 -C /var/tmp/tar/00_tiny.sqfs

00000000  68 73 71 73 f2 01 00 00  4f 9c 12 61 00 00 02 00
00000010  06 00 00 00 01 00 11 00  c0 00 01 00 04 00 00 00
00000020  56 05 bb 0a 00 00 00 00  ec 10 29 00 00 00 00 00
```

```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
```

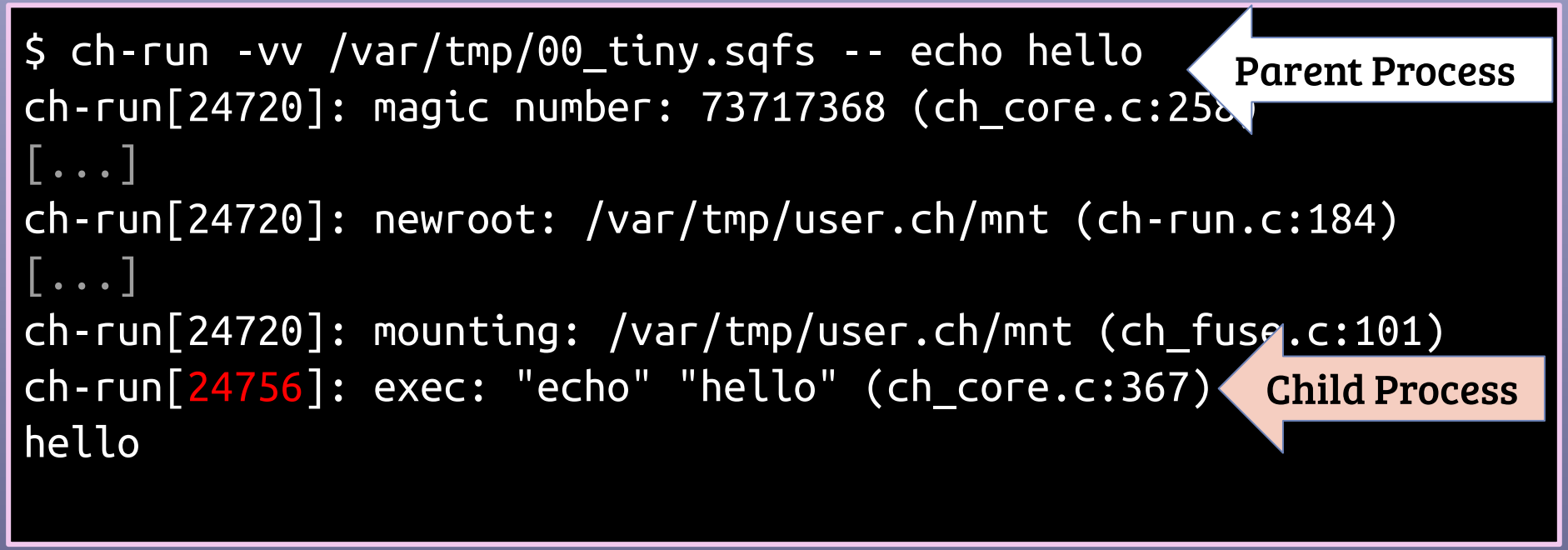
Newroot:

- Usually empty directory
- Where the image is mounted
- Where the user command will run

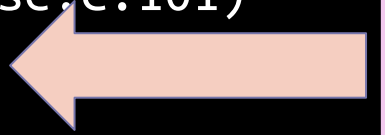
```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
[...]
ch-run[24720]: mounting: /var/tmp/user.ch/mnt (ch_fuse.c:101)
```



```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
[...]
ch-run[24720]: mounting: /var/tmp/user.ch/mnt (ch_fuse.c:101)
ch-run[24756]: exec: "echo" "hello" (ch_core.c:367)
hello
```


$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
[...]
ch-run[24720]: mounting: /var/tmp/user.ch/mnt (ch_fuse.c:101)
ch-run[24756]: exec: "echo" "hello" (ch_core.c:367)
hello
Parent Process
Child Process

```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
[...]
ch-run[24720]: mounting: /var/tmp/user.ch/mnt (ch_fuse.c:101)
ch-run[24756]: exec: "echo" "hello" (ch_core.c:367)
hello
```



```
$ ch-run -vv /var/tmp/00_tiny.sqfs -- echo hello
ch-run[24720]: magic number: 73717368 (ch_core.c:258)
[...]
ch-run[24720]: newroot: /var/tmp/user.ch/mnt (ch-run.c:184)
[...]
ch-run[24720]: mounting: /var/tmp/user.ch/mnt (ch_fuse.c:101)
ch-run[24756]: exec: "echo" "hello" (ch_core.c:367)
hello
ch-run[24720]: unmounting: /var/tmp/user.ch/mnt (ch_fuse.c:69)
```

1. End FUSE loop
2. Remove FUSE signal handlers
3. Deallocate data structures
4. Unmount SquashFS

Building Charliecloud with and without SquashFUSE

Configure checks if you have the right version of SquashFUSE and FUSE3

```
$ ./configure
checking for fuse_set_signal_handlers in -lfuse3... yes
checking for sqfs_ll_mount in -lsquashfuse_ll... yes
checking for ll.h... yes
[...]
```

```
use sqfs workflow: yes
    fuse3 ... yes
    squashfuse ... yes
    ll.h ... yes
```

Changes to configure.ac using Autotools

```
AC_CHECK_LIB([fuse3], [fuse_set_signal_handlers],
[CH_RUN_LIBS+=' -lfuse3'; have_fuse=yes], [have_fuse=no])

AC_CHECK_LIB([squashfuse_ll], [sqfs_ll_mount], [CH_RUN_LIBS+='
-lsquashfuse_ll'; have_sqfuse=yes], [have_sqfuse=no])
```

Makefile

```
if HAVE_SQFUSE
   ch_run_SOURCES += ch_fuse.h ch_fuse.c
endif
```

ch-run.c

```
#ifdef RUN_SQ
   /* sqfs code */
#endif
```

Conclusions

Our New SquashFS Workflow:

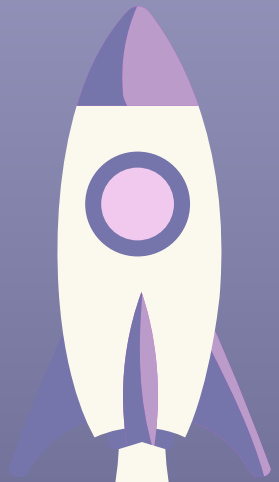
- More user-friendly
- No additional performance cost
- Plays well with srun
- Auto-cleans SquashFS mounts

What I've Done

- Use low-level FUSE API
- Convert prototype to be ready to be used in production

What's Next?

- Merge branch into Charliecloud



Our Team

Megan Phinney

Iowa State University
Computer Engineering
mphinney@iastate.edu

Anna Chernikov

NC State , University of Arizona
Computer Science
chernikov@email.arizona.edu

Jordan Ogas

Creator of R. Peezee

Alfred Torrez

Grill Dad

Shane Goff

Brofessional

Reid Priedhorsky

King of Charliecloud