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
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

<table>
<thead>
<tr>
<th>Container</th>
<th>Slow program</th>
<th>hElLo w0rld</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Java</td>
<td>Python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Debian</td>
<td>CentOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

hElLo w0rld

Ensembl
Perl
MySQL
Ubuntu
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

<table>
<thead>
<tr>
<th>Old SquashFS Workflow</th>
<th>Unpack</th>
<th>Mount</th>
<th>Run</th>
<th>Unmount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ch-mount</td>
<td>ch-run</td>
<td>ch-umount</td>
</tr>
</tbody>
</table>

- 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

Automatically cleans up user mounts
Works with srun
Only 1 User Command

ch-run
Moved SquashFUSE File System Operations to Shared Library

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

Current SquashFUSE

<table>
<thead>
<tr>
<th>hl.c</th>
<th>main()</th>
</tr>
</thead>
<tbody>
<tr>
<td>File System Operations</td>
<td></td>
</tr>
</tbody>
</table>

Our Modified SquashFUSE

<table>
<thead>
<tr>
<th>ops.c</th>
<th>hl.c</th>
</tr>
</thead>
<tbody>
<tr>
<td>File System Operations</td>
<td>main()</td>
</tr>
</tbody>
</table>

squashfuse.so

>`squashfuse(1)`
Linked SquashFUSE Libraries to ch-run

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

```
gcc squashfuse_ll.so ch-run.o
```

Charliecloud

ch-run source code
Total Workflow Time

- 2020 SquashFS Workflow (High Level)
  - Duration (seconds): 0.0

- Old SquashFS Workflow (Low Level)
  - Duration (seconds): 0.1

- Old SquashFS Workflow (High Level)
  - Duration (seconds): 0.2
Demo
$ 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,
setup_namespaces 422: uids=1000,1000,1000, gids=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)
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

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

```
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
$ 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 (chFuse.c:101)
ch-run[24756]: exec: "echo" "hello" (ch_core.c:367)
hello
SIGCHLD
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
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

Credits: This presentation template was created by Slidesgo, including including icons by Flaticon, and infographics & images by Freepik.