# RCUfs: A Read Copy Update file system

## An experimental method for handling file system locking

Callum Massey *<644475@swan.ac.uk>*

Gregynog — January 2014

# Computer Clusters

Why do they exist?

- Improve performance
- Improve reliability

Why is that relevant?

# Shared Data Access

- Multiple processes accessing same data

# Issues

- What if two processes try to write the same file?
- What if there's a read on a file being written to?
- How do you stop processes seeing garbage data?

# Read Copy Update

Locking mechanism that can drop in replace reader-writer locking.

- Reader threads never got blocked
- Readers always return valid results
- Writers lock as normal
- Huge performance increase to read heavy applications

# How?

So how does it work?

- Original data copied elsewhere
- Changes made to this new version
- Global pointer (journal) updated to point to new location
- New readers/writers now see new version
- after all access finished, old data freed

# RCUfs

RCU used extensively in the linux kernel, especially within the network stack however all current usage is restricted to in memory applications.

No one has tried to use it as a locking mechanism in a clustered file system.

# The Plan

- Build a program that spawns multiple processes reading and writing to a file
- Test this using RCU and with reader-writer locking
- Determine the scenarios where RCU is better (Definitely not all of them)

# Progress

- Research, experimenting, learning C
- Writing byte arrays to files pretending to be block devices
- Stripped out and started reimplementing the locking on a FUSE file system
- Backtracked on writing a FUSE file system

# Work to do

- Expand my simple array writer to spawn multiple processes
- Make the writer method use RCU
- Generate performance metrics from this with RCU and r-w locking
- Draw graphs/interpret results

Any Questions?