rCore with Log-Structured File System

Yiwei Li, Zhiyao Li



Background

□ rCore

- rcore-tutorial
- o rcore-tutorial-OS20
- o rcore-fs

Image: recore-fs-ext2fix warnings4 months agoImage: recore-fs-fuseRevert "add inode id and file type in the return value of get_entry i17 days agoImage: recore-fs-hostfsRevert "add inode id and file type in the return value of get_entry i17 days agoImage: recore-fs-mountfsimplement get_entry_with_data for MNode17 days agoImage: recore-fs-ramfsfix lock_multiple in ramfs14 days agoImage: recore-fs-sefsRevert "add inode id and file type in the return value of get_entry i17 days agoImage: recore-fs-sefsRun cargo fmt17 days agoImage: recore-fs-ucorefix warnings4 months agoImage: recore-fsAdd default impl for get_entry_with_metadata17 days agoImage: recore-fsadd simple DevFS10 months ago	rcore-fs-devfs	Revert "add inode id and file type in the return value of get_entry i	17 days ago
rcore-fs-hostfs Revert "add inode id and file type in the return value of get_entry i 17 days ago rcore-fs-mountfs implement get_entry_with_data for MNode 17 days ago rcore-fs-ramfs fix lock_multiple in ramfs 14 days ago rcore-fs-sefs Revert "add inode id and file type in the return value of get_entry i 17 days ago rcore-fs-sefs Run cargo fmt 17 days ago rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-ext2	fix warnings	4 months ago
rcore-fs-mountfs implement get_entry_with_data for MNode 17 days ago rcore-fs-ramfs fix lock_multiple in ramfs 14 days ago rcore-fs-sefs Revert "add inode id and file type in the return value of get_entry i 17 days ago rcore-fs-sefs Run cargo fmt 17 days ago rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-fuse	Revert "add inode id and file type in the return value of get_entry i	17 days ago
rcore-fs-ramfs fix lock_multiple in ramfs 14 days ago rcore-fs-sefs Revert "add inode id and file type in the return value of get_entry i 17 days ago rcore-fs-sfs Run cargo fmt 17 days ago rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-hostfs	Revert "add inode id and file type in the return value of get_entry i	17 days ago
rcore-fs-sefs Revert "add inode id and file type in the return value of get_entry i 17 days ago rcore-fs-sefs Run cargo fmt 17 days ago rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-mountfs	implement get_entry_with_data for MNode	17 days ago
rcore-fs-sfs Run cargo fmt 17 days ago rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-ramfs	fix lock_multiple in ramfs	14 days ago
rcore-fs-ucore fix warnings 4 months ago rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-sefs	Revert "add inode id and file type in the return value of get_entry i	17 days ago
rcore-fs Add default impl for get_entry_with_metadata 17 days ago	rcore-fs-sfs	Run cargo fmt	17 days ago
	rcore-fs-ucore	fix warnings	4 months ago
sefs-fuse add simple DevFS 10 months ago	rcore-fs	Add default impl for get_entry_with_metadata	17 days ago
	sefs-fuse	add simple DevFS	10 months ago



Background

Log-Structured File System (LFS)

- Motivation
 - Growing system memory capacity
 - Large gap between random access and sequential access

Traditional FFS-like system

- Centralized inode map, free data bitmap
- Performance limited by numerous short, random access
- Partly amortized by disk cache
 - New problems: Rigid cache structure, higher energy cost, ...
- No safety guarantees

LFS Introduction

Authors

- Co-founder of VMWare
- Raft protocol, Tcl script language

The Design and Implementation of a Log-Structured File System

Mendel Rosenblum and John K. Ousterhout

Electrical Engineering and Computer Sciences, Computer Science Division
University of California
Berkeley, CA 94720
mendel@sprite.berkeley.edu, ouster@sprite.berkeley.edu

Key ideas

- Decentralizing inode meta -> imap
- Coarse-grained synchronization -> Segment cache
- Mostly appending logs to disk -> Few random accesses
- Crash recovery provided by logs -> Roll forward process

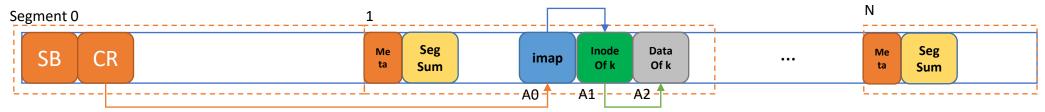
Schedule

What we have done

- Implementing a basic rust OS (rCore-tutorial)
 - Familiar with rust programming
- Integrating Driver and BlockDevice (rCore-20)
 - Disk image mounted and accessed by QEMU
- Implementing a LFS-like system
 - File/Directory creating, accessing, modifying
 - Garbage collection
 - 1100+ lines of rust code
- Comparison with FFS-like system
 - A HDD model
 - SimpleSSD simulation

LFS Structures

 We have implemented most structures in LFS, to construct our new alternative file system on rCore.



- Super block (SB): meta data for entire disk partition
 - Total_blocks, total_segments, unused_blocks, ...
- Checkpoint region (CR): meta data for one checkpoint (for crash recovering)
 - begin_segment_id: u32, current_segment_id: u32, next_ino: u32
- Segment Summary (SegSum): recording liveness of each datablock (for garbage collection)
 - blk_id -> (ino_id, dataentry_id)
- Imap table
 - ino_id -> blk_id

Boundary of memory and storage

- In original LFS paper
 - Segment cache

- □ In our LFS design
 - Synchronized in drop implementation
 - Extending the Dirty struct provided by rcore-fs
 - Dirty() -> writeback is needed
 - Set by DerefMut every time
 - Sync() -> removing dirty bit, for assertion
 - Set manually
 - Stale() -> Current in-memory structure should be allocated a new disk block
 - Set manually, to balance the writeback amount and sequential access

Result

Basic FS operations

- Randomly generating files/directories
- Full system test
- Garbage collection
 - Consistently creates and delete files until exceeding capacity

```
disk
  — dir0
                                       — dir0
       dir32
                                            — dir32
           - dir39
                                                       └─ dir87
                                                     — dir92
                                                   dir45
                                                          - dir52
        file46
                                              - file46
        file70
                                             - file70
       - file80
                                           └─ file80
    file12
                                         - file12
    file17
                                          file17
    file25
                                          · file25
    file3
                                         - file3
    file48
                                          - file48
    file6
                                         file6
                                       └─ file7
  file7
56 directories, 44 files
                                      56 directories, 44 files
```

```
.. test96

INFO:<unknown>: add inode 98 -> 2359 segid 2
enter garbage collecting...
seg 3 unused
.
...
test97

INFO:<unknown>: add inode 99 -> 3134 segid 3
.
...
test98

INFO:<unknown>: add inode 100 -> 3904 segid 3
enter garbage collecting...
seg 2 unused
.
...
test99

test FS done
Image resized.
```

```
mod memory initialized
mod interrupt initialized
mod driver initialized
Loading a LFS disk
content of this directory:
test
write
notebook
read
user shell
temp\overline{1}11
writecreate
read2
hello world
mod fs initialized
Task1: write to a new file
created "temp123"
ready to write file temp123
write to file 'temp123' successfully...
read from file 'temp123' successfully...
content = Hello world!
Thread 1 exit with code 0
Task2: write to a existing file
ready to write file temp111
write to file 'temp111' successfully...
read from file 'temp111' successfully...
content = Hello world again!
Thread 2 exit with code 0
Task3: drop inode and reopen file 1
ready to read file temp123
content = Hello world!
read from file 'temp123' successfully...
Thread 3 exit with code 0
Task4: drop inode and reopen file 2
ready to read file temp111
content = Hello world again!
read from file 'temp111' successfully...
Thread 4 exit with code 0
 rc/process/processor.rs:98: 'all threads terminated, shutting down'
```

Result

- Capacity comparison
 - After qcow2 compression, LFS has a 53% storage overhead over SFS
- Performance simulation
 - Basic IO statistics

	Write count	Read count	IO Transfer length (Bytes)	Image capacity (compressed)
LFS	34582	11419	47120712	4.9MB
SFS	34215	22442	93869060	3.2MB

SSD model result

	LFS	SFS
Simulation Tick	1215821831777725	1221120730998634
Min Latency	2.871473ms	3.376592ms
Max Latency	3052.497074ms	3983.285137ms
Average Latency	1019.119791ms	1481.942723ms
Stdev Latency	3425.016061ms	1460.266471ms
Event Handled	156314	184877

Discussion

Comparing with FFS-like system

- Access "/foo/bar"
- o FFS-like system: reading 3 inodes in inodemap, 3 data blocks in data blocks
- LFS system: reading CR (in memory), 1 imap, 3 inodes and 3 data blocks
 - Likely in one segment

How to balance stale and dirty blocks

- Use append mode except
 - INode metadata modification (atime, mtime, ...)
 - Parent direntry appending/removing
- Adaptive approach: future work
 - Tradeoff between random access cost and log storage overhead

Future work

- It is a rough prototype so far!
 - Not support for double indirect data blocks (easy)
 - Use dedicated inode/meta struct instead of 4K block (easy)
 - Requiring refactoring to enjoy Rust language features (medium)
 - Not support for roll forward process (medium)

Repo links

- https://github.com/leepoly/aoslab
- https://github.com/Pingziwalk/rCore
- https://github.com/leepoly/rcore-fs/

Q&A

Reference

- [1] Rosenblum M, Ousterhout J K. The design and implementation of a log-structured file system[J]. ACM Transactions on Computer Systems (TOCS), 1992, 10(1): 26-52.
- [2] rcore-tutorial. https://rcore-os.github.io/rCore_tutorial_doc/
- [3] rcore-tutorial-OS20. https://os20-rcore-tutorial.github.io/rCore-Tutorial-deploy
- o [4] SFS Analysis. https://os20-rcore-tutorial.github.io/rCore-Tutorial-deploy/docs/lab-5/files/rcore-fs-analysis.pdf
- [5] Jung M, Zhang J, Abulila A, et al. Simplessd: modeling solid state drives for holistic system simulation[J]. IEEE Computer Architecture Letters, 2017, 17(1): 37-41.