Lab 1++: Tune Your FS!

Due: 11-20-2020 23:59 (UTC+8)

General Lab Info can be found in Lab Information.

Introduction

- In this lab, you should tune your YFS implementation in Lab1, and make it approximately match the performance of native file system. We use 'fxmark' benchmark to profile your YFS and the native file system. Your task is to make your YFS run correctly (30 points) and fast (60 points). And 10 points for a extra **doc**.
- If you have questions about this lab, either in programming environment or requirement, please ask TA: Xu Tianqiang (2286455782@qq.com).

Getting started

- Please backup your solution1 to lab2 and lab1 before starting the steps below
  - At first, please remember to save your lab2 solution:

```
% cd lab-cse/lab1
% git commit -a -m “solution for lab2”
```

  - Then, switch to lab1 branch and save the lab1 solution:

```
% git checkout lab1
% git commit -a -m “save solution for lab1, start to tune YFS”
```

  - Then, pull from the repository:

```
% git pull
remote: Counting objects: ...
[new branch] lab1-plus  -> origin/lab1-plus
Already up-to-date
```

  - Then, change to lab1-plus branch:

```
% git checkout lab1-plus
```

- Merge with lab1, and solve the conflict by yourself(if there is):

```
% git merge lab1
```

- There is no modification of your lab1 solution, so you should be able to compile successfully:

```
% make
```

- Make sure there's no error in make.

FxMark YFS

- FxMark stands for "Filesystem Multicore Scalability Benchmark". FXMARK implements 19 microbenchmarks to stress specific components of each file system and includes three application benchmarks to measure the macroscopic scalability behavior.
- In this lab, you will just need to use 'fxmark-yfs' benchmark to profile your YFS implementation. 'fxmar yfs' behaves in a way that continuously creates a file and deletes that file. For correctness, You should make sure that there is no memory leak. For performance, you should make 'works' as many as possible.

  - Use FxMark to profile **native** file system

```
% cd lab-cse/lab1
% mkdir native
% ./fxmark/bin/fxmark --type=YFS --root=./native --ncore=1 --
duration=5
# ncpu secs works works/sec
1 5.000600 574976.000000 114981.402232
```

  - Use FxMark to profile **your YFS**

```
% cd lab-cse/lab1
% sudo ./start.sh
% ./fxmark/bin/fxmark --type=YFS --root=./yfs1 --ncore=1 --duration=5
# ncpu secs works works/sec
1 18445139349741.347656 512.000000 0.000000
% sudo ./stop.sh
```

Guidance

- You can notice there is a huge performance gap between your YFS implementation and native file syste Your job is to make your YFS match the native file system performance.
- Firstly, you should find the bottleneck of the performance of YFS. To find the bottleneck, you can try 'rdtsc' instruction, or functions provided by library. You should write down the mechanism you use and

your findings in the **lab1-plus_[your student id].doc**.

- Secondly, after finding the bottleneck, you can try to speed it up.
- Write down all your findings, optimizations, and the best output generated from fxmark in the **lab1-plus_[your student id].doc**
- After uploading your lab1-plus solutions, we will rank all students according to the 'works' output printe by fxmark. Highly ranked students get higher scores

Handin procedure

- After all above done:

```
% make handin
```

- That should produce a file called lab1-plus.tgz in the directory. Change the file name to your student id:

```
% mv lab1-plus.tgz lab1-plus_[your student id].tgz
```

- Then upload lab1-plus_[your student id].tgz file to ftp://skyele:public@public.sjtu.edu.cn/upload/lab1-plus before the deadline. You are only given the permission to list and create new file, but no overwrite and read. So make sure your implementation has passed all the tests before final submit.