Operating System Design

**Assignment # 3**

Log-structured File System

Due date: December 9, 2013

Professor: Ricardo Bianchini

TA: Chen Li

Group Members (ID#)

Taichi Hirao (134005387)

Michael Spaniol (133002726)

Theophilus Mensah (129006665)

**Objective**

The objective of this assignment was to implement a user-level Log-structured File System (LFS) with FUSE that support basic file/directory operations.

**Architectural Description of Framework**

The implementation of the LFS was achieved by creating an in-memory dictionary data structure that stored all meta-data about each file. Space was created at the beginning of the file to log the free spaces when a file if deleted. The meta-data include the name, name, parent, type, offset, data size, children (if it a directory), date created, and date updated. In-memory global variables were created to keep track of file system statics tics. The file was logged with tags (e.g. <meta-data> …</meta-data>, <block #>…</block #>, and so on) for easy reading. A directory was treated as a file, with its data section updated with the list of the files in the directory.

When a read function is called, the file/directory name is looked-up in the dictionary. If the file exists, the offset to the data for that file/directory is acquired from the dictionary, and the data is read from the specific location in the log file respectively, otherwise an error is generated.

If a create file function is called, a new dictionary entry is created and updated with the information about the new file. If the file already exists, an error is generated. The meta-data is written together with the data to the log file when the write function is called. If create is called and the file is not written to, the meta-data is discarded.

A count of the location of the current end of the file is kept and checked with the file size before any write is executed. If the size if of the new data and the current end of file is greater than the size of the file, the garbage collector is called. When a file is updated, a new log entry is created and the location of the old file is logged as free space in the free space log. When delete file is called, the meta-data in the dictionary is discarded and the free-space section is updated with the location of the deleted file. The directory of that file is updated and re-logged as well as the free-space section.

When the garbage collector is called, it creates a temporal file, copies all live data, with the help of the free-space section log, to the temporal file. The original log file is deleted and the temporal file is renames to the original log file name.

When a directory is created, just as a file, a mete-data entry is created in the dictionary, but this time a log is made in the log file with the data section empty.

When the file system is mounted, an initialize-dictionary function is called to recreate the dictionary according to the current data of the log file. When the file system is un-mounted, the statistics is appended to the stats file in the program directory.

**Conclusion**

The task was very complex, but very intuitive. 40% of time was spent learning about fuse file system and its implementation.