BF Layer System Report

Rut Diane Cuebas (2019-28748)
Jungwoo Yang (2019-23417)

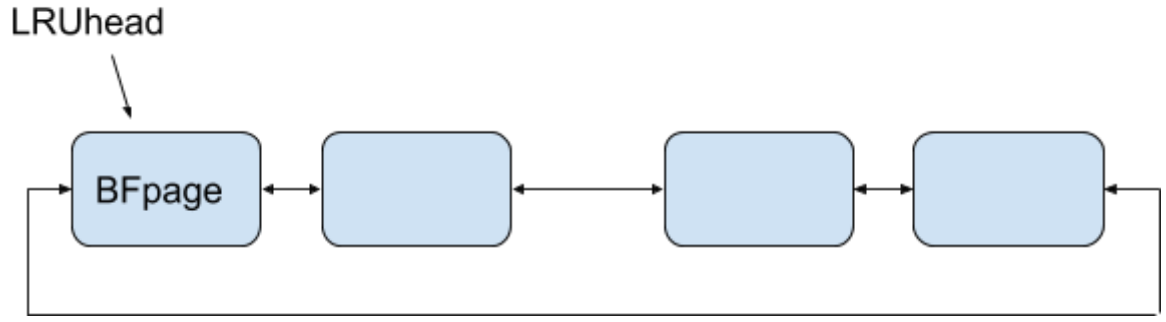
1. Environment

Ubuntu and Windows WSL and Mac

2. File structure

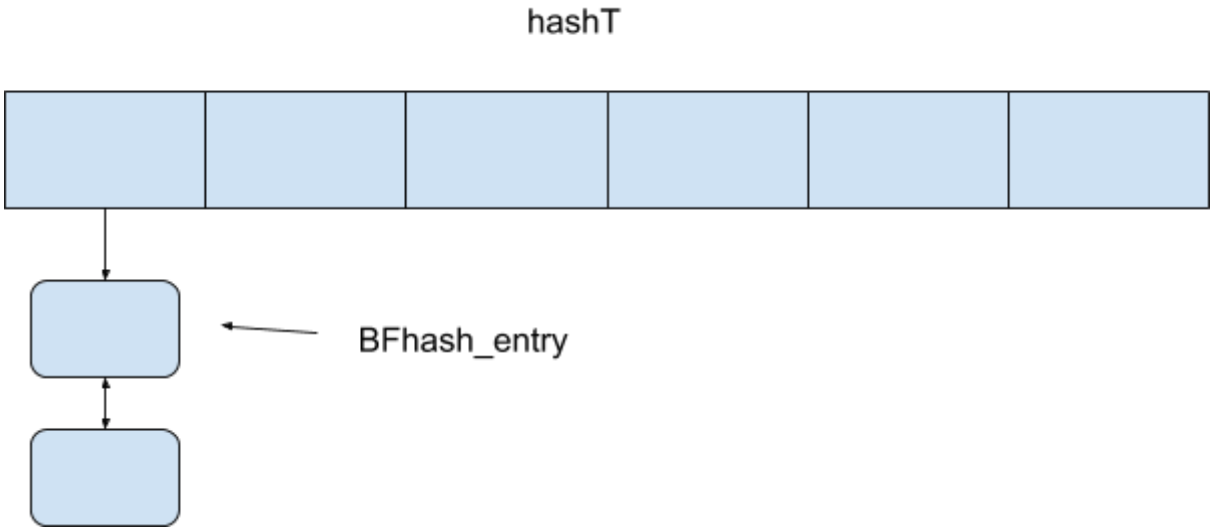
bf.c - contains implementations for bf.h
bfinternal.h - contains prototypes for functions that are used in bf.c
bfinternal.c - contains implementations for the functions in bfinternal.h

3. Data structures



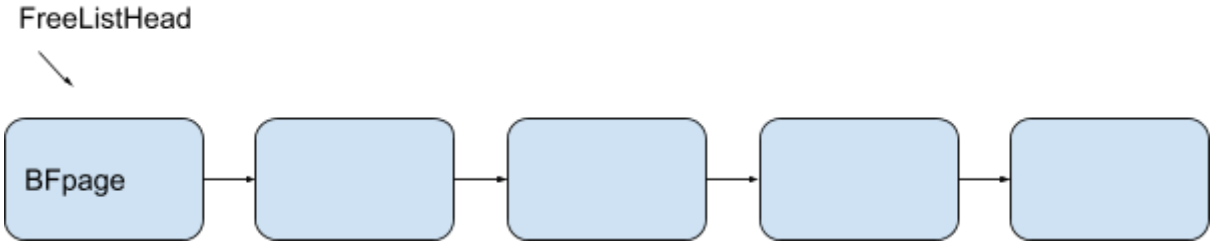
BFentry

- Implemented in circular doubly linked list
- LRUhead points to the LRU end
- Needs to go back one pointer to go to MRU end



hashT

- Uses pagenum and fd to find out the index of the file
- BFhash_entries are doubly linked
- Initialize freeHash[BF_MAX_BUFS] to get rid of unnecessary memory allocation



FreeListHead

- Keeps track of free BFentry
- It takes O(1) time to insert or remove element

4. Implementations

BFentry

- By implementing a circular doubly linked list, we only have to maintain one pointer to the list and going to the other end of the list takes O(n) time.

FreeBufferEntry & freebufferEntryFromHash

- If we try to free a page from page replacement policy we usually have to free only one page and we can then remove from hash
- However, if we free the buffer from BF_FlushBuf(int fd), the program goes through the hash table instead of BFentry. It could be more efficient if the buffer is quite full because the deletion time is O(1).

Hash Table

- Instead of allocating new memory for each BFhash_entry, we can make a table with size BF_MAX_BUFS , which is the upper bound for the number of BF_hash_entry required.