Write Up for Buffer Cache implementation

Akul Bharti, Ankita Yadav, Nikita Bansal and Rohit Shakya.

Data structure: Hash Map, Doubly Linked List , Queue , Classes and Structure.
Programming language: Python

Reason for implementation: The kernel could read and write directly to and from the disk for all the file system accesses, but system response time and throughput will be poor because of the slow disk transfer rate. The kernel therefore attempts to minimize the frequency of disk access by keeping a pool of data buffers, called the *buffer cache*,which contains data in recently used disk blocks. Architecturally, it is positioned between file subsystem and device drivers.

Driver Class : There will be a driver (Master class) class which will initialize all the objects and variables, work as a mediator(Kernel), creating and handling all the processes.

Buffer Header : During system initialization, the kernel (Master class) allocates space for a number of buffers, configurable according to memory size and performance constraints. And in our implementation we have 30 number of buffers and block of buffer header will be a type of structure having block number, device number, status flags, pointer for the data block, next buffer, previous buffer, pointer to previous and next hash queue.

The status of a buffer is a combination of the following conditions:

- Buffer is locked / busy
- Buffer contains valid data
- Kernel must write the buffer contents to disk before reassigning the buffer; called as "delayed-write"
- Kernel is currently reading or writing the contexts of the buffer to disk
- A process is currently waiting for the buffer to become free.

The two sets of pointers in the header are used for traversal of the buffer queues (doubly linked circular lists).

Scenarios for Retrieval of a Buffer

The algorithms for reading and writing disk blocks use the algorithm *getblk*to allocate buffers from the pool. There are 5 typical scenarios the kernel may follow in *getblk*to allocate a buffer for a disk block.

1. Block is found on its hash queue and its buffer is free.
2. Block could not be found on the hash queue, so a buffer from the free list is allocated.

3. Block could not be found on the hash queue, and when allocating a buffer from free list, a buffer marked "delayed write" is allocated. Then the kernel must write the "delayed write" buffer to disk and allocate another buffer.
4. Block could not be found on the hash queue and the free list of buffers is empty.
5. Block was found on the hash queue, but its buffer is currently busy.

Working

- Algorithms used in our implementation: **getblk, brelse.**
- The **Multiprocessing** python module is used to generate processes. However, for simplicity, the relationship is referred to as a parent-child relationship.
- We will also implement **Isolated User Mode (IUM),** which will help in **Process isolation**ie. designed to protect each process from other processes on the operating system. It does so by preventing process A from writing to process B.
- Process isolation can be implemented with virtual address space, where process A's address space is different from process B's address space – preventing A from writing onto B.
- Security is easier to enforce by disallowing inter-process memory access, in contrast with less secure architectures such as DOS in which any process can write to any memory in any other process.
- As Cache's purpose is to provide a fast and efficient way of retrieving data. It needs to meet certain requirements. Such as Fixed size, Fast access and replacement of Entry in case, memory limit is reached.
- In the case of LRU cache we evict least recently used entries so we have to keep track of recently used entries, entries which have not been used for a long time and which have been used recently. plus lookup and insertion operation should be fast enough. When we think about O(1) lookup , data structure comes in our mind is HashMap. HashMap provides O(1) insertion and lookup. but HashMap does not have a mechanism of tracking which entry has been queried recently and which not.
- To track this we require another data-structure which provides fast insertion ,deletion and update, in case of LRU we use Doubly Linked List.
- We will implement a sleepQueue that contains the information of sleep processes.