Problem Set 3

Part E

In order to support writing compressed data to flash, we'll need to add all the information needed to retrieve it as well. Here's a short outline for an implementation:

Writes

Writes will go to the cache as implemented in the assignment. When buffer flush is required, whole buffer will be written to a single page.

- Page mappings (both forward and inverse) will have to be updated.
    - Forward mappings will be changed from `LPN->PPN` to `LPN->(PPN,offset,size)`
    - Inverse mappings will be changed from `PPN->LPN` to `PPN->List[(LPN, offset, validity_state)]` (or hash map instead of list.)
- Flushing of a cache buffer to disk will update the mappings:
    - Update forward mappings for each of the pages in the buffer to point to the newly written page.
    - Replace inverse mapping of the newly-written page with all the logical pages in the buffer.
    - Set inverse mapped compressed page to `validity_state` to invalid.

Reads

Reading a page: 1. If the page is in the cache, return from cache 2. will utilize the offset in the `offset` and `size` stored in the mapping to find and de-compress the page.

Reliability

- In order to cope with power failure during write, we can store the inverse mapping in the spare area of each physical page.
- Save the cache mapping as meta in each page.

GC

Garbage collection can be further optimized: * When copying physical pages, only valid logical pages have to be copied. * When selecting victim blocks, take into account the percent of valid data in each page. * merge valid logical pages into a new page.