Cache.java

frameIndex:

Contains the disk block number for the cached data. Set as -1 when the Entry array is initialized. Nothing has valid block information at the moment.

reference:

set to true if the block is accessed. Reset to false when nextVictim() searchs for next victim.

isDirty:

set to true if the block is written. Reset to false if the block is written back to disk.

nextVictim()

Check for page with frameIndex = -1. Return a free page with a false reference. Iterate through all entries. Check if the page reference is true (recently used), then set it to false before moving to the next. This is to ensure that if there is no page that has a false reference. The next iteration will provide a candidate to be the victim. After finding a victim, check if the victim is written. Written the content from the cache block back to disk.

read()

Check if the corresponding blockID entry can be found in cache. If found, set the reference to true as we are accessing this page. Read the buffer content from the cache. If there is none, call the nextVictim() to look for a free page or a victim. The algorithm is mentioned above. Load the data from disk into the cache block. If there is any content from the cache block, the nextVictim() already written it back to the disk. Data is then load from cache block into the buffer. Afterward, update the page. The frameIndex will be the argument blockID. Set reference to true. Set isDirty to false. Move the victim counter to the next one.

write()

Check if the corresponding blockID entry can be found in cache. If found, set the reference and isDirty to true as we are accessing and modifying this page. Write the buffer content into the cache. If there is none, call the nextVictim() to look for a free page or a victim. The algorithm is mentioned above. Write buffer content into cache block. If there is any content from the cache block, the nextVictim() already written it back to the disk. Afterward, update the page. The frameIndex will be the argument blockID. Set reference to true. Set isDirty to true as we written something in. Move the victim counter to the next one.

sync()

Write all data on cache block into the disk. Set all page isDirty to false.

flush()

Similar to sync(). But also set the frameIndex to -1 and reference to false. Move the victim counter to the start of the entries. Basically, reset to construction setting.

Result

```
shell[1]% Test4 Enabled 1
Test4
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
Random Access Test
Caching: Enabled
Execution time: 36366
Average time: 72
shell[2]% Test4 Disabled 1
Test4
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
Random Access Test
Caching: Disabled
Execution time: 37435
Average time: 74
shell[3]% Test4 Enabled 2
Test4
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
Localized Access Test
Caching: Enabled
Execution time: 2
Average time: 0
shell[4]% Test4 Disabled 2
Test4
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
Localized Access Test
Caching: Disabled
Execution time: 201581
Average time: 403
shell[5]% Test4 Enabled 3
Test4
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
Mixed Access Test
Caching: Enabled
Execution time: 12897
Average time: 25
shell[6]% Test4 Disabled 3
Test4
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
Mixed Access Test
Caching: Disabled
Execution time: 25291
Average time: 50
shell[7]% Test4 Enabled 4
Test4
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
Adversary Access Test
Caching: Enabled
Execution time: 21690
Average time: 43
shell[8]% Test4 Disabled 4
Test4
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
Adversary Access Test
Caching: Disabled
Execution time: 20850
Average time: 41
```

Analysis

Random Access

The cache enabled run is faster than the cache disabled run. Accessing the cached data is much faster than from DISK. Faster reading speed.

Localized Access

The cache enabled run is faster than the cache disabled run. Accessing the cached data is much faster than from DISK. Faster reading speed. Significantly faster for cache because of high ratio cache hits.

Mixed Access

The cache enabled run is faster than the cache disabled run. Accessing the cached data is much faster than from DISK. Faster reading speed. Based on random and localized access. Therefore, reasonable to have similar comparison result.

Adversary Access

The cache enabled run is slower than the cache disabled run. The result is similar to the expected result. This access does not make good use of disk cache. Never accept the same block, therefore cache is not effective. Needing to check the cache first results in a longer execution time.