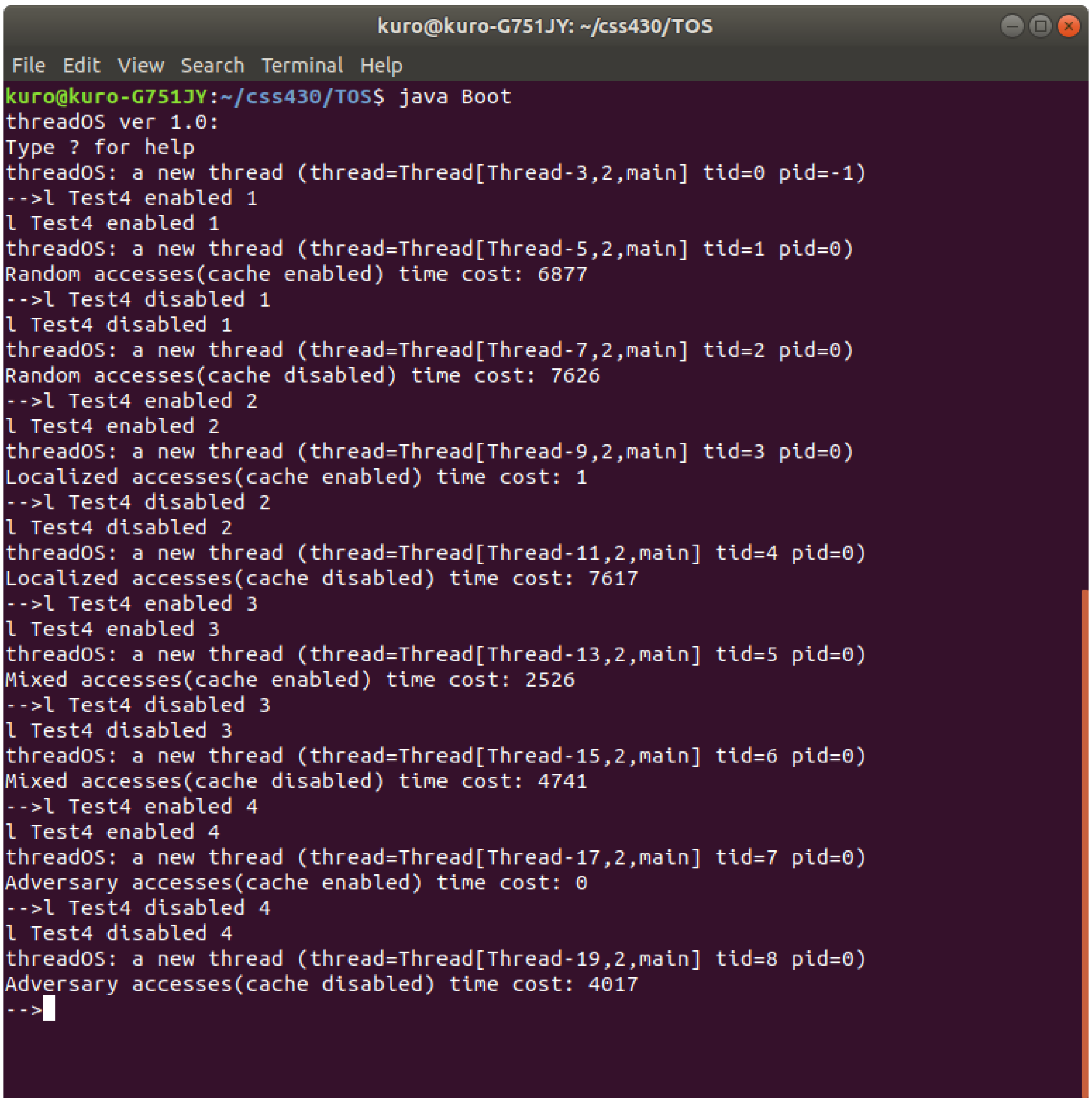
Cache class simulates a CPU cache read and write with enhanced secondchance algorithm. For the block read, I firstly check if the corresponding entry can be found in cache. If not and there is a space, load from disk to free block. However, if there is no space, I find the victim with enhanced second chance algorithm and check the dirty bit. If dirty bit been set to true, copy related cache block to disk, then load data from disk to cache block. The read (int blockID, byte buffer[]) function takes two arguments does what I describe above. The write (int blockID, byte buffer[]) function does kind similar function as read excepted it writes buffer content to cache instead of reading data into buffer. The sync() and flush() function backup all blocks with dirty to DISK, but flush() also initialize all cache blocks


kuro@kuro-G751JY: ~/css430/TOS
File Edit View Search Terminal Help

```
kuro@kuro-G751JY:~/css430/TOS$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test4 enabled 1
l Test4 enabled 1
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
Random accesses(cache enabled) time cost: 6877
-->l Test4 disabled 1
l Test4 disabled 1
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=0)
Random accesses(cache disabled) time cost: 7626
-->l Test4 enabled 2
l Test4 enabled 2
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=0)
Localized accesses(cache enabled) time cost: 1
-->l Test4 disabled 2
l Test4 disabled 2
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=0)
Localized accesses(cache disabled) time cost: 7617
-->l Test4 enabled 3
l Test4 enabled 3
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=0)
Mixed accesses(cache enabled) time cost: 2526
-->l Test4 disabled 3
l Test4 disabled 3
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=0)
Mixed accesses(cache disabled) time cost: 4741
-->l Test4 enabled 4
l Test4 enabled 4
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=0)
Adversary accesses(cache enabled) time cost: 0
-->l Test4 disabled 4
l Test4 disabled 4
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=0)
Adversary accesses(cache disabled) time cost: 4017
-->
```

(Figure 1)

1. Random access

   Cache access faster than without cache.

2. Localized access
   Cache access much faster than without cache.
3. Mixed accesses
   Cache access faster than without cache.
4. Adversary access
   Cache access much faster than without cache.

From Figure 1 we can notice that disk access with cache performs much better than without cache, and it especially obviously in localized access and adversary access. This because the reading speed from cache is much faster than from DISK.