Homework Wan Huzaifah bin Wan Azhar

Answer:



* Commands:
  + Write(12, u)
  + Write(32, M)
  + Read(32) -> M
  + Write(38, 0)
  + Write(36, e)
  + Trim(36)
  + Read(32) -> M
  + Trim(32)
  + Read(12) -> u
  + Read(12) -> u



cmd 0:: write(36, F) -> success

cmd 1:: write(29, 9) -> success

cmd 2:: write(19, I) -> success

cmd 3:: trim(19) -> success

cmd 4:: write(22, g) -> success

cmd 5:: read(29) -> 9

cmd 6:: read(22) -> g

cmd 7:: write(28, e) -> success

cmd 8:: read(36) -> F

cmd 9:: write(49, F) -> success

1. write(36, F) -> success

FTL 36: 0

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F

Live +

1. write(29, 9) -> success

FTL 36: 0 29:1

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9

Live ++

1. write(19, I) -> success

FTL 36: 0 29:1 19:2

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9l

Live +++

1. trim(19) -> success

FTL 36: 0 29:1

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9l

Live ++

1. write(22, g) -> success

FTL 36: 0 29:1 22:3

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lg

Live ++ +

1. read(29) -> 9

FTL 36: 0 29:1 22:3

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lg

Live ++ +

1. read(22) -> g

FTL 36: 0 29:1 22:3

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lg

Live ++ +

1. write(28, e) -> success

FTL 36: 0 29:1 22:3 28:4

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lge

Live ++ ++

1. read(36) -> F

FTL 36: 0 29:1 22:3 28:4

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lge

Live ++ ++

1. write(49, F) -> success

FTL 36: 0 29:1 22:3 28:4 49:5

Block 0 1 2 3 4 5 6

Page 0000000000 1111111111 2222222222 3333333333 4444444444 5555555555 6666666666

0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789

State iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii iiiiiiiiii

Data F9lgeF

Live ++ +++



* -r 20 means that 20% of read commands will fail as it is been given non-live address.

1. Measuring log performance

* Commands:
  + Erase(0) = 1000ms
  + Write(12, u) = 40ms
  + Write(32, M) = 40ms
  + Write(38, 0) = 40ms
  + Write(36, e) = 40ms
  + Trim(22) = 0ms
  + Trim(33) = 0ms
  + Read() = 10ms
  + Read() = 10ms
  + Read() = 10ms
  + Read() = 10ms
* Total time = 1200ms

1. Measuring direct performance

* Commands:
  + Erase(1) = 1000ms
  + Erase(3) = 3000ms
  + Write(12, u) = 40ms
  + Write(32, M) = 40ms x 3
  + Write(38, 0) = 40ms x 2
  + Write(36, e) = 40ms x 1
  + Trim(22) = 0ms
  + Trim(33) = 0ms
  + Read() = 10ms
  + Read() = 10ms
  + Read() = 10ms
  + Read() = 10ms
* Total write = 280ms
* Total erase = 4000ms
* Total read = 70ms
* Total time = 4350ms



* The disk is full so new data cannot be allocated.
* At the end, only trimming operation can be done.
* Even read operation fail because the blocks that the system is trying to read is not live (as in already trimmed)



* For high watermark, “-G”, anything below the total size of block will make the system work.
* For example, for block size of 0 to 6, setting “-G” lower or equal to 7 will works.
* For low watermark, “-g”, there is no one number to make the system work without having high watermark.
* In my investigation, setting “-g” to anything will work as long as “-G” is set properly.



* GC will typically read in data and write it to other block before erasing the block.
* Total time of ideal SSD is 24, 540 ms while total time of log SSD is 126,840 ms, which is 6x more compared to ideal SSD.
* Main wasted time is spent erasing blocks, which is 86000ms compared to ideal SSD of 0ms (no need to erase anything).
* For log SSD:
  + 13 operations to erase
  + 126 operations to write
  + 101 operations to read
* While for ideal SSD:
  + 0 erase, 117 write, 76 read
* This difference is because of Garbage Collector. More write and read is required to transfer data to another block, so that it can erase the block.
* For direct SSD, the total time is 759,440ms, which is 7x worse than log SSD.
* This difference is due to erase time of 520,000ms. It happens because for every write, it needs to erase the block and rewrite it back.
* In essence, log-based SSD is the realistic approach to SSD.



* Using 80/20 makes less erase as 80% of the writes goes to 20% of the block. Less erasing means that the less GC needs to perform cleaning, as in, if the write does not reached low watermark, GC will not run.