**Project 3 Write out**

Tianrun Gu

Frames = 8:

**gcc.trace:**

OPT:

* Total page faults: 13328
* Total writes to disk: 4380

FIFO:

* Total page faults: 29011
* Total writes to disk: 11519

Aging:

In my test, based on the multiple of 10, I found “40” is the best choice for refreshing rate. Following are statics:

R = 30:

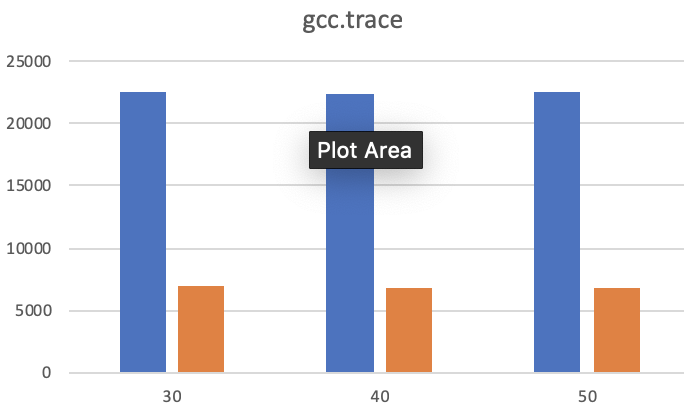
* Total page faults: 22487
* Total writes to disk: 6932

R = 40:

* Total page faults: 22312
* Total writes to disk: 6732

R = 50:

* Total page faults: 22484
* Total writes to disk: 6769



Summary: for gcc.trace, when the frames = 8, the best algorithm should be “OPT” because it has least page faults and disk writes. “OPT” in this case could promise the best CPU and I/O performance. But in the real world, we should use “aging” because it provides better performance on both fields than “FIFO” does.

**gzip.trace**:

OPT:

* Total page faults: 44918
* Total writes to disk: 39844

FIFO:

* Total page faults: 39874
* Total writes to disk: 39844

Aging:

During my test, based on the multiple of 10, I found the best refreshing period should be “20”. Following are statics:

R = 10:

* Total page faults: 39906
* Total writes to disk: 39863

R = 20:

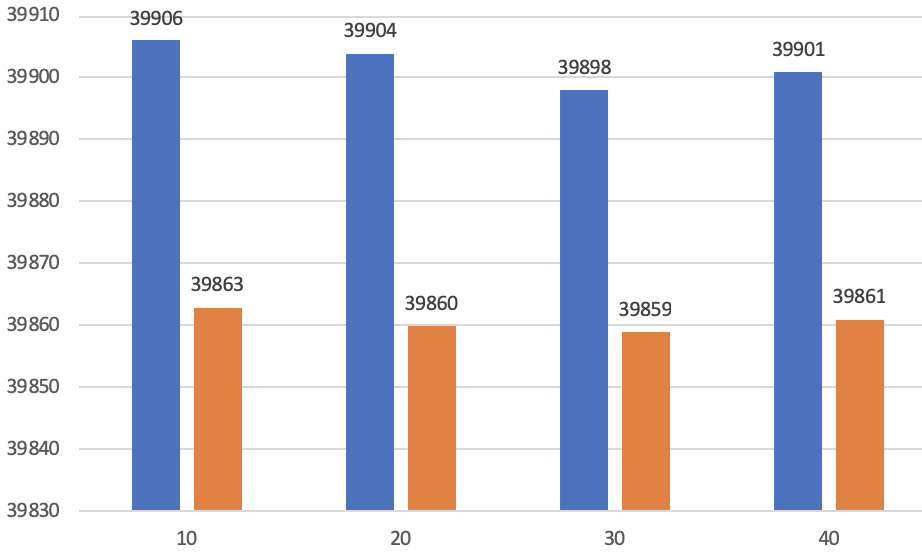
* Total page faults: 39904
* Total writes to disk: 39860

R = 30:

* Total page faults: 39898
* Total writes to disk: 39859

R = 40:

* Total page faults: 39901
* Total writes to disk: 39861



Summary: for gzip.trace, when the frames = 8, the best algorithm should be “FIFO” because all three algorithms have similar number of page faults and number of disk writes. But “FIFO” in this case has relative advantage for both statics. Another reason is that “FIFO” is the easiest one to implement. So “FIFO” is the best for this case.

**swim.trace:**

OPT:

* Total page faults: 4417
* Total writes to disk: 2182

FIFO:

* Total page faults: 13893
* Total writes to disk: 8499

Aging:

During my test, based on the multiple of 5, I found the best refreshing period should be “10” because it has least page faults which could save a lot context switch. Following are statics:

R = 5:

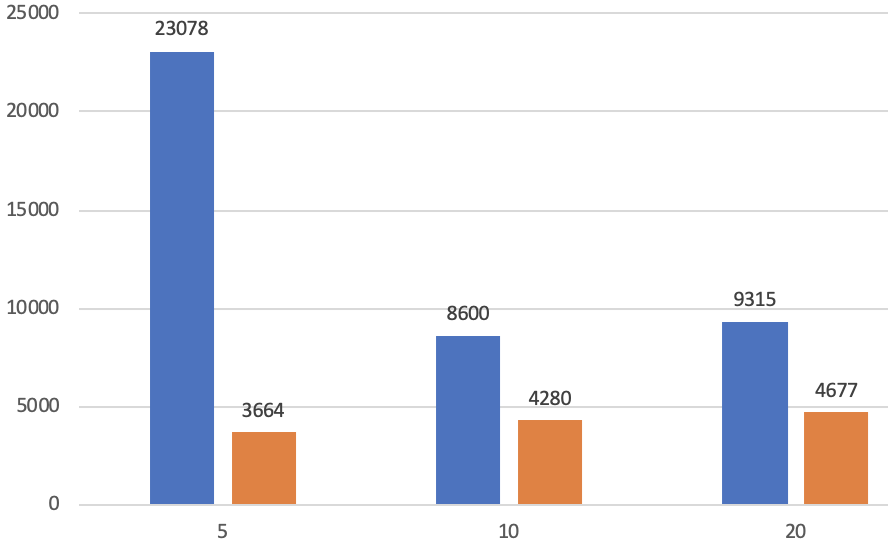
* Total page faults: 23078
* Total writes to disk: 3664

R = 10:

* Total page faults: 8600
* Total writes to disk: 4280

R = 20:

* Total page faults: 9315
* Total writes to disk: 4677



Summary: for swim.trace, I found the “OPT” totally wins this one. “OPT” has advantage in both “page faults” and “disk writes”, which could provide the best CPU and I/O performance. But in the real world, we should use “aging” because it does better on both fields than “FIFO” does.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frames = 16:

**gcc.trace:**

OPT:

* Total page faults: 3020
* Total writes to disk: 1019

FIFO:

* Total page faults: 8568
* Total writes to disk: 3542

Aging:

In my test, based on the multiple of 50 refresh periods, I found “250” is the best choice for refreshing rate. Following are statics:

R = 200:

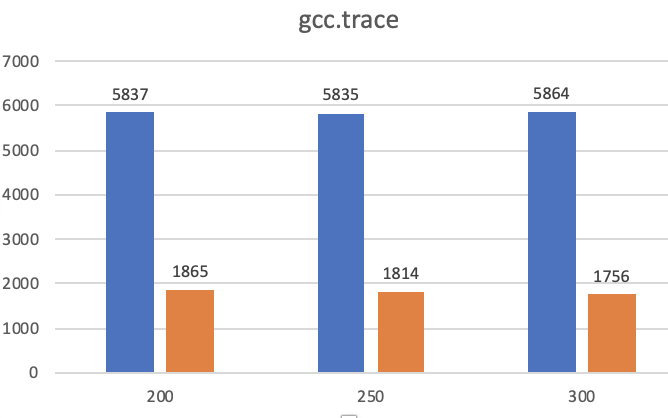
* Total page faults: 5837
* Total writes to disk: 1865

R = 250:

* Total page faults: 5835
* Total writes to disk: 1814

R = 300:

* Total page faults: 5864
* Total writes to disk: 1756



Summary: for gcc.trace, I found the “OPT” totally wins this one. “OPT” has advantage in both “page faults” and “disk writes”, which could provide the best CPU and I/O performance. But in the real world, we should use “aging” because it defeats “FIFO” on both aspects.

**gzip.trace**:

OPT:

* Total page faults: 39856
* Total writes to disk: 39825

FIFO:

* Total page faults: 42384
* Total writes to disk: 39856

Aging:

During my test, based on the multiple of “50”, I found refresh period as “100” might be a good choice. Following are statics:

R = 50:

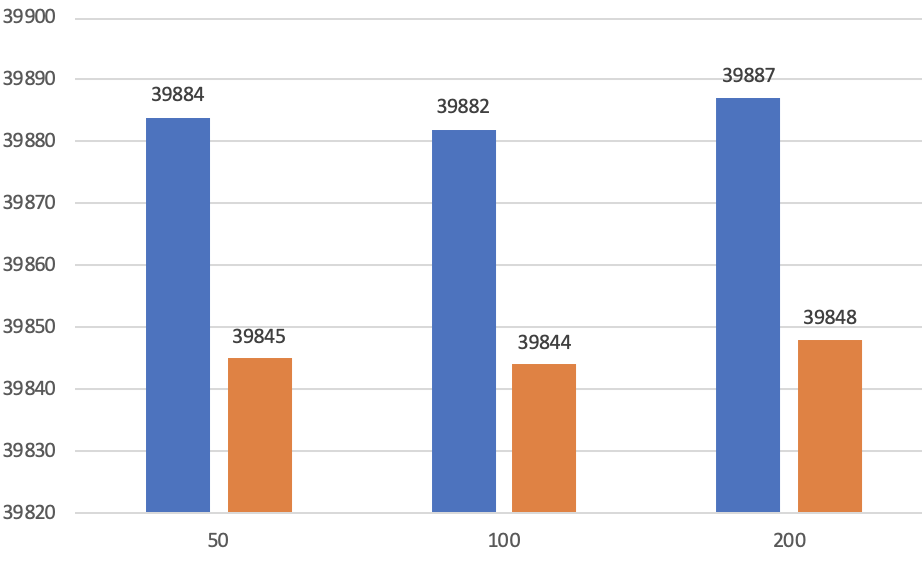
* Total page faults: 39884
* Total writes to disk: 39845

R = 100:

* Total page faults: 39882
* Total writes to disk: 39844

R = 200:

* Total page faults: 39887
* Total writes to disk: 39848



Summary: For this particular test, “OPT” do have a very little advantage for both “page faults” and “writes to disk”. However, if we are considering the real world OS, we would better choose “Aging” because it saves a lot on “page faults” comparing to “FIFO”, which could save a lot context switches for CPU.

**swim.trace:**

OPT:

* Total page faults: 358
* Total writes to disk: 149

FIFO:

* Total page faults: 844
* Total writes to disk: 470

Aging:

R = 250:

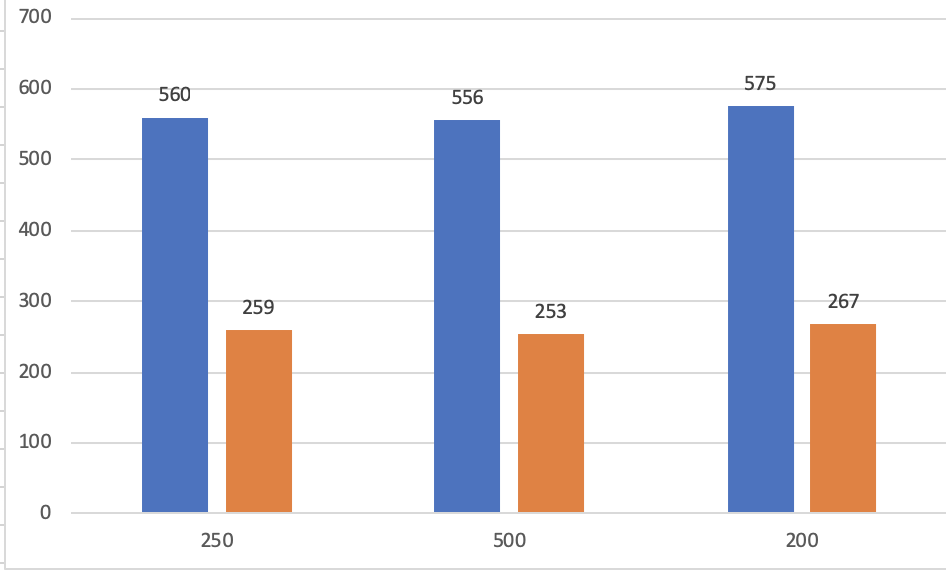
* Total page faults: 560
* Total writes to disk: 259

R = 300:

* Total page faults: 556
* Total writes to disk: 253

R = 350:

* Total page faults: 575
* Total writes to disk: 267



Summary: Again, though OPT looks much better, but we would better choose “Aging” in the real world implementation which saves a lot of overhead comparing to “FIFO”.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frames = 32:

**gcc.trace:**

OPT:

* Total page faults: 491
* Total writes to disk: 215

FIFO:

* Total page faults: 1375
* Total writes to disk: 660

Aging:

In my test, based on the multiple of 50 refresh periods, I found “900” is the best choice for refreshing rate. Following are statics:

R = 850:

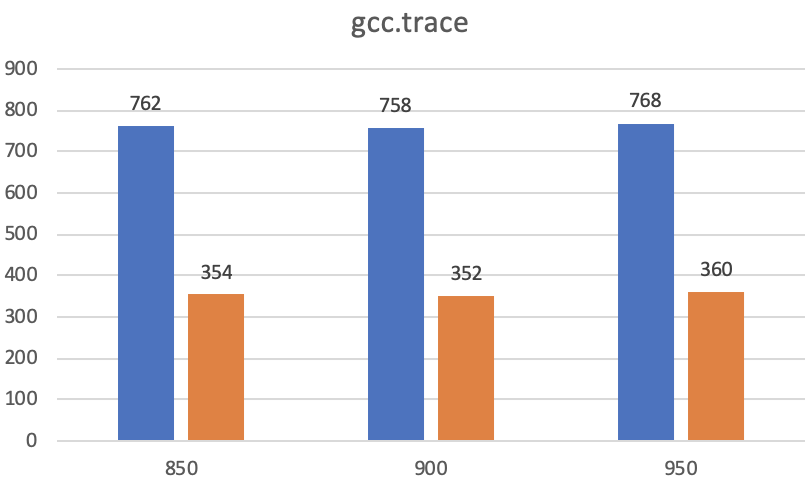
* Total page faults: 762
* Total writes to disk: 354

R = 900:

* Total page faults: 758
* Total writes to disk: 352

R = 950:

* Total page faults: 768
* Total writes to disk: 360

****

Summary: OPT is ideal performer but not ideal to implement. So Aging wins this one over FIFO.

**gzip.trace**:

OPT:

* Total page faults: 39856
* Total writes to disk: 39809

FIFO:

* Total page faults: 41120
* Total writes to disk: 39825

Aging:

During my test, aging algorithm for this file has a very stable performance. It seems refersh period doesn’t change a lot.

R = 20:

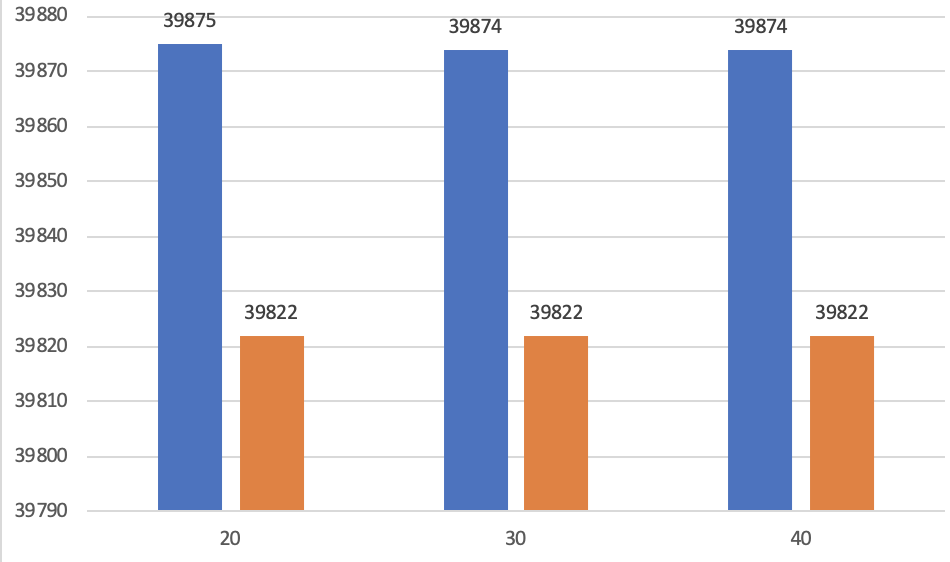
* Total page faults: 39875
* Total writes to disk: 39822

R = 30:

* Total page faults: 39874
* Total writes to disk: 39822

R = 40:

* Total page faults: 39874
* Total writes to disk: 39822



Summary: for gzip.trace, “Aging” has more high efficiency and stable performance. Though it is more difficult to implement than “FIFO”, it worth the engineering effort in the real world.

**swim.trace:**

OPT:

* Total page faults: 144
* Total writes to disk: 60

FIFO:

* Total page faults: 326
* Total writes to disk: 158

Aging:

R= 2000:

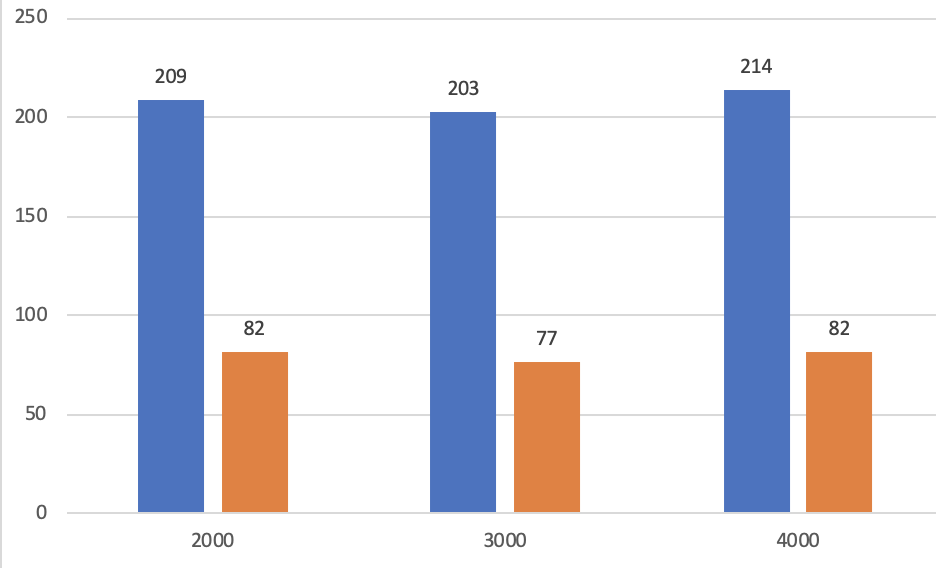
* Total page faults: 209
* Total writes to disk: 82

R = 3000:

* Total page faults: 203
* Total writes to disk: 77

R = 4000:

* Total page faults: 214
* Total writes to disk: 82



Summary: Not real big deal with the difference between the all three algorithms. Considering due to the characteristic of file itself, I suggest it might be a good choice to implement using “FIFO” because it saves engineering effort.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frames = 64:

**gcc.trace:**

OPT:

* Total page faults: 318
* Total writes to disk: 105

FIFO:

* Total page faults: 551
* Total writes to disk: 258

Aging:

In my test, based on the multiple of 10000 refresh periods, I found “20000” is the best choice for refreshing rate. Following are statics:

R = 10000:

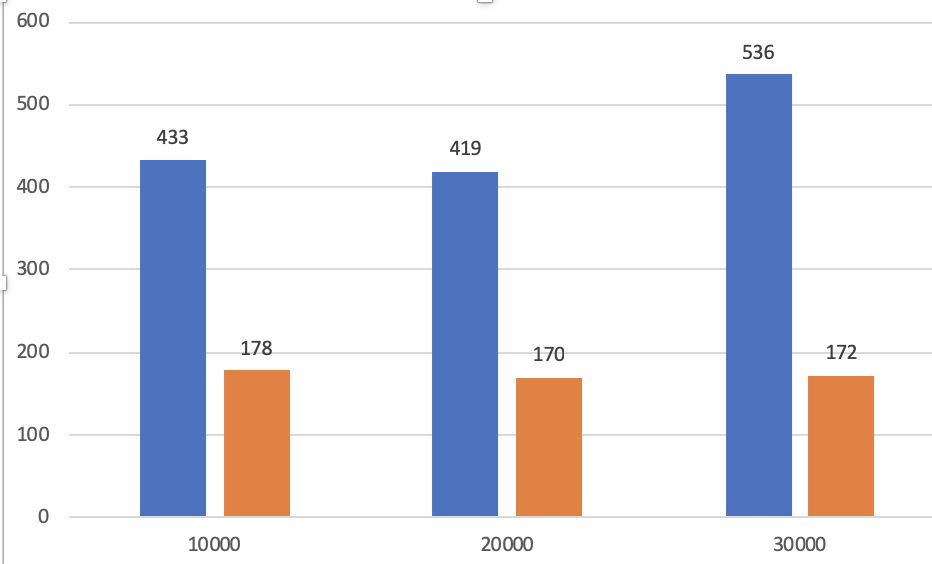
* Total page faults: 433
* Total writes to disk: 178

R = 20000:

* Total page faults: 419
* Total writes to disk: 170

R = 30000:

* Total page faults: 536
* Total writes to disk: 172



Summary: With the frame size becomes much larger, there is just little difference between three algorithms’ performance. But considering the engineering effort, I suggest “FIFO” is a better choice.

**gzip.trace**:

OPT:

* Total page faults: 39856
* Total writes to disk: 39777

FIFO:

* Total page faults: 40496
* Total writes to disk: 39793

Aging:

During my test, aging algorithms has a stable performance on this file even with varying refersh periods.

R = 10:

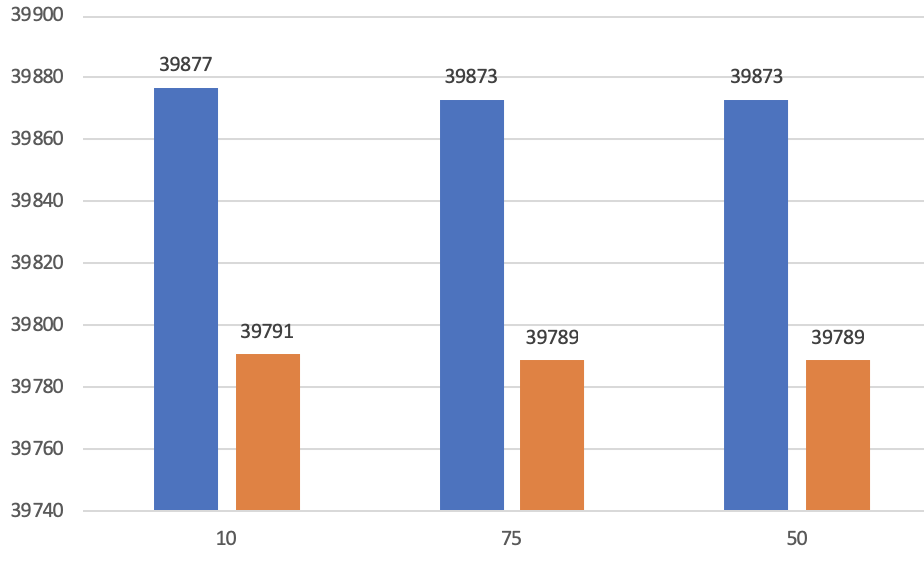
* Total page faults: 39877
* Total writes to disk: 39791

R = 75:

* Total memory accesses: 39873
* Total page faults: 39789

R = 100:

* Total page faults: 39873
* Total writes to disk: 39789



Summary: Again, with the frames size becomes larger, the performance difference between three algorithms is still around “500” on page faults, and even much smaller on disk writes. So I suggest “FIFO” is a better choice for saving engineering effort.

**swim.trace:**

OPT:

* Total page faults: 135
* Total writes to disk: 26

FIFO:

* Total page faults: 177
* Total writes to disk: 69

Aging:

R = 10000:

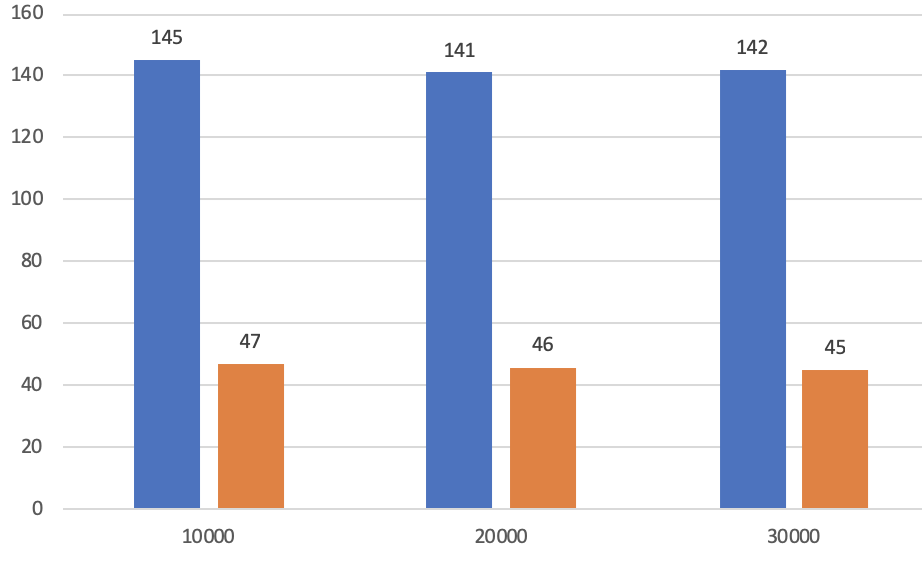
* Total page faults: 145
* Total writes to disk: 47

R = 20000:

* Total page faults: 141
* Total writes to disk: 46

R = 30000:

* Total page faults: 142
* Total writes to disk: 45



Summary: Again, with the larger frame size, the performance difference is really minimal. I suggest we should use “FIFO” to save engineering effort in the real world.