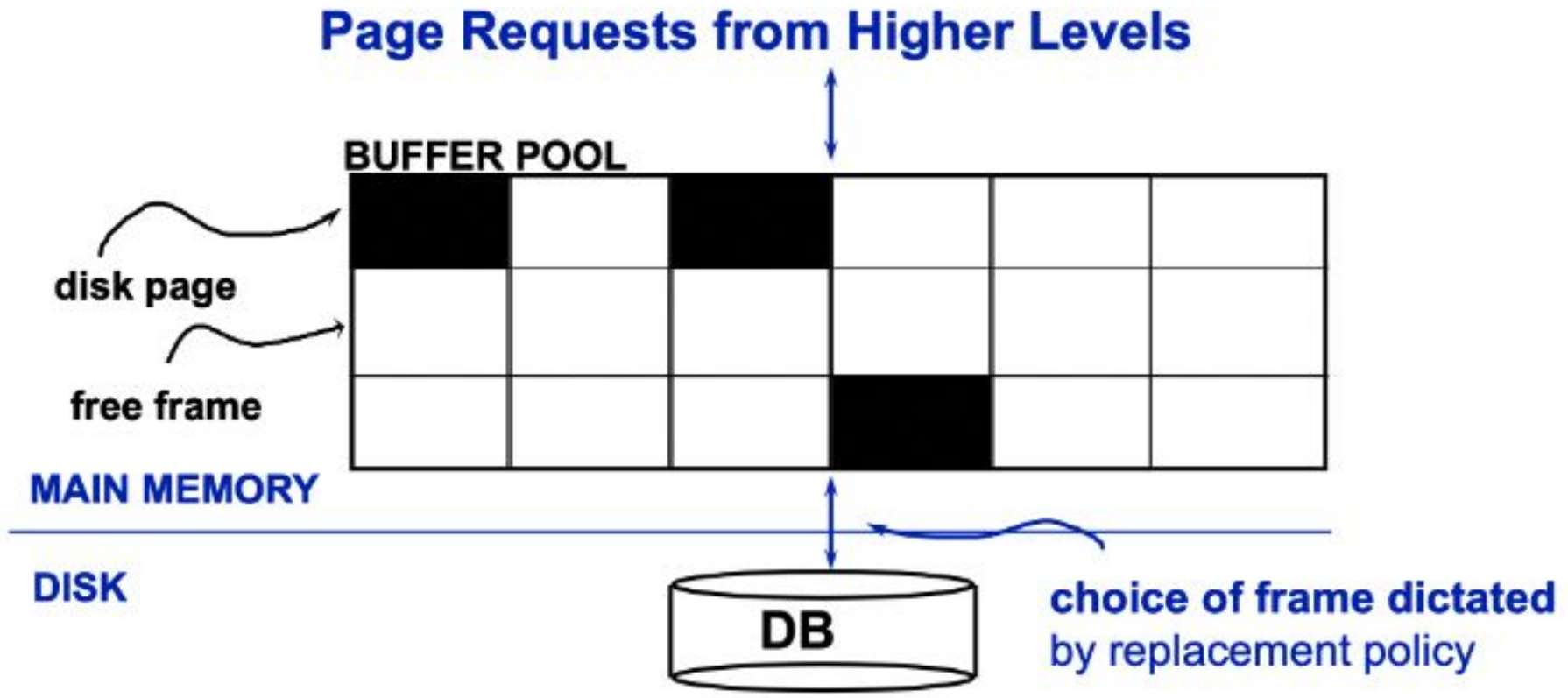
Buffer Pool

CS 561- Systems Project
Harsh Mutha
Aditya Pal
Manind Gera

Problem Statement



Eviction Policies Implemented

LRU

LRU-WSR

LFU

CFLRU

Challenges

- Implementing CFLRU was quite tricky in the start
- Adjusting to C++
- Perf does not work on mac, worked on Azure

Experimental Highlights & Results

Standard Workload

•Parameters :

•b = 150 (no of pages in buffer pool)

•n = 1500 (no of pages in disk)

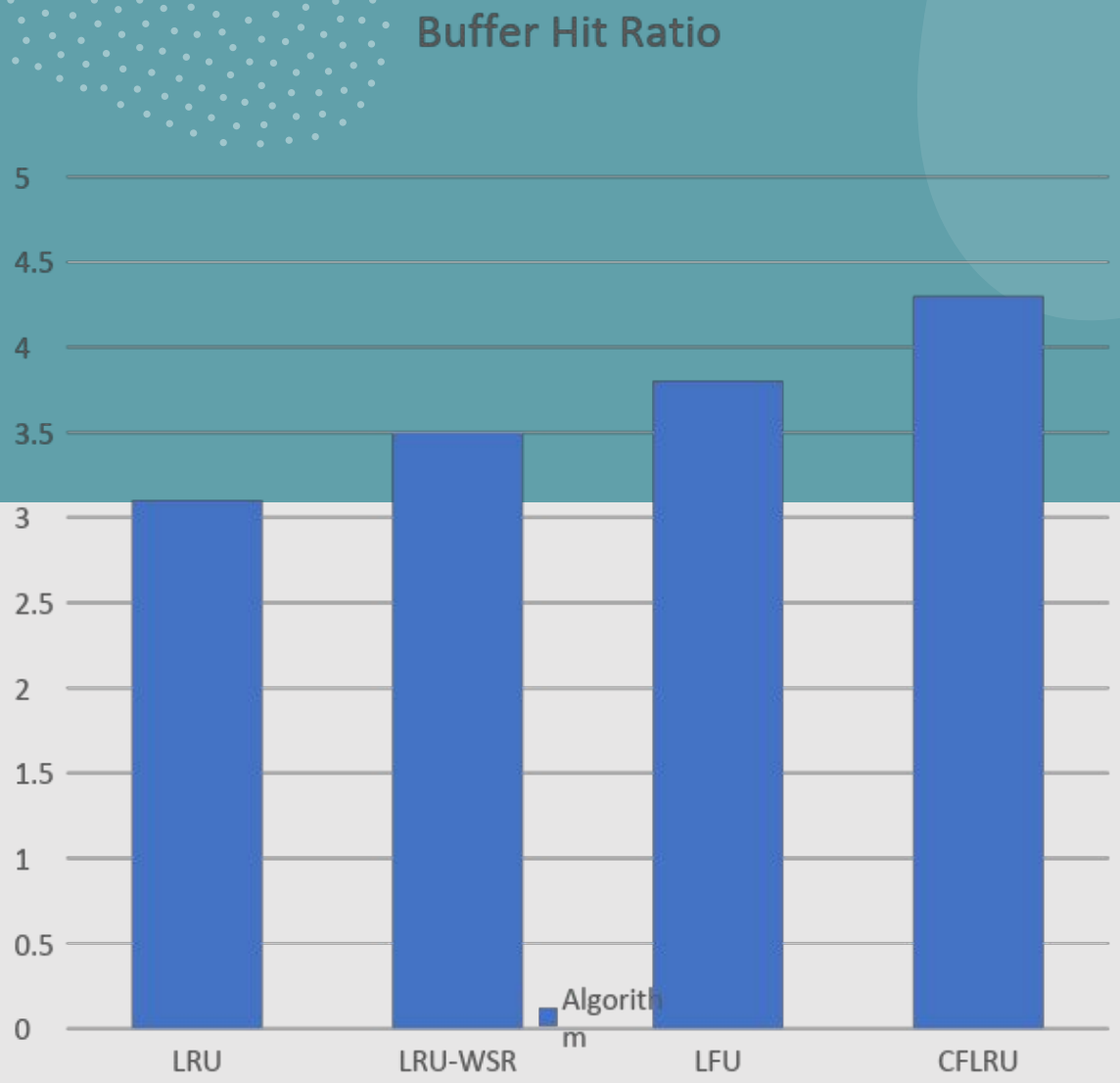
•x = 7500 (no of operations for workload to execute)

•e = 60 (percentage reads)

• a = 0…3 (type of eviction algorithm)

•s = 90 (skew percentage)

•d = 10 (skewed data percentage)



Changing parameters s(skew percent) and d(skewed data percentage)

•Parameters :

•b = 150 (no of pages in buffer pool)

•n = 1500 (no of pages in disk)

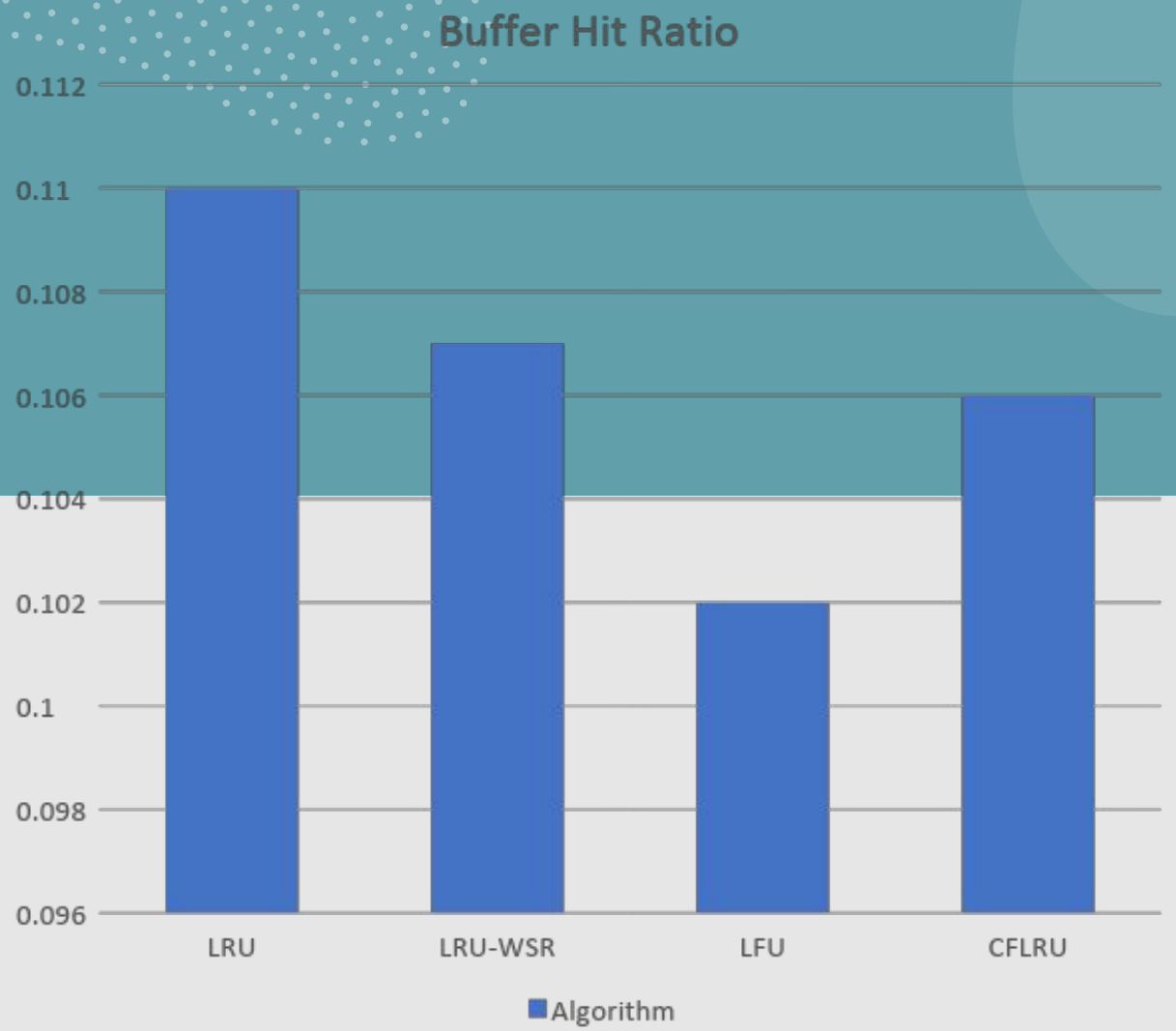
•x = 7500 (no of operations for workload to execute)

•e = 60 (percentage reads)

• a = 0…3 (type of eviction algorithm)

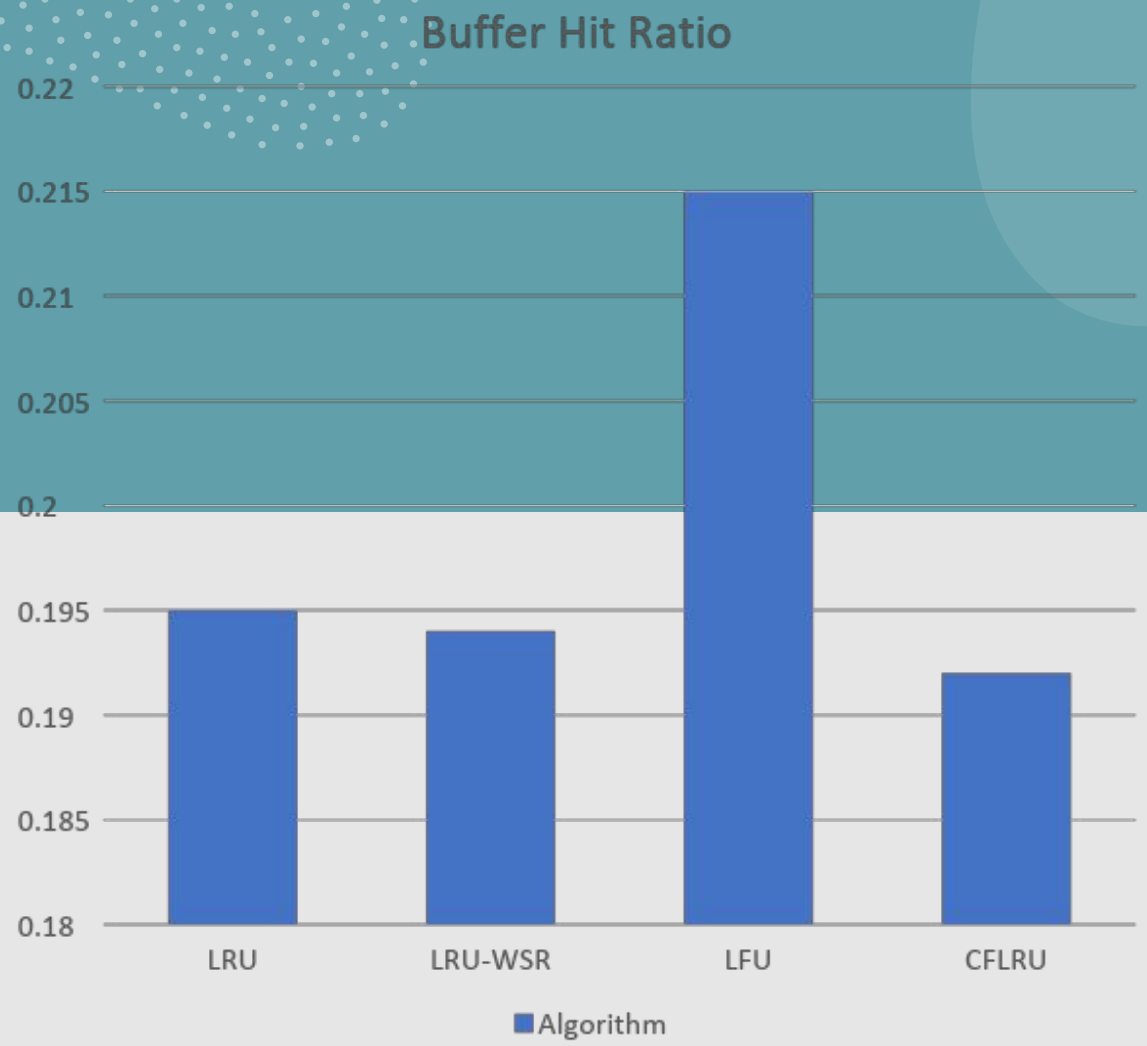
•s = 50 (skew percentage)

•d = 50 (skewed data percentage)



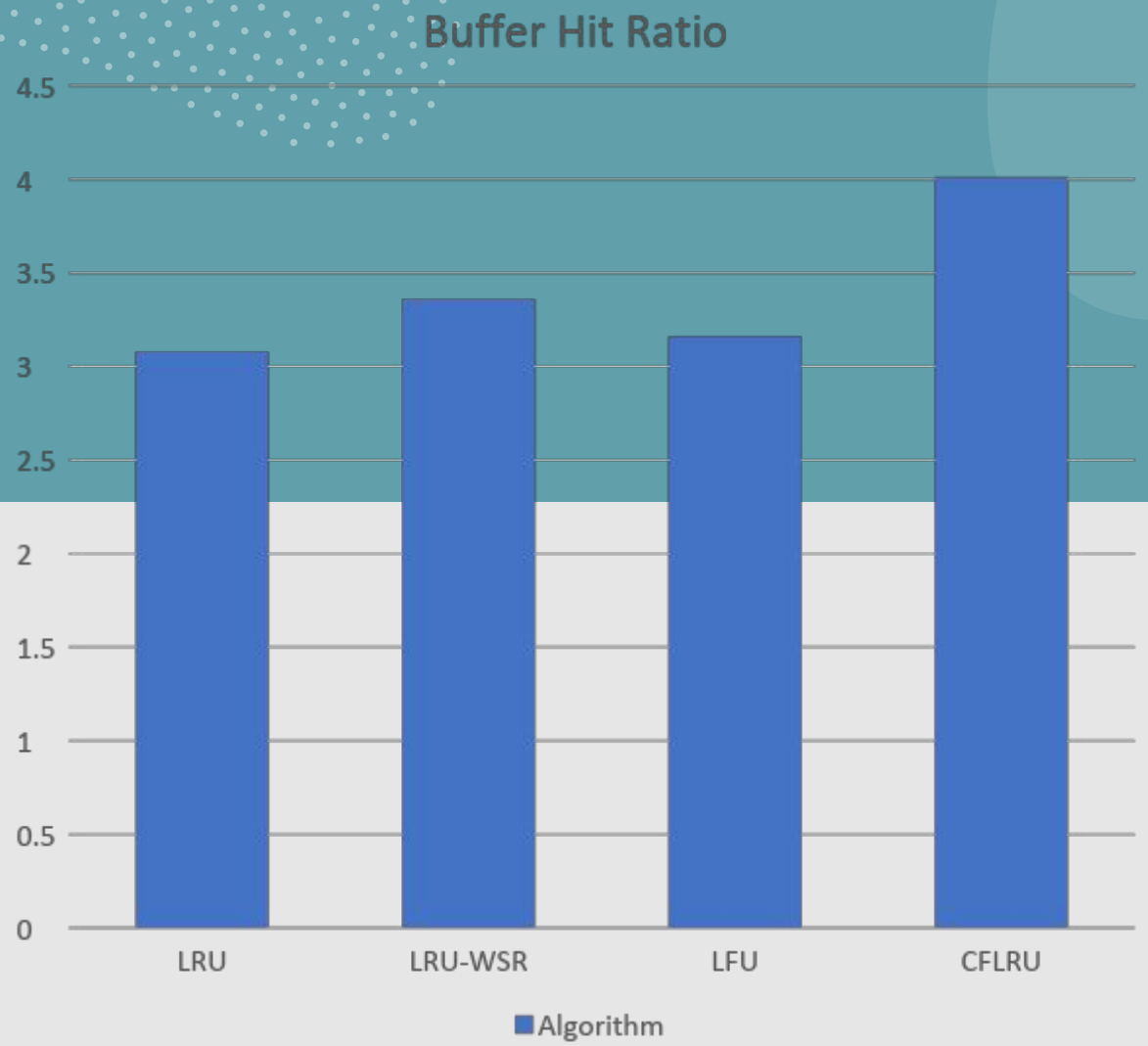
Changing parameters b(size of buffer pool) and d(size of disk)

•Parameters :
•b = 10 (no of pages in buffer pool)
•n = 500 (no of pages in disk)
•x = 7500 (no of operations for workload to execute)
•e = 60 (percentage reads)
• a = 0…3 (type of eviction algorithm)
•s = 90 (skew percentage)
•d = 10 (skewed data percentage)



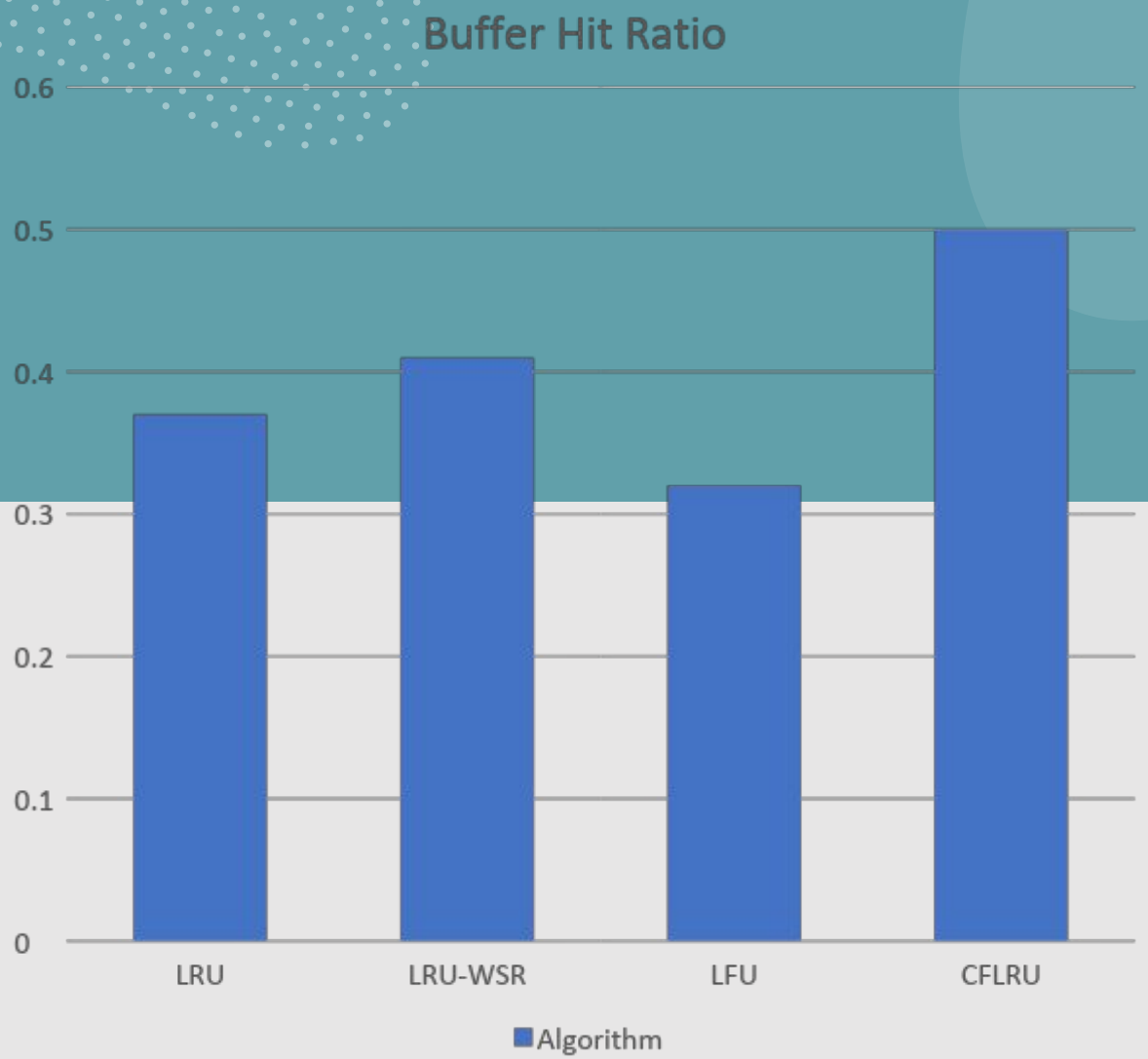
Changing parameter e(percent reads) and making it more write focused

- Parameters :
- b = 150 (no of pages in buffer pool)
- n = 1500 (no of pages in disk)
- x = 7500 (no of operations for workload to execute)
- e = 40 (percentage reads)
- a = 0..3 (type of eviction algorithm)
- s = 90 (skew percentage)
- d = 10 (skewed data percentage)



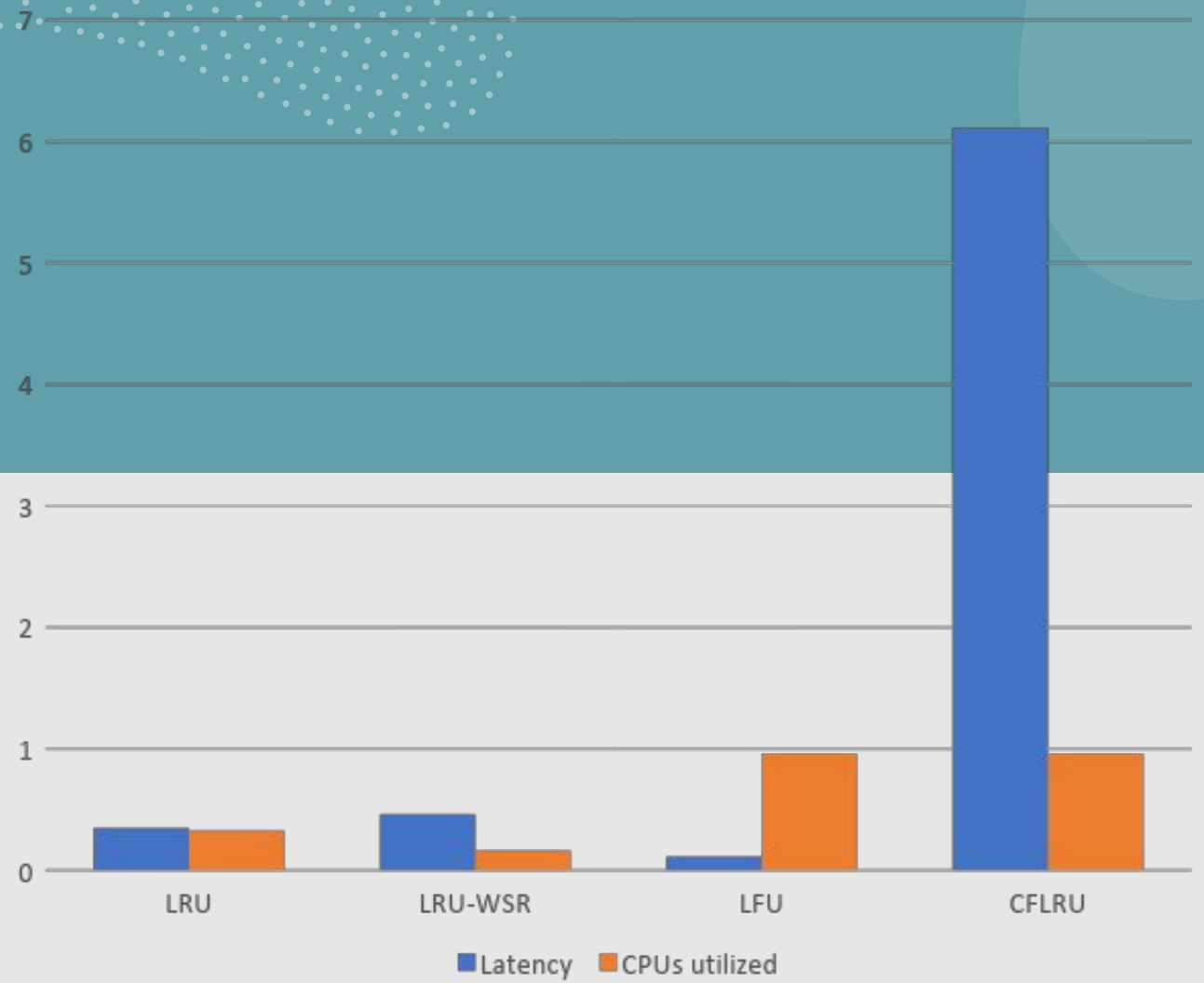
Changing parameter s(skew percent) while keeping d(skewed data percent) unchanged

- Parameters :
- b = 150 (no of pages in buffer pool)
- n = 1500 (no of pages in disk)
- x = 7500 (no of operations for workload to execute)
- e = 60 (percentage reads)
- a = 0…3 (type of eviction algorithm)
- s = 50 (skew percentage)
- d = 10 (skewed data percentage)



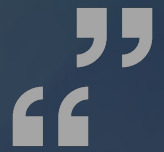
Keeping parameters unchanged, and calculating latency

- Machine Details :
- RAM = 1GiB, vCPUs = 1, Size = Standard B1s
- Parameters :
- b = 150 (no of pages in buffer pool)
- n = 1500 (no of pages in disk)
- x = 7500 (no of operations for workload to execute)
- e = 60 (percentage reads)
- a = 0…3 (type of eviction algorithm)
- s = 50 (skew percentage)
- d = 10 (skewed data percentage)

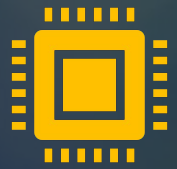


Conclusion

CFLRU seems to be the best option for an eviction policy when latency is ignored. In all our experimental runs, CFLRU had the best buffer hit, buffer miss ratio.



When considering latency, LFU turned out to be the best option, as it plainly involves returning the slot based on minimum number of references.



LRU stands to be the best eviction policy when considering ease of implementation, latency, and CPU utilization.

LRU-WSR seems like a great option when the workload is a write focused rather than read focused. Its hit ratio is only less than CFLRU, but when compounded with latency, LRU-WSR seems like a winner.

Our advice

- Start Early
- Use office Hours
- Modularise the code
- Do your research
- Stack Overflow!

Thank You!

Aditya Pal, Harsh Mutha, Manind Gera.