



# Bufferpool Implementation

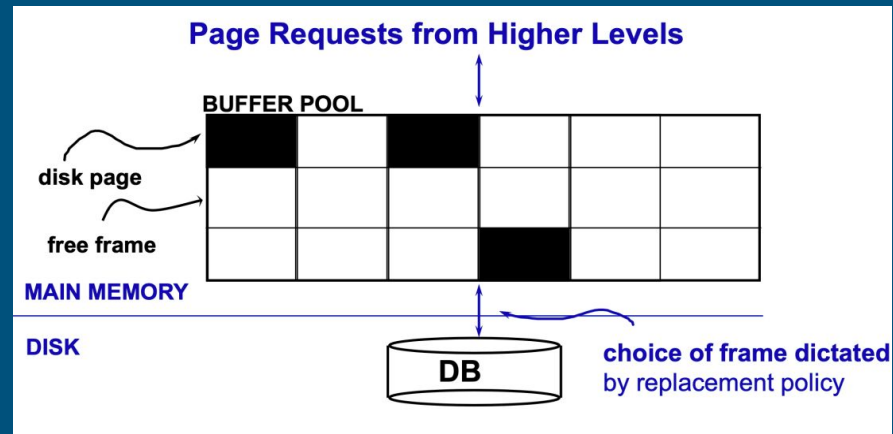
---

Mia Li, Samir Farhat Dominguez,  
Stephany Yipchoy



# Introduction

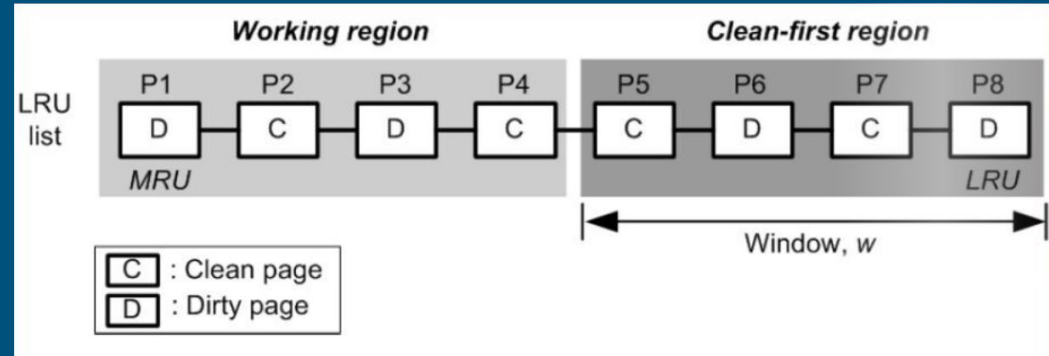
- Bufferpool overview
  - Buffer Hit
  - Buffer Miss
  - Dirty Bits
  - Eviction Strategies
- Optimize for different workloads to **maximize read and write performance** by maximizing hits





# CFLRU Policy

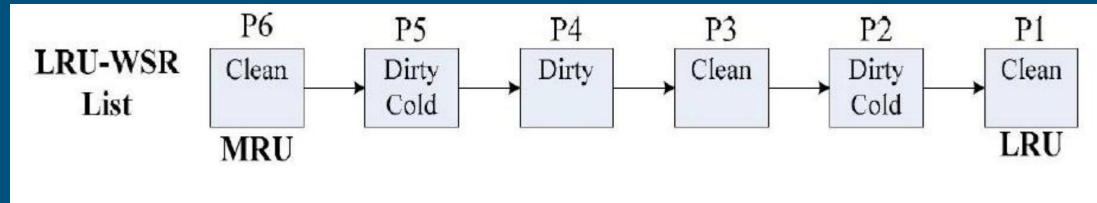
- Separate LRU list into 2 parts
- Keep a certain amount of dirty pages in cache to reduce the number of flash write operations
- we opted for the window to be  $\frac{1}{3}$  of the current amount of pages in the buffer



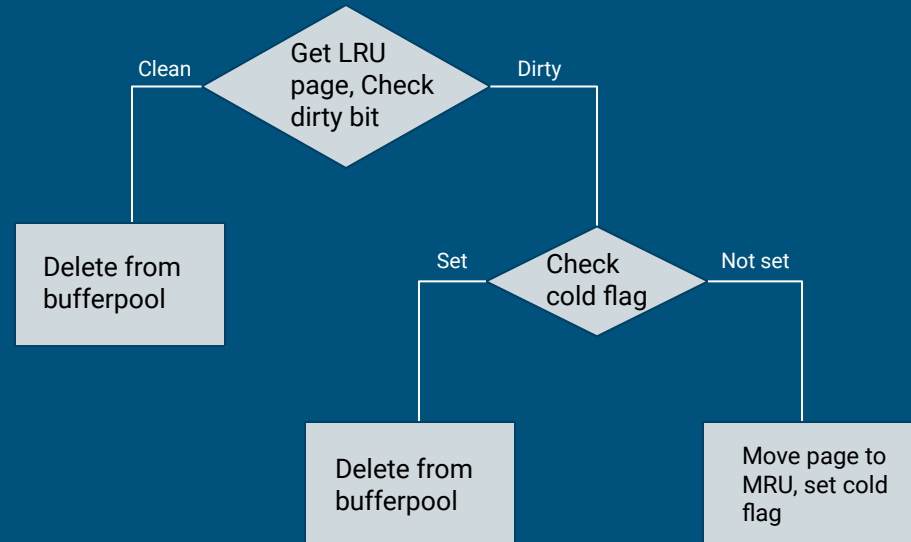
LRU: p8 -> p7 -> p6 -> p5

CFLRU: p7 -> p5 -> p8 -> p6

# LRU-WSR Policy



- Second chance algorithm -> Cold flag can only be set on the second time (reordered writing sequences)
- The only difference between those two policies is that LRU-WSR assign each page with a bit flag called “cold-flag”
- In bufferpool we store **tuple(page id, dirty bit, cold flag)**



# FIFO Policy

- First in first out, implemented by queue
- when a page hits:
  - LRU will move this page to the MRU position
  - FIFO will make no changes to the queue.
- Simple but not efficient for large number of pages → the operating system keeps track of all pages in the memory in a queue

1	2	3	4	5	1	3	1	6	3	2	3
---	---	---	---	---	---	---	---	---	---	---	---

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	6	6	6	6
			4	4	4	4	4	4	3	3	3

M	M	M	M	M	M	H	H	M	M	M	H
---	---	---	---	---	---	---	---	---	---	---	---

M = Miss  
H = Hit

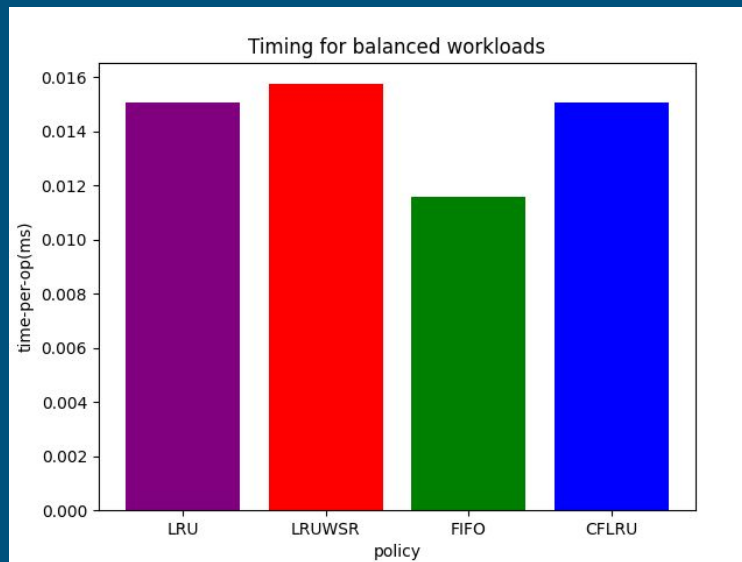
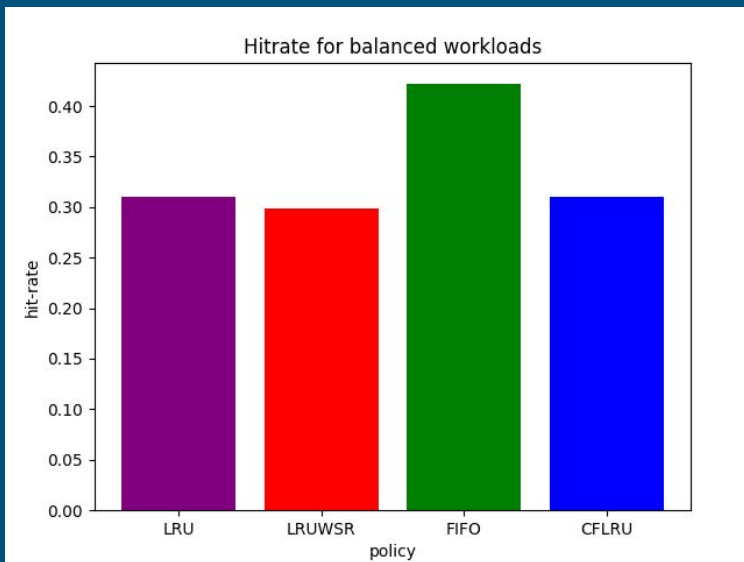
# Disk Implementation

- Disk populated with arbitrary random bytes
- Writes and reads with seekg
  - One continuous character when written over (except last character is '\n')
  - One page per line, no sectors because just a quick computation and isn't productive to objectives
- Keeping file open may not be true reflection of disk access unless disk size exceeds RAM
  - We saw only a slight increase in timings
- Disk object belongs to Buffer class

```
498 |aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
499 |/vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
500 |:cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
501 |:puvfdsaiabbplclamiexcldhcrrxommiktncvtywzvidgkpgrootl
502 |jzvudyshlvvqgpiighbayzewdkvjcdrlrhiwisqpnngwdqclrelsiiv
503 |wcyuurewrcrnsoqkkgkygucflnjvnhxneueksibbytqoscvccbry
504 |wnaowqcmghafzircxrddyovskzuxsdvqtveroidupfbqnuvnoylmm;
505 |.delkiiidujkjnysgtqhtudykqrxzrlhwqsgacrfyaqhpqbxjsecmk
506 |kiyysxytlyqzqkvupgbkrmmxpsistsasssrmkszqaprmfnhwvkgf
507 |eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
508 |eenmtybfsjnonziclcgprzpcylrckmqgavlayrthejwfryrvzmq;
509 |vedaabflvydvicgwmqvetsopopvlnenjlqlltqwrqbmysvrlswejc
510 |dahbpzambqvcvpgawnrxuybvbnixxazebfrcfjualpprsngjlod;
511 |nwrwdpjoboirlmrlidcurwsjuqcwdaocdfnuhbcyqlukwhuzlc
512 |waszswaultfllhgebwnlxqhwczbwksukmtcittytpehclngocccbc
513 |ssssssssssssssssssssssssssssssssssssssssssssssssssssss
514 |.zlmodyvqiprmiitnrxsgnhtfqrrefgqrvpvlzkdlsntncnkuvz
515 |sisiupyvrjzgawroowmeurqzovoylsvfapnufmswxrezqvnqgbkpv
516 |tmcgyfmojkmapzpmhdnmzorvgoizcwuqbapiofartqqjscqxcqc
517 |hutslvubaedujuimtmkynqcuytpocyjtrbgoyirellohtaahmawc
518 |hgddzueaafwirllyfluvwasgpgzufykdggjsbosbvploajtnfjqfl
519 |wrwbgfeweelapfjqdhtyduqgythckeaddbjighmmsodzxwfgpdjh;
520 |aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
521 |mpwgqjkhimxrmezxhknogdlmansxveohvknuzueosvcywigoworl
522 |cymkqofwfbzflompatrkljmwcbzknbiblpszhacgfrvcgvvxh;
523 |icuxeppvoavoygcymzlwuepoi vzlhhcplzmqodleeivcpxddybqsh
524 |bmrjtjgsrypszjsuugi hblpritxuczkaaxuvi cpzagsbpkylsgtvkr
525 |hstszcwdjhqni hfdlctqulfbkekpigi cbjgxoqee fohovmbzfsv
526 |yhlaxejcyzaekhxzgm dtombelxmLvkevrqyqwhxtitbuaatjmxcaj
527 |lapprienttllklclzchbkbwmlgoxybunrahftuxfejuchuzvfl
```

# Experiments

---





# Challenges and Lessons Learned

---

- Disk management
  - At first only appending seemed possible without a complete overwrite
- Seg faults in overall simulation run script
- Construct the bufferpool to be more adaptive
- Learned that it's much easier to have a small workload and know exactly how it should perform
- Read other research for guidance

# Conclusions

- Implemented: FIFO, LRU, CFLRU, LRU-WSR, and Disk Functionality
- For balanced workloads, the simplest approach is also the best
- Intended extensions of this work:
  - Experiment with read heavy and write heavy workloads
  - Add more flexibility to the buffer with more parameters

```
You, 6 days ago | 1 author (You)
15 namespace bufmanager {
    You, 6 days ago | 1 author (You)
16     class Buffer {
17         // This class maintains specific property of the buffer.
18         // You definitely need to modify this part
19         // You need to add more variables here for your implementation. For example, current
20
21     private:
22         Buffer(Simulation_Environment* _env);
23         static Buffer* buffer_instance;
24
25     public:
26         //initizate bufferpool with <page size, dirty page bit>
27         vector< pair<int, bool> > bufferpool;
28         //initizate bufferpool for LRU WSR with <page size, dirty page bit, cold flag>
29         vector< tuple<int, bool, bool> > bufferpool_wsr;
30         //deck as the lru candidate list
31         deque<int> lru_candidate;
32         //for cflru: going to store either pageid of clean or -1
33         deque<int> fifo_candidates;
34         static long max_buffer_size; //in pages
35
36         static Buffer* getBufferInstance(Simulation_Environment* _env);
37         // statistics to track
38         static int buffer_hit;
39         static int buffer_miss;
40         static int read_io;
41         static int write_io;
42         static char disk_write_char;
43         static std::chrono::duration <double, milli> timing;
44
45         // Page replacement algorithm
46         int LRU();
47         int LRUWSR();
48         int FIFO();
49         int CFLRU();
50
51         // Disk parameters
52         std::fstream disk;
53         static int pageSize;
54
55         // Statistics collection
56         static void writeResults();
57         static int printBuffer();
58         static int printStats();
59     };
60
```