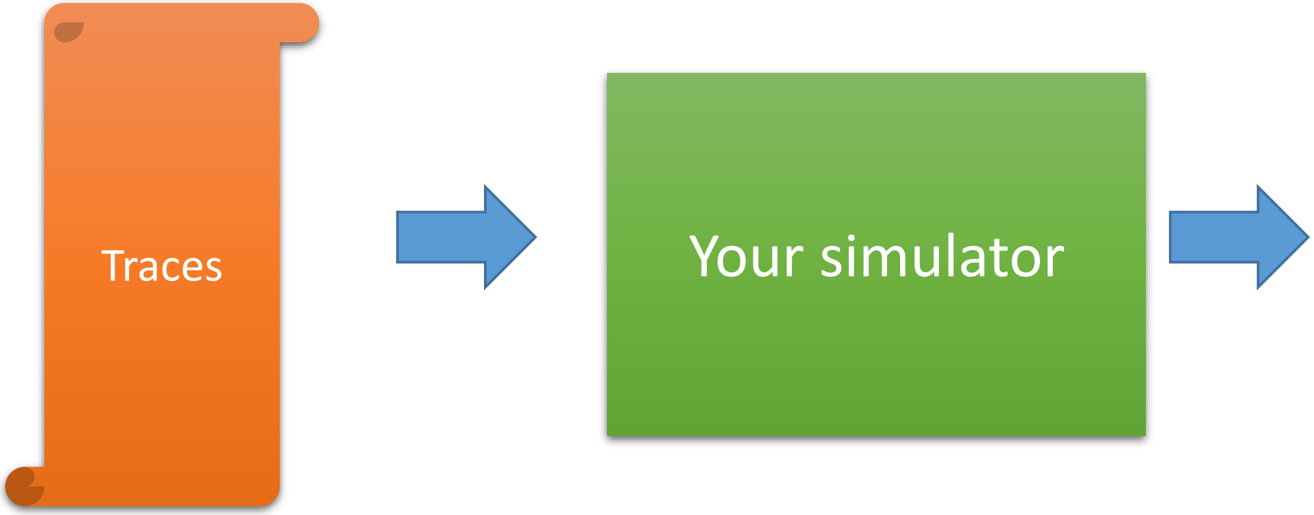
Operating Systems
Programming Assignment #5

Page Replacement Simulation: LRU and CFLRU

Prof. Li-Pin Chang
CS@NYCU

Simulation



Traces

Your simulator

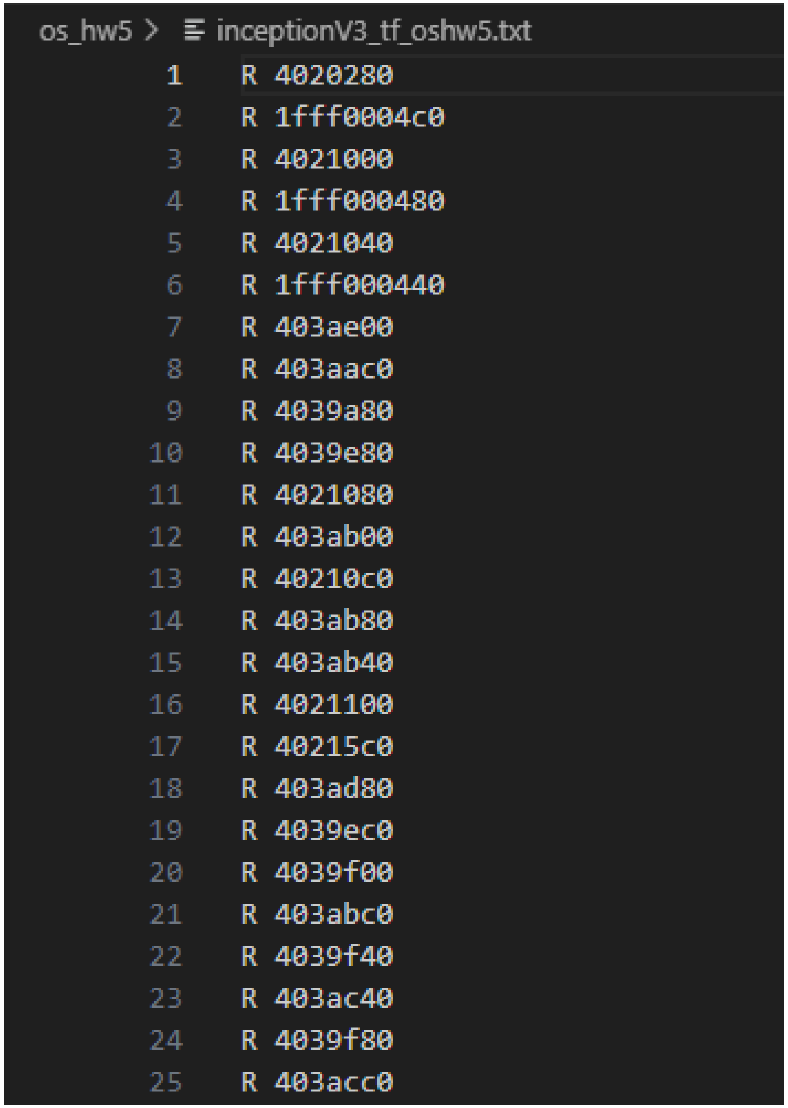
Hit count
Miss count
Write back count
…

Trace File Format

Memory traces of InceptionV3, an DNN running on TensorFlow for image recognition

[Op] [Byte-offset]

- Separated by a space
- Op is either “R” or “W”
- 64-bit byte offset in hex
- Convert the byte offset into page number
    - Cache unit = page size = 4KB
- Total references = 49228943



```
R 4020280
R 1fff0004c0
R 4021000
R 1fff000480
R 4021040
R 1fff000440
R 403ae00
R 403aac0
R 4039a80
R 4039e80
R 4021080
R 403ab00
R 40210c0
R 403ab80
R 403ab40
R 4021100
R 40215c0
R 403ad80
R 4039ec0
R 4039f00
R 403abc0
R 4039f40
R 403ac40
R 4039f80
R 403acc0
```

Page Replacement(LRU)

- Example: Frame #=2

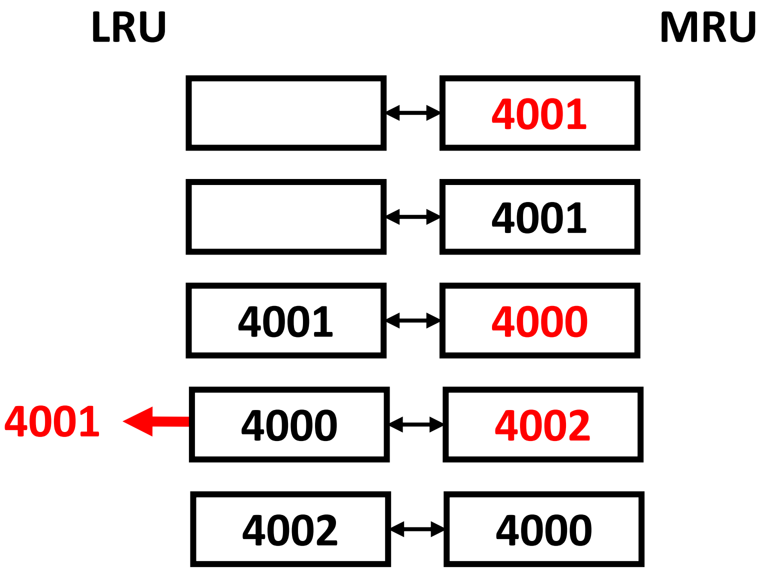
4001 (miss)

4001 (hit)

4000 (miss)

4002 (miss)

4000 (hit)



LRU MRU

4001

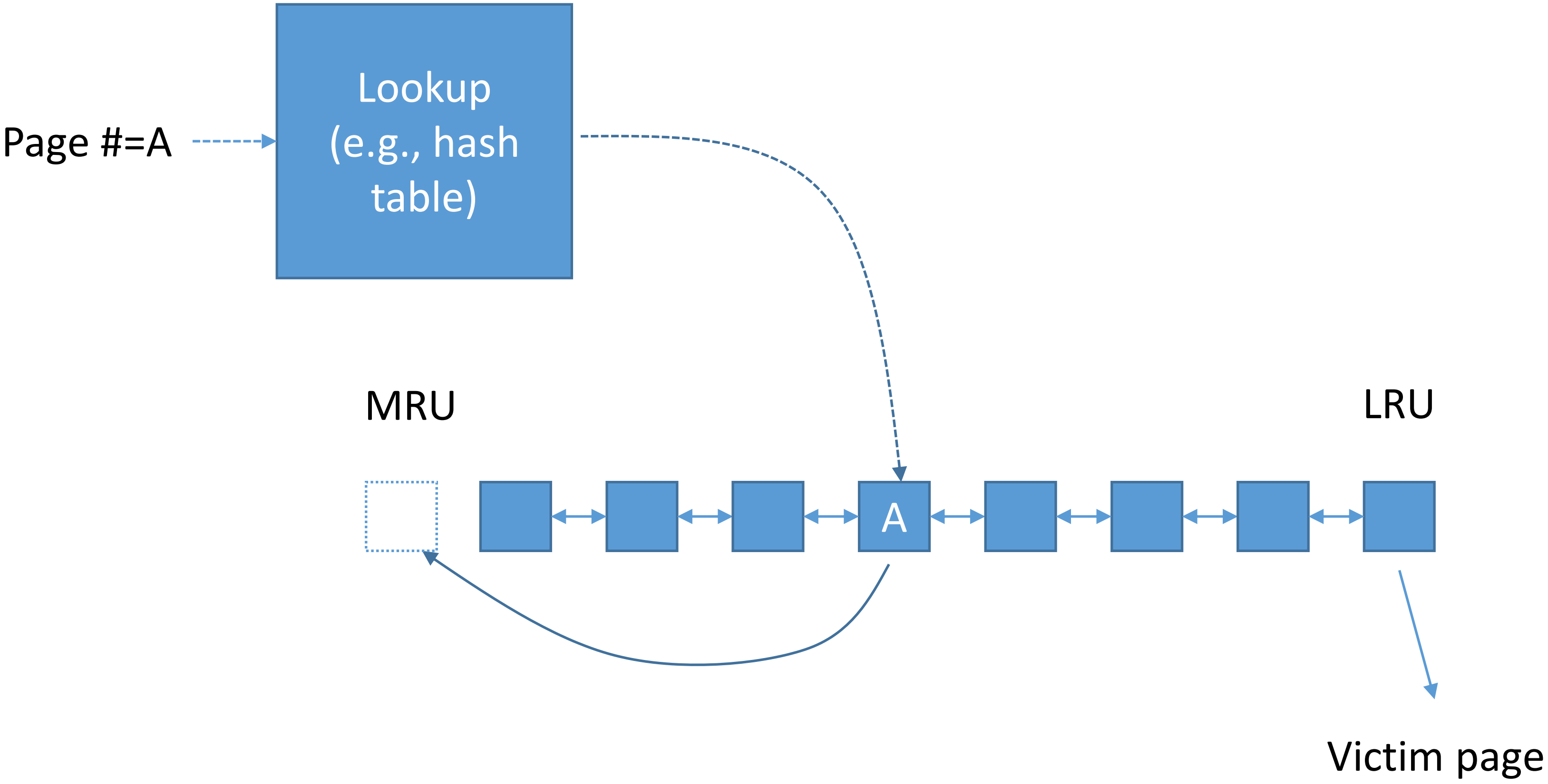
4001

4001 4000

4001 4000 4002

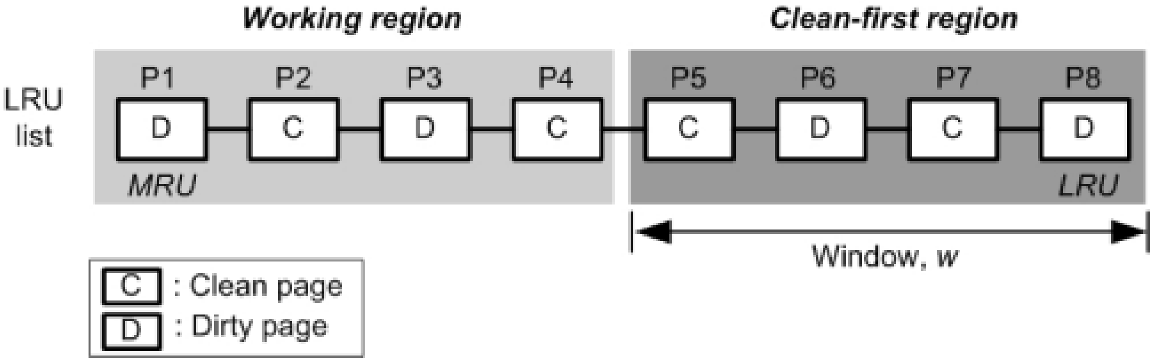
4002 4000

Reference Design for LRU



Clean-First LRU (CFLRU)

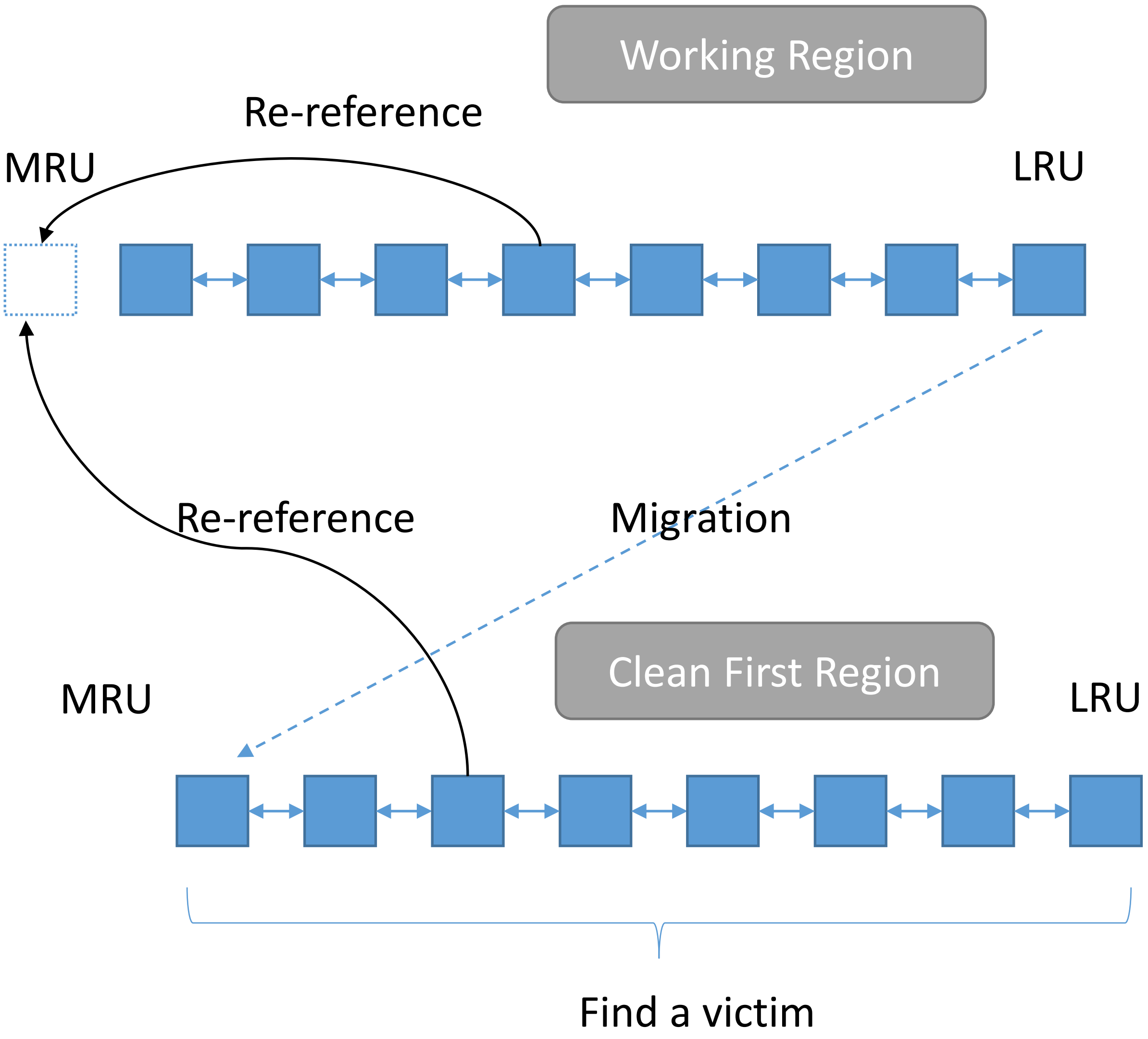
- Like the enhanced second chance algorithm, CFLRU favors clean pages to reduce I/O-write overhead
- Dividing the LRU list into a working region and a clean-first region
- If the working region is full, migrate an LRU page (P4 here) to the clean-first region
- On replacement, from the clean-first region
  1. Select the LRU clean page (P7 here)
  2. If no clean page, select the LRU (dirty) page



Clean-First LRU (CFLRU)

- Based on your LRU implementation
  - Set the clean-first region size to ¼ of total size
  - E.g., cache size=4096 pages → working region size = 3072 pages
- Details
  - The write back count ++ only when a dirty page is evicted
  - A write-generated new page is a dirty one
- Two design issues
  1. How do you control the clean-first region size?
  2. How do you efficiently find a clean page in the clean-first region? (don’t do linear search)

A Reference Design



Page Cache Operations

- Page lookup in LRU/CFLRU
- Find a victim in the clean-first region for CFLRU

- Do not use linear search!!!
  - You will receive a grade penalty if you do
  - Implement your own search, or reuse any existing libraries/classes for searching
  - Duplication in this part does not count
  - Note: do not use STL map/unordered map, too slow

Procedure

1. Read the trace file and load parsed information into a memory buffer
2. Algorithm=LRU
3. For (frame # = 4096, 8192, 16384, 32768 and 65536) do
   - Run simulation
   - Print out page hit count, miss count, page fault ratio, and page write count
4. Print out the total elapsed time of Step 3

5. Algorithm=CFLRU
6. For (frame # = 4096, 8192, 16384, 32768 and 65536) do
   - Run simulation
   - Print out page hit count, miss count, page fault ratio, and page write count
7. Print out the total elapsed time of Step 6

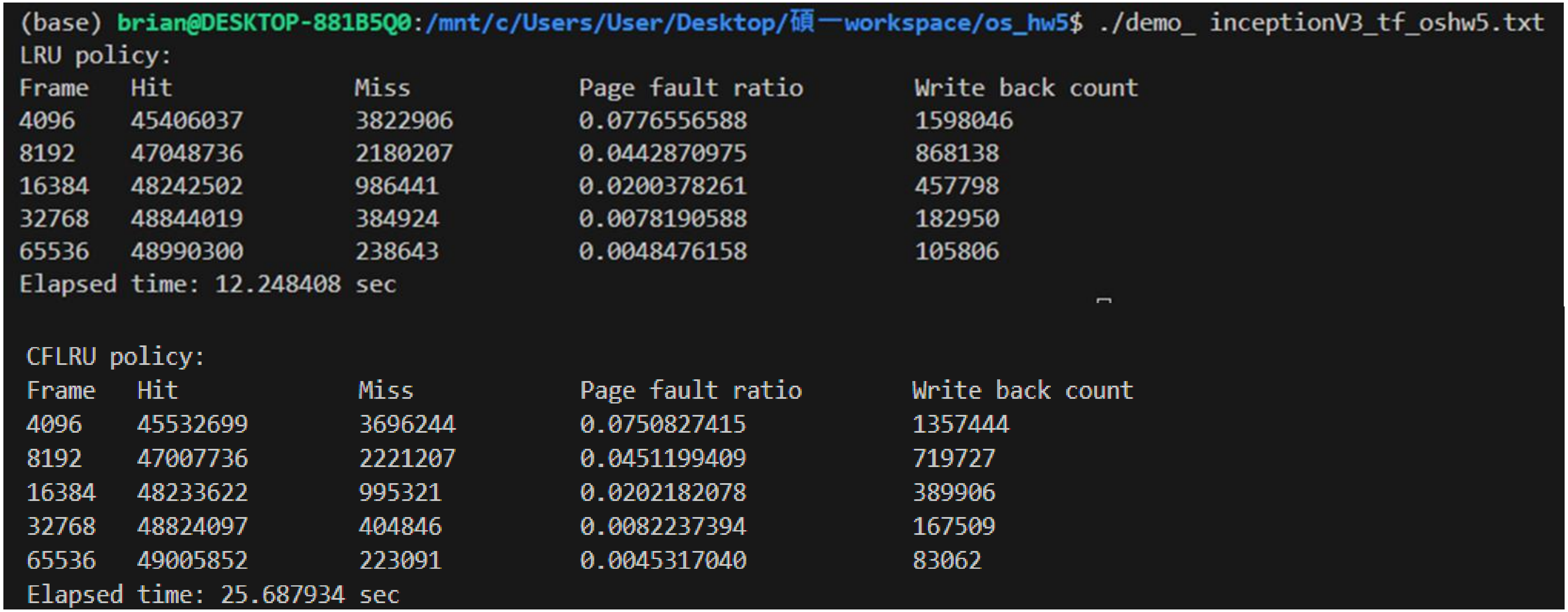
Output Format

```
LRU policy: (\n)
Frame     (\t)     Hit      (\t\t)     Miss     (\t\t)     Page fault ratio  (\t)   Write back count(\n)
4096      (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
8192      (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
16384     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
32768     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
65536     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
Total elapsed time %.6f sec(\n) (\n)

CFLRU policy: (\n)
Frame     (\t)     Hit      (\t\t)     Miss     (\t\t)     Page fault ratio  (\t)   Write back count(\n)
4096      (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
8192      (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
16384     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
32768     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
65536     (\t)     %d       (\t)       %d       (\t\t)     %.10f            (\t\t)  %d                 (\n)
Total elapsed time %.6f sec(\n) (\n)
```

Output Example

Here, gcc use -o0



```
(base) brian@DESKTOP-881B5Q0:/mnt/c/Users/User/Desktop/碩一workspace/os_hw5$ ./demo_ inceptionV3_tf_oshw5.txt
LRU policy:
Frame   Hit             Miss            Page fault ratio        Write back count
4096    45406037        3822906         0.0776556588            1598046
8192    47048736        2180207         0.0442870975            868138
16384   48242502        986441          0.0200378261            457798
32768   48844019        384924          0.0078190588            182950
65536   48990300        238643          0.0048476158            105806
Elapsed time: 12.248408 sec

CFLRU policy:
Frame   Hit             Miss            Page fault ratio        Write back count
4096    45532699        3696244         0.0750827415            1357444
8192    47007736        2221207         0.0451199409            719727
16384   48233622        995321          0.0202182078            389906
32768   48824097        404846          0.0082237394            167509
65536   49005852        223091          0.0045317040            83062
Elapsed time: 25.687934 sec
```

Your results must be exactly the same as shown here (except time)
Hit: requested page is in the cache; miss: otherwise.
Fault ratio = miss # / total reference #
Write back count = how many dirty pages have been evicted (and requiring write-back)

Performance Reference

- 12th Gen Intel(R) Core(TM) i7-12700K
  - 12 seconds for LRU
  - 26 seconds for CFLRU
  - InceptionV3
- Use this to convert your execution time based on the relative CPU performance (single thread)
- If your execution time is way too long, you will receive a score penalty

Correctness

- Except the elapsed time, your results should be exactly the same as in the output example
- The TAs will prepare another workload to validate your implementation
- LRU 60% CFLRU 40%

- Once again, do not use linear search in anywhere of your program. Your program will definitely run too slow and you will receive a score penalty
- Tip: Do not use map/unordered_map from STL as they are poorly implemented and very slow

More details

- Total request #: <= 10^8 references
- Byte offset: 64 bits
- The path+file name of the trace file is an argument of your program (see the screen shot), do not hard-coding the pathname of the trace file
- You can read the trace file into a memory buffer for fast reuse
- Use gettimeofday() to get the total elapsed time

Testing OS Environment

- Ubuntu 22.04+
- Physical installation, VM, or WSL

Credits

- 呂柏勳 許登豪 helped design this project
- Direct all questions to the current TAs