1. **Згенеруйте випадкові адреси з такими аргументами: -s 0 -n 10, -s 1 -n 10 і -s 2 -n 10. Змініть політику з FIFO на LRU на OPT. Обчисліть, чи є кожен доступ у трасах адреси збігами чи промахами.**

./paging-policy.py -s 0 -n 10 -c

./paging-policy.py -s 0 -n 10 -c --policy=LRU

./paging-policy.py -s 0 -n 10 -c --policy=OPT

1. **Для кешу розміром 5 згенеруйте потоки посилань на найгіршу адресу для кожної з таких політик: FIFO, LRU та MRU (потоки посилань на найгіршу ситуацію спричиняють найбільшу кількість промахів). Для найгіршого випадку еталонних потоків, наскільки більший кеш потрібен, щоб значно підвищити продуктивність і наблизитися до OPT?**

./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=FIFO --cachesize=5 -c

./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=LRU --cachesize=5 -c

./paging-policy.py --addresses=0,1,2,3,4,5,4,5,4,5,4,5 --policy=MRU --cachesize=5 -c

Один.

1. **Згенеруйте випадкову трасування (використовуйте python або perl). Як ви очікуєте, що різні політики працюватимуть на такому трасуванні?**

./paging-policy.py -s 0 -n 10 -c

FINALSTATS hits 1 misses 9 hitrate 10.00

./paging-policy.py -s 0 -n 10 -c --policy=LRU

FINALSTATS hits 2 misses 8 hitrate 20.00

./paging-policy.py -s 0 -n 10 -c --policy=OPT

FINALSTATS hits 4 misses 6 hitrate 40.00

./paging-policy.py -s 0 -n 10 -c --policy=UNOPT

FINALSTATS hits 0 misses 10 hitrate 0.00

./paging-policy.py -s 0 -n 10 -c --policy=RAND

FINALSTATS hits 0 misses 10 hitrate 0.00

./paging-policy.py -s 0 -n 10 -c --policy=CLOCK

FINALSTATS hits 1 misses 9 hitrate 10.00

1. **Тепер згенеруйте трасування з певною локальністю. Як можна створити такий слід? Як на ньому працює LRU? Наскільки LRU краще за RAND? Як працює CLOCK? Як щодо CLOCK з різною кількістю тактових бітів?**

$ ./generate-trace.py

[3, 0, 6, 6, 6, 6, 7, 0, 6, 6]

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=LRU -c

FINALSTATS hits 6 misses 4 hitrate 60.00

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=RAND -c

FINALSTATS hits 5 misses 5 hitrate 50.00

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=CLOCK -c -b 2

Access: 3 MISS Left -> [3] <- Right Replaced:- [Hits:0 Misses:1]

Access: 0 MISS Left -> [3, 0] <- Right Replaced:- [Hits:0 Misses:2]

Access: 6 MISS Left -> [3, 0, 6] <- Right Replaced:- [Hits:0 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:1 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:2 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:3 Misses:3]

Access: 7 MISS Left -> [3, 6, 7] <- Right Replaced:0 [Hits:3 Misses:4]

Access: 0 MISS Left -> [3, 7, 0] <- Right Replaced:6 [Hits:3 Misses:5]

Access: 6 MISS Left -> [7, 0, 6] <- Right Replaced:3 [Hits:3 Misses:6]

Access: 6 HIT Left -> [7, 0, 6] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4 misses 6 hitrate 40.00

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=CLOCK -c -b 0

Access: 3 MISS Left -> [3] <- Right Replaced:- [Hits:0 Misses:1]

Access: 0 MISS Left -> [3, 0] <- Right Replaced:- [Hits:0 Misses:2]

Access: 6 MISS Left -> [3, 0, 6] <- Right Replaced:- [Hits:0 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:1 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:2 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:3 Misses:3]

Access: 7 MISS Left -> [3, 0, 7] <- Right Replaced:6 [Hits:3 Misses:4]

Access: 0 HIT Left -> [3, 0, 7] <- Right Replaced:- [Hits:4 Misses:4]

Access: 6 MISS Left -> [3, 7, 6] <- Right Replaced:0 [Hits:4 Misses:5]

Access: 6 HIT Left -> [3, 7, 6] <- Right Replaced:- [Hits:5 Misses:5]

FINALSTATS hits 5 misses 5 hitrate 50.00

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=CLOCK -c -b 1

Access: 3 MISS Left -> [3] <- Right Replaced:- [Hits:0 Misses:1]

Access: 0 MISS Left -> [3, 0] <- Right Replaced:- [Hits:0 Misses:2]

Access: 6 MISS Left -> [3, 0, 6] <- Right Replaced:- [Hits:0 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:1 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:2 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:3 Misses:3]

Access: 7 MISS Left -> [3, 0, 7] <- Right Replaced:6 [Hits:3 Misses:4]

Access: 0 HIT Left -> [3, 0, 7] <- Right Replaced:- [Hits:4 Misses:4]

Access: 6 MISS Left -> [3, 7, 6] <- Right Replaced:0 [Hits:4 Misses:5]

Access: 6 HIT Left -> [3, 7, 6] <- Right Replaced:- [Hits:5 Misses:5]

FINALSTATS hits 5 misses 5 hitrate 50.00

$ ./paging-policy.py --addresses=3,0,6,6,6,6,7,0,6,6 --policy=CLOCK -c -b 3

Access: 0 MISS Left -> [3, 0] <- Right Replaced:- [Hits:0 Misses:2]

Access: 6 MISS Left -> [3, 0, 6] <- Right Replaced:- [Hits:0 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:1 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:2 Misses:3]

Access: 6 HIT Left -> [3, 0, 6] <- Right Replaced:- [Hits:3 Misses:3]

Access: 7 MISS Left -> [3, 6, 7] <- Right Replaced:0 [Hits:3 Misses:4]

Access: 0 MISS Left -> [6, 7, 0] <- Right Replaced:3 [Hits:3 Misses:5]

Access: 6 HIT Left -> [6, 7, 0] <- Right Replaced:- [Hits:4 Misses:5]

Access: 6 HIT Left -> [6, 7, 0] <- Right Replaced:- [Hits:5 Misses:5]

FINALSTATS hits 5 misses 5 hitrate 50.00

1. **Використовуйте програму на зразок valgrind, щоб інструментувати реальну програму та генерувати потік посилань на віртуальну сторінку. Наприклад, запуск valgrind --tool=lackey --trace-mem=yes ls виведе майже повне посилання на кожну інструкцію та посилання на дані, створені програмою ls. Щоб зробити це корисним для наведеного вище симулятора, вам доведеться спочатку перетворити кожне посилання на віртуальну пам’ять у посилання на номер віртуальної сторінки (це робиться шляхом маскування зміщення та зміщення отриманих бітів вниз). Наскільки великий кеш потрібен для трасування програми, щоб задовольнити велику частку запитів? Побудуйте графік його робочого набору зі збільшенням розміру кешу.**

$ valgrind --tool=lackey --trace-mem=yes ls &> ls-trace.txt

// Рахування до 3, ^C

$ ./transform.py // перетворення на VPN

$ ./run.sh // отримання деяких даних

$ ./plot.py

