CSCI 5103 – Project 2

Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

Purpose of the experiments and the experimental setup.

The primary purpose of the experiments and the experimental setup is to evaluate the performance of various page replacement strategies. The reason why this is important is because, in every computer involving either a single core or a multi core processors, to attain a highly efficient fast and usable computers, memory access should be efficiently planned so as to minimize the latency involved in fetching data from the disk to main memory. Since main memory is something which is very limited when compared to disk memory, we need to come up with effective strategies to make use of the limited main memory using some kind of reusability. This is the prime reason for the paging, page table, and strategies to come up. Since there are multiple such strategies, there is a significant scope for some of them to perform better over others depending on the work load and its pattern. For the same reason, in this project, we are evaluating three different strategies (rand, fifo and custom) in three different workload patterns (sort, scan and focus) to see which of them performs better in which type of scenarios.

Machines used for experiments: suman.cs.umn.edu, csel-kh1262-08.cselabs.umn.edu

Commands to test: Explicitly test each case: *./virtmem <npages> <nframes> <rand|fifo|custom> <sort|scan|focus>*

To test all wide range of results, I added a shell script which changes the number of frames from 100 to 10 and displays results on command line. You can tweak the values of pages and frames used. Run '*bash all_tests.sh*' to use this.

Custom page replacement algorithm:

Our custom replacement algorithm is a variant of the FIFO algorithm. The order in which the custom algorithm replaces the pages is given by

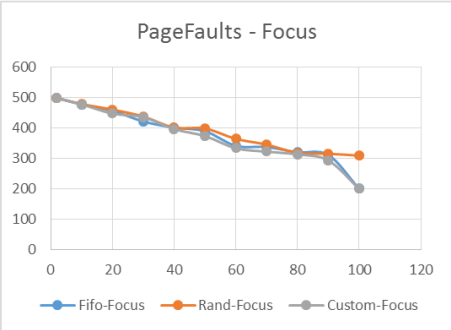
1. Looks for an unused frame if there is one. If there is, we replace that frame.
2. Looks for a frame which is not marked dirty. If there is one, then it looks for the best possible frame which can be replaced, i.e., some frame which has been written oldest, i.e., hoping that there is nothing much more to be written to it when compared to the latest ones which might have a lot of data to be written. Then we replace that selected frame.
3. If none of the above are true, then it removes the page according to the FIFO order i..e., replace the page which has been at the head of our FIFO queue.

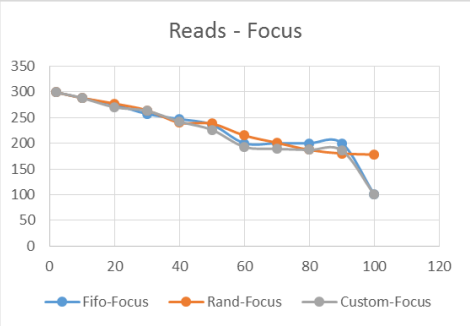
Summary of Results:

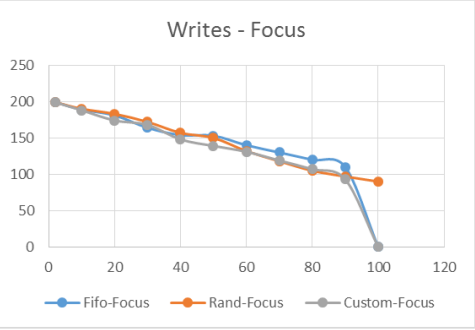
1. Focus Program
   We see that, for the focus program, the page faults/disk read-writes are the highest for rand, followed by FIFO, followed by custom algorithm. However in some instances, we see that either they overlap or the order is altered, this could be possible because of the way random behaves, i.e., sometimes we might get lucky in a way that the pages chosen by random are either free or which involves no disk writes etc. Therefore in such cases alone we kind of see that rand is doing

Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

better over the other algorithms. However, if we see the average trend line, we will get to see the original behavior of rand which is inefficient when compared to the other strategies.
The reason for the first behavior is because, since rand does page replacement very randomly, we might not be able to optimize in many ways like, choosing free pages if available, or choosing pages which are not marked dirty or which have not seen any writes, or choosing the oldest page which has been used, etc. On the other hand, FIFO does better than random because, it is first trying to see for the oldest page being used to replace the new requests. Also, the way focus program is trying to access the pages, accessing data randomly (#41 program.c using rand()), we will have a certain uniform distribution involved, we might see the same page being accessed only after a while because of the uniform random distribution. Therefore we kind of see that FIFO fits better than rand in this case. Our custom algorithm does a little better over the FIFO because, we are trying to see if there are any frames which are not marked dirty i.e., any frames with no write activity yet even before we use FIFO. This way certain disk operations can be reduced when compared to FIFO. However, this does not significantly outperform FIFO, we need to increase the complexity involved with the custom algorithm to use some kind of heuristic based approach, but as suggested in the Project, we have chosen something simple and easy to implement.







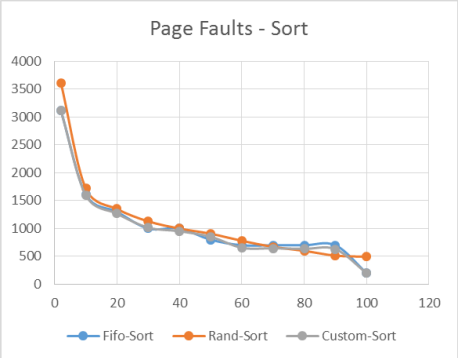
2. Sort Program:

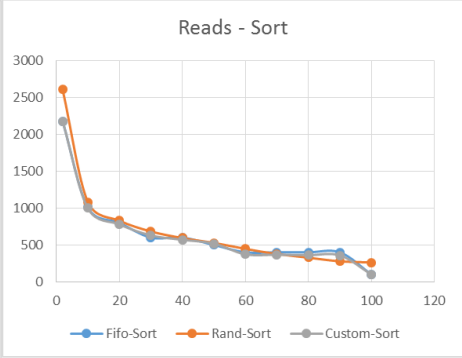
We see that, for the sort program, the page faults/disk read-writes are the highest for rand, followed by FIFO, followed by custom algorithm. However in some instances, we see that either they overlap or the order is altered, this could be possible because of the way random behaves, i.e., sometimes we might get lucky in a way that the pages chosen by random are either free or which involves no disk writes etc. Therefore in such cases alone we kind of see that rand is doing better over the other algorithms. However, if we see the average trend line, we will get to see the original behavior of rand which is inefficient when compared to the other strategies.
The reason for the first behavior is because, since rand does page replacement very randomly, we might not be able to optimize in many ways like, choosing free pages if available, or choosing pages which are not marked dirty or which have not seen any writes, or choosing the oldest page which has been used, etc. On the other hand, FIFO does better than random because, it is first trying to see for the oldest page being used to replace for the new requests. Also, since the way

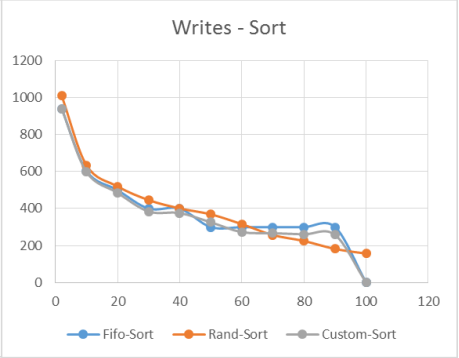
CSCI 5103 – Project 2

Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

sort program is trying to access the pages, sorting using qsort, we will have a set of comparisons around a pivot in a sequential fashion, i.e., page accessed now may not be accessed again until the next iteration where we choose pivot, because of which we see that FIFO fits better than random. Our custom algorithm does a little better over the FIFO because, we are trying to see if there are any frames which are not marked dirty i.e., any frames with no write activity yet even before we use FIFO. This way certain disk operations can be reduced when compared to FIFO. However, this does not significantly outperform FIFO, we need to increase the complexity involved with the custom algorithm to use some kind of heuristic based approach, but as suggested in the Project, we have chosen something simple and easy to implement.





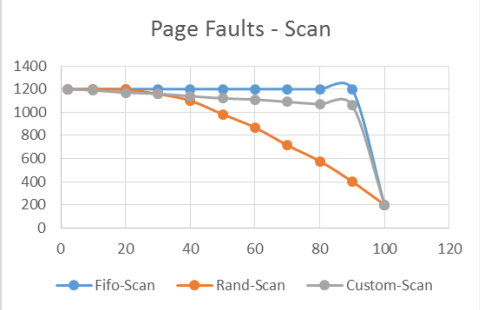


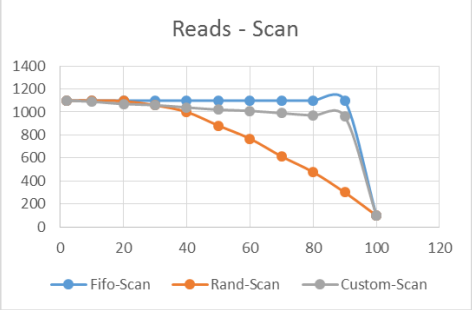
3. Scan Program

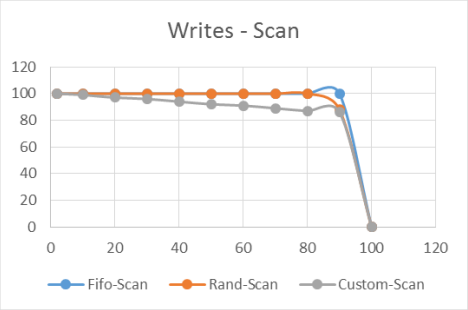
We see that, the page faults/disk read writes are the highest for FIFO, followed by custom, followed by rand algorithm. Although in general, we expect that rand should not do well, the reason why rand does better and FIFO not better is mentioned below. The order in which the pages are being accessed are in such a way that are looped/cyclically read (#85 loops in program.c), But then, FIFO replaces the pages exactly in the same way i.e., exactly in a looped way based on the FIFO order, therefore almost every time we access a page, we are certain that FIFO might have replaced it with some other page already. Therefore, this leads to a lot of disk reads/writes degrading the behavior of FIFO. Here, our custom algorithm does a little better over the FIFO because, we are trying to see if there are any frames which are not marked dirty i.e., any frames with no write activity yet even before we use FIFO. This way certain disk operations can be reduced when compared to FIFO. However, this does not significantly outperform FIFO, we need to increase the complexity involved with the custom algorithm to use some kind of heuristic based approach, but as suggested in the Project, we have chosen something simple and easy to implement. Since rand at least does page replacement at random, we might not be able to see this exactly complementary behavior of page replacements, which is why rand seems to be doing better when compared to the other algorithms.

CSCI 5103 – Project 2

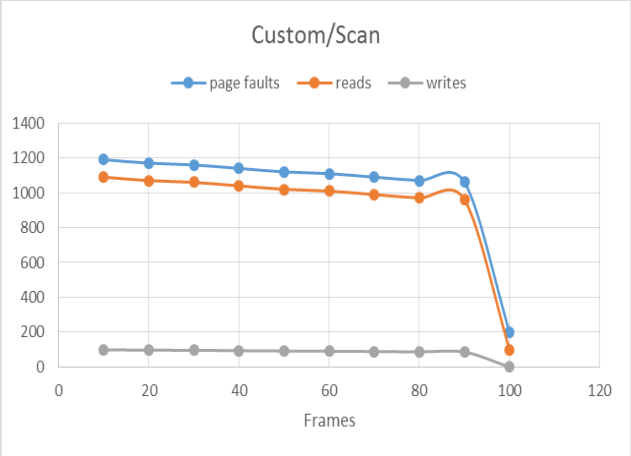
Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

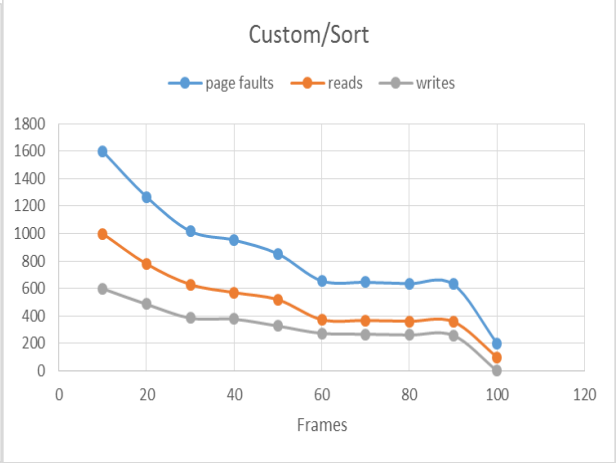


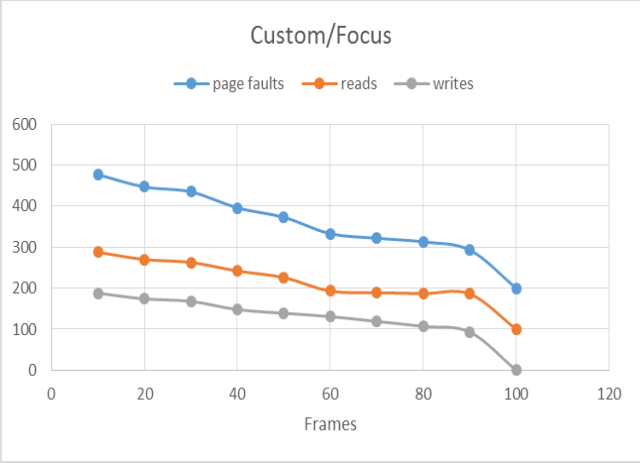


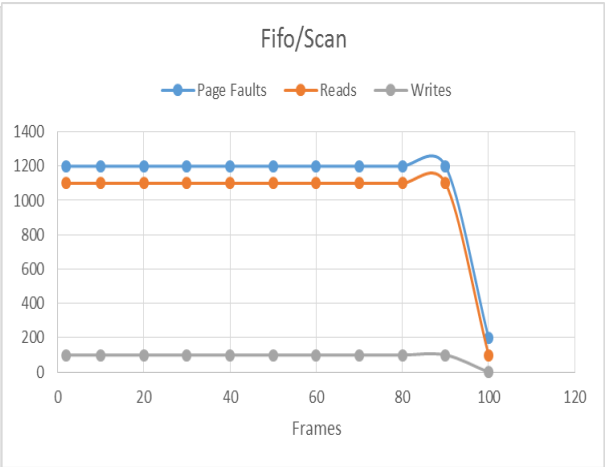


More individual graphs (per program per algorithm):



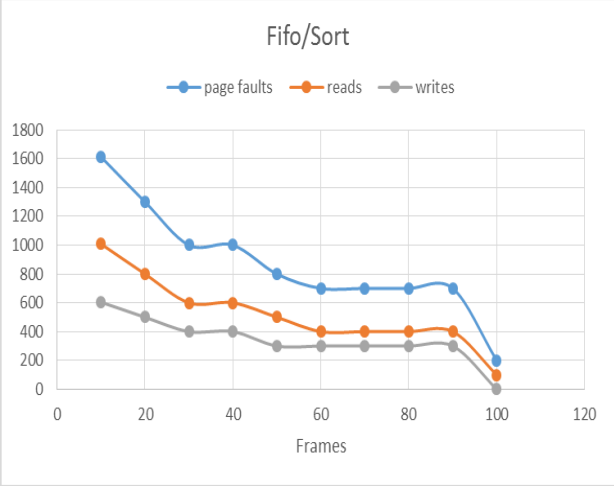


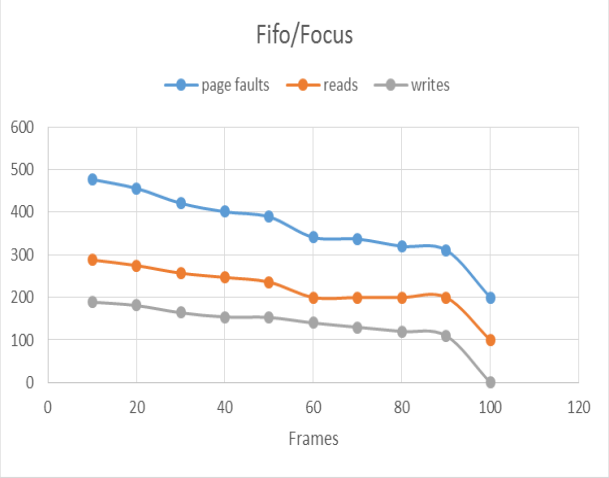


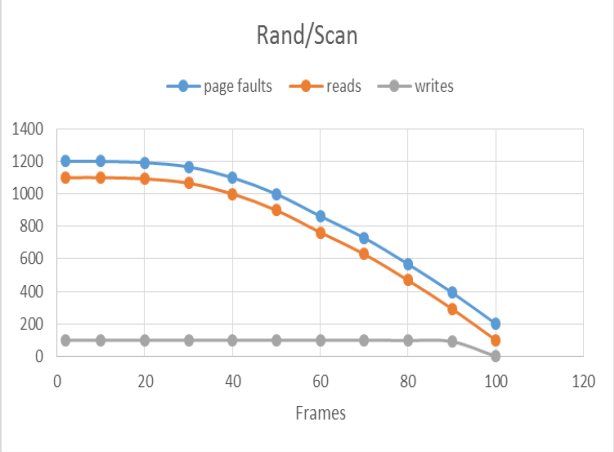


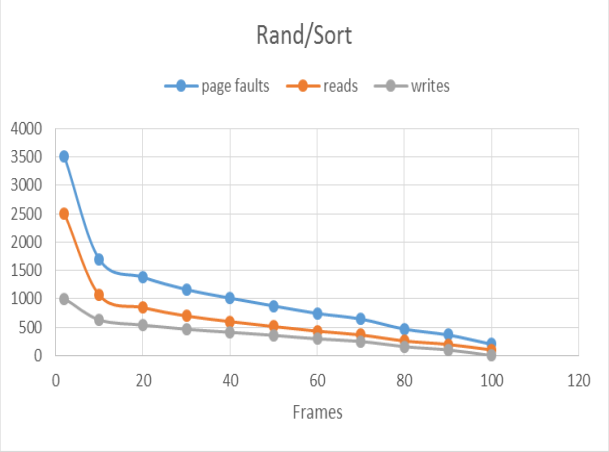
CSCI 5103 – Project 2

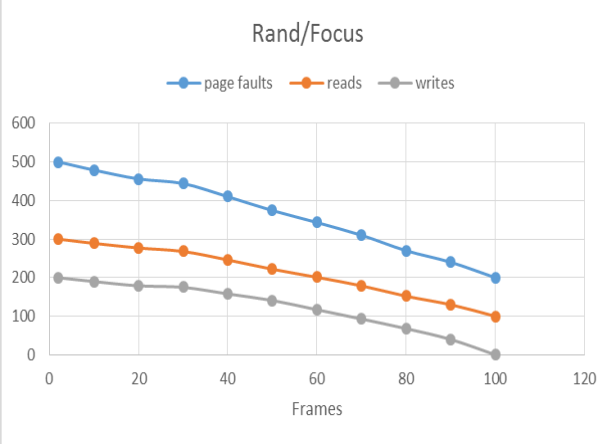
Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu











CSCI 5103 – Project 2

Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

Testcases Covered:

1. See if input is not properly given.
   a. If algorithm is not specified at all
   ```
   kandu009@suman:~/os/2project$ ./virtmem 100 10 scan
   usage: ./virtmem <num_pages> <num_frames> <rand|fifo|custom> <sort|scan|focus>
   ```
   b. If algorithm specified is not supported
   ```
   kandu009@suman:~/os/2project$ ./virtmem 100 10 rand1 scan
   usage: ./virtmem <num_pages> <num_frames> <rand|fifo|custom> <sort|scan|focus>
   ```
   c. If program is not specified at all
   ```
   kandu009@suman:~/os/2project$ ./virtmem 100 10 rand
   usage: ./virtmem <num_pages> <num_frames> <rand|fifo|custom> <sort|scan|focus>
   ```
   d. If program specified is not supported
   ```
   kandu009@suman:~/os/2project$ ./virtmem 100 10 rand scan1
   usage: ./virtmem <num_pages> <num_frames> <rand|fifo|custom> <sort|scan|focus>
   ```
   e. If number of pages < number of frames
   ```
   kandu009@suman:~/os/2project$ ./virtmem 1 10 rand scan
   number of frames must be <= number of pages.
   ```
   f. If number of frames are < 1
   g. If number of pages are < 1
   ```
   kandu009@suman:~/os/2project$ ./virtmem 100 0 rand scan
   number of frames must be at least 1.
   kandu009@suman:~/os/2project$ ./virtmem 0 12 rand scan
   number of pages must be at least 1.
   ```
2. Running with all ranges of pages (Between 1 to 100) with all ranges of algorithms (rand|fifo|custom) with all ranges of programs (sort|scan|focus). It runs as expected.

```
kandu009@suman:~/os/2project$ ./virtmem 100 10 rand sort
sort result is -311357
100, 10, 1716, 1082, 634, 1072
kandu009@suman:~/os/2project$ ./virtmem 100 10 rand scan
scan result is 522240000
100, 10, 1200, 1100, 100, 1090
kandu009@suman:~/os/2project$ ./virtmem 100 10 rand focus
focus result is -2533
100, 10, 478, 288, 190, 278
kandu009@suman:~/os/2project$ ./virtmem 100 10 fifo sort
sort result is -311357
100, 10, 1612, 1008, 604, 998
kandu009@suman:~/os/2project$ ./virtmem 100 10 fifo scan
scan result is 522240000
100, 10, 1200, 1100, 100, 1090
kandu009@suman:~/os/2project$ ./virtmem 100 10 fifo focus
focus result is -2533
100, 10, 477, 288, 189, 278
kandu009@suman:~/os/2project$ ./virtmem 100 10 custom sort
sort result is -311357
100, 10, 1598, 999, 598, 989
kandu009@suman:~/os/2project$ ./virtmem 100 10 custom scan
scan result is 522240000
100, 10, 1190, 1090, 99, 1080
kandu009@suman:~/os/2project$ ./virtmem 100 10 custom focus
focus result is -2533
100, 10, 477, 288, 188, 278
```

CSCI 5103 – Project 2

Ravali Kandur (5084769) kandu009@umn.edu
Charandeep Parisineti(5103173)
paris102@umn.edu

```
kandu009@suman:~/os/2project$ ./virtmem 100 10 custom focus
focus result is -2533
100, 10, 477, 288, 188, 278
kandu009@suman:~/os/2project$ ./virtmem 100 20 custom focus
focus result is -2533
100, 20, 447, 270, 174, 250
kandu009@suman:~/os/2project$ ./virtmem 100 50 custom focus
focus result is -2533
100, 50, 373, 226, 139, 176
^[[Akandu009@suman:~/os/2project$ ./virtmem 100 90 custom focus
focus result is -2533
100, 90, 293, 186, 93, 96
kandu009@suman:~/os/2project$ ./virtmem 100 100 custom focus
focus result is -2533
100, 100, 200, 100, 0, 0
```

3. When number of pages > 100. It runs as expected.

```
kandu009@suman:~/os/2project$ ./virtmem 1000 100 custom focus
focus result is -3344
1000, 100, 3153, 2065, 1072, 1965
```

4. Running on different CS machines. It runs as expected.
    a. suman.cs.umn.edu
    b. csel-kh1262-08.cselabs.umn.edu
    c. local machine

Assumptions:

1. Input command line will be of the exact format shown below

```
printf("usage: ./virtmem <num_pages> <num_frames> <rand|fifo|custom> <sort|scan|focus>
```

2. Number of frames is always <= number of pages
3. Output printed in the following format is acceptable
   <pages> <frames> <pagefaults> <disk_reads> <disk_writes> <disk_reads>
4. Adding additional file with detailed results is not an issue.