Georgios Karagiannis
Roman Sodermans
Michael Tang
Calvin Yang
3/1/2017

CMPS111-02: Assignment 3 Writeup

Introduction:

Upon completing the new (slim chance) paging algorithm, we performed 10 sets of stress tests on both the old and new paging algorithms. To stress the virtual machine, we used the stress package, using the command:

```
date; --vm 48 --vm-bytes 128M --timeout 30s
```

This stresses the virtual machine by having 48 workers, each occupying 128MB of virtual memory. This leads a total needed memory of 6.14 GB. Since our virtual machine is set to run on 4 GB of virtual memory, this stress test will guarantee that pages will get moved from the queues, as well as getting swapped and flushed. This means that we will obtain meaningful data for the counters tracking the number of pages moved from inactive to free/cache and the number of pages queued for flush.

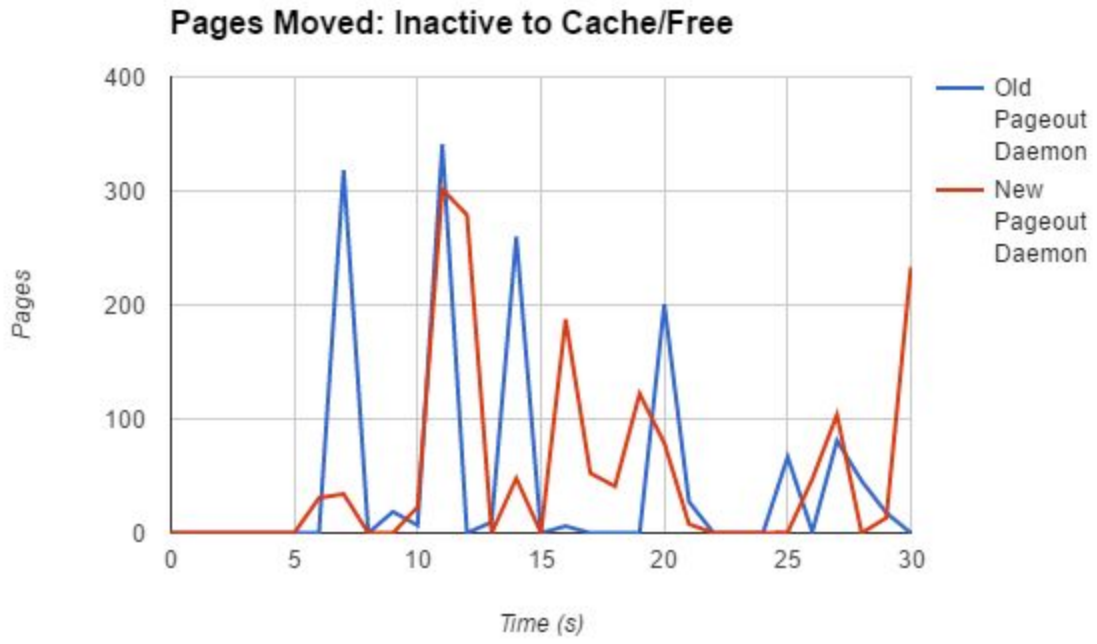
Methodology:

To perform the stress tests, we saved the old paging algorithm code in a child branch from our asgn3 branch. This branch only contains the log. From this, we use the timestamp obtained from 'date;' and pulled the data within stressed time frame from our log file (located in /var/log/debug.log).

We used a python script (included in our asgn 3 directory) to pull the respective data points by parsing through our logged data. We put these data in an excel file, and graphed the average data between 10 trials of stress vs time. Sometimes, in the log files, there were two sets of data for one second of time, therefore, we added the two data together for movement between pageout queues. For the number of total pages in each queue, however, we used the latter logged data. For skipped data entries (seconds ran with no prints to the log), we put in null values.

Note that all of these stress tests were run on the same computer within a set time to gain as accurate data as possible.

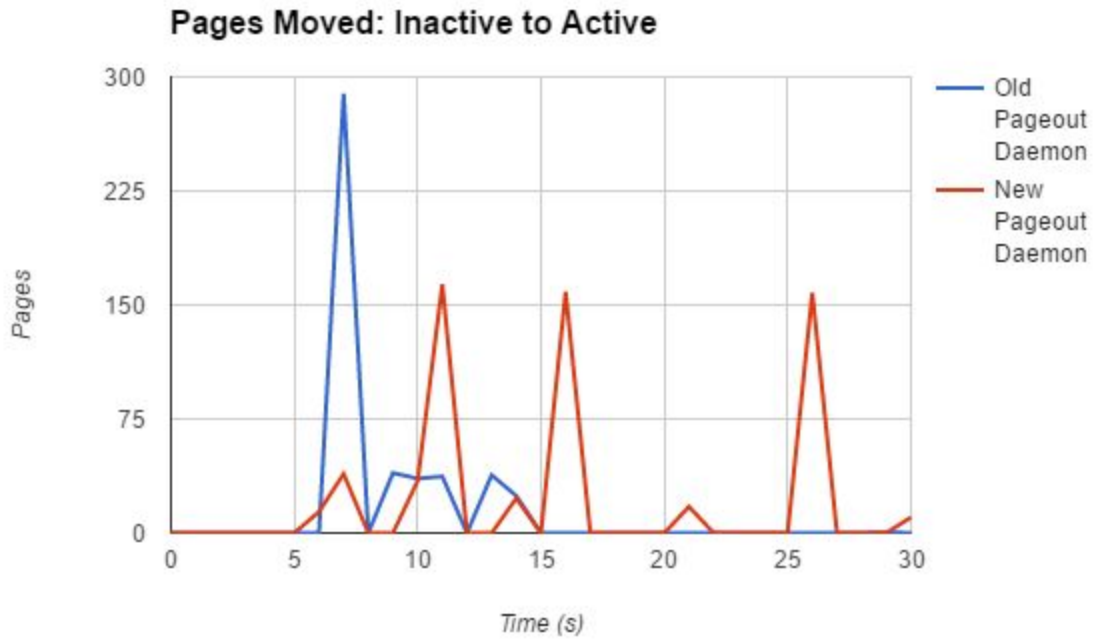
Number of Pages Moved from Inactive to Cache/Free



Graph 1: Pages Moved from Inactive to Cache/Free Queues

Using the old pageout daemon, there were 1401 pages that got moved from the inactive to the cache/free queue. The new pageout daemon moved, on average, 1604 pages between the two queues. We can see that there was an increase of about 200 pages moved in the new (slim chance) paging algorithm.

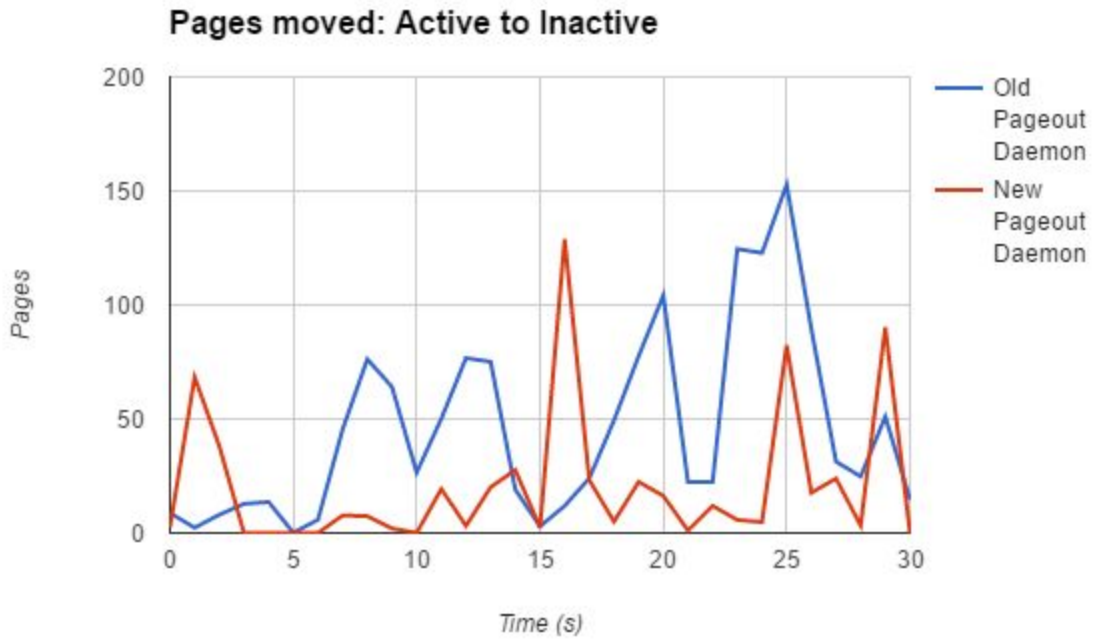
Number of Pages Moved from Inactive to Active Queue



Graph 2: Pages Moved from Inactive to Active Queues

The sum of pages moved from the inactive to the active queue, using the old pageout daemon, was 465, and using the new pageout daemon, there was 618 pages that were moved. It can be seen that using the slim chance algorithm, that there is an overall increase in page movement between the old and the new pageout daemon algorithms.

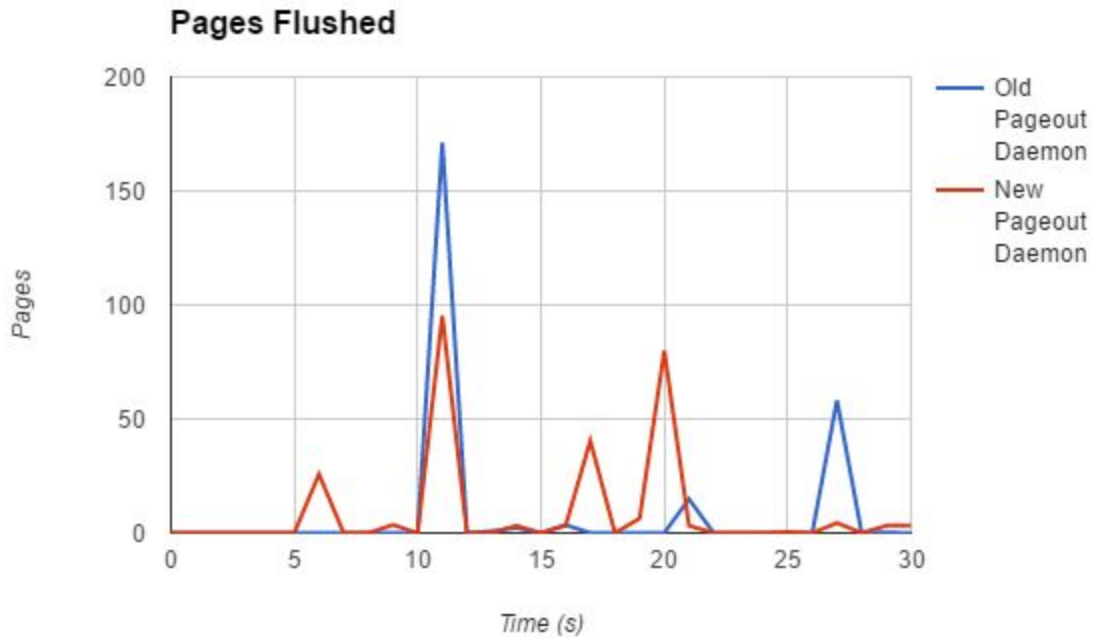
Number of Pages Moved from Active to Inactive Queue



Graph 3: Pages Moved from the Active to Inactive Queues

From our data, a total of 1411 pages were moved from active to inactive with the old pageout daemon, and a total of 635 pages were moved with the new pageout daemon. As can be observed, there was an overall decrease of more than half in pages moved from the active to inactive queues. We start with a large number of pages in the inactive queue (graph 5), meaning that we do not need to move pages from active to inactive as there is no lack of virtual memory yet. Only when virtual memory is running out does the page daemon start scanning the active queue to move pages into the inactive queue and try to free them in order to use the once again.

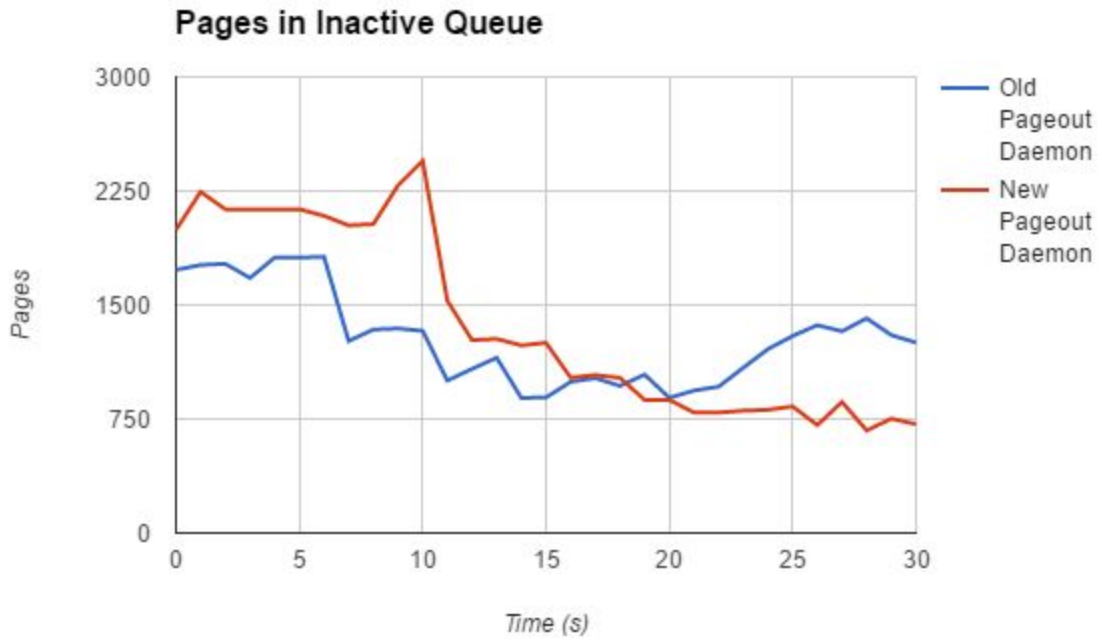
Number of Pages Flushed



Graph 4: Pages Flushed

There were 251 pages flushed with the old paging algorithm and 272 pages flushed with the new paging algorithm. The numbers are similar, with a slight increase of pages being flushed in the slim chance algorithm. The number of flushed pages is small compared to the total number of pages in the inactive queue at any current time. This lack of page movement is because the stress program does not dirty pages as it does not modify any pages.

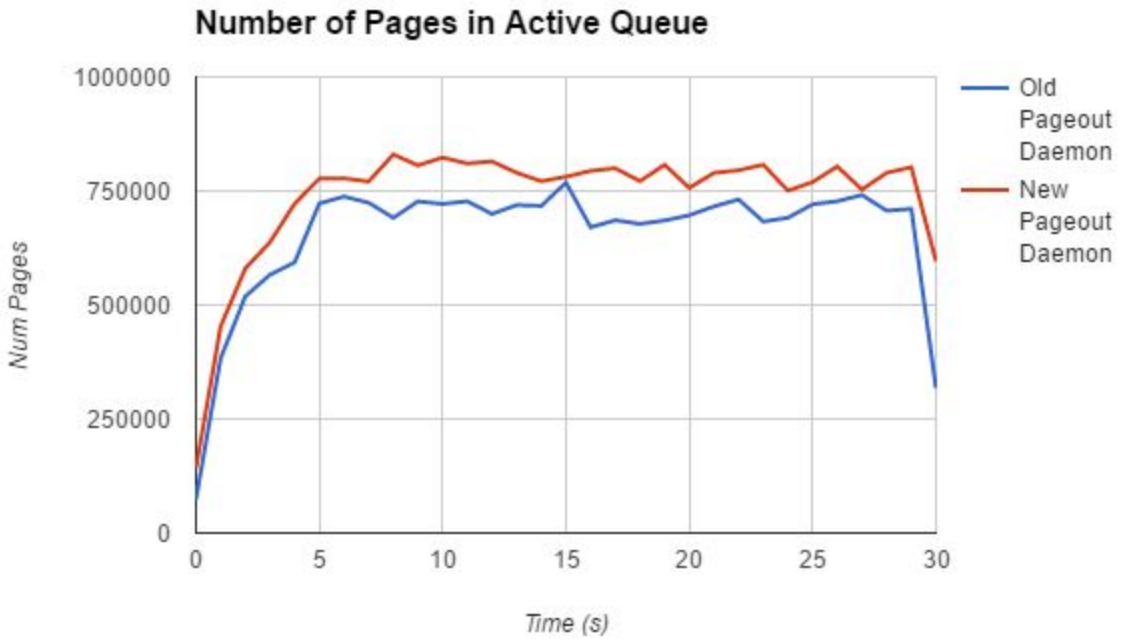
Number of Pages in Inactive Queue



Graph 5: Pages in Inactive Queue

Initially, there is a significantly higher number of pages in the inactive queue (between the new and old pageout daemons), but it progressively becomes having fewer pages in the inactive queue for the new pageout daemon. The drop of number of pages in the inactive queue is justified by the increase of pages being moved from the inactive queue to the cache/free (graph 1) and to the active queue (graph 2) because the virtual machine is in need of more memory due to it running out of virtual memory from the stress program.

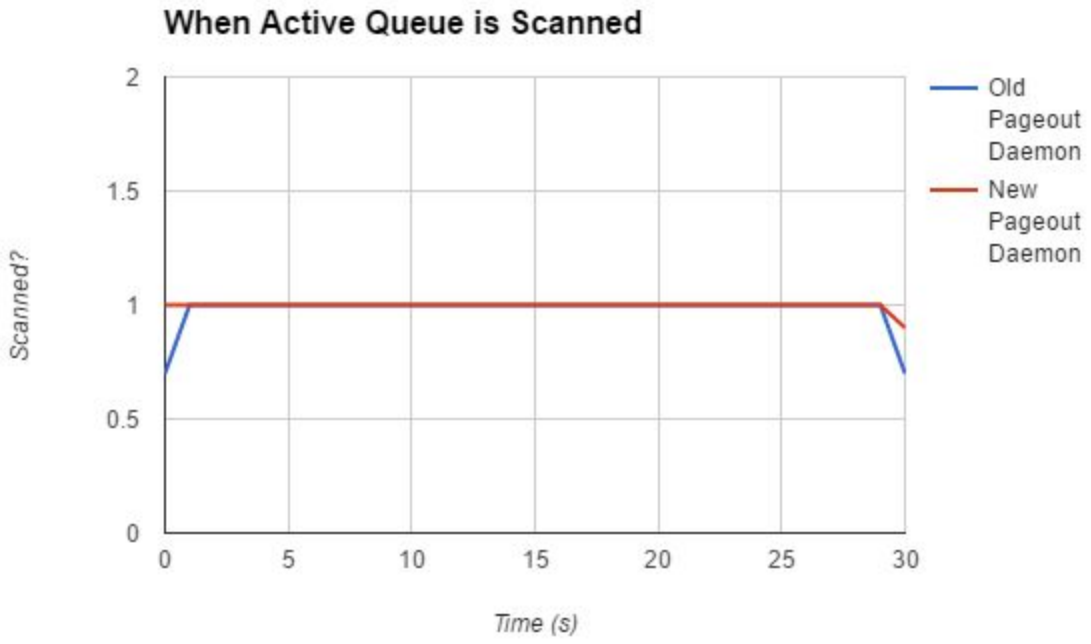
Number of Pages in Active Queue



Graph 6: Pages in Active Queue

From the data, the number of pages in the active queue have no significant difference between the two different pageout daemons. The rate of change within the first 5 seconds of the stress test and the last 5 seconds is similar. This is due to us running the same amount of stress on the Virtual Memory, between the kernels.

When Active Queue is Scanned



Graph 7: Frequency of Active Queue being Scanned

The data shows that in both instances of the kernel, the active queue is almost always scanned. From this we can simply conclude that the active queue is scanned every runthrough of the pageout_scan function. This is expected because there will always be pages within the active queue.

When Inactive Queue is Scanned

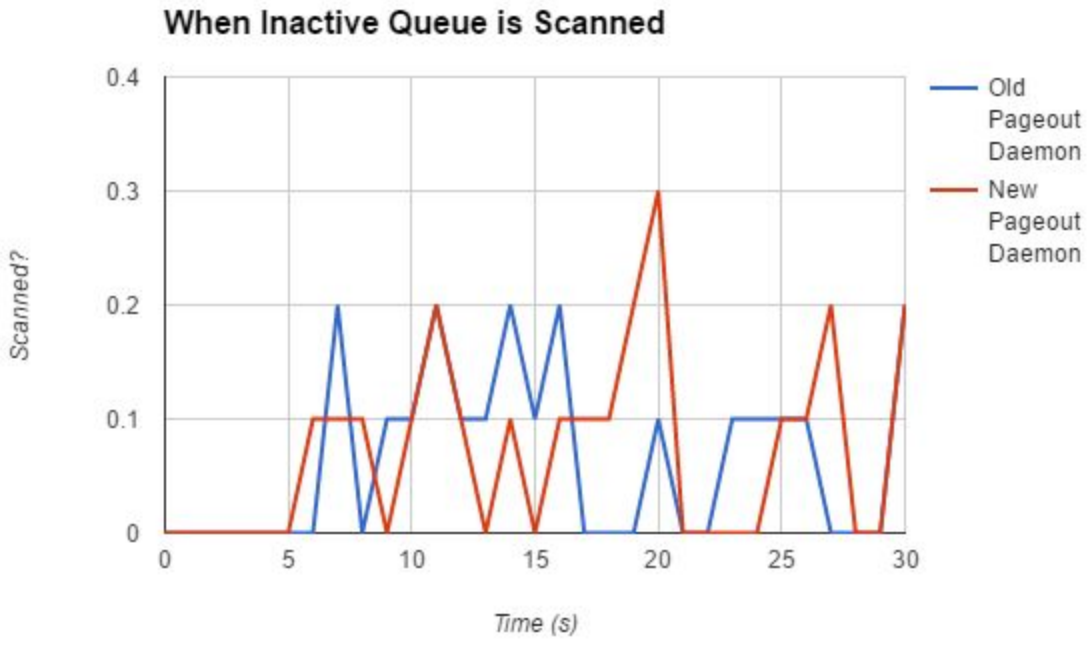


Figure 8: Frequency of Inactive Queue being Scanned

We can see that the inactive queue is scanned at similar rates between the two kernel pageout daemons. This is because the rate of the number of pages leaving the inactive queue should be the same between the two different kernels as they do not depend on activity count. For the new pageout daemon, he two points where there is an increase of the inactive queue being scanned (at t = 12 and at t = 20) is expected since a significant number of pages are flushed (graph 4) at around these times. Furthermore, a large number of pages are moved from inactive to CACHE/FREE (graph 1) as well. Finally, there is also many pages being moved from the inactive to the active queues (graph 2) during this time. Thus, the inactive queue must be scanned more during these times.

Analysis:

The number of pages in the Inactive Queue is initially larger in our new daemon because we are dividing activity count by 2, and adding to the head of its queue instead of the tail. This means that the activity count decays at a higher rate. Also, adding to head means the pages must be considered immediately to move between the paging queues. Dividing the activity count by 2 makes pages go below the activity count threshold much faster which means much more pages are being placed into the inactive queue/free from the active queue.

Observe in Graph 3 (Active to Inactive) that, for the new daemon, at 15 seconds there is a large spike in page movement. This corresponds with the movement of pages (of nearly the same magnitude) from Inactive to Active at 15 seconds, which would explain the lack of influx at 15 seconds in the Size of Active Queue graph for the new daemon. Although many pages are being moved into the inactive queue (from the active queue), the number of pages in the inactive queue is still decreasing (see graph 5 from t = 10 to t = 25). This is due to the fact that pages from the inactive queue are being moved to the cache/free and back to the active queue (observed in graphs 1 and 2 at about t = 10, 15, 25 seconds).