Assignment 3 Conclusions

March 3, 2017

Nikola Panayotov (npanayot@ucsc.edu)
Marko Jerkovic (mjerkovi@ucsc.edu)
Edwin Ramirez (edalrami@ucsc.edu)
Jeremiah Liou (jliou@ucsc.edu)

1 Data and Analysis

In order to analyze the data from each algorithm without stress (during booting) and under stress, we exported the log files and converted them to csv files that we handled with MATLAB scripts that we wrote. This is how we produced our figures that will be our main focus for our analysis. The scripts used to handle the data collected and produce the figures have been included to show that they are in fact ours.

1.1 Boot Up

1.1.1 Number of pages in Inactive Queue vs Active Queue

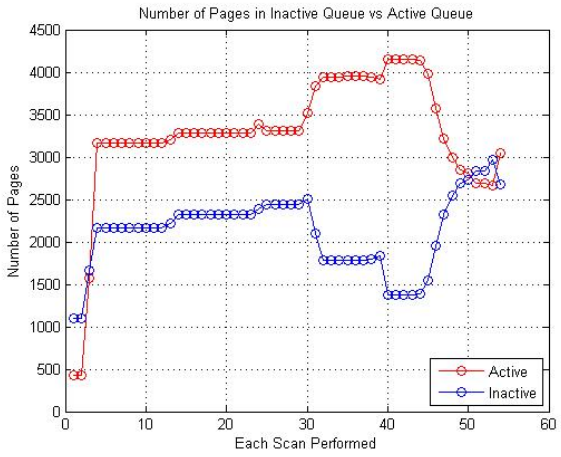

Figure 1: Default Algorithm

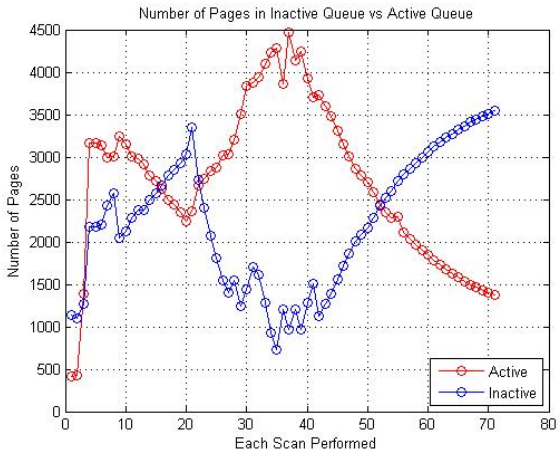

Figure 2: FC Algorithm

In the number of pages in inactive queue vs active queue graph, the fat chance algorithm had much more fluctuating values than the default pager. In the default pager on startup, first the page will be filled up with information for the boot to run. And once it is filled up, there certain pages will need to be moved from the respective places. In the default pager, the graph shows that pages in the inactive queue and active queue are relatively stable in moving pages around. However in the fat chance pager, it shows that pages were moved much more throughout the whole process. By decreasing the activity count much faster in fat chance paging (activity count is halved instead of decremented by subtracting), each page reaches the inactive threshold much quicker, and is subsequently evicted from the active queue. Additionally, the instability observed in the fat chance paging algorithm can also be due to frequently-used pages being evicted as a result of remaining at the front of the active queue with a lower activity count. At the end of the boot up, the number of active pages decreases, while the inactive page count increases. This is most likely due to the system finishing its start-up phase and idly waiting for user input or user processes to execute.

1.1.2 Inactive pages moved to Cache vs Free

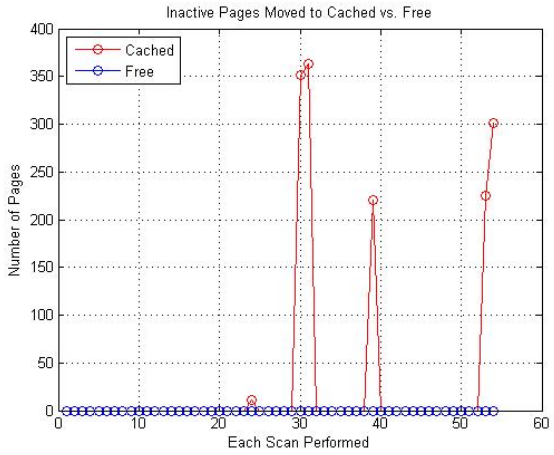

Figure 3: Default Algorithm

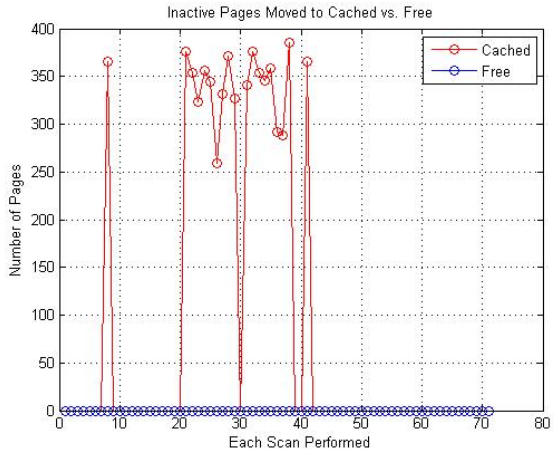

Figure 4: FC Algorithm

By changing how pages enter the inactive queue, the pages moved to cache was also influenced. Since the scan starts from the beginning of the queue, and pages that move to the inactive list from the active queue are moved to the front, it is easier to find a page to be moved to cache. That is because when pages are moved from the inactive list, it is being checked if it is referenced and will more likely be clean. It is clearly observed in the graphs that the fat chance algorithm moves more pages from inactive into cached, essentially freeing them. It is also observed that no pages were ever moved directly from inactive into free. This is justified since in the Implementation of FreeBSD book, it states that the only time this would occur is if a pages contents are invalid, most likely due to an I/O error.

1.1.3 Pages queued for flush

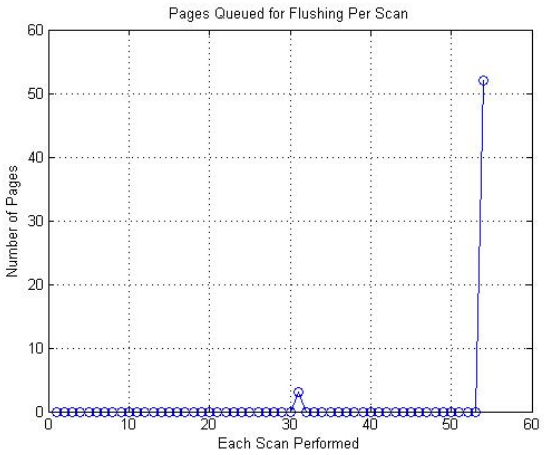

Figure 5: Default Algorithm

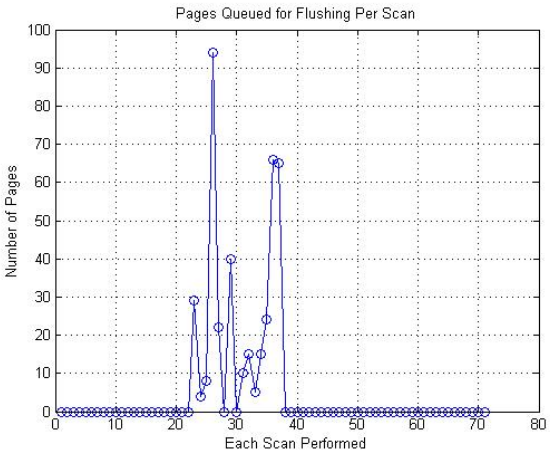

Figure 6: FC Algorithm

During the boot up of the default paging algorithm, almost no pages are queued for flushing until the very end. This is expected, as the pages used during boot up can remain in memory for the duration, and upon fully starting, the pages are queued for flushing in order to create space for pages used for user programs. In the fat chance paging algorithm however, pages are queued for flushing much earlier on. This could be due to more pages entering the inactive queue, and the pager attempting to balance out the queues, by keeping the number of pages in the inactive queue between the minimum and target values, as specified in the Implementation of FreeBSD book.

1.1.4 Active pages moved to Inactive per scan

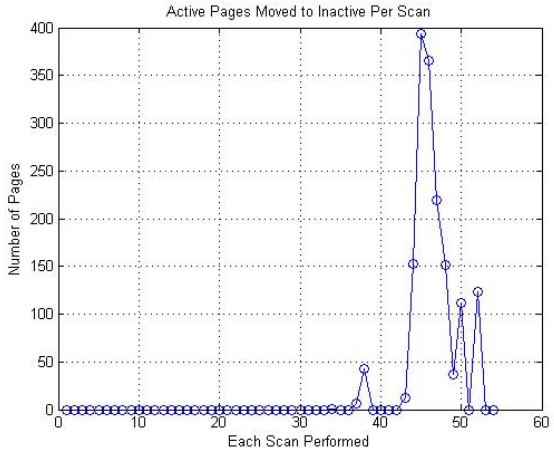

Figure 7: Default Algorithm

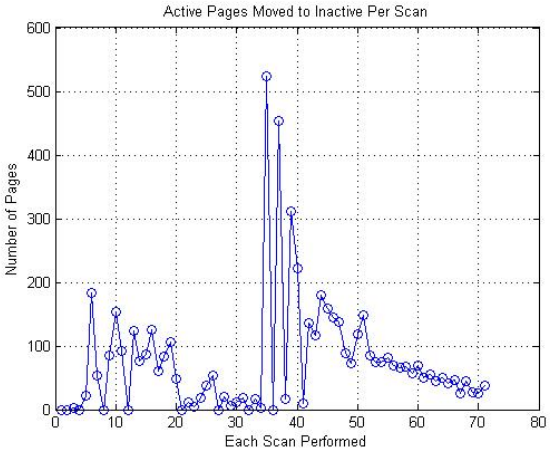

Figure 8: FC Algorithm

With the default algorithm, the number of pages moved from active to inactive is relatively low, until toward the end of boot up. However, in the fat chance algorithm, a lot more active pages were moved to inactive per scan. This is because dividing the activity count in half made pages move onto the inactive queue faster.

1.1.5 Number of Active pages scanned vs Inactive

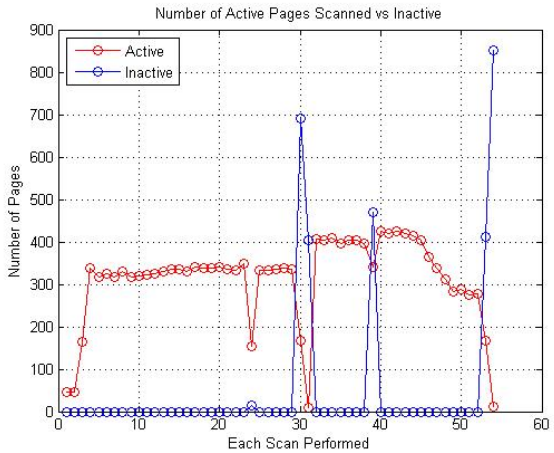

Figure 9: Default Algorithm

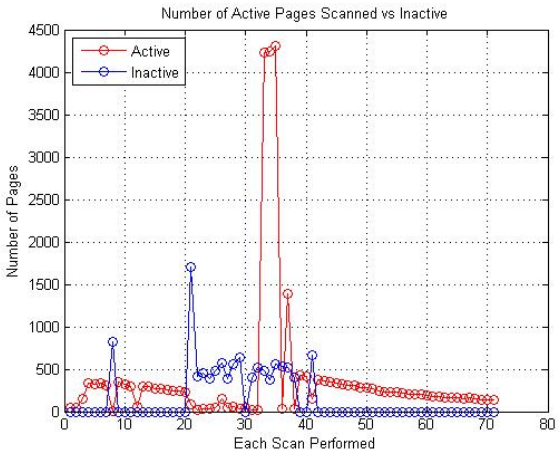

Figure 10: FC Algorithm

In the number of active pages scanned vs inactive, the graph shows that they look relatively similar. However, in the fat chance paging, there was a huge spike scan for the active pages. This could be caused by the fact that the activity count for the pages in the active

queue during boot up was relatively high, and the pager had to scan the entire queue in order to evict a number of pages. After the spike in the number of active pages scanned, the number of active pages scanned falls down to around the same amount as that of the default pager. A possibility of why this happens is that the activity counts of the active pages are now much lower than before, and pages earlier in the queue can be moved from active to inactive.

1.2 Stress Test

To calculate how well each algorithm performed under stress, each algorithm was run 20 times using `stress` to collect substantial data for analysis. Every run had its own respective number of scans collected, thus we had to calculate the average amount of scans per run for each algorithm, and the average number of pages/queues scanned and pages moved per scan. The following figures illustrate the average number of scans plotted against the respective data that was recorded (i.e. pages moved from queues, total pages scanned, active/inactive pages scanned, etc.).The stress command we ran was:

```
stress --vm 16 --vm-bytes 32M --timeout 30s
```

1.2.1 Number of pages in Inactive Queue vs Active Queue

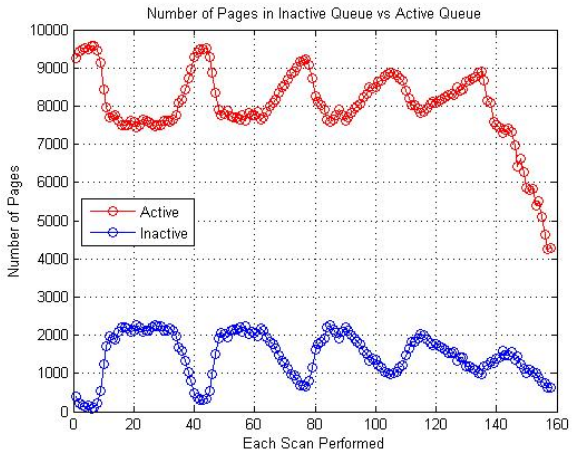

Figure 11: Default Algorithm

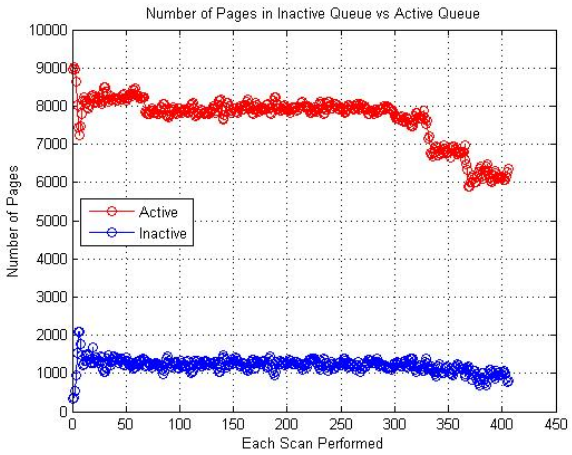

Figure 12: FC Algorithm

The number of pages in the inactive vs the active queue seems to fluctuate a lot more using the default paging algorithm when compared to the fat chance. This could be because the fat chance is much more responsive in moving pages from the active and inactive queue to maintain target thresholds. The responsiveness is a result of the activity count being divided by two rather than being decremented by subtraction.

1.2.2 Inactive pages moved to Cache vs Free

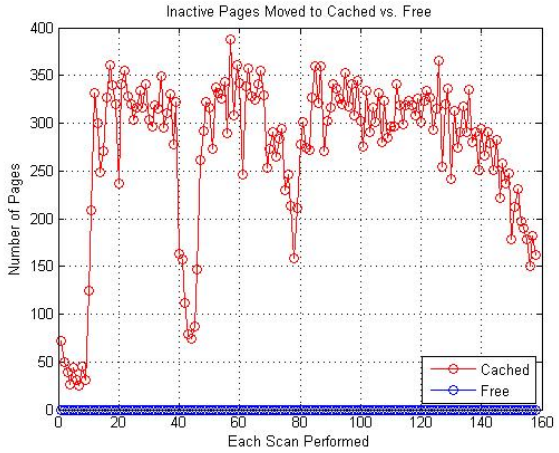

Figure 13: Default Algorithm

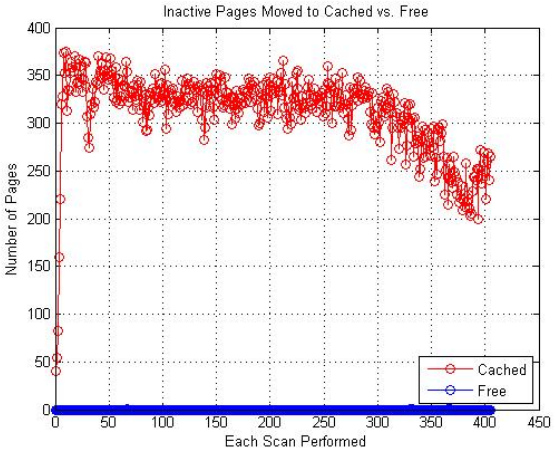

Figure 14: FC Algorithm

The fat chance algorithm cached pages much more often than the default algorithm because pages move much quicker from the active queue to the inactive queue. In order to maintain the target free + cached pages queue ratios, clean pages must be cached much more often than they do using the default algorithm. The default algorithm isn't caching pages at such a high rate because less active pages aren't deactivated as quickly.

1.2.3 Pages queued for flush

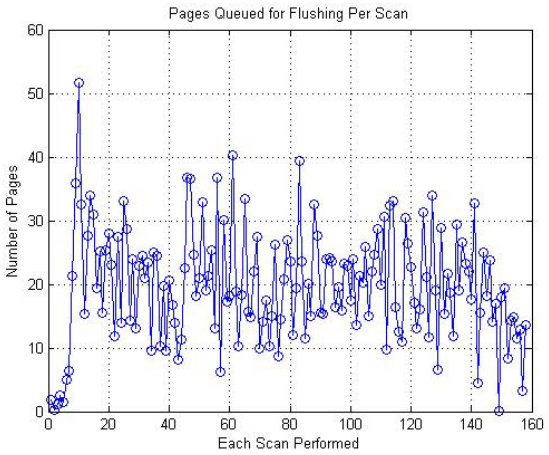

Figure 15: Default Algorithm

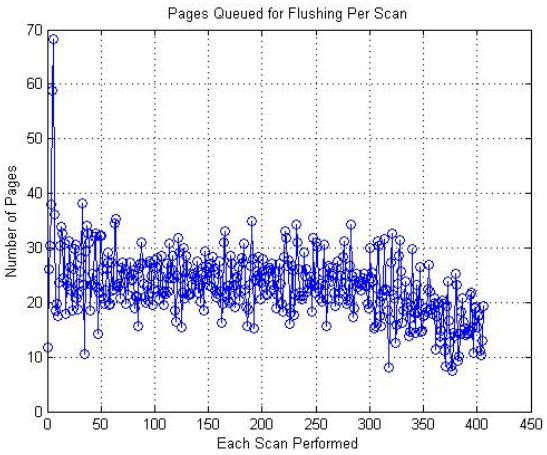

Figure 16: FC Algorithm

The number of pages queued for flushing using the fat chance algorithm is on average much higher than the number of pages queued for flushing using the default algorithm. This is again the result of more pages being deactivated which causes the algorithm to work harder to maintain the target ratios of free + cached pages. The FC algorithm flushes/cleans more pages so on the next scan of the inactive queue there are more clean pages which can be cached. But the number of pages queued for flushing is somewhat limited so the IO doesn't get flooded, so there aren't any spikes of more than 70 pages in the queue for flushing.

1.2.4 Active pages moved to Inactive per scan

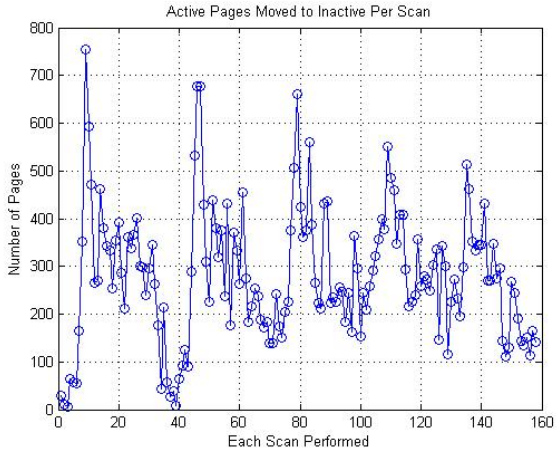

Figure 17: Default Algorithm

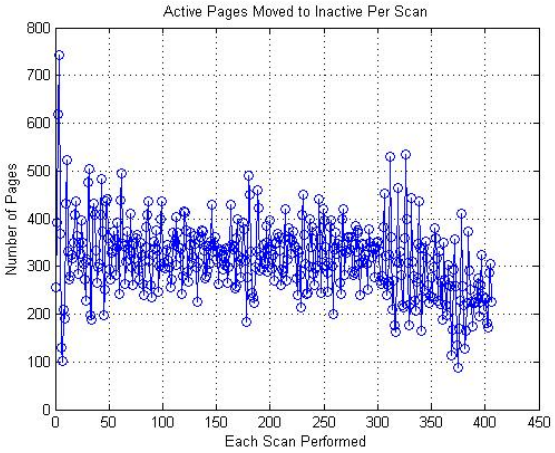

Figure 18: FC Algorithm

The number of pages moved to the inactive queue is much more peaky with the default algorithm. While with the fat chance algorithm, consistently 200 to 500 pages are moved from the active to inactive queue. Once again, this is because a slightly less active page will be deactivated much sooner because the activity count is being halved rather than being decremented by subtraction. This causes the fat chance algorithm to, on average, move more pages from the active to the inactive queue.

1.2.5 Number of Active pages scanned vs Inactive

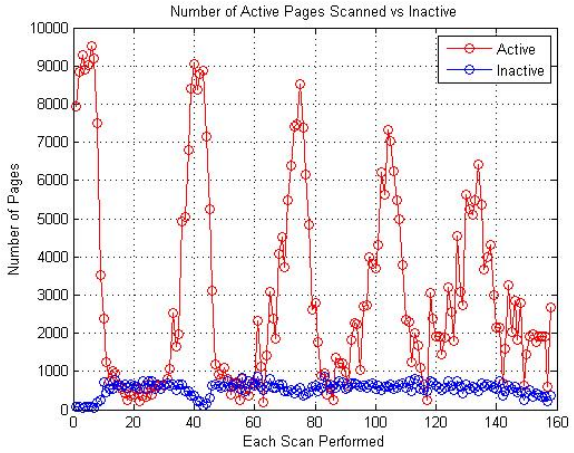

Figure 19: Default Algorithm

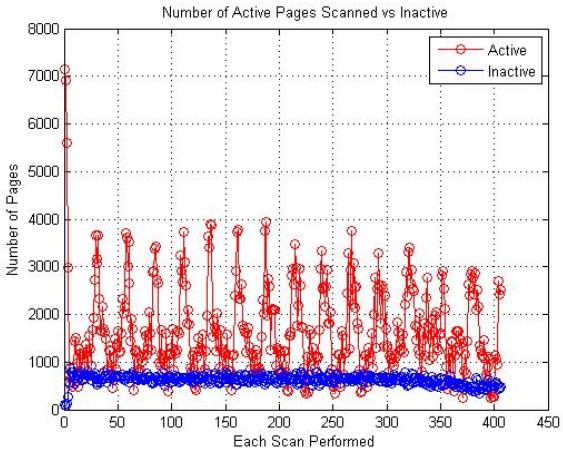

Figure 20: FC Algorithm

The default algorithm scans active pages more often because it sometimes has a lot more pages in the active queue than the fat chance algorithm would have because the fat chance algorithm quickly deactivates pages. So the default algorithm must scan the active queue more often and thus scan more active pages, in order to maintain the target ratios for the queue lengths. The fat chance algorithm consistently scans around 700 to 800 inactive pages because there are constantly pages being deactivated which need to be flushed, while the number of inactive pages scanned with the default algorithm fluctuates more because less pages are being deactivated.

Stress Test Conclusion

Overall the fat chance algorithm did over twice the number of scans in the twenty 30 second runs of the stress test. This not only results in more computations due to running the scanner more often, it also increases the number of page faults due to more slightly less active pages being deactivated and consequently cached/freed sooner which can drastically decrease the speed of a program due to the slow speed of disk reads.