Group 1: Lab 2 Report
Instructor: Dr. Saad Biaz
Institute: Auburn University
Paul Chong (phc0004@auburn.edu)
Justin Sawyer (jts0047@auburn.edu)
Date Submitted: June 20, 2018

**** Note: Specific instructions to include FCFS and SRTF graphs/charts were not present again in the Lab 2 prompt so we left them out as we did in the previous Lab.****

Policy Evaluation

To gather our data for each policy, we compiled/ran our code in the TUX environment and let it generate 250 processes before we compared our numbers to the provided 'Behind the Scenes' numbers. Our processes management numbers, like our previous lab, were very accurate again (discrepancies of .0001% to 2.0%) when compared to the 'Behind the Scenes' numbers. However, since we are closely examining the new 'Average Wait Time in the Job Queue', or 'AWTJQ' metric in this lab, we chose to exempt the 'Behind the Scenes' numbers for this analysis because 'AWTJQ' was not displayed alongside the other metrics in the TUX machines. To gauge our accuracy, we instead relied on our analysis of each policy below and compared our results to our knowledge gained in class about what these numbers should look like.

Analysis of Normal Processes Management Policy (ASIS / Infinite Memory)

We will be observing Appendix A's Chart 1 and the entirety of Appendix B in this portion. By running the c file as given, we could reproduce an illusion or simulation where processes can be taken care of in an atmosphere where there is infinite memory. This was the baseline for our Lab 1, and metrics were taken (Appendix A, Chart 1) to verify the same results and to ensure that we didn't proceed with errors in the base code. By observing our data, we could almost immediately find the striking differences and advantages of having infinite memory.

The first and most obvious is the AWTJQ metric (Appendix B, Chart 6), where it was on average 0.0009, whereas other policies averaged near 0.1 to 0.5. This proves to be a strikingly large gap between a policy with infinite memory and the rest because nothing can come close to it. The only other notable feature of this policy is the average wait time (Appendix B, Chart 5) where it seems to have a much higher wait time than the other policies. This may be explained by how with bigger quanta in an infinite atmosphere, the system will instead begin to slow by taking bigger chunks. Otherwise, this policy aligned with other policies closely in terms of metrics.

Analysis of Paging Policy

We expected paging to have only one noticeable feature and that was to have the worst AWTJQ time. Fortunately, as seen in Appendix B, Chart 6, paging began to deviate from other policies and started a noticeable trend of worsening. The reason why we expected this is because the concept of paging revolves around constantly pulling from the secondary storage of the same-size blocks or pages. This is like comparing a non-preemptive method like FCFS or SJF to Multi-Level Feedback Queue in which the lesser effective methods simply grab whatever they run into and complete the process directly. Paging operates similarly and only grabs the same size blocks from secondary storage to put into the main memory, meaning that it will be far worse than methods like OMAP, Best-Fit, and Worst-Fit since they at least each have checks to see where to put specific blocks of memory when the system is constrained. Other than this, paging, as seen in Appendix B, is very similar to other policies in the remaining metrics.

Analysis of Optimal Memory Allocation Policy (OMAP)

Although by observing the charts in Appendix A and Appendix B it may seem trivial to see how OMAP aligns closely to other policies in their metrics, it is very important to note that although the numbers are comparatively close, OMAP is always at least a little bit ahead of every other policy besides infinite memory. Casting infinite memory aside in this comparison (it is not a fair juxtaposition as infinite atmosphere are infeasible), OMAP performs better than other policies because of the sole fact that it does not care where it allocates the memory. Naturally, by not having to do constant checks of page sizes, efficiency, and specific nodes, OMAP may not perform astoundingly better than paging, best-fit, and worst-fit but will always have a winning edge since the other three methods care about size and placement when compared to other memory blocks or nodes.

Analysis of Best-Fit Policy

For Best-Fit Policy We will be observing Appendix A Chart 4, which is the statistics we have gathered from running our C program with the Best-Fit memory policy. The Best-Fit memory policy will take a process and place it into the smallest open memory spot that is big enough for the process to fit in. For this assignment, it seems that Best-fit policy never did exceedingly well, relative to the other memory policies implemented, for any of the recorded times of the policies. Literally, Best-Fit did about average in every timing. In average wait time, average CPU burst time, and average turnaround time (with very low quanta) it was right in the middle of all the other policies with its time. All in all, The Best-Fit policy is not the best policy, but not the worst as well.

Analysis of Worst-Fit Policy

For Worst-Fit Policy, we will be observing Appendix A Chart 5, which is the statistics we have gathered from running our C program with the Worst-Fit memory policy. The Worst-Fit memory policy basically does the exact opposite of Best-Fit. With Worst-Fit it places a new process in the biggest memory spot that is big enough for the process to fit. Worst-Fit did about average for all the timings, except with average CPU Burst Time. Worst-Fit did worst when quantum = 250, but when the quantum rises high, like when quantum = 500, Worst-Fit actually does the best. Following this trend is shown in Appendix B Chart 3, Worst-Fit policy should have the best CPU burst time when the quantum rises high. Overall, Worst-Fit Policy will do about the same as the other memory policies in this assignment, but with CPU burst time, it seems to do well with very high quantum.

Conclusion

With all the previous observations in mind, we can confidently conclude by confirming that our results produced in this lab have aligned with what we have learned and been tested in during class. Infinite memory showed the tremendous advantages of having almost non-existent AWTJQ when compared to other policies and OMAP also showed its superior edge over other policies (excluding infinite memory) because of how it does not care to perform specific memory allocations. Simple paging showed its weakness of having to only handle the same page or block sizes, especially when the system is constrained and dealing with many complicated processes. Best-Fit and Worst-Fit showed that although they are more realistic in terms of how we may have to handle memory allocation in modern systems, this comes with the heavy and unfortunate price tag of sacrificing turnaround time, response time, throughput, and especially the average with time in the job queue. The policies did not do exceedingly well but they performed relatively good numbers considering the environment that they were being tested in. As we concluded in Lab 1, this specific assignment has reinforced our learning and the concepts that we have been taught and tested in during class. Have a great summer!

Appendix A

Chart 1

Normal Processes Management Numbers (Infinite Memory Simulation)

Policy	Policy #	TAT	RT	CBT	T	AWT	AWTJQ
FCFS	1	17.64442	3.047076	0.971351	0.987758	12.43534	0.000958
SRTF	2	4.173769	1.412055	0.971077	1.048127	2.097974	0.00101
RR (Q = 10ms)	3, 10	10.23431	0.099957	0.969449	1.013891	8.952775	0.000984
RR (Q = 20ms)	3, 20	10.19887	0.190994	0.971757	1.013915	8.802644	0.000981
RR (Q = 50ms)	3, 50	10.90213	0.462383	0.977336	1.013814	9.153792	0.000997
RR (Q = 250ms)	3, 250	14.64876	1.852088	0.980493	0.9872	10.95575	0.001028
RR (Q = 500ms)	3, 500	16.37551	2.616493	0.980581	0.98304	11.6654	0.001011

Chart 2

Paging Policy Numbers

Policy	Policy #	TAT	RT	CBT	T	AWT	AWTJQ
FCFS	1	17.15644	3.439813	0.971265	0.98767	11.56753	0.571889
SRTF	2	4.17356	1.412368	0.971038	1.048085	2.098345	0.001053
RR (Q = 10ms)	3, 10	10.26379	0.100318	0.969268	1.013832	8.981197	0.001062
RR (Q = 20ms)	3, 20	10.22819	0.192112	0.971609	1.01385	8.831831	0.001026
RR (Q = 50ms)	3, 50	10.93009	0.462311	0.977304	1.013781	9.17722	0.001016
RR (Q = 250ms)	3, 250	14.52419	1.975541	0.98024	0.987072	10.71222	0.161976
RR (Q = 500ms)	3, 500	16.16518	2.859196	0.980961	0.983298	11.19528	0.337795

Chart 3

OMAP Policy Numbers

Policy	Policy #	TAT	RT	CBT	T	AWT	AWTJQ
FCFS	1	17.19411	3.413348	0.97109	0.987492	11.60891	0.540231
SRTF	2	4.174195	1.412899	0.970905	1.047942	2.098596	0.001216
RR (Q = 10ms)	3, 10	10.60194	0.104472	0.966599	1.013796	9.309839	0.001214
RR (Q = 20ms)	3, 20	10.33066	0.1929	0.970398	1.01371	8.929999	0.00119
RR (Q = 50ms)	3, 50	10.945	0.463816	0.976738	1.013641	9.185956	0.001173
RR (Q = 250ms)	3, 250	14.52005	1.983741	0.980103	0.986935	10.72079	0.162178
RR (Q = 500ms)	3, 500	16.13964	2.827537	0.980874	0.983211	11.21922	0.299752

Chart 4

Best Fit Policy Numbers

Policy	Policy #	TAT	RT	CBT	T	AWT	AWTJQ
FCFS	1	17.19594	3.416202	0.971134	0.987536	11.61106	0.540072
SRTF	2	4.175159	1.412669	0.970769	1.047795	2.101575	0.001158
RR (Q = 10ms)	3, 10	10.49847	0.102769	0.968138	1.013827	9.209833	0.001146
RR (Q = 20ms)	3, 20	10.26941	0.193301	0.970975	1.013773	8.870048	0.00115
RR (Q = 50ms)	3, 50	10.92019	0.461707	0.976672	1.013349	9.167934	0.001137
RR (Q = 250ms)	3, 250	14.501	1.978346	0.98021	0.987043	10.69565	0.162122
RR (Q = 500ms)	3, 500	16.14011	2.826067	0.980976	0.983313	11.21326	0.299651

<u>Chart 5</u>

Worst Fit Policy Numbers

Policy	Policy #	TAT	RT	CBT	T	AWT	AWTJQ
FCFS	1	17.20267	3.41853	0.971031	0.987432	11.61596	0.540186
SRTF	2	4.171783	1.424427	0.970829	1.04786	2.103694	0.0012
RR (Q = 10ms)	3, 10	10.67592	0.104905	0.96571	1.013723	9.381994	0.001279
RR (Q = 20ms)	3, 20	10.4217	0.195185	0.969649	1.01376	9.016386	0.001209
RR (Q = 50ms)	3, 50	11.0227	0.468525	0.976457	1.013574	9.263185	0.001207
RR (Q = 250ms)	3, 250	14.53506	1.991033	0.980795	0.987802	10.72898	0.167215
RR (Q = 500ms)	3, 500	16.15672	2.841877	0.979554	0.982484	11.22846	0.299688

Appendix B

Chart 1

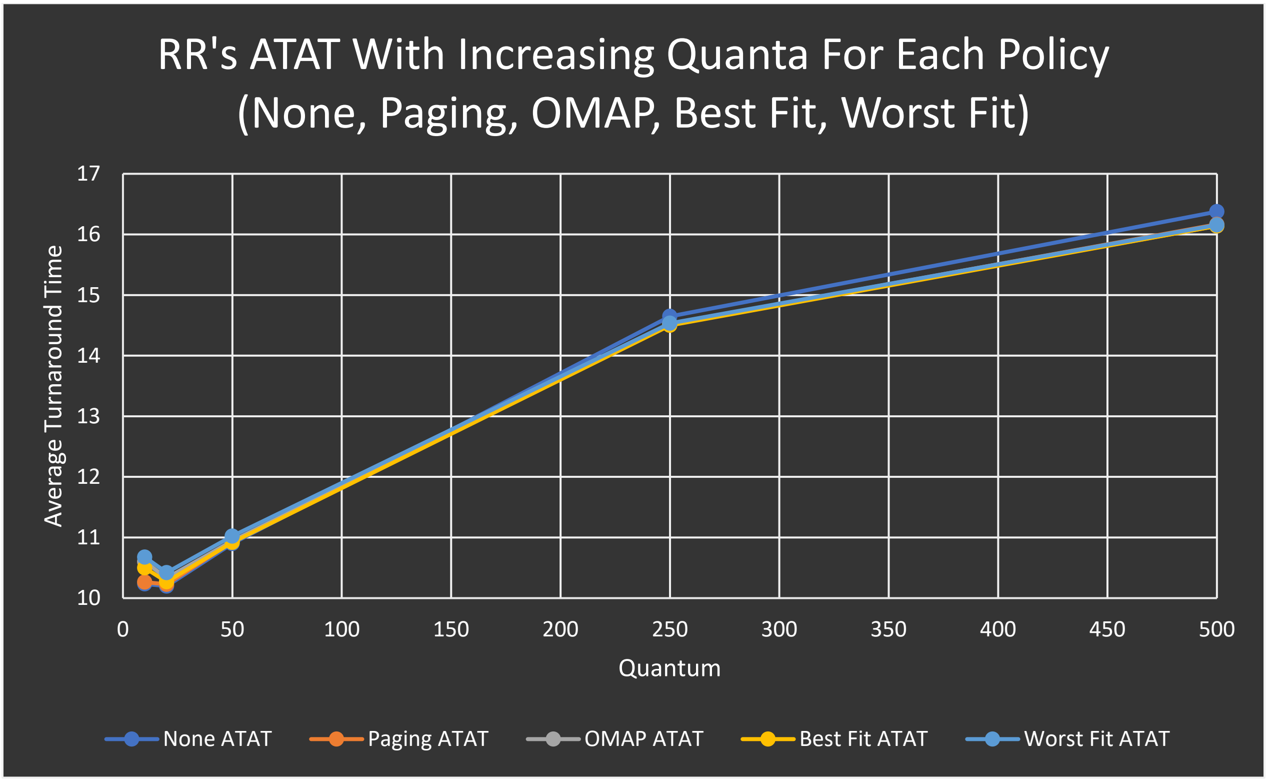


Chart 2

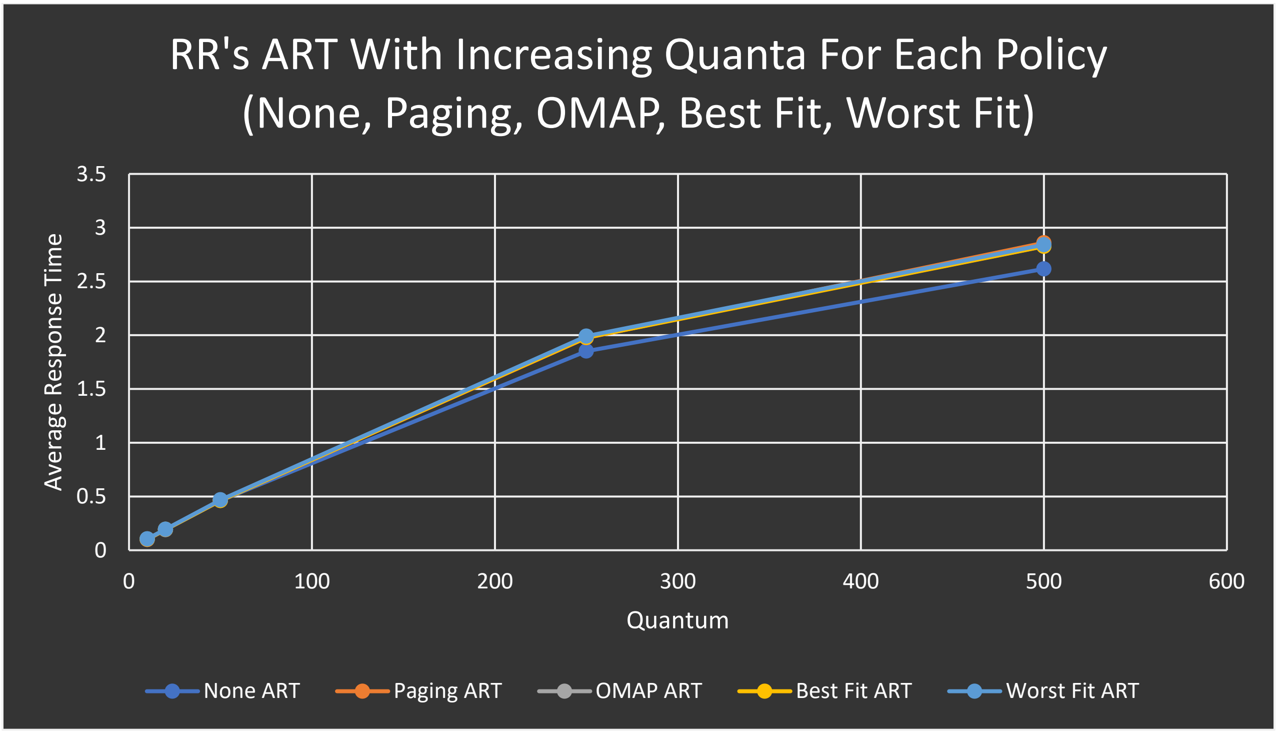


Chart 3

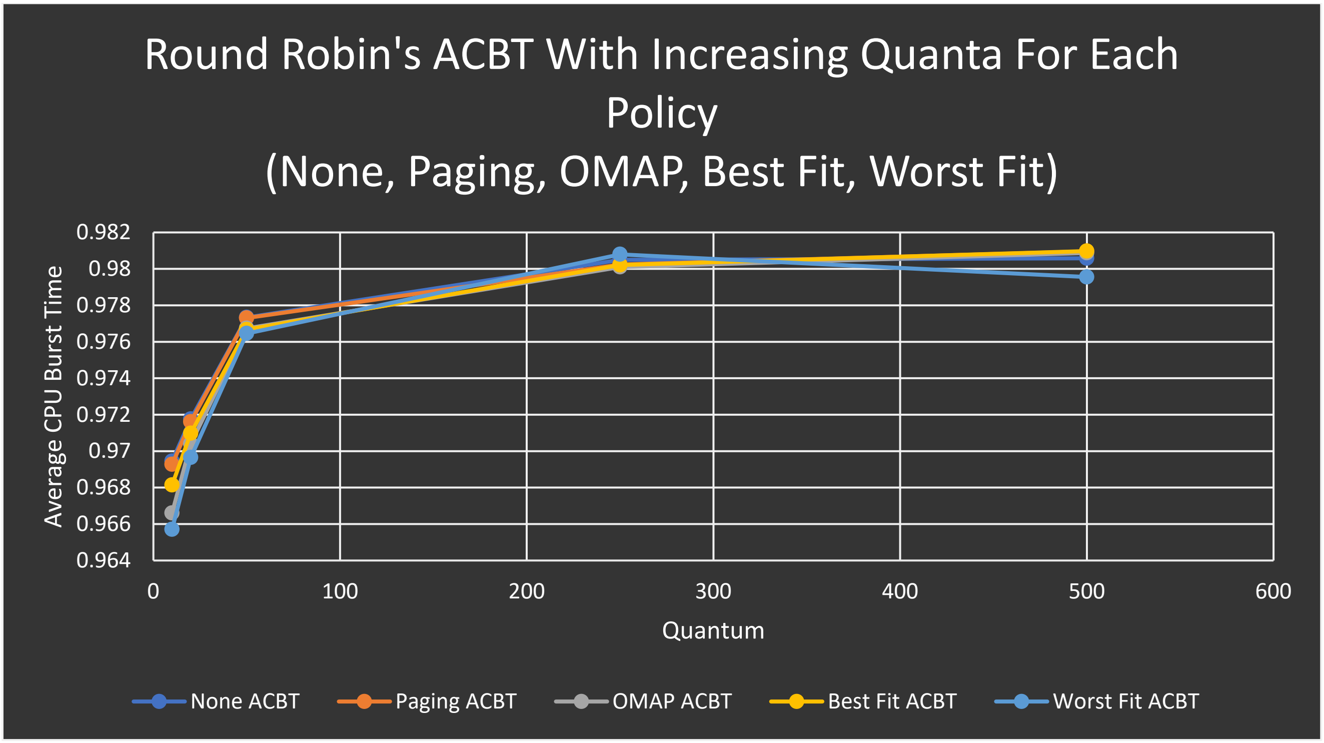


Chart 4

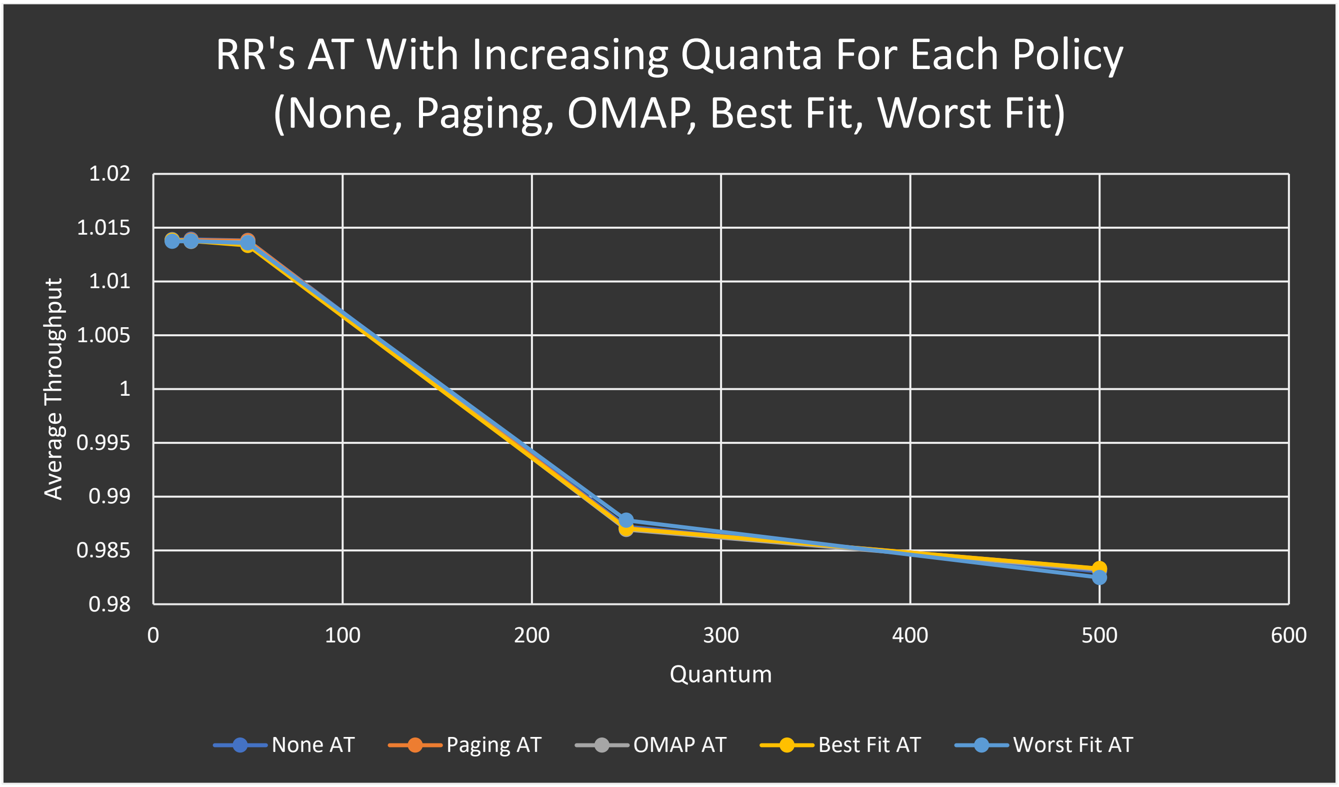


Chart 5

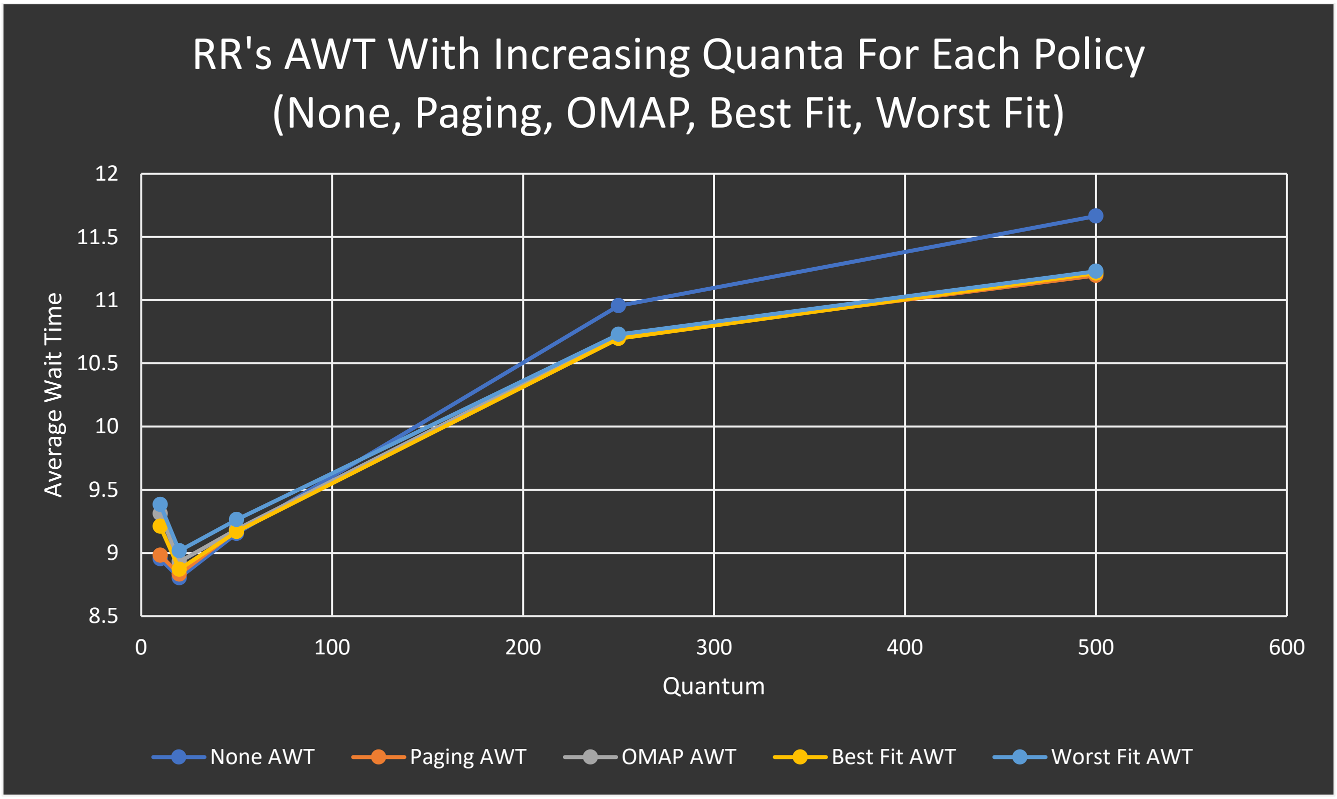
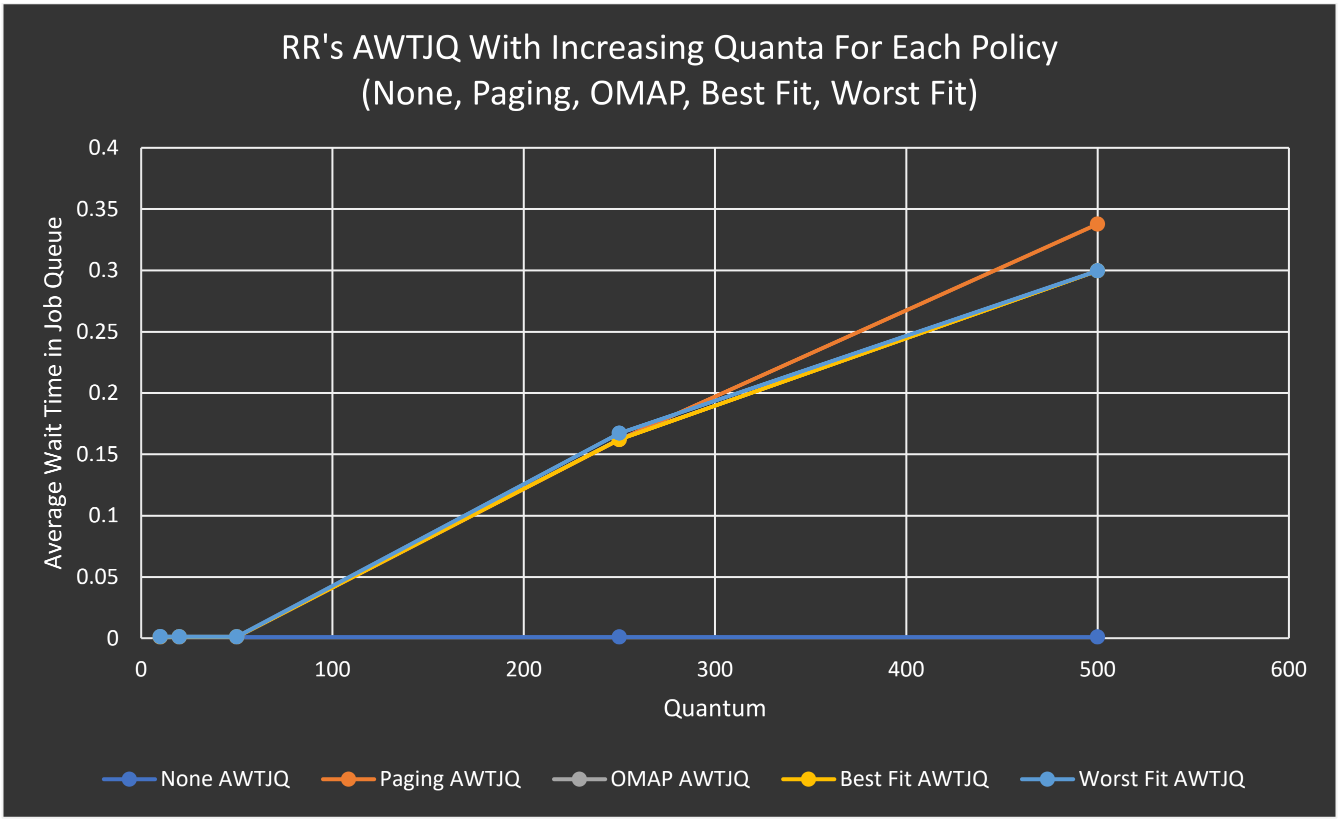


Chart 6


RR's AWTJQ With Increasing Quanta For Each Policy
(None, Paging, OMAP, Best Fit, Worst Fit)
Average Wait Time in Job Queue
Quantum
None AWTJQ
Paging AWTJQ
OMAP AWTJQ
Best Fit AWTJQ
Worst Fit AWTJQ