Abstract

To learn more about the different scheduling policies, I devised three test programs. One program was CPU intensive. The second program was I/O intensive. The third program was a mixture of the two other programs. I then ran each program with a different scheduler and number of processes. I measured the elapsed time, cpu usage, and number of context switches for all tests. The data I yielded suggests that the SCHED_FIFO Policy is the most efficient use of CPU but not the best choice when running simultaneous processes. The SCHED_OTHER and SCHED_RR policies are both ideal for running simultaneous processes, but the SCHED_OTHER policy is the quickest and scales the best.

Introduction

There are many scheduling policies available to an Operating System, each with its own pros and cons and specialties. The purpose of these tests is to compare three rather common scheduling algorithms used by the Linux Operating System. In order to do this I wrote three test programs that each tested a different facet of the scheduling algorithms. I ran each combination of test (twenty-seven in total) ten times (two hundred seventy tests). I compared the data from each test program to determine what sorts of processes are better suited to each scheduling algorithm and how each scheduling algorithm scaled.

Method

To gather my data I wrote three test programs and a script to run them.

The first test program is a modified version of the pi.c file given as an example program. It calculates Pi through 100,000,000 iterations. It takes two command line arguments: the scheduling policy and the number of processes to run. The file then sets the scheduling policy and forks itself based on the number of processes it needs to run.

The second test program is a modified version of the rw.c file given as an example program. It reads a specified number of bytes from a file and then writes these bytes to another file. It takes up to 6 arguments, where the first 2 are mandatory. The first argument is the scheduling policy. The second is the number of processes to run. The third argument is the number of bytes to transfer. The fourth argument is how many bytes can be transferred at a time. The fifth argument is the input file name. The sixth argument is the output file name. It reads and writes 1,024,000 Bytes in 1,024-Byte size chunks.

The third test program is a mixture of the pi.c and rw.c programs. The program is stored in mixed.c It takes the same arguments as rw.c. It alternatively calculates Pi through 100,000,000 iterations and reads and writes 1,024,000 Bytes in 1,024-Byte size chunks. It does this 10 times.

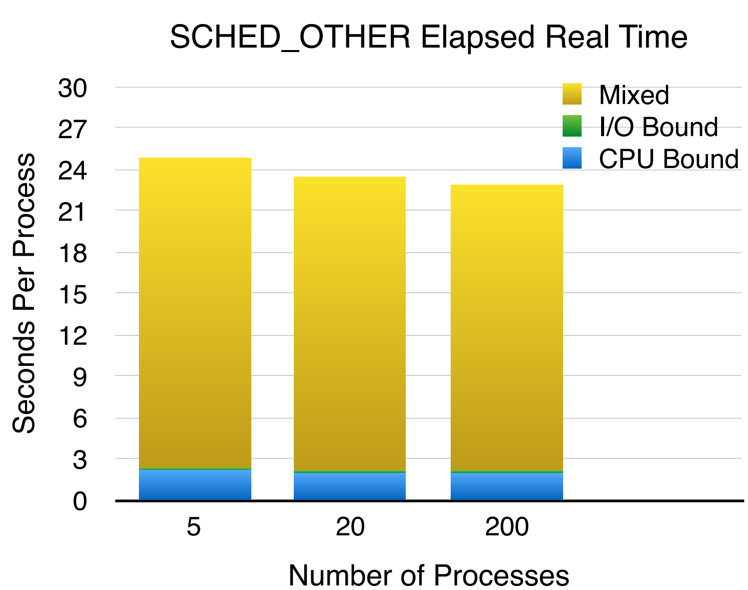
The script runs all 27 test combinations consecutively once. It records each test combination into one of three files: cpu_results for the CPU-bound results, io_results fro the I/O-bound results, and mixed_results for the mixed program results. The results are taken from the linux time command.

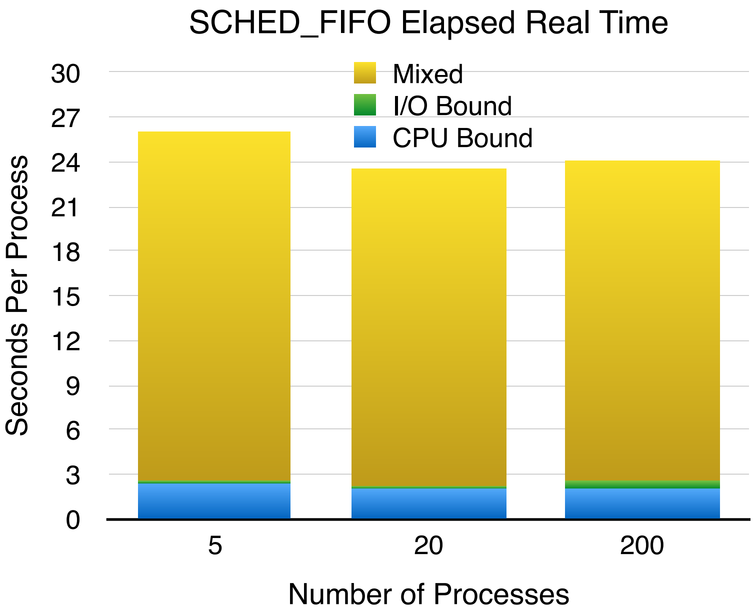
I tested scalability by running the programs using five processes, twenty processes, and two hundred processes. Each scheduling policy and program type is tested with each level of scalability.

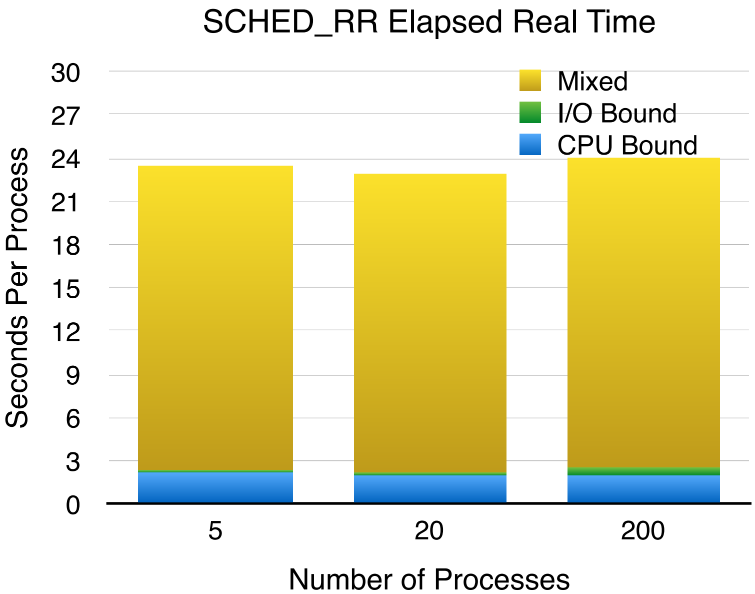
I ran these tests ten times to minimize faulty data. In total I ran 270 tests.

I ran these tests on the Computer Science Virtual Machine, which I run in VMWare Fusion on my MacBook Pro. During the tests I closed all other applications on my Mac and also turned off both the Wi-Fi and Bluetooth. Since all of my files are located inside Dropbox, this stopped Dropbox from trying to sync the many generated output files. In the Virtual Machine I had only the terminal and Sublime Text open.

Results







Each scheduling policy resulted in overall similar trends, but slight variations exist within the data. After calculating the average case for each of the twenty seven tests, I then calculated the per-process results for each test.

Elapsed Real Time

5 PROCESSES

Overall, the SCHED_RR Policy is the fastest scheduler for five simultaneous processes, with a total running time of 23.51 seconds per process.

The slowest policy is the SCHED_FIFO Policy, with a total running time of 26 seconds per process.

For CPU Bound programs, the SCHED_RR policy is quickest, with an average time of 2.13 seconds per process. The slowest policy is the SCHED_FIFO policy, with an average time of 2.3 seconds per process.

For I/O Bound programs, the SCHED_FIFO Policy is quickest, with an average time of 0.169 seconds per process. The slowest policy is the SCHED_RR policy, with an average time of 0.25 seconds per process.

For Mixed programs, the SCHED_RR Policy is quickest, with an average time of 21.13 seconds per process. The slowest policy is the SCHED_OTHER policy, with an average time of 22.43 seconds per process.

20 PROCESSES

Overall, the SCHED_RR Policy is quickest when running twenty processes, with a total time of 22.92 seconds per process.

The slowest policy is the SCHED_FIFO Policy, with a total running time of 23.51 seconds.

For CPU Bound programs, the quickest policy is the SCHED_OTHER Policy, with an average time of 1.96 seconds per process. The slowest policy is the SCHED_RR Policy, with a time of 2.13 seconds per process.

For I/O Bound programs, the quickest policy is the SCHED_FIFO Policy, with an average time of 0.16 seconds per process. The slowest policy is the SCHED_RR Policy with 0.25 seconds per process.

For Mixed programs, the SCHED_RR Policy is quickest, with an average time of 20.76 seconds per process. The slowest policy is the SCHED_FIFO policy, with an average time 21.34 seconds per process.

200 PROCESSES

Overall, the SCHED_OTHER Policy is quickest when running two hundred processes, with a total time of 22.99 seconds per process.

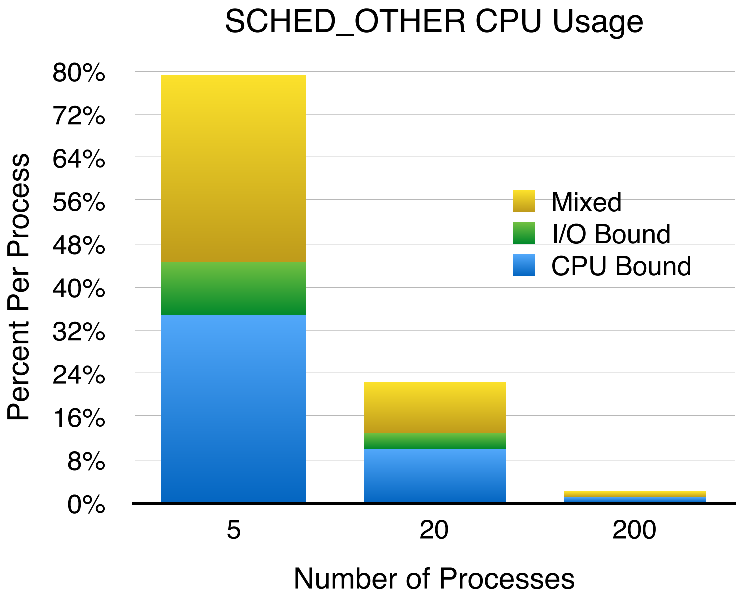
The slowest policy is the SCHED_FIFO Policy, with a total running time of 24.01 seconds.

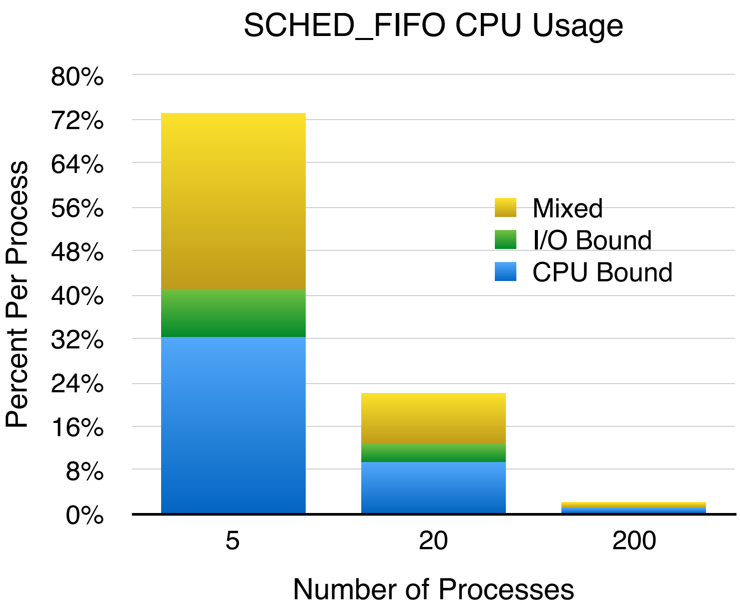
For CPU Bound programs, the quickest policy is the SCHED_OTHER Policy, with an average time of 1.92 seconds per process. The slowest policy is the SCHED_RR Policy with an average time of 2.02 seconds per process.

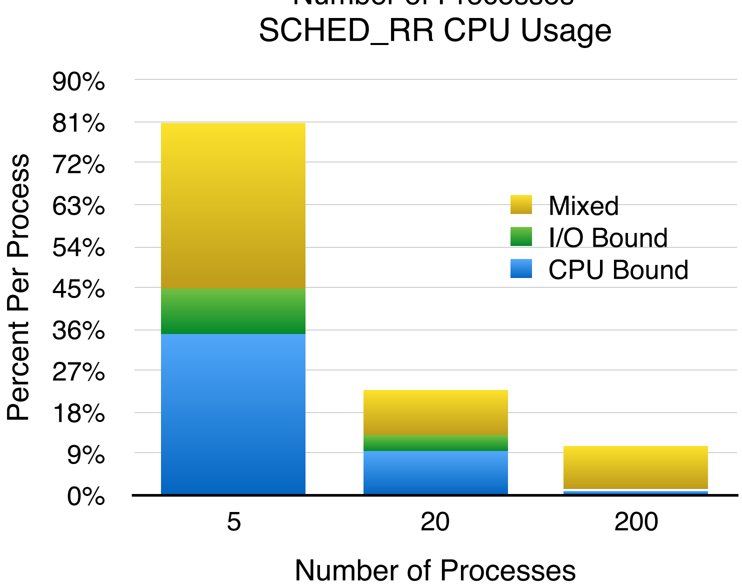
For I/O Bound programs, the SCHED_OTHER Policy is the quickest policy, with an average time of 0.21 seconds per process. The slowest policy is the SCHED_RR policy, with an average time of 0.54 seconds per process.

For Mixed programs, the SCHED_OTHER Policy is the quickest policy, with an average time of 20.86 seconds per process. The SCHED_FIFO policy is the slowest, with an average time of 21.48 seconds per process.

CPU Usage







5 PROCESSES

Overall, the SCHED_FIFO Policy consumed the least amount of CPU while running five processes, with a total amount of 73.04% of the CPU per process. The SCHED_RR policy used the most CPU for five simultaneous processes, with a total of 80.5% of the CPU per process.

For CPU Bound programs, the SCHED_FIFO Policy consumed the least amount of CPU, using an average of 32.06% of the CPU per process. The SCHED_RR Policy consumed the most CPU, using an average of 34.88% per process.

For I/O Bound programs, the SCHED_FIFO Policy consumed the least amount of CPU, using an average of 8.86% of the CPU per process. The SCHED_RR Policy consumed the most CPU, using an average of 9.74% of the CPU per process.

For Mixed programs, the SCHED_FIFO Policy consumed the least amount of CPU, using an average of 32.12% of the CPU per process. The SCHED_RR Policy used the most CPU, approximately 35.88% of the CPU per process.

20 PROCESSES

Overall, the SCHED_FIFO Policy consumed the least amount of CPU while running twenty processes, with a total of 22.11% of the CPU per process. The SCHED_RR Policy consumed the most CPU, with a total of 22.57% of the CPU per process.

For CPU Bound programs, the SCHED_FIFO Policy consumed the least amount of CPU, with an average of 9.24% of the CPU per process. The SCHED_OTHER Policy consumed the most CPU, with an average of 9.91% of the CPU per process.

For I/O Bound programs, the SCHED_OTHER Policy used the least amount of CPU, approximately 3.055% of the CPU per process. The SCHED_RR Policy consumed the most CPU, using an average of 3.89% of the CPU per process.

For Mixed programs, the SCHED_FIFO Policy consumed the least amount of CPU, approximately 9.12% of the CPU per process. The SCHED_RR Policy consumed the most CPU, using an average of 9.37% of the CPU per process.

200 PROCESSES

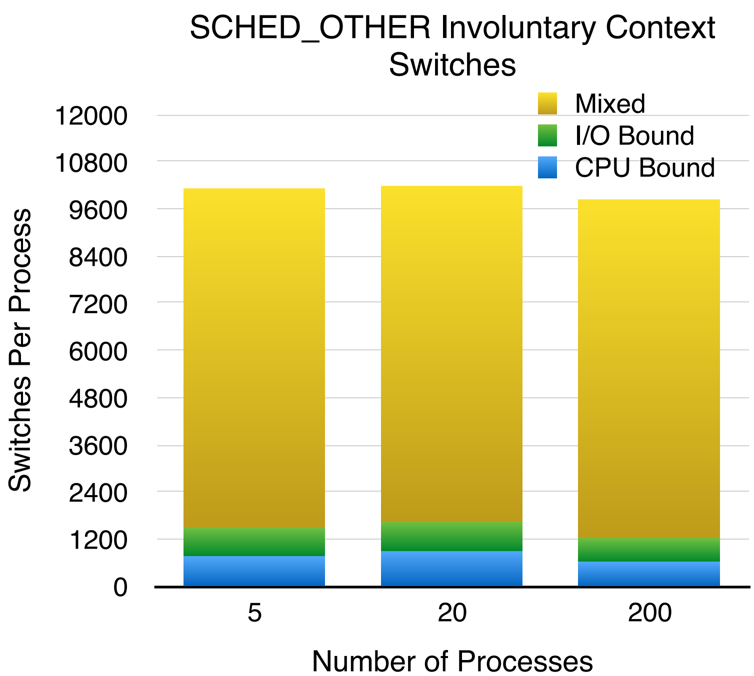
Overall, the SCHED_FIFO Policy consumed the least CPU while running 200 processes, with a total amount of 2.124% of CPU per process. The SCHED_RR Policy consumed the most CPU, using 10.61% of the CPU per process.

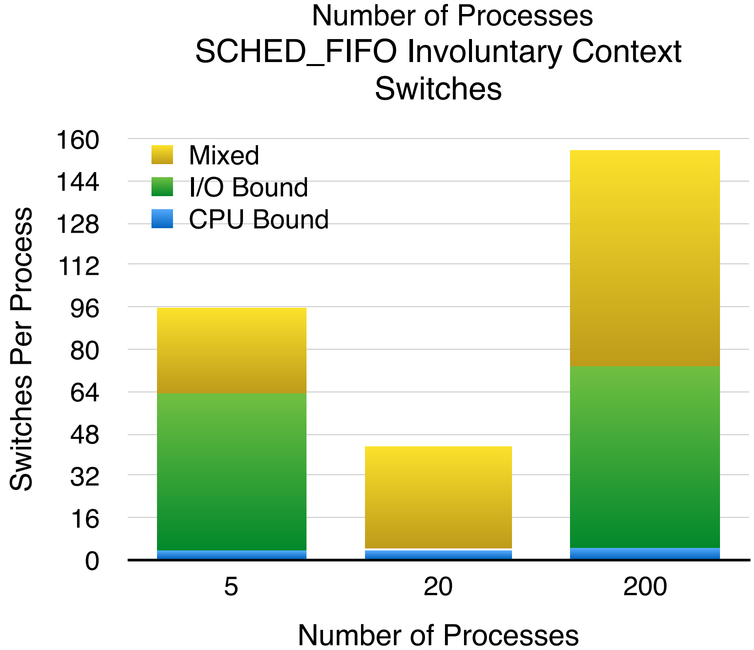
For CPU Bound programs, the SCHED_FIFO Policy consumed the least amount of CPU. It used 0.94% of the CPU per process. The SCHED_OTHER Policy used the most CPU, using 0.992% of the CPU per process.

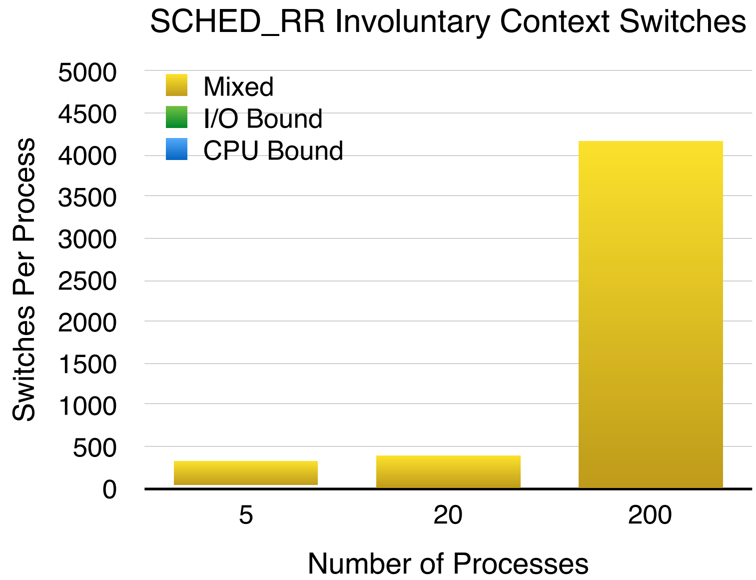
For I/O Bound programs, the SCHED_RR Policy used the least CPU, approximately 0.19% of the CPU per process. The SCHED_FIFO policy used the most CPU, approximately 0.24% of the CPU per process.

For Mixed programs, the SCHED_FIFO Policy used the least CPU, approximately 0.943% of the CPU per process. The SCHED_RR Policy used the most CPU per process, approximately 9.47% of the CPU per process.

Involuntary Context Switches







5 PROCESSES

Overall, the SCHED_FIFO Policy has the least involuntary context switches while running five simultaneous processes, with a total of 96 switches per process. The SCHED_OTHER Policy has the most Involuntary Context Switches, with a total of 10,093 switches per process.

For CPU Bound programs, the SCHED_FIFO Policy has the least involuntary context switches, about 3 switches per process. The SCHED_OTHER Policy has the most, with 775 context switches per process.

For I/O Bound programs, the SCHED_RR Policy has the least involuntary context switches, about 0 per process. The SCHED_OTHER Policy has the most, about 696 switches per process.

For Mixed programs, the SCHED_FIFO Policy has the least switches, about 33 per process. The SCHED_OTHER Policy has the most switches, about 8,622 per process.

20 PROCESSES

Overall, the SCHED_FIFO Policy has the least number of involuntary context switches while running twenty simultaneous processes, a total of about 43 switches per process. The SCHED_OTHER Policy has the most involuntary context switches, a total of about 10,155 switches per process.

For CPU Bound programs, the SCHED_FIFO scheduling process has the least amount of context switches, about 4 switches per process. The SCHED_OTHER process has the most context switches, about 887 switches per process.

For I/O Bound programs, the SCHED_FIFO and SCHED_RR Policies both have the least amount of context switches, about 0 per process. However, SCHED_FIFO has the smaller number of 0.21 switches per process, as opposed to SCHED_RR's 0.35 switches per process. The SCHED_OTHER Policy has the most context switches, about 727 switches per process.

For Mixed programs, the SCHED_FIFO Policy has the least amount of involuntary context switches, about 39 switches per process. The policy with the most involuntary context switches is the SCHED_OTHER policy, with about 8,541 context switches per process.

200 PROCESSES

Overall, the SCHED_FIFO Policy has the least number of involuntary context switches while running 200 simultaneous processes, totaling about 155 switches per process. The SCHED_OTHER Policy has the most involuntary context switches, totaling about 9,816 switches per process.

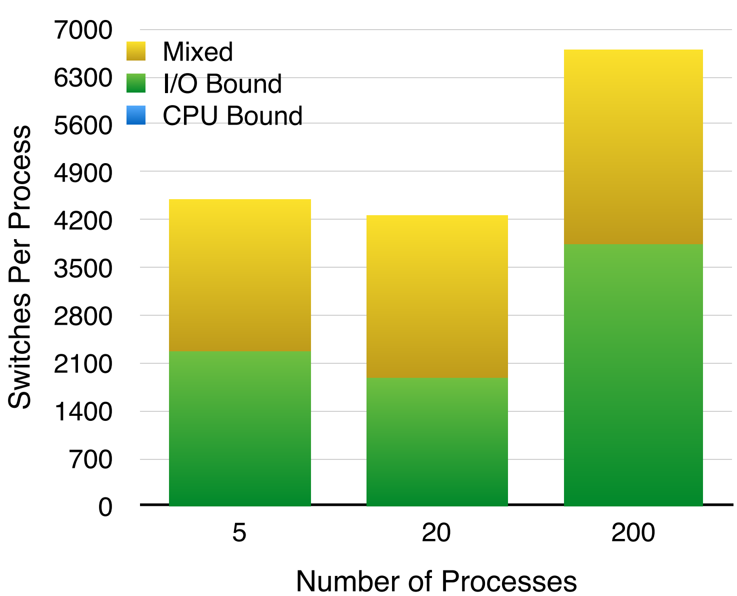
For CPU Bound programs, the SCHED_FIFO Policy has the least switches, with about 4 switches per process. The SCHED_OTHER Policy has the most switches, about 629 switches per process.

For I/O Bound programs, the SCHED_RR Policy used the least switches, with about 0 switches per process. The SCHED_OTHER Policy has the most switches, about 609 switches per process.

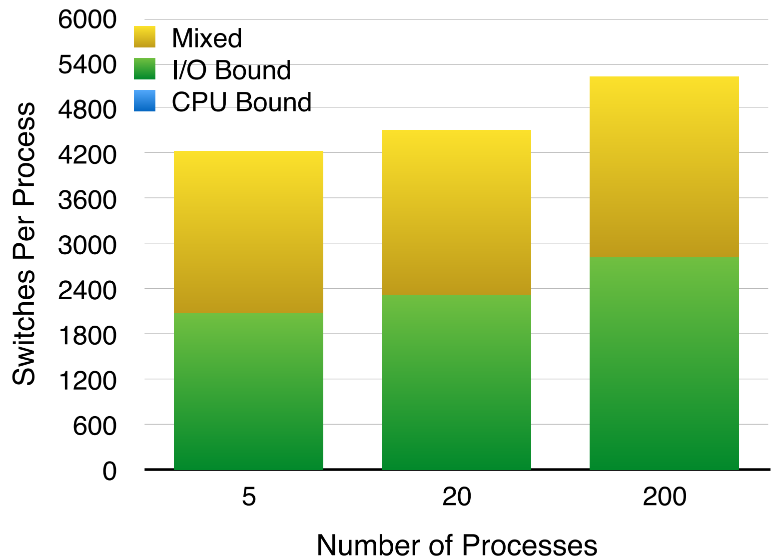
For Mixed programs, the SCHED_FIFO Policy has the least involuntary switches, about 82 per process. The SCHED_OTHER Policy has the most involuntary switches, about 8,578 per process.

Voluntary Context Switches

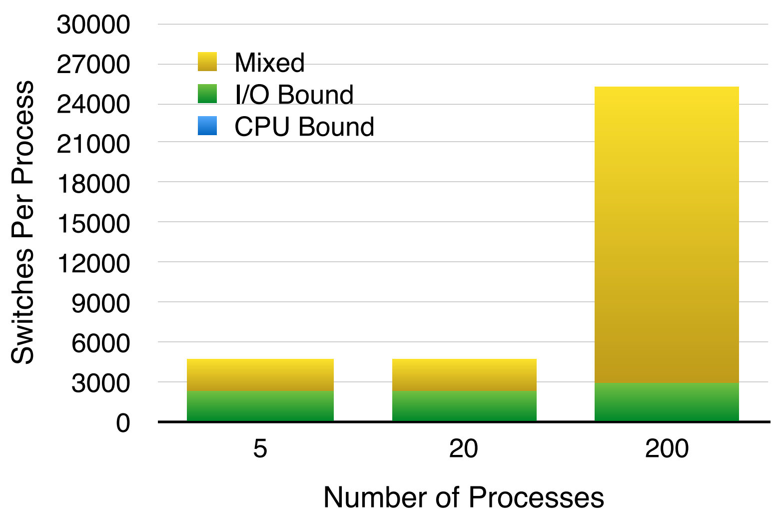
SCHED_OTHER Voluntary Context Switches



SCHED_FIFO Voluntary Context Switches



SCHED_RR Voluntary Context Switches



5 PROCESSES

Overall, the SCHED_FIFO Policy has the least number of voluntary context switches, totaling about 4,241 switches per process. The SCHED_RR Policy has the most voluntary context switches, totaling about 4,694 context switches.

For CPU Bound programs, the SCHED_OTHER Policy has the least number of voluntary context switches, about 2 switches per process. The SCHED_FIFO Policy has the most voluntary context switches, about 4 per process.

For I/O Bound programs, the SCHED_FIFO Policy has the least number of voluntary context switches, about 2,085 switches per process. The SCHED_OTHER Policy has the most voluntary context switches, about 2,279 switches per process.

For Mixed programs, the SCHED_FIFO Policy has the least voluntary context switches, about 2,152 switches per process. The SCHED_RR

Policy has the most voluntary context switches, about 2,451 per process.

20 PROCESSES

Overall, the SCHED_OTHER Policy has the least number of voluntary context switches while running twenty processes. The switches total about 4,274 switches per process. The SCHED_RR Policy has the most voluntary context switches, totaling about 4,794 context switches per process.

For CPU Bound programs, all three policies have an average context switch of about 1 per process. The SCHED_FIFO Policy has the smallest number with 1.25 switches per process, followed by the SCHED_OTHER Policy with 1.34 switches per process, and finished with the SCHED_RR Policy with 1.36 switches per process.

For I/O Bound programs, the SCHED_OTHER Policy has the least voluntary switches, about 1,890 switches per process. The SCHED_RR Policy has the most voluntary switches, about 2,353 switches per process.

For Mixed programs, the SCHED_FIFO Policy has the least voluntary switches, about 2,172 switches per process. The SCHED_RR Policy has the most switches, about 2,440 switches per process.

200 PROCESSES

Overall, the SCHED_FIFO Policy has the least number of voluntary context switches while running 200 simultaneous processes. It switches a total of 5,222 times per process.

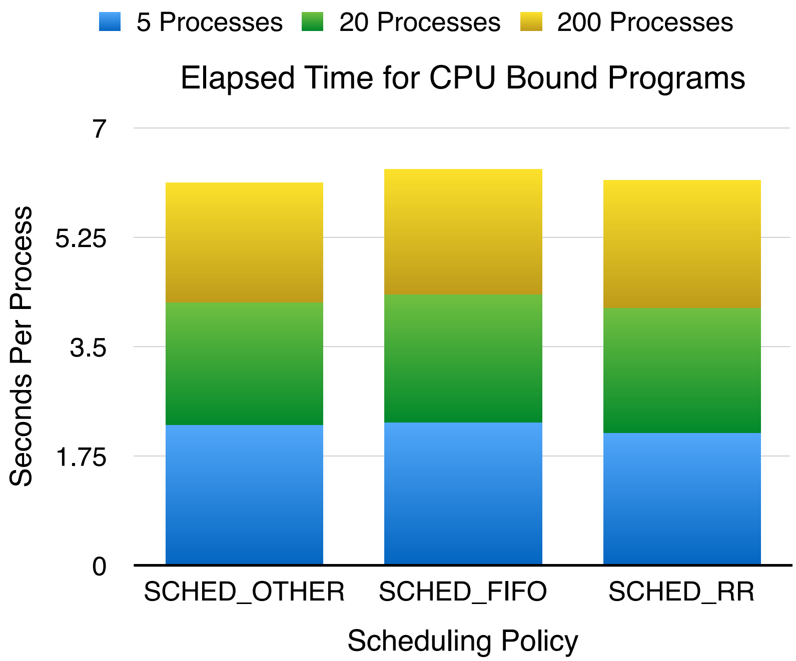
For CPU Bound programs, all three policies have an average context switch of about 1 per process. The SCHED_FIFO Policy has the smallest number with 1.03 switches per process, followed by the SCHED_RR Policy with 1.04 switches per process, and finished with the SCHED_OTHER Policy with 1.08 switches per process.

For I/O Bound programs, the SCHED_RR Policy and SCHED_FIFO Policy have the least number of voluntary context switches, about 2,820 per process. The SCHED_OTHER Policy has the most voluntary context switches, about 3,817 switches per process.

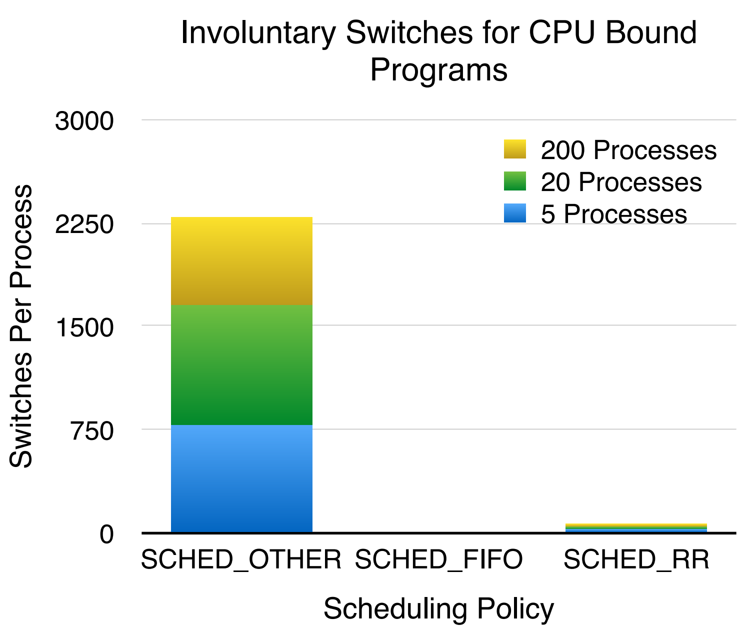
For Mixed programs, the SCHED_FIFO Policy has the least number of voluntary context switches, about 2,401 switches per process. The SCHED_RR Policy has the most voluntary context switches, about 22,324 switches per process.

Analysis

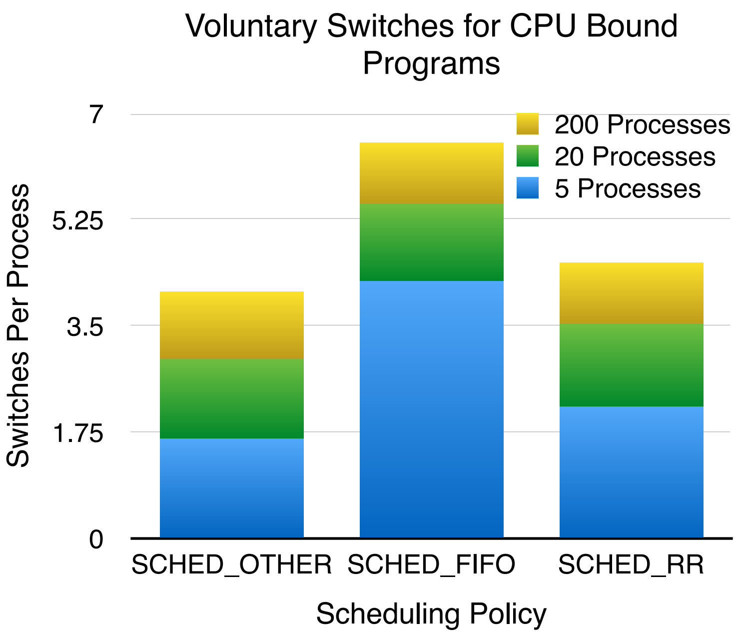
CPU Bound Processes



For CPU Bound processes, no policy is drastically faster than another. Each policy gave a total of about 6 seconds per process. However, of the three scheduling policies, the SCHED_OTHER Policy undergoes many more context switches per process, a total of about 2,295. Thus, even with a significant overhead, the SCHED_OTHER Policy is quite fast.



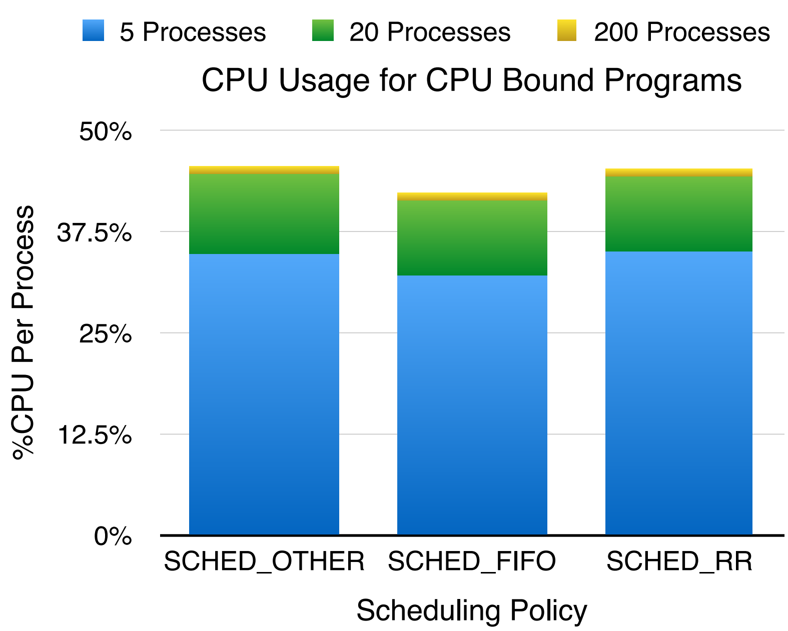
Comparatively, the SCHED_RR and SCHED_FIFO Policies took about the same amount



of time with far less context switches: about 18 per process for SCHED_FIFO and 75 per process for SCHED_RR. Based on time and overhead, then, CPU Bound processes can be well handled by any of the three policies.

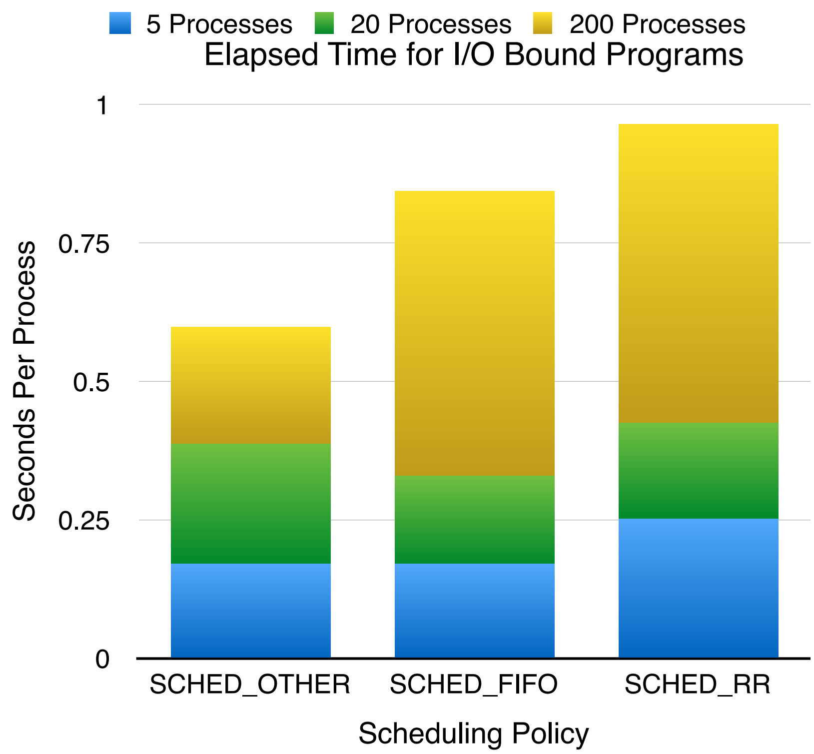
The CPU is best utilized by the SCHED_FIFO Policy, which has little context switching and so wastes less CPU in context switch overhead. The SCHED_FIFO Policy has about 0.42 context switches for every percentage of CPU, or 1 switch for every 2% use of the CPU.

The SCHED_RR Policy has about 1.65 context switches for every percentage of CPU, or about 3 switches for every 2% of the CPU. The SCHED_OTHER Policy has about 50.35 switches for every percentage of CPU.



Therefore, while the SCHED_FIFO Policy used less CPU overall, the SCHED_OTHER Policy most efficiently uses the CPU to where context switch overhead is negligible.

I/O Bound Processes



For I/O Bound Processes, the SCHED_OTHER Policy is drastically faster than the SCHED_FIFO and SCHED_RR Policies. The SCHED_OTHER Policy is faster because for the I/O processes it scaled better overall. The SCHED_OTHER Policy can run 200 simultaneous I/O processes much quicker and more efficiently than SCHED_FIFO and SCHED_RR.

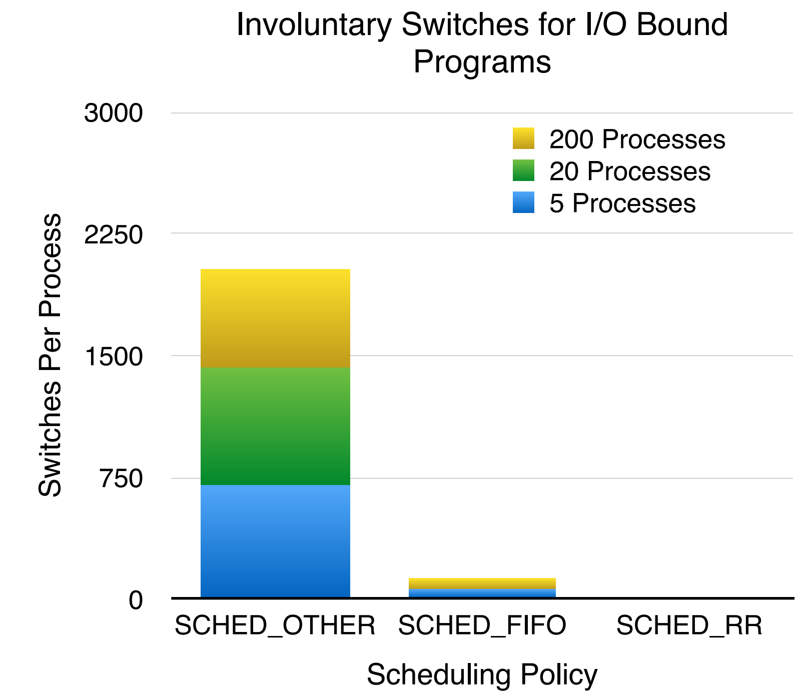
However, for 20 or less processes, SCHED_FIFO is quicker.

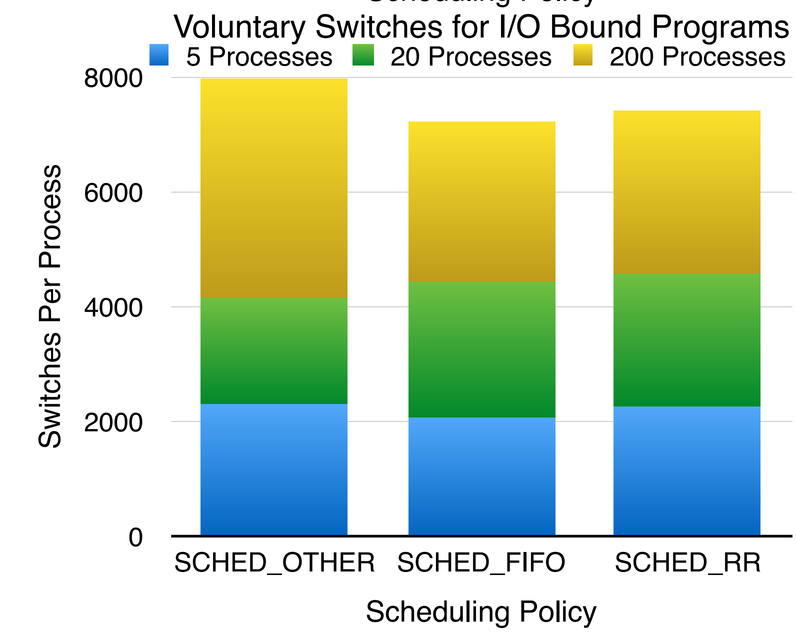
The SCHED_OTHER Policy still has the most context switches, about 10,017 per process. This means that it has 770.61 switches for every 1% of CPU used.

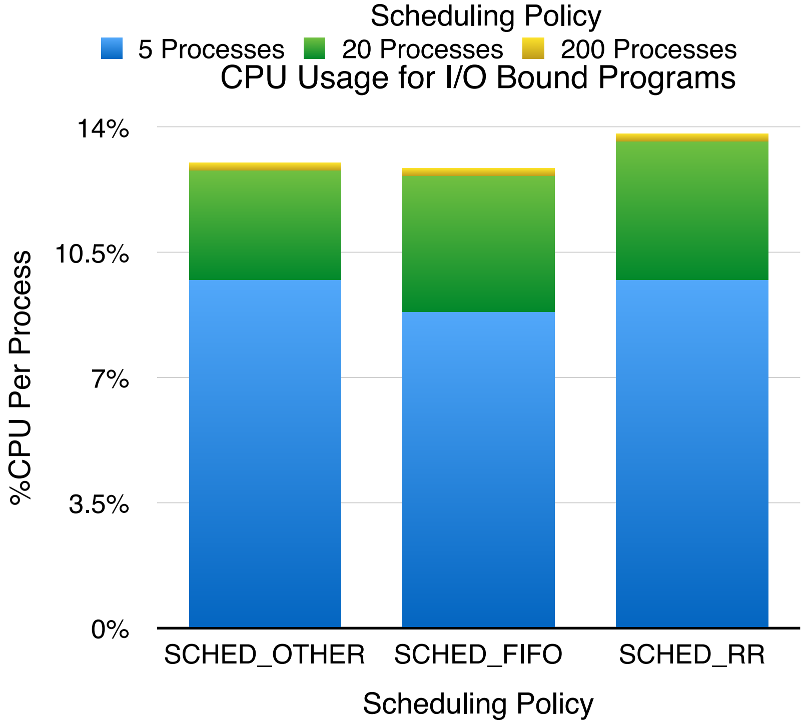
The SCHED_FIFO Policy has about 7,373 switches per process, which is 574.16 switches for every 1% of CPU used.

The SCHED_RR Policy has about 7,414 switches per process, which is 536.51 switches for every 1% of the CPU used.

Therefore, the SCHED_OTHER Policy most efficiently uses the CPU to where context switch overhead is negligible and the I/O work can finish

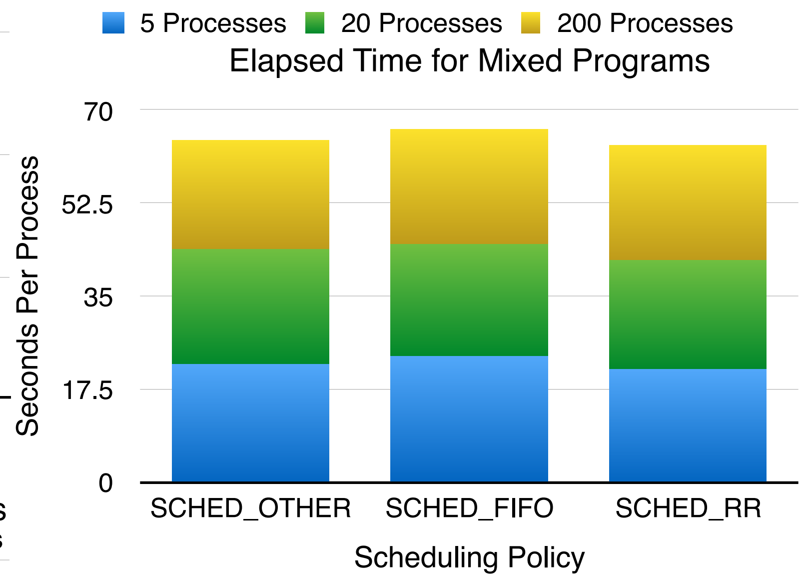




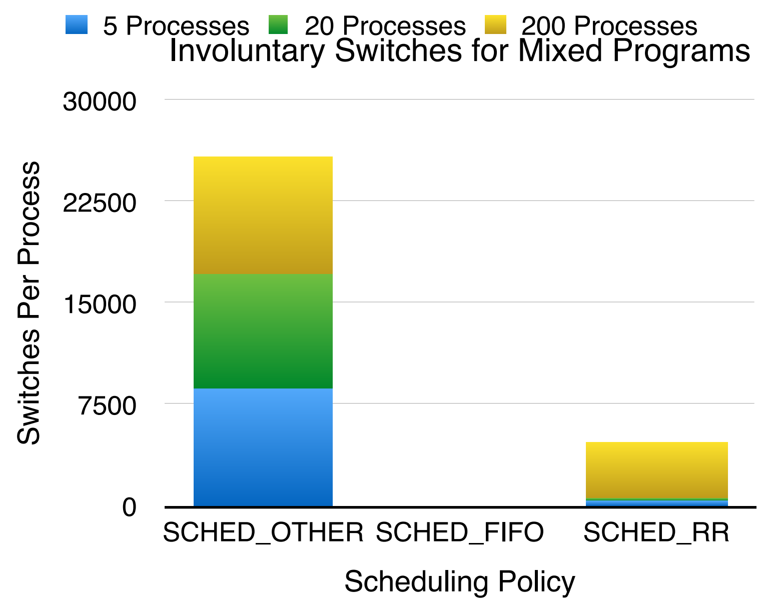


quickly.

Mixed Processes



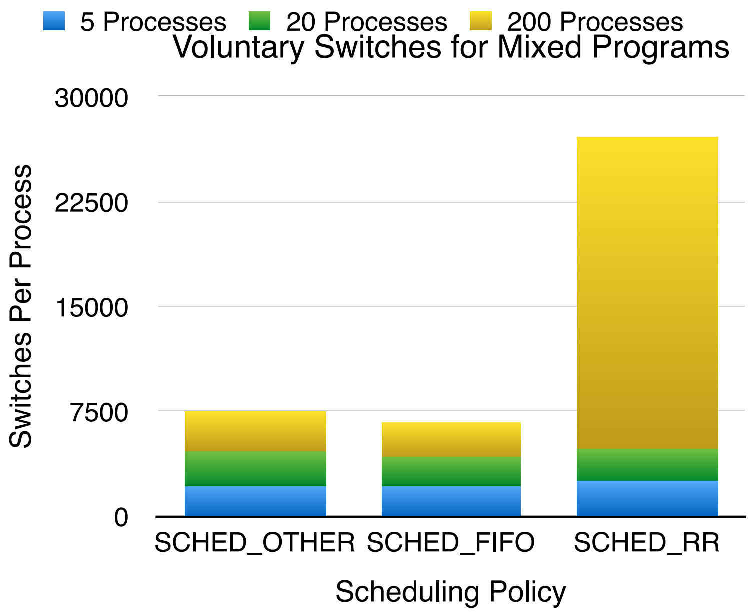
For the mixed processes each scheduling algorithm results in rather similar elapsed times. The SCHED_FIFO algorithm is the slowest, and the SCHED_RR algorithm is the quickest.



The SCHED_OTHER Policy has about 33,251 switches per process, which is 741.94 switches for every 1% of CPU used.

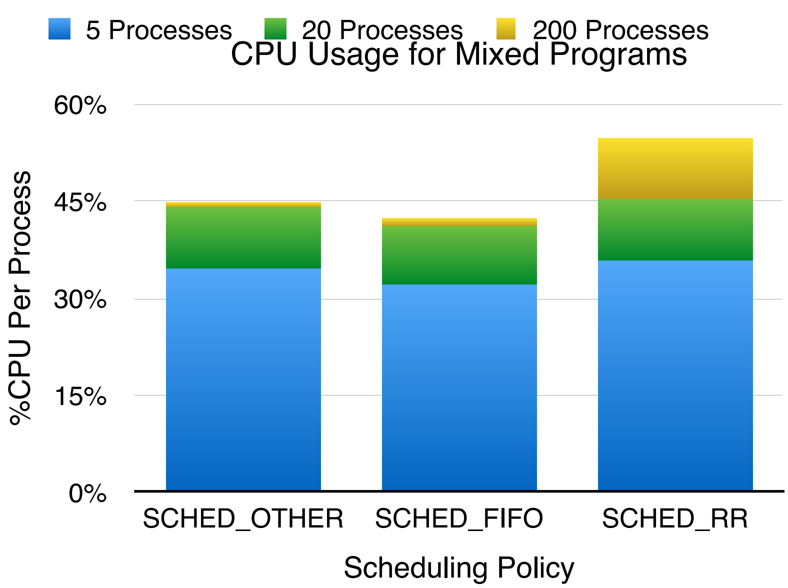
The SCHED_FIFO Policy has about 6,878 switches per process, which is 163.05 switches for every 1% of CPU used.

The SCHED_RR Policy has about 32,050 switches per process, which is 585.7 switches for every 1% of CPU used.



The SCHED_FIFO Policy used the least CPU, while the SCHED_RR Policy used the most. As the number of processes increased, the SCHED_OTHER and SCHED_FIFO Policies decreased the amount of CPU per process. The SCHED_RR Policy improved CPU usage from 5 processes to 20 processes, but did not improve from 20 processes to 200 processes.

Therefore, the SCHED_OTHER Policy most efficiently uses the CPU to where context switch overhead is negligible and the process can finish quickly.



Conclusion

Each policy has its own pros and cons. I will outline some of each for each policy.

SCHED_FIFO

In terms of run time, the SCHED_FIFO Policy does not scale very well. For CPU bound processes, the amount of time per process decreases as more processes are added, but imperceptibly so. For I/O processes, the amount of time per process increases, quite dramatically. Mixed processes scale the same way as CPU processes.

The SCHED_FIFO Policy uses less CPU and has less context switches than the other policies. It also has a high priority ("sched_setscheduler(2) - linux," 2013). This makes it ideal for processes involving interaction with the user, as these processes will be moved to the top of the run queue and will have very quick response time. A good example is a word processor, such as Microsoft Word or Apple Pages.

However, because a SCHED_FIFO task cannot be interrupted ("sched_setscheduler(2) - linux," 2013), if the task is CPU intensive at all it quickly takes over all other resources and does not allow other processes to run. This is very detrimental to efficiently utilizing the CPU, as only one process at a time can be completed.

SCHED_OTHER

In terms of run time, the SCHED_OTHER Policy actually scales fairly well. The more processes there are, the less time per process is needed. For CPU bound processes, the amount of time per process stays about the same as the number of processes increases. For I/O bound processes, the amount of time per process increases, but imperceptibly so. For Mixed processes, the amount of time per process stays about the same decreasing slightly.

The SCHED_OTHER Policy is very fast and uses a moderate amount of CPU. It has the most context switches, and therefore the most context switch overhead. This is because of the low priority of the SCHED_OTHER Policy ("sched_setscheduler(2) - linux," 2013). This is ideal for processes that run in the background, for example, an app like Dropbox. The background process will be scheduled in such a way that it will be completed rather quickly and it will not take away valuable time from other more important processes.

However, the sheer number of context switches creates a very large overhead, during which the CPU does nothing productive and simply wastes time.

SCHED_RR

In terms of run time, the SCHED_RR Policy does not scale very well. It actually becomes faster going from a single digit number of processes to tens of processes, but then becomes worse for running hundreds of processes. For CPU bound processes, the amount of time per process stays about the same as the number of processes increases. For I/O bound processes, the amount of time per process decreases in the tens of processes but then substantially increases in the hundreds of processes. For Mixed processes, the amount of time per process stays about the same.

The SCHED_RR Policy is not the fastest nor the most CPU efficient. However, its high priority ("sched_setscheduler(2) - linux," 2013) and use of round robin time slices means that it can run multiple processes at once and push them through faster than SCHED_FIFO. This makes it ideal for processes that require a fast reaction time but not necessarily are I/O bound, for example a web browser.

However, as the SCHED_RR still has many less context switches than the SCHED_OTHER Policy, meaning that it will be slightly slower.

References

sched_setscheduler(2) - linux manual page. (2013, 09 17). Retrieved from http://man7.org/linux/man-pages/man2/sched_setscheduler.2.html

Appendix A

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
		5	9.76	19.23	0.1	198%	2992	7
			12.24	19.73	0.18	162%	3729	8
			11.66	18.8	0.48	165%	4595	8
			11.67	19.05	0.25	165%	4414	8
			11.93	18.97	0.3	161%	4555	8
			11.7	19.01	0.48	166%	4603	7
			11.57	18.9	0.38	166%	4288	11
			12.51	19.17	0.26	155%	4644	8
			9.69	19.15	0.12	198%	2434	9
			9.7	19.09	0.14	198%	2513	8
		Average	11.243	19.11	0.269	173.4%	3876.7	8.2
		Per Process	2.2486	3.822	0.0538	34.68%	775.34	1.64
			40.15	78.96	0.25	197%	20676	28

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
	CPU Bound	20	38.98	77.32	0.22	198%	14351	24
			39.06	77.41	0.26	198%	19698	31
			39.32	77.56	0.18	197%	19540	28
			39.01	77.34	0.22	198%	20042	23
			39.05	77.29	0.36	198%	19834	31
			39.39	78	0.08	198%	19656	27
			38.91	77.33	0.16	199%	17474	24
			38.88	76.93	0.58	199%	11810	26
			39.28	77.81	0.5	199%	14311	26
		Average	39.203	77.595	0.281	198.1%	17739.2	26.8
		Per Process	1.96015	3.87975	0.01405	9.905%	886.96	1.34
		200	447.07	854.91	10	193%	229840	244
			372.81	736.32	5.69	199%	103417	215
			374.33	740.49	4.48	199%	127666	205
			386.62	764.84	4.66	199%	126510	210
			374.99	742.19	4.28	199%	128700	216
			373.82	740.38	3.78	199%	131004	218
			382.19	759.6	2.27	199%	108723	207
			382.14	759.74	1.96	199%	118173	213
			373.02	737.14	6.7	199%	90180	220
			373.55	735.86	8.94	199%	92829	217
		Average	384.054	757.147	5.276	198.4%	125704.2	216.5
		Per Process	1.92027	3.785735	0.02638	0.992%	628.521	1.0825
			1.8	0	0.96	53%	2962	12360
			1.21	0	0.36	29%	3551	11970

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
SCHED_OTHER	I/O Bound	5	0.7	0	0.33	47%	2317	10436
			0.8	0	0.34	42%	4675	11128
			0.7	0	0.34	48%	2243	10249
			0.7	0	0.31	44%	5074	10482
			0.74	0	0.37	50%	4179	10775
			0.69	0	0.34	50%	3697	10926
			0.63	0	0.39	61%	3347	13683
			0.61	0	0.38	62%	2767	11916
		Average	0.858	0	0.412	48.6%	3481.2	11392.5
		Per Process	0.1716	0	0.0824	9.72%	696.24	2278.5
		20	21.32	0.01	15.25	71%	15734	49068
			2.19	0	1.36	62%	12429	33618
			2.62	0	1.27	48%	14684	35680
			3.33	0	1.28	38%	12313	37121
			2.05	0	1.28	62%	16920	31696
			3.12	0	1.32	42%	15481	37215
			2.35	0	1.38	58%	17976	33616
			2.53	0	1.28	50%	14219	35863
			1.57	0	1.44	91%	13149	41871
			1.67	0	1.49	89%	12549	42268
		Average	4.275	0.001	2.735	61.1%	14545.4	37801.6
		Per Process	0.21375	0.00005	0.13675	3.055%	727.27	1890.08
			96.31	0.29	75.95	79%	157606	786537
			41.23	0.04	14.13	34%	133012	652890
			45.03	0.11	13.62	30%	125630	661257

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
		200	32.52	0.18	13.77	42%	107961	642008
			41.97	0.12	13.32	32%	127583	628413
			35.86	0.01	12.94	36%	138694	657298
			29.66	0.09	13.66	46%	124202	654124
			40.74	0.06	12.82	31%	131332	674011
			23.38	0.12	17.58	75%	91123	1162113
			37.01	0.09	16.64	45%	81284	1114696
		Average	42.371	0.111	20.443	45%	121842.7	763334.7
		Per Process	0.211855	0.000555	0.102215	0.225%	609.2135	3816.6735
		5	111.51	188.18	3.61	171%	44473	11118
			110.05	183.13	6.21	172%	49639	11159
			110.51	183.46	6.23	171%	44827	11619
			110.4	185.12	5.99	173%	44205	10297
			105.99	183.46	5.76	178%	43464	10502
			102.26	184.4	3.76	184%	36567	10196
			111.35	190.59	5.11	175%	48352	10221
			170.3	186.56	6.3	113%	70050	10243
			94.45	185.57	1.43	197%	24625	11739
			94.79	185.68	2	197%	24880	14711
		Average	112.161	185.615	4.64	173.1%	43108.2	11180.5
		Per Process	22.4322	37.123	0.928	34.62%	8621.64	2236.1
			412.79	788.07	8.57	192%	226010	47062
			391.81	733.4	18.53	191%	166367	46251
			398.18	754.86	14.41	193%	176940	45172
			405.42	771.65	12.91	193%	192609	46637

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
	Mixed	20	399.6	758.55	13.37	193%	150399	43259
			608.61	754.42	9.68	125%	193437	46461
			413.62	778.2	16.4	192%	171765	47843
			464.94	781.51	16.42	171%	170515	53414
			379.43	749.39	6.43	199%	132964	48905
			386.36	762.22	6.65	199%	127128	51588
		Average	426.076	763.227	12.337	184.8%	170813.4	47659.2
		Per Process	21.3038	38.16135	0.61685	9.24%	8540.67	2382.96
		200	4230.49	8099.93	75.13	193%	2337160	492544
			4020.79	7791.36	68.3	195%	2027304	653830
			4033.08	7788.75	90.8	195%	1754975	667513
			4102.39	7914.39	90.68	195%	1828016	538309
			4081.87	7830.94	106.86	194%	1823175	551034
			5187.16	7861.74	71.36	152%	1794589	634293
			4077.97	7879.83	102.98	195%	1553248	537185
			4057.27	7856.53	56.25	195%	1663436	520619
			3943.33	7757.8	100.58	199%	1188771	618876
			3984.18	7831.11	108.07	199%	1185205	568902
		Average	4171.853	7861.238	87.101	191.2%	1715587.9	578310.5
		Per Process	20.859265	39.30619	0.435505	0.956%	8577.9395	2891.5525
		5	11.49	18.62	0	162%	16	10
			11.49	18.44	0	160%	18	18
			11.54	18.45	0	159%	15	23
			11.44	18.39	0	160%	14	18
			11.41	18.42	0	161%	14	20

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
			11.58	18.43	0.01	159%	18	24
			11.34	18.38	0.03	162%	17	50
			11.43	18.50	0	161%	18	28
			11.37	18.49	0.01	162%	18	10
			11.88	18.75	0	157%	20	11
		Average	11.497	18.487	0.005	160.3%	16.8	21.2
		Per Process	2.2994	3.6974	0.001	32.06%	3.36	4.24
		20	42.12	76.53	0.56	183%	76	25
			38.81	73.97	0	190%	76	25
			39.58	74.03	0.01	187%	78	25
			38.92	74.11	0	190%	76	25
			39.72	74.08	0.02	186%	79	25
			39.98	74.18	0.02	185%	76	25
	CPU Bound		42.06	74	0	175%	73	25
			39.11	74.24	0	189%	80	25
			39.85	74.26	0.01	186%	78	25
			41.79	74.19	0.01	177%	73	25
		Average	40.194	74.359	0.063	184.8%	76.5	25
		Per Process	2.0097	3.71795	0.00315	9.24%	3.825	1.25
		200	445.84	843.37	0.12	189%	876	205
			390.29	740.08	0.08	189%	776	205
			395.69	746.93	0.06	188%	780	205
			410.64	776.32	0.05	189%	811	206
			394.37	745.81	0.13	189%	785	205
			393.59	742.19	0.05	188%	773	205

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
			409.33	772.37	0.04	188%	805	205
			407.52	768.42	0.13	188%	797	205
			390.87	742.8	0.05	190%	782	205
			393.49	747.71	0.05	190%	786	205
		Average	403.163	762.6	0.076	188.8%	797.1	205.1
		Per Process	2.015815	3.813	0.00038	0.944%	3.9855	1.0255
		5	2.12	0	0.6	28%	2962	12360
			0.71	0	0.33	46%	5	10075
			0.68	0	0.34	51%	4	11600
			0.71	0	0.31	44%	3	10035
			0.73	0	0.34	46%	12	10024
			0.69	0	0.3	45%	2	9992
			0.67	0	0.31	47%	3	10072
			0.68	0	0.32	47%	2	10080
			0.72	0	0.32	44%	6	9992
			0.75	0	0.34	45%	3	10020
		Average	0.846	0	0.351	44.3%	300.2	10425
		Per Process	0.1692	0	0.0702	8.86%	60.04	2085
		20	13.41	0	15.22	113%	5	54124
			1.86	0	1.32	70%	4	40065
			3.77	0	1.3	34%	3	40167
			1.66	0	1.48	89%	7	39964
			1.74	0	1.24	71%	2	48756
			1.74	0	1.44	83%	2	50809
SCHED_FIFO	I/O Bound		1.82	0	1.35	73%	3	40235

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
			1.96	0	1.38	70%	4	39907
			2.02	0	1.46	72%	4	57305
			1.88	0	1.4	74%	7	56362
		Average	3.186	0	2.759	74.9%	4.1	46769.4
		Per Process	0.1593	0	0.13795	3.745%	0.205	2338.47
		200	106.89	0.14	136.97	128%	8	582033
			38.65	0.02	16.95	43%	4	520097
			35.43	0.02	14.29	40%	4	543261
			29.67	0.03	15.48	52%	3	412898
			618.41	0.51	18.77	3%	2	638680
			35.86	0.01	12.94	36%	138694	657298
			20.88	0.01	15.48	74%	6	598443
			63.73	0.40	10.86	17%	27	562560
			37.81	0.02	16.42	43%	2	553761
			40.46	0	15.66	38%	5	571637
		Average	102.779	0.116	27.382	47.4%	13875.5	564066.8
		Per Process	0.513895	0.00058	0.13691	0.237%	69.3775	2820.334
		5	121.24	193.82	0.37	160%	167	10130
			116.27	186	0.42	160%	167	10283
			117.25	187.68	0.48	160%	161	10032
			117.69	188.92	0.5	160%	164	10791
			116.42	186.14	0.54	160%	159	10162
			116.21	186.02	0.36	160%	160	10418
			120.47	193.96	0.5	161%	169	10250
			120.39	194.36	0.45	161%	178	12042

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
			114.5	186.72	0.4	163%	154	10076
			116.27	187.04	0.4	161%	157	13396
		Average	117.671	189.066	0.442	160.6%	163.6	10758
		Per Process	23.5342	37.8132	0.0884	32.12%	32.72	2151.6
		20	446.62	815.57	1.9	183%	816	41198
			402.13	756.13	1.76	188%	780	40965
			425.02	775.25	1.69	182%	775	40329
			436.27	789.6	1.34	181%	775	40505
			427.9	775.34	1.35	181%	766	42020
	Mixed		427.41	770.53	1.93	180%	737	52897
			433.46	795.3	1.96	183%	799	46948
			430.52	789.35	1.66	183%	803	41036
			415.08	755.26	1.57	182%	753	48455
			423.58	767.52	1.49	181%	759	40023
		Average	426.799	778.985	1.665	182.4%	776.3	43437.6
		Per Process	21.33995	38.94925	0.08325	9.12%	38.815	2171.88
		200	4470.14	8411.64	15.54	188%	87778	480246
			4255.25	8033.84	18.52	189%	8411	406860
			4276.332	8050.82	17.46	188%	8421	508938
			4330.65	8178.1	18.44	189%	8552	481598
			4281.59	8098.89	18.3	189%	8485	522321
			4296.53	8086.74	17.76	188%	8467	508719
			4323.46	8144.54	17	188%	8499	452188
			4236.11	7995.14	34.92	189%	8327	533472
			4251.67	8036.55	19.11	189%	8410	500090

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
			4244.13	8024.3	21.17	189%	8404	406899
		Average	4296.5862	8106.056	19.822	188.6%	16375.4	480133.1
		Per Process	21.482931	40.53028	0.09911	0.943%	81.877	2400.6655
		5	10.31	19.16	0	185%	111	11
			10.51	18.64	0	177%	115	11
			11.09	18.44	0	166%	125	10
			10.63	18.47	0	173%	115	10
			11.14	18.44	0	165%	123	11
			11.31	18.51	0.01	163%	122	11
			10.25	18.64	0	181%	116	11
			10.39	18.48	0.01	177%	114	11
			11.08	18.50	0	166%	122	12
			9.72	18.58	0.01	191%	107	10
		Average	10.643	18.586	0.003	174.4%	117	10.8
		Per Process	2.1286	3.7172	0.0006	34.88%	23.4	2.16
	CPU Bound	20	43.4	77.32	0.5	179%	481	27
			39.45	73.95	0.01	187%	455	27
			39.48	73.91	0.02	187%	462	26
			39.61	74.38	0	187%	468	25
			39.54	74.22	0	187%	457	29
			39.49	74.06	0.01	187%	451	28
			39.5	74.02	0.01	187%	462	27
			39.35	73.96	0	187%	455	28
			39.53	74.23	0	187%	462	28
			39.59	74.28	0.02	187%	455	26

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
		Average	39.894	74.433	0.057	186.2%	460.8	27.1
		Per Process	1.9947	3.72165	0.00285	9.31%	23.04	1.355
		200	450.07	851.53	0.1	189%	5246	209
			389.75	740.30	0.03	189%	4589	206
			398.6	754.68	0.06	189%	4686	208
			410.02	779.63	0.1	190%	4733	209
			397.65	756.07	0.06	190%	4677	207
			392.36	744.62	0.06	189%	4634	206
			411.02	779.69	0.03	189%	4777	209
			410.23	778.83	0.16	189%	4828	210
			391.65	743.29	5	189%	4582	212
			397.01	752.61	0.04	189%	4641	205
		Average	404.836	768.125	0.564	189.2%	4739.3	208.1
		Per Process	2.02418	3.840625	0.00282	0.946%	23.6965	1.0405
		5	6.15	0	2.83	45%	5	12732
			0.75	0	0.36	47%	4	10027
			0.62	0	0.32	52%	2	10116
			0.92	0	0.33	36%	5	14372
			0.76	0	0.32	42%	0	10080
			0.62	0	0.3	48%	5	10005
			0.76	0	0.44	58%	3	11308
			0.67	0	0.36	54%	4	10015
			0.64	0	0.32	50%	4	10037
			0.74	0	0.4	55%	4	13340
		Average	1.263	0	0.598	48.7%	3.6	11203.2

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
SCHED_RR	I/O Bound	Per Process	0.2526	0	0.1196	9.74%	0.72	2240.64
		20	17.42	0.03	19.31	111%	32	55529
			1.9	0	1.34	70%	4	48394
			1.71	0	1.52	88%	5	46736
			1.75	0	1.2	68%	4	43155
			2.07	0	1.51	72%	6	41623
			1.77	0	1.2	68%	4	47223
			2.41	0	1.45	60%	5	39930
			2.36	0.01	1.35	57%	2	46561
			1.34	0	1.52	113%	5	48854
			1.92	0	1.37	71%	2	52511
		Average	3.465	0.004	3.177	77.8%	6.9	47051.6
		Per Process	0.17325	0.0002	0.15885	3.89%	0.345	2352.58
		200	55.64	0.03	44.84	80%	39	483101
			79.33	0.06	13.61	17%	4	559299
			39.03	0.02	14.42	37%	5	598658
			40.57	0.01	14.52	35%	6	532912
			18.79	0.01	15.48	82%	5	583563
			629.61	0.46	19.87	3%	13	644259
			44.73	0.03	13.31	29%	11	589432
			79.55	0.05	14.38	18%	19	526485
			45.36	0.01	17.12	37%	24	562350
			39.95	0	16.18	40%	17	559169
		Average	107.256	0.068	18.373	37.8%	14.3	563922.8

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
		Per Process	0.53628	0.00034	0.091865	0.189%	0.0715	2819.614
	Mixed	5	105.4	195.68	0.43	186%	1624	10116
			102.32	186.31	0.46	182%	1572	9877
			103.77	187.55	0.42	181%	1612	9952
			108.78	188.71	0.46	173%	885	14270
			104.53	186.8	0.45	179%	1608	10813
			112.97	185.89	0.45	164%	1362	11426
			105.24	193.27	0.35	183%	1699	13120
			108.69	195.75	0.6	180%	1551	13546
			104.73	187.43	0.4	179%	1639	14925
			99.93	187.42	0.4	187%	1615	14527
		Average	105.636	189.481	0.442	179.4%	1516.7	12257.2
		Per Process	21.1272	37.8962	0.0884	35.88%	303.34	2451.44
		20	427.28	805.29	1.82	188%	8083	40711
			403.13	761.57	1.75	189%	7208	47690
			409.51	775.3	1.94	189%	7684	59531
			417.86	787.84	1.87	188%	7781	41341
			411.07	773.36	1.9	188%	7632	40124
			431.08	780.64	2.06	181%	7457	60991
			419.43	790.72	1.79	188%	7732	50021
			426.61	788.12	1.59	185%	7714	40190
			398.21	54.82	1.83	190%	7286	53737
			407.13	765.24	1.87	188%	7677	53602
		Average	415.131	708.29	1.842	187.4%	7625.4	48793.8

SCHEDULING	Program Type	Number of Processes	Elapsed Real Time (seconds)	CPU Seconds As User	CPU Seconds As Supervisor	Percentage of CPU used by this Job	Involuntary Switches	Voluntary Switches
		Per Process	20.75655	35.4145	0.0921	9.37%	381.27	2439.69
		200	4465.02	8411.64	15.54	188%	87778	480246
			4240.26	8044.6	17	190%	81500	450353
			4267.45	8079.38	15.15	189%	82042	462258
			4327.13	8166.66	15.34	189%	82597	477147
			4276.55	8115.11	15.49	190%	84020	382080
			4203.33	7976.64	16.08	190%	82879	384907
			4252.17	8047.96	14.91	189%	81895	493591
			4258.83	8069.09	29.07	190%	83238	495707
			4242.16	8049.5	16.52	190%	81573	379054
			4235.56	8031.29	16.22	189%	82423	459482
		Average	4276.846	8099.187	17.132	189.4%	82994.5	446482.5
		Per Process	21.38423	404.95935	0.8566	9.47%	4149.725	22324.125

Appendix B

testscript

The file testscript begins by cleaning and then making all of the code necessary for running the tests. Then it runs all 9 combinations of CPU tests. Then it runs all 9 combinations of I/O tests. Then it runs all 9 combinations of Mixed tests. Each test uses the linux time command to monitor how the scheduling policies work. The results of the time command are written to three different files: cpu_results for the CPU tests, io_reslts for the I/O tests, and mixed_results for the mixed tests.

Makefile

The Makefile creates an input file for the rw.c and mixed.c files to read from. It also builds pi.c, rw.c, and mixed.c, along with some other helpful files. It contains a clean command that removes all of the test output files, executables, object files, temp files, and other log files.

pi.c

The pi.c file takes two arguments: the scheduling policy and the number of processes to run. If neither of these arguments are supplied the program quits. It then forks the specified number of processes. Each child process then calculates pi through 100,000,000 iterations then returns. The parent waits on all children to complete.

rw.c

The rw.c file takes six arguments: the scheduling policy, number of processes to run, the total amount of Bytes to transfer, the number of Bytes to transfer per write, the input filename, and the output filename. The program then forks the specified number of processes. Each child process reads from the input file then writes to an output file until all Bytes have been transferred. The parent waits on all children to complete.

mixed.c

The mixed.c file takes six arguments: the scheduling policy, number of processes to run, the total amount of Bytes to transfer, the number of Bytes to transfer per write, the input filename, and the output filename The program then forks the specified number of processes. Each child alternates in calculating pi through 100,000,000 iterations and reading and writing data. They alternate ten times. The parent waits on all children.