Kevin Rau

101289616

Han - Rahat

PA\_4

Linux Scheduling Analysis

**Abstract**

Through testing and analysis of three scheduling policies: CFS, Round Robin, and FIFO on CPU, I/O and Mixed within the Linux Scheduler, it was found that when processes increase in magnitude, SCHED\_OTHER was the best to schedule for any task with low Wall time. While Round Robin faired well in the small magnitudes of processes, it fell short with one case on 100 processes with Mixed. Round Robin showed to have fewer context switches than CFS, but was slower. I/O was the fastest running process, only taking less than seconds with CPU and Mixed taking the longest.

**Introduction**

The work done to draw conclusions and build benchmarking was done on 3 schedulers, 3 process types, and 3 magnitudes of processes running simultaneously, ending with 27 cases. Being able to scale separate schedulers to different processes types with three different priority scales made it possible to see which scheduler was the best choice to run on many processes. After running on all 27 cases, data was saved as a .csv file and used to create graphical representations to help draw conclusions.

**Methods**

The benchmarks created gather data for Wall time, User Time, System Time, CPU, I-switch and V-switch between the 27 test cases. To test CPU processing, the pi-sched.c was used to compute pi a numerous amount of times and spawned children accordingly. To test I/O, rw.c was to read and write to a file then used and made to spawn children as well. To test both styles of processing, mixed.c was made out of pi-sched.c and coded to write to a file rather than print out to the terminal. The testscript file was added to do the 3 schedulers and 3 types of processing, running this 3 times gave the 27 test cases. Additional runs were performed to gather more data for more accurate results and written out to a .csv file.

**Results**

CPU 10 Processes:

I/O 10 Processes:

Mixed 10 Processes:

CPU 100 Processes:

I/O 100 Processes:

Mixed 100 Processes:

CPU 1000 Processes:

I/O 1000 Processes:

Mixed 1000 Processes;

**Analysis**

The expectations that from the results showed that SCHED\_OTHER averagely started to perform better when the number of processes increases. While taking up more CPU and having a heavy amount of context switches, it was able to get the job done more efficiently with multitasking. Where SCHED\_OTHER really started to work exceptionally well is with the “mixed.c” processes, where on average if completed in less time and had less CPU % usage.

SCHED\_FIFO seemed to be in between SCHED\_OTHER and SCHED\_RR with its use of CPU, but seeming to pull benchmarks that came close to SCHED\_RR. While not it fell short with “mixed.c” it performed well with CPU.

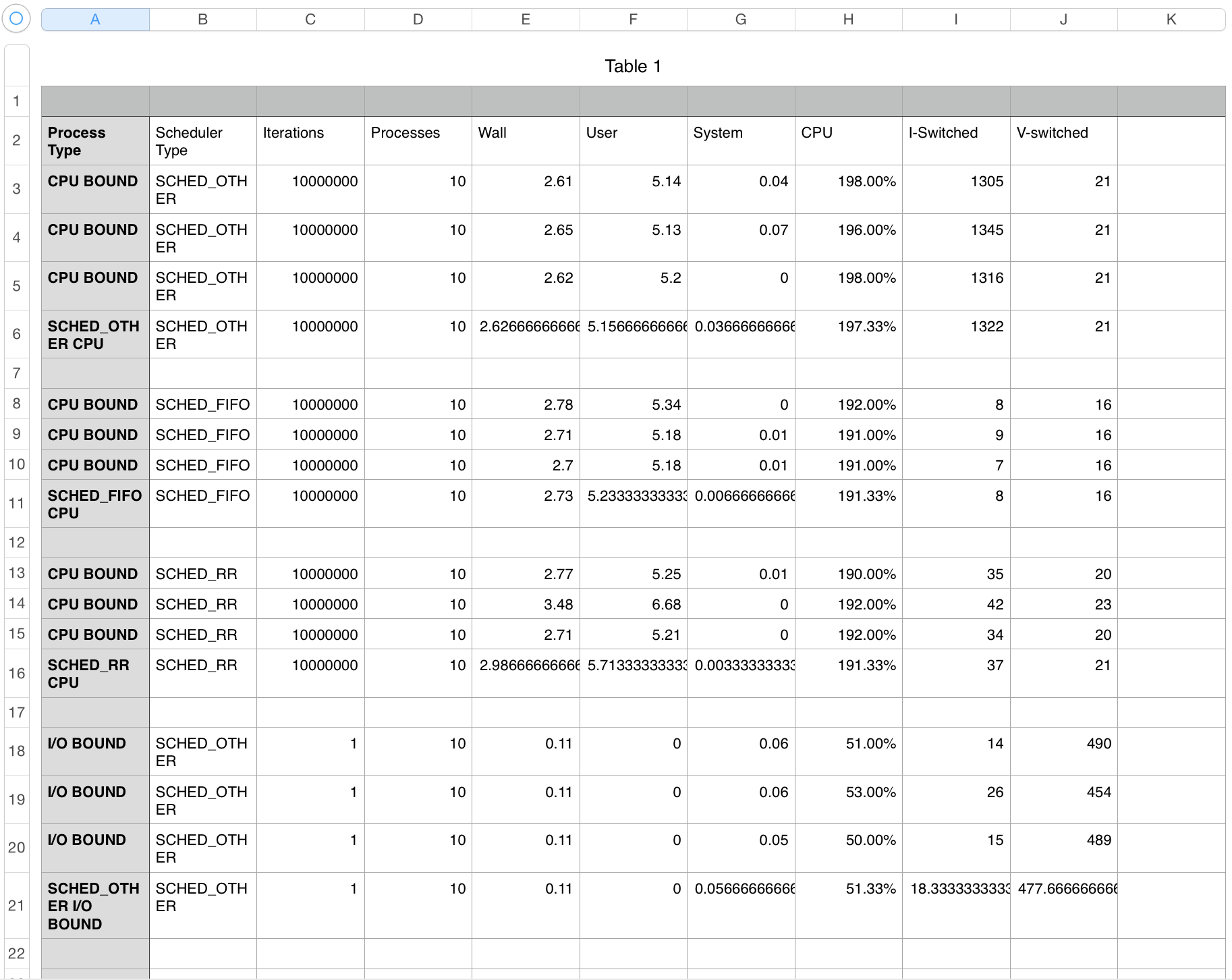
SCHED\_RR performed very well with low processes, but struggled with more processes. The time for Wall increased with “mixed.c” as well as CPU usage with I/O. SCHED\_RR did perform the best with CPU on 1000 processes.

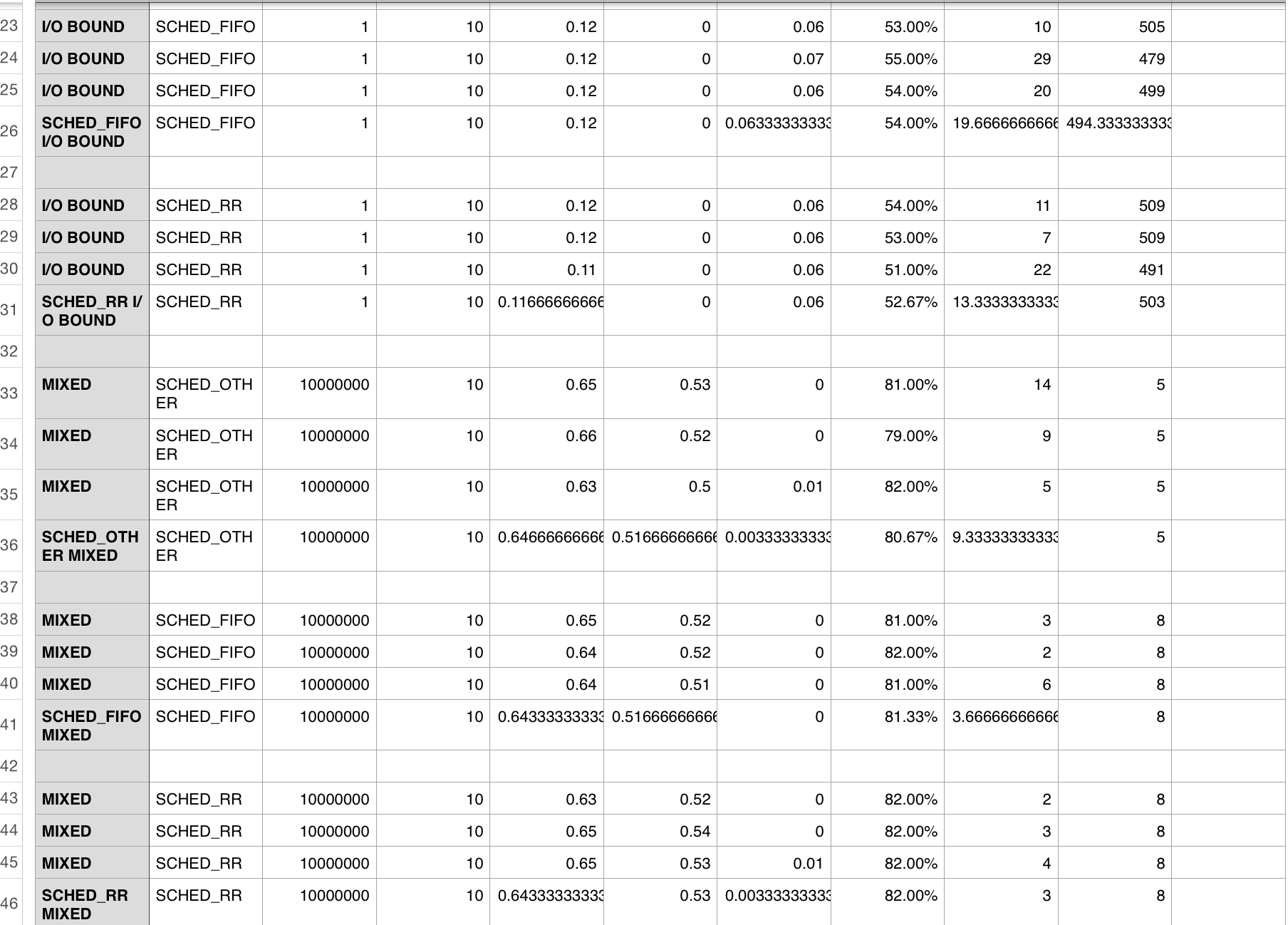
SCHED\_FIFO and SCHED\_RR averagely had the best CPU for all test cases, some though SCHED\_OTHER had better when it came to I/O.

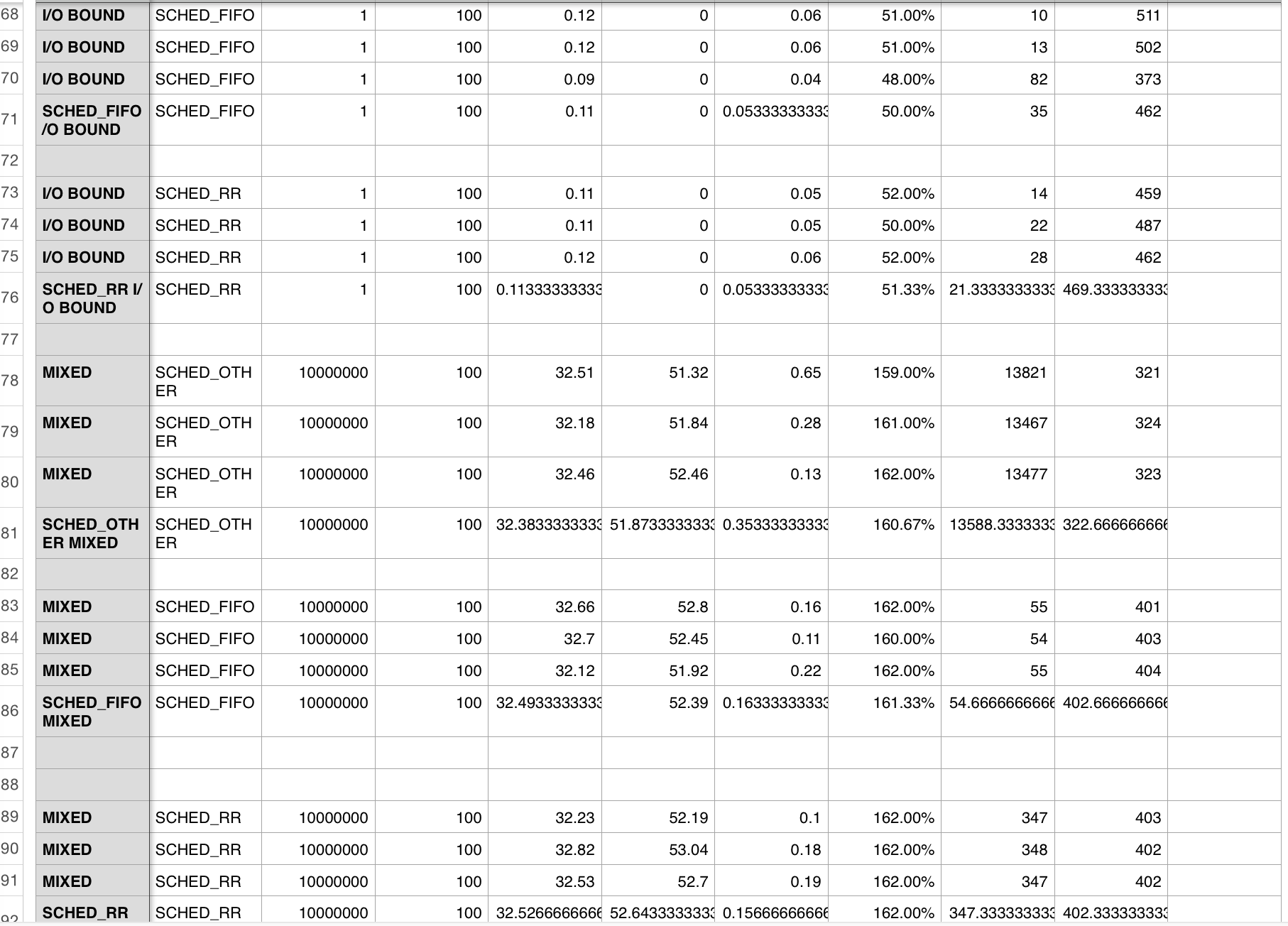
Overall, when it came to multitasking, SCHED\_OTHER was the best, Individual tasks SCHED\_RR came second, and SCHED\_FIFO in last.

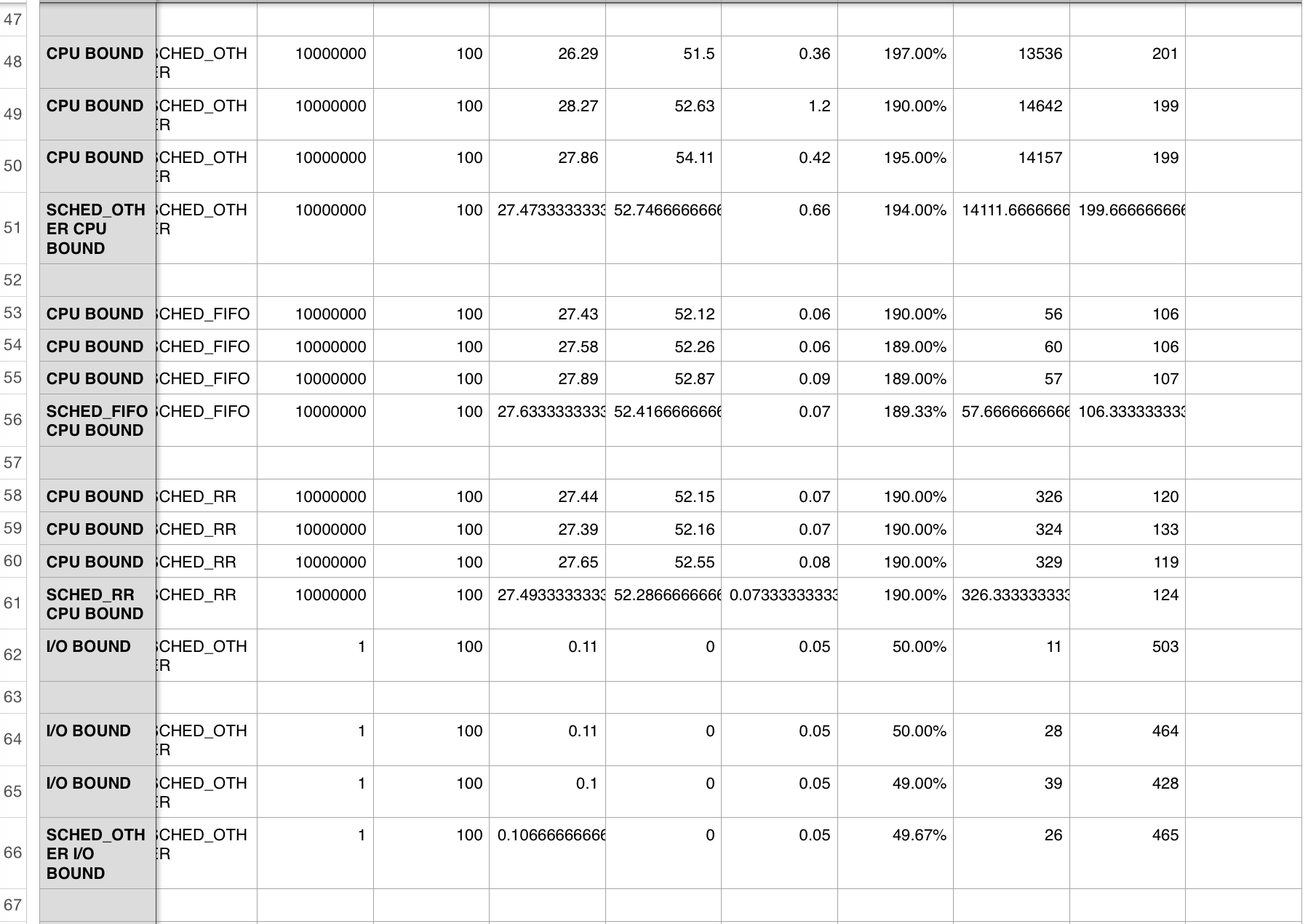
**Conclusion**

While SCHED\_RR and SCHED\_FIFO did perform very well with individual tasks on the CPU and I/O side, SCHED\_OTHER did end up being able to quickly multitask the multi jobs more efficiently and faster. SCHED\_OTHER did take up a lot of CPU usage and had many context switches, but it was able to get the job done still where crashes were possible with SCHED\_RR and SCHED\_FIFO. SINCE SCHED\_OTHER is CFS, as processes continue in number, the native Linux scheduler used makes sense because it is able to handle the multitasking and many processes that are thrown at it.

**APPENDIX A.**

****



****

