Task 2: Experiment

To determine how many parallel processes should be used to perform our parallel_convert program in the least amount of time we will monitor the Real, User, and System process times using the bash command: *time*. First we made sure there were 40 images in our input directory to maintain consistency across tests. We run our time command as follows:

/usr/bin/time -a -o time.txt -f "<parallel processes>;%E; %U; %S;" ./parallel_convert <parallel processes> output_dir input_dir >/dev/null

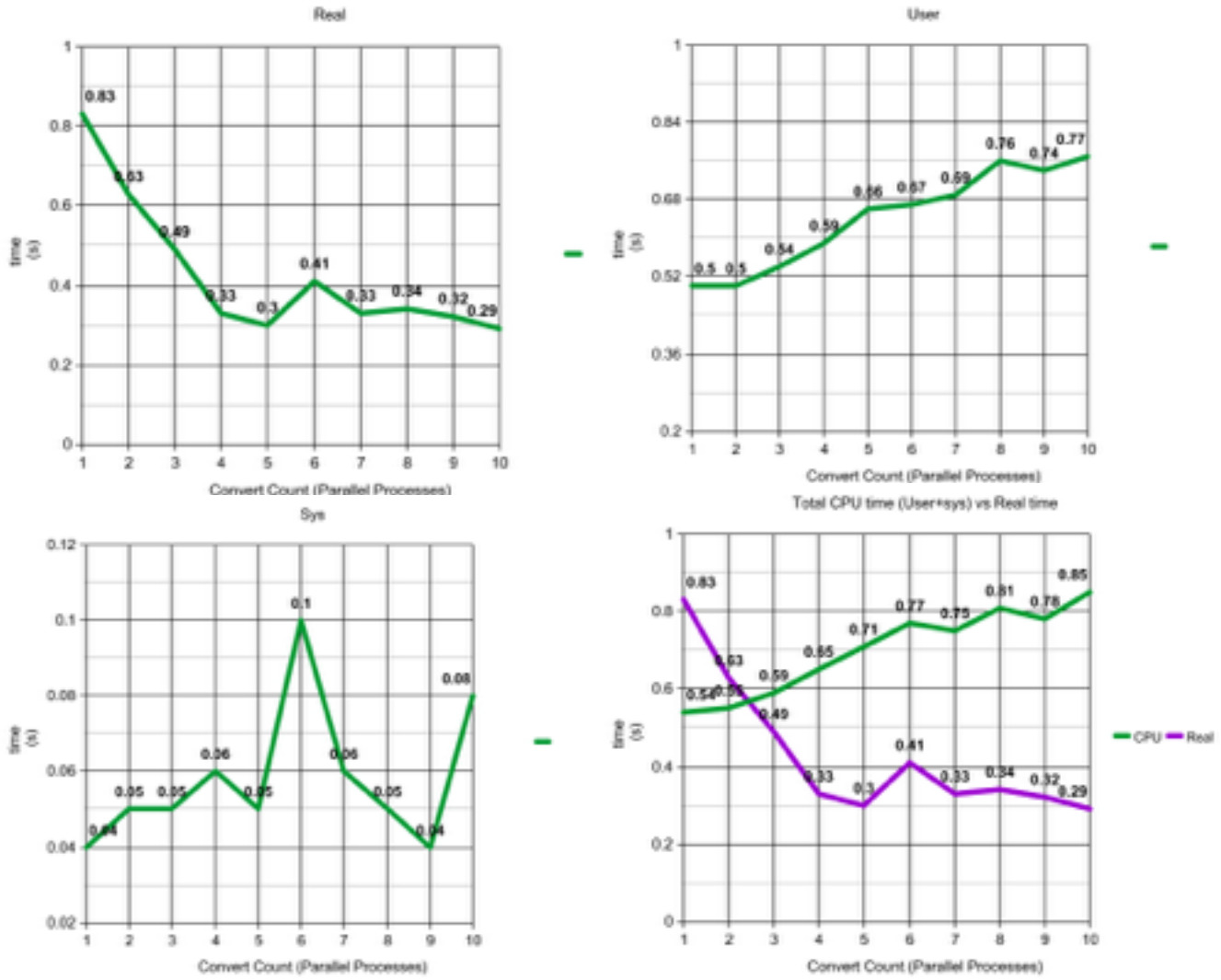
The -f option allows us to to set the format of the output. It is of the format <parallel processes>*;%E; %U; %S.* %E is the Real time, %U is the User time and %S is the Sys time. The -o option will output the timing analysis to a file time.txt. We then suppress stdout using >/dev/null so printing to stdout does not affect our timing analysis. This command is run with parallel processes 1 through 10 to get timing analysis for different amounts of parallel converting. After outputing 10 timing analyses we run:

(echo "Parallel Processes;Real;User;Sys" ; cat time.txt) | sed 's/;/\t/g' > time.csv

This takes the information from our time.txt file of the form "*<parallel processes>;%E; %U; %S;"* and places them in columns labeled Parallel Processes, Real, User and Sys in a .csv file, time.csv. This gives us our table of timing data.

Parallel Processes	Real	User	Sys
1	0:00.83	0.5	0.04
2	0:00.63	0.5	0.05
3	0:00.49	0.54	0.05
4	0:00.33	0.59	0.06
5	0:00.30	0.66	0.05
6	0:00.41	0.67	0.1
7	0:00.33	0.69	0.06
8	0:00.34	0.76	0.05
9	0:00.32	0.74	0.04
10	0:00.29	0.77	0.08

From the table we can create graphs of process times vs number of parallel converts.



The Real time is the physical, real-world “wall-clock” time that the program takes to complete. This time is a reference for the time the program takes from start to finish including any time processes may need to wait during the process execution. This is the most obvious measure of time being improved while we increase the number of parallel processes. When looking at the minimum amount of time to complete our image conversion we would want the most parallel conversion processes (10).

The User and Sys times are the time the CPU spends executing processes. User time represents time the CPU spends in user-mode. The Sys time represents time the CPU spends in Kernel mode. Most of the process time is spent in User mode as kernel mode is for the most low-level hardware and memory access. The combined User+Sys time represents the total CPU time of our program. As we can see our Real time is actually lower than our combined CPU time after 2 parallel converts. This is due to our parallel converts still taking up CPU time but due to multithreading we can have multiple processes accessing CPU resources. These still individually count towards active CPU time. As we want a PID divisible by 2 or 3 we may need to create more processes if we are using more parallel processes as the possible PIDs have more variance. This results in an increasing CPU time and a not as linear decrease in Real time as we increase the number of parallel processes.