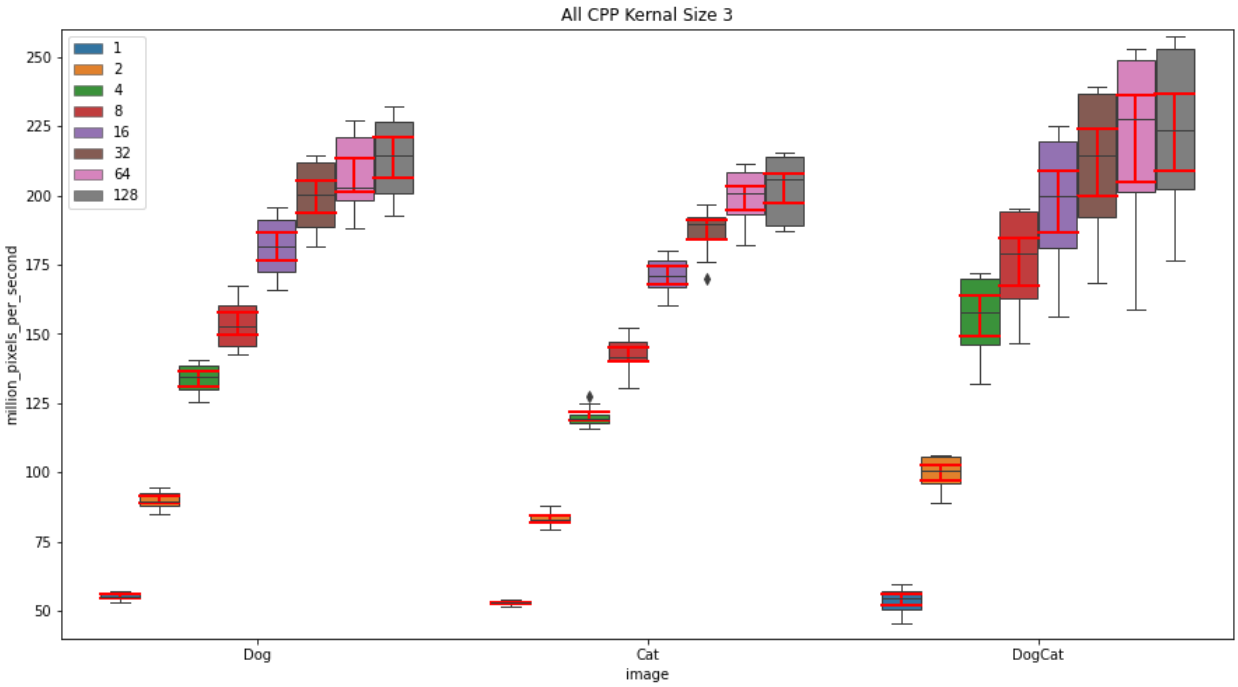
Comparison of Runtimes

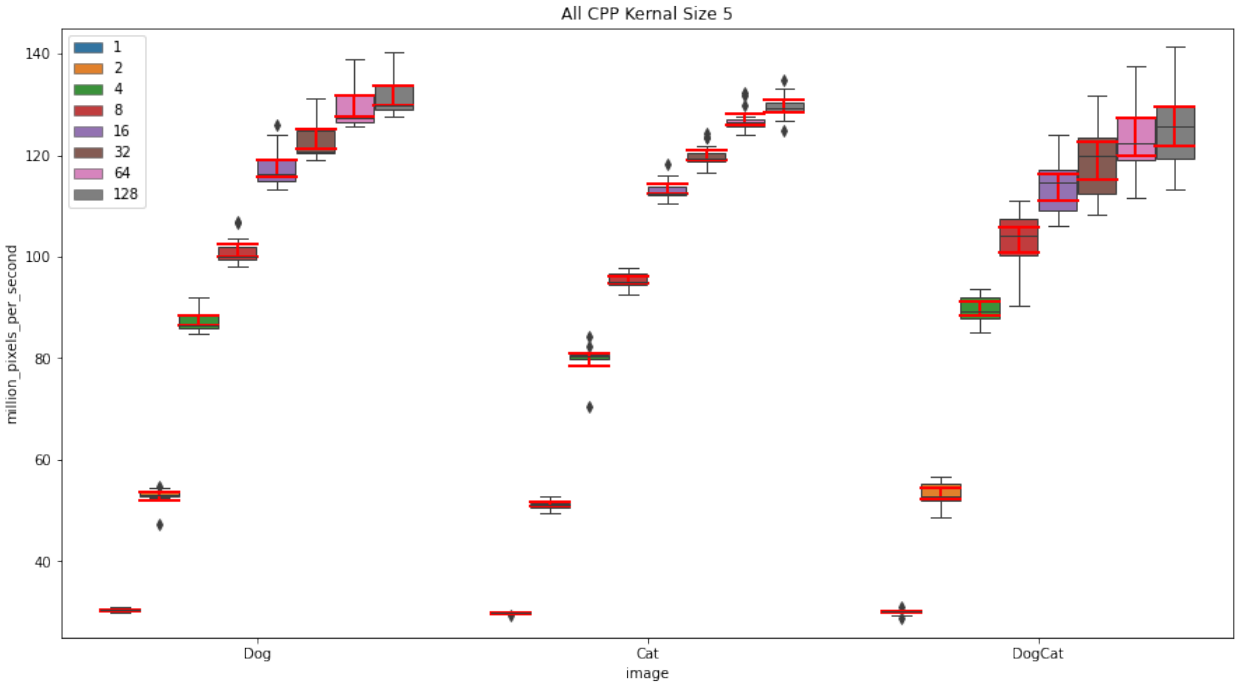
Experiment 1 What is the correct number of threads

For the following charts a decorated boxplot is used. The shaded region with a line in represent the median and the inner two quartiles. The whiskers attached to the plot represent the outer two quartiles and the ride lines represent a confidence interval of 95%. Any outliers outside of the 2 outer quartiles will be marked as points.

Native Code

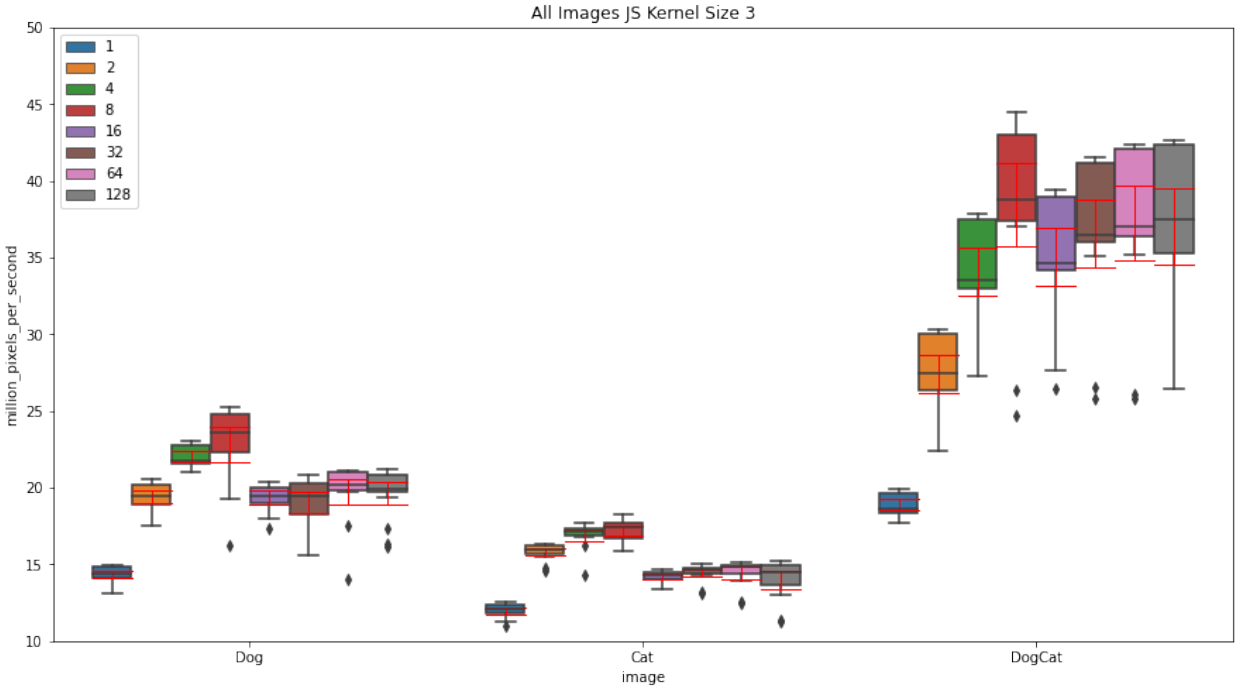
The first step is to find the correct number of threads for execution.

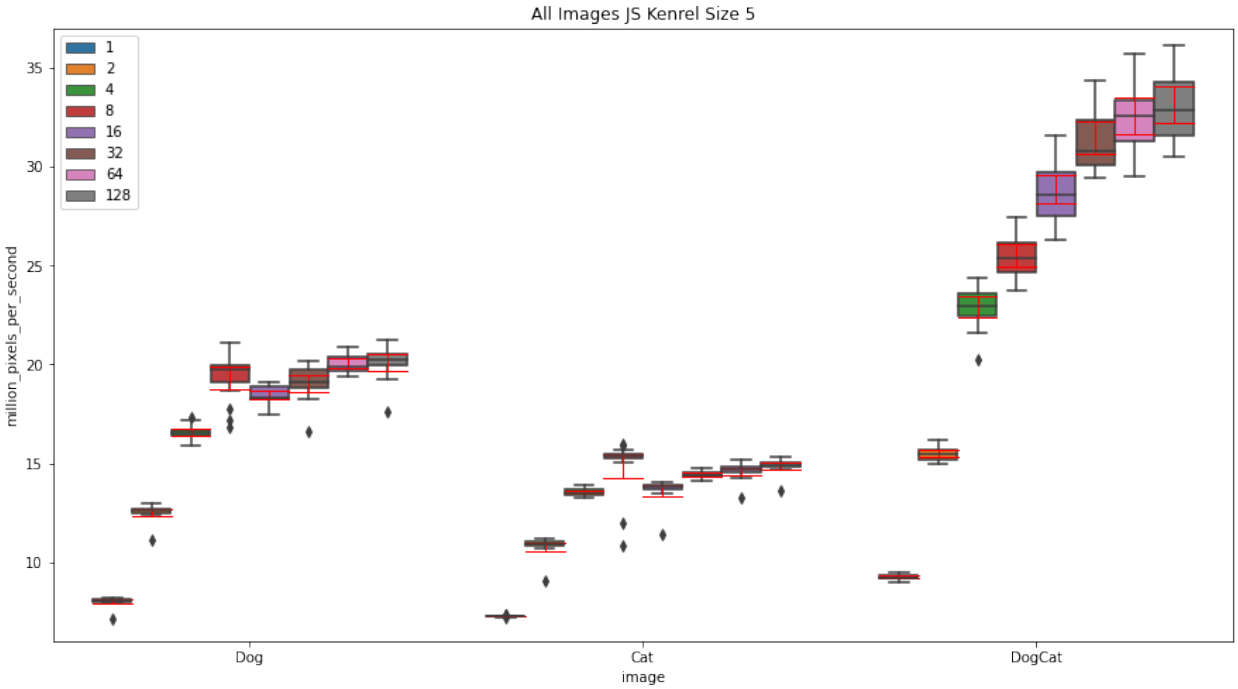




With the native code increasing the threads seems to increase the performance all the way to 128. This would likely not be the case on a system that had full access to the processor. The machine this program ran on had 8 cores, so the increase in performance likely came from the operating system choosing the program's threads with a higher likelihood since there is more of them. There is a clear benefit in threading the program. For later sections 128 threads will be used to compare to the single threaded version.

JavaScript Code





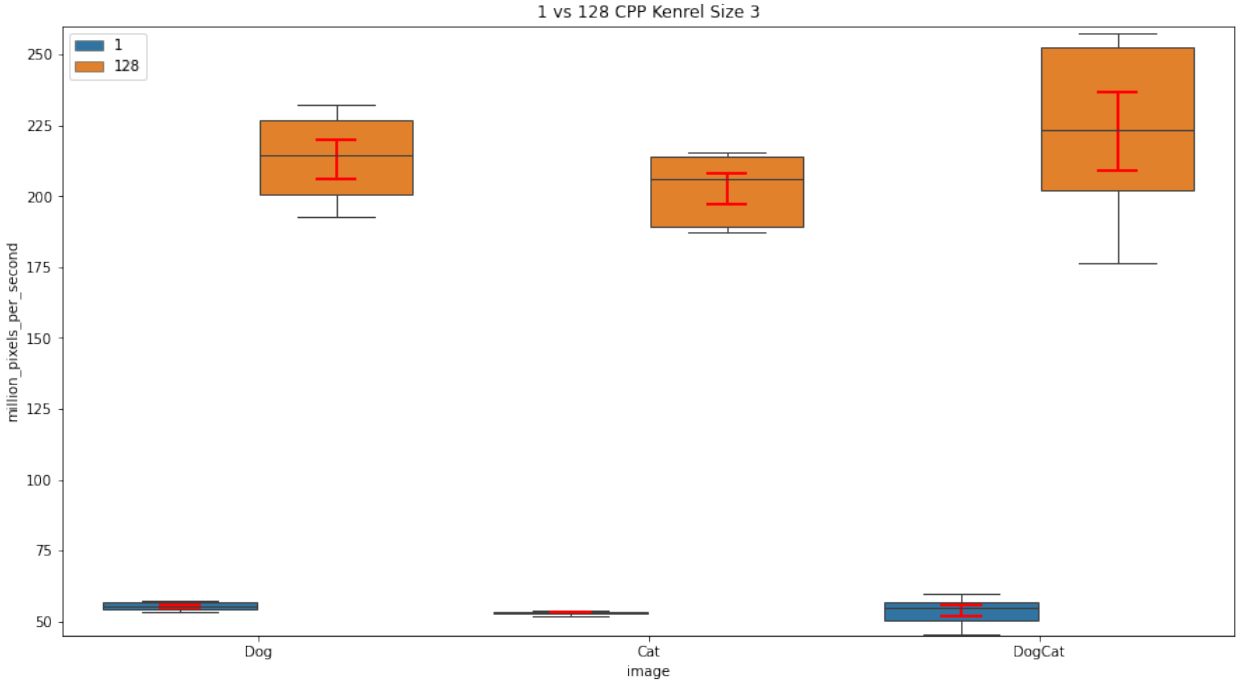
The story around JS improvement is a little different. For the smaller two images 8 threads seems to be the best option. For the largest image There is an increase in performance for kernel size 5, and there is an unclear benefit for kernel size 3. That being considering the image that usually gets streamed. There is a lot of reasons that could result in the drop in performance past 8 threads, but it is expected that native C++ code would handle threads more efficiently than its

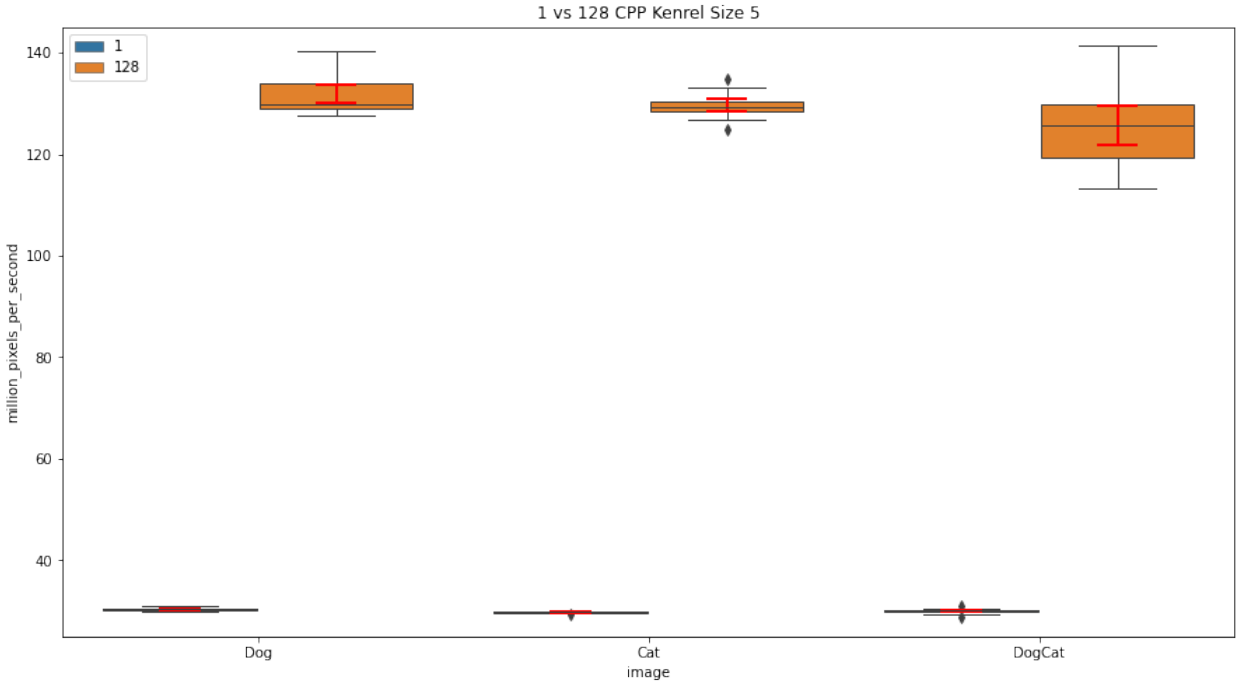
transpiled counterpart. For analysis purposes 8 threads will be compared to the single threaded version in the next sections.

Experiment 2 Best of threads versus 1 Thread

Native Code

For native code 1 thread is going to be compared to its 128 thread counterpart.

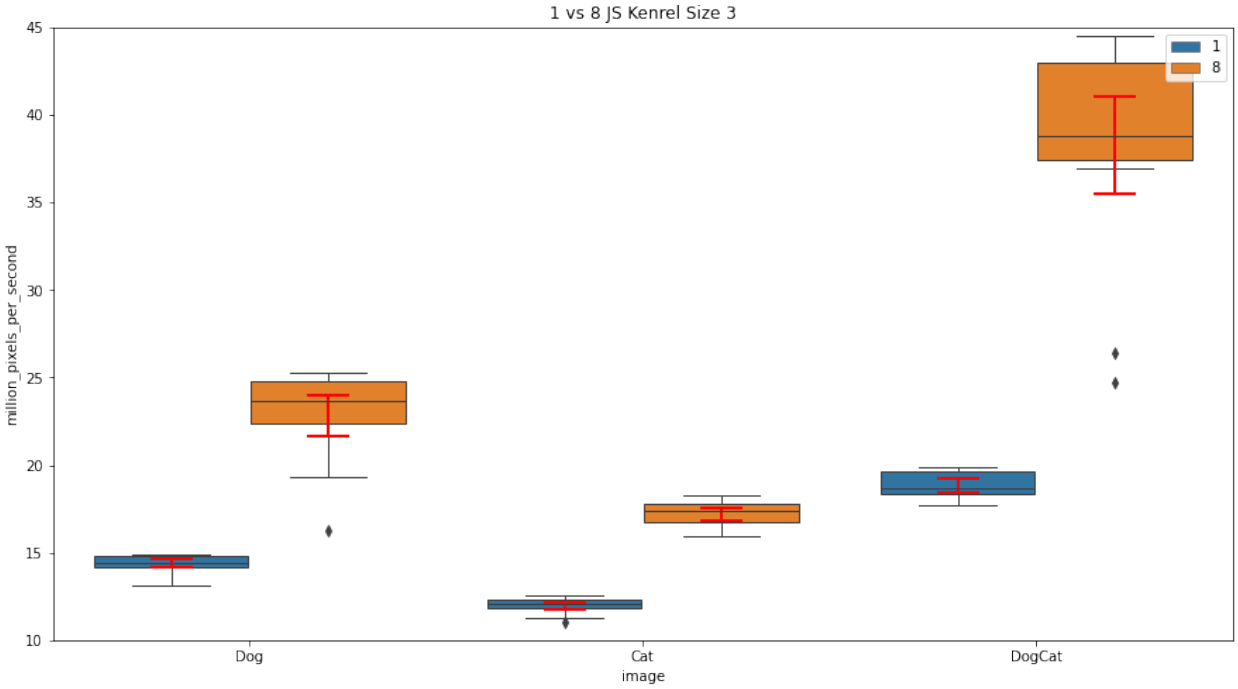


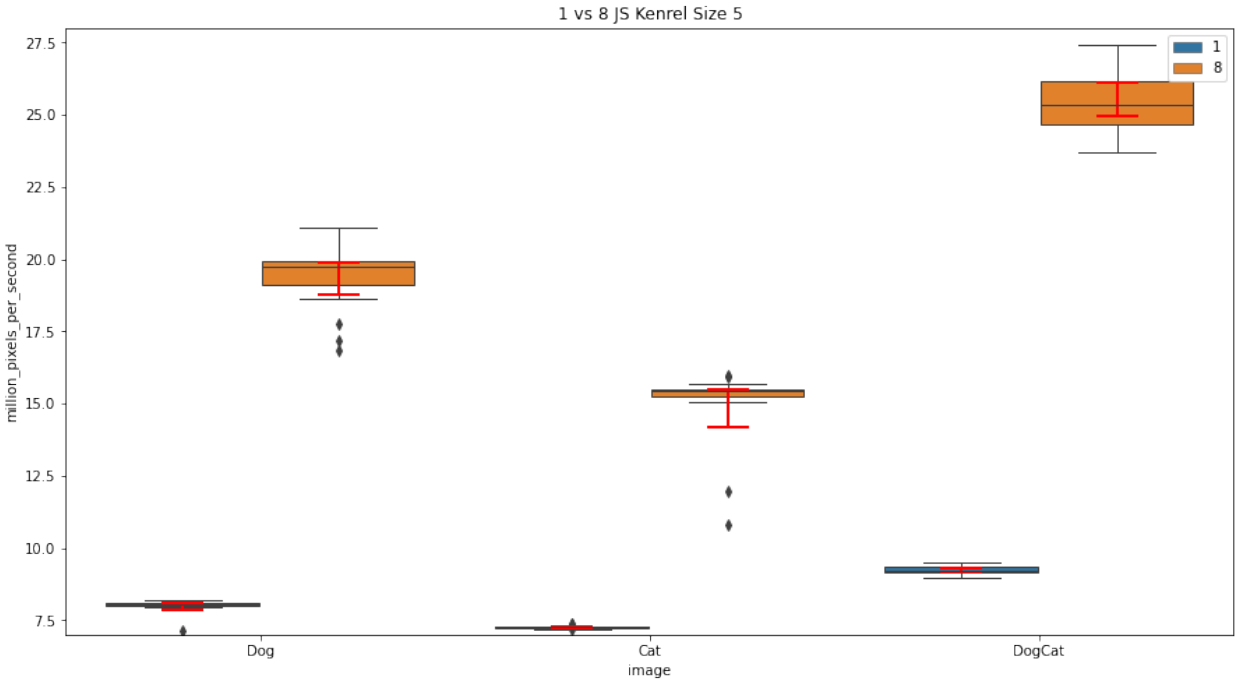


These charts speak for themselves when inspecting the confidence intervals. While 128 threads might be an overkill number of threads, but the improvement is clear.

JavaScript Code

When comparing the single threaded version to 8 threads is significantly fast as expected.





In a very similar fashion, the confidence intervals in the above charts clearly show the improvements available for threading the application. Given this improvement 8 threads give a clear benefit and will be used in the next analysis.

Frame Rate Analysis in JavaScript

Before starting the frame rate analysis, it is appropriate to apply a T-test to the means in question to ensure there is a statistical difference between them.

Image	Kernel Size	Execution	Threads	Mean Million pixels /sec	Confidence
Dog	3	CPP	1	55.41	
Dog	3	CPP	128	213.5	1.913×10^{-28}
Cat	3	CPP	1	52.96	
Cat	3	CPP	128	202.8	4.40×10^{-31}
DogCat	3	CPP	1	54.02	
DogCat	3	CPP	128	223.0	8.102×10^{-20}
Dog	5	CPP	1	30.42	
Dog	5	CPP	128	131.8	8.916×10^{-39}
Cat	5	CPP	1	29.80	
Cat	5	CPP	128	129.6	2.508×10^{-45}
DogCat	5	CPP	1	30.02	
DogCat	5	CPP	128	125.5	2.337×10^{-29}
Dog	3	JS	1	14.39	
Dog	3	JS	8	22.92	2.021×10^{-14}
Cat	3	JS	1	12.00	
Cat	3	JS	8	17.18	1.092×10^{-21}
DogCat	3	JS	1	18.87	
DogCat	3	JS	8	38.56	2.511×10^{-14}
Dog	5	JS	1	8.023	
Dog	5	JS	8	19.37	1.721×10^{-26}
Cat	5	JS	1	7.269	
Cat	5	JS	8	14.96	8.490×10^{-20}
DogCat	5	JS	1	9.246	
DogCat	5	JS	8	25.50	3.958×10^{-31}

After applying a ttest to all of the pairing of single threaded vs. multithreaded there is a clear statistical. The following section will contain theoretical frame rates based on these speeds.

Dog image in CPP 1920x1080	
Kernel Size 3 – 1 thread $2.07\ mill\ pixels * \left(\frac{1\ sec}{54.41\ mill\ pixels}\right)$ $= 0.0380$ sec *or* 26.29 *Hz*	**Kernel Size 3 – 128 thread** $2.07\ mill\ pixels * \left(\frac{1\ sec}{213.5\ mill\ pixels}\right)$ $= 0.009696$ sec *or* 103.1 *Hz*
Kernel Size 5 – 1 thread $2.07\ mill\ pixels * \left(\frac{1\ sec}{30.42\ mill\ pixels}\right)$ $= 0.06804$ sec *or* 14.70 *Hz*	**Kernel Size 5 – 128 thread** $2.07\ mill\ pixels * \left(\frac{1\ sec}{131.8\ mill\ pixels}\right)$ $= 0.0380$ sec *or* 63.67 *Hz*

Cat image in CPP 1344x837	
Kernel Size 3 – 1 thread $1.124\ mill\ pixels * \left(\frac{1\ sec}{52.96\ mill\ pixels}\right)$ $= 0.02122$ sec *or* 47.12 *Hz*	**Kernel Size 3 – 128 threads** $1.124\ mill\ pixels * \left(\frac{1\ sec}{202.8\ mill\ pixels}\right)$ $= 0.005542$ sec *or* 180.4*Hz*
Kernel Size 5 – 1 thread $1.124\ mill\ pixels * \left(\frac{1\ sec}{29.80\ mill\ pixels}\right)$ $= 0.03772$ sec *or* 26.51 *Hz*	**Kernel Size 5 – 128 thread** $1.124\ mill\ pixels * \left(\frac{1\ sec}{129.6mill\ pixels}\right)$ $= 0.008673$ sec *or* 115.3 *Hz*

DogCat image in CPP 4040x2693	
Kernel Size 3 – 1 thread $10.8\ mill\ pixels * \left(\frac{1\ sec}{54.02\ mill\ pixels}\right)$ $= 0.1999$ sec *or* 5.002 *Hz*	**Kernel Size 3 – 128 threads** $10.8\ mill\ pixels * \left(\frac{1\ sec}{223.0\ mill\ pixels}\right)$ $= 0.04843$ sec *or* 20.65 *Hz*
Kernel Size 5 – 1 thread $10.8\ mill\ pixels * \left(\frac{1\ sec}{30.02\ mill\ pixels}\right)$ $= 0.3598$ sec *or* 2.780 *Hz*	**Kernel Size 5 – 128 thread** $10.8\ mill\ pixels * \left(\frac{1\ sec}{125.5\ mill\ pixels}\right)$ $= 0.0861$ sec *or* 11.62 *Hz*

With all of these calculations we can see a clear frame rate increase. If the goal was to have an application stream the filters in real time for the 1080p image (dog.png) thee frame rate is well above what would be needed. The large image still doesn't have a good frame rate, but 11.62 Hz

is a whole lot more reasonable for a user than 2.780. The next section will compare JavaScript application; 1 vs. 8 threads.

Dog image in JS 1920x1080	
Kernel Size 3 – 1 thread $2.07\ mill\ pixels * \left(\frac{1\ sec}{14.39\ mill\ pixels}\right)$ $= 0.1438\ \text{sec}\ or\ 6.952\ Hz$	**Kernel Size 3 – 8 threads** $2.07\ mill\ pixels * \left(\frac{1\ sec}{22.92\ mill\ pixels}\right)$ $= 0.09031\ \text{sec}\ or\ 11.07\ Hz$
Kernel Size 5 – 1 thread $2.07\ mill\ pixels * \left(\frac{1\ sec}{8.023\ mill\ pixels}\right)$ $= 0.2580\ \text{sec}\ or\ 3.876\ Hz$	**Kernel Size 5 – 8 thread** $2.07\ mill\ pixels * \left(\frac{1\ sec}{19.37\ mill\ pixels}\right)$ $= 0.1069\ \text{sec}\ or\ 9.357\ Hz$

Cat image in JS 1344x837 1344x837	
Kernel Size 3 – 1 thread $1.124\ mill\ pixels * \left(\frac{1\ sec}{12.00\ mill\ pixels}\right)$ $= 0.09367\ \text{sec}\ or\ 10.68\ Hz$	**Kernel Size 3 – 8 threads** $1.124\ mill\ pixels * \left(\frac{1\ sec}{17.18\ mill\ pixels}\right)$ $= 0.06542\ \text{sec}\ or\ 15.28\ Hz$
Kernel Size 5 – 1 thread $1.124\ mill\ pixels * \left(\frac{1\ sec}{7.264\ mill\ pixels}\right)$ $= 0.1547\ \text{sec}\ or\ 6.463\ Hz$	**Kernel Size 5 – 8 thread** $1.124\ mill\ pixels * \left(\frac{1\ sec}{14.96\ mill\ pixels}\right)$ $= 0.07513\ \text{sec}\ or\ 13.31\ Hz$

DogCat image in JS 4040x2693	
$10.8\ mill\ pixels * \left(\frac{1\ sec}{18.87\ mill\ pixels}\right)$ $= 0.5723\ \text{sec}\ or\ 1.747\ Hz$	**Kernel Size 3 – 8 threads** $10.8\ mill\ pixels * \left(\frac{1\ sec}{38.56\ mill\ pixels}\right)$ $= 0.2801\ \text{sec}\ or\ 3.570\ Hz$
Kernel Size 5 – 1 thread $10.8\ mill\ pixels * \left(\frac{1\ sec}{9.240\ mill\ pixels}\right)$ $= 1.168\ \text{sec}\ or\ 0.8556\ Hz$	**Kernel Size 5 – 8 thread** $10.8\ mill\ pixels * \left(\frac{1\ sec}{25.50\ mill\ pixels}\right)$ $= 0.4235\ \text{sec}\ or\ 2.361\ Hz$

The increase in performance is clear in JavaScript as well. From a user's perspective an increase from 6.463 Hz to 13.31 Hz would be dramatic. Threading this application clearly increases the resolution that would be possible to be used.