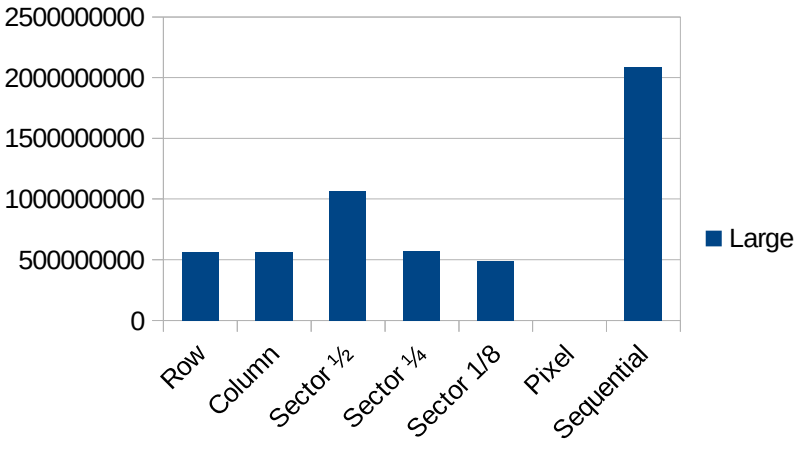
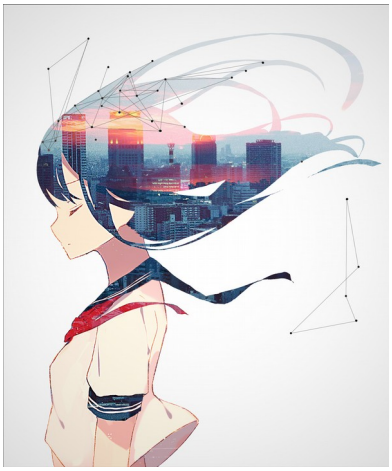
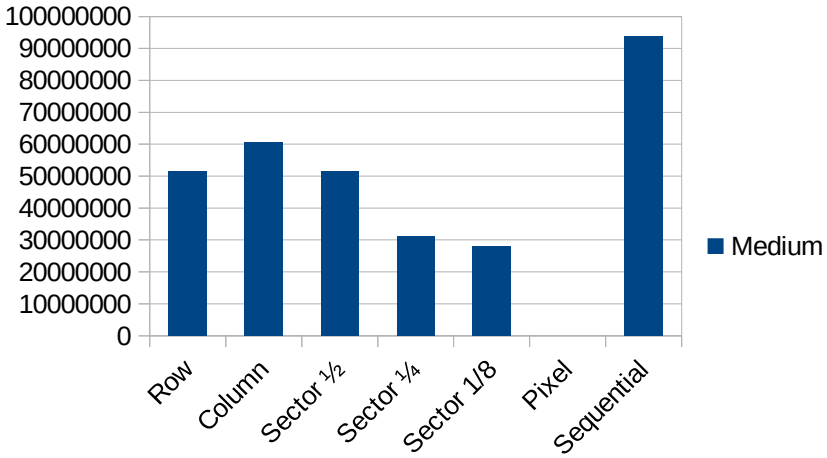
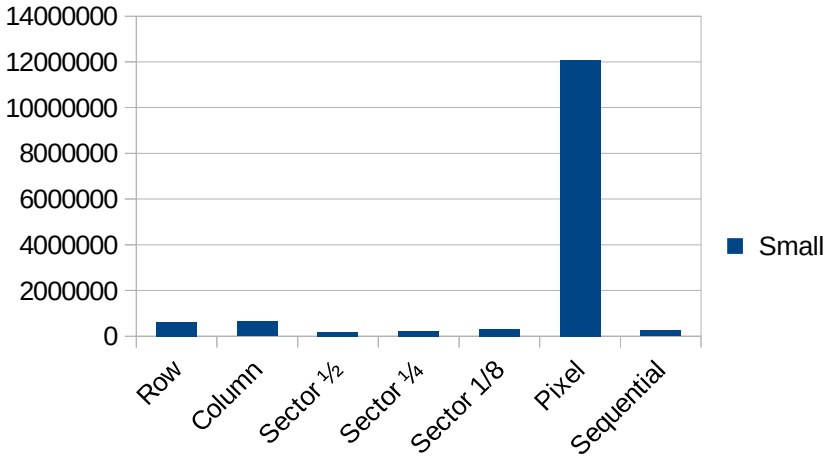
Experiment results:

Picture size	Row	Sector 1/2	Sector 1/4	Sector 1/8	Column	Sequential	Pixel
1913x2297	563649483	1049333116	567167572	491887228	557730766	2114107105	invalid
	565505279	1042036787	578963603	486864241	574428667	2008650981	
	561238184	1079642786	565546062	485812978	549768118	2007797404	
	557156073	1077395602	558126298	497133125	572603751	2232350809	
average:	561887225	1062102072.75	567450883.75	490424393	563632825.5	2090726574.75	
20x20	664575	162720	251036	299247	629857	215071	12641468
	607261	219546	233303	305838	600345	236630	12701041
	594748	166603	207862	322040	647832	263390	11198800
	600362	231234	202934	315193	684962	270187	11814574
average:	616736.5	195025.75	223783.75	310579.5	640749	246319.5	12088970.75
640x384	47343955	50241727	31687128	27668052	60093100	88013612	invalid
	51574538	52397931	31073844	27823769	58891815	88913428	
	53118998	50622586	30946525	27182085	63083823	99333304	
	53304542	52273732	30713719	29834078	60407771	99117273	
average:	51335508.25	51383994	31105304	28126996	60619127.25	93844404.25	









Firstly, Sectors works well for all sizes of pictures. There is a significant amount of time reduced comparing to sequential processing. Also, dividing the task by 4 is 1/3 faster than dividing the task by 2, but there is no significant difference between processing in 4 sectors and 8 sectors.

Secondly, processing pixel by pixel is extremely slow as it creates too many threads, and it only works on small pictures as we are not allowed to create millions of threads at the same time.

Thirdly, if a picture have width lager than height (e.g the medium sized picture), then the time to process picture column by column will be larger than processing row by row, vice versa. However this effect is not very significant on very large pictures as each thread will have more work to do, so the amount of time used to create thread is insignificant.