Fig 1. The result of nvprof.

When the size is greater than 100,000, the GPU version begin to be noticeably better than CPU version. At 1,000,000 and 10,000,000 N, the GPU version is much better, but there is no difference between 1,000,000 and 10,000,000.

At N = 10, 100, 1000 the cudaMalloc occupied over 90% time. At N = 10,000 and 100,000, it decreased to 67% and 24%, respectively. At N >= 1,000,000, it occupied less than 1%.

At N = 10 and 100, cudaLaunch occupied less 1% of time. At N = 10,000 and 100,000, cudaLaunch occupied 28% and 24%, respectively. When N is >= 1,000,000, the cudaLaunch occupied more than 95%.

As shown in Fig 1, when N is >= 1,000,000, GPU version is best(GPU time / CPU time is greater than 5), cudaMalloc occupied less than 1% and cudaLaunch occupied more than 95%.

When the GPU version is not better than CPU(N <= 100,000), cudaMalloc occupied more than 24% of time. Other than that, the % of cudaMalloc time is less than 1%. If we get the advantage of GPU for this algorithm, the sample size should be over 1,000,000. If not, CPU is better than GPU. The cudaMalloc is memory overhead. The size N <= 100,000 is too small to hide device memory allocation. The prolonged time global memory access in gpu has negative effect on performance. The gpu can work after global memory access.