Programming Assignment 1

Han Chen

011469432

**Task 1: Draw Memory Mountain**

1.Using function

store\_double\_data\_to\_csv("double\_data.csv", ROW, COLUMN); /\* Store double data into a csv file \*/  
store\_float\_data\_to\_csv("float\_data.csv", ROW, COLUMN); /\* Store double data into a csv file \*/

to randomly generate double and float matrix.

2.Using function

read\_double\_data\_from\_csv("double\_data.csv"); /\* Load data from a csv file \*/

to load the data from csv to the data matrix, and then get the benchmark result and memory mountain graph.

**Benchmark result for double matrix:**

Clock frequency is approx. 2597.8 MHz

Memory mountain (MB/sec)

Max Elements: 16785409

s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53 s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64

32m 3015.0 2674.1 2628.5 2516.7 2547.7 2185.2 2138.4 2000.2 1769.3 1596.7 1447.1 1326.4 1219.0 1171.9 1115.3 1060.6 1037.0 1039.5 1052.3 908.3 896.3 908.9 920.7 892.3 892.0 744.0 733.7 749.5 742.7 759.0 814.2 840.7 846.8 840.8 836.5 860.3 844.6 829.1 801.0 844.0 853.4 942.6 973.0 970.1 978.2 983.0 997.9 812.8 1015.0 1050.0 1056.9 1112.3 1182.4 1225.5 1228.8 1289.1 1301.8 1304.9 1308.1 1366.2 1332.2 1446.0 1456.8 895.3

16m 2796.6 2880.6 3013.8 2876.7 2750.0 2617.7 2429.0 2390.9 2163.4 1714.0 1564.0 1427.8 1364.1 1302.1 1260.2 1230.1 1143.5 1101.8 1041.6 1053.2 1106.0 1094.4 1145.9 1195.0 1268.9 1361.9 1364.0 1426.8 1443.7 1491.3 1551.7 1067.3 1612.5 1473.5 1693.3 1744.7 1644.8 1711.6 1690.5 1792.5 1782.7 1715.3 1785.7 1836.9 1832.9 1691.8 2031.7 1383.4 1817.0 1990.3 1997.8 2026.6 1982.3 2029.1 1969.5 2142.6 2134.5 2142.0 2137.2 2100.2 2104.4 2060.9 2140.7 995.3

8m 3095.4 3040.2 3005.9 2885.8 2760.3 2637.1 2480.6 2372.4 2254.3 2067.2 1504.8 1432.6 1401.9 1352.8 1362.6 1310.2 1362.2 1404.3 1519.4 1963.1 1978.2 2038.5 2028.7 2097.9 2127.6 2131.8 2130.7 2125.8 2135.1 2145.0 2215.5 1219.2 1618.9 1719.9 1711.4 1818.7 1903.6 1975.6 1965.0 1898.7 2031.9 2117.4 2141.3 2127.2 2117.7 2132.3 2146.8 2121.9 2138.0 2130.1 2127.9 2118.3 2110.9 2105.8 2104.1 2107.3 2112.8 2109.0 2102.7 2103.5 2092.6 2095.0 2094.7 1095.4

4m 2395.4 2394.0 2570.0 2652.2 2716.0 2781.1 2752.6 2801.2 2782.0 2741.0 2731.8 2717.5 2675.5 2643.8 2677.0 2701.3 2734.2 2778.7 2759.1 2833.9 2901.6 2840.1 2970.8 3060.9 3060.8 3064.0 3063.1 3058.9 3058.0 3061.1 3056.4 2588.2 2679.9 3056.5 2656.9 2644.6 2636.5 2658.9 2678.3 2680.3 2661.2 2688.4 2693.6 2678.0 2684.2 3060.2 3053.9 2765.6 2705.0 2711.4 2696.2 2700.1 2667.9 2669.2 2700.5 2689.4 2682.4 2657.6 2672.9 2697.4 2678.8 2685.2 3049.4 2387.9

2m 3105.4 3103.4 3105.2 3105.7 3103.4 3101.3 3100.3 3097.0 3094.7 3093.0 3094.0 3093.3 3087.0 3080.1 3069.8 3064.8 3060.4 3054.5 3048.4 3047.2 3039.2 3021.8 2981.9 2912.4 2874.2 2838.2 2827.7 2830.7 2793.3 2776.5 2709.6 2960.3 2677.5 2680.6 2654.0 2611.5 2634.4 2653.8 2675.0 2671.0 2662.4 2685.1 2676.8 2676.2 2680.0 2691.5 2685.6 2752.7 2701.0 2683.4 2661.7 2679.6 2644.2 2650.7 2690.6 2660.0 2658.7 2661.8 2623.5 2655.4 2665.2 2676.1 2642.1 2914.9

1024k 3105.6 3106.6 3103.5 3104.5 3103.2 3100.5 3098.6 3097.4 3092.4 3092.9 3094.5 3092.0 3087.6 3078.3 3068.8 3064.0 3059.6 3052.2 3048.7 3040.4 3033.3 3012.2 2984.9 2915.3 2882.9 2858.9 2844.0 2825.0 2791.7 2784.8 2724.4 2967.3 2637.6 2668.6 2655.5 2604.0 2663.7 2669.2 2719.0 2675.8 2698.6 2759.2 2707.2 2731.8 2788.5 2792.0 2763.3 2907.3 2829.9 2792.7 2770.0 2803.0 2835.8 2872.1 2848.1 2861.3 2885.5 2892.9 2919.5 2906.5 3010.3 3031.2 2926.3 2885.6

512k 3105.1 3105.7 3105.3 2919.5 2924.4 2911.2 2922.2 3095.7 3093.1 3092.6 3089.1 3095.0 3089.4 3078.7 3069.2 3059.2 3063.3 2883.0 2862.6 2872.4 2871.8 2885.7 2882.8 2883.9 2888.6 2857.5 2891.5 2889.5 2879.4 2887.3 2906.5 2863.9 2914.7 2887.3 2883.7 2906.2 2907.0 2915.0 2892.9 2921.7 2913.9 2883.1 2895.7 2916.8 2883.5 2906.7 2913.0 2909.5 2918.3 2916.9 2912.9 2907.6 2911.6 2909.2 2898.9 2919.7 2910.5 2913.4 2915.4 2913.2 2909.3 2825.7 2786.7 2852.0

256k 2926.6 3016.3 3015.1 3013.2 3012.3 3013.2 2937.2 2979.8 3004.6 3004.5 3009.4 3003.3 2991.9 3008.2 3008.2 2999.1 3009.2 3005.9 3007.4 3009.5 3004.9 3005.8 3002.2 3010.1 3005.8 3002.9 3000.5 2996.7 3007.4 3002.6 2996.9 2985.6 2994.0 2994.8 2994.0 2991.0 3000.2 2996.6 2996.9 2986.6 2996.8 2988.0 2998.2 2987.5 2995.3 2990.4 2986.0 2984.2 2993.5 2982.5 2991.2 2989.9 2984.0 2992.4 2982.0 2979.1 2985.4 2981.1 2991.8 2975.9 2981.5 2987.9 2968.0 2872.8

128k 3003.3 3000.4 2969.6 2989.0 3101.4 3101.8 3100.6 3096.8 3091.4 3097.1 3093.5 3100.9 3097.4 3088.0 3084.2 3096.8 3084.3 3087.6 3080.1 3089.6 3075.3 3090.1 3072.3 3082.7 3079.6 3077.0 3072.6 3077.0 3074.8 3062.6 3085.3 3070.0 3059.3 3077.7 3068.2 3064.2 3069.2 3079.0 3062.7 3063.5 3068.3 3072.5 3057.2 3055.4 3060.2 3058.4 3061.2 3038.6 3068.3 3067.1 3057.0 3044.9 3062.2 3057.7 3061.6 3051.7 3034.8 3062.9 3065.8 3050.4 3049.3 3044.3 3022.1 3036.7

64k 3106.7 3106.6 3108.6 3096.4 3099.4 3096.2 3095.8 3094.1 3077.6 3081.8 3077.8 3081.4 3072.7 3075.5 3072.5 3068.3 3083.3 3072.2 3066.4 3068.0 3070.2 3070.0 3053.3 3056.9 3069.9 3056.3 3048.8 3045.6 3047.0 3047.1 3057.8 3026.4 3033.4 3041.3 3035.7 3046.4 3038.8 3042.6 3022.4 3026.5 3030.2 3042.5 3036.1 3017.3 3021.1 3017.4 3027.9 3015.4 3009.8 3033.1 3012.8 3014.1 3006.5 3001.1 3003.9 3003.4 2996.9 2998.9 3003.8 2992.4 2980.9 2984.7 2969.0 2942.7

32k 3107.2 3104.1 3100.3 3084.3 3090.7 3074.7 3070.8 3075.4 3070.2 3070.2 3065.1 3070.2 3056.3 3051.7 3050.4 3043.7 3056.2 3046.4 3034.3 3026.5 3019.8 3026.7 3017.4 3025.6 3011.7 3014.1 2995.4 2991.6 2986.7 2992.4 2984.7 2975.6 2982.7 2978.2 2972.6 2984.9 2970.4 3001.8 2948.9 2971.6 2990.9 2971.1 2954.6 2928.5 3183.9 3211.2 3245.0 3244.9 3286.7 2923.7 2946.7 2933.0 2887.5 2904.6 2905.5 2889.2 2889.8 2895.6 2895.2 2871.3 2882.3 2869.6 2862.0 2842.1

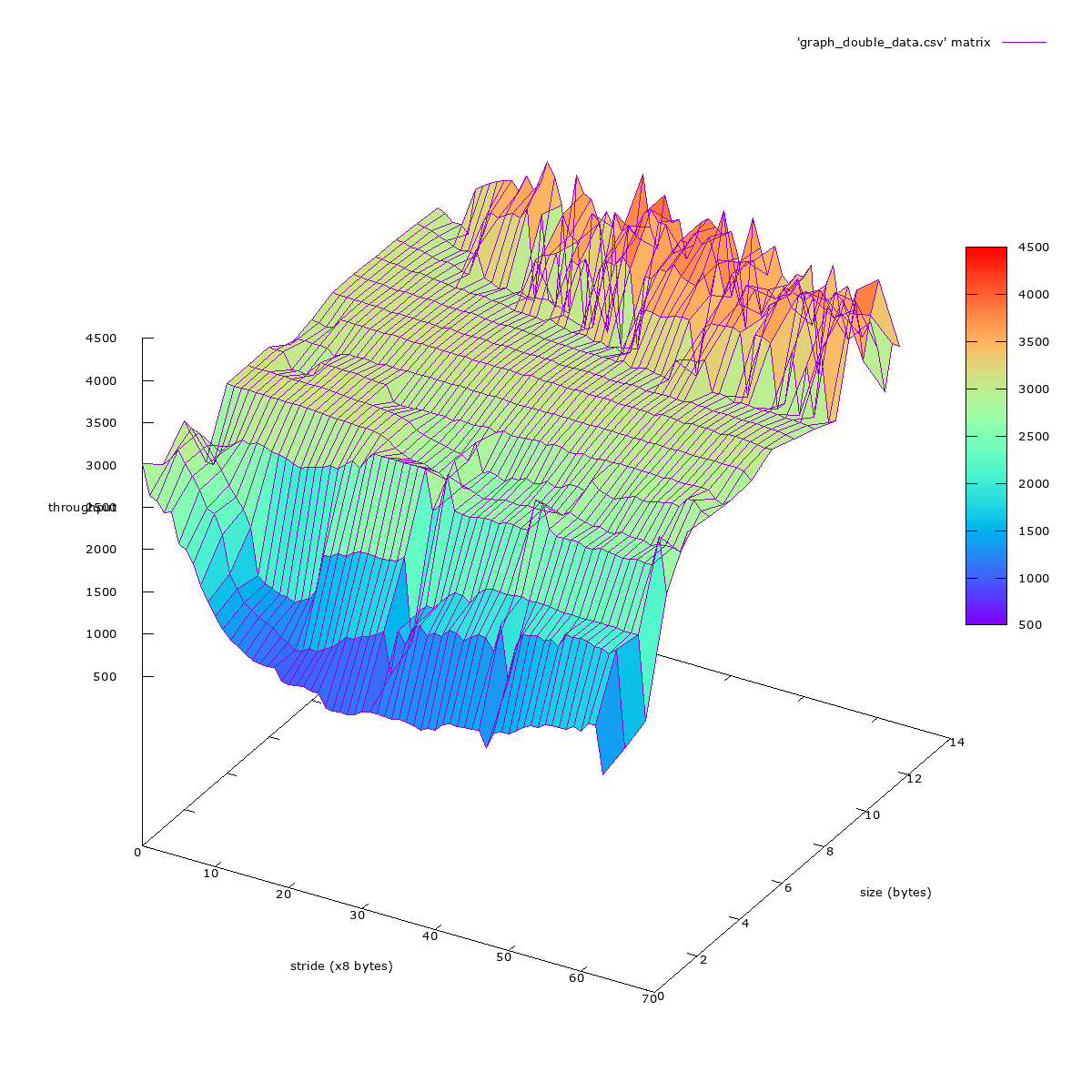
16k 3103.2 3102.3 3086.8 3084.3 3072.4 3056.9 3064.0 3061.2 3038.6 3030.8 3022.0 3010.2 3025.2 2997.5 3005.1 3009.3 2985.3 2984.9 2969.9 2938.7 2945.2 2928.5 2936.0 2923.6 2964.4 2922.6 2904.6 2878.3 2895.6 2894.7 2845.8 2866.6 2825.7 2852.9 2801.4 2841.4 3318.6 3392.9 3388.5 3362.4 3387.4 3422.8 3436.7 3426.9 2781.2 3375.3 3450.6 3460.4 3470.7 3481.5 3445.9 2709.7 3460.1 3482.9 3414.0 2671.0 2761.4 3391.6 2665.2 3283.4 2698.5 3463.8 2680.3 2725.6

8k 3095.9 3082.5 3056.9 3054.2 3039.5 3010.2 2997.5 3009.3 2984.9 2938.7 2928.5 3244.9 2933.0 2911.4 2859.7 2891.5 2852.9 2897.1 2856.3 3320.4 3399.9 3426.9 3476.8 2768.3 3453.2 3409.7 3514.0 2728.7 3455.6 3546.1 3535.2 3654.1 3539.9 2619.7 3493.6 3510.2 3633.7 3626.9 3736.6 3629.9 3590.1 3567.5 3739.3 3501.4 3940.1 3557.0 3830.7 3505.0 3740.0 3714.5 3407.0 3577.7 3637.0 3845.8 3627.2 3792.8 3869.7 3815.6 3200.9 3758.6 3955.8 3896.8 3752.4 4055.2

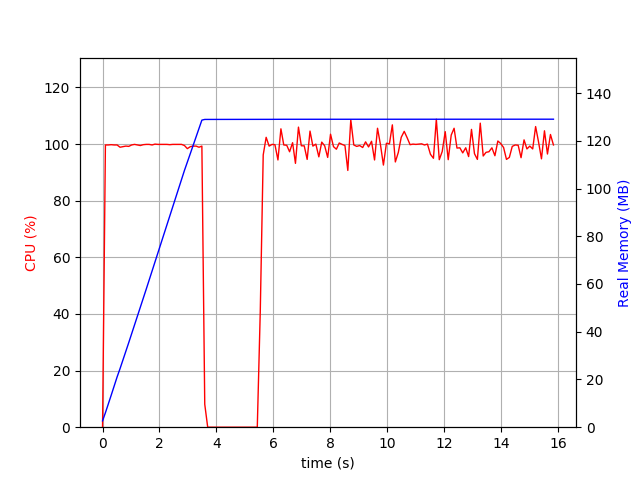
4k 3071.8 3050.7 3036.0 2995.7 2963.3 2923.6 2900.3 2916.9 2883.0 3320.4 3451.4 3407.2 3467.5 3511.9 3476.5 3614.4 2555.3 3595.8 2610.0 2547.9 3670.9 3451.4 3400.1 2400.2 3780.8 3707.8 2421.4 3870.3 3663.0 3758.6 3648.0 3958.6 4026.7 3896.8 3896.8 3580.0 3760.0 3756.3 3896.8 3785.4 3896.8 3599.9 4255.1 3264.9 3693.8 3729.2 3766.9 3154.5 3477.8 3394.0 3463.8 3493.6 3704.3 3479.3 3204.0 3647.0 3547.1 3367.6 3447.1 3154.5 3626.2 2956.2 3517.9 3958.6

2k 3043.7 2995.7 2914.0 2916.9 2825.8 3381.2 3448.0 3500.2 3552.5 3580.8 3606.0 3505.0 3707.8 3578.2 3680.3 3958.6 3801.7 3494.7 2241.7 3896.8 3705.7 3660.6 3400.1 3561.6 3394.0 2026.3 3479.3 3386.5 4132.9 3533.1 3297.3 3958.6 3660.6 3711.2 3275.5 3463.8 3760.0 3824.6 3752.4 3680.3 3743.9 3463.8 2654.3 3983.4 3653.2 3361.9 3723.6 3409.7 3328.5 3463.8 3463.8 3618.4 2350.4 3432.9 3696.9 3117.4 3497.1 3247.3 2944.2 3154.5 2857.6 2381.4 2969.0 2969.0

**Memory mountain for double matrix:**



**CPU/Memory Utilization for double matrix:**



3. Using function

read\_float\_data\_from\_csv("float\_data.csv"); /\* Load data from a csv file \*/

to load the float data from csv to the data matrix, and then get the benchmark result and memory mountain graph.

**Benchmark result for float matrix:**

Clock frequency is approx. 2593.0 MHz

Memory mountain (MB/sec)

Max Elements: 16785409

s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53 s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64

32m 3064.8 3011.5 3001.4 2872.2 2734.9 2575.3 2415.6 2391.7 2084.2 1631.9 1495.7 1386.8 1286.6 1225.7 1188.2 1155.3 1089.6 1027.1 994.4 961.0 962.9 961.9 974.4 986.8 982.4 986.8 930.7 920.8 919.9 910.7 912.1 928.9 907.6 885.7 882.8 892.4 877.4 874.8 829.7 881.4 882.7 835.2 885.6 931.1 942.4 1028.9 1081.8 906.0 1025.7 1120.3 1080.6 1153.1 1174.4 1217.8 1206.4 1391.0 1262.4 1359.4 1319.6 1386.6 1408.8 1405.9 1467.5 887.1

16m 3088.8 3018.1 2996.5 2873.5 2747.2 2590.4 2431.3 2359.5 2179.7 1843.5 1549.9 1441.0 1354.4 1265.7 1209.9 1189.3 1137.1 1107.6 1088.9 1104.7 1071.9 1116.0 1160.6 1277.7 1325.1 1348.4 1397.5 1384.1 1450.8 1483.7 1470.0 1176.9 1510.2 1549.1 1575.5 1613.5 1719.7 1689.8 1736.7 1750.7 1789.4 1836.9 1854.8 1925.3 1943.2 1988.6 2037.8 1526.9 1861.5 1955.6 2060.1 2052.9 2046.1 2034.4 2044.1 2039.8 2113.2 2184.6 2156.3 2197.5 2198.8 2137.4 2175.4 1051.7

8m 3086.1 3033.6 2995.5 2877.6 2742.5 2623.8 2483.1 2368.0 2245.4 2084.1 1940.8 1787.4 1419.1 1266.6 1239.0 1200.5 1722.6 1786.2 1850.6 1938.5 2007.6 2109.8 2137.8 2223.3 2178.5 2208.2 2193.2 2220.3 2218.3 2245.1 2220.2 1241.1 2092.3 2095.9 2158.8 2211.8 2215.6 2251.3 2191.0 2251.8 2233.8 2223.5 2237.7 2217.8 2248.7 2256.0 2248.2 2221.7 2247.5 2246.1 2238.8 2223.9 2222.6 2234.4 2229.5 2219.4 2217.1 2210.0 2199.9 2190.1 2181.8 2173.2 2186.9 934.5

4m 2829.5 2816.8 2822.3 2811.9 2791.2 2756.2 2746.2 2742.9 2800.1 2780.0 2755.3 2728.0 2700.4 2660.0 2629.5 2636.8 2689.9 2614.3 2676.2 2690.0 2778.5 2739.1 2871.7 2874.6 2876.6 2868.1 2880.6 2822.6 2874.5 2878.0 2869.1 2285.4 2847.2 2881.3 2830.7 2848.8 2841.2 2838.7 2838.3 2836.5 2847.3 2855.1 2850.6 2854.1 2867.1 2867.3 2863.9 2864.8 2864.2 2867.5 2855.4 2852.1 2851.2 2842.4 2850.3 2847.9 2845.3 2828.4 2858.9 2849.5 2850.8 2847.8 2844.9 2141.8

2m 2920.1 2909.9 2913.8 2909.3 2891.3 2886.3 2882.6 2882.2 2876.1 2863.2 2872.2 2868.6 2850.8 2833.4 2861.0 2839.3 2854.9 2845.7 2846.9 2833.5 2856.0 2863.4 2857.9 2874.8 2858.5 2863.6 2851.6 2871.7 2851.7 2862.0 2871.9 2754.1 2865.9 2855.9 2846.1 2862.1 2863.0 2861.4 2853.3 2859.5 2865.5 2853.6 2853.2 2866.4 2854.1 2862.8 2855.4 2866.3 2859.2 2866.9 2864.5 2868.5 2863.2 2853.1 2863.0 2856.7 2860.1 2853.3 2855.9 2842.9 2853.2 2855.4 2848.3 2747.4

1024k 2921.6 2916.8 2918.6 2914.3 2904.4 2897.2 2902.5 2906.9 2903.9 2902.6 2904.2 2899.6 2899.0 2891.6 2885.4 2881.1 2882.0 2878.9 2877.3 2874.4 2876.2 2877.2 2875.4 2873.7 2878.9 2873.8 2873.1 2872.1 2872.2 2875.1 2867.2 2864.3 2858.7 2867.7 2860.2 2866.9 2862.6 2862.0 2864.9 2864.7 2869.3 2861.4 2861.6 2863.5 2870.3 2867.1 2872.9 2861.1 2875.6 2875.9 2861.0 2870.0 2873.3 2870.6 2865.5 2871.5 2867.0 2869.5 2879.8 2872.0 2874.8 2874.1 2885.3 2856.2

512k 2921.3 2909.9 2916.1 2913.7 2900.9 2897.2 2901.1 2906.3 2905.2 2904.4 2902.5 2897.2 2893.4 2888.1 2891.3 2877.3 2880.7 2886.0 2880.2 2880.2 2882.6 2876.0 2877.1 2879.1 2878.6 2884.2 2884.8 2875.9 2887.9 2881.9 2875.6 2862.0 2890.1 2879.9 2877.5 2883.0 2878.4 2882.8 2877.5 2880.2 2891.2 2885.4 2888.2 2890.7 2886.9 2886.5 2892.5 2878.7 2897.1 2897.8 2884.2 2888.7 2887.2 2891.1 2891.5 2884.5 2895.9 2890.7 2887.4 2876.1 2889.5 2893.4 2886.4 2846.7

256k 2921.8 2905.7 2916.7 2913.7 2903.3 2884.3 2901.4 2910.0 2903.7 2910.5 2910.1 2903.9 2893.8 2898.6 2903.0 2889.6 2900.7 2897.6 2891.2 2888.0 2887.4 2893.9 2903.0 2894.0 2896.0 2895.1 2896.4 2888.6 2897.2 2889.3 2891.9 2878.3 2896.8 2888.2 2889.7 2881.5 2891.8 2887.7 2889.2 2888.8 2883.0 2889.8 2878.2 2891.5 2878.1 2884.6 2883.0 2878.1 2894.4 2879.8 2893.6 2876.6 2883.3 2890.8 2882.0 2887.2 2883.3 2887.6 2885.9 2873.9 2880.6 2889.6 2872.6 2849.0

128k 2922.6 2902.2 2923.4 2917.4 2903.8 2881.8 2905.5 2909.0 2903.4 2906.8 2898.8 2897.5 2891.2 2892.7 2894.4 2887.7 2889.8 2893.0 2878.4 2890.8 2885.7 2886.1 2881.2 2887.5 2878.5 2888.0 2893.4 2883.1 2893.3 2878.2 2883.5 2881.4 2873.0 2902.4 2880.8 2882.9 2880.9 2894.2 2883.0 2885.4 2875.5 2893.5 2880.3 2879.2 2879.8 2883.5 2882.5 2849.8 2881.8 2879.8 2874.9 2873.5 2875.5 2878.9 2884.7 2872.9 2860.5 2877.0 2873.8 2854.4 2862.1 2876.0 2856.7 2836.8

64k 2923.6 2918.6 2915.0 2913.4 2893.4 2881.6 2892.0 2900.3 2891.2 2887.8 2894.8 2872.3 2893.2 2874.9 2884.1 2879.9 2885.7 2886.4 2892.4 2887.4 2883.2 2887.8 2885.7 2879.9 2867.6 2873.5 2876.2 2875.6 2874.2 2871.7 2852.0 2842.8 2850.0 2871.7 2858.7 2860.1 2859.4 2865.6 2839.8 2858.2 2838.1 2840.6 2855.3 2843.1 2851.5 2831.6 2828.3 2836.1 2841.7 2844.9 2838.2 2841.0 2831.2 2823.4 2833.3 2824.8 2826.7 2825.8 2821.8 2825.9 2824.4 2808.2 2814.9 2760.1

32k 2919.8 2914.6 2920.9 2905.9 2907.6 2894.6 2895.5 2886.1 2879.3 2891.3 2883.5 2879.9 2876.0 2886.6 2868.8 2855.1 2871.7 2863.6 2865.6 2858.2 2844.6 2830.6 2844.7 2845.2 2835.4 2855.9 2849.0 2819.5 2814.9 2803.5 2796.7 2806.8 2795.9 2818.3 2789.7 2822.5 2812.3 2800.9 2764.1 2765.2 2784.7 2778.2 2775.1 2748.1 3054.5 3016.7 3022.3 3070.2 2767.0 2748.2 2738.0 2778.2 2734.6 2728.0 2755.1 2718.5 2756.3 2698.2 2725.6 2733.2 2698.5 2705.7 2707.5 2698.4

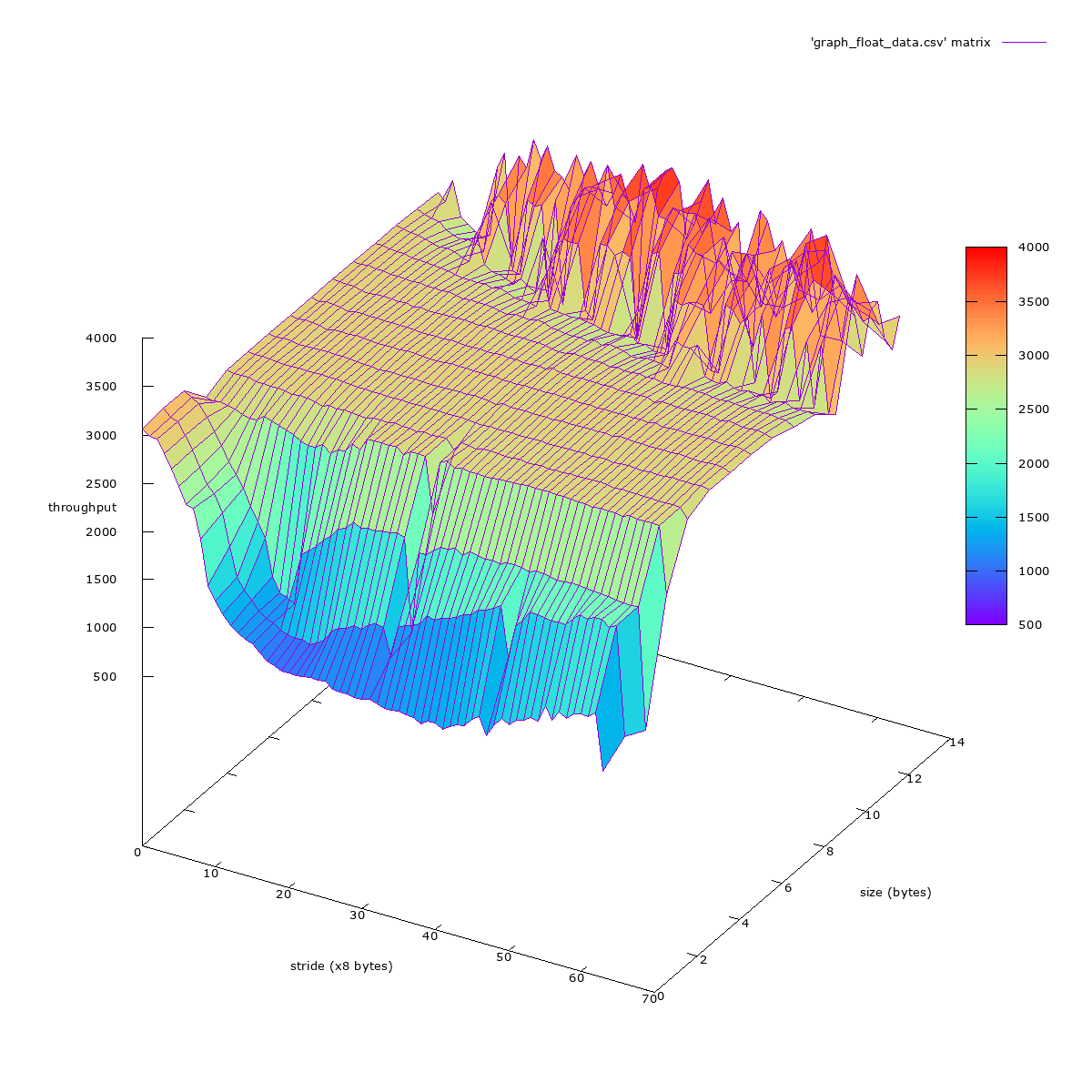
16k 2913.4 2917.8 2904.1 2898.7 2899.2 2875.2 2881.1 2870.5 2863.6 2862.1 2847.3 2836.1 2821.4 2819.5 2820.3 2824.7 2805.7 2822.5 2800.9 2765.2 2763.0 2763.9 2747.3 3059.5 2757.2 2750.1 2747.1 2738.1 2749.0 2733.2 2727.3 2709.4 2690.6 2688.0 2696.7 2731.0 2665.4 2673.6 2630.6 2638.1 2639.3 3220.6 3186.9 3236.9 3254.6 3250.4 2593.0 2600.6 3331.0 3286.5 3176.9 3267.2 2601.4 3329.1 2484.3 2575.4 3235.6 3293.8 2547.0 3188.7 2517.8 3198.8 2497.0 2514.4

8k 2914.6 2898.7 2889.3 2876.7 2865.9 2840.6 2819.5 2836.8 2815.8 2787.0 2756.0 3070.2 2768.8 2718.5 2712.2 2687.5 2653.7 2731.0 2673.6 2638.1 3220.6 2593.0 3183.1 3274.9 3261.2 3267.2 2601.6 2523.8 3323.7 3339.1 3259.8 3319.0 3280.9 2469.5 2862.1 3382.8 3411.0 2445.1 3241.2 3265.2 3583.4 3511.3 3519.1 2940.8 3521.8 3444.4 3638.6 3498.5 3669.7 3464.4 3191.4 3571.0 3273.1 3625.4 3690.0 3441.6 3944.7 3584.4 3032.5 3390.8 3696.4 3565.4 3439.7 3771.6

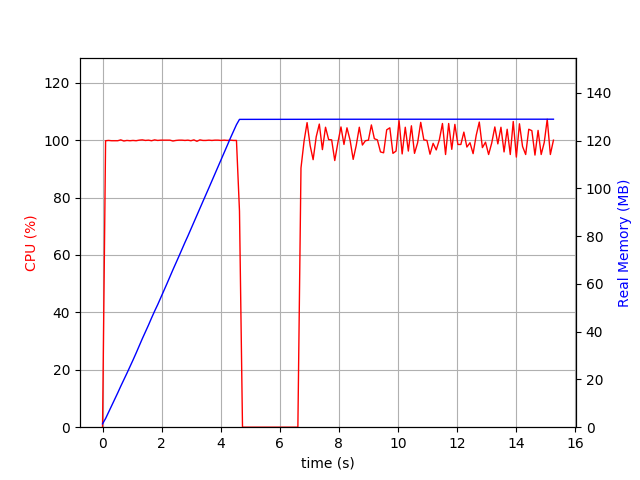
4k 2897.1 2867.4 2845.2 2836.8 2772.4 2737.5 2718.5 2731.7 2718.5 2651.3 3194.0 3274.9 3215.7 2540.8 2528.2 3319.0 3275.4 3306.8 2445.1 3390.8 2340.9 3349.3 3496.6 3613.2 3643.6 3700.9 3915.4 3505.3 3656.1 3598.4 2194.1 3771.6 3827.7 3535.9 3527.7 3662.6 3656.8 3650.7 3679.2 3480.1 3667.2 3224.6 3519.1 3091.6 3104.8 3118.6 3418.0 2900.1 3165.0 2838.3 2963.4 3487.1 3327.7 3601.4 2998.1 3154.8 3835.5 3241.2 2795.6 3265.2 3217.2 3056.0 3121.2 3073.2

2k 2879.9 2806.8 3049.0 2687.5 2638.1 2585.4 2540.8 2593.0 3306.8 3480.1 2387.6 3498.5 3392.5 3711.5 3526.5 3687.8 3457.3 3407.1 3303.0 3673.4 3493.3 3653.8 3393.8 3241.2 3387.6 3611.7 3353.0 3154.8 3781.4 3526.5 3565.4 3771.6 3827.7 3704.3 3133.2 3300.2 3565.4 3817.5 3370.9 3673.4 3343.6 3457.3 2437.4 2208.8 3646.4 3565.4 3279.4 3403.3 3126.8 3241.2 2881.1 3611.7 3284.5 3426.5 3198.0 3111.6 3025.2 2836.1 2448.9 2938.7 3056.0 2674.0 2593.0 2963.4

**Memory mountain for float matrix:**



**CPU/Memory Utilization for float matrix:**



**Task 2: Benchmark Vector/Matrix computation**

1. Add implementation:

/\* Matrix add \*/  
**void** addTwoMatrix(**double**\* A, **double**\* B, **double**\* result, **int** row\_A, **int** col\_A, **int** row\_B, **int** col\_B)  
{  
 **if** (row\_A != row\_B || col\_A != col\_B) {  
 printf("Either column or row between Matrix A and Matrix B is not the same.");  
 exit(1);  
 }  
  
 **for** (**int** i = 0; i < row\_A; i++) {  
 **for** (**int** j = 0; j < col\_A; j++) {  
 result[i\*col\_A + j] = A[i\*col\_A + j] + B[i\*col\_A + j];  
 }  
 }  
}

1. Subtract implementation

/\* Matrix subtract \*/  
**void** subtractTwoMatrix(**double**\* A, **double**\* B, **double**\* result, **int** row\_A, **int** col\_A, **int** row\_B, **int** col\_B)  
{  
 **if** (row\_A != row\_B || col\_A != col\_B) {  
 printf("Either column or row between Matrix A and Matrix B is not the same.");  
 exit(1);  
 }  
  
 **for** (**int** i = 0; i < row\_A; i++) {  
 **for** (**int** j = 0; j < col\_A; j++) {  
 result[i\*col\_A + j] = A[i\*col\_A + j] - B[i\*col\_A + j];  
 }  
 }  
}

1. Multiply implementation

/\* multiply \*/  
**void** multiply(**int** m1, **int** m2, **int** mat1[][m2],  
 **int** n1, **int** n2, **int** mat2[][n2], **int** res[m1][n2])  
{  
 **int** x, i, j;  
 **for** (i = 0; i < m1; i++) {  
 **for** (j = 0; j < n2; j++) {  
 res[i][j] = 0;  
 **for** (x = 0; x < m2; x++) {  
 \*(\*(res + i) + j) += \*(\*(mat1 + i) + x) \*  
 \*(\*(mat2 + x) + j);  
 }  
 }  
 }  
 **for** (i = 0; i < m1; i++) {  
 **for** (j = 0; j < n2; j++) {  
 printf("%d ", \*(\*(res + i) + j));  
 }  
 printf("\n");  
 }  
}

1. Convolution

bool convolve1D(float\* in, float\* out, int dataSize, float\* kernel, int kernelSize)

{

int i, j, k;

// check validity of params

if(!in || !out || !kernel) return false;

if(dataSize <=0 || kernelSize <= 0) return false;

// start convolution from out[kernelSize-1] to out[dataSize-1] (last)

for(i = kernelSize-1; i < dataSize; ++i)

{

out[i] = 0; // init to 0 before accumulate

for(j = i, k = 0; k < kernelSize; --j, ++k)

out[i] += in[j] \* kernel[k];

}

// convolution from out[0] to out[kernelSize-2]

for(i = 0; i < kernelSize - 1; ++i)

{

out[i] = 0; // init to 0 before sum

for(j = i, k = 0; j >= 0; --j, ++k)

out[i] += in[j] \* kernel[k];

}

return true;

}

bool convolve2D(unsigned char\* in, unsigned char\* out, int dataSizeX, int dataSizeY,

float\* kernel, int kernelSizeX, int kernelSizeY)

{

int i, j, m, n;

unsigned char \*inPtr, \*inPtr2, \*outPtr;

float \*kPtr;

int kCenterX, kCenterY;

int rowMin, rowMax; // to check boundary of input array

int colMin, colMax; //

float sum; // temp accumulation buffer

// check validity of params

if(!in || !out || !kernel) return false;

if(dataSizeX <= 0 || kernelSizeX <= 0) return false;

// find center position of kernel (half of kernel size)

kCenterX = kernelSizeX >> 1;

kCenterY = kernelSizeY >> 1;

// init working pointers

inPtr = inPtr2 = &in[dataSizeX \* kCenterY + kCenterX]; // note that it is shifted (kCenterX, kCenterY),

outPtr = out;

kPtr = kernel;

// start convolution

for(i= 0; i < dataSizeY; ++i) // number of rows

{

// compute the range of convolution, the current row of kernel should be between these

rowMax = i + kCenterY;

rowMin = i - dataSizeY + kCenterY;

for(j = 0; j < dataSizeX; ++j) // number of columns

{

// compute the range of convolution, the current column of kernel should be between these

colMax = j + kCenterX;

colMin = j - dataSizeX + kCenterX;

sum = 0; // set to 0 before accumulate

// flip the kernel and traverse all the kernel values

// multiply each kernel value with underlying input data

for(m = 0; m < kernelSizeY; ++m) // kernel rows

{

// check if the index is out of bound of input array

if(m <= rowMax && m > rowMin)

{

for(n = 0; n < kernelSizeX; ++n)

{

// check the boundary of array

if(n <= colMax && n > colMin)

sum += \*(inPtr - n) \* \*kPtr;

++kPtr; // next kernel

}

}

else

kPtr += kernelSizeX; // out of bound, move to next row of kernel

inPtr -= dataSizeX; // move input data 1 raw up

}

// convert negative number to positive

\*outPtr = (unsigned char)((float)fabs(sum) + 0.5f);

kPtr = kernel; // reset kernel to (0,0)

inPtr = ++inPtr2; // next input

++outPtr; // next output

}

}

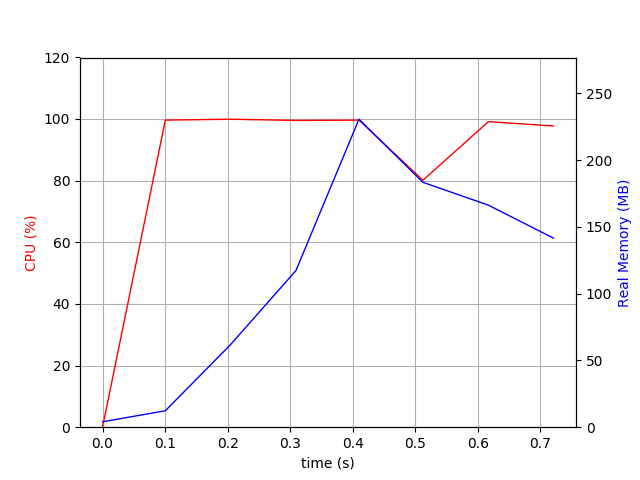
return true;

}

BLAS: It is easy to set up in C, but unfortunately, I cannot successfully run this one to get the result.

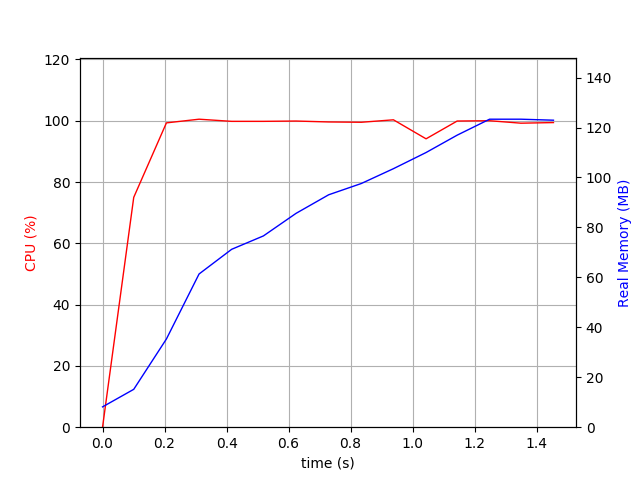
NUMPY: It is easy to set up in python. Matrix computation in numpy are very fast.

Here is the CPU&memory graph for add, subtract, multiply:



Tensorflow: Tensor flow is a library which is mostly from deep learning, it is designed to perform best for GPUs. The GPUs are installed on the PCIe buses, and its communication is slower compare to the commucation between CPU and memorys.

Here is the CPU&Memory graph for Tensorflow for multiply:



**Reference**

1. CPU&memory utilization: <https://pypi.org/project/psrecord/>
2. Convolution for 1D&2D: <http://www.songho.ca/dsp/convolution/convolution.html>
3. Multiply: <https://www.geeksforgeeks.org/c-program-multiply-two-matrices/>
4. BLAS: <https://www.gnu.org/software/gsl/manual/html_node/BLAS-Examples.html#BLAS-Examples>