Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 3, Number 1, pages 86–90, January 2014

Comparison of GPU Implementations of Row-wise and Column-wise Fundamental Algorithms

Daisuke Takafuji, Yasuaki Ito and Koji Nakano
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, JAPAN

***Abstract*—In this paper, we treat GPU implementations of the following three fundamental algorithms for each 1-dimensional array, which is one of given L 1-dimensional arrays: (1) calculating the prefix sum, (2) sorting by Bubble sort, (3) sorting by Bitonic sort. There are two methods to store L 1-dimensional arrays as a 2-dimensional array: one is to store each 1-dimensional array as a row-wise array, and the other is to do each 1-dimensional array as a column-wise one. A 2-dimensional array stored by the first or second method is called by a row-wise array or a column-wise array, respectively. We compared with the computation times of applying GPU implementation of above three algorithms to a row-wise array and a column-wise array. The experiment results show that GPU implementation of column-wise prefix sum is speedup factor up to 46.53 over that of row-wise one, and that GPU implementation of column-wise Bubble sort or Bitonic sort is to 58.52 or 17.86, respectively, over row-wise one.**

I. はじめに

本稿では，L 本の配列 A^i $(0 \leq i < L, |A^i| = N)$ が与えられたとき，各 A^i に対する次のアルゴリズムの GPU 実装を扱う．

1) prefix sum を計算．
2) Bubble sort によって昇順に整列．
3) Bitonic sort によって昇順に整列．

L 本の配列 A^i を 2 次元配列 `fA` に保存するとき，次の 2 通りで保存することが可能である．

Row-wise array:
各配列 A^i $(0 \leq i < L)$ に対し，`fA[i][j]` $= A^i_j$ $(0 \leq j < N)$ (Fig. 1 参照).
Column-wise array:
各配列 A^i $(0 \leq i < L)$ に対し，`fA[j][i]` $= A^i_j$ $(0 \leq j < N)$ (Fig. 2 参照).

Row-wise に保存されている L 本の配列 A^i $(0 \leq i < L, |A^i| = N)$ に対し，操作 P を実行するとき，Row-wise 操作 P と呼び，それらが Column-wise に保存されているとき，Column-wise 操作 P と呼ぶ．

本稿では，Row-wise 操作 P と Column-wise 操作 P を GPU 上で実装し，その実行時間を CPU 実装のものと比較した結果を報告する．GPU 実装において，配列 A^i に対する操作を 1 thread で実行するものとする．ここで，操作 P は，1) prefix sum の計算，2) Bubble sort による整列，3) Bitonic sort による整列である．その結果，prefix sum の計算の場合，Column-wise は Row-wise に比べて最大で 46.53 倍の高速化となった．同様に，Bubble sort および Bitonic sort による整列の場合，それぞれ最大で 58.52 倍および 17.86 倍の高速化となった．

II. GPU 実装

GPU (Graphics Processing Unit) は，画像処理計算の高速化を目的に設計された専用回路である [1]．最近の GPU は汎用計算のために設計され従来の CPU 計算にも応用でき，多くのアプリケーション開発者に注目されている [1]．NVIDIA は，自社が提供する GPU を動作させるための並列計算機アーキテクチャを提供しており，これを *CUDA* (Compute Unified Device Architecture) [2] という．CUDA は，NVIDIA 社 GPU での仮想命令セットと並列計算のメモリを提供する．

GPU は複数のストリーミング・マルチプロセッサ（streaming multiprocessor）で構成されており，各ストリーミング・マルチプロセッサには CUDA コアと呼ばれる演算装置が含まれている（Fig. 3 参照）．GPU のメモリは高速で小容量なオンチップメモリ（shared memory）と低速で大容量なオフチップメモリ（global memory）で構成されている（Fig. 3 参照）．一方，CUDA の並列計算モデルは，グリッド（grid），スレッド・ブロック（thread block），スレッド（thread）と呼ばれる階層で構成されている．1 つのグリッドは複数のスレッド・ブロックで構成されており，各スレッド・ブロックも複数のスレッドで構成されている．1 スレッドは 1 つの CUDA コアで実行され，1 スレッド・ブロックは 1 つのストリーミング・マルチプロセッサで実行される．

CUDA による GPU 実装において，高速化のために重要

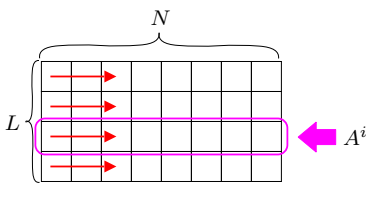

Fig. 1. Row-wise array.

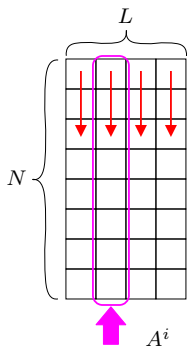

Fig. 2. Column-wise array.

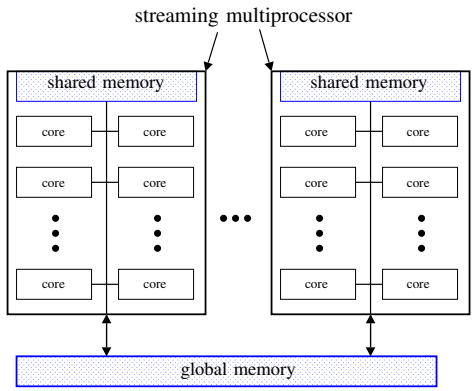

Fig. 3. CUDA hardware architecture.

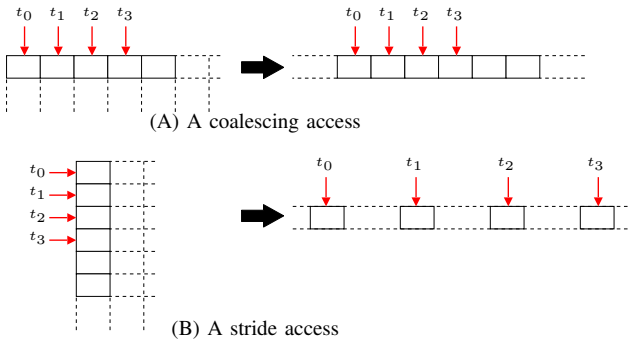

Fig. 4. An example of a coalescing access and a stride access.

な点を記す．スレッドは並列に動作するとき 32 スレッドごとに動作するため，32 スレッドを 1 ワープ（warp）と呼ぶ．1 ワープ内の 32 スレッドは並列に動作しており，if 文などの分岐がある場合にはその then 節と else 節の両方を実行するため，動作遅延の要因になり，可能な限りこのような記述は避けるべきである．スレッドによって 2 次元配列の行にある連続した場所がアクセスされるとき，golobal memory のアドレス空間にある対応する場所は同時にアクセスされる．このようなアクセス方法を coalescing access とよぶ（Fig. 4 (a) 参照）．しかし，もし列にある連続した場所がアクセスされるならば遠く離れた場所が同時にアクセスされることになる．このようなアクセス方法を stride access とよぶ（Fig. 4 (b) 参照）．

A. *prefix sum* の計算における *GPU* 実装

本節では，Row-wise prefix sum と Column-wise prefix sum の CUDA [3] による GPU プログラムの実装例を記す．

配列 X ($|X| = m$) に対し，配列 Y ($|X| = m$) が $Y_j = \sum_{k=0}^{j} X_k$ を満たすならば，Y は X の prefix sum である．ただし，配列 X の要素を X_j ($0 \leq j < m$) と表す．

X の prefix sum Y を計算するアルゴリズムを Algorithm 1 に記す．また，Row-wise prefix sum，Column-wise prefix sum の CUDA プログラムをそれぞれ Fig. 5, 6 に記す．

このことから，多くの配列が与えられたとき，各配列に対しある操作を 1 thread で実行させる CUDA プログラムは非常に簡単であることがわかる．

Algorithm 1 Calculate Prefix Sum Y of X

$Y_0 = X_0$.
for $j = 1$ to $m - 1$ **do**
  $Y_j = X_j + X_{j-1}$.
end for

```
1  __device__ void psum_core(DATA *fA)
2  {
3      int i,j,t;
4
5      for(i=1;i<N;i++) {
6          fA[i] += fA[i-1];
7      }
8  }
9  __global__ void psum_row_wise
                      (DATA *fA)
10 {
11     int id, p;
12     DATA *fB;
13     id = blockIdx.x * blockDim.x
14                + threadIdx.x;
15     p = id * N;
16     fB = &fA[p];
17     psum_core(fB);
18 }
```

Fig. 5. CUDA program for Row-wise prefix sum.

III. 実験結果

A. 実験環境

NVIDIA 社が提供する並列計算アーキテクチャCUDA を使用して，prefix sum の計算, bubble sort, bitonic sort のアルゴリズムを GPU 上に実装し，NVIDIA GTX680 を使用して実験を行った．1 スレッド・ブロック当たりのスレッド数 T を $T = 64$ とし，起動するスレッド・ブロック数 B を $B = L/T$ とした．計算中の配列 A^i は global memory に保存したまま，読み書きを行った．

また，CPU 実装の実行時間を計測するため，Intel Core i7 3.5GHz を使用した．

B. 入力データ

入力に使用した配列 A^i ($0 \leq i < L$) に対し，$|A^i| = N = 1024$，$L \in \{64 \times 2^k \mid 1 \leq k \leq 10\}$ とする．配列は，int 型，

```
1  __global__ void psum_column_wise
                         (DATA *fA)
2  {
3      int id, i;
4      id = blockIdx.x * blockDim.x
5               + threadIdx.x;
6      for(i=1;i<N;i++) {
7          fA[i * L + id]
8           += fA[(i-1) * L + id];
9      }
10 }
```

Fig. 6. CUDA program for Column-wise prefix sum.

TABLE I. COMPARISON OF GPU AND CPU TIME (MS) FOR CALCULATING PREFIX SUM, WHERE EACH INPUT IS THE INT DATA TYPE. AND RATIO REPRESENTS GPU TIME OF ROW-WISE/GPU TIME OF COLUMN-WISE.

$T \times B = L$	GPU		Ratio	CPU
	Row-wise	Column-wise	(R/C)	
$64 \times 1 = 64$	1.257	0.073	17.22	0.121
$64 \times 2 = 128$	1.259	0.108	11.66	0.254
$64 \times 4 = 256$	1.262	0.108	11.69	0.495
$64 \times 8 = 512$	1.265	0.112	11.29	0.987
$64 \times 16 = 1024$	1.712	0.121	14.15	2.343
$64 \times 32 = 2048$	3.694	0.141	26.20	3.924
$64 \times 64 = 4096$	9.609	0.231	41.60	7.904
$64 \times 128 = 8192$	24.520	0.527	46.53	16.524
$64 \times 256 = 16384$	42.252	1.118	37.80	31.517
$64 \times 512 = 32768$	74.270	2.256	32.92	63.386
$64 \times 1024 = 65536$	138.266	4.581	30.18	126.631

TABLE II. COMPARISON OF GPU AND CPU TIME (MS) FOR EXECUTING BUBBLE SORT, WHERE EACH INPUT IS THE INT DATA TYPE WITH TYPE S. AND RATIO REPRESENTS GPU TIME OF ROW-WISE/GPU TIME OF COLUMN-WISE.

$T \times B = L$	GPU		Ratio	CPU
	Row-wise	Column-wise	(R/C)	
$64 \times 1 = 64$	1,719.475	132.342	12.99	43.810
$64 \times 2 = 128$	1,719.473	129.744	13.25	91.868
$64 \times 4 = 256$	1,719.471	175.638	9.79	179.701
$64 \times 8 = 512$	1,720.505	188.563	9.12	349.210
$64 \times 16 = 1024$	1,724.047	195.293	8.83	697.334
$64 \times 32 = 2048$	2,034.586	208.852	9.74	1,393.435
$64 \times 64 = 4096$	6,463.130	235.995	27.39	2,787.310
$64 \times 128 = 8192$	17,945.799	306.639	58.52	5,575.071
$64 \times 256 = 16384$	35,588.621	618.229	57.57	11,147.293
$64 \times 512 = 32768$	71,059.414	1,236.398	57.47	22,314.887
$64 \times 1024 = 65536$	141,566.953	2,476.417	57.17	44,609.561

float 型，double 型を用意し，各要素を次のように設定した．

1) prefix sum の計算の場合：A^i に対し，$A^i_j = 1$ $(0 \leq j < N)$ である．
2) bubble sort による昇順への整列の場合：
type S: 各 A^i は $A^i_{j-1} > A^i_j$ $(1 \leq j < N)$ を満たす．
type D: 各 A^i は i が偶数ならば，$A^i_{j-1} > A^i_j$ を満たし，奇数ならば，$A^i_{j-1} \leq A^i_j$ $(1 \leq j < N)$ を満たす．(つまり，i が偶数なる A^i に対し，bubble sort の実行において入れ換え操作を実行し，奇数では実行しない．)
3) bitonic sort および quick sort による昇順への整列：各 A^i に対し，A^i_j $(0 \leq j < N)$ はランダム発生させた数値である．

TABLE III. COMPARISON OF GPU AND CPU TIME (MS) FOR EXECUTING BUBBLE SORT, WHERE EACH INPUT IS THE INT DATA TYPE WITHE TYPE D. AND RATIO REPRESENTS GPU TIME OF ROW-WISE/GPU TIME OF COLUMN-WISE.

$T \times B = L$	GPU		Ratio	CPU
	Row-wise	Column-wise	(R/C)	
$64 \times 1 = 64$	1,311.014	128.398	10.21	48.718
$64 \times 2 = 128$	1,311.015	128.856	10.17	92.926
$64 \times 4 = 256$	1,311.006	175.699	7.46	175.550
$64 \times 8 = 512$	1,310.908	188.570	6.95	353.238
$64 \times 16 = 1024$	1,313.293	195.281	6.73	697.498
$64 \times 32 = 2048$	1,378.023	204.841	6.73	1,392.545
$64 \times 64 = 4096$	3,824.223	233.160	16.40	2,782.425
$64 \times 128 = 8192$	11,243.516	307.050	36.62	5,567.844
$64 \times 256 = 16384$	22,355.297	618.945	36.12	11,134.103
$64 \times 512 = 32768$	44,588.641	1,236.623	36.06	22,261.717
$64 \times 1024 = 65536$	88,917.430	2,476.595	35.90	44,535.740

TABLE IV. COMPARISON OF GPU AND CPU TIME (MS) FOR EXECUTING BITONIC SORT, WHERE EACH INPUT IS THE INT DATA TYPE. AND RATIO REPRESENTS GPU TIME OF ROW-WISE/GPU TIME OF COLUMN-WISE.

$T \times B = L$	GPU		Ratio	CPU
	Row-wise	Column-wise	(R/C)	
$64 \times 1 = 64$	54.637	12.705	4.30	4.827
$64 \times 2 = 128$	55.160	14.427	3.82	7.875
$64 \times 4 = 256$	55.364	16.378	3.38	15.840
$64 \times 8 = 512$	55.502	16.409	3.38	31.836
$64 \times 16 = 1024$	56.875	16.586	3.43	63.534
$64 \times 32 = 2048$	81.883	17.416	4.70	127.422
$64 \times 64 = 4096$	219.930	19.401	11.34	254.714
$64 \times 128 = 8192$	532.955	29.836	17.86	508.033
$64 \times 256 = 16384$	1046.493	60.429	17.32	1015.430
$64 \times 512 = 32768$	2093.193	118.326	17.69	2034.253
$64 \times 1024 = 65536$	4173.395	236.217	17.67	4065.493

TABLE V. COMPARISON OF GPU TIME (MS) FOR CALCULATING ROW-WISE PREFIX SUM, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	1.257	1.254	1.404
$64 \times 2 = 128$	1.259	1.261	1.405
$64 \times 4 = 256$	1.262	1.264	1.407
$64 \times 8 = 512$	1.265	1.264	1.477
$64 \times 16 = 1024$	1.712	1.797	1.826
$64 \times 32 = 2048$	3.694	3.492	2.956
$64 \times 64 = 4096$	9.609	9.158	8.686
$64 \times 128 = 8192$	24.520	25.015	22.212
$64 \times 256 = 16384$	42.252	38.947	40.123
$64 \times 512 = 32768$	74.270	75.356	75.358
$64 \times 1024 = 65536$	138.266	137.412	132.858

C. 実験結果

まず，prefix sum の計算における Row-wise prefix sum, Column-wise prefix sum と CPU 実装の計算時間 (ms) の比較結果を，Table I に示す．bubble sort による整列における Row-wise bubble sort と Column-wise prefix sum と CPU 実装の計算時間 (ms) の比較結果を，Table II, III に示す．bitonic sort による整列における Row-wise bubble sort と Column-wise prefix sum と CPU 実装の計算時間 (ms) の比較結果を，Table IV に示す．

coalesced access の影響が大きく，いずれの操作 P においても，Row-wise 操作 P に比べて Column-wise 操作 P は高速な時間で解を出力している．

また，Table XVII に，quick sort の CPU 実装と bitonic sort の GPU 実装および CPU 実装の実行時間を示す．

また，row-wise 操作 P と同じ実行結果を得る方法として，入力配列を転置し，column-wise 操作 P を実行後，さらに転置するものも考えられる．そこで，各操作について，実

TABLE VI. COMPARISON OF GPU TIME (MS) FOR CALCULATING COLUMN-WISE PREFIX SUM, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	0.073	0.073	0.233
$64 \times 2 = 128$	0.108	0.108	0.232
$64 \times 4 = 256$	0.108	0.108	0.232
$64 \times 8 = 512$	0.112	0.112	0.234
$64 \times 16 = 1024$	0.121	0.119	0.249
$64 \times 32 = 2048$	0.141	0.141	0.294
$64 \times 64 = 4096$	0.231	0.231	0.444
$64 \times 128 = 8192$	0.527	0.512	0.963
$64 \times 256 = 16384$	1.118	1.102	2.008
$64 \times 512 = 32768$	2.256	2.230	4.008
$64 \times 1024 = 65536$	4.581	4.551	8.319

TABLE VII. COMPARISON OF GPU TIME (MS) FOR EXECUTING ROW-WISE BUBBLE SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY WITH TYPE S.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	1719.475	1719.345	1692.204
$64 \times 2 = 128$	1719.473	1719.351	1692.202
$64 \times 4 = 256$	1719.471	1719.733	1692.322
$64 \times 8 = 512$	1720.505	1721.328	1692.270
$64 \times 16 = 1024$	1724.047	1725.173	1694.113
$64 \times 32 = 2048$	2034.586	2037.865	2081.349
$64 \times 64 = 4096$	6463.130	6554.951	6553.022
$64 \times 128 = 8192$	17945.799	17884.131	16923.510
$64 \times 256 = 16384$	35588.621	35600.543	33750.898
$64 \times 512 = 32768$	71059.414	70946.758	67512.336
$64 \times 1024 = 65536$	141566.953	141293.203	135075.234

TABLE VIII. COMPARISON OF GPU TIME (MS) FOR EXECUTING COLUMN-WISE BUBBLE SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY WITH TYPE S.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	132.342	133.463	162.538
$64 \times 2 = 128$	129.744	127.167	205.120
$64 \times 4 = 256$	175.638	174.930	217.445
$64 \times 8 = 512$	188.563	188.341	221.561
$64 \times 16 = 1024$	195.293	195.087	228.728
$64 \times 32 = 2048$	208.852	204.795	256.631
$64 \times 64 = 4096$	235.995	233.073	314.623
$64 \times 128 = 8192$	306.639	306.982	594.058
$64 \times 256 = 16384$	618.229	618.540	1178.075
$64 \times 512 = 32768$	1236.398	1236.820	2392.119
$64 \times 1024 = 65536$	2476.417	2477.715	4726.197

験した結果を Table XIII–XVI に記す.

IV. まとめ

本稿では，L 本の配列 A^i $(0 \leq i < L, |A^i| = N)$ が与えられたとき，各 A^i に対し次の操作を GPU の 1 thread で実行する GPU プログラムを作成し，その実行時間を比較した.

1) prefix sum を計算.
2) Bubble sort によって昇順に整列.
3) Bitonic sort によって昇順に整列.

その結果，prefix sum の計算の場合，Column-wise は Row-wise に比べて最大で 46.53 倍の高速化となった．同様に，Bubble sort および Bitonic sort による整列の場合，それぞれ最大で 58.52 倍および 17.86 倍の高速化となった.

REFERENCES

[1] W. W. Hwu, *GPU Computing Gems Emerald Edition*. Morgan Kaufmann, 2011.

[2] NVIDIA Corporation, “NVIDIA CUDA C programming guide version 5.0,” 2012.

TABLE IX. COMPARISON OF GPU TIME (MS) FOR EXECUTING ROW-WISE BUBBLE SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY WITH TYPE D.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	1311.014	1310.844	1283.557
$64 \times 2 = 128$	1311.015	1310.848	1283.558
$64 \times 4 = 256$	1311.006	1310.860	1283.586
$64 \times 8 = 512$	1310.908	1311.294	1283.674
$64 \times 16 = 1024$	1313.293	1314.785	1285.251
$64 \times 32 = 2048$	1378.023	1387.924	1354.350
$64 \times 64 = 4096$	3824.223	3829.140	3860.494
$64 \times 128 = 8192$	11243.516	11200.067	10993.290
$64 \times 256 = 16384$	22355.297	22288.547	21946.367
$64 \times 512 = 32768$	44588.641	44525.281	43878.613
$64 \times 1024 = 65536$	88917.430	88698.133	87762.063

TABLE X. COMPARISON OF GPU TIME (MS) FOR EXECUTING COLUMN-WISE BUBBLE SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY WITH TYPE D.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	128.398	128.373	159.144
$64 \times 2 = 128$	128.856	128.754	205.097
$64 \times 4 = 256$	175.699	175.612	217.460
$64 \times 8 = 512$	188.570	188.544	221.563
$64 \times 16 = 1024$	195.281	195.124	228.690
$64 \times 32 = 2048$	204.841	204.836	256.574
$64 \times 64 = 4096$	233.160	233.153	314.627
$64 \times 128 = 8192$	307.050	306.813	596.615
$64 \times 256 = 16384$	618.945	619.438	1185.501
$64 \times 512 = 32768$	1236.623	1236.644	2400.929
$64 \times 1024 = 65536$	2476.595	2476.131	4746.782

TABLE XI. COMPARISON OF GPU TIME (MS) FOR EXECUTING ROW-WISE BITONIC SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	54.637	54.638	48.754
$64 \times 2 = 128$	55.160	55.154	49.327
$64 \times 4 = 256$	55.364	55.365	49.413
$64 \times 8 = 512$	55.502	55.516	49.554
$64 \times 16 = 1024$	56.875	56.810	51.215
$64 \times 32 = 2048$	81.883	82.500	85.290
$64 \times 64 = 4096$	219.930	221.742	222.284
$64 \times 128 = 8192$	532.955	532.960	465.082
$64 \times 256 = 16384$	1046.493	1055.633	931.229
$64 \times 512 = 32768$	2093.193	2078.727	1823.677
$64 \times 1024 = 65536$	4173.395	4137.641	3646.143

[3] ——, “NVIDIA CUDA C programming guide version 4.2,” 2012.

TABLE XII. COMPARISON OF GPU TIME (MS) FOR EXECUTING COLUMN-WISE BITONIC SORT, INPUTTING THE INT DATA TYPE, THE FLOAT DATA TYPE AND THE DOUBLE DATA TYPE ARRAY.

$T \times B = L$	int data type	float data type	double data type
$64 \times 1 = 64$	12.705	12.702	15.182
$64 \times 2 = 128$	14.427	12.919	17.894
$64 \times 4 = 256$	16.378	16.394	17.903
$64 \times 8 = 512$	16.409	16.413	17.941
$64 \times 16 = 1024$	16.586	16.585	18.170
$64 \times 32 = 2048$	17.416	17.433	19.535
$64 \times 64 = 4096$	19.401	19.385	23.135
$64 \times 128 = 8192$	29.836	29.749	41.649
$64 \times 256 = 16384$	60.429	60.130	84.554
$64 \times 512 = 32768$	118.326	120.020	168.364
$64 \times 1024 = 65536$	236.217	236.211	355.753

TABLE XIII. COMPARISON OF GPU TIMES (MS) OF ROW-WISE AND COLUMN-WISE PREFIX SUM WITH TRANSPOSING, WHERE EACH INPUT IS THE INT DATA TYPE. NOTE THAT EACH GPU TIME OF (B) IS THE SUM OF THAT OF COLUMN-WISE PREFIX SUM AND TWICE GPU TIME OF TRANSPOSING.

$T \times B = L$	(A) Row-wise	(B)	Column-wise	Transpose
$64 \times 1 = 64$	1.257	0.089	0.073	0.008
$64 \times 2 = 128$	1.259	0.140	0.108	0.016
$64 \times 4 = 256$	1.262	0.164	0.108	0.028
$64 \times 8 = 512$	1.265	0.222	0.112	0.055
$64 \times 16 = 1024$	1.712	0.291	0.121	0.085
$64 \times 32 = 2048$	3.694	0.505	0.141	0.182
$64 \times 64 = 4096$	9.609	0.915	0.231	0.342
$64 \times 128 = 8192$	24.520	1.905	0.527	0.689
$64 \times 256 = 16384$	42.252	3.976	1.118	1.429
$64 \times 512 = 32768$	74.270	7.864	2.256	2.804
$64 \times 1024 = 65536$	138.266	15.547	4.581	5.483

TABLE XIV. COMPARISON OF GPU TIMES (MS) OF ROW-WISE AND COLUMN-WISE BUBBLE SORT WITH TRANSPOSING, WHERE EACH INPUT IS THE INT DATA TYPE WITH TYPE S. NOTE THAT EACH GPU TIME OF (B) IS THE SUM OF THAT OF COLUMN-WISE PREFIX SUM AND TWICE GPU TIME OF TRANSPOSING.

$T \times B = L$	(A) Row-wise	(B)	Column-wise	Transpose
$64 \times 1 = 64$	1719.475	132.358	132.342	0.008
$64 \times 2 = 128$	1719.473	129.776	129.744	0.016
$64 \times 4 = 256$	1719.471	175.694	175.638	0.028
$64 \times 8 = 512$	1720.505	188.673	188.563	0.055
$64 \times 16 = 1024$	1724.047	195.463	195.293	0.085
$64 \times 32 = 2048$	2034.586	209.216	208.852	0.182
$64 \times 64 = 4096$	6463.130	236.679	235.995	0.342
$64 \times 128 = 8192$	17945.799	308.017	306.639	0.689
$64 \times 256 = 16384$	35588.621	621.087	618.229	1.429
$64 \times 512 = 32768$	71059.414	1242.006	1236.398	2.804
$64 \times 1024 = 65536$	141566.953	2487.383	2476.417	5.483

TABLE XV. COMPARISON OF GPU TIMES (MS) OF ROW-WISE AND COLUMN-WISE BUBBLE SORT WITH TRANSPOSING, WHERE EACH INPUT IS THE INT DATA TYPE WITH TYPE D. NOTE THAT EACH GPU TIME OF (B) IS THE SUM OF THAT OF COLUMN-WISE PREFIX SUM AND TWICE GPU TIME OF TRANSPOSING.

$T \times B = L$	(A) Row-wise	(B)	Column-wise	Transpose
$64 \times 1 = 64$	1311.014	128.414	128.398	0.008
$64 \times 2 = 128$	1311.015	128.888	128.856	0.016
$64 \times 4 = 256$	1311.006	175.755	175.699	0.028
$64 \times 8 = 512$	1310.908	188.680	188.570	0.055
$64 \times 16 = 1024$	1313.293	195.451	195.281	0.085
$64 \times 32 = 2048$	1378.023	205.205	204.841	0.182
$64 \times 64 = 4096$	3824.223	233.844	233.160	0.342
$64 \times 128 = 8192$	11243.516	308.428	307.050	0.689
$64 \times 256 = 16384$	22355.297	621.803	618.945	1.429
$64 \times 512 = 32768$	44588.641	1242.231	1236.623	2.804
$64 \times 1024 = 65536$	88917.430	2487.561	2476.595	5.483

TABLE XVI. COMPARISON OF GPU TIMES (MS) OF ROW-WISE AND COLUMN-WISE BITONIC SORT WITH TRANSPOSING, WHERE EACH INPUT IS THE INT DATA TYPE. NOTE THAT EACH GPU TIME OF (B) IS THE SUM OF THAT OF COLUMN-WISE PREFIX SUM AND TWICE GPU TIME OF TRANSPOSING.

$T \times B = L$	(A) Row-wise	(B)	Column-wise	Transpose
$64 \times 1 = 64$	57.902	12.675	12.659	0.008
$64 \times 2 = 128$	55.850	14.470	14.438	0.016
$64 \times 4 = 256$	56.085	16.447	16.391	0.028
$64 \times 8 = 512$	56.288	16.509	16.399	0.055
$64 \times 16 = 1024$	57.404	16.679	16.509	0.085
$64 \times 32 = 2048$	85.609	17.465	17.101	0.182
$64 \times 64 = 4096$	243.867	18.549	17.865	0.342
$64 \times 128 = 8192$	555.985	25.436	24.058	0.689
$64 \times 256 = 16384$	1098.188	51.392	48.534	1.429
$64 \times 512 = 32768$	2169.079	103.530	97.922	2.804
$64 \times 1024 = 65536$	4305.168	206.827	195.861	5.483

TABLE XVII. COMPARISON OF GPU TIMES (MS) AND CPU TIMES (MS) OF EXECUTING BITONIC SORT AND CPU TIMES OF EXECUTING QUICK SORT. NOTE THAT EACH INPUT IS THE INT DATA TYPE.

	GPU		CPU	
$T \times B = L$	Row-wise	Column-wise	bitonic sort	quick sort
$64 \times 1 = 64$	54.637	12.705	4.827	1.459
$64 \times 2 = 128$	55.160	14.427	7.875	3.470
$64 \times 4 = 256$	55.364	16.378	15.840	6.535
$64 \times 8 = 512$	55.502	16.409	31.836	9.351
$64 \times 16 = 1024$	56.875	16.586	63.534	17.243
$64 \times 32 = 2048$	81.883	17.416	127.422	28.945
$64 \times 64 = 4096$	219.930	19.401	254.714	51.304
$64 \times 128 = 8192$	532.955	29.836	508.033	99.965
$64 \times 256 = 16384$	1046.493	60.429	1015.430	193.383
$64 \times 512 = 32768$	2093.193	118.326	2034.253	386.760
$64 \times 1024 = 65536$	4173.395	236.217	4065.493	776.619