[CSED490C] Assignment Report: Lab3_cuda

- Student Id : 20220848
- Name : 선민수

1. Answering follwing questions

Q: How many bytes of data (both read and write) are moved from host to device when using CUDA Streams?

A: Total bytes of data moved from host to device (or vice versa) is `2 * inputLength * sizeof(float)`. Using CUDA Streams does not decrease the total bytes of data moved between host and device, but hide latency.

Q: When should one use pinned memory? Explain.

A: Pinned memory can be used when hiding latency of moving data between host and device. In the case which do not have to hide latency, pinned memory is not necessary.

2. Template.cu

```
#include <gputk.h>

#define N_STREAM 32
#define THREADS_PER_BLOCK 128

__global__ void vecAdd(float *in1, float *in2, float *out, int len) {
  int index = threadIdx.x + blockIdx.x * blockDim.x;
  if (index < len) {
    out[index] = in1[index] + in2[index];
  }
}

int main(int argc, char **argv) {
  gpuTKArg_t args;
  int inputLength;
  float *hostInput1;
  float *hostInput2;
  float *hostOutput;
  float *deviceInput1;
  float *deviceInput2;
  float *deviceOutput;
  unsigned int numStreams;

  args = gpuTKArg_read(argc, argv);

  gpuTKTime_start(Generic, "Importing data and creating memory on host");
  hostInput1 =
      (float *)gpuTKImport(gpuTKArg_getInputFile(args, 0), &inputLength);
  hostInput2 =
      (float *)gpuTKImport(gpuTKArg_getInputFile(args, 1), &inputLength);
  hostOutput = (float *)malloc(inputLength * sizeof(float));
  gpuTKTime_stop(Generic, "Importing data and creating memory on host");

  gpuTKLog(TRACE, "The input length is ", inputLength);

  gpuTKTime_start(GPU, "Allocating Pinned memory.");

  //@@ Allocate GPU memory here using pinned memory here
  cudaHostRegister(hostInput1, inputLength * sizeof(float), cudaHostRegisterDefault);
  cudaHostRegister(hostInput2, inputLength * sizeof(float), cudaHostRegisterDefault);
  cudaHostRegister(hostOutput, inputLength * sizeof(float), cudaHostRegisterDefault);
```

```
cudaMalloc((void **)&deviceInput1, inputLength * sizeof(float));
cudaMalloc((void **)&deviceInput2, inputLength * sizeof(float));
cudaMalloc((void **)&deviceOutput, inputLength * sizeof(float));

//@@ Create and setup streams
//@@ Calculate data segment size of input data processed by each stream
cudaStream_t *streams;
numStreams = N_STREAM;
streams = (cudaStream_t *)malloc(numStreams * sizeof(cudaStream_t));

for (int i = 0; i < numStreams; i++) cudaStreamCreate(streams + i);

gpuTKTime_start(Compute, "Performing CUDA computation");
//@@ Perform parallel vector addition with different streams.
size_t segmentSize = (inputLength - 1) / numStreams + 1;
for (unsigned int s = 0; s<numStreams; s++){
        //@@ Asynchronous copy data to the device memory in segments
        //@@ Calculate starting and ending indices for per-stream data
        size_t p = s * segmentSize;
        size_t len = (s == numStreams - 1) ? inputLength - p : segmentSize;
        cudaMemcpyAsync(deviceInput1 + p, hostInput1 + p, len * sizeof(float), cudaMe
        cudaMemcpyAsync(deviceInput2 + p, hostInput2 + p, len * sizeof(float), cudaMe

        //@@ Invoke CUDA Kernel
        //@@ Determine grid and thread block sizes (consider ococupancy)
        dim3 gridDim_((len - 1) / THREADS_PER_BLOCK + 1, 1, 1);
        dim3 blockDim_(THREADS_PER_BLOCK, 1, 1);
        vecAdd<<<gridDim_, blockDim_, 0, streams[s]>>>(deviceInput1 + p, deviceInput2

        //@@ Asynchronous copy data from the device memory in segments
        cudaMemcpyAsync(hostOutput + p, deviceOutput + p, len * sizeof(float), cudaMe
}

//@@ Synchronize
cudaDeviceSynchronize();

gpuTKTime_stop(Compute, "Performing CUDA computation");


gpuTKTime_start(GPU, "Freeing Pinned Memory");
//@@ Destory cudaStream
for (int i = 0; i < numStreams; i++) cudaStreamDestroy(streams[i]);
```

```
  //@@ Free the GPU memory here
  cudaFree(deviceInput1);
  cudaFree(deviceInput2);
  cudaFree(deviceOutput);

  gpuTKTime_stop(GPU, "Freeing Pinned Memory");

  gpuTKSolution(args, hostOutput, inputLength);

  free(hostInput1);
  free(hostInput2);
  free(hostOutput);

  return 0;
}
```

3. Execution times

Execution Systems

All compilation and the executions are made on docker container.

TITANXP

```
srun -p titanxp -N 1 -n 6 -t 02:00:00 --gres=gpu:1 --pty /bin/bash -l
```

- Cluster : `cse-cluster1.postech.ac.kr`
- Docker Image : `nvidia:cuda/12.0.1-devel-ubuntu22.04`
- Driver Version : `525.85.12`
- Cuda Version : `12.0`

Execution

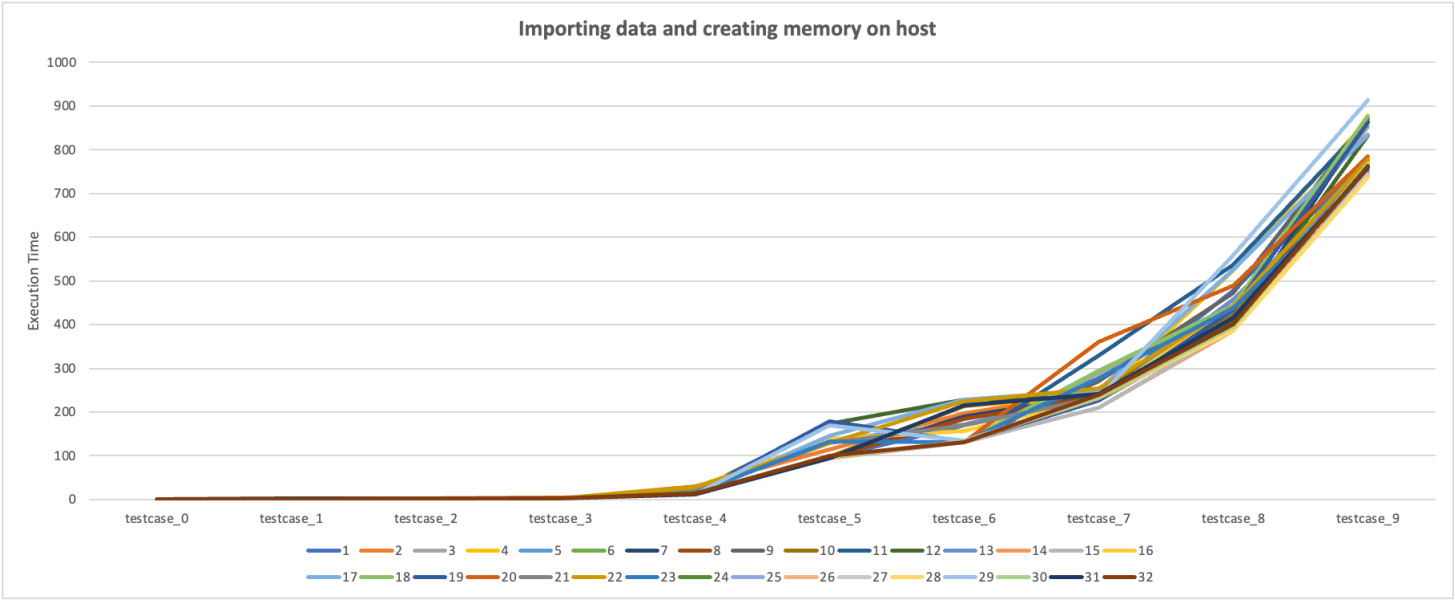
- All the time measurement unit is millisecond(ms).
- Single integer from index names or column names indicates the `N_STREAM` which indicates the number of stream used in that experiment.

Execution Script

```
base="/workspace/csed490c-01/Lab3_cuda"
for streamSize in {1..32}
do
    cd $base/sources
    sed -i "3c\#define N_STREAM $streamSize" template.cu
    make template
    echo > $base/result_$streamSize
    for idx in {0..9}
    do
        cd $base/sources/VectorAdd/Dataset/$idx
        ./../../../StreamVectorAdd_template -e output.raw -i input0.raw,input1.raw -o o
        echo >> $base/result_$streamSize
    done
done
```

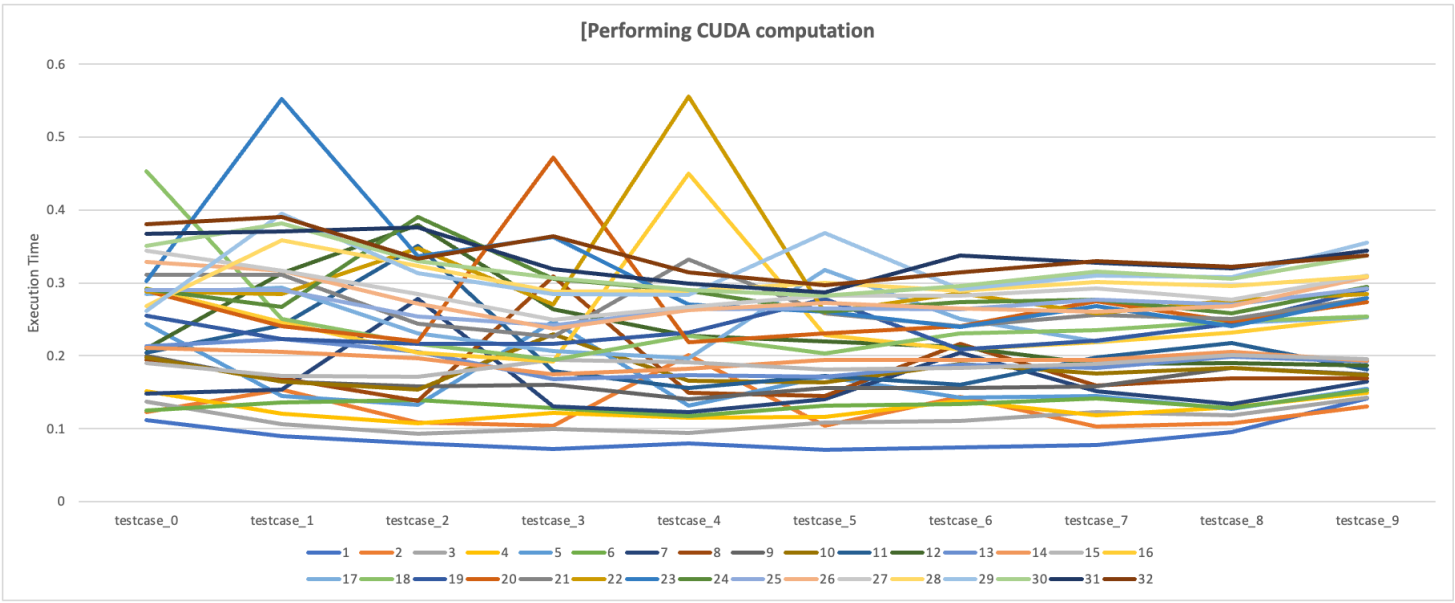
1 [Importing data and creating memory on host]

	testcase_0	testcase_1	testcase_2	testcase_3	testcase_4	testcase_5	testcase_6	testcase_7	testcase_8	testcase_9
1	1.11617	1.6003	1.81665	1.88427	12.7922	96.0677	170.291	236.191	477.374	777.516
2	1.26055	2.26477	2.27974	2.33833	29.3098	115.036	198.064	244.918	417.785	763.272
3	1.01436	1.47429	1.68426	1.80271	12.0203	98.069	131.156	289.026	444.775	853.236
4	1.05241	1.59132	1.71556	1.80329	12.3382	94.3232	214.542	236.429	523.792	869.354
5	1.20446	1.72032	1.89647	2.81838	13.3602	133.97	171.886	246.828	425.56	764.067
6	0.818011	1.33316	1.60124	1.76558	11.9745	94.1231	132.967	226.869	399.73	761.825
7	0.949974	1.61537	3.23355	1.93684	12.2942	94.2903	190.645	237.108	399.261	871.321
8	1.0381	1.56524	1.73023	3.65432	13.5665	95.4972	184.072	235.693	437.383	760.829
9	1.04966	1.61165	1.68515	1.97967	12.2906	94.1101	130.362	270.55	472.613	871.713
10	1.0347	1.50665	1.66402	2.65528	14.3294	98.0297	213.977	247.722	420.507	762.595
11	1.236	2.16593	3.45579	2.26868	12.802	95.5846	130.649	329.567	536.803	863.462
12	1.05178	2.3662	3.32196	2.84337	19.3144	173.853	227.621	235.202	406.804	831.634
13	1.02812	1.62533	1.76793	1.76634	12.0477	94.9838	131.788	226.074	457.204	752.103
14	1.00268	1.60959	1.76026	1.82101	13.1481	96.579	131.465	210.491	386.724	740.753
15	0.915411	1.34266	1.59135	1.78078	12.0792	93.9821	131.811	210.094	390.016	737.923
16	1.06205	1.60661	1.71854	1.80208	28.5451	138.614	156.267	233.604	437.821	878.283
17	1.01579	1.88198	1.95792	1.89039	12.628	144.974	228.196	250.556	526.291	834.124
18	1.3561	1.62106	1.79573	1.76605	12.025	94.3482	131.221	294.245	443.732	876.428
19	0.936151	1.4489	1.68019	1.88721	13.1153	178.051	133.585	227.309	434.724	861.039
20	1.23276	1.70221	1.85759	3.90486	13.0369	94.648	131.186	360.813	488.337	784.261
21	1.04076	1.86922	1.81966	1.87379	19.6134	128.833	170.181	234.762	397.081	774.94
22	1.00929	1.55461	2.38825	2.25812	29.1762	128.552	225.536	253.886	441.385	777.249
23	1.01892	2.61078	2.50755	2.7529	14.806	132.053	131.836	277.854	437.817	740.99
24	0.990666	1.50006	2.49171	2.34494	15.9645	96.6039	131.759	235.088	389.597	744.609
25	0.949871	1.60686	1.73355	1.79037	12.0517	94.6737	131.201	236.942	385.913	739.769
26	1.03543	1.63162	1.82786	1.77559	12.0857	95.0153	132.148	232.844	386.086	743.595
27	1.0858	1.5724	1.8489	1.8417	12.7755	98.8067	133.205	237.619	386.623	741.211
28	0.911865	1.91066	2.14546	2.00552	12.7633	95.4907	132.14	235.149	386.411	736.493
29	1.05146	2.16661	2.07808	2.0322	12.2182	169.721	135.64	233.074	558.709	913.725
30	1.03063	1.86273	2.05034	1.93852	12.3234	96.0368	132.359	233.565	395.409	768.906
31	1.04428	1.91442	2.05423	1.93592	12.6692	94.814	216.355	241.312	415.23	758.277
32	1.04987	1.63699	1.81867	2.20526	13.378	100.21	131.503	239.203	400.837	763.25



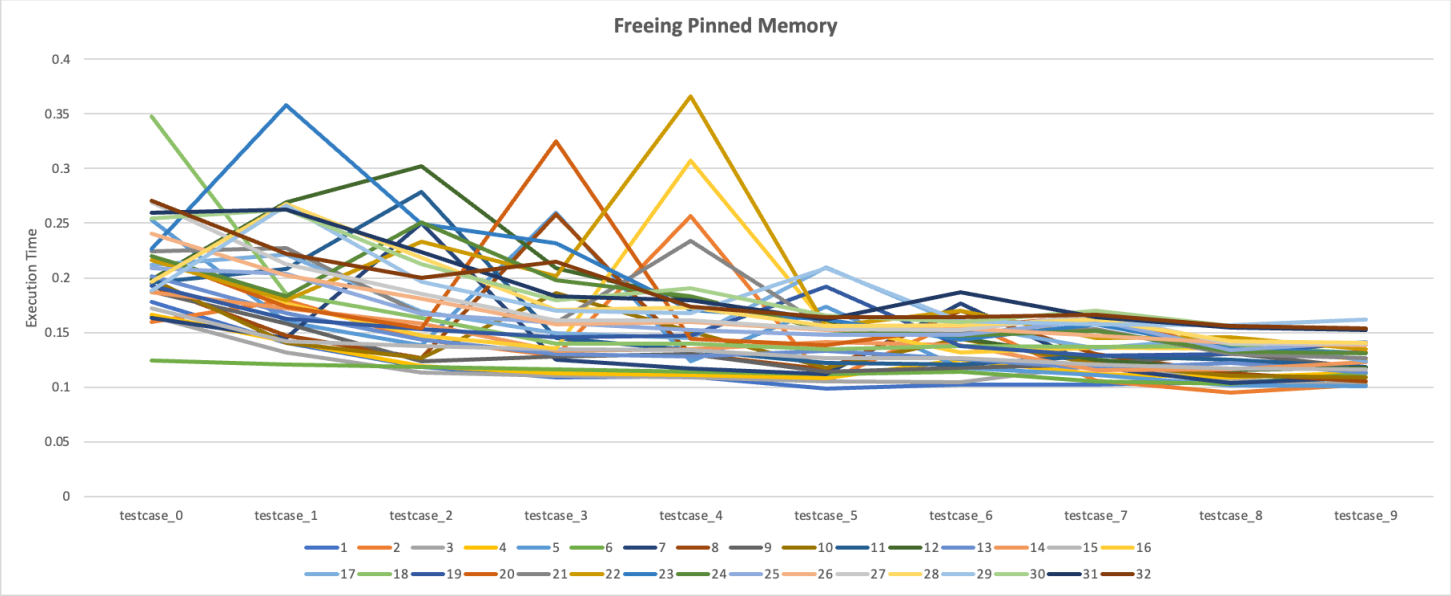
2 [Performing CUDA computation]

	testcase_0	testcase_1	testcase_2	testcase_3	testcase_4	testcase_5	testcase_6	testcase_7	testcase_8	testcase_9
1	0.111652	0.090156	0.079725	0.072564	0.079837	0.070708	0.074067	0.077797	0.095576	0.141377
2	0.123178	0.153566	0.108482	0.104374	0.20073	0.103845	0.143755	0.102829	0.106821	0.130864
3	0.136909	0.106539	0.092712	0.099161	0.093554	0.108407	0.111156	0.122792	0.118167	0.142956
4	0.151132	0.12077	0.107466	0.121611	0.115432	0.116196	0.137609	0.118789	0.129412	0.149066
5	0.243675	0.14444	0.133165	0.247096	0.131124	0.170699	0.142017	0.144876	0.127047	0.154958
6	0.125278	0.13579	0.139165	0.127751	0.117323	0.132056	0.134291	0.141579	0.12803	0.153006
7	0.148589	0.153367	0.277691	0.130456	0.122824	0.140729	0.204148	0.151244	0.133575	0.164623
8	0.194572	0.169013	0.138239	0.309131	0.149156	0.144915	0.21583	0.159137	0.168597	0.168691
9	0.199973	0.164596	0.158159	0.16058	0.140811	0.155969	0.155553	0.158439	0.183132	0.174861
10	0.197161	0.165128	0.153972	0.229951	0.165906	0.163999	0.189034	0.175111	0.182871	0.173816
11	0.203767	0.241693	0.350854	0.179089	0.155844	0.172572	0.159952	0.197885	0.217612	0.181247
12	0.20995	0.313474	0.379848	0.263893	0.227384	0.21986	0.211711	0.188539	0.190166	0.187003
13	0.213292	0.22254	0.204872	0.168018	0.17373	0.170727	0.187795	0.183296	0.198629	0.191957
14	0.211059	0.205623	0.196853	0.174917	0.182069	0.193923	0.194008	0.19381	0.205274	0.192942
15	0.19045	0.172223	0.171568	0.194346	0.191145	0.180837	0.183387	0.188474	0.200981	0.195906
16	0.292505	0.245829	0.204053	0.192196	0.449995	0.228723	0.20917	0.218949	0.231446	0.253148
17	0.284349	0.293966	0.23061	0.206539	0.196803	0.318075	0.25055	0.219951	0.245503	0.252667
18	0.45366	0.250171	0.216066	0.194124	0.226841	0.203545	0.230486	0.235158	0.247404	0.254338
19	0.254436	0.22278	0.216216	0.216725	0.231648	0.278449	0.20829	0.221246	0.244184	0.291367
20	0.288564	0.240958	0.219451	0.472231	0.218882	0.231138	0.241036	0.275094	0.247532	0.273345
21	0.310957	0.31116	0.243375	0.225872	0.331809	0.257701	0.24079	0.255482	0.250665	0.27952
22	0.287894	0.28457	0.347835	0.270374	0.555173	0.258819	0.285827	0.258292	0.275854	0.284572
23	0.302001	0.551959	0.336476	0.362566	0.27063	0.260708	0.239389	0.26776	0.240668	0.278924
24	0.290824	0.266867	0.390204	0.306002	0.289585	0.260009	0.273449	0.27748	0.258661	0.29485
25	0.290445	0.29014	0.253788	0.241498	0.263374	0.264433	0.2638	0.277461	0.270522	0.291839
26	0.328729	0.316144	0.27184	0.237367	0.263002	0.272838	0.265184	0.260426	0.268531	0.307445
27	0.344394	0.316165	0.284253	0.249445	0.267054	0.282823	0.282015	0.292012	0.277009	0.31012
28	0.267653	0.357939	0.322745	0.287431	0.288939	0.299954	0.289173	0.300858	0.295778	0.309226
29	0.261939	0.395066	0.313578	0.284114	0.283873	0.368393	0.291652	0.309851	0.308104	0.355077
30	0.351005	0.381851	0.329869	0.30709	0.290301	0.282023	0.296168	0.315729	0.305777	0.337378
31	0.367452	0.371025	0.375687	0.318271	0.29928	0.286328	0.337048	0.327456	0.319471	0.344096
32	0.380323	0.390856	0.33337	0.363492	0.314055	0.29622	0.314312	0.329802	0.321984	0.336968



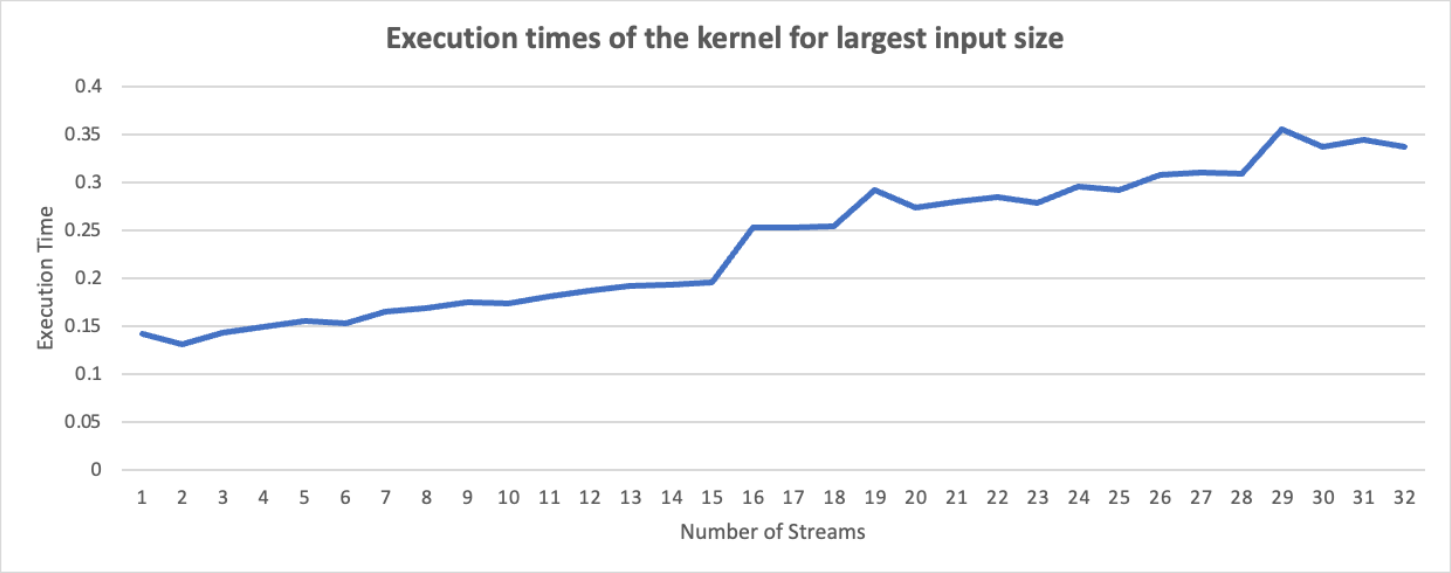
3 [Freeing Pinned Memory]

	testcase_0	testcase_1	testcase_2	testcase_3	testcase_4	testcase_5	testcase_6	testcase_7	testcase_8	testcase_9
1	0.177809	0.140458	0.118449	0.108643	0.109601	0.098995	0.102412	0.102426	0.104563	0.102649
2	0.159553	0.179614	0.143418	0.129064	0.256278	0.105067	0.160956	0.105905	0.094804	0.102551
3	0.165122	0.131603	0.113049	0.110563	0.109263	0.105636	0.104852	0.119949	0.106901	0.102421
4	0.166102	0.142231	0.119446	0.112818	0.111254	0.107961	0.124842	0.112123	0.108711	0.113361
5	0.253023	0.159334	0.138395	0.259173	0.123621	0.173246	0.117505	0.11133	0.101911	0.100714
6	0.124765	0.120795	0.118624	0.116324	0.114476	0.111873	0.114209	0.10561	0.103397	0.112308
7	0.163073	0.144943	0.250098	0.125465	0.117196	0.111945	0.176814	0.118646	0.103541	0.109094
8	0.192874	0.147442	0.126868	0.258343	0.129933	0.116646	0.169835	0.130259	0.112437	0.105546
9	0.187424	0.157853	0.123962	0.127812	0.130353	0.114236	0.117512	0.121069	0.131346	0.117826
10	0.198667	0.140747	0.126606	0.186118	0.150532	0.118003	0.144066	0.121113	0.110393	0.109182
11	0.195542	0.20794	0.278838	0.145252	0.133843	0.122149	0.120694	0.128831	0.124878	0.118554
12	0.198902	0.26878	0.302061	0.208584	0.180185	0.16036	0.145184	0.125433	0.115799	0.117699
13	0.201875	0.16767	0.14332	0.129958	0.127927	0.132878	0.125867	0.116702	0.121999	0.113082
14	0.186937	0.17192	0.158084	0.133182	0.134641	0.141395	0.138603	0.115874	0.116018	0.122858
15	0.172389	0.141959	0.13733	0.134332	0.133927	0.126789	0.126586	0.120656	0.116929	0.116083
16	0.220121	0.176649	0.146867	0.135342	0.30722	0.157377	0.131784	0.135966	0.135066	0.124837
17	0.211848	0.221577	0.168913	0.149439	0.144804	0.209517	0.157152	0.135667	0.145747	0.123442
18	0.347917	0.18516	0.163031	0.139592	0.139911	0.134607	0.137265	0.137095	0.137775	0.130852
19	0.192971	0.162446	0.153411	0.145808	0.147021	0.192046	0.137721	0.128732	0.130028	0.141539
20	0.219976	0.173482	0.154069	0.324627	0.143914	0.138337	0.154894	0.164998	0.133404	0.131884
21	0.2242	0.227476	0.166783	0.158703	0.233603	0.154452	0.149607	0.152736	0.130105	0.126539
22	0.215889	0.17917	0.233375	0.201468	0.365907	0.155421	0.170014	0.144976	0.145675	0.13476
23	0.226538	0.357602	0.249261	0.231586	0.170454	0.161636	0.143866	0.157086	0.135252	0.131179
24	0.219919	0.183197	0.250524	0.197509	0.18298	0.152541	0.147474	0.151564	0.132785	0.131211
25	0.209174	0.203488	0.166882	0.15904	0.15241	0.147553	0.148213	0.161278	0.134155	0.141369
26	0.240631	0.202597	0.181103	0.157093	0.159648	0.152499	0.153542	0.147521	0.140281	0.138197
27	0.268697	0.212191	0.185527	0.161012	0.161179	0.152736	0.152435	0.16645	0.139986	0.140083
28	0.19604	0.267692	0.218607	0.170415	0.173198	0.156139	0.156716	0.161578	0.14228	0.140533
29	0.188617	0.266143	0.196467	0.169709	0.167714	0.208854	0.160159	0.157784	0.156486	0.161914
30	0.254512	0.262155	0.212175	0.179499	0.190355	0.165671	0.15999	0.170247	0.156342	0.153675
31	0.259361	0.26216	0.223743	0.182986	0.179441	0.161554	0.186902	0.163715	0.154164	0.152408
32	0.270848	0.222326	0.199914	0.214892	0.173843	0.164192	0.164393	0.166417	0.155635	0.153563



4 [Execution times of the kernel for largest input size]

	1	2	3	4	5	6	7	8	9	10
testcase_9	0.141377	0.130864	0.142956	0.149066	0.154958	0.153006	0.164623	0.168691	0.174861	0.173816
	11	12	13	14	15	16	17	18	19	20
testcase_9	0.181247	0.187003	0.191957	0.192942	0.195906	0.253148	0.252667	0.254338	0.291367	0.273345
	21	22	23	24	25	26	27	28	29	30
testcase_9	0.27952	0.284572	0.278924	0.29485	0.291839	0.307445	0.31012	0.309226	0.355077	0.337378
	31	32								
testcase_9	0.344096	0.336968								



Some comments of `Execution times of the kernel for largest input size`

: We can see the execution time of kernel increases as the number of streams increasing. It is not what we expected, which expected to decrease as the number of streams increasing. The main reason of why this happend is related to the occupancy. As the number of threads per block is set to `128` in the code, we cannot see clear effect of increasing the number of streams with such size of input. With enough size of input provided to the kerenl, the execution time is expected to decrease.