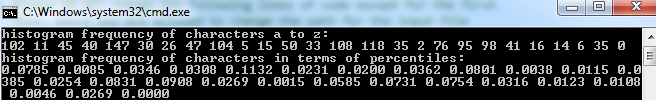
Final project : Histogram

1. Running the CPU version of the code :



b) Code after modification :

#include "J:\ami\ami\common\book.h"

#include <cuda\_runtime\_api.h>

#include "device\_launch\_parameters.h"

#include <stdio.h>

#include <cuda.h>

#include <stdlib.h>

#include <time.h>

#include "..\..\ami\ami\common\cpu\_bitmap.h"

#include<math.h>

#include "J:\ami\ami\common\cpu\_anim.h"

/\*

\*

\* Author: Brady Chen 4/12/2017

\* Modified By:

\* <Amikar Divij> <5/5/2017>

\*

\* This is a C code for the computation of a histogram of data from an input text file. The

\* text file contains multiple lines of characters. The code generate the frequency histogram

\* of characters from the input file.

\*/

/\*

\*

\* Definition of important values:

\* MAX\_TEXT\_LINES -- maximum number of lines for the input file. We initially set to 1000

\* If your file contains more than 1000, just change the value

\* MAX\_LINE\_LENGTH -- the length of each line. You change the value

\* NUMBER\_OF\_CHARS -- this the total number of characters used in the input file. We set

\* the number to 128 which could include all the standard characters in

\* the ASCII code table. If your input file contains more than 128 standard

\* characters, please increase the number. For example, you should use 256

\* for extended ASCII code.

\* TOTAL\_SIZE -- this is the total number of characters in the input file.

\*

\* NOTE: You can change the values of the varibales if necessary but please don't change the variable

\* names themselves

\*/

#define MAX\_TEXT\_LINES 4400

#define MAX\_LINE\_LENGTH 75

#define NUMBER\_OF\_CHARS 128

#define TOTAL\_SIZE (MAX\_TEXT\_LINES + 10)\*(MAX\_LINE\_LENGTH + 10)

/\*

\* Declaraion of the arrays of characters

\* aTextData -- this is a two dimensional array of char. It contains all the characters

\* inputted from the input file. The first index record the lines and the

\* second index record the columns in each line. We add 10 on MAX\_TEXT\_LINES

\* and MAX\_LINE\_LENGTH as safety spaces to crash on off-by-one errors.

\*

\* NOTE: No need to change this part

\*/

char aTextData[MAX\_TEXT\_LINES + 10][MAX\_LINE\_LENGTH + 10];

char aFlattenedData[(MAX\_TEXT\_LINES + 10)\*(MAX\_LINE\_LENGTH + 10)];

/\*

\*

\* Declaration of function

\* histogram() -- This function takes two arguments

\* aText -- a two dimensional array of char.

\* histo -- a one dimensional array which returns the frequency histogram

\* of the characrers in the array aText

\*/

\_\_global\_\_ void gpu\_histogram(unsigned char \*aText, unsigned int \*histo);

int main(void)

{

/\*

\* Declaration of variables

\* histo -- contains the frequency histogram of the characters

\* lineBuffer -- line buffer for reading

\* iTextSize -- how large is the array

\* iTextUsed -- how many lines contained

\*/

unsigned int histo[NUMBER\_OF\_CHARS];

char lineBuffer[MAX\_LINE\_LENGTH + 10];

int iTextSize = MAX\_TEXT\_LINES;

int iTextUsed = 0;

// clean the memory with zeros

memset(aTextData, 0, sizeof(aTextData));

memset(lineBuffer, 0, sizeof(lineBuffer));

/\*

\* Data read and conversion: The following lines of code do the following

\* 1. Find the input file (in this case "input.txt)

\* 2. Read the data from the input file and store it in a two dimensional

\* array of char

\*

\*

\*/

char \*inputFile = "J:/FSD/FSD/input\_test.txt"; // you may need to change the path depending on

// where you put the input file

// read input file into the array. use binary mode "rb"

FILE \*fin = fopen(inputFile, "rb");

if (!fin) {

printf("cannot read %s\n", inputFile);

return 1;

}

// while not end of file, read another line

while (fgets(lineBuffer, sizeof(lineBuffer)-10, fin) != 0)

{

// check if array has space for another line

if (iTextUsed >= iTextSize)

{

printf("overflow: too many text lines\n");

return 1;

}

// strip CR/LF from line endings so we get pure text

char \*psz = strchr(lineBuffer, '\r'); if (psz) \*psz = '\0';

psz = strchr(lineBuffer, '\n'); if (psz) \*psz = '\0';

// store the line

strncpy(aTextData[iTextUsed], lineBuffer, MAX\_LINE\_LENGTH);

aTextData[iTextUsed][MAX\_LINE\_LENGTH - 1] = '\0'; // safety

iTextUsed++;

}

fclose(fin);

/\* end of the Data read and conversion \*/

/\*

\* Function call to compute the frequency histogram of the data

\* 1. declare’s the blocks and threads

\* 2. call’s the implemented kernel function gpu\_histogram() .

\* Please note that the kernel function DOES NOT take array as argument,

\* not to mention two dimensional array in this case.

\*/

unsigned int \*histo1;

unsigned char \*aText;

cudaMalloc((void\*\*)&aText, TOTAL\_SIZE);

cudaMemcpy(aText, aTextData, TOTAL\_SIZE,

cudaMemcpyHostToDevice);

cudaMalloc((void\*\*)&histo1,

NUMBER\_OF\_CHARS \* sizeof(int));

cudaMemset(histo1, 0,

NUMBER\_OF\_CHARS \* sizeof(int));

// kernel launch - 2x the number of mps gave best timing

cudaDeviceProp prop;

cudaGetDeviceProperties(&prop, 0);

unsigned int blocks = prop.multiProcessorCount;

gpu\_histogram << <blocks \* 2, NUMBER\_OF\_CHARS >> >(aText, histo1);

cudaMemcpy(histo, histo1,

NUMBER\_OF\_CHARS \* sizeof(int),

cudaMemcpyDeviceToHost);

/\*

\*

\* Output of histogram results

\* the following is the output of the histogram frequency of characters in

\* the order of ASCII code. the array histo[] contains the frequency histogram

\* of 128 characters defined in ASCII.

\* 1. The decimal values for the letters are

\* A to Z -- 65 to 90

\* a to z -- 97 to 122

\* 2. the following lines of code print the histogram results for a to z, A to \* Z. and both lower and upper case a to z with their percentiles.

\*

\*

\*/

unsigned int histocount = 0; // total character count

printf("histogram frequency of characters a to z: \n");

for (int i = 97; i < 97 + 26; i++) {

histocount += histo[i];

printf("%d ", histo[i]);

}

printf("\n");

printf("histogram frequency of characters in terms of percentiles: \n");

for (int i = 97; i < 97 + 26; i++) {

printf("%.4f ", (float)histo[i] / (float)histocount);

}

printf("\n");

unsigned int histocount1 = 0; // total character count

printf("histogram frequency of characters A to Z: \n");

for (int i = 65; i < 65 + 26; i++) {

histocount1 += histo[i];

printf("%d ", histo[i]);

}

printf("\n");

printf("histogram frequency of characters in terms of percentiles for A to Z: \n");

for (int i = 65; i < 65 + 26; i++) {

printf("%.4f ", (float)histo[i] / (float)histocount1);

}

printf("\n");

unsigned int histocount2 = 0;

printf("histogram frequency of characters A to Z and a to z: \n");

for (int i = 0; i < 0 + 26; i++) {

histocount2 += histo[65 + i];

printf("%d ", histo[65+i]+histo[97+i]);

}

printf("\n");

unsigned int histocount3 = 0;

histocount3 = histocount + histocount1;

printf("histogram frequency of characters in terms of percentiles for A to Z and a to z: \n");

for (int i = 0; i < 0 + 26; i++) {

printf("%.4f ", ((float)histo[65 + i] + (float)histo[97 + i] ) / (float)histocount3);

}

/\* code above here for computing the frequency histogram for all letters

\* regardless low or up cases.

\*

\*/

cudaFree(histo);

cudaFree(aText);

cudaFree(aTextData);

return 0;

}

/\*

\*

\* Implementation of function

\* histogram() -- This function takes two pointer arguments

\* aText -- a two dimensional array of char.

\* histo -- a one dimensional array which returns the frequency histogram

\* of the characrers in the array aText

\*it used atomicadd to perform the increment function without causing any error due to \*different threads accessing the same variable

\*/

\_\_global\_\_ void gpu\_histogram(unsigned char \*aText, unsigned int \*histo)

{

int i = threadIdx.x + blockIdx.x \* blockDim.x;

int stride = blockDim.x \* gridDim.x;

while (i < TOTAL\_SIZE){

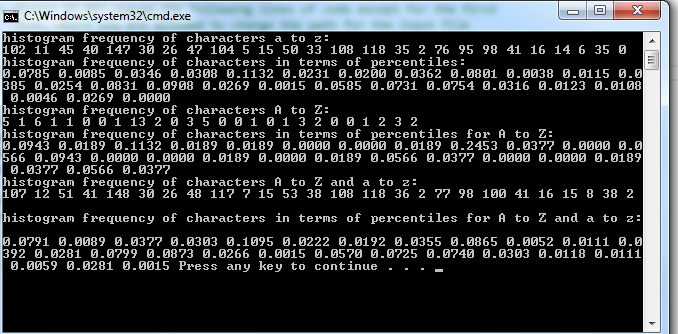
atomicAdd(&(histo[aText[i]]), 1);

i += stride;

}

}

c ) Result of GPU histogram on input\_test



d) Result of GPU histogram on input\_data

