**PP LAB WEEK-9**

# DSE VI-A2 Divansh Prasad 210968140

1) Write a program in CUDA to count the number of times a given word is repeated in a sentence.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SENTENCE\_LENGTH 1000

#define MAX\_WORD\_LENGTH 100

\_\_global\_\_ void countWordOccurrences(char \*sentence, char \*word, int \*count, int sentenceLength, int wordLength) {

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

int wordCount = 0;

if (tid < sentenceLength - wordLength + 1) {

int match = 1;

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

if (sentence[tid + i] != word[i]) {

match = 0;

break;

}

}

if (match) {

atomicAdd(count, 1);

}

}

}

int main() {

char sentence[MAX\_SENTENCE\_LENGTH];

char word[MAX\_WORD\_LENGTH];

int sentenceLength, wordLength;

printf("Enter a sentence: ");

fgets(sentence, MAX\_SENTENCE\_LENGTH, stdin);

sentenceLength = strlen(sentence);

if (sentence[sentenceLength - 1] == '\n') {

sentence[sentenceLength - 1] = '\0';

sentenceLength--;

}

printf("Enter the word to count: ");

scanf("%s", word);

wordLength = strlen(word);

char \*d\_sentence, \*d\_word;

int \*d\_count;

int count = 0;

cudaMalloc(&d\_sentence, sentenceLength \* sizeof(char));

cudaMalloc(&d\_word, wordLength \* sizeof(char));

cudaMalloc(&d\_count, sizeof(int));

cudaMemcpy(d\_sentence, sentence, sentenceLength \* sizeof(char), cudaMemcpyHostToDevice);

cudaMemcpy(d\_word, word, wordLength \* sizeof(char), cudaMemcpyHostToDevice);

cudaMemcpy(d\_count, &count, sizeof(int), cudaMemcpyHostToDevice);

int threadsPerBlock = 256;

int blocksPerGrid = (sentenceLength + threadsPerBlock - 1) / threadsPerBlock;

countWordOccurrences<<<blocksPerGrid, threadsPerBlock>>>(d\_sentence, d\_word, d\_count, sentenceLength, wordLength);

cudaMemcpy(&count, d\_count, sizeof(int), cudaMemcpyDeviceToHost);

printf("Number of occurrences of '%s' in the sentence: %d\n", word, count);

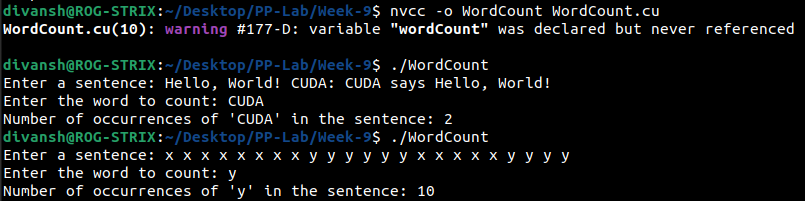
cudaFree(d\_sentence);

cudaFree(d\_word);

cudaFree(d\_count);

return 0;

}



2) Write a CUDA program that reads a string S and produces the string RS as follows:

Input String S: PCAP

Output String RS: PCAPPCAPPCP

Note: Each work item copies the required number of characters from S to RS.

#include <stdio.h>

#include <stdlib.h>

#define MAX\_STRING\_LENGTH 1000

\_\_global\_\_ void repeatString(char \*S, char \*RS, int length, int S\_length) {

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

if (tid < length) {

RS[tid] = S[tid % S\_length];

}

}

int main() {

char S[MAX\_STRING\_LENGTH];

char RS[MAX\_STRING\_LENGTH \* 3]; // Assuming the output string can be at most 3 times the length of the input string

int length;

printf("Enter a string: ");

fgets(S, MAX\_STRING\_LENGTH, stdin);

length = strlen(S);

if (S[length - 1] == '\n') {

S[length - 1] = '\0';

length--;

}

int S\_length = length; // Store the length of input string

length \*= 3; // Adjust length for output string

char \*d\_S, \*d\_RS;

cudaMalloc(&d\_S, S\_length \* sizeof(char));

cudaMalloc(&d\_RS, length \* sizeof(char));

cudaMemcpy(d\_S, S, S\_length \* sizeof(char), cudaMemcpyHostToDevice);

int threadsPerBlock = 256;

int blocksPerGrid = (length + threadsPerBlock - 1) / threadsPerBlock;

repeatString<<<blocksPerGrid, threadsPerBlock>>>(d\_S, d\_RS, length, S\_length);

cudaMemcpy(RS, d\_RS, length \* sizeof(char), cudaMemcpyDeviceToHost);

printf("Input String S: %s\n", S);

printf("Output String RS: %s\n", RS);

cudaFree(d\_S);

cudaFree(d\_RS);

return 0;

}

