GPU Assignment - 2

Report by Kaustav Vats (2016048)

Assigning a word from a file to each thread in the grid. Checking for the pattern with offset 0, 1 & 2 on the same thread. 4 bytes word are assigned row wise in the grid [Threads in First row get 4 bytes in sequence].

Naive approach to do pattern matching on GPU would be to directly read 4 bytes from global memory. Which is very costly since we are trying to match word with various offsets, so we also required next 4 bytes. This means that each thread is doing two read operations from global memory.

Since each thread is reading extra 4 bytes, which are also read by next thread in the sequence. So i solved this problem using shared memory.

I created a 1D shared memory in which each block do at least 1 read operation from global memory and some of the boundary thread do 2 read operations.

```
const dim3 block_size(512, 1);
const dim3 num_blocks(65536, 1);
__shared__ unsigned int sm_text[TOTAL+1]; // TOTAL = 512
```

In this Total is BlockDim.x, Right Boundary threads do Read operation twice.

Since the keywords used for pattern matching are used again and again by all threads, I created a 1D Shared Memory Array to store those elements.

```
__shared__ unsigned int sm_words[MAX_WORDS];
```

To increase the frequency of each word match, I'm doing atomic add to increment frequency for each keyword in global memory. Initially i was trying to do frequency increment in shared memory then in the end of each thread, transferring data from shared memory to global memory.

Note:-

Its mentioned on many of the blogs that atomicAdd is slower for shared memory as compared to global memory.

Timings\File Size	Small	Medium	Large
CPU Time	0.447937s	1.15207s	2.28262s
GPU Kernel Time	0.958176ms	1.91203ms	3.44141ms
GPU Kernel + Memory Transfer	2.41875ms	5.0512ms	9.14464ms
Speedup	467.489	602.535	663.28
Speedup with memory Transfer	185.193	228.078	249.612