GPU Assignment - 2

Report by Kaustav Vats (2016048)

Assign consecutive word(4 bytes) from the input file to each thread in the grid. Check 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 thread do at least 1 read operation from global memory and last thread in block does 2 read operations.

```
const dim3 block_size(32, 1);
const dim3 num_blocks(ceil(len/32), 1);
__shared__ unsigned int sm_text[TOTAL+1]; // TOTAL = 32
```

Total is equal to BlockDim.x, last thread does two read operation.

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. I also noticed that no of atomic add for a word doesn't affect total time taken by the kernel.

Timings\File Size	Small	Medium	Large
CPU Time	0.448603s	1.1442s	2.30286s
GPU Kernel Time	0.502656ms	1.25555ms	2.47811ms
GPU Kernel + Memory Transfer	1.86173ms	4.3671ms	8.29613ms
Speedup	879.639	911.311	929.279
Speedup with memory Transfer	237.498	262.004	277.582