Successfully read 1201748 bytes from the file.

CPU Results:

Found 0 occurrences of word: sword in 7 ms

Found 20 occurrences of word: fire in 7 ms

Found 52 occurrences of word: death in 6 ms

Found 75 occurrences of word: love in 6 ms

Found 12 occurrences of word: hate in 6 ms

Found 8016 occurrences of word: the in 7 ms

Found 481 occurrences of word: man in 6 ms

Found 203 occurrences of word: woman in 6 ms

GPU Results:

Processed token 'sword' with occurrences: 0 in 2.90579 ms

Processed token 'fire' with occurrences: 20 in 0.0256 ms

Processed token 'death' with occurrences: 48 in 0.022528 ms

Processed token 'love' with occurrences: 73 in 0.019456 ms

Processed token 'hate' with occurrences: 12 in 0.024512 ms

Error during kernel execution for token 'the': an illegal memory access was encountered

Error allocating device memory for token 'man': an illegal memory access was encountered

Error allocating device memory for token 'woman': an illegal memory access was encountered

C:\set10108-cw\set10108\labs\cw1\build\Debug\cw1.exe (process 13516) exited with code 0 (0x0).

To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .

Successfully read 1201748 bytes from the file.

CPU Results:

Found 0 occurrences of word: sword in 11 ms

Found 20 occurrences of word: fire in 10 ms

Found 52 occurrences of word: death in 10 ms

Found 75 occurrences of word: love in 11 ms

Found 12 occurrences of word: hate in 7 ms

Found 8016 occurrences of word: the in 7 ms

Found 481 occurrences of word: man in 6 ms

Found 203 occurrences of word: woman in 6 ms

GPU Results:

Processed token 'sword' with occurrences: 0 in 3.70893 ms

Processed token 'fire' with occurrences: 20 in 0.030464 ms

Processed token 'death' with occurrences: 51 in 0.026624 ms

Processed token 'love' with occurrences: 75 in 0.031744 ms

Processed token 'hate' with occurrences: 12 in 0.032768 ms

Error during kernel execution for token 'the': an illegal memory access was encountered

Error allocating device memory for token 'man': an illegal memory access was encountered

Error allocating device memory for token 'woman': an illegal memory access was encountered

C:\set10108-cw\set10108\labs\cw1\build\Debug\cw1.exe (process 23832) exited with code 0 (0x0).

To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .

Successfully read 1201748 bytes from the file.

CPU Results:

Found 0 occurrences of word: sword in 6 ms

Found 20 occurrences of word: fire in 6 ms

Found 52 occurrences of word: death in 6 ms

Found 75 occurrences of word: love in 6 ms

Found 12 occurrences of word: hate in 6 ms

Found 8016 occurrences of word: the in 8 ms

Found 481 occurrences of word: man in 10 ms

Found 203 occurrences of word: woman in 6 ms

GPU Results:

Processed token 'sword' with occurrences: 0 in 2.77872 ms

Processed token 'fire' with occurrences: 20 in 0.068608 ms

Processed token 'death' with occurrences: 51 in 0.029696 ms

Processed token 'love' with occurrences: 75 in 0.034816 ms

Processed token 'hate' with occurrences: 12 in 0.03584 ms

Error during kernel execution for token 'the': an illegal memory access was encountered

Error allocating device memory for token 'man': an illegal memory access was encountered

Error allocating device memory for token 'woman': an illegal memory access was encountered

C:\set10108-cw\set10108\labs\cw1\build\Debug\cw1.exe (process 25112) exited with code 0 (0x0).

To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .

It looks like you're encountering an **illegal memory access** error during the execution of your GPU kernel for some tokens, specifically 'the', 'man', and 'woman'. This type of error typically happens when you try to access memory that is out of bounds or not properly synchronized in CUDA. Let's go through the possible reasons for this issue and how you can fix it.

**Causes of Illegal Memory Access**

1. **Out-of-Bounds Access in Shared Memory**:
   * In your kernel, you're copying data into shared memory, and the size of shared memory is based on blockSize. However, if idx (the global index for threads) exceeds the bounds of the input data or if there are edge cases where some threads in the last block have idx values beyond dataSize, this could lead to out-of-bounds memory access.
2. **Token Length and Memory Bounds**:
   * When processing the token 'the', which is a common word and might appear frequently in the text, the number of occurrences might be large, increasing the number of memory operations. You might be reading or writing outside the allocated memory region for shared memory or global memory (such as accessing sharedData[threadIdx.x] where threadIdx.x exceeds the bounds of sharedData).
3. **Thread Synchronization Issues**:
   * If the memory loading phase is not correctly synchronized (e.g., missing \_\_syncthreads() or threads not waiting for each other before accessing shared memory), some threads may access memory before it's fully loaded, leading to invalid memory access.

**Steps to Debug and Fix the Issue**

Here are steps you can take to identify and fix the problem:

**1. Check Data Size in Kernel**

In the calc\_token\_occurrences\_kernel, the index idx can exceed the size of the data when you're accessing global memory or shared memory. You should check whether the thread index exceeds the data size before performing any operations. Modify the part where you load data into shared memory like this:

cpp

Copy code

\_\_global\_\_ void calc\_token\_occurrences\_kernel(char \*data, int dataSize, char \*token, int tokenLen, int \*numOccurrences)

{

// Allocate shared memory for the chunk of text data

extern \_\_shared\_\_ char sharedData[];

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

// Each thread loads one character from global memory to shared memory

if (idx < dataSize) {

sharedData[threadIdx.x] = data[idx];

}

// Make sure all threads in the block have loaded their data into shared memory

\_\_syncthreads();

// Now operate on the shared memory instead of global memory

if (idx < dataSize - tokenLen && threadIdx.x < blockDim.x - tokenLen) {

// Use custom GPU function to compare token with the substring in sharedData

if (gpu\_strncmp(&sharedData[threadIdx.x], token, tokenLen)) {

// Check if the match is a standalone word

bool validPrefix = (idx == 0) || (sharedData[threadIdx.x - 1] < 'a' || sharedData[threadIdx.x - 1] > 'z');

bool validSuffix = (idx + tokenLen >= dataSize) || (sharedData[threadIdx.x + tokenLen] < 'a' || sharedData[threadIdx.x + tokenLen] > 'z');

if (validPrefix && validSuffix) {

atomicAdd(numOccurrences, 1);

}

}

}

}

**2. Allocate Sufficient Shared Memory**

Ensure that enough shared memory is allocated when launching the kernel. You’re currently allocating shared memory based on the block size, but the actual amount of shared memory required may need to be adjusted if your text data chunks are larger or if you have more complex memory access patterns.

Make sure you're allocating enough shared memory for each block to avoid out-of-bounds access:

cpp

Copy code

size\_t sharedMemorySize = blockSize \* sizeof(char); // Ensure enough shared memory

calc\_token\_occurrences\_kernel<<<gridSize, blockSize, sharedMemorySize>>>(d\_data, fileSize, d\_token, tokenLen, d\_numOccurrences);

If the token length exceeds the block size (for very common words like 'the'), shared memory usage can exceed the allocated size, leading to memory errors. Ensure that the token length and block size are well-matched to prevent this.

**3. Handle Edge Cases for Global Memory Access**

When the input data size is not evenly divisible by the block size, some threads in the last block may access memory out of bounds. To fix this, ensure that you handle the case where idx exceeds dataSize - tokenLen. You already have a check, but make sure it applies consistently:

cpp

Copy code

if (idx < dataSize - tokenLen && threadIdx.x < blockDim.x - tokenLen)

This prevents threads in the last block from trying to access memory beyond the end of data.

**4. Use CUDA Tools to Debug**

You can use CUDA’s built-in debugging tools like cuda-memcheck or cuda-gdb to get more detailed information about illegal memory accesses. Running your code through cuda-memcheck will give you precise details about which thread is causing the memory access violation and at which location.

bash

Copy code

cuda-memcheck ./your\_program

**5. Check Return Values of CUDA API Calls**

It’s good practice to check the return values of all CUDA API calls to ensure that there are no silent errors. For example, always check the result of cudaMemcpy, cudaMalloc, and cudaDeviceSynchronize.

**Summary of Fixes:**

1. **Add bounds checking** when loading data into shared memory (idx and threadIdx.x must be within bounds).
2. **Ensure sufficient shared memory is allocated** based on the block size and token length.
3. **Use cuda-memcheck to debug** the specific location of illegal memory access.
4. **Check CUDA return values** for all API calls to handle errors gracefully.

Try these steps and let me know if the issue persists. If you want, I can walk you through implementing and testing any of these suggestions!