Description | Code | Questions | Attempts | History

Machine Problem Code

Compile & Run

```
// MP Reduction
// Given a list (lst) of length n
// Output its sum = lst[0] + lst[1] + ... + lst[n-1];

#include <wb.h>

#define BLOCK_SIZE 512 //@@ You can change this

#define wbCheck(stmt)
  do {
    cudaError_t err = stmt;
    if (err != cudaSuccess) {
      wbLog(ERROR, "Failed to run stmt ", #stmt);
      wbLog(ERROR, "Got CUDA error ...  ", cudaGetErrorString(err));
      return -1;
    }
  } while (0)

void total(float *input, float *output, int len) {
  //@@ Load a segment of the input vector into shared memory
  //@@ Traverse the reduction tree
  //@@ Write the computed sum of the block to the output vector at the
  //@@ correct index
}

int main(int argc, char **argv) {
  int ii;
  wbArg_t args;
  float *hostInput;  // The input 1D list
  float *hostOutput; // The output list
  float *deviceInput;
  float *deviceOutput;
  int numInputElements;  // number of elements in the input list
```

```
34   int numOutputElements; // number of elements in the output list
35
36   args = wbArg_read(argc, argv);
37
38   wbTime_start(Generic, "Importing data and creating memory on host");
39   hostInput =
40       (float *)wbImport(wbArg_getInputFile(args, 0), &numInputElements);
41
42   numOutputElements = numInputElements / (BLOCK_SIZE << 1);
43   if (numInputElements % (BLOCK_SIZE << 1)) {
44     numOutputElements++;
45   }
46   hostOutput = (float *)malloc(numOutputElements * sizeof(float));
47
48   wbTime_stop(Generic, "Importing data and creating memory on host");
49
50   wbLog(TRACE, "The number of input elements in the input is ",
51         numInputElements);
52   wbLog(TRACE, "The number of output elements in the input is ",
53         numOutputElements);
54
55   wbTime_start(GPU, "Allocating GPU memory.");
56   //@@ Allocate GPU memory here
57
58   wbTime_stop(GPU, "Allocating GPU memory.");
59
60   wbTime_start(GPU, "Copying input memory to the GPU.");
61   //@@ Copy memory to the GPU here
62
63   wbTime_stop(GPU, "Copying input memory to the GPU.");
64   //@@ Initialize the grid and block dimensions here
65
66   wbTime_start(Compute, "Performing CUDA computation");
67   //@@ Launch the GPU Kernel here
68
69   cudaDeviceSynchronize();
70   wbTime_stop(Compute, "Performing CUDA computation");
71
72   wbTime_start(Copy, "Copying output memory to the CPU");
73   //@@ Copy the GPU memory back to the CPU here
74
75   wbTime_stop(Copy, "Copying output memory to the CPU");
76
77   /********************************************************************
78    * Reduce output vector on the host
```

```
   * NOTE: One could also perform the reduction of the output vector
   * recursively and support any size input. For simplicity, we do not
   * require that for this lab.
   ********************************************************************/
  for (ii = 1; ii < numOutputElements; ii++) {
    hostOutput[0] += hostOutput[ii];
  }

  wbTime_start(GPU, "Freeing GPU Memory");
  //@@ Free the GPU memory here

  wbTime_stop(GPU, "Freeing GPU Memory");

  wbSolution(args, hostOutput, 1);

  free(hostInput);
  free(hostOutput);

  return 0;
}

```