Assignment 2

Wadood Chaudhary

1. Non-Memory Coalescent

```
__kernel void dot_non_memory_coalscent(const int rows,const int
cols,__global int* matrix,__global int* vector,__global int*
result) {
    int tid = get_global_id(0);
    if (tid<rows) {
      int sum =0;
      for (int c=0;c<cols;c++) {
        int m = tid*cols+c;
        if (tid < 3) printf("(%i,%i) -> %i \n",tid,c,m);
        sum = sum + (matrix[m]*vector[c]);
      }
      result[tid]=sum;
    }

}
```

OpenCL Program

```
void execute (char title[],size_t global, size_t local,char program_name[],char kernel_func[],struct GPU *gpu,
struct Data *data) {
    clock_t t = clock();
    data->global= global; //MIN(global,1024);
    data->local = local; //MIN(local,512);

    printf("\n**********  %s  **********\n",title);
    createKernel(program_name,kernel_func,gpu);
    setKernelArgs(gpu,*data);
    runKernel(gpu,data);
    double time_taken = ((double)(clock()-t))/(CLOCKS_PER_SEC);
    printf("\n GPU: %s took %f seconds to execute...", title,time_taken);
    printResults(*data);
    printf("\n");
    printf(SEPARATOR);

}
```

OpenCL Program

```
void execute (char title[],size_t global, size_t local,char program_name[],char kernel_func[],struct GPU *gpu,
struct Data *data) {
    clock_t t = clock();
    data->global= global; //MIN(global,1024);
    data->local = local; //MIN(local,512);
    printf("\n**********  %s  **********\n",title);
    createKernel(program_name,kernel_func,gpu);
    setKernelArgs(gpu,*data);
    runKernel(gpu,data);

}

struct GPU initGPU() {
    struct GPU gpu;
    cl_int err;
    cl_device_id device;              // compute device id
    cl_context context;               // compute context
    cl_platform_id platform;
    err = clGetPlatformIDs(1, &platform, NULL);
    err = clGetDeviceIDs(platform, CL_DEVICE_TYPE_GPU, 1, &device, NULL);
    context = clCreateContext(NULL, 1, &device, NULL, NULL, &err);
    gpu.context=context;
    gpu.device=device;
    return gpu;
}
```

Non-Memory Coalescent

```
(0,0) -> 0
(1,0) -> 3
(2,0) -> 6
(0,1) -> 1
(1,1) -> 4
(2,1) -> 7
(0,2) -> 2
(1,2) -> 5
(2,2) -> 8
```

2. Memory Coalescent

```
__kernel void dot_memory_coalscent(const int rows,const int
cols,__global int* matrix,__global int* vector,__global int*
result) {

    int tid = get_global_id(0);
    if (tid<rows) {
        int sum =0;
        for (int c=0;c<cols;c++) {
            int m = tid+rows*c;
            if (tid < 3) printf("(%i,%i) -> %i \n",tid,c,m);
            sum = sum + (matrix[m]*vector[c]);
        }
        result[tid]=sum;
    }
}
```

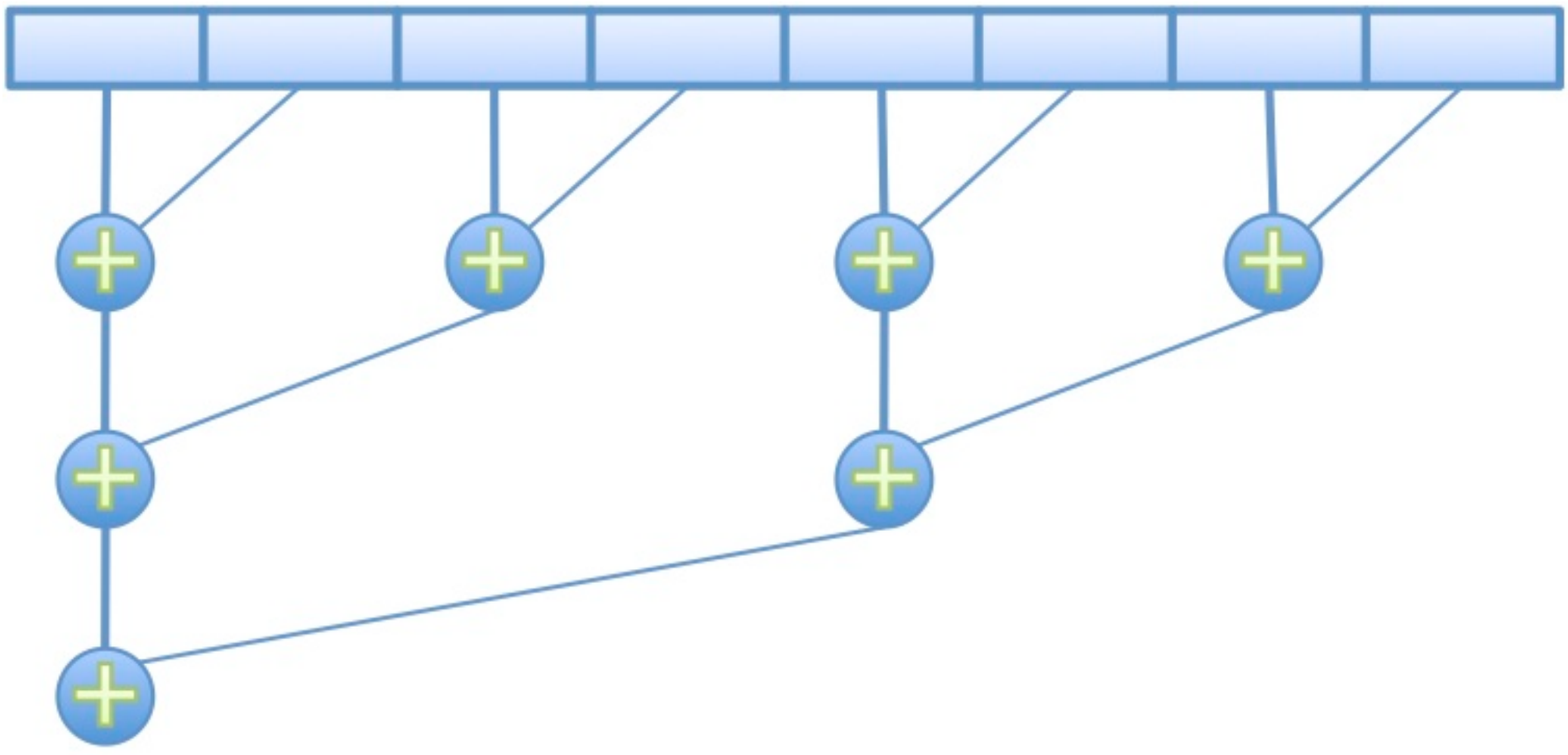
Memory Coalescent

```
(0,0) -> 0
(1,0) -> 1
(2,0) -> 2
(0,1) -> 3
(1,1) -> 4
(2,1) -> 5
(0,2) -> 6
(1,2) -> 7
(2,2) -> 8
```

Comparison

```
arg1=/home/w/wac3/Assignment2/simdata1.txt
 arg2=/home/w/wac3/Assignment2/vectordata1.txt
 arg3=/home/w/wac3/Assignment2/
 rows=1600
 cols=1000
Matrix Data Transferred to the Memory....
Matrix rows x cols = 1600000
Vector Data Transferred to the Memory....
*** CPU: Dot Product took 0.010000 seconds to execute

 New Vector:1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
 Matrix:1,0,1,0,1,2,0,1,0,1,0,2,1,2,0,1,1,1,0,2,
 Correct:1012,1014,1041,986,1002,999,1024,1011,1027,982,989,987,1017,1050,981,1006,1003,995,1011,985,
------------------------------------------------------------------

**********  Non-Memory Coalscent:  *****************

 GPU: Non-Memory Coalscent: took 0.010000 seconds to execute...
 Results(1012,1014,1041,986,1002,999,1024,1011,1027,982,989,987,1017,1050,981,1006,1003,995,1011,985,)
------------------------------------------------------------------

********** CPU: Matrix Transpose took 0.010000 seconds to execute...
**********  Memory Coalscent:  *****************

 GPU: Memory Coalscent: took 0.000000 seconds to execute...
 Results(1012,1014,1041,986,1002,999,1024,1011,1027,982,989,987,1017,1050,981,1006,1003,995,1011,985,)
------------------------------------------------------------------
~
```

3. Reduce Filter



3. Reduce Filter

```
__kernel void dot_memory_coalscent_2(const int
rows,const int cols,__global int* matrix,__global
int* vector,__global int* result) {
        int tid = get_global_id(0);
        int c =   get_local_id(0);
        if (tid<rows*cols) {
                int m = tid;
                 matrix[m] = matrix[m]*vector[c];
        }
}
```

3. Reduce Filter

```
for(i = 0; i < pass_count; i++){
            size_t global = group_counts[i] * work_item_counts[i];
            size_t local = work_item_counts[i];
            unsigned int operations = operation_counts[i];
            unsigned int entries = entry_counts[i];
            size_t shared_size = typesize * 1 * local * operations;
            pass_swap = pass_input;
            pass_input = pass_output;
            pass_output = pass_swap;
            setKernelArgs(gpu,&pass_output,&pass_input,shared_size,&entries);
            runKernel(gpu, 1, NULL, &global, &local, 0, NULL, NULL);

        }
```