Assignment 10

The given DAG with 3 kernels, the edge denotes that the output of the parent node is the input of the child node. The kernels are executing in out-of-order fashion. Assume the target hardware has one AMD CPU and one AMD GPU. All the inputs and output buffers are of the same size and dimension. queueCPU and queueGPU are the command queues for CPU and GPU, datsize is the variable for data size, d_a, d_c, d_e and h_a, h_c, h_e are the cl_buffers and memory at host side respectively. globalws and localws are the global worksize and local worksize respectively. E1, E2, E3, E4, E5 and E6 are the cl_events.

![DAG Diagram]

COMMON DATA FOR QUESTIONS 1 TO 3

1) 8 points

Due on 2020-04-08, 23:59 IST.
As per our records you have not submitted this assignment.

https://onlinecourses.nptel.ac.in/noc20_cs41/unit?unit=36&assessment=139
**Heterogeneous Computing**

- Lecture 49: OpenCL - Heterogeneous Computing (Contd.)
- Lecture 50: OpenCL - Heterogeneous Computing (Contd.)
- Lecture 51: OpenCL - Heterogeneous Computing (Contd.)
- Lecture 52: OpenCL - Heterogeneous Computing (Contd.)
- Lecture 53: OpenCL - Heterogeneous Computing (Contd.)

**Quiz: Assignment 10**

- Week 10 Feedback Form

---

**Lecture Material**

**Choices**

- NULL, 0, 1, 2, 3, 4, 5, 6, E1, E2, E3, E4, E5, E6
  - a) i -> 0, ii -> NULL, iii -> 0, iv -> NULL
  - b) i -> 0, ii -> NULL, iii -> 1, iv -> E1
  - c) i -> 1, ii -> E1, iii -> 0, iv -> NULL
  - d) i -> 0, ii -> NULL, iii -> 0, iv -> E1

---

No, the answer is incorrect.
Score: 0
Accepted Answers:
- a)
2) a) $v \rightarrow 1$, $v_i \rightarrow E2$, $v_{ii} \rightarrow 1$, $v_{iii} \rightarrow E3$
b) $v \rightarrow 2$, $v_i \rightarrow E1, E2$, $v_{ii} \rightarrow 1$, $v_{iii} \rightarrow E3$
c) $v \rightarrow 1$, $v_i \rightarrow E1$, $v_{ii} \rightarrow 1$, $v_{iii} \rightarrow E2$
d) $v \rightarrow 1$, $v_i \rightarrow E1$, $v_{ii} \rightarrow 1$, $v_{iii} \rightarrow E2$

No, the answer is incorrect.
Score: 0
Accepted Answers:
d)

3) a) $i_{x} \rightarrow 1$, $x \rightarrow E4$, $x_{i} \rightarrow 1$, $x_{ii} \rightarrow E5$
b) $i_{x} \rightarrow 2$, $x \rightarrow E4, E5$, $x_{i} \rightarrow 1$, $x_{ii} \rightarrow E6$
c) $i_{x} \rightarrow 2$, $x \rightarrow E4, E5$, $x_{i} \rightarrow 1$, $x_{ii} \rightarrow E5$
d) $i_{x} \rightarrow 1 \rightarrow E3$, $x_{i} \rightarrow 1$, $x_{ii} \rightarrow E4$

No, the answer is incorrect.
Score: 0
Accepted Answers:
c)

4) Consider the same DAG as in question 1. The kernels are executed in in-order fashion. Assume the target hardware has one AMD GPU. All the inputs and output buffers are of the same size and dimension. What will be the efficient method of synchronization?

Options

a) Command-queue barrier for each command
b) clFlush at the end of all commands
c) clFinish at the end of all commands
d) Waiting on an event associated with each command

No, the answer is incorrect.
Score: 0
Accepted Answers:
5) 25 points
Consider an incomplete CUDA code using streams which processes a 2D Matrix of dimensions \( W \times W \) using a 1D Kernel. The matrix is stored in row-major format. The 1D Kernel processes the array in the following way. For the first row, the kernel processes \( W \) elements. For the second row, the kernel processes \( W/2 \) elements. For the third row, the kernel processes \( W/4 \) elements and so on. Fill in the blanks.

```c
// initiate all asynchronous transfers to the device

// NSTREAM is Where
for (int i = 0; i < NSTREAM; ++i){
    int ioffset = _____; int iElem = _____; iBytes = iElem * sizeof(float);
    cudaMemcpyAsync(&d_A[ioffset], &h_A[ioffset], iBytes, cudaMemcpyHostToDevice, stream[i]);
}

for (int i = 0; i < NSTREAM; ++i){
    int ioffset = _____; int iElem = _____;
    process <<<grid.block.0,stream[i]>>> (&d_A[ioffset], &d_B[ioffset], &d_C[ioffset], iElem);
}

for (int i = 0; i < NSTREAM; ++i){
    int ioffset = _____; iElem = _____; iBytes = iElem * sizeof(float);
    cudaMemcpyAsync(&h_op[ioffset], &d_A[ioffset], iBytes, cudaMemcpyDeviceToHost, stream[i]);
}

The values of iElem and ioffset are the same for all the cases. Select the correct option from below.

a) ioffset = i*W, iElem = W/(1<<i)

b) ioffset = i*W, iElem = i*W

c) ioffset = i*W, iElem = i*W/(i<<1)

d) ioffset = i, iElem = i*W
6) Suppose an application that uses OpenCL for certain operations also uses a third-party library that also happens to use OpenCL internally to accelerate some algorithms on a device having a single OpenCL-supported device. What will be the preferred choice for context and command queues.

Options

a) Multiple Command Queues With Different Contexts In Single Device
b) Multiple Command Queues With Same Context In Single Device Example
c) Single Command Queue With Single Context In Single Device
d) Single Command Queue With Different Context In Single Device Example

No, the answer is incorrect.
Score: 0
Accepted Answers: a)

7) What is the advantage of implementing multiple command queues with same context in a single device for an OpenCL code.

Options

a) Improves synchronization and thus helps in faster data processing
b) Reduces kernel launch overhead by having multiple command queues
c) Pipeline can be formed for better utilization by not having the device sit idle waiting for data
d) Improves data sharing across multiple command queues in the same context thus improving the performance

No, the answer is incorrect.
Score: 0
Accepted Answers: a)
8) **20 points**

Consider the reduction kernel to find the maximum of a given data set \(2^{24}\) elements.

```c
__global__ void max ( int * g_idata , int * g_odata ,
unsigned int n){
    __shared__ int sdata [2048];
    unsigned int tid = threadIdx .x;
    unsigned int i = blockIdx .x * ( blockDim .x * 2) +
threadIdx .x;
    sdata [ tid ] = g_idata [i] + g_idata [i+blockDim .x];
    __syncthreads ();
    for ( unsigned int s= blockDim .x /2; s>0; s >>=1) {
        if ( tid < s)
            sdata [ tid ] =max(sdata [ tid ], sdata [ tid +
s]);
        __syncthreads ();
    }
    if ( tid == 0)
        g_odata [ blockIdx .x] = sdata [0];
}
```

If you apply thread coarsening with coarsening factor 4 for a kernel. The maximum for a size of a work group that can be launched is 2048 work-items. The kernel invocation command is given below. What is the content of the arrays `global_work_size` and `local_work_size`?

```c
er = clEnqueueNDRangeKernel (commands, max_coarsened, 1, NULL,
&global_work_size, &local_work_size, 0, NULL, NULL);
```

**Options**

a) global_work_size = \(\{8192,1,1\}\), local_work_size = \(\{2048,1,1\}\)
b) global_work_size = \(\{4194304,1,1\}\), local_work_size = \(\{2048,1,1\}\)
c) global_work_size = \(\{32768,1,1\}\), local_work_size = \(\{512,1,1\}\)
d) global_work_size = \(\{67108864,1,1\}\), local_work_size = \(\{512,1,1\}\)
c) 
d)
No, the answer is incorrect.
Score: 0
Accepted Answers:
b)