Matrix Multiplication

DPC++ program

In [1]:

```
%%writefile ~/arc/matrix_multiplication.cpp
#include <CL/sycl.hpp>
#include <iostream>
#include <sycl/ext/intel/fpga_extensions.hpp>
#include <chrono>
using namespace sycl;

#define MATRIX_SIZE 1024

int main(int argc, char* argv[]) {

    size_t N = MATRIX_SIZE;
    bool printResult = false;
    bool validateResult = true;

    std::cout << "Matrix size = [ " << N << " x " << N << " ]\n";

    // define vectors for matrices
    std::vector<float> in1(N * N);
    std::vector<float> in2(N * N);
    std::vector<float> out(N * N);
    std::vector<float> val(N * N);

    // load input vectors
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            in1[i * N + j] = rand() % 9 + 1;
            in2[i * N + j] = rand() % 9 + 1;
        }
    }

    // create the queue
    queue queue(property::queue::enable_profiling {});
    std::cout << "Offload Device: " << queue.get_device().get_info<info::device::name>() << "\n";

    // create a two-dimensional NxN range object
    range<2> num_items{N,N};

    // create buffers
    buffer in1_buffer(in1);
    buffer in2_buffer(in2);
    buffer out_buffer(out);

    auto device_start = std::chrono::high_resolution_clock::now();
    auto event = queue.submit([&](handler& handler) {

        // create accessors for the input/output buffers
        auto in1_accessor = in1_buffer.get_access<access::mode::read>(handler);
        auto in2_accessor = in2_buffer.get_access<access::mode::read>(handler);
        auto out_accessor = out_buffer.get_access<access::mode::write>(handler);

        // perform operation using parallel_for
        // 1st param: num work items
        // 2nd param: kernel to specify what to do per work item
        handler.parallel_for(num_items, [=](item<2> item) {
            const int i = item.get_id(0);
            const int j = item.get_id(1);
            for (int k = 0; k < N; k++) {
                out_accessor[i * N + j] += in1_accessor[i * N + k] * in2_accessor[k * N + j];
            }
        });

    });
    auto device_stop = std::chrono::high_resolution_clock::now();
    auto device_duration = std::chrono::duration_cast<std::chrono::milliseconds>(device_stop - device_start);

    // allow read access for output buffer
    out_buffer.get_access<access::mode::read>();

    // get reported times from kernel event profile
    auto kernel_end = event.get_profiling_info<info::event_profiling::command_end>();
    auto kernel_start = event.get_profiling_info<info::event_profiling::command_start>();
    auto kernel_duration = (kernel_end - kernel_start) / 1.0e6;

    // host computation for validation and timing comparision
    auto host_start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                val[i * N + j] += in1[i * N + k] * in2[k * N + j];
            }
        }
    }
    auto host_stop = std::chrono::high_resolution_clock::now();
    auto host_duration = std::chrono::duration_cast<std::chrono::milliseconds>(host_stop - host_start);

    // validate
    if(validateResult){
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if ((out[i * N + j] - val[i * N + j]) > 1e-6) {
                    std::cout << "Incorrect values from device.\n";
                    return -1;
                }
            }
        }
    }

    // print
    if(printResult){
        std::cout << "\n";
        for (int i=0; i<N; i++){
            std::cout << "[ ";
            for (int j=0; j<N; j++){
                std::cout << in1[i*N+j] << " ";
            }
            if(i==0){
                std::cout << "] * [ ";
            }
            else {
                std::cout << "]   [ ";
            }
            for (int j=0; j<N; j++){
                    std::cout << in2[i*N+j] << " ";
            }
            if(i==0){
                std::cout << "] = [ ";
            }
            else {
                std::cout << "]   [ ";
            }
            for (int j=0; j<N; j++){
                    std::cout << out[i*N+j] << " ";
            }
            std::cout <<  "]\n";
        }
    }

    // display timing results
    std::cout <<"\nHost time to execute kernel: \n " << kernel_duration << " milliseconds\n"
        << "Device compute time: \n " << device_duration.count() << " milliseconds\n";
    if(validateResult){
        std::cout << "\nversus...\n"
        << "Sequential host compute time: \n " << host_duration.count() << " milliseconds\n\n";
    }

    return 0;
}
```

```
Overwriting /home/u167808/arc/matrix_multiplication.cpp
```

Shell script to compile and run program

In [2]:

```
%%writefile ~/arc/matrix_multiplication.sh
#!/bin/bash
source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1

echo ====================
echo matrix_multiplication
dpcpp ~/arc/matrix_multiplication.cpp -o ~/arc/matrix_multiplication -w -O3
~/arc/matrix_multiplication
echo ====================
```

```
Overwriting /home/u167808/arc/matrix_multiplication.sh
```

Script to queue jobs on Intel DevCloud

In [3]:

```
%%writefile ~/arc/submit_job.sh
#==========================================
# Copyright © 2020 Intel Corporation
#
# SPDX-License-Identifier: MIT
#==========================================
# Script to submit job in Intel(R) DevCloud
# Version: 0.72
#==========================================

if [ -z "$1" ]; then
    echo "Missing script argument, Usage: ./q run.sh"
elif [ ! -f "$1" ]; then
    echo "File $1 does not exist"
else
    echo "Job has been submitted to Intel(R) DevCloud and will execute soon."
    echo ""
    script=$1
    property=$2
     if [ "$property" == "GPU GEN9" ]; then
             value="gen9"
        elif [ "$property" == "GPU Iris XE Max" ]; then
            value="iris_xe_max"
        elif [ "$property" == "CPU Xeon 8153" ]; then
            value="renderkit"
        elif [ "$property" == "CPU Xeon 8256" ]; then
            value="stratix10"
        elif [ "$property" == "CPU Xeon 6128" ]; then
            value="skl"
        else
            value="gen9"
    fi
    if [ "$property" == "{device.value}" ]; then
        echo "Selected Device is: GPU"
    else
        echo "Selected Device is: "$property
    fi
    echo ""
    # Remove old output files
    rm *.sh.* > /dev/null 2>&1
    # Submit job using qsub
    qsub_id=`qsub -l nodes=1:$value:ppn=2 -d . $script`
    job_id="$(cut -d'.' -f1 <<<"$qsub_id")"
    # Print qstat output
    qstat
    # Wait for output file to be generated and display
    echo ""
    echo -ne "Waiting for Output "
    until [ -f $script.o$job_id ]; do
        sleep 1
        echo -ne "▒"
        ((timeout++))
        # Timeout if no output file generated within 60 seconds
        if [ $timeout == 60 ]; then
            echo ""
            echo ""
            echo "TimeOut 60 seconds: Job is still queued for execution, check for output file later ($script.o$job_id)"
            echo ""
            break
        fi
    done
    # Print output and error file content if exist
    if [ -n "$(find -name '*.sh.o'$job_id)" ]; then
        echo " Done⬇"
        cat $script.o$job_id
        cat $script.e$job_id
        echo "Job Completed in $timeout seconds."
        rm *.sh.*$job_id > /dev/null 2>&1
    fi
fi
```

```
Overwriting /home/u167808/arc/submit_job.sh
```

Execute program

In [4]:

```
! chmod 755 ~/arc/submit_job.sh; chmod 755 ~/arc/matrix_multiplication.sh; ~/arc/submit_job.sh ~/arc/matrix_multiplication.sh "GPU Gen9";
```

```
Job has been submitted to Intel(R) DevCloud and will execute soon.

Selected Device is: GPU Gen9

Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2034760.v-qsvr-1           ...ub-singleuser u167808         00:00:23 R jupyterhub
2034795.v-qsvr-1           ...iplication.sh u167808                0 Q batch

Waiting for Output ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

TimeOut 60 seconds: Job is still queued for execution, check for output file later (/home/u167808/arc/matrix_multiplication.sh.o2034795)

 Done⬇

################################################################################
#      Date:           Wed 09 Nov 2022 03:40:33 AM PST
#    Job ID:           2034795.v-qsvr-1.aidevcloud
#      User:           u167808
# Resources:           neednodes=1:gen9:ppn=2,nodes=1:gen9:ppn=2,walltime=06:00:00
################################################################################

====================
matrix_multiplication
Matrix size = [ 1024 x 1024 ]
Offload Device: Intel(R) UHD Graphics P630 [0x3e96]

Host time to execute kernel:
 167.894 milliseconds
Device compute time:
 274 milliseconds

versus...
Sequential host compute time:
 1387 milliseconds

====================

################################################################################
# End of output for job 2034795.v-qsvr-1.aidevcloud
# Date: Wed 09 Nov 2022 03:40:57 AM PST
################################################################################

Job Completed in 60 seconds.
```