Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о лабораторной работе №3

«Несколько этапные неграфические вычисления на GPU»

по дисциплине

«Специализированные ЭВМ»

Выполнили:

Студенты группы 045

Вашкулатов Н.А.

Анохин В.А.

Проверил:

доц. каф. ЭВМ Устюков Д.И.

доц. каф. ЭВМ Муратов Е.Р.

**Цель работы:** изучение языка OpenCL на примере несколько этапных неграфических вычислений на GPU.

**Ход работы**

1. Используйте программу, написанную на лабораторной работе №1 как основу для модернизации. Внесите в ней изменения обеспечивающие возможность не копировать буфер с устройства после вычислений а повторно использовать его для аналогичного вычисления. Чтобы в результате появилась надпись HelloWorld необходимо изменить строку

const char\* input = "GdkknVnqkc";

на

const char\* input = "FcjjmUmpjb";

Выполните программу так чтобы kernel вызывался последовательно два раза. результат правильного выполнения – HelloWorld на экране монитора.

**Код программы OpenCL:**

\_\_kernel void helloworld( global char\* in, global char\* out)

{

int num = get\_global\_id(0);

out[num] = in[num] + 1;

}

**Код программы:**

void Init() {

cl\_uint numPlatforms;

cl\_platform\_id platform = NULL;

cl\_int status = clGetPlatformIDs(0, NULL, &numPlatforms);

isSuccess(status);

cout << "NumPlatforms: " << numPlatforms << endl;

if (numPlatforms > 0)

{

cl\_platform\_id\* platforms = (cl\_platform\_id\*)malloc(numPlatforms \* sizeof(cl\_platform\_id));

status = clGetPlatformIDs(numPlatforms, platforms, NULL);

isSuccess(status);

platform = platforms[0];

free(platforms);

}

cl\_uint numDevices = 0;

cl\_device\_id\* devices;

status = clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, 0, NULL, &numDevices);

isSuccess(status);

cout << "NumDevices: " << numDevices << endl;

devices = (cl\_device\_id\*)malloc(numDevices \* sizeof(cl\_device\_id));

status = clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, numDevices, devices, NULL);

isSuccess(status);

cl\_context context = clCreateContext(NULL, 1, devices, NULL, NULL, NULL);

cl\_command\_queue commandQueue = clCreateCommandQueue(context, devices[0], 0, NULL);

cout << endl << "Run programs" << endl;

const char\* filename = "HelloWorld\_Kernel.cl";

const char\* filename2 = "arr2.cl";

const char\* filename3 = "arr3.cl";

cout << endl << "First program" << endl;

HelloWOrld(filename, context, devices, commandQueue);

cout << endl << "Second program" << endl;

arr1(filename2, context, devices, commandQueue);

cout << endl << "Third program" << endl;

arr2(filename3, context, devices, commandQueue);

status = clReleaseCommandQueue(commandQueue);

status = clReleaseContext(context);

if (devices != NULL)

{

free(devices);

devices = NULL;

}

}

void HelloWOrld( const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue)

{

cl\_int status;

string sourceStr;

status = convertToString(filename, sourceStr);

const char\* source = sourceStr.c\_str();

size\_t sourceSize[] = { strlen(source) };

cl\_program program = clCreateProgramWithSource(context, 1, &source, sourceSize, NULL);

status = clBuildProgram(program, 1, devices, NULL, NULL, NULL);

const char\* input = "FcjjmUmpjb";

size\_t strlength = strlen(input);

char\* output = (char\*)malloc(strlength + 1);

cl\_mem inputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, (strlength + 1) \* sizeof(char), (void\*)input, NULL);

cl\_kernel kernel = clCreateKernel(program, "helloworld", NULL);

status = clSetKernelArg(kernel, 0, sizeof(cl\_mem), (void\*)&inputBuffer);

status = clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&inputBuffer);

size\_t global\_work\_size[1] = { strlength };

status = clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueReadBuffer(commandQueue, inputBuffer, CL\_TRUE, 0, strlength \* sizeof(char), output, 0, NULL, NULL);

output[strlength] = '\0';

cout << output << endl;

status = clReleaseKernel(kernel);

status = clReleaseProgram(program);

status = clReleaseMemObject(inputBuffer);

if (output != NULL)

{

free(output);

output = NULL;

}

}



Рисунок 1 – Результат выполнения программы

Измените размер буфера до 1000 элементов, заполните значениями и обработайте данные согласно варианту задания.

2. Заполните массив элементами типа int4 (размер элемента 4 байта) значениями по возрастанию, на первом шаге вычислите A.x=A.x+A.z, на втором A.y=trunc((A.w+A.x)/3).

**Код программы OpenCL:**

\_\_kernel void mul( global int4\* A, global int4\* out)

{

int num = get\_global\_id(0);

out[num].yzw = A[num].yzw;

out[num].x = A[num].x + A[num].z;

}

\_\_kernel void mul2( global int4\* A, global int4\* out)

{

int num = get\_global\_id(0);

out[num].xzw = A[num].xzw;

out[num].y = trunc((A[num].w+A[num].x)/3.0);

}

**Код программы:**

void arr1(const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue)

{

cl\_int status;

string sourceStr;

status = convertToString(filename, sourceStr);

const char\* source = sourceStr.c\_str();

size\_t sourceSize[] = { strlen(source) };

cl\_program program = clCreateProgramWithSource(context, 1, &source, sourceSize, NULL);

status = clBuildProgram(program, 1, devices, NULL, NULL, NULL);

const size\_t size = 1000;

cout << "Input data: " << endl;

cl\_int4 input[size];

for (size\_t i = 1; i <= size; i++) {

input[i-1].s[0] = i\*10;

input[i-1].s[1] = i+i/2;

input[i-1].s[2] = i\*2;

input[i-1].s[3] = i/2\*5;

}

for (size\_t i = 0; i < 10; i++) {

cout << "(" << input[i].s[0] << ", " << input[i].s[1] << ", " << input[i].s[2] << ", " << input[i].s[3] << ") ";

}

cout << endl;

cl\_int4\* output = (cl\_int4\*)malloc(size);

cl\_mem inputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_WRITE | CL\_MEM\_COPY\_HOST\_PTR, (size) \* sizeof(cl\_int4), (void\*)input, NULL);

cl\_mem outputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_WRITE | CL\_MEM\_COPY\_HOST\_PTR, (size) \* sizeof(cl\_int4), (void\*)output, NULL);

cl\_kernel kernel = clCreateKernel(program, "mul", NULL);

cl\_kernel kernel2 = clCreateKernel(program, "mul2", NULL);

status = clSetKernelArg(kernel, 0, sizeof(cl\_mem), (void\*)&inputBuffer);

status = clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&outputBuffer);

status = clSetKernelArg(kernel2, 0, sizeof(cl\_mem), (void\*)&outputBuffer);

status = clSetKernelArg(kernel2, 1, sizeof(cl\_mem), (void\*)&inputBuffer);

size\_t global\_work\_size[1] = { size };

status = clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueNDRangeKernel(commandQueue, kernel2, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueReadBuffer(commandQueue, inputBuffer, CL\_TRUE, 0, size \* sizeof(cl\_int4), output, 0, NULL, NULL);

cout << "Output data: " << endl;

for (size\_t i = 0; i < 10; i++)

{

cout << "(" << output[i].s[0] << ", " << output[i].s[1] << ", " << output[i].s[2] << ", " << output[i].s[3] << ") ";

}

cout << endl;

//status = clReleaseKernel(kernel);

status = clReleaseKernel(kernel2);

status = clReleaseProgram(program);

status = clReleaseMemObject(inputBuffer);

status = clReleaseMemObject(outputBuffer);

}

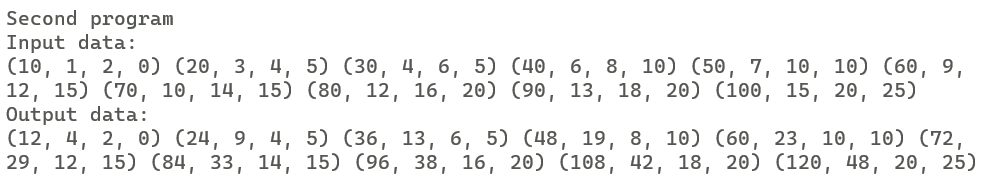


Рисунок 2 – Результат выполнения программы

3. Заполните массив элементами типа integer (размер элемента 4 байта) значениями 1, на первом шаге вычислите𝐴i = 𝐴i+1 + i, на втором для каждых элементов с нечетным значением вычесть 1 для элементов с четным значением поменять знак.

**Код программы OpenCL:**

\_\_kernel void sum( global int\* A, global int\* out)

{

int num = get\_global\_id(0);

int size = get\_global\_size(0);

if(num < size){

out[num] = A[num+1]+num;

}

else out[num] = A[num];

}

\_\_kernel void change( global int\* A, global int\* out)

{

int num = get\_global\_id(0);

if (A[num] % 2 == 0){

out[num] = -A[num];

}

else{

out[num] = A[num] - 1;

}

}

**Код программы:**

void arr2(const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue)

{

cl\_int status;

string sourceStr;

status = convertToString(filename, sourceStr);

const char\* source = sourceStr.c\_str();

size\_t sourceSize[] = { strlen(source) };

cl\_program program = clCreateProgramWithSource(context, 1, &source, sourceSize, NULL);

status = clBuildProgram(program, 1, devices, NULL, NULL, NULL);

const size\_t size = 100;

int input[size];

for (size\_t i = 0; i < size; i++) {

input[i] = 1;

}

int\* output = (int\*)malloc(size);

cl\_mem inputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_WRITE | CL\_MEM\_COPY\_HOST\_PTR, (size) \* sizeof(int), (void\*)input, NULL);

cl\_mem outputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_WRITE | CL\_MEM\_COPY\_HOST\_PTR, (size) \* sizeof(int), (void\*)output, NULL);

cl\_kernel kernel = clCreateKernel(program, "sum", NULL);

cl\_kernel kernel2 = clCreateKernel(program, "change", NULL);

status = clSetKernelArg(kernel, 0, sizeof(cl\_mem), (void\*)&inputBuffer);

status = clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&outputBuffer);

status = clSetKernelArg(kernel2, 0, sizeof(cl\_mem), (void\*)&outputBuffer);

status = clSetKernelArg(kernel2, 1, sizeof(cl\_mem), (void\*)&inputBuffer);

size\_t global\_work\_size[1] = { size };

status = clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueNDRangeKernel(commandQueue, kernel2, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

status = clEnqueueReadBuffer(commandQueue, inputBuffer, CL\_TRUE, 0, size \* sizeof(int), output, 0, NULL, NULL);

cout << "Output data: " << endl;

for (size\_t i = 0; i < size; i++)

{

cout << output[i] << " ";

}

cout << endl;

//status = clReleaseKernel(kernel);

status = clReleaseKernel(kernel2);

status = clReleaseProgram(program);

status = clReleaseMemObject(inputBuffer);

status = clReleaseMemObject(outputBuffer);

}

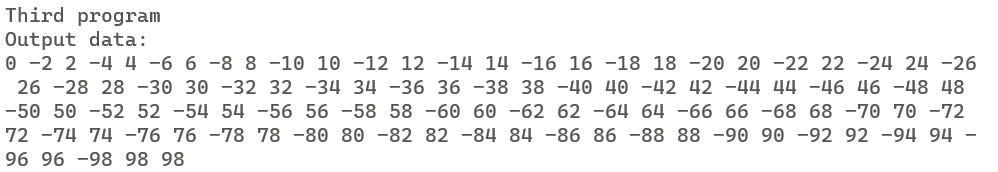


Рисунок 3 – Результат выполнения программы

**Вывод**: в ходе работы был изучен язык OpenCL на примере несколько этапных неграфических вычислений на GPU.