Применение технологии OpenCL для ускорения вычисления интегралов

А.А. Бурцев

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, burtsev@niisi.msk.ru

Аннотация. Статья посвящена применению технологии OpenCL, позволяющей использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Рассматриваются варианты параллельных программ, разработанных для ускорения операции вычисления интегралов в среде OpenCL.

Ключевые слова: параллельное программирование, технология OpenCL, гетерогенные системы, численные методы вычисления интегралов

1. Введение

Современные высокопроизводительные системы для ускорения вычислительных программ, как правило, предлагают освоить те или иные особые технологии разработки параллельной программы, ориентированные на исполнение такой программы на многоядерных процессорах с общей памятью (OpenMP [1, п.5.1]) или на семействе компьютеров, связанных сетью (MPI [1, п.5.2]). Однако, сегодня можно ускорить выполнение своей вычислительной программы даже на одном компьютере, если в его составе имеется мощная видеокарта, которая поддерживает технологию OpenCL [2].

Технология OpenCL позволяет ускорить вычислительную программу за счёт её особого распараллеливания для последующего её исполнения на массиве однородных специализированных процессоров, функционирующих в составе современной графической видеокарты. Сначала аналогичную технологию (CUDA) предложила компания NVIDIA, потом − компания Apple, а Khronos Group приняла эту технологию за основу и довела до стандарта, назвав её **OpenCL** (**Open C**omputing **L**anguage).

Данная статья продолжает знакомство с технологией OpenCL, начатое автором в работах [3-5]. В них рассматривались приёмы разработки в среде OpenCL эффективных программ для решения отдельных задач линейной алгебры и цифровой обработки сигналов. В частности, рассматривались варианты построения по технологии OpenCL программ, ускоряющих операцию перемножения больших матриц, и программ, ускоряющих в среде OpenCL выполнение операции быстрого преобразования Фурье (БПФ для комплексных векторов большой длины вида $N=2^P$).

Полученный опыт разработки OpenCL-программ позволяет утверждать, что с помощью технологии OpenCL действительно можно существенно ускорить выполнение таких вычислительных программ, в которых требуется осуществлять однотипные операции над огромными массивами данных. Так, например, операцию перемножения больших матриц вещественных чисел (размером 2048×2048) удалось ускорить примерно в **15** раз (см. табл. 6 в [3]) на обычном настольном компьютере с универсальным процессором CPU Intel i59400 (с частотой 2.9 ГГц) и встроенным графическим процессором GPU UHD 630 (с частотой 350 МГц). А на компьютере с процессором CPU Intel i3-2100 (3.1 ГГц) и подключённой специализированной видеокартой NVidia GeForce 1050ti (1392 МГц) удалось добиться ускорения для той же операции в **95** раз (см. табл. 9 в [3]). На тех же компьютерах удалось значительно ускорить и операцию БПФ для комплексных векторов: например, для векторов длины $N=2^{24}$ получены показатели ускорения соответственно в **32** раза и в **190** раз (см. табл. 1 и табл. 2 в [4]).

Следует, однако, заметить, что для совершения в среде OpenCL операций над массивами приходится непроизводительно тратить время на передачу исходных данных из основной памяти компьютера в память OpenCL-устройства, а после – копировать обратно результаты совершённой операции. И поэтому можно надеяться, что для вычислительных задач, в которых такие передачи данных сведены к минимуму (или вообще отсутствуют), применение технологии OpenCL позволит обеспечить более значительные показатели ускорения. Приближённое вычисление интегралов как раз и относится к такой группе вычислительных задач, выполнение которых можно легко распараллелить, а передачи данных свести с минимуму.

2. Последовательная программа для вычисления интегралов

Необходимость приближённого вычисления интегралов возникает всякий раз, когда невозможно (или затруднительно) выразить требуемый интеграл аналитическим методом в виде законченной формулы с применением элементарных функций. Известен целый ряд подобных так называемых «неберущихся» интегралов (см. [6]). Среди них отметим интеграл Френеля:

$$\int sin(x^2)\,dx$$

а также интеграл Пуассона:

$$\int e^{-x^2}dx$$

В дальнейшем будем использовать формулы этих интегралов при демонстрации примеров применения программ, разработанных автором в среде OpenCL для приближённого вычисления интегралов.

Для вычисления интегралов применяются разнообразные численные методы. Самый простой из них – метод прямоугольников. Он позволяет для интеграла вида:

$$\int_A^B f(x)\ \ dx$$

получить приближённое значение по формуле:

$$S = h \times \sum_{k=0}^{N-1} f(x_k)\,,$$

где: $h = \frac{B-A}{N}$, $x_k = A + k \cdot h$,

вычислив значение интегрируемой функции лишь в нескольких точках заданного отрезка.

Точность вычисления по такой формуле определяется числом **N,** на которое разбивается отрезок интегрирования [A,B]. И значение интеграла вычисляется, как сумма площадей всех прямоугольников, образованных в результате такого разбиения. Предполагается, что для достижения лучшей точности следует задавать большее значение N.

В обычной (последовательной) программе алгоритм вычисления интеграла по формуле прямоугольников можно выразить такой Си-функцией:

```
real Intgrl(int N, real A, real B){
 int k;  real x, h, Sum;
  Sum=0.0; h= (B-A)/(real)N;
  for (k=0; k < N; k++ )
  { x = A + k*h; Sum= Sum + F(x); }
  return (Sum*h);
}//Intgrl
```

Предполагается, что в программе предварительно определяется тип **real** для представления вещественных чисел с одинарной:

```
#define real float
```

или двойной точностью:

```
#define real double
```

а также Си-функция **F** для вычисления интегрируемой функции. При этом формулу для вычисления такой функции удобно задавать как define-переменную:

```
#define FUNC sin(x*x) /* для Френеля */
//#define FUNC exp(-x*x) /* для Пуассона */
real F(real x) { return FUNC; }
```

Это позволяет быстро перенастроить всю программу на вычисление интеграла другой функции и/или для другого типа вещественных чисел.

3. Программа для вычисления интегралов в среде OpenCL

В среде OpenCL, где могут параллельно функционировать несколько вычислительных ядер в качестве исполнителей, можно обеспечить одновременное вычисление значений интегрируемой функции сразу в нескольких точках. Просуммировав затем эти значения, можно в итоге получить результат интегрирования значительно быстрее. Но для организации такого процесса параллельных вычислений должна быть построена соответствующая OpenCL-программа, т.е. программа, осуществляющая требуемое распараллеливание в среде OpenCL.

Всякая программа, разработанная в расчёте на её параллельное исполнение в среде OpenCL, состоит из основной программы (OpenCL-приложения на языке Си) и нескольких так называемых процедур ядра (на языке OpenCL). Основная программа запускается на основном процессоре (хосте), подготавливает OpenCL-среду и периодически запускает в ней приготовленные процедуры ядра параллельно сразу на всех её вычислительных узлах − так называемых обрабатывающих элементах (processing element).

Для подготовки OpenCL-среды необходимо проинициализировать OpenCL-платфому, дескрипторы OpenCL-устройств, приготовить OpenCL-контекст и создать в нём дескриптор очереди команд. Требуется также приготовить объекты для представления в памяти OpenCL-устройства данных, подлежащих обработке, дескриптор для каждой процедуры ядра, и особый объект, содержащий в откомпилированной форме код всех процедур ядра. Как обеспечить все эти подготовительные действия, подробно объяснялось в предыдущих статьях автора (см.

п. 2.3-2.5 в [3] и п.3.2 в [4]).

Здесь же сосредоточим внимание, главным образом, на процедурах ядра, отражающих специфику выполнения задачи интегрирования, и той части основной программы, которая обеспечивает их запуск и параллельное исполнение на массиве вычислительных ядер.

3.1. Процедуры OpenCL-ядра для вычисления интеграла

Вычисление значения интеграла в среде OpenCL выполним в два этапа. На первом этапе будем запускать на всех исполнителях процедуру ядра **_Intgrl** (см. далее), а на втором этапе запустим процедуру ядра **_Sum** лишь на одном исполнителе.

Процедуры ядра, предназначенные для исполнения в среде OpenCL на каждом обрабатывающем элементе (ОЭ), подготовим в предположении, что при их параллельном запуске становятся известны общее количество таких исполнителей **P**, а также количество **Q** и размер **WS** рабочих групп (Working Group), в которые их можно объединять для совместной работы (предполагается, что **P** делится нацело на **WS**).

И будем полагать, что в качестве параметров процедуре ядра можно передать границы отрезка интегрирования [**A**,**B**] и количество точек **N**, на которых следует вычислять значения интегрируемой функции. А формула вычисления предназначенной для интегрирования функции задаётся (как и в Си-программе) с помощью **define**-переменной **FUNC**:

```
#define FUNC sin(x*x) /* для Френеля */
//#define FUNC exp(-x*x) /* для Пуассона */
real F(real x) { return FUNC; }
```

На первом этапе каждый исполнитель, узнав свой глобальный номер **i** в массиве всех исполнителей, сам определяет количество и район расположения точек отрезка, назначенных ему для обработки. Для этого он определяет, сколько точек должен обрабатывать каждый исполнитель **n**=**N/P,** и с какой по порядку точки отрезка ему следует их отсчитывать **p**= **i**×**n**. (Заметим, что в случае, когда **N** не делится нацело на **P**, необходимо скорректировать вычисление **n**=**N/P+1** и количество обрабатываемых точек для последнего исполнителя **n**=**N**−**P**.)

Вычисленные во всех точках значения функции необходимо просуммировать. Пусть сначала каждый исполнитель просуммирует вычисленные им значения функции в назначенных ему точках. И полученную сумму **S** сразу умножит на величину шага **h=(B-A)/N**.

Эти итоговые результаты, полученные таким способом каждым исполнителем, снова необходимо просуммировать. И процесс такого суммирования тоже желательно распараллелить. Для этого объединим исполнителей (ОЭ) в рабочие группы. В локальной памяти каждой такой группы выделим место для массива (**L**), в который поручим каждому исполнителю этой группы записать полученный им итоговый результат на позицию, соответствующую его локальному номеру **j** (в этой группе). Дождёмся, когда все исполнители группы совершат такую запись. И после этого поручим одному из исполнителей группы (с номером **j**=0) просуммировать все значения массива **L** и записать итоговый результат всей группы в глобальный массив **Sum** на позицию, соответствующую номеру данной рабочей группы **g**.

Такой алгоритм действий оформим в виде процедуры ядра **_Intgrl**, которая будет запускаться на каждом исполнителе на 1-ом этапе:

```
__kernel void _Intgrl
(int N, real A, real B,
 __global real *Sum, __local real *L )
{ int i,k,n,p; real x,h,S,Ai ;
 P=get_global_size(0); // кол-во всех исполнителей
 i=get_global_id(0);// глоб.номер исполнителя
 j=get_local_id(0); //его локал.номер в Раб.Группе
 g=get_group_id(0); // номер Раб.Группы
 WS=get_local_size(0); // кол-во эл-тов в Раб.Группе
 n= N/P; if((N%P)>0) n=n+1;
// n= кол-во точек для каждого исполнителя
 p= i*n; // номер его стартовой точки
 if((n+p)>N) { n=N-p; if(n<0) n=0; }
//n=кол-во точек,обрабатываемых этим исполнителем
 h=(B-A)/N; S=0.0; Ai= A+p*h;
 for (k=0; k<n; k++)
 { x=Ai+k*h; S= S + F(x); }
 L[j]= S*h;
 barrier(CLK_LOCAL_MEM_FENCE);
 if (j==0) { S= L[0];
  for (k=1; k<WS; k++ ) S= S+L[k];
  Sum[g]= S;
 }//if j==0
}//_Intgrl
```

Для завершения вычисления значения интеграла в среде OpenCL осталось выполнить (на 2-ом этапе) процедуру ядра **_Sum**, чтобы просуммировать значения, которые были записаны в массив **Sum** на 1-ом этапе:

```
__kernel void _Sum (int K,
__global real *Sum,__global real *Res)
{ int i; real S; S= 0.0;
 for (i=0; i<K; i++) S= S+Sum[i];
 *Res= S;
}//_Sum
```

Эти процедуры ядра вместе с необходимыми макроопределениями сосредоточим в файле “Intgrl.cl”, строки которого будут анализироваться при компоновке программного кода ядра в основной программе.

3.2. Организация запуска процедур ядра в основной программе

Предположим, что в основной программе уже осуществлены все необходимые действия по подготовке OpenCL-среды и в глобальной памяти OpenCL-устройства приготовлены объекты для представления массива **Sum** и результата интегрирования **Res**:

```
cl_uint szM= sizeof(cl_mem);
cl_uint szR= sizeof(real);
cl_mem memSum=clCreateBuffer(context,
CL_MEM_READ_WRITE, Q*szR, NULL, NULL);
cl_mem memRes=clCreateBuffer(context,
CL_MEM_READ_WRITE, szR, NULL, NULL);
```

А также образован объект **prgrm**, содержащий в откомпилированной форме код всех процедур ядра, и для каждой процедуры ядра создан дескриптор:

```
cl_kernel knI, knS;
 knI=clCreateKernel (prgrm, "_Intgrl",NULL);
 knS=clCreateKernel (prgrm, "_Sum", NULL);
```

Теперь для выполнения в среде OpenCL операции вычисления интеграла оформим функцию **clIntgrl** (с такими же параметрами, как и у Си-функции **Intgrl**, описанной в п.2):

```
real clIntgrl(int N, real A, real B)
```

Для осуществления операции вычисления интеграла в среде OpenCL в теле этой функции предусмотрим последовательность действий из следующих 5-ти частей:

1. Назначение 5-ти фактических параметров для процедуры ядра **_Intgrl**:

```
cl_uint szI= sizeof(int);
clSetKernelArg(knI,0,szI,&N);
clSetKernelArg(knI,1,szR,&A);
clSetKernelArg(knI,2,szR,&B);
clSetKernelArg(knI,3,szM,&memSum);
clSetKernelArg(knI,4, szR*WS, NULL);
```

2. Запуск процедуры ядра **_Intgrl** в среде OpenCL на множестве из **P** исполнителей, сосредоточенных в рабочих группах по **WS** элементов в каждой, с ожиданием её завершения всеми исполнителями:

```
size_t gWS[1]={P}; size_t lWS[1]={WS};
cl_event evI;
clEnqueueNDRangeKernel(cmndQ,
 knI,1,NULL,gWS,lWS,0,NULL,&evI);
clFinish(cmndQ);
```

3. Назначение 3-х фактических параметров для процедуры ядра **_Sum**:

```
clSetKernelArg(knS,0,szI,&Q);
clSetKernelArg(knS,1,szM,&memSum);
clSetKernelArg(knS,2,szM,&memRes);
```

4. Запуск процедуры ядра **_Sum** на одном исполнителе с ожиданием её завершения:

```
size_t gWS[0]={1}; cl_event evS;
clEnqueueNDRangeKernel(cmndQ,
 knS,1,NULL,gWS,NULL,0,NULL,&evS);
clFinish(cmndQ);
```

5. Выгрузка полученного результата интегрирования из объекта **memRes**:

```
real IntRes;
clEnqueueReadBuffer(cmndQ,
 memRes,CL_TRUE,0,szR,&IntRes,0,0,0);
 return IntRes;
```

4. Результаты ускорения операции вычисления интегралов в среде OpenCL

Чтобы оценить, насколько удалось ускорить операцию вычисления интеграла с помощью применения технологии OpenCL, была составлена программа (на языке Си), которая выполняла операцию вычисления интеграла два раза. Сначала она вызывала функцию **Intgrl** для обычного (последовательного) вычисления интеграла основным процессором, а затем вызывала функцию **clIntgrl,** чтобы подготовить OpenCL-среду и выполнить в ней ту же операцию множеством параллельно функционирующих вычислительных ядер OpenCL-устройства.

Данная программа была разработана как консольное приложение в среде MS Visual Studio 2017. Она компоновалась с различными вариантами задания define-переменных **FUNC** и **real** с тем, чтобы настроить её на вычисление различных интегралов с одинарной и двойной точностью представления вещественных чисел.

Программа запускалась в различных вариантах компоновки на двух разных платформах. Под ОС Windows-10 на аппаратной конфигурации, содержащей основной процессор CPU Intel-i59400 с частотой 2.9 Ггц и встроенный графический процессор GPU UHD 630 с частотой 350 МГц («платформа **Intel**»). И под ОС Windows-7 на аппаратной конфигурации, содержащей процессор Intel i3-2100 с частотой 3.1 Ггц и видеокарту NVidia GeForce 1050ti с частотой 1392 Мгц («платформа **NVidia**»).

Варианты программы, скомпонованные для вещественных чисел одинарной точности, многократно прогонялись для вычисления представленных ранее интегралов Френеля и Пуассона на обозначенных отрезках с заданием различного количества точек **N**. При этом замерялись усреднённые показатели времени $\mathbf{T_{CPU}}$ и $\mathbf{T_{CL}}$, затраченные на выполнение функций **Intgrl** и **clIntgrl**, а также вычислялся коэффициент ускорения как отношение $\mathbf{T_{CPU}/T_{CL}}$.

Результаты ускорения, полученные OpenCL-программой вычисления интегралов для платформы Intel, представлены в таблицах 1-2.

Таблица 1. Ускорение операции вычисления интеграла Френеля в среде OpenCL на платформе Intel (для P=2048,WS=128 на [-5.0,+5.0] = 1.0558)

R	$N=2^R$	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10	1024	0.047454	0.218957	0.023915	0.217	1.9843
11	2048	0.094304	0.229216	0.024582	0.411	3.8363
12	4096	0.187053	0.211167	0.025249	0.886	7.4083
13	8192	0.377172	0.212242	0.026499	1.777	14.233
14	16384	0.769306	0.235982	0.028582	3.260	26.916
15	32768	1.519438	0.240124	0.032749	6.328	46.397
16	65536	3.035782	0.248154	0.042666	12.23	71.152
17	131072	5.895864	0.275902	0.060915	21.37	96.788
18	262144	11.860596	0.309114	0.097832	38.37	121.23
19	524288	23.913322	0.387304	0.172248	61.74	138.83
20	1048576	47.874524	0.571722	0.321914	83.74	148.72
21	2097152	95.581944	0.959715	0.620829	99.59	153.96
22	4194304	188.21317	1.557385	1.218161	120.9	154.51
23	8388608	382.50166	2.754440	2.413906	138.9	158.46
24	16777216	774.74099	5.146215	4.833396	**150.5**	**160.29**

T_{CPU} – время исполнения на CPU функции **Intgrl**
T_{CL} – время исполнения функции **clIntgrl** в OpenCL
$T_{Я}$ – время исполнения OpenCL-ядер
(все времена даны в миллисекундах)
K1 – общий коэффициент ускорения = T_{CPU} / T_{CL}
K2 – «чистый» коэффициент ускорения = T_{CPU} / $T_{Я}$

Таблица 2. Ускорение операции вычисления интеграла Пуассона в среде OpenCL на платформе Intel (для P=2048,WS=128 на [-5.0,+5.0] = 1.7724)

R	$N=2^R$	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10	1024	0.042415	0.221982	0.023332	0.191	1.8179
11	2048	0.083851	0.210888	0.023999	0.398	3.4939
12	4096	0.166911	0.211253	0.024332	0.790	6.8597
13	8192	0.334737	0.216543	0.025082	1.546	13.346
14	16384	0.677476	0.250684	0.026665	2.703	25.407
15	32768	1.380362	0.243700	0.029665	5.664	46.532
16	65536	2.653926	0.242956	0.035999	10.92	73.722
17	131072	5.475312	0.259368	0.055832	21.11	98.068
18	262144	10.887064	0.279966	0.073748	38.89	147.63
19	524288	21.416330	0.349006	0.123498	61.36	173.41
20	1048576	43.283448	0.432322	0.223498	100.1	193.66
21	2097152	87.152944	0.719005	0.427664	121.2	203.79
22	4194304	173.91973	1.167820	0.831829	148.9	209.08
23	8388608	344.49520	1.949570	1.642242	176.7	209.77
24	16777216	693.86598	3.610980	3.259735	**192.2**	**212.86**

T_{CPU} – время исполнения на CPU функции **Intgrl**
T_{CL} – время исполнения функции **clIntgrl** в OpenCL
$T_{Я}$ – время исполнения OpenCL-ядер
(все времена даны в миллисекундах)
K1 – общий коэффициент ускорения = T_{CPU} / T_{CL}
K2 – «чистый» коэффициент ускорения = T_{CPU} / $T_{Я}$

Из этих таблиц видно, что с помощью применения технологии OpenCL операцию вычисления интегралов удаётся ускорить ещё в большей степени, чем операции перемножения матриц и быстрого преобразования Фурье. Так, например, вычисление интеграла Френеля на отрезке [-5.0,+5.0] с разбиением его на $N=2^{24}$ частей в среде OpenCL ускоряется в **150** раз, а вычисление интеграла Пуассона – в **192** раза. Для сравнения напомним, что умножение матриц с таким же количеством элементов ($2048\times2048=2^{24}$) на платформе Intel удавалось ускорить лишь в 15 раз, а операцию быстрого преобразования Фурье (БПФ) для векторов длины $N=2^{24}$ – в 32 раза.

А с применением специализированной видеокарты операцию вычисления интегралов можно ускорить ещё в большей степени, что подтверждают результаты прогона той же OpenCL-программы для платформы NVidia, представленные в таблицах 3-4.

Таблица 3. Ускорение операции вычисления интеграла Френеля в среде OpenCL на платформе NVidia (для P=2048,WS=128 на [-5.0,+5.0] = 1.0558)

R	$N=2^R$	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10	1024	0.060003	0.201311	0.009088	0.298	6.6024
11	2048	0.120007	0.216712	0.008384	0.554	14.314
12	4096	0.240013	0.217312	0.007744	1.104	30.993
13	8192	0.490028	0.217512	0.008608	2.253	56.927
14	16384	0.960056	0.221812	0.009856	4.328	97.408
15	32768	1.920110	0.210412	0.011200	9.125	171.44
16	65536	3.880222	0.233813	0.012384	16.59	313.32
17	131072	7.860450	0.227613	0.017728	34.53	443.39
18	262144	15.540888	0.240213	0.027136	64.69	572.70
19	524288	31.001774	0.244814	0.044224	126.6	701.02
20	1048576	61.963544	0.296616	0.080928	208.9	765.66
21	2097152	124.00709	0.368021	0.127744	336.9	970.75
22	4194304	248.56421	0.465526	0.228480	533.9	1087.9
23	8388608	496.12838	0.704540	0.449536	704.2	1103.6
24	16777216	995.85696	1.197568	0.891712	**831.6**	**1116.8**

Таблица 4. Ускорение операции вычисления интеграла Пуассона в среде OpenCL на платформе NVidia (для P=2048,WS=128 на [-5.0,+5.0] = 1.7724)

R	$N=2^R$	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10	1024	0.050003	0.213812	0.008480	0.234	5.8966
11	2048	0.110006	0.211612	0.008928	0.520	12.322
12	4096	0.230013	0.213912	0.008480	1.075	27.124
13	8192	0.450025	0.208111	0.009184	2.162	49.001
14	16384	0.960054	0.216012	0.007904	4.444	121.46
15	32768	1.820104	0.226412	0.009344	8.039	194.79
16	65536	3.600206	0.230813	0.010208	15.59	352.68
17	131072	7.160410	0.220812	0.011488	32.43	623.29
18	262144	14.280818	0.210812	0.014400	67.74	991.72
19	524288	28.961656	0.249014	0.021504	116.3	1346.8
20	1048576	57.323280	0.264815	0.033792	216.5	1696.4
21	2097152	115.50661	0.277015	0.060416	416.9	1911.8
22	4194304	231.16323	0.280016	0.095520	825.5	2420.1
23	8388608	461.52640	0.399522	0.152544	1155	3025.5
24	16777216	925.10291	0.521029	0.295296	**1775**	**3132.8**

В этих таблицах отмечено, что вычисление интеграла Френеля на том же отрезке [-5.0,+5.0] с разбиением на $N=2^{24}$ частей в среде OpenCL ускоряется в **831** раз, а вычисление интеграла Пуассона – в **1775** раза. Эти результаты можно сравнить с показателями ускорения на той же платформе (NVidia) операции перемножения

матриц такого же размера (2^{24}) и операции быстрого преобразования Фурье для векторов той же длины, упомянутых ранее (95 и 190 раз).

5. Заключение

Полученные результаты практически подтверждают, что операция вычисления интегралов, не требующая перекачки большого объёма данных между основной памятью и памятью OpenCL-устройства, в лучшей степени пригодна для её ускорения с помощью применения технологии OpenCL.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения», Рег. № 122041100063-0.

Applying OpenCL Technology to Accelerating the Calculation of Integrals

A.A. Burtsev

Abstract. The article focuses on the use of OpenCL technology, which allows you to use powerful GPU resources to improve the performance of computing programs. Variants of parallel programs designed to accelerate the operation of calculating integrals in the OpenCL environment are considered.

Keywords: parallel programming, OpenCL technology, heterogeneous systems, numerical methods for calculating integrals.

Литература

1. В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. Спб., БХВ-Петербург, 2004.

2. Официальный OpenCL–сайт организации Khronos Group, http://www.khronos.org/opencl/

3. А.А. Бурцев. Оптимизация операции перемножения матриц на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, 100–112.

4. А.А. Бурцев. Ускорение быстрого преобразования Фурье на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 11 (2021), № 4, 27–37.

5. А.А. Бурцев. Оптимизация операции быстрого преобразования Фурье в среде OpenCL. // «Труды НИИСИ РАН», Т.12 (2022), №1-2, 11-27.

6. Неберущиеся интегралы, https://www.matematicus.ru/vysshaya-matematika/integralnoe-ischislenie/neberushhiesya-integraly