Расширение DVMH-модели для работы с нерегулярными сетками*

В.А. Бахтин[12], А.С. Колганов[12], В.А. Крюков[12], Н.В. Поддерюгина [1], С.В. Поляков [1], М.Н. Притула [1]
Институт Прикладной Математики им. М.В. Келдыша РАН [1], Московский Государственный Университет им. М.В. Ломоносова[2]

Параллельные вычислительные технологии (ПаВТ'2016)
Parallel Computational Technologies (PCT'2016)

* Работа поддержана программой Президиума РАН No II.4П и грантом РФФИ 16-07-01067 А.

Модель DVMH

Модель параллельного программирования DVMH построена на парадигме параллелизма по данным. В основе этой модели лежит понятие распределенного многомерного массива. При этом у каждого процессора имеется не только локальная часть распределенного массива, но и так называемые теневые грани – копии элементов из локальных частей соседних процессоров, через которые осуществляется эффективное взаимодействие процессоров. Распределение вычислений производится посредством их отображения на распределенные массивы.

Модель DVMH позволяет при помощи небольших изменений последовательного Фортран или Си кода распараллелить программу на ЦПУ+ГПУ кластер и подходит в первую очередь для написания параллельных программ на регулярных прямоугольных сетках, но и некоторые виды программ на нерегулярных сетках возможно распараллелить имеющимися средствами.

Применение модели DVMH для задачи на треугольной сетке

Была распараллелена задача теплопроводности на нерегулярной треугольной сетке на кластер с ГПУ.

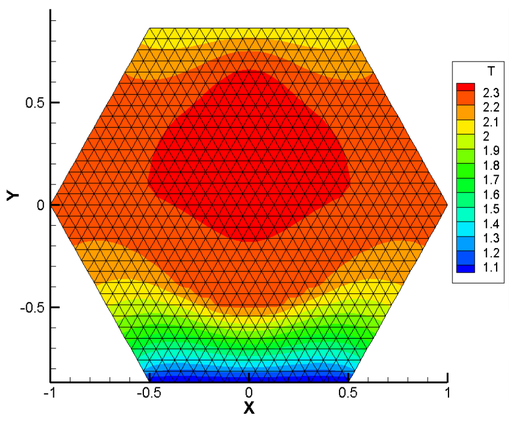

Схема	Ускорение на ЦПУ		
	12 ядер	24 ядра	96 ядер
Явная	8,02	15,07	63,3
Неявная	3	6	21,7

Схема	Ускорение на ГПУ		
	3 ГПУ	6 ГПУ	24 ГПУ
Явная	63	116	316
Неявная	31	57	142

Недостатки полученного варианта распараллеливания:

- использование механизма теневых граней приводит к избыточности обменов и потребления памяти;
- нетривиальное определение ширины теневых граней ложится на пользователя;
- необходимость специального упорядочения (нумерации) сеточных элементов;
- распределение только блочное и принципиально связано с нумерацией;
- невозможность задания согласованного распределения сеточных элементов (ячейки, узлы, ребра).

Расширение DVMH-модели

Предлагается вариант расширения модели DVMH, который бы, с одной стороны, органично вписывался в существующую модель DVMH, дополняя ее конструкции, а с другой стороны позволял бы снять известные проблемы и ограничения при распараллеливании задач на нерегулярных сетках, причем не потеряв значительно в эффективности параллельного выполнения.

Новые возможности расширения модели DVMH для работы с нерегулярными сетками:

- задание произвольных поэлементных распределений, в том числе получаемые пакетами Metis, Chaco, …;
- построение согласованных распределений на основе имеющихся (блочных или поэлементных);
- задание произвольных по содержанию буферов удаленных элементов с эффективным однородным доступом к ним и обновлением;
- возможность реорганизации данных – оптимизации шаблона доступа к памяти путем изменения порядка хранения локальных элементов;
- сохранение быстрого доступа к распределенным массивам с помощью механизмов перехода на локальную индексацию.

Шаблон типа крест на регулярной двумерной сетке

```
float B[N][N];
float A[N][N];
...
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    B[i][j] = (A[i-1][j]+A[i][j-1]
      + A[i][j+1]+A[i+1][j])/4.0f;
  }
...
```

```
#pragma dvm array distribute[block][block]
float B[N][N];
#pragma dvm array align([i][j] with B[i][j]) \
  shadow[1:1][1:1]
float A[N][N];
...
#pragma dvm parallel([i][j]on B[i][j])shadow_renew(A)
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    B[i][j] = (A[i-1][j]+A[i][j-1]
      + A[i][j+1]+A[i+1][j])/4.0f;
  }
...
```

Вычисления на нерегулярной сетке, применение исходной DVMH-модели

```
float B[N];
float A[N];
int E[N][K];
float W[N][K];
...
for (i = 0; i < N; i++) {
  float v = 0;
  for (j = 0; j < K; j++) {
    if (E[i][j] >= 0)
      v += W[i][j] * A[E[i][j]];
  }
  B[i] = v;
}
...
```

```
#pragma dvm array distribute[block]
float B[N];
#pragma dvm array align([i] with B[i]) shadow[??:??]
float A[N];
#pragma dvm array align([i][] with B[i])
int E[N][K];
#pragma dvm array align([i][] with B[i])
float W[N][K];
...
#pragma dvm parallel([i] on B[i]) shadow_renew(A)
for (i = 0; i < N; i++) {
  float v = 0;
  for (j = 0; j < K; j++) {
    if (E[i][j] >= 0)
      v += W[i][j] * A[E[i][j]];
  }
  B[i] = v;
}
...
```

Вычисления на нерегулярной сетке, простой пример для расширения

```
float B[N];
float A[N];
int E[N][K];
float W[N][K];
...
for (i = 0; i < N; i++) {
  float v = 0;
  for (j = 0; j < K; j++) {
    if (E[i][j] >= 0)
      v += W[i][j] * A[E[i][j]];
  }
  B[i] = v;
}
...
```

```
#pragma dvm array distribute[block]
float B[N];
#pragma dvm array align([i] with B[i])
float A[N];
#pragma dvm array align([i][] with B[i])
int E[N][K];
#pragma dvm array align([i][] with B[i])
float W[N][K];
int map[N];
...
#pragma dvm redistribute(B[indirect(map)])
#pragma dvm add_shadow(A([E[i][]] with B[@i]))
#pragma dvm tie(B[:], A[:], E[:][], W[:][])
#pragma reference_region reference(A[:]<= E except(<0))
{
#pragma dvm parallel([i] on B[i]) shadow_renew(A) byref
for (i = 0; i < N; i++) {
  float v = 0;
  for (j = 0; j < K; j++) {
    if (E[i][j] >= 0)
      v += W[i][j] * A[E[i][j]];
  }
  B[i] = v;
}
}
...
```

http://dvm-system.org/

Примеры использования новых конструкций

Согласованное (производное) распределение вершин и ячеек

```
#pragma dvm array distribute[indirect(part)]
float cells[Nc];
#pragma dvm array align([i][] with cells[i])
int vertOfCell[Nc][3];
#pragma dvm array distribute[derived( [vertOfCell[i][0:2]] \
  with cells[@i] overlay=own )]
float vert[Nv];
```

Описание распределенной матрицы в формате CSR

```
#pragma dvm array distribute[indirect(partition)]
int rowBegin[M];
#pragma dvm array align([i] with rowBegin[i])
int rowEnd[M];
#pragma dvm array distribute[derived([rowBegin[i]:rowEnd[i]]\
  with rowBegin[@i])]
int column[K];
#pragma dvm array align([i] with column[i])
float matrix[K];
```

Поэлементное распределение, полученное с помощью METIS

```
int part[N];
METIS_PartGraphKway(..., dvmh_get_num_procs(1), ..., part);
#pragma dvm array distribute[indirect(part)]
float A[N];
```

Поэлементное распределение, заданное распределенным массивом

```
#pragma dvm array distribute[block]
int map[N];
fread(map, sizeof(int) * N, 1, f);
#pragma dvm array distribute[indirect(map)]
float A[N];
```

Задание поэлементных теневых граней (соседей)

```
#pragma dvm array distribute[indirect(partition)]
float cells[Nc];
#pragma dvm array align([i][] with cells[i])
int vertOfCell[Nc][3], cellNeigh[Nc][3];
#pragma dvm array distribute[derived( [vertOfCell[i][0:2]]\
  with cells[@i] overlay=over )]
float vert[Nv];
#pragma dvm add_shadow(vert( [vertOfCell[cellNeigh[i][0:2]][0:2]]\
  with cells[@i] name=neigh ))
```