Федеральное государственное бюджетное учреждение науки
Институт теоретической и прикладной механики
им. С.А. Христиановича
Сибирского отделения Российской академии наук

Приоритетное направление III.22. Механика жидкости, газа и плазмы, многофазных и неидеальных сред, механика горения, детонации и взрыва.

III.22.5.2. Математическое моделирование в современных задачах аэротермодинамики высокоскоростного полета.

Тема: Развитие новых методов эффективного численного моделирования задач аэротермодинамики

Структура оператора столкновений в нелинейном кинетическом уравнении Больцмана обуславливает большие вычислительные затраты при его численном решении. Поэтому проводить расчеты двумерных течений разреженного газа с высоким разрешением на основе решения уравнения Больцмана за приемлемое время возможно только на гибридных кластерах с узлами, содержащими наряду с центральными процессорами графические ускорители. Соответственно, целесообразно расширить модель программирования MPI включением технологии CUDA или иной технологии SIMD параллелизма, поддерживаемой графическими процессорами. На кластере НГУ нами отрабатывается гибридная модель параллельного программирования, используемая при расчетах течений разреженного газа.

Основные модули программы написаны на языке FORTRAN. Некоторые из этих модулей являются универсальными и их исходный код может использоваться в программных пакетах, реализующих различные математические модели газовой динамики. Эффективность этого кода подтверждена многими расчетами. С другой стороны разработку нового модуля для расчета интеграла столкновений опирающегося на интерфейс программирования CUDA целесообразно проводить с использованием языков C/C++. В частности потому, что компилятор nvcc компании Nvidia, генерирующий надёжный и производительный код для графических процессоров, распространяется бесплатно, в то время, как единственный компилятор CUDA Fortran компании PGI имеет платную лицензию. В связи с тем, что разные части программы должны компилироваться разными компиляторами, используется динамическая компоновка двоичных модулей. Для этого код, выполняющий вычисление интеграла столкновений на графическом процессоре, реализован в виде разделяемой библиотеки, загружаемый в процессы MPI динамически (рис.1) Для согласования имен на двоичном уровне экспортируемые из библиотеки функциидолжны быть объявлены с модификатором ``extern C``, чтобы избежать декорирования имени, и при компиляции главного модуля компилятором fortran следует использовать опцию -fno-underscoring.

Предварительная декомпозиция по данным заключается в том, чтокаждому MPI процессу выделяется подобласть физического пространства

, содержащая N узлов расчетной сетки. Для этих подобластей, без распараллеливания по пространству скоростей, проводится расчет свободномолекулярного переноса. Для вычисления итеграла столкновений подобласть физического пространства, предназначенная для каждого MPI процесса, делится на меньшие равные подобласти с N/NG узлами, где NG равно количеству свободных графических процессоров на узле.Фрагментом декомпозиции при распараллеливании этих вычислений является область фазового пространства, содержащая Nvx*Nvy*Nvz*N/NG узлов, где Nvx, Nvy, Nvz — размерности расчетной сетки в скоростном пространстве.

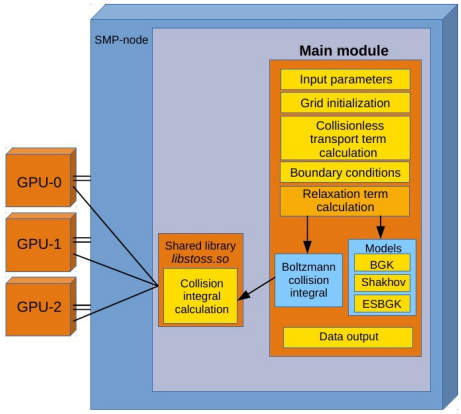

Рис 1 — Главные компоненты приложения

На рисунке 2 представлена диаграмма активности, иллюстрирующая выполнение программы в рамках одного процесса MPI. На рисунке 3 представлена диаграмма активности вычисления интеграла столкновений на одном GPU.

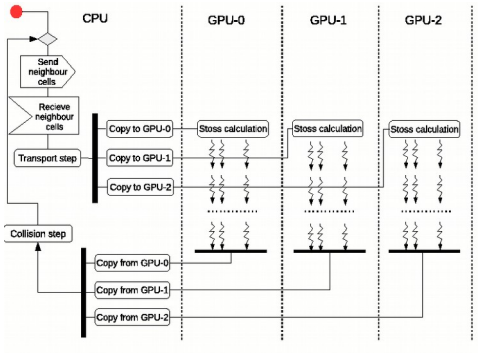

Рис 2 - Диаграмма активности одного MPI процесса.

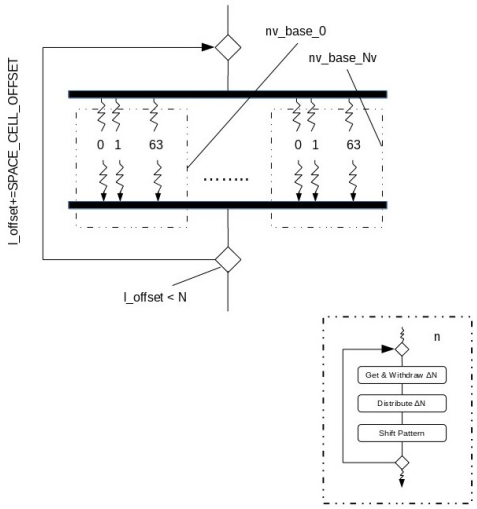

Рис 3 — Диаграмма активности вычисления интеграла столкновений.

В листинге 1 представлены фрагменты кода, реализующего запуск ядра для вычисления интеграла столкновений на нескольких устройствах. Следует отметить, что при распараллеливании вычислений на нескольких GPU мы не используем потоки потоки выполнения на центральном процессоре, а используем потоки CUDA (CUDA Stream) для асинхронного вызова ядер.

Листинг 1

```
#ifdef COMPUTE_CAPABILITY_3
#define REAL double
#else
#define REAL float
#endif

#define SPACE_CELL_OFFSET 64

int NG;
int* assigned_devices;
REAL** Df_device;
REAL** St_device
.....
extern "C"
void init_stoss(GridParams* g, int* info_devs, int &errorCode){
  .....
  NG=info_devs[0];

  assigned_devices=(int*)calloc(NG, sizeof(int));
  for(int idev=0;idev<NG;idev++)
    assigned_devices[idev]=info_devs[idev+1];
  .....
  REAL** Df_device=(REAL**)calloc(NG, sizeof(REAL*));
  REAL** St_device=(REAL**)calloc(NG, sizeof(REAL*));

  for(int idev=0;idev< NG; idev++){
    cudaSetDevice(assigned_devices[idev]);
    cudaMalloc((void **) &Df_device[idev], size_of_Df/NG);
    cudaMalloc((void **) &St_device[idev], size_of_Df/NG);
  }
  .....
}
```

```
extern "C"
void stoss(double *Df, double *St, int &errorCode){

/*
Копирование значений функции распределения в подобластях
декомпозиции физического пространства на GPU, соответственно
их количеству NG:
*/
  for(int idev=0;idev < NG; idev++){
    cudaSetDevice(assigned_devices[idev]);
    cudaMemcpy(Df_device[idev], Df+idev*nvx*nvy*nvz*N/NG,
                size_of_Df/NG, cudaMemcpyHostToDevice);
    }

  for(int l_offset=0; l_offset < N/NG; l_offset+=SPACE_CELL_OFFSET){

//А с и н х р о н н ы й  запуск ядра для вычисления интеграла столкновений:
    for(int idev=0;idev< NG; idev++){
      cudaSetDevice(assigned_devices[idev]);
      cudaEventRecord( mdEventStart[idev], 0 );
      gStossCalc<<<dim3(NumOfBlocks), dim3(SPACE_CELL_OFFSET))>>>
      (Df_device[idev], St_device[idev], l_offset);
    }
    for(int idev=0;idev< NG; idev++){
      cudaSetDevice(assigned_devices[idev]);
      cudaEventRecord( mdEventStop[idev], 0);
      cudaEventSynchronize(mdEventStop[idev]);
    }
  }
/*
Копирование результатов вычисления интеграла столкновений
с каждого GPU в память вычислительного узла.
*/
    for(int idev=0;idev< NG; idev++){
      cudaSetDevice(assigned_devices[idev]);
      cudaMemcpy(St+idev*nvx*nvy*nvz*N/NG, St_device[idev],
                  size_of_Df/NG, cudaMemcpyDeviceToHost);
    }
}
```

ПУБЛИКАЦИИ

1. E. Malkov, A. Kokhanchik, S. Poleshkin, Y. Bondar High-accuracy deterministic solution of the Boltzmann equation for the shock wave structure // Shock Waves (*Impact factor: 0.743*), 2015, V. 25, Issue 4, pp 387–397.
2. E. A. Malkov, S. O. Poleshkin, A. N. Kudryavtsev and A. A. Shershnev Numerical approach for solving kinetic equations in two-dimensional case on hybrid computational clusters //AIP Conference Proceedings V. 1770, 030077 (2016); doi: http://dx.doi.org/10.1063/1.4964019