# Стандарт директивного программирования OpenACC

# OpenACC

- Набор директив для написания гетерогенных программ, задействующих как центральный, так и графический процессор.
- Используется для распараллеливания программ на языках C, C++ и Fortran.
- Является открытым стандартом.
- CAPS, Cray, NVIDIA и PGI.

OpenACC
DIRECTIVES FOR ACCELERATORS

# OpenACC

- Совместимость компиляторов:
  - разные компиляторы должны поддерживать одинаковый набор директив и библиотек;
- Совместимость устройств:
  - поддержка любых ускорителей от устаревающих до еще не вышедших;
  - разделение кода на CPU и GPU больше не требуется;
- Совместимость производительности:
  - грамотно написанный код одинаково хорошо исполняется как на CPU, так и на GPU.

OpenACC
DIRECTIVES FOR ACCELERATORS

# SAXPY: OpenMP

- Простота;
- Открытый стандарт;
- Высокая производительность.

```
void saxpy(int n, float a, float *x, float *restrict y)
{
   #pragma omp parallel
   for (int i = 0; i < n; i < n)
      y[i] = a * x[i] + y[i];
}

...
// Perform SAXPY on 1M elements
saxpy(1<<20, 2.0, x, y);
...
```

# SAXPY: OpenACC

- Простота;
- Открытый стандарт;
- Высокая производительность.

```
void saxpy(int n, float a, float *x, float *restrict y)
{
   #pragma acc parallel
   for (int i = 0; i < n; i < n)
      y[i] = a * x[i] + y[i];
}

...
// Perform SAXPY on 1M elements
saxpy(1<<20, 2.0, x, y);
...
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Модель исполнения

- CPU:
  - выполняет большую часть программы;
  - выделяет память на ускорителе;
  - инициирует копирование данных из CPU в GPU;
  - отправляет код ядра на ускоритель;
  - устанавливает ядра в очередь для исполнения;
  - ожидает выполнения ядра;
  - инициирует копирование данных из GPU в CPU;
  - освобождает память ускорителя;
- GPU:
  - исполняет ядра;
  - выполняет асинхронную передачу данных;

OpenACC
DIRECTIVES FOR ACCELERATORS

# Модель исполнения

- 3 уровня исполнения:
  - gang (block);
  - worker (warp);
  - vector (threads-in-warp);
- Зависит от компилятора.

# Синтаксис директив

- Fortran:

```
!$acc directive [атрибут [, атрибут] …]
        структурированный блок
!$acc directive
```

- C:

```
#pragma acc directive [атрибут [, атрибут] …]
        структурированный блок
```

- Компиляция (PGI):
  - pgfortran –acc –ta=nvidia,time –Minfo=accel <filename>;
  - pgcc –acc –ta=nvidia,time –Minfo=accel <filename>;

OpenACC
DIRECTIVES FOR ACCELERATORS

# Parallel

- Генерация региона параллельных вычислений:

- parallel:

```
#pragma acc parallel [атрибут [, атрибут] …]
        структурированный блок
```

- kernels (генерация нового ядра для каждого цикла):

```
#pragma acc kernels [атрибут [, атрибут] …]
        структурированный блок
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Атрибуты parallel

- Основные:
  - if (condition);
  - async [(exp)];
  - num_gangs (exp);
  - num_workers (exp);
  - vector_length (exp);
  - reduction (operator:list).

- Атрибуты данных:
  - copy* (list);
  - create (list);
  - present (list);
  - present_or_copy* (list);
  - present_or_create (list);
  - deviceptr (list);
  - private (list);
  - firestprivate (list);

* in | out

OpenACC
DIRECTIVES FOR ACCELERATORS

# Loop

- Указание параметров цикла:

- loop:

```
#pragma acc loop [атрибут [, атрибут] …]
        for loop
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Атрибуты loop

- Дополнительные:
  - collapse (n);
  - async [(exp)];
  - gang (exp);
  - worker (exp);
  - vector (exp);
  - seq;
  - independent;
  - private (list);
  - reduction (operator:list).

OpenACC
DIRECTIVES FOR ACCELERATORS

# Data

- Указание региона заданной операции с данными:

- data:

array[начало : длина]

- Например:

a[2:n] - a[2], a[3], …, a[2+n-1].

OpenACC
DIRECTIVES FOR ACCELERATORS

# Data: пример 1

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int a [10000];
    int b [10000];

    #pragma acc parallel
    for (int i = 0; i < 10000; i++)
    {
        a[i] = i - 100 + 23;
    }

    #pragma acc parallel
    for (int j = 0; j < 10000; j++)
    {
        b[j] = a[j] - j - 10 + 213;
    }

    return 0;
}
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Data: пример 2

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int a [10000];
    int b [10000];

    #pragma acc data copyout (a[0:10000], b[0:10000])
    {
    #pragma acc parallel
    for (int i = 0; i < 10000; i++)
    {
         a[i] = i - 100 + 23;
    }

    #pragma acc parallel
    for (int j = 0; j < 10000; j++)
    {
        b[j] = a[j] - j - 10 + 213;
    }
    }
    return 0;
}
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Остальные директивы

- host_data:
  - делает адрес данных на ускорителе доступным для хоста;
- cache:
  - кэширует данные через программно управляемый кэш;
- update:
  - обновляет существующие данные после их изменения;
- wait:
  - ожидает выполнения асинхронных операций на ускорителе;
- declare:
  - указывает, что необходимо выделить память на ускорителе для использования в рамках региона data.

# OpenACC и OpenMP

```
//Multi-GPU execution with OpenMP:

#include <openacc.h>
#include <omp.h>

#pragma omp parallel num_threads(number_of_gpus)
{
   int id = omp_get_thread_num();
   acc_set_device_num(id, acc_device_nvidia );

   #pragma acc parallel
   {
   //Do something on GPU id;
   }
}
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# OpenACC и MPI

```
//Multi-GPU execution with MPI:

#include <openacc.h>
#include <mpi.h>

int my_rank;
MPI_Comm_rank(MPI_COMM_WORLD, & my_rank);
int num_dev = acc_get_num_devices(acc_device_nvidia);
int id = my_rank % num_dev;
acc_set_device_num(id, acc_device_nvidia );

#pragma acc parallel
{
   //Do something on GPU id;
}
```

OpenACC
DIRECTIVES FOR ACCELERATORS

# Заключение

- Простота понимания:
  - синтаксис аналогичен OpenMP;
- Простота использования:
  - программа для расчетов на видеокарте без единой строчки CUDA;
- Универсальность:
  - поддержка CPU и множества ускорителей;
- Опасность:
  - за целостностью данных необходимо следить самостоятельно.