

Параллельное программирование

Сайт дисциплины:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF>

Варианты заданий:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF/VarsPP2024.htm>

Список задач:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF/Tasks2024.htm>

Курс в системе dispace:

<http://dispace.edu.nstu.ru/didesk/course/show/4872>

Тест в системе DiSpace

Малявко Александр Антонович

E-mail: a.malyavko@corp.nstu.ru

Литература:

- *Малявко А. А. Параллельное программирование на основе технологий OpenMP, CUDA, OpenCL и MPI : учеб. пособие / А. А. Малявко . - 3-е изд., испр. и доп.. - Москва : Юрайт, 2021. - 135 с - (Бакалавр. Академический курс). - ISBN 978-5-534-14116-0.*
- *Малявко А. А. Параллельное программирование на основе технологий OpenMP, MPI, CUDA : учеб. пособие / А. А. Малявко . - 2-е изд., испр. и доп.. - Москва : Юрайт, 2017. - 115 с - (Бакалавр. Академический курс). - ISBN 978-5-534-02916-1.*
- *Малявко А. А. Параллельное программирование на основе технологий OpenMP, MPI, CUDA : учебное пособие. Изд-во НГТУ: Новосибирск, 2015. Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215088*
- *Биллиз В.А. Параллельные вычисления и многопоточное программирование. М: Издательство: ИНТУИТ, 2013.*
- *Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2004.*
- *Баканов В.М. Параллельные вычисления: учеб. пособие, М.: Изд-во МГУПИ, 2006.*
- *Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. — Н.Новгород.: Изд-во ННГУ 2003.*
- *Корнеев В. Д. Параллельное программирование кластеров : учеб. пособие, Новосибирск, Изд-во НГТУ, 2008.*

Практически все это и еще многое находится в электронном виде на сайте

Русскоязычные интернет-ресурсы:

- <http://www.parallel.ru>
- <http://www.openmp.ru>
- http://parallel.ru/tech/tech_dev/mpi.html
- http://www.nvidia.ru/object/cuda_home_new_ru.html
- <http://software.intel.com/ru-ru>
- <http://www2.sssc.ru/>
- <http://www.ops.rsu.ru>
- <http://www.t-platforms.ru/>
- <http://www.ccas.ru/>
- <http://www.jssc.ru/scomputers.shtml>
- <http://www.csa.ru/>
- <http://www.elbrus.ru/>
- <http://www.paracomtech.ru/>
- <http://www.botik.ru/PSI/RCMS/>

Англоязычные интернет-ресурсы:

- <http://www.openmp.org>
- <http://www.mcs.anl.gov/mpi>
- http://www.nvidia.com/object/cuda_home_new.html

Причины и потребности

1. Предел производительности одной последовательно исполняющей инструкции вычислительной машины (ядра) уже достигнут, это примерно 10-80 Gflops в секунду. Поэтому все большее распространение получают многоядерные процессоры и другие виды параллельных вычислительных систем.

2. Существует обширный круг задач, которые требуют значительно большей производительности. Их невозможно решить на одиночной последовательной машине.

Параллельные системы

Существует большое количество различных способов организации параллельных вычислений и, соответственно, архитектур вычислительных систем:

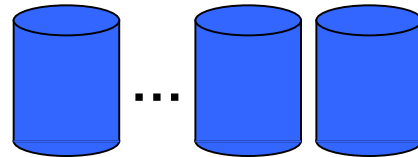
- системы с общей памятью:
 - много потоков команд (но, как правило, одна программа) / много потоков данных (MIMD-SPMD) – технология **OpenMP**, многопоточное программирование в Java, C++, ...;
 - один поток команд / много потоков данных (SIMD) – технологии **OpenCL**, **NVIDIA CUDA**, NVIDIA OpenACC, Microsoft AMP, ...;
- системы с распределенной памятью
 - много потоков команд (одна программа) / много потоков данных (MIMD) – технологии **MPI**, PGAS, ...;

Вычислительный кластер

Интернет,
интранет

Управляющая
подсистема

Система
хранения данных



СХД

СХД

Память

Кэш

Ядро

Ядро

Ядро

ВА
(GPU)

Сеть управления

Сеть синхронизации

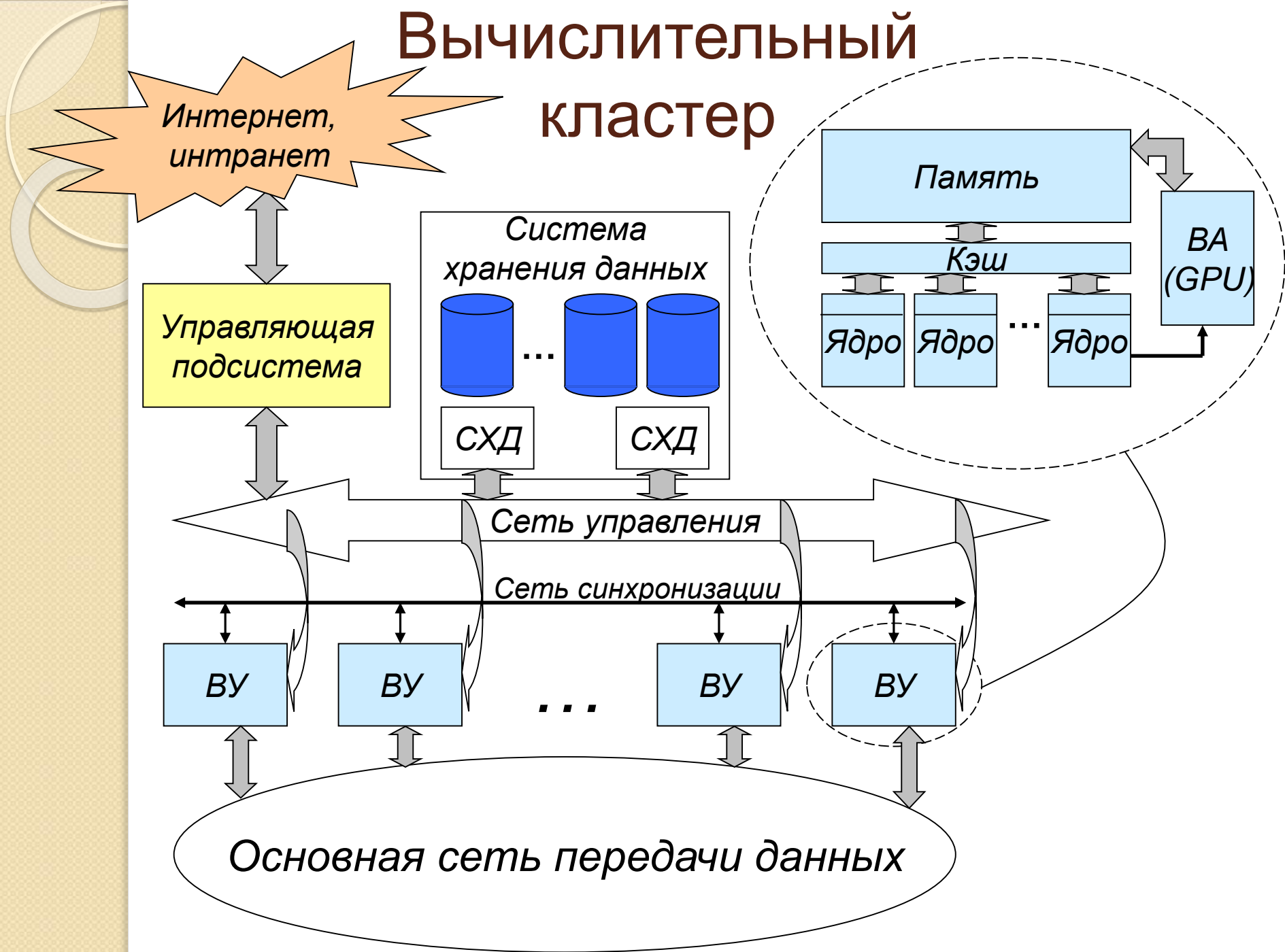
ВУ

ВУ

ВУ

ВУ

Основная сеть передачи данных



Параллельные вычисления

Параллельными вычислениями называется такая организация вычислительного процесса, при которой решением одной задачи одновременно занимается более, чем одно вычислительное устройство (ядра, процессора), каждое из которых выполняет свою собственную часть общей работы.

Большие задачи. Это такие задачи, для решения которых требуется недопустимо долгое выполнение программы на одиночной последовательной машине (месяцы, годы, десятки лет).

Гнезда циклов

Для больших задач типичной является ситуация, когда основной объем вычислений определяется совокупностью вложенных друг в друга циклов:

```
for( i = 0; i < N; i += 1)
    for( j = 0; j < M; j += 1)
        for( k = R - 1; k >= 0 ; k -= 1)
            ...
```

Пространством итераций гнезда тесно вложенных циклов называют множество целочисленных векторов $\{ [0, 0, R-1, \dots], [0, 0, R-2], \dots \}$, координаты которых задаются значениями параметров циклов данного гнезда.

Задача распараллеливания в этом случае сводится к разбиению этого множества векторов на подмножества таким образом, чтобы итерации этих подмножеств могли быть выполнены параллельно.

Распараллеливание циклов

Распараллеливаться может только такой цикл, итерации которого информационно не зависят друг от друга

Это значит, что ни в одной итерации цикла не используются значения, вырабатываемые какой-либо другой итерацией.

Распределение итераций цикла (1)

Пусть в программе есть цикл, между итерациями которого нет информационной зависимости, например:

```
for( i = 0; i < N; i += 1)
```

```
    arr[i] = <выражение, не содержащее arr[ k ] (k != i)>;
```

Итерации этого цикла могут быть выполнены в любом порядке, в том числе – одновременно друг с другом.

Для параллельной системы из n вычислительных элементов (процессоров) может быть реализовано:

1. **Блочное распределение** по N/n итераций (если N не делится нацело на n , то загрузка процессоров будет неодинаковой).

```
for( i = 0; i < N0; i += 1)           //доля процессора 0, N0 = N / n
```

```
    arr[i] = f( i, ...);
```

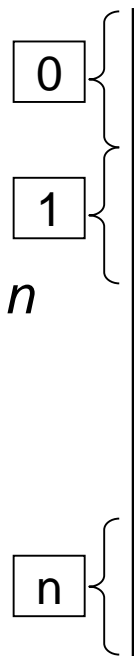
```
for( i = N0; i < N1; i += 1)         //доля процессора 1, N1 = 2*N / n
```

```
    arr[i] = f( i, ...);
```

...

```
for( i = Nk; i < N; i += 1)           //доля процессора n-1, ...
```

```
    arr[i] = f( i, ...);
```



Распределение итераций цикла (2)

2. **Циклическое распределение** тоже по N / n итераций

for($i = 0; i < N; i += n$)

//доля процессора 0

arr[i] = f(i, ...);

for($i = 1; i < N; i += n$)

//доля процессора 1

arr[i] = f(i, ...);

...

for($i = n-2; i < N; i += n$)

//доля процессора $n-1$

arr[i] = f(i, ...);

3. **Блочно-циклическое распределение:**

for ($k = 0; k < N; k += m$)

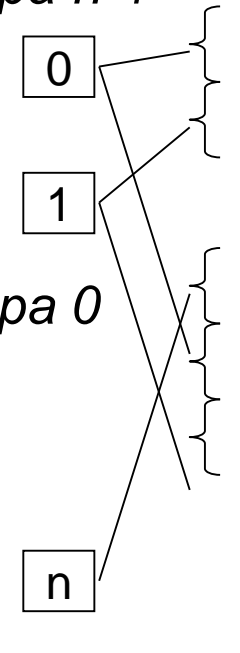
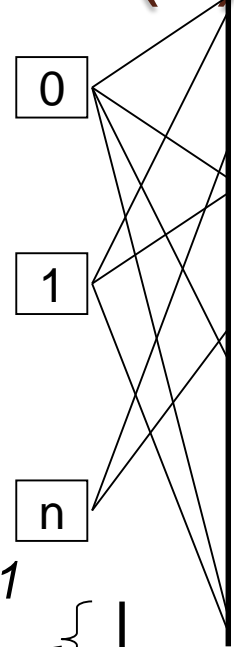
*for($i = k * n; i < (k+1) * n; i += 1$) //доля процессора 0*

arr[i] = f(i, ...);

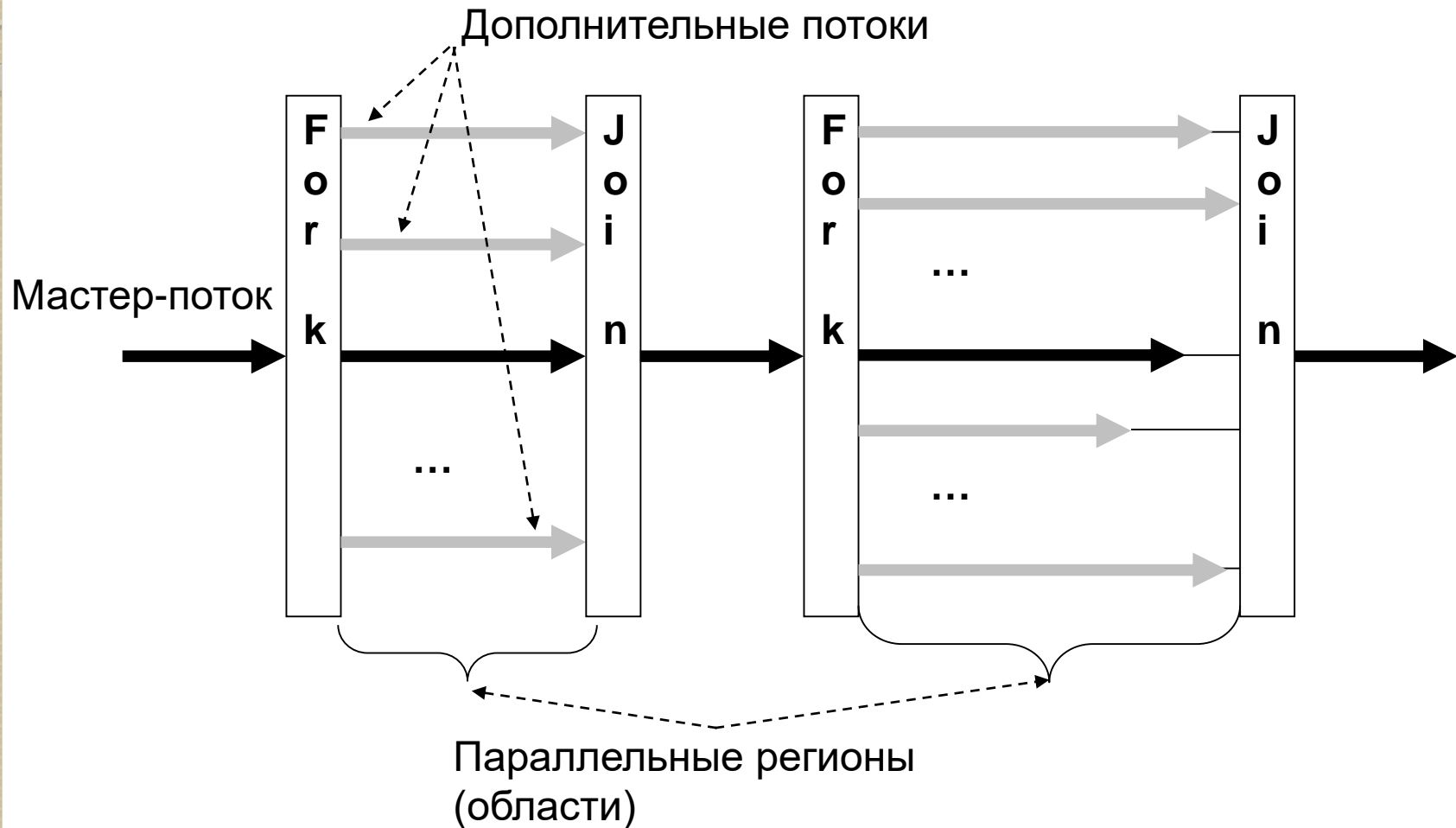
for ($k = 0; k < N; k += m$)

*for($i = (k+1) * n; i < (k+2) * n; i += 1$)*

arr[i] = f(i, ...);



Система параллельного программирования OpenMP



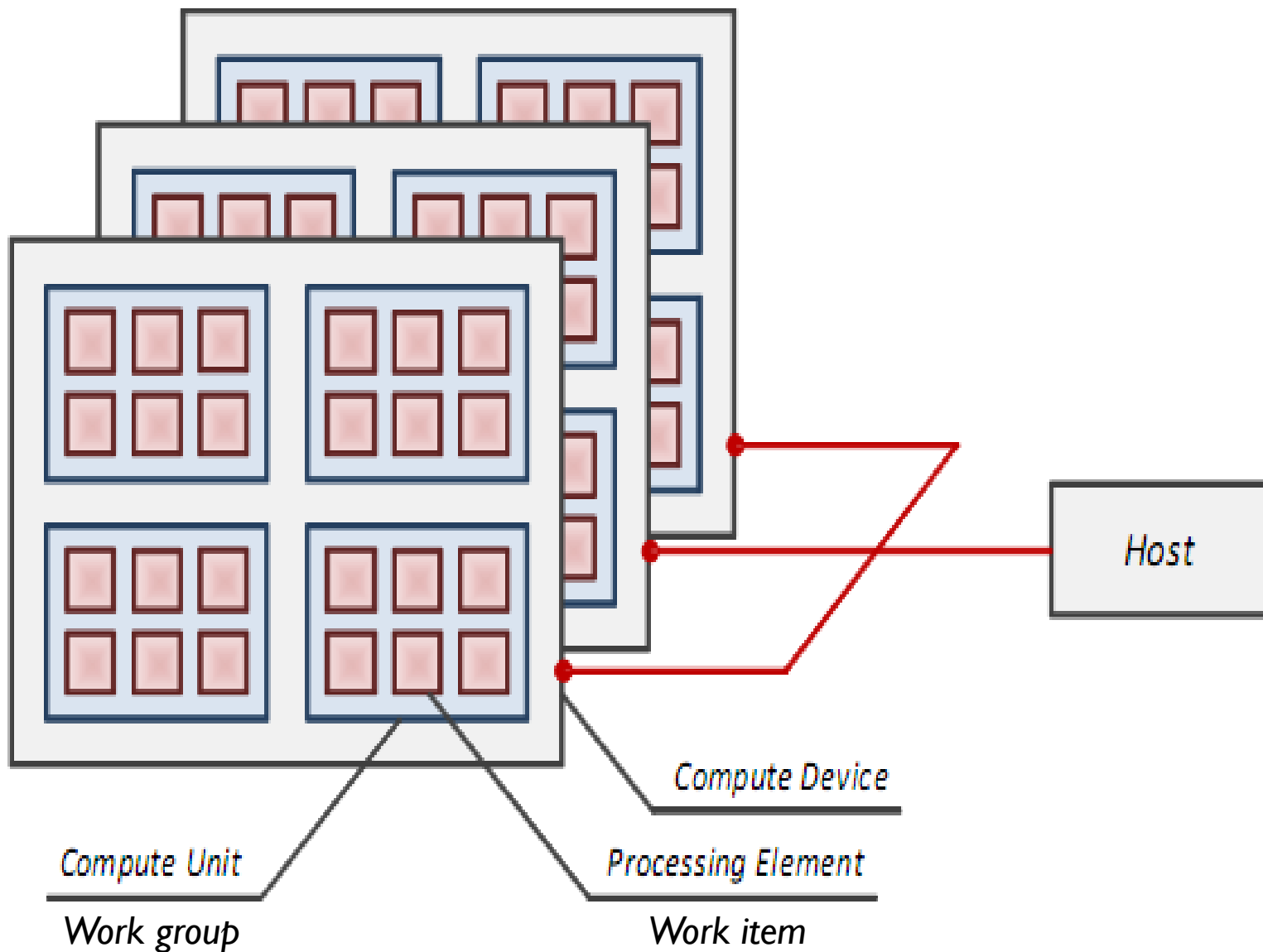
Фрагмент программы на OpenMP

```
#include <omp.h>
```

```
...
main () {
    int size, index, count = COUNT;
    double sum;
    double *arrayOfValues = new double( count );
    ... //инициализация элементов массива
    #pragma omp parallel private( index ) reduction(+ : sum)
    {
        index = omp_get_thread_num( );
        size = omp_get_num_threads( );
        while( index < count ){
            sum += arrayOfValues[ index ];
            index += size;
        }
    } //при выходе из параллельного региона
    //переменная sum будет иметь корректное
    //значение суммы всех элементов массива
    ...
}
```

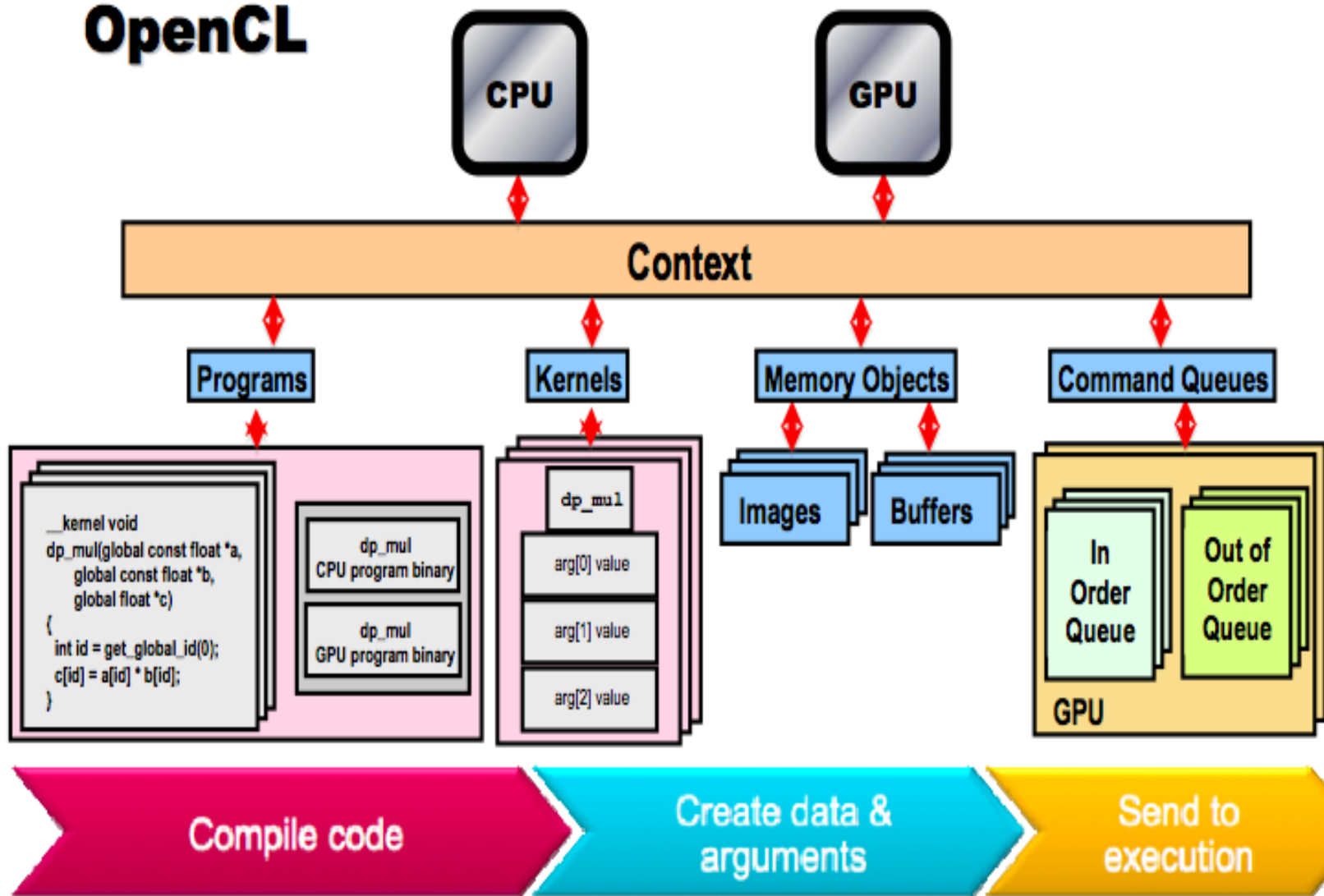

OpenCL и CUDA

Модель вычислительной системы



Общая схема работы OpenCL

OpenCL



Пример программы сложения векторов на CUDA

1

```
const int N = 1048576; //Размер вектора в элементах = 256 * 4096
const int dataSize = N * sizeof(float); //размер вектора в байтах
cudaError_t errCode; // Для получения кода ошибки
```

//Код программы ядра для GPU

//Попарное сложение элементов вектора:

//один CUDA-поток выполняет сложение одной пары элементов

```
__global__ void vectorAdd( float *resultV, float *srcA, float *srcB ){
```

//вычисление индекса складываемых элементов:

```
    int idx = blockDim.x * blockIdx.x + threadIdx.x;
```

//вычисление результата:

```
    resultV[ idx ] = srcA[ idx ] + srcB[ idx ];
```

```
}
```

```
int main( argc, argv ) {
```

// 1.Выделение памяти CPU:

```
    float *h_A = ( float * )malloc( dataSize );
```

```
    float *h_B = ( float * )malloc( dataSize );
```

```
    float *h_C = ( float * )malloc( dataSize );
```

// 2.Инициализировать векторы h_A[] и h_B[]

Пример программы сложения векторов на CUDA

2

// 3.Выделение памяти GPU

```
float *d_A, *d_B, *d_C;
```

```
cudaMalloc( (void **)&d_A, dataSize );
```

// Обработка ошибок, пример, дальше отсутствуют

```
if( (errCode = cudaGetLastError()) != CUDA_SUCCESS ){  
    printf("Error: %s", cudaGetErrorString( errCode ));  
    return 1;  
}
```

```
cudaMalloc( (void **)&d_B, dataSize );
```

```
cudaMalloc( (void **)&d_C, dataSize );
```

// 4.Скопировать входные данные в GPU для обработки

```
cudaMemcpy( d_A, h_A, dataSize, cudaMemcpyHostToDevice );
```

```
cudaMemcpy( d_B, h_B, dataSize, cudaMemcpyHostToDevice );
```

// 5.Запустить ядро из N / 256 блоков по 256 потоков

//Предполагается, что N кратно 256

//Инициализация ядра в GPU:

```
vectorAdd<<< N / 256, 256 >>>( d_C, d_A, d_B );
```

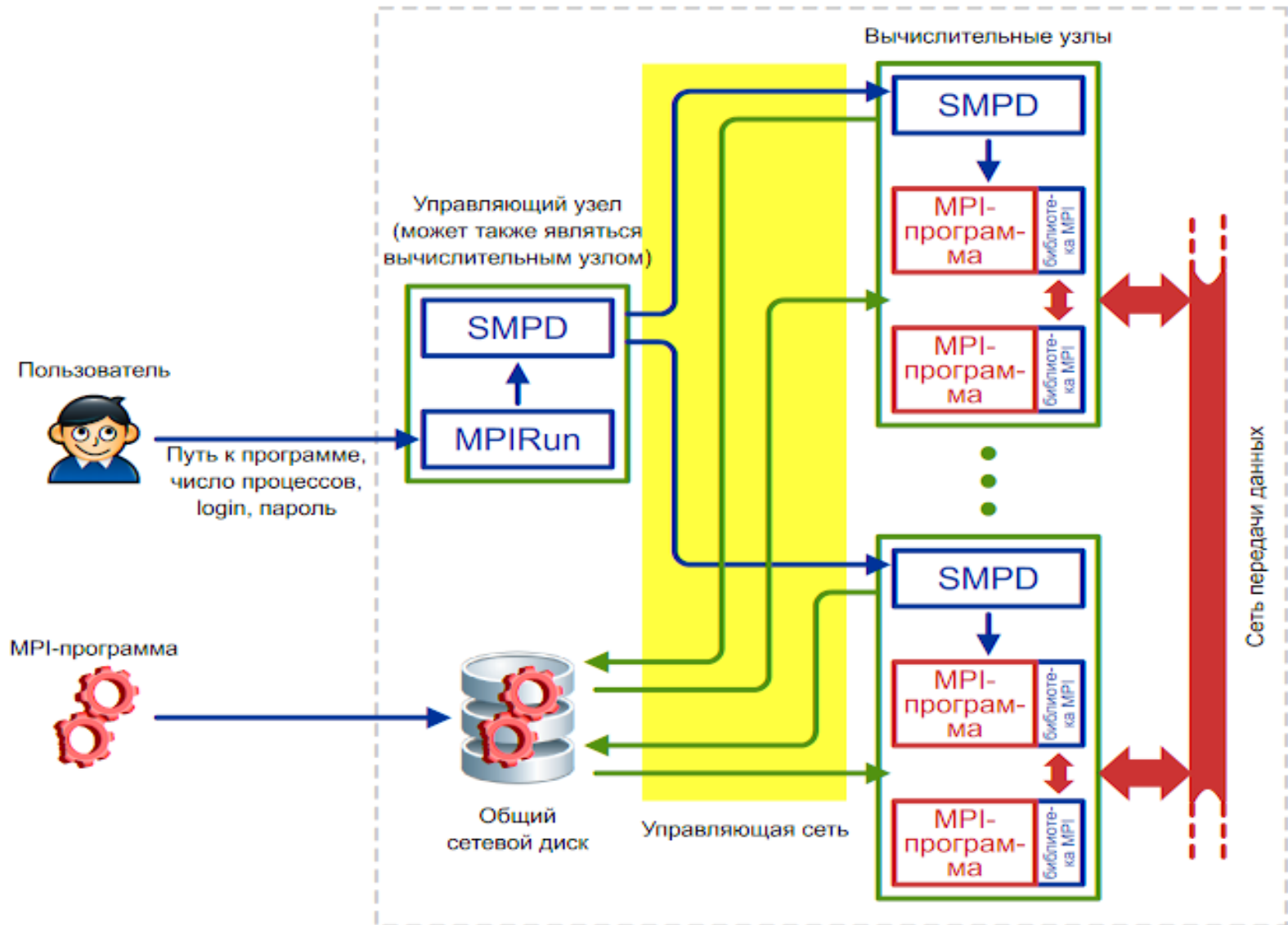
// 6.Переписать результаты работы GPU в память CPU:

```
cudaMemcpy( h_C, d_C, dataSize, cudaMemcpyDeviceToHost );
```

// 7.Обработка результатов работы GPU

```
...  
}
```

MPI: Взаимодействие пользователя, программы, MPI и ОС



Простой пример программы на MPI:

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char* argv[ ])
{   int rank, n, i, message;
    MPI_Status status;
    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &n );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    if ( rank == 0 ){
        printf ( "\n Hello from process %3d", rank );
        for ( i=n; i > 0; i -= 1 ){
            MPI_Recv( &message, 1, MPI_INT, i, MPI_ANY_TAG,
                      MPI_COMM_WORLD, &status );
            printf( "\n Hello from process %3d", message );
        }
    }else
        MPI_Send( &rank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD );
    MPI_Finalize();
    return 0;
}
```


Параллельное программирование

Сайт дисциплины:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF>

Варианты заданий:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF/VarsPP2024.htm>

Список задач:

<http://vt.cs.nstu.ru/~malyavko/ParProgrZF/Tasks2024.htm>

Курс в системе dispace:

<http://dispace.edu.nstu.ru/didesk/course/show/4872>

Тест в системе DiSpace

Малявко Александр Антонович

E-mail: a.malyavko@corp.nstu.ru