ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА СИГНАЛОВ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ

Лектор: д.т.н., доцент
Хамдамов Уткир Рахматиллаевич
utkir.hamdamov@mail.ru

Источник

- MPI tutorials
  https://mpitutorial.com/tutorials/
- Beginner's Guide to MPI
  https://www.eecis.udel.edu/~pollock/367/manual/manual.html
- Introducing MPI Installation and MPI Hello World - VS2017
  https://www.youtube.com/watch?v=IW3SKDM_yEs
- MPI Hello World!
  https://www.youtube.com/watch?v=yYlJc9REY60
- Хамдамов У.Р. ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА СИГНАЛОВ. ТАТУ, Алоқачи нашриёти, 2022. – 312 б.

Содержание

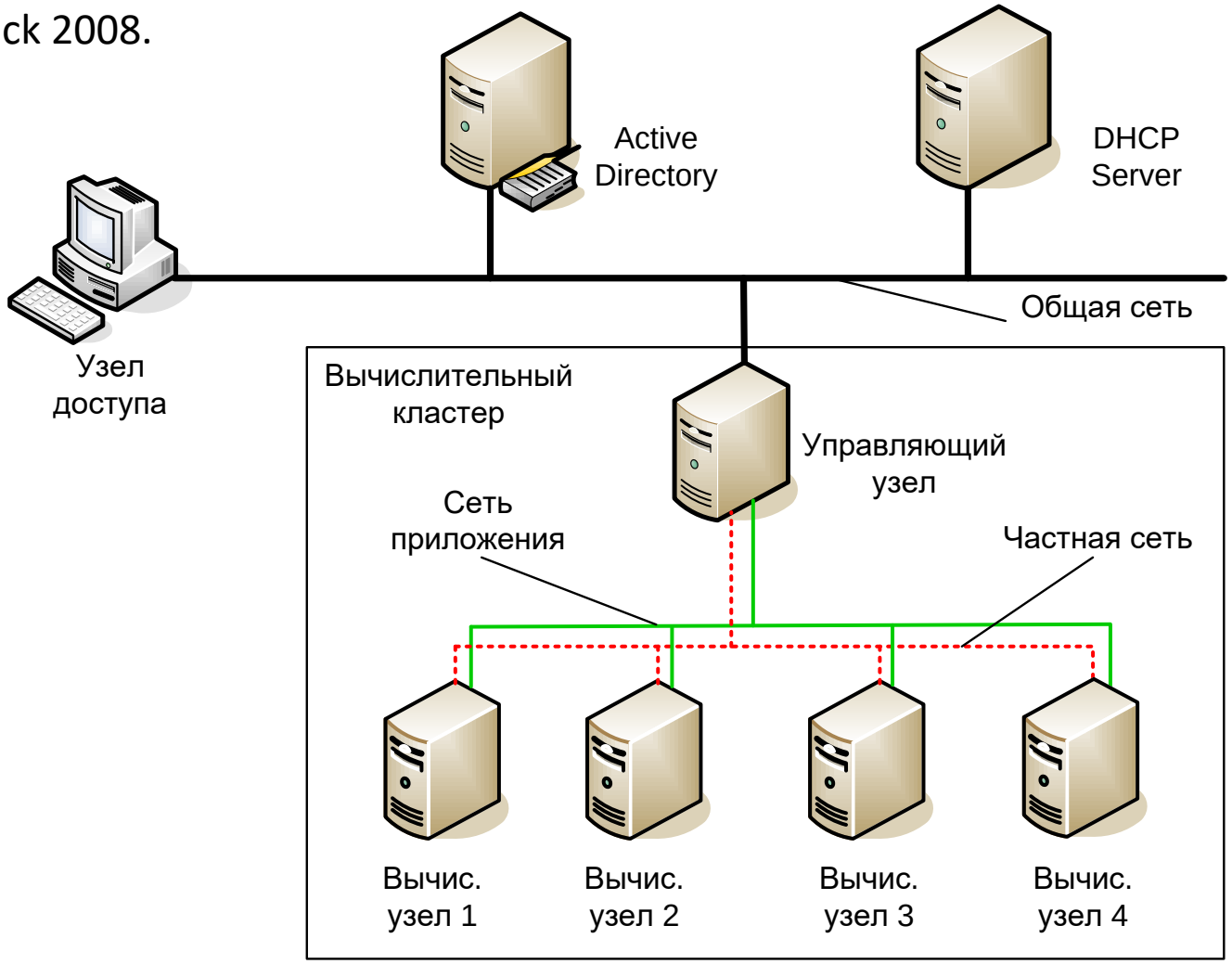
- Вычислительный кластер и топология сети
- Развертывание вычислительного кластера
- Отладка приложений на MPI-кластере
- Порядок написание программы на основе MPI
- Порядок выполнение MPI-программы
- Исходный код MPI-программы

Вычислительный кластер

- Вычислительный кластер это совокупность распределенных вычислительных узлов, объединенных высокоскоростными каналами связи, представляющая с точки зрения пользователя единый вычислительный ресурс.
- Основное предназначение вычислительного кластера – выполнение большого объема расчетов, с которым не справляются современные персональные компьютеры.
- По типу архитектуры кластер относится к системам с распределенной памятью, при этом каждый узел кластера в отдельности представляет собой систему с общей памятью.
- Вычислительный кластер в основном состоит из следующих компонентов:
  - **управляющий узел;**
  - **вычислительный узел;**
  - **узел доступа;**
  - **коммуникационная сеть.**

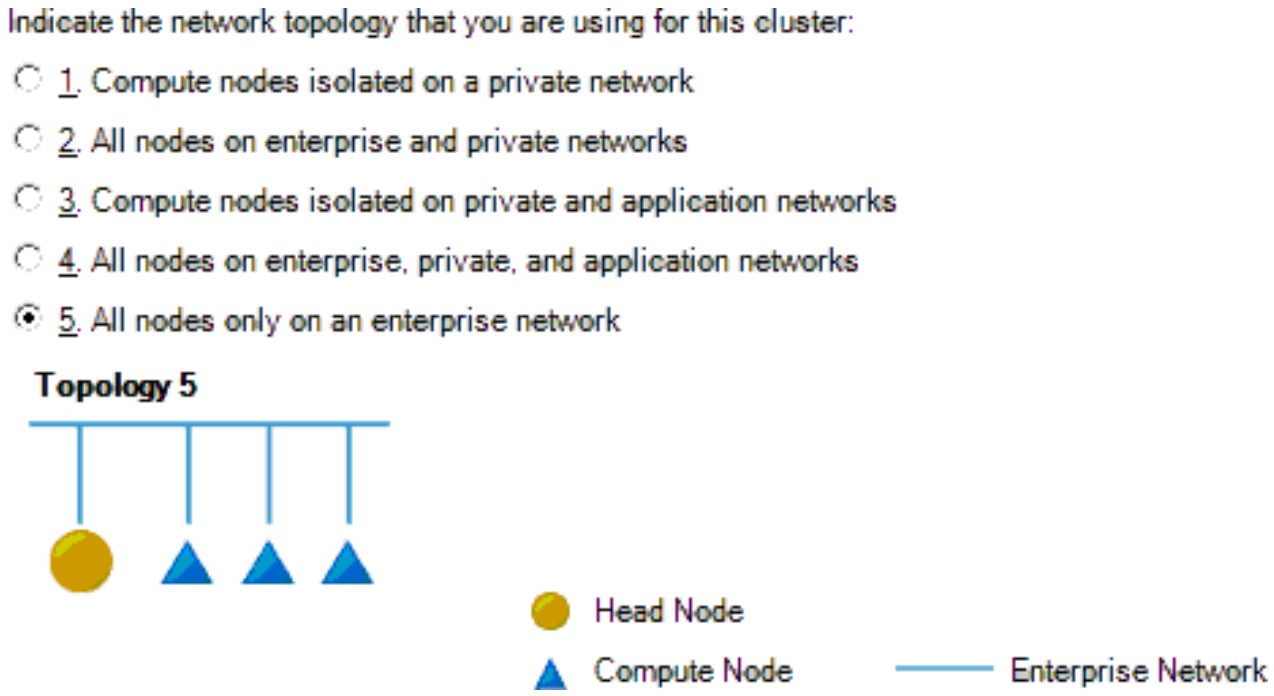
Вычислительный кластер

- В составе вычислительного кластера используются настроенные узлы с операционной системой (ОС) Windows HPC Server 2008 R2 и пакет программ HPC Pack 2008.



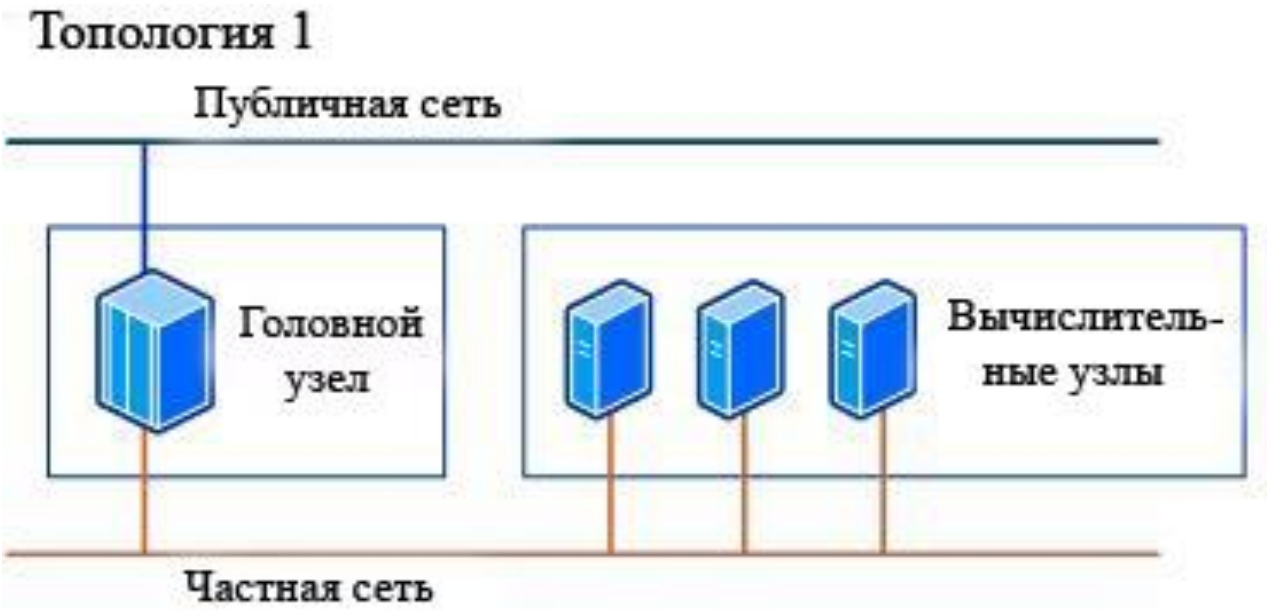
Топология сети кластера

- **Общедоступная сеть** (Enterprise) – сеть организации, соединенная с ведущим и (возможно) вычислительными узлами кластера.
- **Частная сеть** (Private) – выделенная сеть, предназначенная для коммуникации между узлами вычислительного кластера.
- **Сеть приложения** (Application) – выделенная сеть с высокой пропускной способностью и низкими задержками, используемая для MPI - коммуникаций между узлами.



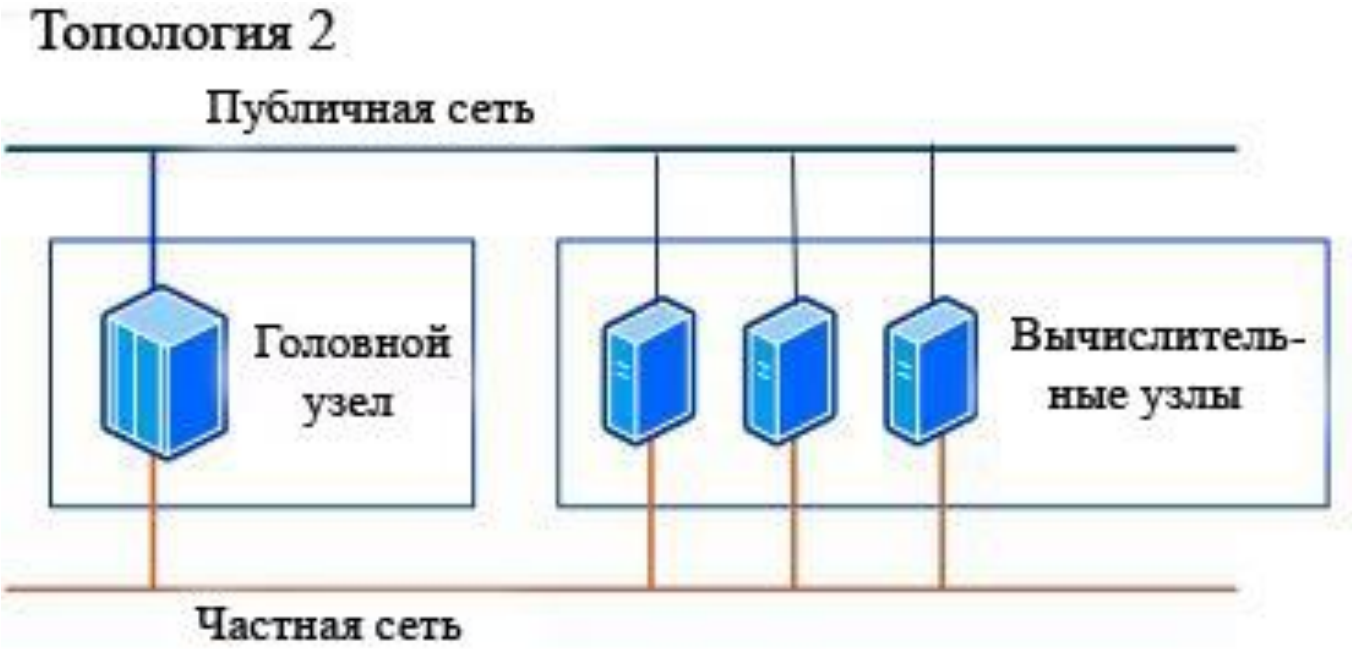
Топология сети кластера

- 1. Вычислительные узлы изолированы в частную сеть (Compute nodes isolated on private networks).
- В этом случае управляющий-узел, имеющий **две сетевые карты**, обеспечивает трансляцию адресов между вычислительными узлами, каждый из которых по единственной сетевой карте подключен к частной сети.



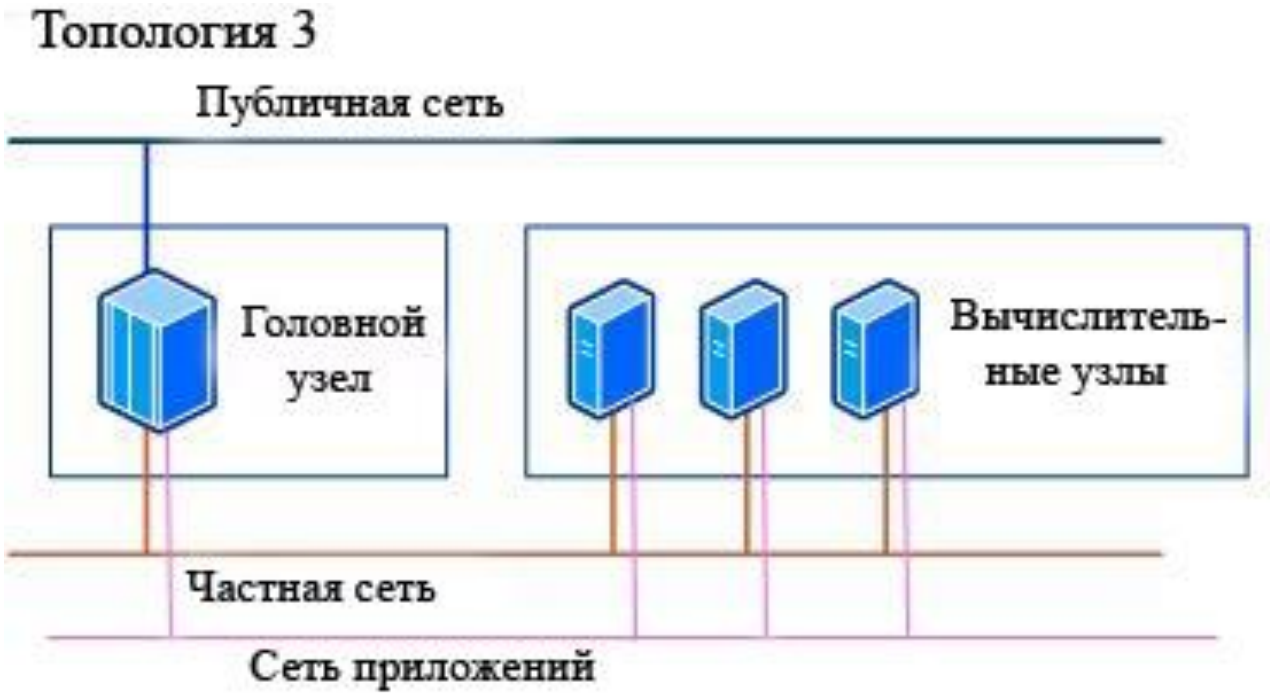
Топология сети кластера

- 2. Все узлы подключены и к публичной, и к частной сети (All nodes on enterprise and private networks), для этого используются по **две сетевые карты** на узел



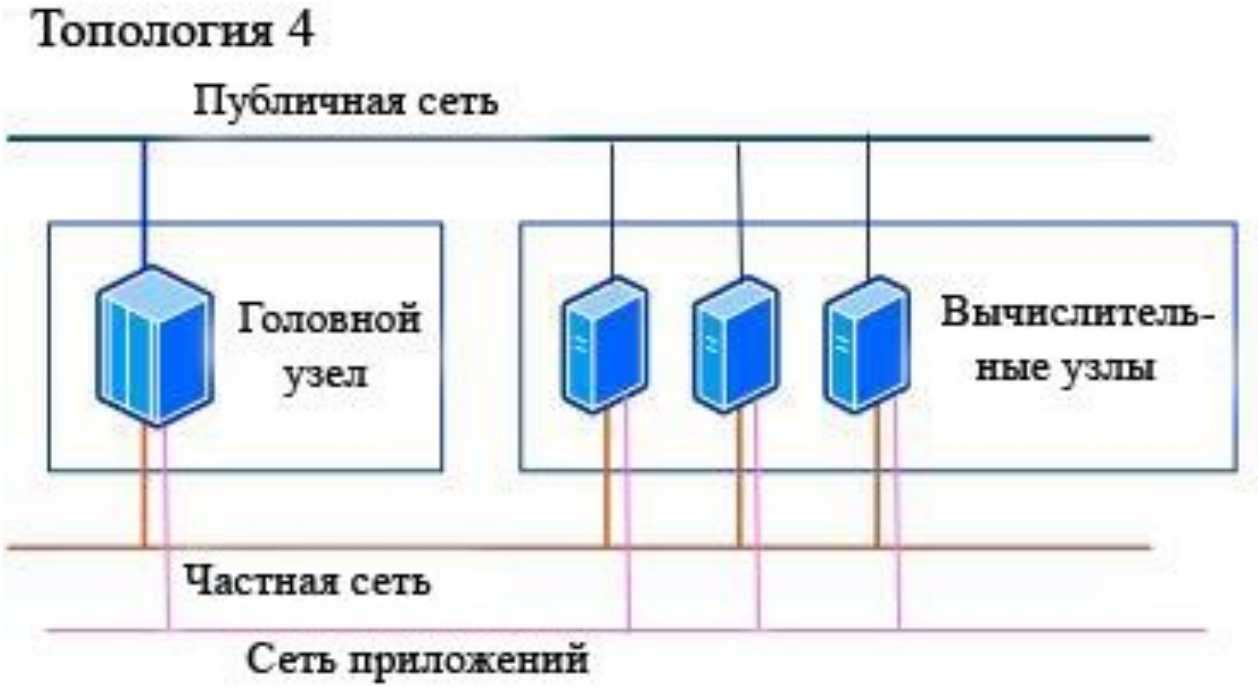
Топология сети кластера

- 3. Вычислительные узлы изолированы в частной сети и сети приложения (Compute nodes isolated on private and application networks). В данном случае управляющий узел использует три сетевые карты: для подключения к общедоступной, к частной сети и к сети MPI.



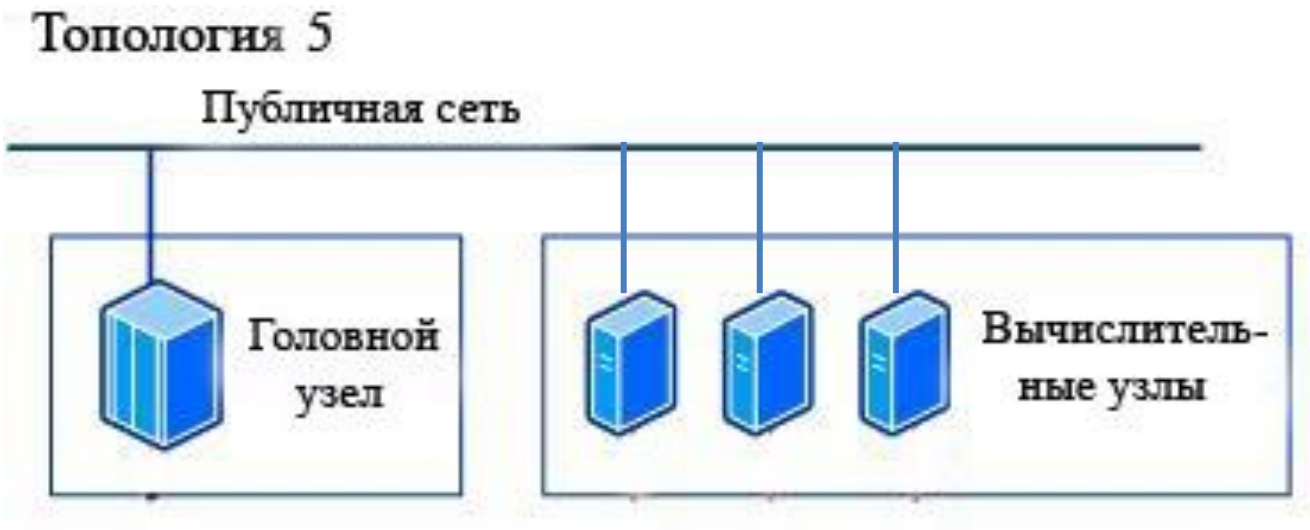
Топология сети кластера

- 4. Все узлы подключены к публичной и к частной сети, а также к сети приложений (All nodes on enterprise, private and application networks). В таком случае в каждом узле используются все три сетевые карты



Топология сети кластера

- 5. Все узлы подключены только к общедоступной сети (All nodes only on an enterprise network)

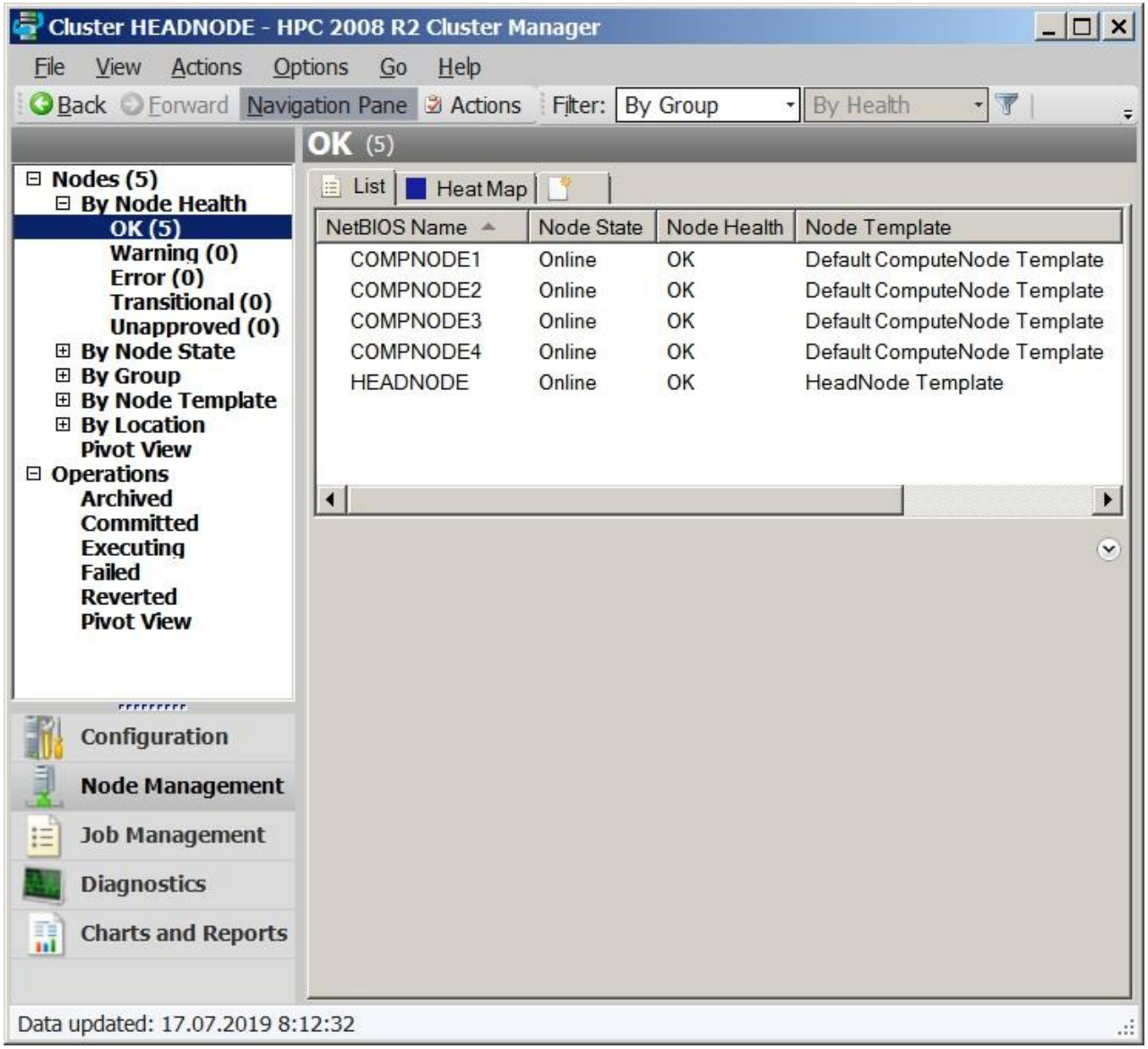


Развертывание кластера

- Развертывание вычислительного кластера начинается с установки и настройки ведущего узла, состоящих из следующих этапов:
  - 1. Установка HPC Server на ведущем узле;
  - 2. Добавление компьютера в состав какого-либо существующего домена Active Directory, либо создание нового домена и назначение ведущего узла контроллером домена;
  - 3. Установка Microsoft HPC Pack Server 2008 на ведущем узле.

- Процесс развертывания кластера состоит из следующих этапов и основных задач:
  - 1) Планирование инфраструктуры кластера;
  - 2) Настройка сетевой среды;
  - 3) Настройка управляющего узла;
  - 4) Настройка узлов-вычислителей;
  - 5) Ввод узлов в кластер.

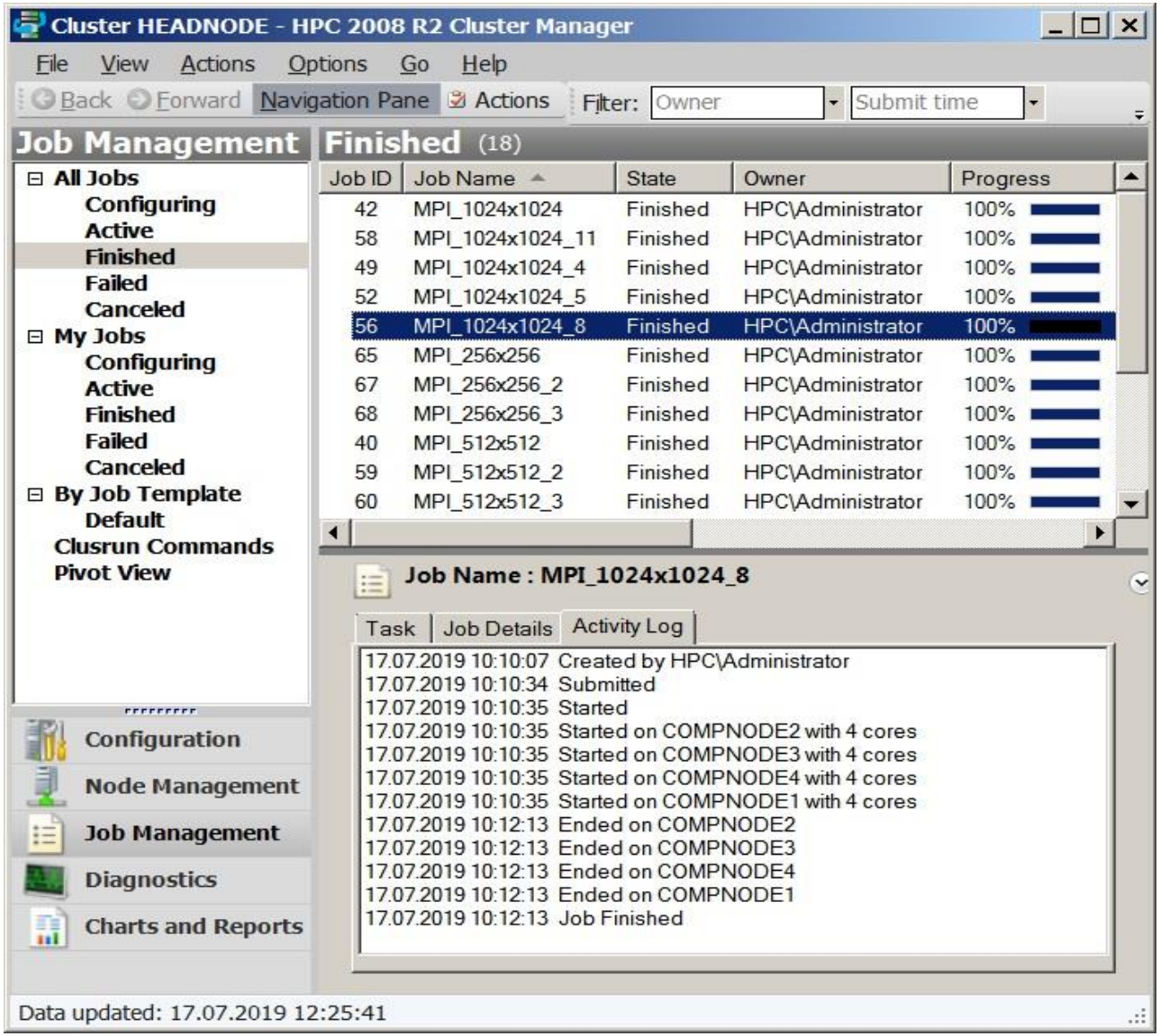
HPC Cluster manager



Порядок работы с кластером

- Общий цикл работы пользователя с вычислительным кластером состоит из следующих этапов:
  - удаленный вход на кластер;
  - копирование данных между кластером и компьютером пользователя;
  - редактирование исходных текстов программ;
  - компиляция программ;
  - запуск задач и работа с очередью;
  - завершение сеанса.

Управление заданиями



Отладка приложений на MPI-кластере

При отладке приложений на отладчике MPI-кластера необходимо учитывать следующие требования:

- На компьютере разработчика, то есть на узле доступа должна быть установлена среда разработки **Visual Studio 2010 Professional Edition** или более высокой версии с удаленным отладчиком.
- Пользователь должен иметь **права администратора кластера**, потому что под этим аккаунтом будут осуществлены вычисления.
- Visual Studio должен иметь **доступ ко всем вычислительным узлам**, на которых будут запускаться сеансы отладки. Потому что, приложение будет разработано на управляющем узле кластера или на выделенном узле доступа.
- Для отправки программы (задания) в кластер HPC с клиентского компьютера, требуется установка пакета **Microsoft HPC Pack 2008**

Процесс создания проекта MPI

Для создания проекта распределенной обработки сигналов на базе кластера необходимо выполнить следующие действия:

1. Запустить среду разработку Visual Studio 2010.
2. Создать консольное приложение Win32 на C++ с именем **ParallelHWT** без предкомпилированных заголовков.
3. Указать в свойстве проекта **ParallelHWT** местоположение файлов заголовков MS MPI, местоположение файла библиотеки Microsoft HPC Pack 2008 SDK и поддержка OpenMP для языка C/C++:
   - Свойства ParallelHWT / Свойства конфигурации / Каталоги VC++ / Каталоги включения: - добавить следующее местоположение файлов заголовков MS MPI:
   - C:\Program Files\Microsoft HPC Pack 2008 SDK\Include;
   - Свойства ParallelHWT / Свойства конфигурации / Каталоги VC++ / Каталоги библиотек: - добавить следующее местоположение файла библиотеки Microsoft HPC Pack 2008 SDK:
     - - для отладки 32-разрядное приложение:
     - C:\Program Files\Microsoft HPC Pack 2008 SDK\Lib\i386;
     - - для отладки 64-разрядное приложение:
     - C:\Program Files\Microsoft HPC Pack 2008 SDK\Lib\amd64;
   - Свойства ParallelHWT / Свойства конфигурации / Компоновщик / Ввод / Дополнительные зависимости: - добавить «msmpi.lib»;
   - Для поддержки многоядерной обработки интерфейсом OpenMP необходимо указать:
   - Свойства ParallelHWT / Свойства конфигурации / C/C++ / Язык / Поддержка Open MP: - установить Да (/openmp).

Настройка и запуск отладчика MPI-кластера

- Существует три варианта настройки и запуска отладчика для выполнения параллельных вычислений с помощью MPI:
  - Отладка одного процесса MPI на локальном компьютере;
  - Отладка нескольких процессов MPI на локальном компьютере;
  - Отладка нескольких процессов MPI в кластере

Отладка нескольких процессов MPI на компьютере

- Отладчика MPI-кластера можно запустить с несколькими процессами MPI (в данном случае четыре процесса), запущенными на локальном компьютере. Для запуска локального сеанса отладки для приложения ParallelHWT необходимо выполнить следующие действия:
  - Свойства ParallelHWT / Свойства конфигурации / Отладка / Отладчик для запуска: - выбрать элемент **Отладчик MPI-кластера**;
  - Свойства ParallelHWT / Свойства конфигурации / Отладка / Среда выполнения: - указать **localhost/4**. Это означает, что головной узел – localhost, количество процессов – 4.
- После вышеприведенных настроек, чтобы запустить программу необходимо установить точку останова в теле параллельного цикла for и запустить отладчик.
- В результате появится пять окон консоли: одно окно **cmd.exe** и четыре окна **ParallelHWT.exe** (по одному для каждого запущенного процесса). В окне консоли, соответствующем процессу с рангом 0, будут выведены результаты вычислений.

Отладка нескольких процессов MPI на кластере

Чтобы запустить отладчика MPI в вычислительном кластере для приложения ParallelHWT необходимо выполнять следующие действия:

1. Свойства ParallelHWT / Свойства конфигурации / Отладка / Отладчик для запуска: - выбрать элемент **Отладчик MPI-кластера**;
2. Свойства ParallelHWT / Свойства конфигурации / Отладка / Среда выполнения: - выбрать элемент **Правка узла Hpc**. В диалоговом окне «**Выбор узла»** установить следующие значения:
   - Головной узел: - выбрать имя головного узла кластера, который необходимо использовать.
   - Количество процессов: - установить значение **4**.
   - Планировать один процесс на: - выбрать способ выделения процессов. Можно выделить один процесс на **ядро**, **сокет** или **узел**.

Отладка нескольких процессов MPI на кластере

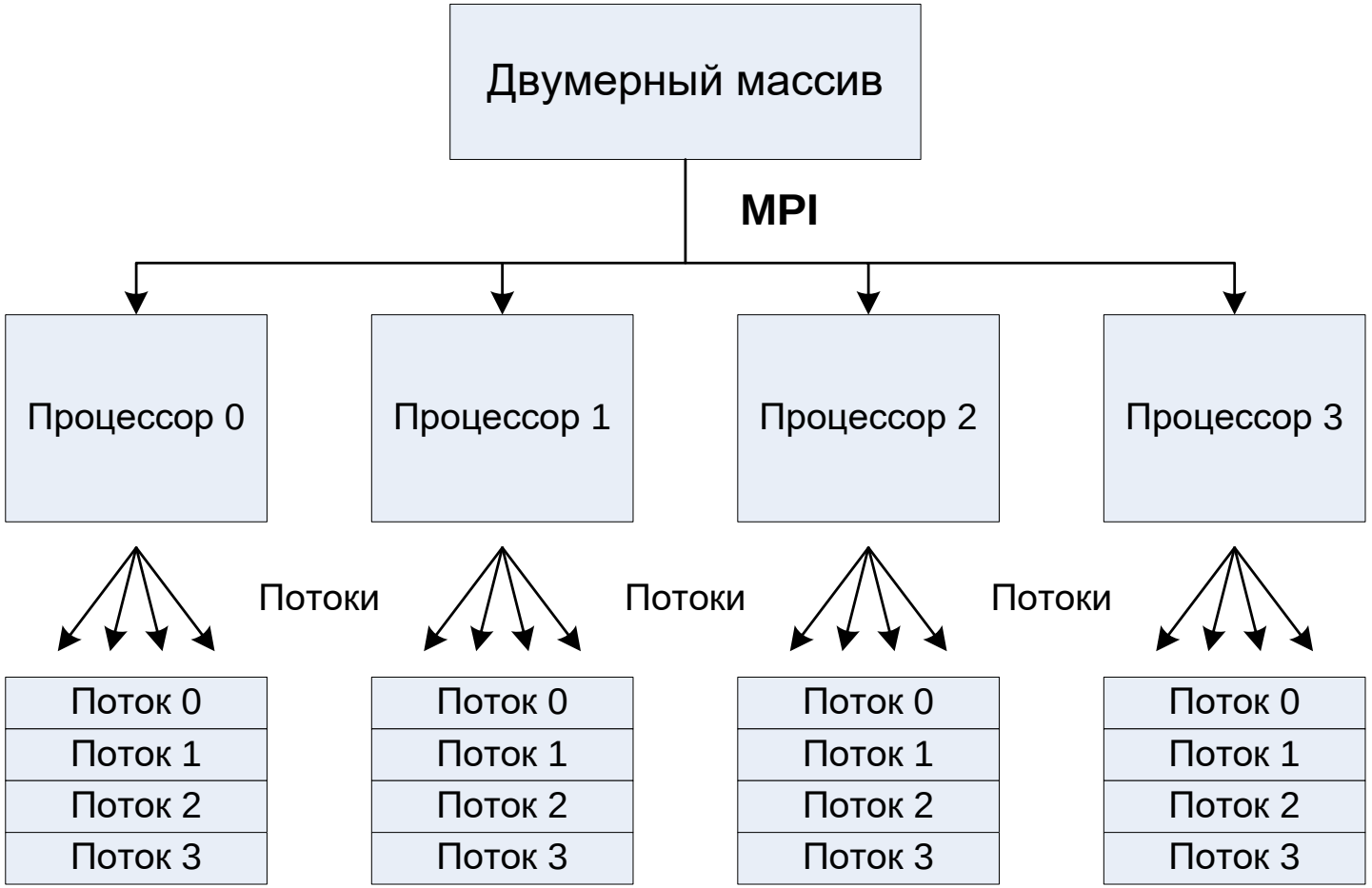
Чтобы запустить отладчика MPI в вычислительном кластере для приложения ParallelHWT необходимо выполнять следующие действия:

3. Свойства ParallelHWT / Свойства конфигурации / Отладка / Каталог развертывания: - указать общий каталог на головном узле. Если каталог развертывания не существует, то пользователь имеет права на запись в указанный корневой каталог, тогда каталог развертывания будет создан автоматически. Общий каталог **CcpSpoolDir** создается при установке пакета HPC Pack 2008 на головном узле.

4. Свойства ParallelHWT / Свойства конфигурации / Отладка / Рабочий каталог: - указать локальный рабочий каталог на каждом вычислительном узле. Например, для данного проекта можно ввести строку *C:\Users\<UserName>\ParallelHWT* где *<UserName>* - имя пользователя системы.

5. Свойства ParallelHWT / Свойства конфигурации / Отладка / Дополнительные файлы, которые необходимо развернуть: - добавить файл отладочной библиотеки OpenMP: **vcomp100d.dll** для поддержки интерфейса OpenMP в многоядерной обработке вычислительных узлах. Этот файл по умолчанию находится на следующем пути: C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\redist\Debug_NonRedist\x86\Microsoft.VC100.DebugOpenMP\vcomp100d.dll

Порядок написание программы на основе MPI



Порядок написание программы на основе MPI

- Чтобы запустить приложение MPI, все программы MPI должны содержать только один вызов MPI_Init, который обеспечивает коммуникацию. Формат вызова MPI_Init на языке C/C++ выглядит следующим образом:
  - **MPI_Init (& argc, & argv);**
- MPI_Init должен быть использован до вызова любой другой процедуры MPI. Перед выходом из процесса MPI необходимо вызвать MPI_Finalize для очистки состояний MPI. Данный вызов имеет следующий формат:
  - **MPI_Finalize ();**
- В среде MPI все узлы в группе идентифицируются уникальными значениями, называемыми рангом. В MPI приложениях для получения ранга процесса необходимо вызвать команду MPI_Comm_rank. Формат вызова определяется следующим:
  - **MPI_Comm_rank (MPI_Comm_world, & rank);**
- Чтобы узнать размер коммуникаций, необходимо вызвать команду MPI_Comm_size. Формат вызова MPI_Comm_size определяется следующим образом:
  - **MPI_Comm_size (MPI_Comm_world, & size);**

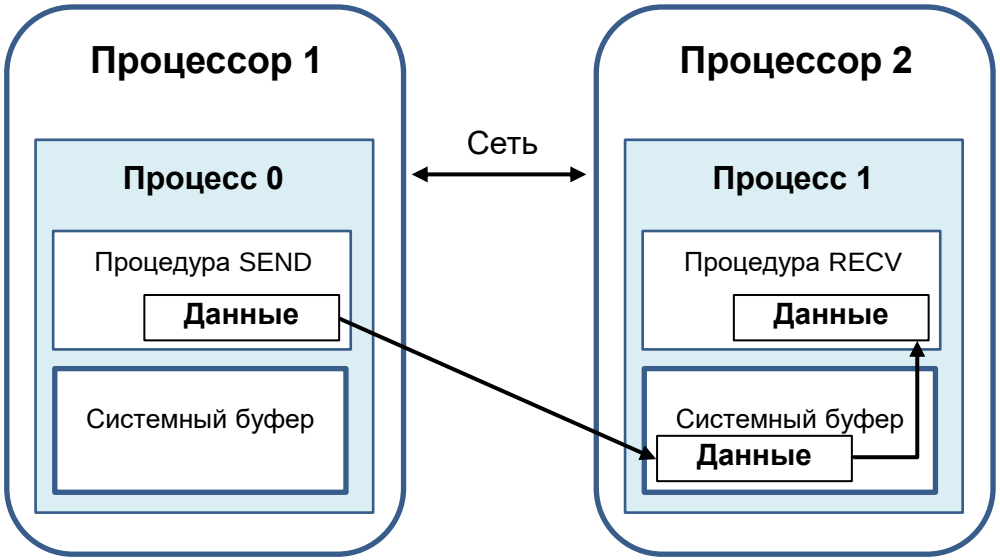
Порядок написание программы на основе MPI

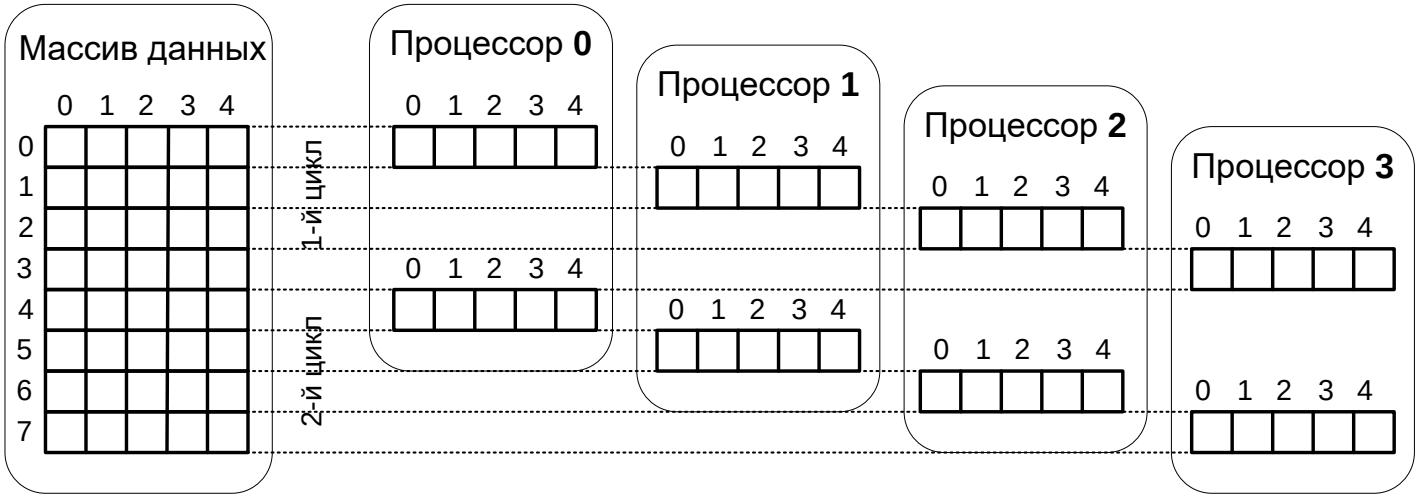
- Для разработки и запуска MPI программ, выполняющих распределенные и параллельные вычисления необходимо определить набор и порядок MPI функций.
- Минимальным набором функций для запуска MPI приложений в многопроцессорных системах являются следующие:
  - MPI_Init ; Инициализация MPI
  - MPI_Comm_size ; Определение количества процессов
  - MPI_Comm_rank ; Определение ранг вызывающего процесса
  - MPI_Send ; Передача сообщений
  - MPI_Recv ; Получение сообщений
  - MPI_Finalize ; Завершение MPI

Порядок написание программы на основе MPI

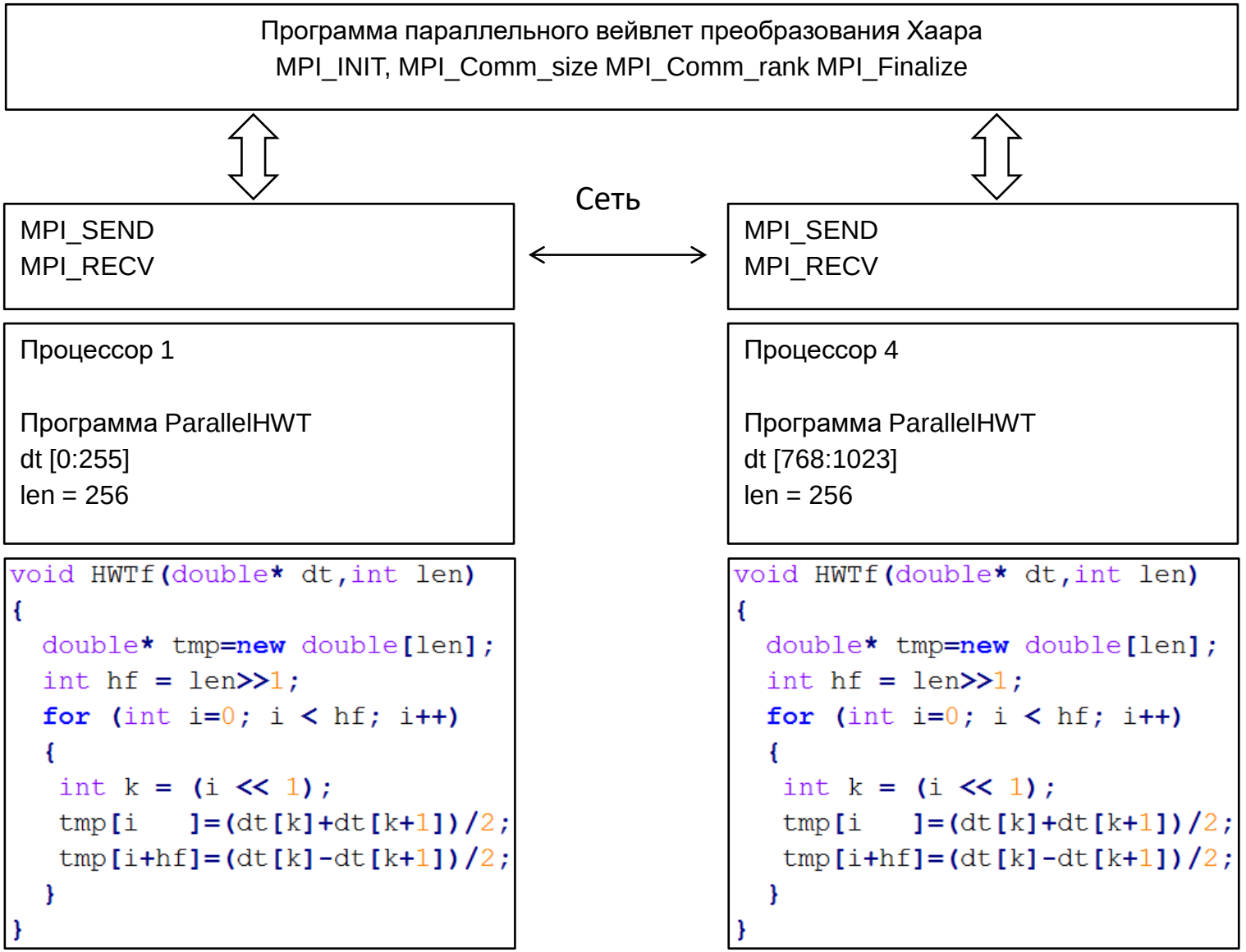
- **MPI_Send**(Sender, n*m, MPI_FLOAT, i, TagA, COMM);
  - Sender – буфер передачи данных;
  - n*m – количество передаваемых элементов;
  - MPI_FLOAT – тип передаваемых данных;
  - i – номер процесса назначения;
  - TagA – метка сообщения;
  - COMM – коммуникатор для взаимодействия процессов;
- **MPI_Recv**(Receiver, n*m, MPI_FLOAT, root, TagA, COMM, &status);
  - Receiver - буфер приема данных;
  - n*m – количество принимаемых элементов;
  - MPI_FLOAT – тип принимаемых данных;
  - root – номер процесса источника;
  - TagA – метка сообщения;
  - COMM – коммуникатор для взаимодействия процессов;
  - Status – состояние сообщения.

Порядок написание программы на основе MPI





Порядок написание программы на основе MPI

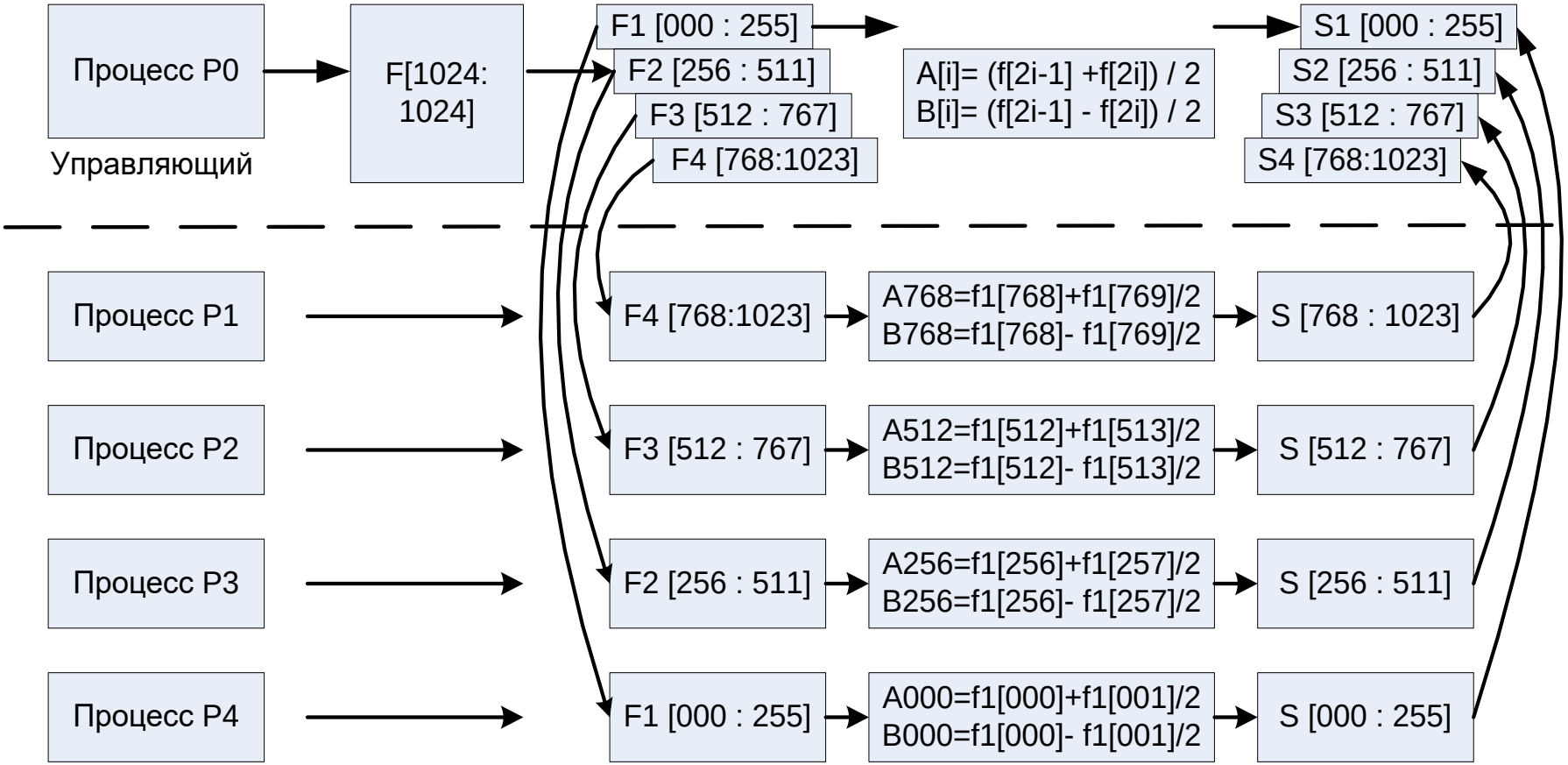


Порядок выполнение MPI-программы

- Для осуществления параллельного преобразования сигнала управляющая программа на каждый процессор загружает экземпляр программы вейвлетного преобразования Хаара (ParallelHWT), создавая отдельные процессы.
- Данные к процессам доставляются с помощью процедур **MPI_SEND** и **MPI_RECV**.
- В данном случае для распределения данных используется массив dt[x] с размером len = 256. В первом цикле обработки каждый процессор получает следующие данные:
  - Процессор 1: dt [0:255] с размером len = 256.
  - Процессор 2: dt [256:511] с размером len = 256.
  - Процессор 3: dt [512:767] с размером len = 256.
  - Процессор 4: dt [768:1023] с размером len = 256.
- При обработке двумерных сигналов каждый процессор получает массив данных с одинаковым количеством элементов. При этом основная MPI-программа создаёт управляющий процесс (P0) для создания других процессов, распределения данных и сбора результатов

Порядок выполнение MPI-программы

Находясь в управляющем процессе, основное тело программы загружает копии параллельных участков программ на отдельные процессоры, создавая в нем процессы P1, P2, P3, P4. Следовательно, распределяет строки элементов массива данных к соответствующим процессам. В итоге собирает результаты параллельных вычислений из процессов.



Исходный код MPI-программы

- Программный код обработки одномерного сигнала на основе MPI:

```
#include "mpi.h"
#include <iostream>

using namespace std;

const int Tag = 0;
const int root = 0;

int main() {
     int rank, commSize;

     MPI_Init(NULL, NULL);
     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
     MPI_Comm_size(MPI_COMM_WORLD, &commSize);

     double *arr, *buffer;
     int n=8192;
     MPI_Status status;
```

Исходный код MPI-программы

- Программный код обработки одномерного сигнала на основе MPI:

```
if (root == rank)
{
 // объявление одномерного массива
 arr = new double[n];
 for (int i = 0; i < n; ++i)
 arr[i] = rand() % 5 + 1;
 // Определение размера блока
 int partSize = n/commSize;
 // Перемещение указателя
 int shift = n%commSize;
 // Передача блоков данных к процессам кроме 0-го процесса
 for (int i = root+1; i < commSize; ++i)
       {
        MPI_Send(arr + shift + partSize*i, partSize, MPI_DOUBLE,
                        i, Tag, MPI_COMM_WORLD);
       }
 // Вычисление 1-го блока в 0-ом процессе
 hwt_array(arr, shift + partSize);
 cout << "Rank" << root << " finished" << endl;

 // Прием результатов из других процессов
 buffer = new double[partSize];
 for (int i = root+1; i < commSize; ++i)
       {
        MPI_Recv(buffer, partSize, MPI_DOUBLE, i, Tag, MPI_COMM_WORLD,&status);
        cout << "Rank" << i << " finished" << endl;
       }
}
```

Исходный код MPI-программы

- Программный код обработки одномерного сигнала на основе MPI:

```
else
 {
        // Определение параметров отправленных данных
        MPI_Probe(root, Tag, MPI_COMM_WORLD, &status);

        // Определение фактически принятого кол-ва элементов массива
        MPI_Get_count(&status, MPI_DOUBLE, &n);
        arr = new double[n];

        // Прием блока массива из 0-го процесса
        MPI_Recv(arr, n, MPI_DOUBLE, root, Tag, MPI_COMM_WORLD,&status);

        // Вычисление
        hwt_array(arr, n);

        // Передача результатов к 0-му процессу
        MPI_Send(arr, n, MPI_DOUBLE, root, Tag, MPI_COMM_WORLD);
 }

 delete[] arr;
 system("pause");

 MPI_Finalize();
}
```

Исходный код MPI-программы

- Программный код обработки одномерного сигнала на основе MPI:

```
// Функция вейвлет преобразования одномерных сигналов

double hwt_array(double *array, int n)
      {
            int half = n >> 1;
            for (int i = 0; i < half; ++i)
            {
                  int k = (i << 1);
                  double a = array[k  ];
                  double b = array[k+1];
                  array[k  ] = (a + b) / 2;
                  array[k+1] = (a - b) / 2;
            }
            return array[n];
      }
```

Исходный код MPI-программы

- Программный код для Hello world на основе MPI:

```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, rank %d out of %d processors\n",
           processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

Спасибо за внимание