Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Образовательный комплекс

Введение в методы параллельного программирования

Лекция 6.

Параллельное программирование на основе MPI

Гергель В.П., профессор, д.т.н.
Кафедра математического обеспечения ЭВМ

Производные типы данных в MPI…

- Во всех ранее рассмотренных примерах использования функций передачи данных предполагалось, что сообщения представляют собой некоторый непрерывный вектор элементов предусмотренного в MPI типа,
- В общем случае, необходимые к пересылке данные могут располагаться не рядом и состоять из разного типа значений:
  - разрозненные данные могут быть переданы с использованием нескольких сообщений (такой способ ведет к накоплению латентности множества выполняемых операций передачи данных),
  - разрозненные данные могут быть предварительно упакованы в формат того или иного непрерывного вектора (появление лишних операций копирования данных).

Производные типы данных в MPI…

При разработке параллельных программ иногда возникает потребность передавать данные разных типов (например, структуры) или данные, расположенные в несмежных областях памяти (части массивов, не образующих непрерывную последовательность элементов).

Создание производных типов для использования в коммуникационных операциях вместо предопределенных типов MPI - механизм эффективной пересылки данных в упомянутых выше случаях.

Производные типы MPI не являются в полном смысле типами данных, как это понимается в языках программирования. Они не могут использоваться ни в каких других операциях, кроме коммуникационных.

Производные типы данных в MPI…

Производный тип MPI представляет собой скрытый (opaque) объект – (карту типа), который специфицирует две вещи: последовательность базовых типов и последовательность смещений. Значения смещений не обязательно должны быть неотрицательными, различными и упорядоченными по возрастанию.

Использование производного типа в функциях обмена сообщениями можно рассматривать как трафарет, наложенный на область памяти, которая содержит передаваемое или принятое сообщение.

Производные типы данных в MPI…

- *Производный тип данных* в MPI - описание набора значений предусмотренного в MPI типа, значения могут не располагаться непрерывно по памяти:
  - Задание типа в MPI принято осуществлять при помощи *карты типа* (*type map*) в виде последовательности описаний входящих в тип значений, каждое отдельное значение описывается указанием типа и смещения адреса месторасположения от некоторого базового адреса:

$$\textbf{\textit{TypeMap}} = \{(type_0, disp_0), (type_1, disp_1), \dots, (type_{n-1}, disp_{n-1})\}$$

  - Часть карты типа с указанием только типов значений именуется в MPI *сигнатурой типа*:

$$\textbf{\textit{TypeSignature}} = \{type_0, type_1, \dots, type_{n-1}\}$$

Производные типы данных в MPI…

- **Пример.**
  - Пусть в сообщение должны входить значения переменных:

```
double a; /* адрес 24 */
double b; /* адрес 40 */
int    n; /* адрес 48 */
```

  - Тогда производный тип для описания таких данных должен иметь карту типа следующего вида:

```
{(MPI_DOUBLE,0),
 (MPI_DOUBLE,16),
 (MPI_INT,24)
}
```

Производные типы данных в MPI…

- Для производных типов данных в MPI используется следующих ряд новых понятий:
  - *нижняя граница* типа:

$$lb\ (TypeMap) = \min_j (disp_j)$$

  - *верхняя граница* типа:

$$ub\ (TypeMap) = \max_j (disp_j + sizeof\ (type_j)) + \Delta$$

  - *протяженность* типа (размер памяти в байтах, который нужно отводить для одного элемента производного типа):

$$extent\ (TypeMap) = ub\ (TypeMap) - lb\ (TypeMap)$$

  - *размер* типа данных - это число байтов, которые занимают данные.

Различие в значениях протяженности и размера состоит в величине округления для выравнивания адресов.

Производные типы данных в MPI…

- Для получения значения протяженности и размера типа в MPI предусмотрены функции:

```
int MPI_Type_extent ( MPI_Datatype type, MPI_Aint *extent );
int MPI_Type_size   ( MPI_Datatype type, MPI_Aint *size );
```

- Определение нижней и верхней границ типа может быть выполнено при помощи функций:

```
int MPI_Type_lb ( MPI_Datatype type, MPI_Aint *disp );
int MPI_Type_ub ( MPI_Datatype type, MPI_Aint *disp );
```

- Важной и необходимой при конструировании производных типов является функция получения адреса переменной:

```
int MPI_Address ( void *location, MPI_Aint *address );
```

Производные типы данных в MPI…

Определение и использование производных типов включает следующие шаги (указаны имена MPI функций):

-Определение имени типа – MPI_Datatype;
-Конструирование типа (создание карты типа) – MPI_Type_ …;
-Регистрация производного типа – MPI_Commit;
-Использование имени типа в операциях передачи (параметр type);
-Уничтожение (аннулирование) производного типа – MPI_Type_free.

Производные типы данных в MPI…

- Способы конструирования производных типов данных:
  - ***Непрерывный*** способ позволяет определить непрерывный набор элементов существующего типа как новый производный тип,
  - ***Векторный*** способ обеспечивает создание нового производного типа как набора элементов существующего типа, между элементами которого существуют регулярные промежутки по памяти. При этом, размер промежутков задается в числе элементов исходного типа,
  - ***Индексный*** способ отличается от векторного метода тем, что промежутки между элементами исходного типа могут иметь нерегулярный характер,
  - ***Структурный*** способ обеспечивает самое общее описание производного типа через явное указание карты создаваемого типа данных.

Производные типы данных в MPI…

- **Непрерывный способ конструирования:**

```
int MPI_Type_contiguous(int count,MPI_Data_type oldtype,
                MPI_Datatype *newtype);
```

– Как следует из описания, новый тип *newtype* создается как *count* элементов исходного типа *oldtype*. Например, если исходный тип данных имеет карту типа

```
{ (MPI_INT,0),(MPI_DOUBLE,8) },
```

то вызов функции *MPI_Type_contiguous* с параметрами

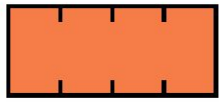
```
MPI_Type_contiguous (2, oldtype, &newtype);
```

приведет к созданию типа данных с картой типа:

```
{ (MPI_INT,0),(MPI_DOUBLE,8),(MPI_INT,16),(MPI_DOUBLE,24) }.
```

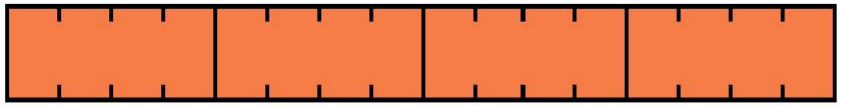
Графическая интерпретация работы конструктора MPI_Type_contiguous

oldtype = MPI_REAL



count = 4

newtype



Производные типы данных в MPI…

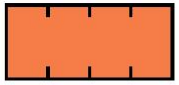
- **Векторный способ конструирования…**
  - при векторном способе новый производный тип создается как набор блоков из элементов исходного типа, при этом между блоками могут иметься регулярные промежутки по памяти.

```
int MPI_Type_vector ( int count, int blocklen, int stride,
  MPI_Data_type oldtype, MPI_Datatype *newtype ),
где
- count     - количество блоков,
- blocklen  - размер каждого блока,
- stride    - количество элементов, расположенных между двумя
              соседними блоками
- oldtype   - исходный тип данных,
- newtype   - новый определяемый тип данных.
```

  - Если интервалы между блоками задаются в байтах, а не в элементах исходного типа данных, следует использовать функцию:

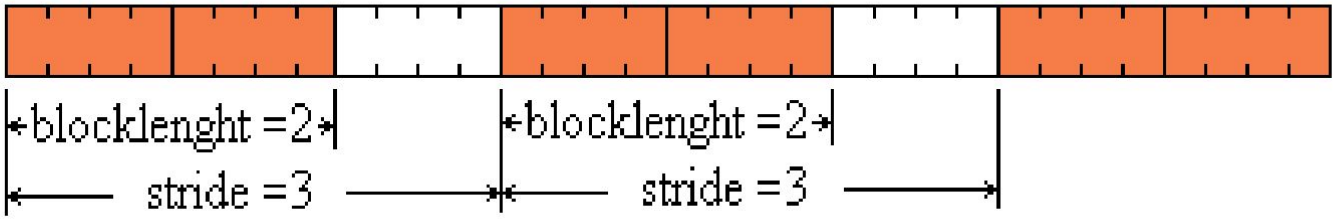
```
int MPI_Type_hvector ( int count, int blocklen, MPI_Aint stride,
   MPI_Data_type oldtype, MPI_Datatype *newtype );
```

Графическая интерпретация работы конструктора MPI_Type_vector.

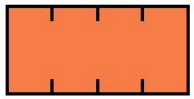
oldtype = MPI_REAL 

count = 3, blocklenght = 2, stride = 3

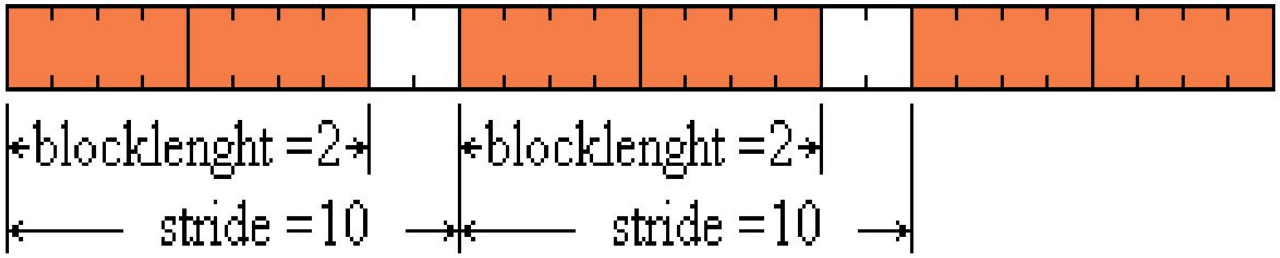
newtype



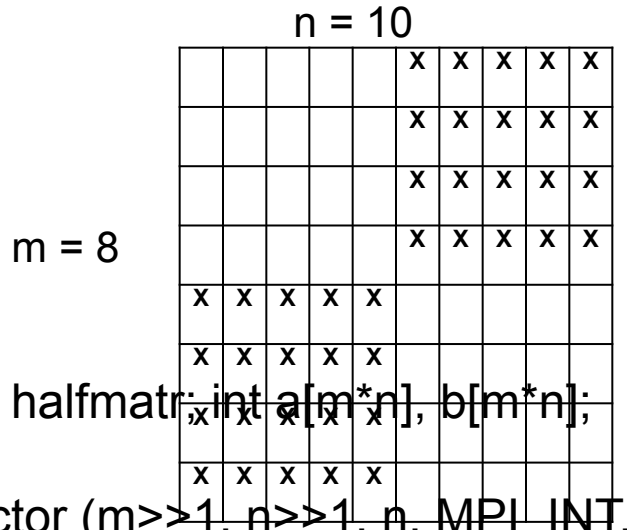
Графическая интерпретация работы конструктора MPI_Type_hvector

oldtype = MPI_REAL 

count = 3, blocklenght = 2, stride = 10

newtype 

Векторный тип данных




```
MPI_Datatype halfmatr; int a[m*n], b[m*n];
…
MPI_Type_vector (m>>1, n>>1, n, MPI_INT, &halfmatr);
MPI_Type_commit (&halfmatr);
MPI_Send (&a[n>>1], 1, halfmatr, 1, 3, MPI_COMM_WORLD);
MPI_Send (&a[m>>1 * n], 1, halfmatr, 1,4, MPI_COMM_WORLD);
…
MPI_Recv (&b[n>>1], 1, halfmatr, 0, 3, MPI_COMM_WORLD, &stat);
MPI_Recv (&b[(m>>1) * n], 1, halfmatr, 0, 4, MPI_COMM_WORLD, &stat);
…
MPI_Type_free (&helfmatr);
```

Производные типы данных в MPI…

- **Векторный способ конструирования:**
  - создание производных типов для описания подмассивов многомерных массивов

```
int MPI_Type_create_subarray ( int ndims, int *sizes,
    int *subsizes, int *starts, int order,
    MPI_Data_type oldtype, MPI_Datatype *newtype ),
где
- ndims     - размерность массива,
- sizes     - количество элементов в каждой размерности исходного
              массива,
- subsizes  - количество элементов в каждой размерности определяемого
              подмассива,
- starts    - индексы начальных элементов в каждой размерности
              определяемого подмассива,
- order     - параметр для указания необходимости переупорядочения,
- oldtype   - тип данных элементов исходного массива,
- newtype   - новый тип данных для описания подмассива.
```

Производные типы данных в MPI…

- **Индексный способ конструирования:**
  - новый производный тип создается как набор блоков разного размера из элементов исходного типа, при этом между блоками могут иметься разные промежутки по памяти.

```
int MPI_Type_indexed ( int count, int blocklens[], int
indices[],
  MPI_Data_type oldtype, MPI_Datatype *newtype ),
где
- count      – количество блоков,
- blocklens – количество элементов в каждов блоке,
- indices    – смещение каждого блока от начала типа (в
количестве
               элементов исходного типа),
- oldtype    – исходный тип данных
- newtype    – новый определяемый тип данных.
```

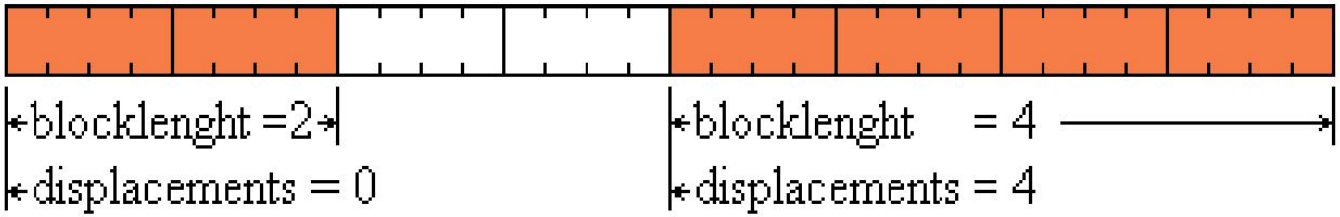
  - Если интервалы между блоками задаются в байтах, а не в элементах исходного типа данных, следует использовать функцию:

```
int MPI_Type_hindexed ( int count, int blocklens[],
 MPI_Aint indices[],MPI_Data_type oldtype,MPI_Datatype
*newtype);
```

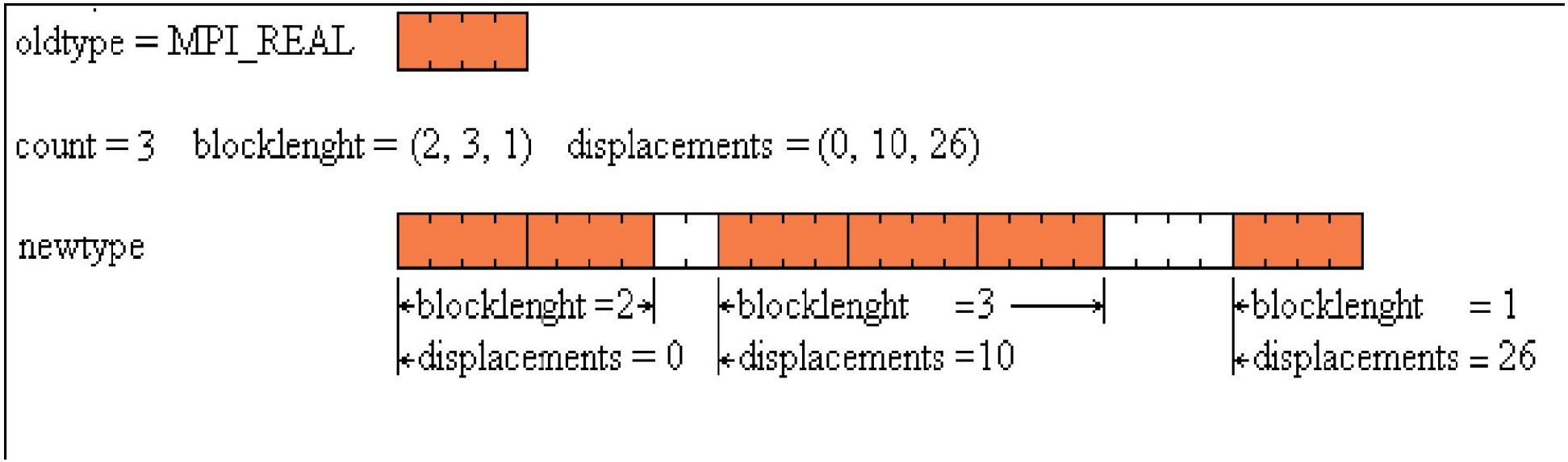
Графическая интерпретация работы конструктора MPI_Type_indexed

oldtype = MPI_REAL 

count = 2 blocklenght = (2, 4) displacements = (0,4)

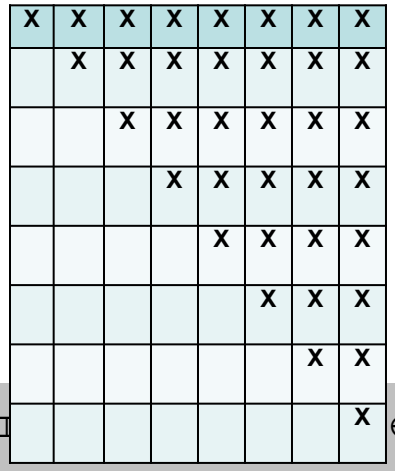
newtype 

Графическая интерпретация работы конструктора MPI_Type_hindexed



Производные типы данных в MPI…

- **Индексный способ конструирования:**
  - конструирования типа для описания верхней треугольной матрицы размером *nxn*:



```
// конструирование типа д                ерхней треугольной матрицы
for ( i=0, i<n; i++ ) {
  blocklens[i] = n - i;
  indices[i]   = i * n + i;
}
MPI_Type_indexed ( n, blocklens, indices, &UTMatrixType,
            &ElemType );
```

Производные типы данных в MPI…

- **Структурный способ конструирования:**
  - является самым общим методом конструирования производного типа данных при явном задании соответствующей карты типа:

```
int MPI_Type_struct ( int count, int blocklens[],
  MPI_Aint indices[], MPI_Data_type oldtypes[],
  MPI_Datatype *newtype ),
где
 - count       – количество блоков,
 - blocklens   – количество элементов в каждов блоке,
 - indices     – смещение каждого блока от начала типа (в
байтах),
 - oldtypes    – исходные типы данных в каждом блоке в
отдельности,
 - newtype     – новый определяемый тип данных.
```

Графическая интерпретация работы конструктора MPI_Type_struct

oldtypes = (MPI_INT, MPI_SHORT, MPI_CHAR)

count = 3 blocklenght = (1, 6, 4) displacements = (0, 12, 26)

newtype

Производные типы данных в MPI…

- **Объявление производных типов и их удаление:**
  - перед использованием созданный тип *должен быть объявлен* :

```
int MPI_Type_commit (MPI_Datatype *type );
```

  - При завершении использования производный тип должен быть аннулирован при помощи функции:

```
int MPI_Type_free (MPI_Datatype *type );
```

Заключение…

- Во второй презентации раздела рассмотрены имеющиеся в MPI операции передачи данных между двумя процессами, а также возможные режимы выполнения этих операций.
- Обсуждается вопрос организации неблокирующих обменов данными между процессами.
- Подробно рассмотрены коллективные операции передачи данных.
- Представлены все основные способы конструирования и использования производных типов данных в MPI.

Вопросы для обсуждения

- Достаточность состава поддерживаемых в MPI операций передачи данных.
- Рекомендации по использованию разных способов конструирования типов данных.

Темы заданий для самостоятельной работы...

- **Операции передачи данных между двумя процессами**
  1. Подготовьте варианты ранее разработанных программ с разными режимами выполнения операций передачи данных. Сравните время выполнения операций передачи данных при разных режимах работы.
  2. Подготовьте варианты ранее разработанных программ с использованием неблокирующего способа выполнения операций передачи данных. Оцените необходимое количество вычислительных операций, для того чтобы полностью совместить передачу данных и вычисления. Разработайте программу, в которой бы полностью отсутствовали задержки вычислений из-за ожидания передаваемых данных.
  3. Выполните задание 3 с использованием операции одновременного выполнения передачи и приема данных. Сравните результаты вычислительных экспериментов.

Темы заданий для самостоятельной работы...

- **Коллективные операции передачи данных**
  4. Разработайте программу-пример для каждой имеющейся в MPI коллективной операции.
  5. Разработайте реализации коллективных операций при помощи парных обменов между процессами. Выполните вычислительные эксперименты и сравните время выполнения разработанных программ и функций MPI для коллективных операций.
  6. Разработайте программу, выполните эксперименты и сравните результаты для разных алгоритмов реализации операции сбора, обработки и рассылки данных всех процессам (функция MPI_Allreduce).

Темы заданий для самостоятельной работы

Производные типы в MPI

7. Разработайте программу-пример для каждого имеющегося в MPI способа конструирования производных типов данных.
8. Разработайте программу-пример с использованием функций упаковки и распаковки данных. Выполните эксперименты и сравните с результатами при использовании производных типов данных.
9. Разработайте производные типы данных для строк, столбцов, диагоналей матриц.
10. Разработайте программу-пример для каждой из рассмотренных функций для управления процессами и коммуникаторами.
11. Разработайте программу для представления множества процессов в виде прямоугольной решетки. Создайте коммуникаторы для каждой строки и столбца процессов. Выполните коллективную операцию для всех процессов и для одного из созданных коммуникаторов. Сравните время выполнения операции.
12. Изучите самостоятельно и разработайте программы-примеры для передачи данных между процессами разных коммуникаторов.

Ссылки

- Информационный ресурс Интернет с описанием стандарта MPI: http://www.mpiforum.org
- Одна из наиболее распространенных реализаций MPI библиотека MPICH представлена на http://www-unix.mcs.anl.gov/mpi/mpich
- Библиотека MPICH2 с реализацией стандарта MPI-2 содержится на http://www-unix.mcs.anl.gov/mpi/mpich2
- Русскоязычные материалы о MPI имеются на сайте http://www.parallel.ru

Литература…

- **Гергель В.П.** (2007). Теория и практика параллельных вычислений. – М.: Интернет-Университет, БИНОМ. Лаборатория знаний.
- **Воеводин** В.В., Воеводин Вл.В. (2002). Параллельные вычисления. – СПб.: БХВ-Петербург.
- **Немнюгин** С., Стесик О. (2002). Параллельное программирование для многопроцессорных вычислительных систем – СПб.: БХВ-Петербург.
- **Group**, W., Lusk, E., Skjellum, A. (1994). Using MPI. Portable Parallel Programming with the Message-Passing Interface. –MIT Press.
- **Group**, W., Lusk, E., Skjellum, A. (1999a). Using MPI - 2nd Edition: Portable Parallel Programming with the Message Passing Interface (Scientific and Engineering Computation). - MIT Press.

Литература

- **Group**, W., Lusk, E., Thakur, R. (1999b). Using MPI-2: Advanced Features of the Message Passing Interface (Scientific and Engineering Computation). - MIT Press.
- **Pacheco,** P. (1996). Parallel Programming with MPI. - Morgan Kaufmann.
- **Quinn**, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
- **Snir**, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. (1996). MPI: The Complete Reference. - MIT Press, Boston, 1996.

Следующая тема

- **Параллельное программирование на основе MPI - 3**

Авторский коллектив

Гергель В.П., профессор, д.т.н., руководитель

Гришагин В.А., доцент, к.ф.м.н.

Сысоев А.В., ассистент (раздел 1)

Лабутин Д.Ю., ассистент (система ПараЛаб)

Абросимова О.Н., ассистент (раздел 10)

Гергель А.В., аспирант (раздел 12)

Лабутина А.А., магистр (разделы 7,8,9, система ПараЛаб)

Сенин А.В. (раздел 11)

О проекте

Целью проекта является создание образовательного комплекса "Многопроцессорные вычислительные системы и параллельное программирование", обеспечивающий рассмотрение вопросов параллельных вычислений, предусматриваемых рекомендациями Computing Curricula 2001 Международных организаций IEEE-CS и ACM. Данный образовательный комплекс может быть использован для обучения на начальном этапе подготовки специалистов в области информатики, вычислительной техники и информационных технологий.

Образовательный комплекс включает **учебный курс "Введение в методы параллельного программирования" и лабораторный практикум "Методы и технологии разработки параллельных программ"**, что позволяет органично сочетать фундаментальное образование в области программирования и практическое обучение методам разработки масштабного программного обеспечения для решения сложных вычислительно-трудоемких задач на высокопроизводительных вычислительных системах.

Проект выполнялся в Нижегородском государственном университете им. Н.И. Лобачевского на кафедре математического обеспечения ЭВМ факультета вычислительной математики и кибернетики (http://www.software.unn.ac.ru). Выполнение проекта осуществлялось при поддержке компании Microsoft.