Образовательные ресурсы и лучшая практика ИТ-образования

УДК 004.42 DOI 10.25559/SITITO.2017.4.357

Абрамян М.Э.

Южный федеральный университет, г. Ростов-на-Дону, Россия

ЭЛЕКТРОННЫЙ ЗАДАЧНИК ПО ПАРАЛЛЕЛЬНОМУ ПРОГРАММИРОВАНИЮ НА БАЗЕ ИНТЕРФЕЙСА MPI СТАНДАРТА 2.0

Аннотация

В статье рассматривается один из подходов к разработке электронных задачников по параллельному MPI-программированию и описывается реализованный на основе этого подхода электронный задачник Programming Taskbook for MPI-2, содержащий 250 учебных заданий (http://ptaskbook.com/ru/ptformpi2/). Описывается история создания данного задачника, дается обзор входящих в него групп. На примере решения одной из задач демонстрируются особенности применения задачника на лабораторных занятиях. Подробно рассматриваются группы заданий, связанные с такими новыми возможностями интерфейса MPI стандарта 2.0, как параллельный файловый ввод-вывод, удаленный доступ к памяти (односторонние коммуникации) и динамическое создание процессов; приводятся примеры формулировок заданий из этих групп. Кроме того, подробно обсуждаются особенности завершающей группы заданий, посвященной параллельным матричным алгоритмам.

Ключевые слова

Электронный задачник, параллельное программирование, технология MPI, параллельные алгоритмы.

Abramyan M.E.

Southern Federal University, Rostov-on-Don, Russia

THE ELECTRONIC BOOK OF EDUCATIONAL TRAINING TASKS ON PARALLEL PROGRAMMING BASED ON THE MPI 2.0 STANDARD

Abstract

We discuss one of approaches to the development of educational parallel software and describe the Programming Taskbook for MPI-2 (http://ptaskbook.com/ru/ptformpi2/), a courseware that includes 250 training tasks on various topics of MPI. We describe the history of its development, as well as give an overview of its task groups. Usage of the Programming Taskbook for MPI-2 at laboratory classes is illustrated by example of solving one of the training tasks. We also describe task groups related to such new features of MPI 2.0 interface as parallel input-output, remote memory access (one-sided communications), and dynamic creation of processes, and give formulations of some tasks from these groups. In addition, we discuss some features of the final task group devoted to parallel matrix algorithms.

Keywords

Educational software, courseware, parallel programming, Message Passing Interface (MPI).

Введение

Технология MPI (Message Passing Interface) входит в любой современный курс параллельного программирования, поскольку в настоящее время она является наиболее распространенным средством разработки параллельных программ, основанных на передаче сообщений [1, 2].

Имеется большое число разработок, связанных с проведением практикума по MPI-программированию (см., например, [3–8]). В большинстве подобных разработок основное внимание уделяется вопросам реализации стандартных параллельных алгоритмов, относящихся к матричным вычислениям [3, 6, 7], численным методам интегрирования и решения дифференциальных уравнений [6, 7], параллельной реализации дискретного преобразования Фурье [5, 7] и т. п. Ряд заданий состоит в модификации имеющихся программ, в том числе и достаточно больших [4]. Отладка и тестирование программ нередко проводится на удаленном учебном вычислительном кластере [3]. При этом изучению собственно библиотеки MPI и ее многообразных возможностей уделяется недостаточно внимания. В ряде пособий описывается только минимально необходимый набор команд, обеспечивающий двусторонние коммуникации [4, 8], в других дополнительно рассматриваются базовые возможности, связанные с коллективными взаимодействиями [5, 7]. Задачи на создание новых коммуникаторов (в том числе на использование виртуальных топологий), если и присутствуют, то в крайне незначительном числе [3, 6], а средства, выходящие за рамки стандарта MPI 1.1, в практикумах обычно не применяются.

Между тем, начинать «практическое» изучение MPI было бы желательно именно с освоения возможностей самой библиотеки, и только потом переходить к применению этих возможностей для реализации сложных параллельных алгоритмов и к исследованию эффективности реализаций этих алгоритмов. Если, приступая к реализации алгоритмов, студент не будет иметь опыта работы с библиотекой MPI, он неизбежно будет допускать ошибки, обусловленные незнанием особенностей применения тех или иных функций данной библиотеки. В силу специфики разработки параллельных программ поиск и исправление таких ошибок может превратиться в очень сложную проблему, особенно при тестировании и отладке параллельных программ на удаленном кластере.

Для изучения библиотеки MPI целесообразно включать в практикум набор небольших задач, посвященных ее отдельным возможностям. Заметим, что подобные задачи присутствуют в ряде практикумов [3], но в незначительном количестве, которое не позволяет ознакомиться с важными особенностями тех или иных разделов библиотеки. Выполнение таких заданий не требует ресурсов суперкомпьютера или вычислительного кластера; удобнее использовать привычную для студента интегрированную среду на локальном компьютере, что существенно ускорит процесс разработки учебных программ.

Разумеется, этап начального освоения библиотеки MPI желательно пройти достаточно быстро. В этом может помочь специализированное обучающее средство — *электронный задачник* [9]. В данной работе рассматриваются различные аспекты реализации электронного задачника по параллельному программированию на основе технологии MPI, основанной на возможностях универсального электронного задачника по программированию Programming Taskbook.

Электронный задачник Programming Taskbook как платформа для разработки специализированных электронных задачников

Электронный задачник Programming Taskbook разрабатывался автором как компьютерная система, которая подключается к среде разработки и взаимодействует с учебными программами, выполняя следующие действия:

- генерация для требуемого учебного задания программы-заготовки и ее загрузка в используемую среду разработки;
- передача учебной программе автоматически сгенерированных наборов исходных данных и получение от нее результатов;
- дополнительный контроль правильности операций ввода-вывода;
- автоматическая проверка предложенного решения на полном наборе тестовых данных при однократном запуске учебной программы из среды разработки;
- наглядное отображение всей информации, связанной с учебным заданием.

Ключевым здесь является аспект *взаимодействия* задачника со средой разработки. Интегрированная среда разработки сама по себе является электронным ресурсом, используемым при изучении программирования, однако этот ресурс существует в отрыве от традиционных задачников по программированию, реализованных в лучшем случае в виде гипертекстовых документов. Для того чтобы ускорить и сделать более эффективным (и более интересным) процесс выполнения учебных заданий, и требуется интеграция задачника в среду программирования. В результате подобной интеграции используемая среда программирования превращается в учебную среду, сохраняя при этом все свои базовые

возможности, упрощающие разработку программ (контекстная справка, средства IntelliSense, встроенные средства отладки и т. д.).

Применяя современные технологии программирования, можно обеспечить такое важное свойство электронного задачника, как *универсальность*: возможность его адаптации к новым средам разработки и даже к новым языкам без коренной переработки. Это позволяет поддерживать задачник в актуальном состоянии, поскольку в противном случае он устарел бы одновременно с той средой разработки, на которую был ориентирован. При этом универсальность должна сочетаться с *расширяемостью*: возможностью включать в состав задачника новые группы заданий, причем таким образом, чтобы новые группы сразу становились доступными для всех поддерживаемых задачником языков и сред программирования (если специфика задач не накладывает ограничений на используемый язык).

Разработанный на основе перечисленных выше принципов электронный задачник Programming Taskbook не только позволил существенно ускорить и сделать более эффективным процесс изучения базовых тем школьного и вузовского курса программирования, но и стал использоваться как платформа для разработки специализированных электронных задачников, посвященных различным технологиям программирования, библиотекам или алгоритмам [10]. Первым примером подобного специализированного задачника стал электронный задачник по параллельному программированию для технологии MPI стандарта 1.1 [11].

Основные возможности электронного задачника по параллельному программированию

Первая версия электронного задачника по параллельному MPI-программированию Programming Taskbook for MPI (PT for MPI) была разработана в 2009 году.

Этот задачник обеспечивал выполнение всех перечисленных в предыдущем пункте действий; в частности, он передавал учебной программе исходные данные, проверял правильность результатов, полученных программой и сохранял сведения о каждом тестовом испытании программы в специальном файле. Кроме того, в задачнике PT for MPI были реализованы дополнительные возможности, связанные со спецификой выполнения заданий по параллельному программированию на основе передачи сообщений:

- демонстрационный просмотр заданий, не требующий использования параллельного режима;
- создание для выбранного задания проекта-заготовки с подключенными к нему модулями библиотеки MPI;
- особый механизм, обеспечивающий выполнение учебной программы в параллельном режиме при ее обычном запуске из среды разработки: запущенная программа выполняет запуск приложения MPIRun.exe из комплекса MPICH, которое, в свою очередь, запускает программу в параллельном режиме (все процессы выполняются на локальном компьютере);
- передача каждому процессу параллельной программы его собственного набора исходных данных;
- получение от каждого процесса требуемых результатов и их автоматическая пересылка в главный процесс для проверки и отображения в окне задачника;
- вывод информации об ошибках времени выполнения (в том числе ошибках, возникших при выполнении функций MPI) и ошибках ввода-вывода с указанием рангов процессов, в которых эти ошибки произошли;
- возможность вывода отладочной информации для каждого процесса в специальном разделе окна задачника;
- автоматическая выгрузка из памяти всех запущенных процессов даже в случае зависания параллельной программы.

Использование исходных данных, подготовленных задачником для каждого процесса параллельной программы, наглядный вывод в одном окне всех результатов, полученных каждым процессом, и их автоматическая проверка, а также дополнительные средства отладки позволили студенту сосредоточиться на алгоритме реализации параллельной программы, выполняющей задание, существенно ускорили разработку алгоритма и обеспечили его надежное тестирование.

В состав комплекса PT for MPI была включена группа учебных заданий MPIBegin, которая содержала 100 заданий, предназначенных для изучения возможностей библиотеки MPI 1.1. Кроме того, реализованные в задачнике PT for MPI средства автоматического запуска и отладки параллельных приложений дали возможность использовать его для разработки и тестирования параллельных программ, не связанных с конкретными учебными задачами. С этой целью в задачник PT for MPI была включена

вспомогательная группа MPIDebug из 36 заданий, каждое из которых обеспечивало запуск параллельной программы непосредственно из среды разработки, причем количество процессов определялось порядковым номером задания (от 1 до 36). При выполнении заданий на разработку стандартных параллельных алгоритмов студент мог использовать эту дополнительную возможность для написания и предварительной отладки на локальном компьютере своих параллельных программ.

Таким образом, применение комплекса PT for MPI позволило устранить все отмеченные выше проблемы, характерные для традиционной организации начальной части практикума по распределенному программированию.

Особенности реализации задачника по параллельному программированию с поддержкой стандарта MPI-2

В 2017 году был реализован новый вариант электронного задачника по параллельному программированию, получивший название Programming Taskbook for MPI-2 (PT for MPI-2). Его главной особенностью является наличие групп заданий, связанных с возможностями стандарта MPI версии 2.0.

Задания, включенные в задачник PT for MPI-2, могут выполняться на языке C++ во всех программных средах для этого языка, поддерживаемых задачником Programming Taskbook версии 4.17:

- Microsoft Visual Studio 2008, 2010, 2012, 2013, 2015, 2017;
- Code::Blocks версии 13 или выше.

Для компиляции и запуска в параллельном режиме применяются компоненты системы MPICH — одной из популярных свободно распространяемых систем поддержки MPI (Аргоннская национальная лаборатория США). Задачник PT for MPI-2 может использоваться совместно с двумя версиями данной системы для Windows: MPICH 1.2.5 (поддерживает стандарт MPI 1.1) и MPICH2 1.3 (поддерживает стандарт MPI 2.1). Все действия по подключению компонентов системы MPICH к учебным проектам задачник выполняет автоматически. Заметим, что использование более старой версии MPICH 1.2.5 оправдано, если изучаются только те возможности MPI, которые имеются в стандарте 1.1, и при этом используются компьютеры со сравнительно невысоким быстродействием.

В отличие от задачника PT for MPI, в состав которого входит одна большая группа MPIBegin, задачник PT for MPI-2 включает девять групп. Первые пять из них содержат все задания из группы MPIBegin и ряд новых заданий (всего 124 задания); заключительные четыре группы являются новыми и включают 126 заданий. Ниже для каждой группы приводится ее название, краткое описание и количество заданий, входящих в эту группу:

- MPI1Proc: процессы и их ранги (10 заданий);
- MPI2Send: обмен сообщениями между отдельными процессами (32);
- MPI3Coll: коллективные взаимодействия (28);
- MPI4Type: производные типы и упаковка данных (22);
- MPI5Comm: группы процессов и коммуникаторы (32);
- MPI6File: параллельный ввод-вывод файловых данных (MPI-2) (30);
- MPI7Win: односторонние коммуникации (MPI-2) (30);
- MPI8Inter: интеркоммуникаторы и динамическое создание процессов (MPI-2) (22);
- MPI9Matr: параллельные матричные алгоритмы (44).

Ряд заданий, перенесенных из группы MPIBegin, снабжен новыми указаниями.

При разработке новых (и модификации имеющихся) заданий широко использовались новые возможности электронного задачника Programming Taskbook последних версий, в частности, наличие варианта окна задачника с динамической компоновкой, позволяющей отображать на экране большие формулировки заданий и большие наборы исходных и результирующих данных, возможность добавления к формулировке задания иллюстраций (использована в заключительных заданиях группы MPI5Comm, посвященных топологиям графа), возможность создания специальных программ-заготовок для отдельных заданий (использована в группе MPI9Matr, посвященной параллельным матричным алгоритмам).

Задачник PT4 for MPI-2 включает конструктор, позволяющий разрабатывать новые групп заданий по параллельному программированию. Группы должны оформляться в виде динамических библиотек (dll-файлов); полученные библиотеки достаточно скопировать в рабочий каталог студента или в системный каталог задачника Programming Taskbook, чтобы они стали доступны для выполнения. Вариант конструктора, включенный в задачник PT4 for MPI-2, предназначен для разработки динамических библиотек в среде Microsoft Visual Studio на языке C++ и состоит из двух файлов:

pt4taskmaker.cpp и pt4taskmaker.h. Подробное описание процесса разработки новых заданий приводится в справочной системе PT for MPI-2 Info, также входящей в состав задачника.

Пример выполнения учебного задания

В качестве примера рассмотрим задание из подгруппы «Виртуальные топологии», входящей в группу MPI5Comm «Группы процессов и коммуникаторы».

MPI5Comm17. Число процессов K кратно трем: $K = 3N$, $N > 1$. В процессах 0, N и $2N$ дано по N целых чисел. Определить для всех процессов декартову топологию в виде матрицы размера $3 \times N$, после чего, используя функцию MPI_Cart_sub, расщепить матрицу процессов на три одномерные строки (при этом процессы 0, N и $2N$ будут главными процессами в полученных строках). Используя одну коллективную операцию пересылки данных, переслать по одному исходному числу из главного процесса каждой строки во все процессы этой же строки и вывести полученное число в каждом процессе (включая процессы 0, N и $2N$).

Приступая к выполнению задания, студент может выбрать требуемую среду программирования для языка C++ из числа имеющихся на данном компьютере и создать в ней проект-заготовку для нужного задания, используя специальный программный модуль PT4Load, входящий в состав задачника Programming Taskbook. Модуль PT4Load позволяет выбрать не только среду программирования, но и вариант системы MPICH, если на компьютере установлены оба поддерживаемые задачником версии. В случае, если выбрано задание по параллельному программированию, к созданному проекту автоматически подключаются модули с реализацией библиотеки MPI из комплекса MPICH, а в код генерируемых cpp-файлов добавляются операторы, необходимые практически в любой параллельной программе. Приведем текст файла MPI5Comm17.cpp, созданного в составе проекта для выполнения задания MPI5Comm17:

```
#include "pt4.h"
#include "mpi.h"

void Solve()
{
    Task("MPI5Comm17");
    int flag;
    MPI_Initialized(&flag);
    if (flag == 0)
        return;
    int rank, size;
    MPI_Comm_size(MPI_COMM_WORLD,
&size);
    MPI_Comm_rank(MPI_COMM_WORLD,
&rank);

}
```

В данном фрагменте указываются необходимые заголовочные файлы и определяется функция Solve, в которой требуется запрограммировать решение задачи. Первым оператором функции Solve является вызов функции Task, инициализирующей выбранное задание. Прочие операторы позволяют определить, запущено ли задание в параллельном режиме, а также узнать число процессов параллельного приложения и ранг текущего процесса. Другие фрагменты программного кода, в том числе стартовая функция main, не требуют редактирования в процессе выполнении задания и поэтому вынесены в другие файлы проекта. Отсутствие в функции Solve вызовов функций MPI_Init и MPI_Finalize, инициализирующих и завершающих параллельный режим, объясняется тем, что задачник самостоятельно вызывает эти функции, поскольку и до, и после вызова функции Solve он выполняет ряд действий в параллельном режиме, описываемых далее.

Для выполнения задания MPI5Comm17 в функцию Solve достаточно добавить следующий фрагмент кода:

```
MPI_Comm comm;
int dims[] = {3, size / 3},
periods[] = {0, 0};
MPI_Cart_create(MPI_COMM_WORLD,
2, dims, periods, 0, &comm);
MPI_Comm comm_sub;
int remain_dims[] = {0, 1};
MPI_Cart_sub(comm, remain_dims,
&comm_sub);
MPI_Comm_size(comm_sub, &size);
MPI_Comm_rank(comm_sub, &rank);
int b, *a = new int[size];
if (rank == 0)
  for (int i = 0; i < size; ++i)
    pt >> a[i];
MPI_Scatter(a, 1, MPI_INT, &b, 1,
MPI_INT, 0, comm_sub);
pt << b;
delete[] a;
```

Операторы ввода-вывода, использующие специальный поток pt, позволяют учебной программе получать исходные данные, подготовленные задачником, и передавать задачнику результаты для их проверки и отображения на экране.

Как уже было отмечено при описании основных возможностей задачника, запуск разработанной программы непосредственно из интегрированной среды включает специальный механизм, обеспечивающий выполнение

программы в параллельном режиме (при этом сам задачник выбирает количество используемых процессов, учитывая особенности решаемой задачи). При однократном запуске программы из интегрированной среды сразу выполняется проверка ее параллельного варианта на нескольких тестовых наборах исходных данных, которые генерируются задачником с использованием датчика случайных чисел. Проверка выполняется до тех пор, пока не будет обнаружена ошибка или пока не будет пройдено требуемое количество тестов (равное пяти).

Результаты выполнения учебной программы немедленно отображаются на экране (рис. 1). При этом студент может ознакомиться с предложенным набором исходных данных, результирующими данными, полученными каждым процессом его параллельной программы, а также контрольным («правильным») набором результатов. Помимо вывода на экран результаты каждого запуска программы фиксируются в специальном зашифрованном файле результатов, просмотреть который можно с помощью программного модуля PT4Results, входящего в состав задачника. Итак, разработав короткую программу и использовав очень простые средства для ввода и вывода, студент на практике смог ознакомиться с особенностями использования достаточно сложной функции MPI_Cart_sub, а также еще раз воспользоваться ранее изученной функцией коллективного обмена MPI_Scatter, применив ее к созданным коммуникаторам. Следует подчеркнуть, что перед началом выполнения функции Solve задачник распределяет все исходные данные по тем процессам, в которых их требуется ввести, а после завершения функции Solve он автоматически собирает из различных процессов полученные в них результаты для того чтобы проанализировать их и отобразить в наглядном виде в окне задачника (которое связано с главным процессом). Таким образом, благодаря применению задачника существенно упрощаются те этапы выполнения задания, которые не связаны с разработкой алгоритма, а именно ввод исходных данных и вывод результатов.

Важной функцией как базового варианта задачника Programming Taskbook, так и его дополнения — задачника по параллельному программированию — является автоматическая проверка правильности операций ввода-вывода и отображение в окне задачника сообщений о различных видах обнаруженных ошибок. При выполнении заданий по параллельному программированию задачник выводит описание ошибок, выявленных в каждом процессе параллельного приложения.

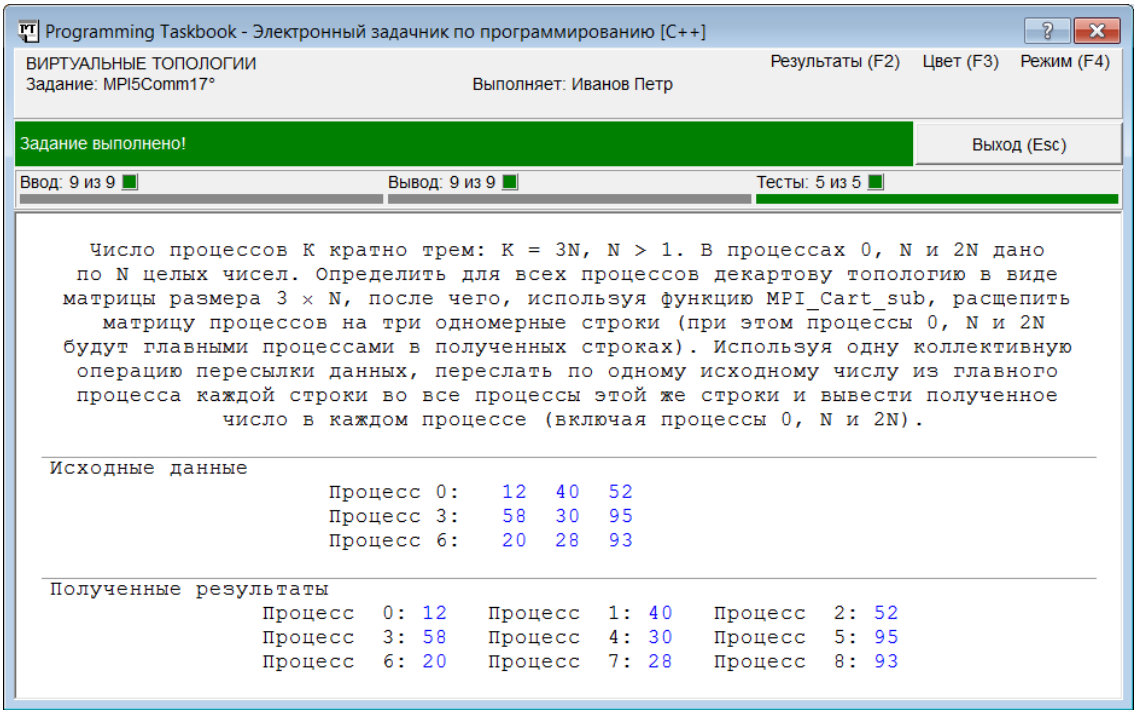

Рис. 1. Вид окна задачника Programming Taskbook при успешном выполнении задания MPI5Comm17

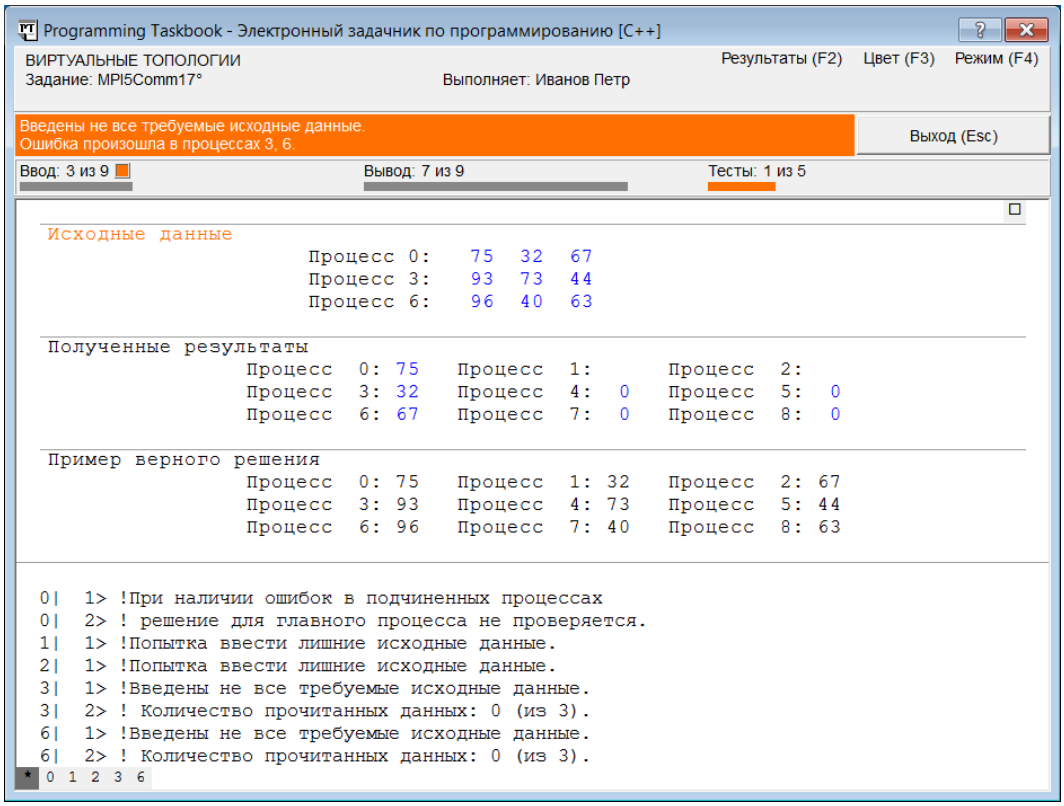

Рис. 2. Вид окна задачника Programming Taskbook при ошибочном выполнении задания MPI5Comm17

В качестве примера на рис. 2 приводится окно задачника при попытке выполнить задание MPI5Comm17, указав неверные значения массива флагов remain_dims:

```
int remain_dims[] = {1, 0};
```

В целях уменьшения размеров окна в нем скрыта формулировка задания. Все выявленные ошибки отображаются в специальном разделе отладки, который появляется в нижней части окна задачника (в начале каждой строки раздела отладки указан ранг процесса, с которым связано соответствующее сообщение). В информационном разделе в верхней части окна выводится информация об одном из видов обнаруженных ошибок, а ниже, в разделе индикаторов, указывается, сколько элементов исходных данных и результатов было введено и выведено программой.

Следует заметить, что студент сам может выводить в разделе отладки любые данные, используя входящие в состав задачника функции Show и ShowLine; при этом данные связываются с тем процессом, в котором они были выведены. Используя набор маркеров, расположенных на нижней границе окна, можно переключаться к фрагменту окна отладки, связанному с процессом требуемого ранга; маркер «*» позволяет просмотреть полное содержимое окна отладки, включающее данные для всех процессов.

Указанные возможности существенно облегчают отладку параллельных программ и тем самым ускоряют время выполнения учебных заданий.

Группы заданий, связанные с новыми возможностями стандарта MPI-2

Задачник PT for MPI-2 содержит большое количество заданий, связанных с новыми средствами, включенными в стандарт MPI-2. Эта особенность отличает его от большинства образовательных ресурсов, посвященных технологии MPI, поскольку они обычно ориентируются на стандарт MPI 1.1.

Именно с целью поддержки новых возможностей в число доступных для задачника систем MPICH была включена система MPICH2 версии 1.3, которая поддерживает стандарт MPI-2 в полном объеме.

Средствам стандарта MPI-2 посвящены группы MPI6File (параллельный файловый ввод-вывод), MPI7Win (односторонние коммуникации) и MPI8Inter (интеркоммуникаторы и динамическое порождение процессов). Кроме того, задания, связанные с возможностями MPI-2, добавлены в некоторые группы, посвященные базовым средствам MPI. Такова последняя подгруппа группы MPI4Type, посвященная новой коллективной функции MPI_Alltoallw, и последняя подгруппа группы MPI5Comm, посвященная новому виду виртуальной

топологии — топологии распределенного графа.

Задания каждой из новых групп охватывают практически все возможности, связанные с изучаемым разделом MPI-2. Так, в группе MPI6File вначале изучаются варианты локальных функций файлового ввода-вывода (MPI_File_read_at и MPI_File_write_at вместе со вспомогательной функцией MPI_File_get_size и MPI_File_read и MPI_File_write вместе со вспомогательными функциями MPI_File_seek и MPI_File_get_position), затем варианты коллективных функций (MPI_File_read_all и MPI_File_write_all, MPI_File_read_at_all и MPI_File_write_at_all, MPI_File_read_ordered и MPI_File_write_ordered вместе со вспомогательной функцией MPI_File_seek_shared), а завершающая подгруппа содержит большое количество заданий на создание и использование сложных вариантов файловых видов данных (определяемых с помощью функции MPI_File_set_view).

В качестве иллюстрации приведем формулировки двух заданий из завершающей подгруппы группы MPI6File.

MPI6File21. В главном процессе дано имя существующего файла целых чисел, содержащего $3K$ элементов, где K — количество процессов. В каждом процессе прочесть и вывести три элемента A, B, C, расположенных в исходном файле в следующем порядке (индекс указывает ранг процесса): A_0, A_1, …, A_{K-1}, B_0, B_1, …, B_{K-1}, C_0, C_1, …, C_{K-1}. Для этого использовать единственный вызов коллективной функции MPI_File_read_all, предварительно определив новый файловый вид данных с базовым типом MPI_INT, подходящим смещением (своим для каждого процесса) и новым файловым типом, состоящим из одного целочисленного элемента и завершающего пустого промежутка, размер которого равен протяженности набора из $K - 1$ целого числа.

MPI6File30. В главном процессе дано имя файла и число N, которое может лежать в диапазоне от 2 до 5. Кроме того, в каждом процессе даны два целых числа I и J, определяющие позицию (номер строки и столбца) некоторого квадратного блока блочной прямоугольной матрицы размера $(K/3) \times 3$ блоков, где K — количество процессов (число K кратно 3). Значения I лежат в диапазоне от 1 до $K/3$, значения J лежат в диапазоне от 1 до 3; все процессы содержат различные позиции блоков. Каждый блок представляет собой квадратную матрицу целых чисел порядка N. Создать новый файл целых чисел с указанным именем и записать в него блочную матрицу размера $(K/3) \times 3$, причем каждый процесс должен записать в матрицу блок в позиции (I, J), а все элементы блока, записанного процессом ранга R ($R = 0, 1, \ldots, K - 1$), должны быть равны числу R. Для этого использовать единственный вызов функции MPI_File_write_all, предварительно определив новый файловый вид данных с базовым типом MPI_INT, подходящим смещением (своим для каждого процесса) и новым файловым типом, состоящим из K целочисленных элементов и завершающего пустого промежутка подходящего размера. Для передачи значения N во все процессы использовать коллективную функцию MPI_Bcast.

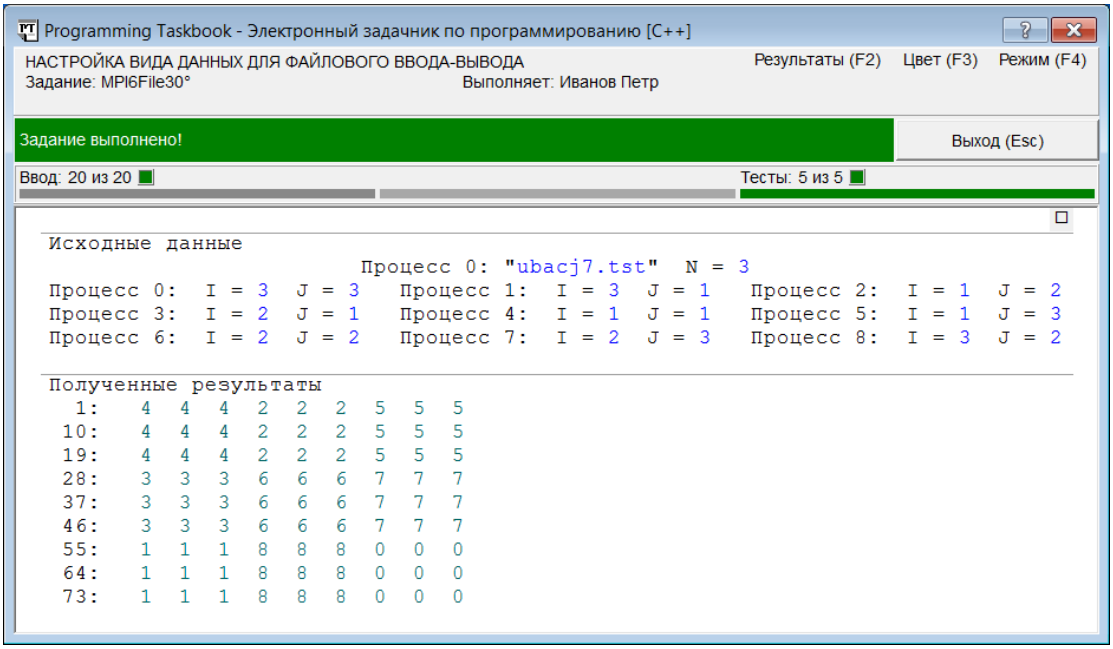

Рис. 3. Вид окна задачника Programming Taskbook при успешном выполнении задания MPI6File30

На рис. 3 приведено окно задачника с сообщением об успешном выполнении задания MPI6File30.

При реализации заданий, связанных с параллельным файловым вводом-выводом, была использована возможность базового варианта электронного задачника Programming Taskbook, позволяющая включать в задания двоичные и текстовые файлы в качестве элементов исходных и результирующих данных [12]. Исходные файлы создаются задачником при каждом запуске программы, а файлы, созданные учебной программой, автоматически проверяются. Кроме того, содержимое всех файлов, используемых в задании, отображается в окне задачника в наглядном виде, что позволяет студенту, в частности, легко обнаружить ошибки в содержимом тех файлов, которые были созданы при выполнении задания. Следует подчеркнуть, что в заданиях группы MPI6File (как и в средствах MPI, связанных с файловым вводом-выводом) используются двоичные файлы, содержимое которых нельзя просмотреть в обычных текстовых редакторах.

В полном объеме представлены в задачнике PT for MPI-2 и возможности стандарта MPI-2, связанные с удаленным доступом к памяти (односторонними коммуникациями). Посвященная этой теме группа MPI7Win состоит из двух подгрупп. В первой подгруппе изучаются варианты односторонних коммуникаций с простейшей синхронизацией, основанной на применении коллективной функции MPI_Win_fence, и рассматриваются все возможные виды односторонних пересылок (MPI_Get, MPI_Put, MPI_Accumulate) при наличии как одного, так и нескольких созданных окон доступа. Во второй подгруппе изучаются более сложные виды синхронизации, использующие либо четверку функций MPI_Win_start, MPI_Win_complete, MPI_Win_post, MPI_Win_wait, либо функции MPI_Win_lock и MPI_Win_unlock. В качестве иллюстрации приведем одно задание из каждой подгруппы.

MPI7Win16. В каждом процессе дана одна строка вещественной квадратной матрицы A порядка K, где K — количество процессов (процесс ранга R содержит R-ю строку матрицы в предположении, что строки нумеруются от 0). Кроме того, в каждом процессе дано вещественное число B. Во всех процессах определить окно доступа, содержащее строку матрицы A, и, используя требуемое число вызовов функции MPI_Accumulate в каждом процессе ранга R ($R = 0, \ldots, K - 1$), заменить в строке матрицы из следующего процесса все элементы, меньшие числа B из процесса R, на это число (процессы перебираются циклически). Затем, используя K вызовов функции MPI_Get в каждом процессе, получить и вывести столбец преобразованной матрицы A с номером, совпадающим с рангом процесса (столбцы также нумеруются от 0). **Указание**. При выполнении этого задания в каждом процессе необходимо *трижды* вызывать функцию синхронизации MPI_Win_fence.

MPI7Win21. Количество процессов K — четное число. В главном процессе дан массив A из $K/2$ вещественных чисел и массив N целых чисел того же размера. Все элементы массива N различны и лежат в диапазоне от 1 до $K - 1$. В каждом подчиненном процессе определить окно доступа из одного вещественного числа и, используя требуемое количество вызовов функции MPI_Put в главном процессе, переслать в каждый из подчиненных процессов ранга N_I ($I = 0, \ldots, K/2 - 1$) число A_I и вывести полученное число (в остальных подчиненных процессах вывести вещественное число 0.0). Для синхронизации использовать функции MPI_Win_post и MPI_Win_wait в подчиненных процессах и функции MPI_Win_start и MPI_Win_complete в главном процессе.

Завершает набор групп, посвященных новым возможностям стандарта MPI-2, группа MPI8Inter, в которой описываются средства, связанные с интеркоммуникаторами и динамическим созданием процессов. В первой подгруппе данной группы рассматриваются различные варианты создания интеркоммуникаторов, причем особое внимание уделяется функциям MPI_Comm_create и MPI_Comm_split, возможности которых по созданию интеркоммуникаторов были существенно расширены в стандарте MPI-2. Вторая подгруппа посвящена коллективным операциям для интеркоммуникаторов, которые также появились только в стандарте MPI-2. Наконец, третья подгруппа связана с динамическим порождением процессов и включает различные варианты подобного порождения, а также действия по последующей перекомпоновке имеющихся процессов в новые коммуникаторы, в том числе с применением клиент-серверного механизма объединения групп процессов (с помощью функций MPI_Open_port, MPI_Publish_name и MPI_Comm_accept на стороне группы-сервера и функций MPI_Lookup_name и MPI_Comm_connect на стороне группы-клиента).

С целью поддержки заданий на динамическое создание процессов были дополнены возможности задачника, связанные с применением раздела отладки: в этот раздел

могут записывать данные не только исходные процессы параллельного приложения, но и те новые процессы, которые были созданы в ходе работы приложения, причем каждый из таких процессов снабжается уникальным идентификатором, позволяющим однозначно идентифицировать данные, выведенные в этом процессе.

Приведем формулировку одного из начальных заданий подгруппы, связанной с порождением новых процессов.

MPI8Inter16. В каждом процессе дан массив из K вещественных чисел, где K — количество процессов. Используя один вызов функции MPI_Comm_spawn с первым параметром «ptprj.exe», создать K новых процессов. С помощью коллективной функции MPI_Reduce_scatter_block переслать в созданный процесс ранга R ($R = 0, \ldots, K - 1$) максимальный из элементов исходных массивов с индексом R и отобразить полученные максимальные элементы в разделе отладки, используя в каждом новом процессе функцию Show. Затем с помощью функций MPI_Send и MPI_Recv переслать найденный максимальный элемент из нового процесса ранга R ($R = 0, \ldots, K - 1$) в исходный процесс того же ранга и вывести полученные элементы в исходных процессах.

Окно задачника при правильном решении задачи MPI8Inter16 приведено на рис. 4. В этом окне следует обратить внимание на содержимое раздела отладки, в котором выведены данные из новых процессов, получивших идентификаторы «a0»–«a5».

Группа заданий на разработку параллельных матричных алгоритмов

Особое место в задачнике PT for MPI-2 занимает последняя группа MPI9Matr. В то время как предыдущие группы предназначены для изучения возможностей библиотеки MPI, данная группа посвящена параллельным алгоритмам (а именно алгоритмам матричного умножения), при реализации которых можно использовать различные средства MPI.

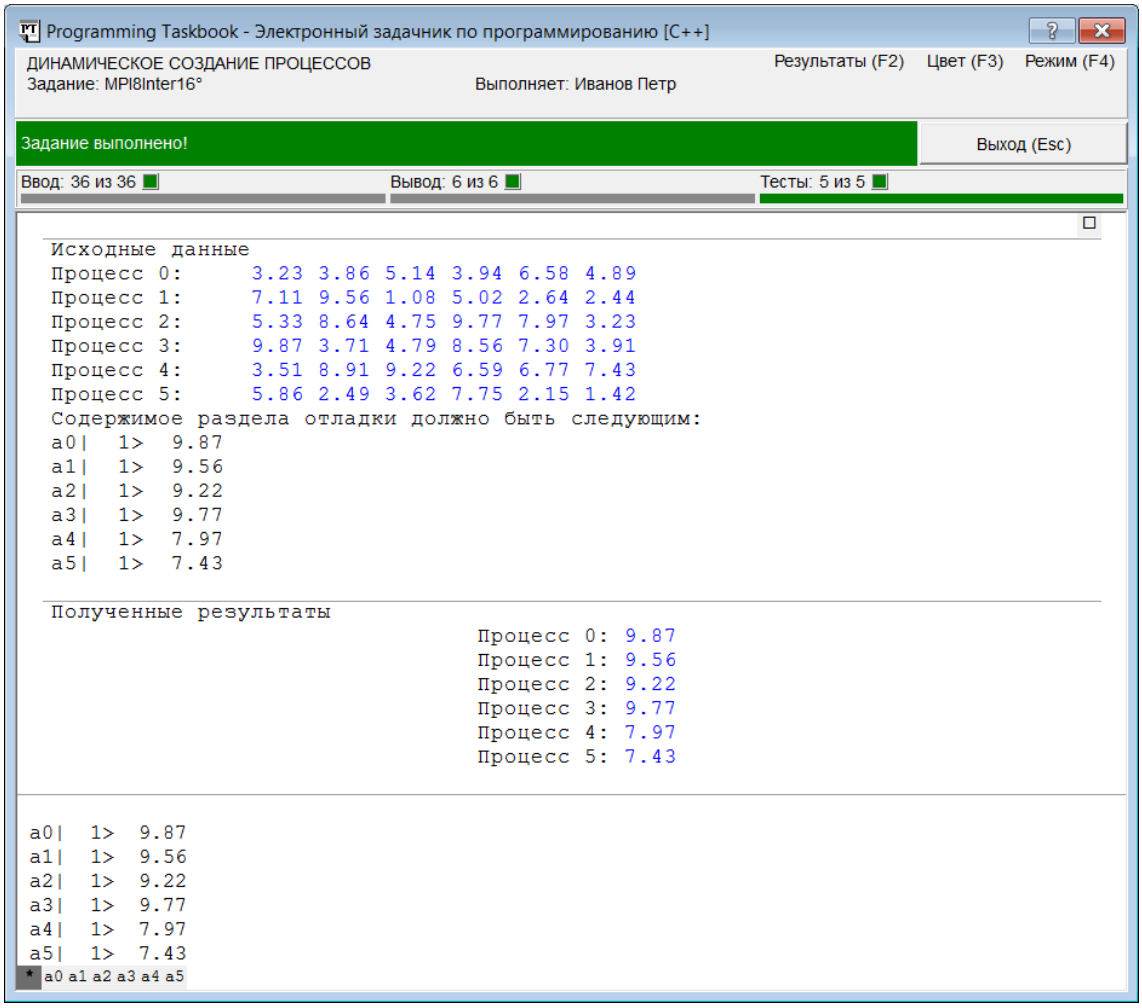

Рис. 4. Вид окна задачника Programming Taskbook при успешном выполнении задания MPI8Inter16

Таким образом, при выполнении заданий из этой группы студент сможет изучить важный класс параллельных алгоритмов, применяя при этом многие из ранее изученных возможностей MPI.

Хотя матричные алгоритмы являются наиболее часто используемой и хорошо проработанной темой практикумов по параллельному программированию (см., например, [3, 6, 7]), применение электронного задачника позволяет сделать изучение этой темы более эффективным. Связано это с тем, что сложные алгоритмы состоят из нескольких этапов, и проверка правильности выполнения каждого этапа является достаточно трудоемкой задачей (как для студента, так и для преподавателя). В то же время, подобная проверка необходима, поскольку любые ошибки, допущенные студентом при реализации какого-либо этапа, не позволят реализовать алгоритм в полном объеме.

Электронный задачник позволяет упростить процесс разработки сложного алгоритма путем его разбиения на последовательность сравнительно простых заданий, каждое из которых посвящено реализации одного из этапов алгоритма. При этом в каждом задании задачник может автоматически генерировать тот набор данных, который должен быть получен к началу прохождения изучаемого этапа алгоритма; кроме того, он обеспечивает автоматическую проверку правильности результатов, полученных при выполнении данного этапа. Такой подход, с одной стороны, позволяет сделать учебные программы более краткими и независимыми друг от друга (поскольку для реализации любого последующего этапа алгоритма не требуется включать в программу все предыдущие этапы), а с другой стороны, позволяет надежно и быстро проверить правильность реализации каждого этапа. Разумеется, следует предусматривать и итоговые задания, в которых алгоритм надо реализовать в полном объеме; эти задания не будут вызывать у студента особых трудностей, если он уже выполнил задания, связанные с каждым из этапов изучаемого алгоритма. Можно еще более упростить выполнение итоговых заданий, если в каждом из предыдущих заданий потребовать, чтобы решение было представлено в виде отдельной функции; тогда в итоговом задании будет достаточно вызвать ранее разработанные функции в требуемом порядке. Подобный подход, помимо прочего, позволяет выработать у студента навыки модульной разработки сложных программ.

Описанные выше идеи, связанные с использованием электронного задачника при изучении алгоритмов, ранее были положены в основу электронного задачника Programming Taskbook for Bioinformatics, посвященного строковым алгоритмам поиска и неточного сопоставления [13–14]. Такой же подход использован и при составлении заданий группы MPI9Matr. Кроме вводного задания, посвященного реализации непараллельного варианта алгоритма матричного умножения, группа состоит из 4 подгрупп, каждая из которых посвящена одной из модификаций параллельного алгоритма. Рассмотрены два варианта ленточного алгоритма и два варианта блочного алгоритма (алгоритмы Кэннона и Фокса).

Задания в каждой подгруппе организованы по единой схеме. Вначале даются задания, в которых предлагается реализовать вспомогательные функции, требуемые на каком-либо этапе алгоритма. Например, подгруппа, посвященная блочному алгоритму Кэннона, начинается с двух вводных заданий. В первом требуется описать вспомогательную функцию для определения нового типа данных MPI, с помощью которого можно организовать пересылку прямоугольного блока исходной матрицы без его промежуточного копирования во вспомогательный массив. Во втором вводном задании надо описать вспомогательную функцию, позволяющую определить на исходном наборе процессов топологию двумерной квадратной циклической решетки.

Затем идет задание, в котором требуется ввести исходные матрицы в главном процессе и организовать пересылку нужных фрагментов (полос или блоков) во все процессы приложения, оформив эту пересылку в виде вспомогательной функции. В последующих заданиях исходные данные уже распределены по процессам самим задачником, и надо реализовать один из этапов алгоритма умножения. Например, для алгоритма Фокса имеется задание, в котором блок матрицы A требуется переслать во все процессы соответствующей строки декартовой решетки (первый этап алгоритма), и задание, в котором требуется выполнить перемножение блоков и циклический сдвиг блоков B для каждого столбца декартовой решетки (второй этап). В последующих заданиях надо организовать выполнение этих этапов в цикле. Отдельное задание посвящено пересылке полученных результатов в главный процесс. В итоговом задании требуется объединить все ранее разработанные этапы алгоритма.

Кроме того, в каждой подгруппе предусмотрен дополнительный набор заданий, в

котором исходные матрицы A и B надо прочесть из исходных файлов и сохранить в новом файле полученное произведение, используя средства параллельного файлового ввода-вывода. В соответствующем итоговом задании надо объединить этапы, связанные с файловой обработкой, с этапами параллельного умножения, реализованными в предыдущих заданиях.

Чтобы обеспечить большее разнообразие задач, для каждого из рассматриваемых алгоритмов матричного умножения предлагается использовать различные средства MPI. В более простом варианте ленточного алгоритма, в котором матрицы разбиваются на горизонтальные полосы, не требуется определять новые типы данных, тогда как в более сложном варианте (использующем для одной матрицы горизонтальные, а для другой — вертикальные полосы) необходимо определить вспомогательный тип MPI (соответствующие типы MPI требуется также определять и при реализации блочных алгоритмов). Далее, в блочных алгоритмах (в отличие от ленточных) необходимо применять вспомогательные коммуникаторы с виртуальной топологией, причем в алгоритме Кэннона достаточно определить топологию двумерной решетки, а в алгоритме Фокса требуется дополнительно выполнить расщепление этой решетки на строки и столбцы. В алгоритме Фокса и алгоритме Кэннона требуется использовать разные средства для начальной пересылки блоков в подчиненные процессы и итогового объединения фрагментов полученного произведения: в одном случае это функции MPI_Send и MPI_Recv, в другом — коллективная функция MPI_Alltoallw.

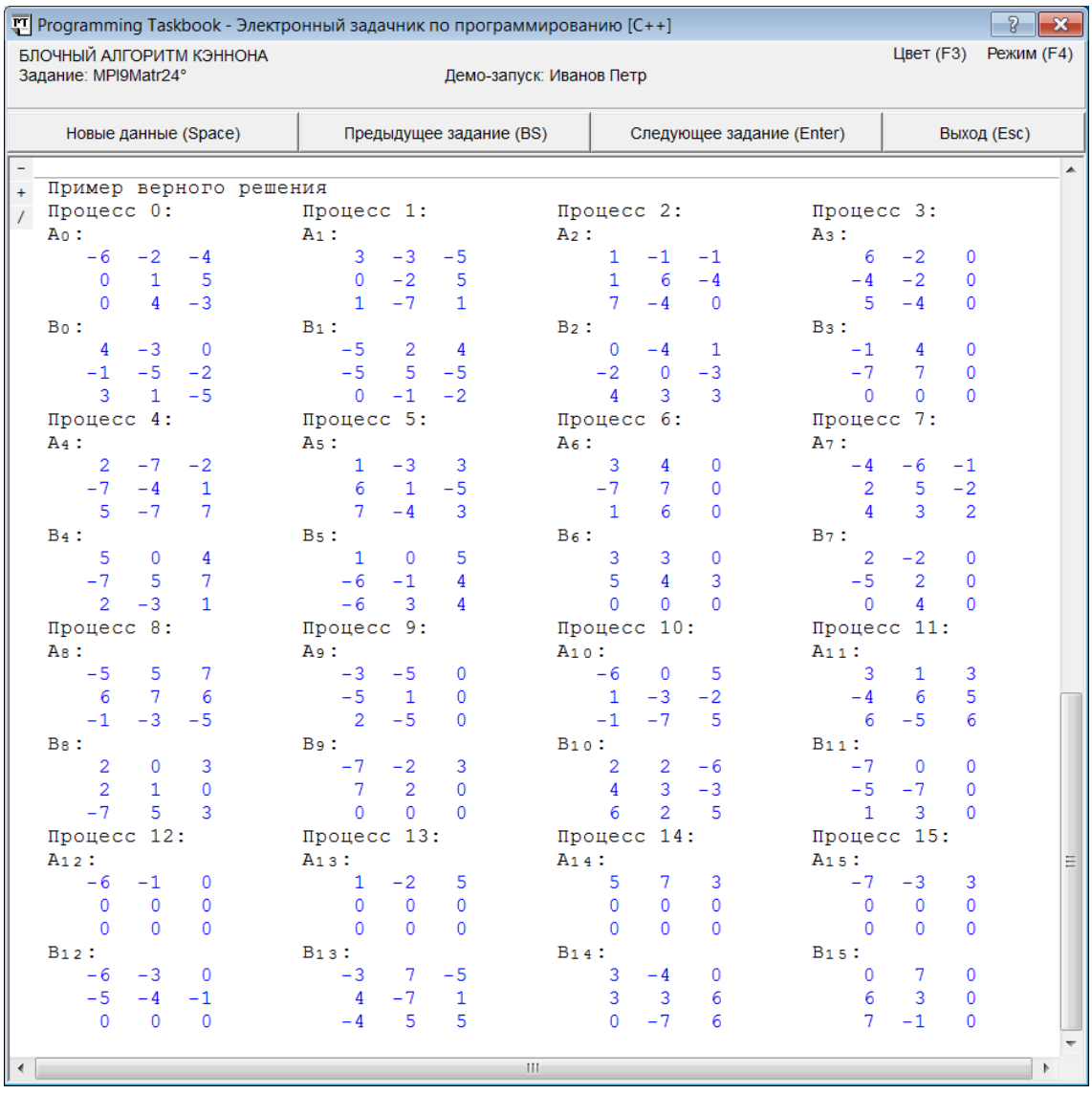

Рис. 5. Вид нижней части окна задачника с примером верного решения задачи MPI9Matr24

Существенно упрощает выполнение заданий то обстоятельство, что задачник обеспечивает наглядное отображение в своем окне как исходных данных, так и результатов, полученных программой. В качестве иллюстрации можно привести вид окна с примером верного решения одного из заданий подгруппы, связанной с алгоритмом Кэннона, в котором требуется организовать начальное перераспределение блоков по процессам приложения (рис. 5).

Заключение

Описанные выше возможности электронного задачника по параллельному программированию Programming Taskbook for MPI-2 позволяют применять его на различных уровнях обучения как в бакалаврских, так и в магистерских образовательных программах. На начальном уровне можно ограничиться возможностями стандарта MPI-1, на продвинутом уровне основное внимание можно уделить средствам, появившимся в стандарте MPI-2. Изучение матричных алгоритмов можно начать с заданий, включенных в группу MPI9Matr, после чего выполнить адаптацию полученных программ для выполнения на учебном кластере. Интересным вариантом учебных заданий продвинутого уровня могут стать задания на разработку новых групп учебных заданий с использованием конструктора учебных заданий PT4TaskMaker. Эти задания могут быть посвящены реализации других видов параллельных алгоритмов (например, алгоритмов решения краевых задач или задач взаимодействия набора тел) и оформлены по образцу задач группы MPI9Matr.
При использовании задачника PT for MPI-2 на лабораторных занятиях в распоряжении преподавателя имеется весь арсенал вспомогательных средств, предусмотренных для задачника Programming Taskbook [15]. В частности, можно легко генерировать варианты индивидуальных заданий (поскольку в каждой группе заданий, входящих в задачник PT for MPI-2, имеется не менее двух, а чаще три или четыре однотипных задачи, посвященные той или иной возможности библиотеки MPI). Разработанные программные средства для преподавателя входят в комплекс Teacher Pack и подробно описываются в соответствующем разделе сайта электронного задачника http://ptaskbook.com. С этого сайта можно скачать и дистрибутивы как базового задачника Programming Taskbook, так и задачника Programming Taskbook for MPI-2.

Благодарности

Электронный задачник Programming Taskbook for MPI-2 разработан при поддержке Стипендиальной программы Благотворительного фонда В. Потанина в рамках выполнения проекта «Электронный задачник по параллельному MPI-программированию» — одного из проектов-победителей грантового конкурса для преподавателей магистратуры 2016–2017 года в номинации «Новые методы и специальные навыки в обучении».

Литература

1. MPI: A Message-Passing Interface Standard. Version 1.1: June, 1995 / Message Passing Interface Forum, 2003. 238 pp. [Электронный ресурс] URL: http://mpi-forum.org/docs/mpi-1.1/mpi1-report.pdf (дата обращения 14.08.2017).
2. MPI: A Message-Passing Interface Standard. Version 2.2 / Message Passing Interface Forum, 2009. 647 pp. [Электронный ресурс] URL: http://mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf (дата обращения 14.08.2017).
3. Антонов А.С. Вычислительный практикум по технологии MPI. [Электронный ресурс] URL: https://parallel.ru/tech/tech_dev/MPIcourse (дата обращения 14.08.2017).
4. Оленев Н.Н. Виртуальный курс «Параллельное программирование в интерфейсе MPI». — Вычислительный центр им. А.А Дородницына РАН. [Электронный ресурс] URL: http://www.ccas.ru/mmes/educat/lab04k/ (дата обращения 14.08.2017).
5. Практикум по параллельному программированию / С.В. Борзунов, С.Д.Кургалин, А.В.Флегель. — СПб.: БХВ, 2017. — 236 с.
6. Рычков А.Д. Лабораторные работы по параллельным вычислительным технологиям. — Новосибирск, 2013. — 45 с.
7. Сысоев А.В. Высокопроизводительные вычисления в учебном процессе и научных исследованиях. — Нижний Новгород, 2006. — 90 с.
8. Удалова Ю.В., Кузьмин Д.А. Параллельное программирование: лабораторный практикум. — Красноярск: СФУ, 2012. [Электронный ресурс] URL: http://s3.docme.ru/store/data/001155098.pdf?key=61c279035a8e4c22c955578ebdc4feb6&r=1&fn=1155098.pdf&t=1502646039966&p=600 (дата обращения 14.08.2017).
9. Абрамян М.Э. Об архитектуре универсального электронного задачника по программированию // Информатизация образования и науки. — 2015, № 3 (27). — С. 134–150.
10. Абрамян М.Э. Об использовании задачника Programming Taskbook в качестве платформы для разработки специализированных электронных задачников / Ершовская конференция по информатике 2014. Секция «Информатика образования». Новосибирск: Изд-во СО РАН, 2014. — С. 1–8.
11. Абрамян М.Э. Электронный задачник по параллельному программированию на основе технологии MPI // Компьютерные инструменты в образовании. — 2011, № 6. — С. 47–54.
12. Абрамян М.Э. Использование электронного задачника при выполнении заданий, связанных с обработкой файловых данных // Компьютерные инструменты в образовании. — 2014, № 3. — С. 45–57.
13. Абрамян М.Э. Реализация электронного задачника по строковым алгоритмам биоинформатики // Компьютерные

инструменты в образовании. — 2012. № 2. — С. 49–58.
14. Абрамян М.Э. Использование электронного задачника по строковым алгоритмам биоинформатики // Компьютерные инструменты в образовании. — 2012. № 3. — С. 47–56.
15. Абрамян М.Э. Использование специализированного программного обеспечения для преподавателя при организации и проведении лабораторных занятий по программированию // Информатика и образование. — 2011, № 5. — С. 78–80.

References

1. MPI: A Message-Passing Interface Standard. Version 1.1: June, 1995 / Message Passing Interface Forum, 2003. 238 pp. [Electronic resource] URL: http://mpi-forum.org/docs/mpi-1.1/mpi1-report.pdf (date of view 14.08.2017).
2. MPI: A Message-Passing Interface Standard. Version 2.2 / Message Passing Interface Forum, 2009. 647 pp. [Electronic resource] URL: http://mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf (date of view 14.08.2017).
3. Antonov A.S. Computing workshop on MPI technology (in Russian). [Electronic resource] URL: https://parallel.ru/tech/tech_dev/MPIcourse (date of view 14.08.2017).
4. Olenev N.N. Virtual course «Parallel programming in MPI interface» (in Russian). — Dorodnitsyn Computing Centre, RAS. [Electronic resource] URL: http://www.ccas.ru/mmes/educat/lab04k/ (date of view 14.08.2017).
5. Workshop on parallel programming (in Russian) / S.V. Borzunov, S.D.Kurgalin, A.V.Flegel. — SPb.: BHV Publ., 2017. — 236 pp.
6. Rychkov A.D. Labs on parallel computing technologies (in Russian). — Novosibirsk, 2013. — 45 pp.
7. Sysoev A.V. High-performance computing in education and science (in Russian). — Nizhny Novgorod, 2006. — 90 pp.
8. Udalova Yu.V., Kuzmin D.A. Parallel Programming: Workshop (in Russian). — Krasnoyarsk: SibFU, 2012. [Electronic resource] URL: http://s3.docme.ru/store/data/001155098.pdf?key=61c279035a8e4c22c955578ebdc4feb6&r=1&fn=1155098.pdf&t=1502646039966&p=600 (date of view 14.08.2017).
9. Abramyan M.E. On the architecture of the universal problem book on programming (in Russian) // Informatizaciya obrazovaniya i nauki. — 2015, № 3 (27). — P. 134–150.
10. Abramyan M.E. On using the Programming Taskbook as a platform for specialized educational software development (in Russian) / A.P. Ershov Informatics Conference 2014. Educational Informatics Workshop. Novosibirsk: SB RAS Publ., 2014. — P. 1–8.
11. Abramyan M.E. The electronic book of educational training tasks on parallel MPI programming (in Russian) // Kompjuternye instrumenty v obrazovanii. — 2011, № 6. — P. 47–54.
12. Abramyan M.E. The application of the electronic taskbook in the study of data file processing (in Russian) // Kompjuternye instrumenty v obrazovanii. — 2014, № 3. — P. 45–57.
13. Abramyan M.E. The electronic book of educational training tasks on string algorithms of bioinformatics (in Russian) // Kompjuternye instrumenty v obrazovanii. — 2012. № 2. — P. 49–58.
14. Abramyan M.E. Using of the electronic book of educational training tasks on string algorithms of bioinformatics (in Russian) // Kompjuternye instrumenty v obrazovanii. — 2012. № 3. — P. 47–56.
15. Abramyan M.E. Using teacher-assisted software for laboratory classes on programming (in Russian) // Informatika i obrazovanie. — 2011, № 5. — P. 78–80.

Поступила: 31.08.2017

Об авторе:

Абрамян Михаил Эдуардович, кандидат физико-математических наук, доцент кафедры алгебры и дискретной математики Института математики, механики и компьютерных наук им. И.И. Воровича, Южный федеральный университет, mabr@math.sfedu.ru

Note on the author:

Abramyan Mikhail, Candidate of Physical and Mathematical Sciences, Associate Professor of the Algebra and Discrete Mathematics Department of the Vorovich Institute of Mathematics, Mechanics and Computer Science, Southern Federal University, mabr@math.sfedu.ru