**BC/NW 2018 № 2 (33):7.1**

**РАЗРАБОТКА ПРОГРАММЫ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ, ИСПОЛЬЗУЮЩИХ MPI ФУНКЦИИ**

Кожемякин А.В.

Ошибки, возникающие в связи с использованием *MPI* функций, чаще всегосвязаны с неправильной пересылкой данных между процессами: пересылка данных не в том формате, который нужен, ожидание данных от процесса, который ничего не посылает, неиспользование части параллельных процессов. Подобные ошибки не всегда нарушают работу программы, но могут искажать результаты вычислений или снижать ее быстродействие.

Целью работы является разработка приложения анализирующего правильность использования функций *MPI* и правильность задания параметров этих функций. Разработанное приложение должно анализировать правильность использования функций *MPI* [1] на примере модели параллельной программы из четырёх процессов и правильность задания параметров этих функций. Предполагается, что проверяемая программа работоспособна и задача анализа должна сводиться к проверке корректности ее работы.

Для создания данного программного продукта использовалась среда *Visual Studio* *2010* [2]. *Visual Studio* *2010* это интегрированная среда, упрощающая создание, отладку и развертывание приложений, включает поддержку языков *C#* *4.0* и *Visual Basic .NET* *10*.*0*, а также языка *F#*.

Разработка состоит из двух частей:

1. Заголовочный файл *AMPI.h*, содержащий измененные *MPI* функции и служащий для получения данных, необходимых для работы приложения.
2. Приложение, анализирующее правильность использования функций *MPI* в проверяемой работе.

Заголовочный файл *AMPI.h* пользователь подключает в проверяемую программу вместо заголовочного файла *MPI.h*. *AMPI.h* содержит в себе доработанные *MPI* функции, которые, кроме своих стандартных действий, также записывают подставляемые им при вызове параметры в отдельный файл. Программа, являющаяся второй частью работы, анализирует этот файл и выводит на экран сгруппированные *MPI* функции, давая возможность пользователю увидеть, как работает написанная им программа и найти ошибки, если они есть.

**Постановка задачи**

*MPI* - это стандарт на программный инструментарий для обеспечения связи между ветвями параллельного приложения [3,4]. Полная версия интерфейса содержит описание более 125 процедур и функций. Поскольку основой *MPI* является библиотека, то при компиляции программ необходимо подключить соответствующие библиотечные модули и заголовочный файл *MPI.h*. *MPI* поддерживает работу с языками Фортран и Си.

Под пользовательской программой понимается программа, написанная с использованием *MPI* функций.

Предполагается, что пользовательская программа будет работоспособна, но используемые в неё *MPI* функции могут быть использованы некорректно и задача разрабатываемого приложения будет сведена к их обнаружению и анализу. Некорректность их использования может выражаться, например, в неправильной пересылке данных между параллельными ветвями программы, в этом случае программа будет работать, но могут появляться искажения в результатах работы, потеря некоторых данных или может снижаться производительность программы. Чтобы избежать подобных ошибок, необходимо проверить, как используются и взаимодействуют между собой применяемые *MPI* функции. Поиск ошибок в коде программы может быть излишне трудоёмким ввиду его сложности, объема или иных причин. Разрабатываемое приложение как раз и служит для облегчения анализа программ, написанных с использованием *MPI* функций и нахождения в них возможных ошибок.

Для этого необходимо выполнить ряд действий:

1. Получение информации о *MPI* функциях в проверяемой пользовательской программе.
2. Разработка приложения, которое на основе получаемых данных выдаст пользователю информацию о написанных им *MPI* функциях в виде, удобном для анализа.

Приложение будет считывать файл с данными о вызовах *MPI* функций и показывать пользователю информацию о тех из них, которые участвовали в передаче данных внутри проверяемой программы (т.е. кроме функции инициализации, окончания параллельной части программы и некоторых других). Результаты приложение должно выводить на экран, в виде, удобном для анализа.

**Определение MPI функций для проверки**

На данном этапе Приложение будет работать с ограниченным числом MPI функций. Это будут наиболее часто используемые и наиболее актуальные функции: MPI\_Send, MPI\_Comm\_rank, MPI\_Comm\_size, MPI\_Recv, MPI\_Scatter, MPI\_Barrier, MPI\_Gather.

Это функции используемые для задания числа потоков, для передачи данных между потоками программы и для их синхронизации.

1. Определение основных ошибок и параметров для тестирования

Основные ошибки, допускаемые при работе с *MPI* функциями, это ошибки в передачи данных между процессами. Так же встречаются следующие ошибки: процесс был не задействован (не содержит функций пересылки и приема данных), содержит только один вид функций (например, если в процессе будут выведены только *MPI\_Reсv*, это будет означать, что программа ждет пересылки данных от процесса, который ничего не отправляет и данные будут утеряны).

Исходя из этого, в анализируемых функциях будут проверяться именно эти ошибки

1. Разработка программной реализации алгоритма

Исходными данными приложения являются написанные и распределённые по процессам модели параллельной программы функции *MPI* с заданными параметрами.

Результатом работы приложения должны быть визуально представленные результаты действий функций *MPI* по передаче данных в разные этапы «выполнения» модели параллельной программы.

Для выполнения работы необходимо выполнить следующие задачи:

1. Написание заголовочного файла, содержащего библиотеку с измененными *MPI* функциями и подключаемого в проверяемую программу вместо файла *MPI.h* для получения данных, необходимых для дальнейшей работы. Для написания библиотеки используется язык программирования Си и среда разработки *Visual Studio 2010.*
2. Разработка приложения, позволяющего пользователю проанализировать написанные им *MPI* функции из проверяемой программы и обнаружить возможные ошибки. В качестве входных данных, приложение использует файл, полученный в результате выполнения проверяемой программы с подключенным к ней заголовочным файлом *AMPI*.*h*. Функции располагаются в порядке, удобном для их анализа, взаимосвязанные функции располагаются вместе. Также приложение содержит краткую инструкцию. Приложение разрабатывается в среде разработки *Visual Studio 2010* на языке *С#.*

Схема алгоритма анализа пользовательской программы представлена на рисунке 1.



Рис. 1. Схема алгоритма анализа пользовательской программы

В рабочую пользовательскую программу подключается заголовочный файл *AMPI.h*, производящий временную подмену *MPI* функций, после чего она выполняется и в результате подмены функций, создается файл с данными об этих функциях.

Файл с данными поступает на обработку в приложение, которое производит анализ *MPI* функций, информация о которых была получена из файла, группирует их в соответствии с их применением в пользовательской программе и выводит результаты на экран.

Схема алгоритма подмены функций *MPI* представлена на рисунке 2.



Рис. 2. Схема алгоритма подмены функций *MPI*

Заголовочный файл *AMPI.h* библиотеки *AMPI* подключается в проверяемую программу, замещая собой библиотеку *MPI* для временной подмены её функций. Суть ее работы заключается в том, что вызов стандартных *MPI* функций заменяется на вызов специальных *АMPI* функций. Созданные *АMPI* функции в качестве своих параметров, принимают параметры соответствующих им *MPI* функций из проверяемой задачи и выполняют те же задачи. Каждая из *АMPI* функций выполняет два действия: запись в отдельный текстовый файл значения своих параметров и вызов «своей» *MPI* функции.

Таким образом, *AMPI* библиотека выполняет внутри проверяемой программы те же действия, что и *MPI*, но кроме этого создает файл с фактическими параметрами *MPI* функций, используемый в последующем анализе правильности работы и написания этих самых функций.

Схема алгоритма анализа данных и представления результатов показана на рисунке 3.



Рис. 3. Схема алгоритма анализа данных и представления результатов

Приложение считывает данные из файла (файл выбирается пользователем) и собирает однотипные функции (например, *MPI\_Send)* в массивы для их последующей обработки. Полученные массивы анализируются и выводятся на экран в соответствии с их применением в пользовательской программе (например, функции *MPI\_Send* и *MPI\_Reсv* сортируются по процессу который производит передачу данных). Если обнаруживается ошибка, связанная с неправильной пересылкой данных между процессами, на экран выводится соответствующее сообщение.

Схема анализа массивов функций и вывод их на экран, представлена на рисунке 4.



Рис. 4. Схема анализа массивов функций и вывод их на экран

Анализ полученных данных выполняется по нажатию кнопки «Анализ». Массивы, сформированные при загрузке данных, анализируются и последовательно выводятся на экран. Сначала выводятся массивы *MPI\_Comm\_size\_array* и *MPI\_Comm\_rank\_ array* в порядке поступления, т.к. они не влияют непосредственно на работу программы. Далее выводятся элементы массивов *MPI\_Send\_array* и *MPI\_Reсv\_array* в следующем порядке: выводится номер процесса (от 0 до *n*, где *n* определяется из полученной функции *MPI\_Comm\_size*), затем из массива *MPI\_Send\_array* выбираются элементы (*MPI*\_*Send* функции), вызываемые в этом процессе, а затем из массива *MPI*\_*Reсv*\_*array* выбираются элементы (*MPI\_Reсv* функции), которые должны принимать данные от *MPI\_Send*, принадлежащих текущему процессу.

Аналогично обрабатываются массивы *MPI\_Scatter\_array* и *MPI\_Gather\_array*.

В случае, если обнаруживается, что какой-то процесс был не задействован (не содержит функций пересылки и приема данных) или содержит только один вид функций (например, если в процессе будут выведены только *MPI\_Reсv*, это будет означать, что программа ждет пересылки данных от процесса, который ничего не отправляет и данные будут утеряны), то на экран будет выведено сообщение о том, что распараллеливаемая часть программы работает некорректно.

Пользовательский интерфейс представлен на рисунках 5 и 6.

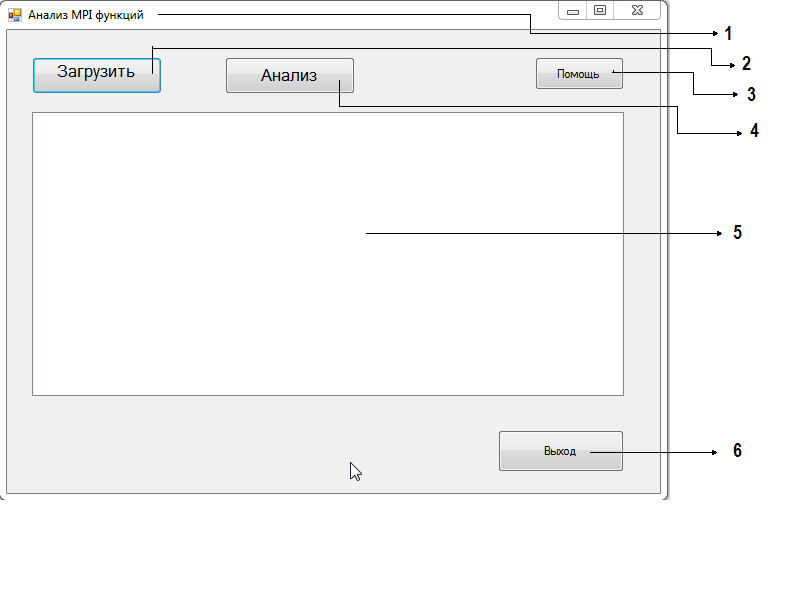


Рис. 5. Пользовательский интерфейс

Пользовательский интерфейс состоит из следующих составных частей:

1. 1 – название приложения.
2. 2 – кнопка для загрузки файла с данными и их вывода на экран. Алгоритм работы представлен на рисунке 7.
3. 3 – кнопка, отображающая инструкцию по использованию приложения. Алгоритм работы представлен на рисунке 8.
4. 4 – кнопка для запуска анализа данных, полученных из загруженного файла. Алгоритм работы представлен на рисунке 10.
5. 5 – Текстовое поле, в которое выводится вся информация, предоставляемая пользователю.
6. 6 – кнопка для закрытия приложения.

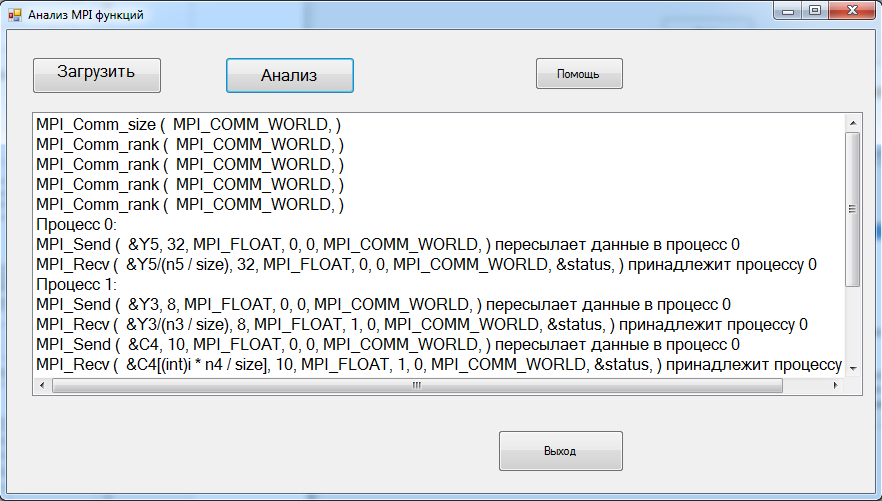


Рис. 6. Пользовательский интерфейс, демонстрирующий работу приложения, после получения данных и нажатия кнопки «Анализ»

**Финальное тестирование на тестовой программе**

Для выявления функциональной пригодности и надежности приложения было проведено несколько тестов. Данные тесты проверяют корректность работы приложения в различных ситуациях:

1. В случае, когда пользовательская программа полностью корректна и ошибок в использовании *MPI* функций нет. Пример результата тестирования представлен на рисунке 11 и 12.
2. В случае, когда пользовательская программа содержит ошибки в использовании *MPI* функций (например, одни и те же данные посылаются дважды). Пример результата тестирования представлен на рисунке 13 и 14.
3. В случае, когда пользователь не загрузил данные и сразу нажал на кнопку «Анализ». Пример результата тестирования представлен на рисунке 15.

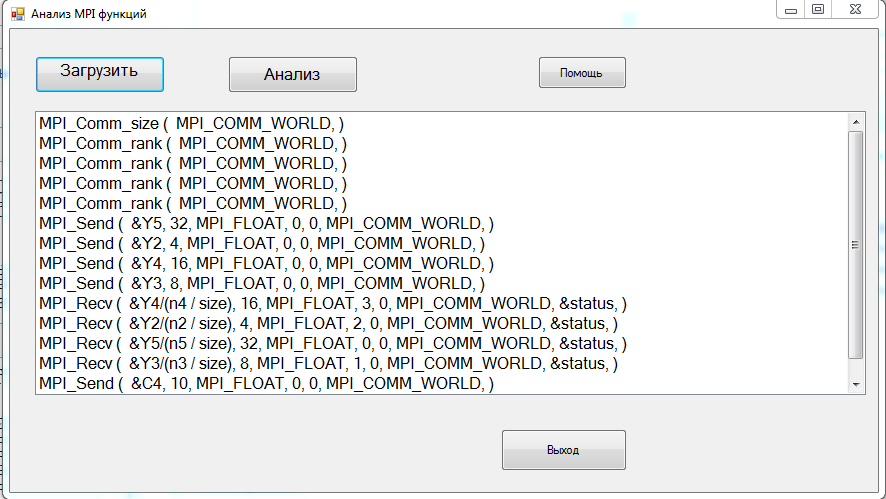


Рис. 11. Проверка корректности работы приложения при нажатии кнопки «Загрузить»

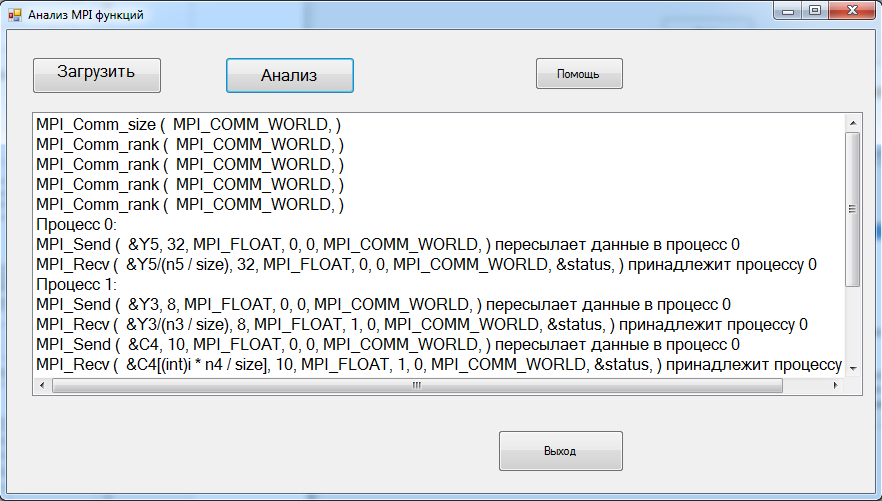


Рис. 12. Проверка корректности работы приложения при нажатии кнопки «Анализ»

Из результата данного теста можно сделать вывод, что приложение работает корректно и выводит связанные между собой *MPI* функции последовательно.

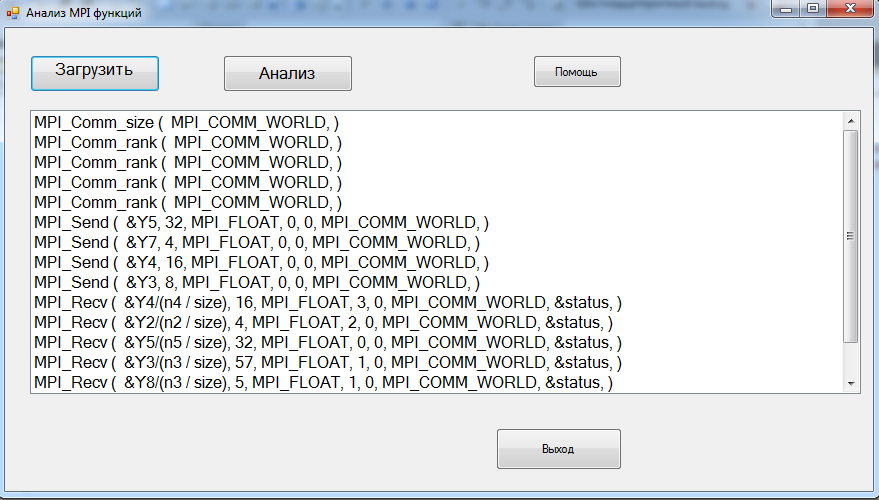


Рис. 13. Проверка корректности работы приложения при нажатии кнопки «Загрузить»

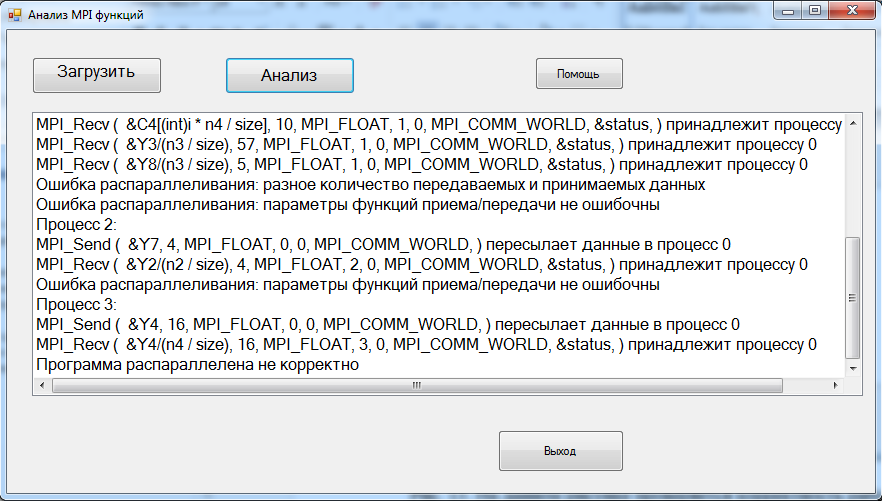


Рис. 14. Проверка корректности работы приложения при нажатии кнопки «Анализ»

Из результата данного теста можно сделать вывод, что приложение работает корректно, сообщая пользователю о том, что распараллеливание происходит неверно (Обнаружено несовпадение в параметрах и ошибки в передаче данных между процессами).

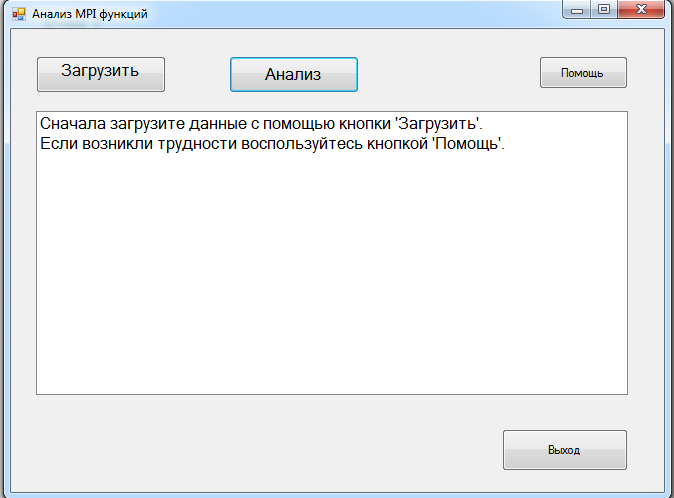


Рис. 15. Проверка корректности работы приложения при нажатии кнопки «Анализ», без предварительной загрузки данных

**Заключение**

Для реализации поставленной задачи было разработано приложение, написанное на языке программирования *С*#, способное анализировать пользовательские программы и проверять корректность используемых в них *MPI* функций. Оно имеет простой интерфейс и встроенную инструкцию. Разработан заголовочный файл *AMPI.h,* написанный на языке программирования Си и необходимый для получения данных о пользовательской программе.

**Литература**

1. Антонов А.С. Параллельное программирование с использованием технологии *MPI*: Учебное пособие. -М.: Изд-во МГУ, 2004.-71 с.
2. *Microsoft Visual Studio*<http://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:Microsoft_Visual_Studio>
3. М. Э. Абрамян. Библиотека *MPI* 1.1

<http://edu.mmcs.sfedu.ru/pluginfile.php/3285/mod_resource/content/1/MPILibrary.pdf>

1. Кластеры на многоядерных процессорах: Учебное пособие / И.И. Ладыгин, А.В. Логинов, А.В. Филатов, С.Г. Яньков; под ред. И.И. Ладыгина. – М.: Издательский дом МЭИ, 2008. – 112 с.
2. Филатов А.В. Параллельное программирование вычислительных систем. Лабораторные работы №1-4: методическое пособие / А.В. Филатов. – М.: Издательство МЭИ, 2015. – 54 с.