Технологии параллельных систем и распределенных вычислений

Лабораторная работа №9. Введение в MPI

Message Passing Interface (MPI, интерфейс передачи сообщений) — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.

MPI является наиболее распространенным стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу.

Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Java, Си и Си++.

В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP ориентирован на системы с общей памятью (многоядерные с общим кешем). Обе технологии могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

Установка необходимых программных средств.

В данном лабораторном практикуме предполагается выполнение работ в операционной системе (ОС) Linux. Одним из распространенных дистрибутивов является Ubuntu Linux. На момент написания этого текста актуальна версия 14.04. Скачать дистрибутив можно с сайта http://www.ubuntu.com/. Установить дистрибутив возможно как на обычный компьютер, так и в виртуальной машине. Для простоты и удобства рекомендуется выполнить установку в виртуальной машине, например VirtualBox (актуальная версия 4.3.14, http://www.virtualbox.org/). Для упрощения процесса установки и сокращения временных затрат можно воспользоваться готовыми образами операционных систем, их можно найти на сайте http://virtualboximages.com/. После скачивания необходимого образа запускаем виртуальную машину → создать → имя: произвольное; тип: Linux; версия: Ubuntu → Рекомендованный объем памяти 512 Мб → использовать существующий жесткий диск; выбрать скачанный образ → создать. Запустить ОС. В случае использования готового образа с сайта virtualboximages.com, логин/пароль для в хода в систему adminuser/adminuser соответственно.

После запуска операционной системы необходимо установить среду разработки Code::Blocks (версия 13.12, http://www.codeblocks.org/). Для этого в терминале необходимо выполнить команду:

```
sudo apt-get update
sudo apt-get install codeblocks
```

А так же установить библиотеку mpich2, которая является реализацией стандарта MPI:

```
sudo apt-get install libcr-dev mpich2 mpich2-doc
```

После чего запустить Code::Blocks и создать в нем проект консольного приложения File→ New → Project → Console Application и ответить на вопросы диалога о названии проекта, его месторасположении и т.д.

После чего необходимо выполнить настройку среды разработки: Settings → Enviroment → Terminal to launch console programms: xterm -T $TITLE -e mpirun -np 4. Здесь 4 означает число процессов в процессе работы программы.

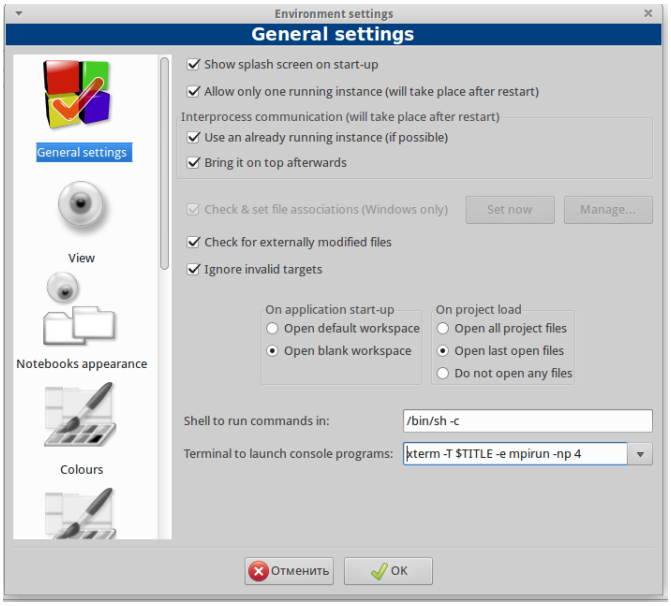

Рис. 1 — Окно настроек среды разработки

Подробнее о настройке **Visual Studio** для работы с MPI смотрите ссылки:
http://iproc.ru/programming/mpich-windows/
http://iproc.ru/drafts/microsoft-mpi/

Простейшая программа, которая выводит «Hello world» от имени каждого процесса, показана в листинге. Результат работы программы показан на рис. 2.

```
#include <mpi/mpi.h> /* MPI library */
#include <stdio.h>

int main (int argc, char* argv[])
{
  int rank, size;

  MPI_Init (&argc, &argv); // starts MPI
  MPI_Comm_rank (MPI_COMM_WORLD, &rank); // get current process id
  MPI_Comm_size (MPI_COMM_WORLD, &size); // get number of //processes
  printf( "Hello world from process %d of %d\n", rank, size );
  MPI_Finalize();
  return 0;
}
```

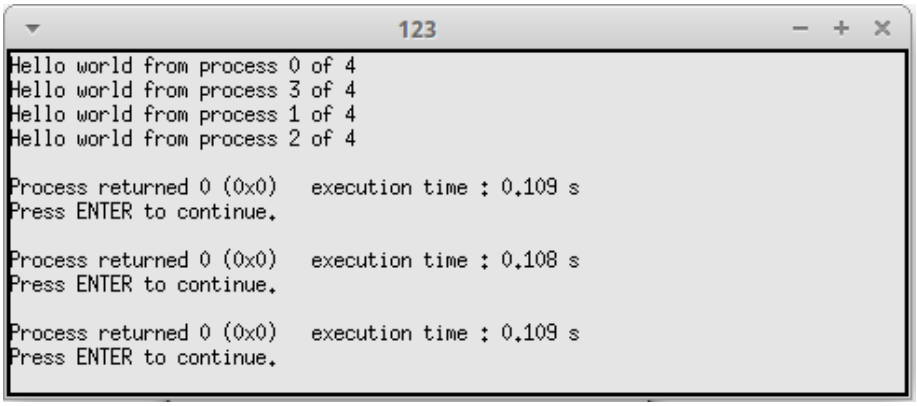

Рис. 2 — Пример работы программы

Задание. Набрать программу, которая вычисляет скалярное произведение двух массивов с помощью библиотеки MPI. Протестировать работу программы и построить график времени работы программы в зависимости от числа потоков.