# MPI. Теория

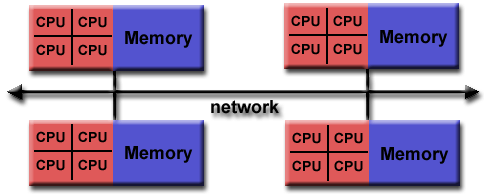
## Определение MPI

**Message Passing Interface** (MPI, интерфейс передачи сообщений) — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком (англ.) и другими.

MPI является наиболее распространённым 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.



Хотя программы, использующие MPI, разрабатываются как использующие только распределённую память, конкретные реализации от производителей могут (и это приветствуется Форумом) использовать особенности конкретных устройств для повышения производительности.

## Основания для использования MPI в области распределения вычислений

**Стандартизация.** MPI – это единственная библиотека обмена сообщениями, которую можно считать стандартом в области. Она поддерживается практически на всех высокопроизводительных платформах. Все предшествующие подобные библиотеки были вытеснены этой.

**Портабельность (переносимость).** Следует из предыдущего. Редактировать код придётся совсем немного или совсем нисколько в случае необходимости поддержания или перехода приложения на другую платформу, поддерживающую MPI.

**Перспектива быстродействия**. Имплементируя библиотеку на своём устройстве, производители могут оптимизировать исполнение на уровне «железа».

**Функциональность**. Библиотека MPI-3 уже включает более 430 примитивов, включая большинство содержавшихся в MPI-1 и MPI-2.

**Доступность**. Существует множество реализаций и от сообщества, и от производителей.

## Документация

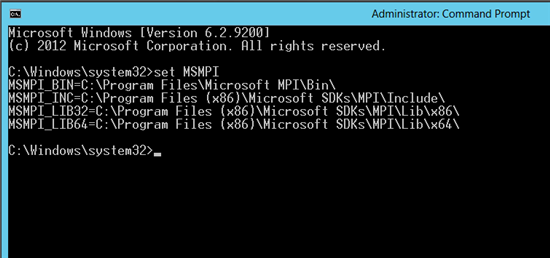
По MPI доступна в сети и вместе с данными методическими указаниями. Файл mpi31-report.pdf. На момент составления указаний документация распространялась через сайт [www.mpi-forum.org](http://www.mpi-forum.org), находилась по адресу [www.mpi-forum.org/docs/](http://www.mpi-forum.org/docs/).

# MPI. Установка и запуск

## Установка HPC Pack 2008 SDK

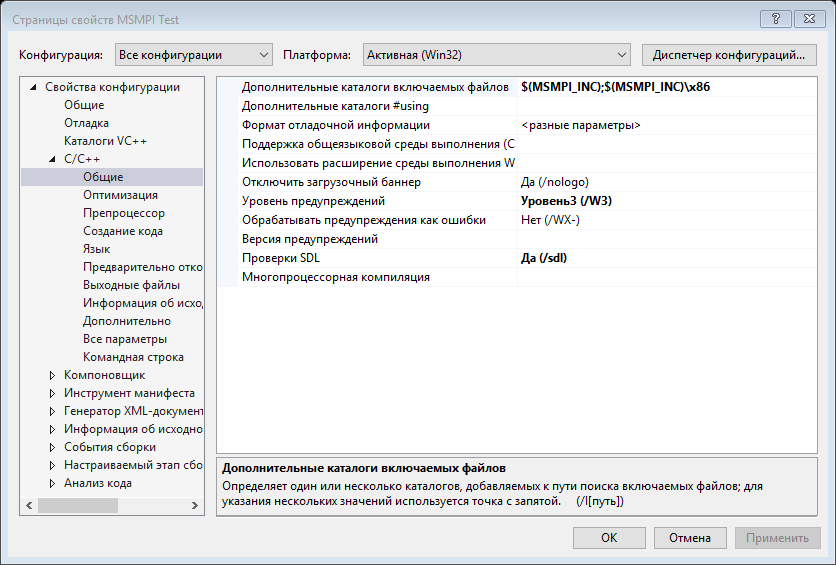
Установка HPC Pack SDK похожа на установку других программных продуктов Microsoft:

1. скачайте 32-х битную или 64-х битную версию, в соответствии с версией вашей операционной системы;
2. запустите скачанный инсталлятор;
3. примите лицензионное соглашение;
4. нажимайте кнопку «Next» до тех пор, пока она не превратится в кнопку «Finish»; На этом настройка клиентов заканчивается, для запуска ожидания команды на вычисление используется программа smpd с параметром -d. Помните, что на всех используемых в кластере машинах должно использоваться одно имя пользователя с одинаковым непустым паролем.
5. После установки стоит проверить установились ли переменные среды для пакета. В случае их отсутствия стоит установить их вручную командой set, иначе при настройке среды программирования придётся использовать абсолютные пути.

[](http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-68-36/step2.png)

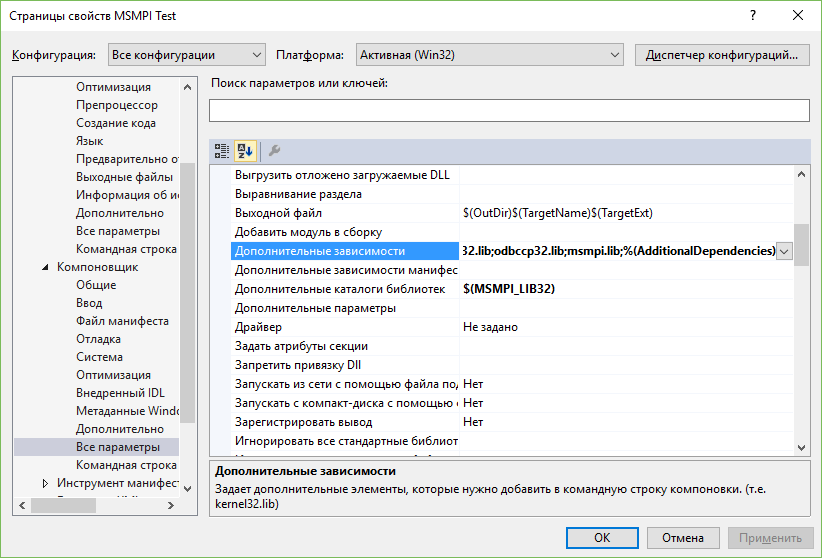
## Настройка среды программирования

1. Откройте Visual Studio и создайте новый проект консольного приложения Visual C++ Win32 со стандартными настройками.
2. В свойствах проекта -> С/С++ настройте дополнительные директории включаемых файлов (additional include directories) как $(MSMPI\_INC);$(MSMPI\_INC)\x86 для 32 битного приложения (или $(MSMPI\_INC);$(MSMPI\_INC)\x64 для 64х битного)



Не забудьте переключить редактируемую конфигурацию на все, дабы не иметь проблем при переключении между релизной и отладочной.

1. Настройте компоновщик(linker) для использования библиотеки MS-MPI. Для этого добавьте в дополнительные каталоги библиотек (Свойства->компоновщик->общие) $(MSMPI\_LIB32) (или $(MSMPI\_LIB64) для 64х битного приложения) и добавьте msmpi.lib в дополнительные зависимости (Свойства->компоновщик->ввод)



1. Проект готов для использования. Можете протестировать его при помощи простенького хеллоу ворлда, заменив код main.cpp на следующий

#include "stdafx.h"

#include <mpi.h>

#include <iostream>

using namespace std;

int main(int argc, char \*\*argv)

{

MPI\_Init(&argc, &argv);

int rank, size;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

if (rank)

{

char buf[] = "Hello!";

MPI\_Send(buf, sizeof(buf), MPI\_CHAR, 0, 0, MPI\_COMM\_WORLD);

}

else {

cout << "Process 0 started" << endl;

for (int i(1); i<size; ++i)

{

MPI\_Status s;

MPI\_Probe(MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &s);

int count;

MPI\_Get\_count(&s, MPI\_CHAR, &count);

char \*buf = new char[count];

MPI\_Recv(buf, count, MPI\_CHAR,

MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &s);

cout << "Message from process " << s.MPI\_SOURCE << ": "

<< buf << endl;

delete[] buf;

}

cout << "Done." << endl;

}

if (rank == 0)

{

getchar();

}

MPI\_Finalize();

return 0;

}

## Запуск программы в параллельном режиме

Для запуска MPI-программ в комплекте MS MPI идут программы mpiexec.exe и smpd.exe, путь к которым при установке HPC Pack SDK добавился в переменную окружения PATH. Это означает, что вы можете вызывать mpiexec.exe из командной строки, находясь в любом месте файловой системы.

Для запуска нашей программы на одном компьютере достаточно только программы mpiexec.exe. Она запустит требуемое число экземпляров MPI-программы и снабдит их информацией для связи друг с другом. Запустите командную строку и перейдите в папку с исполняемым файлом вашей программы с помощью команды cd, затем запустите программу с помощью mpiexec:

mpiexec %your\_program\_name% -n %number\_of\_processes%

Где %your\_program\_name% - имя исполняемого файла вашей программы,   
%number\_of\_processes% - число процессов, которые следует запустить на локальной машине.

Для запуска на множестве машин необходимо предоставить им всем доступ к исполняемому файлу вашей программы либо скопировав его на каждую используемую машину (что не слишком удобно при использовании большого количества машин), либо использовать сетевую папку. Кроме того, на всех ЭВМ командой

smpd -d

должен быть запущен менеджер процессов.

Можно использовать следующую команду запуска:

mpiexec -hosts 2 hostA 1 hostB 1 -wdir \\hostA\c$\SomeDirectory MPIHelloWorld.exe

Где параметром -hosts указывается число машин, за которым следуют имена хостов с числом процессов для запуска, параметром -wdir – рабочая папка, в данном случае сетевая, и последним параметром без ключа указывается имя исполняемого файла. Для запуска на множестве машин также можно использовать параметр -machinefile с указанием текстового файла в формате

hostname %number\_of\_processes%

по одной записи на строку.