# Параллельное программирование

MPI

# MPI (Message Passing Interface)

- Наиболее распространенный интерфейс обмена данными в параллельных системах
- Основное использование кластеры и суперкомпьютеры
- Основан на идеологии обмена сообщениями между параллельным  процессами
- Имеет большое количество независимых реализаций
- http://www.mpi-forum.org/

# История

- 1992-93 Начало разработки
- 1994 - Спецификаций MPI 1
- 1995 - MPI 1.1 (первая реализация 2002)
- 1997 - MPI 2.0
- 2008 - MPI 2.1
- 2009 - MPI 2.2
- 2012 - MPI 3.0

# Реализации

- MPICH/MPICH-2
- OpenMPI
- IntelMPI
- MS-MPI
- Oracle HPC cluster
- Boost.MPI

# Модель программирования

- Single Program Multiply Data
- N параллельных процессов.
- Количество процессов определяется в момент запуска (MPI 2 могут добавляться динамически)
- Все процессы могут выполняться на разных процессорах
- Каждый процесс имеет свое адресное пространство

# Операции передачи данных

Основа MPI это передача сообщений

- point-to-point (парные)
- collective (несколько процессов)

# Коммуникатор

- Процессы объединяются в группы
- Коммуникатор - специальный объект, объединяющий группу и контекст.
- Необходим для передачи сообщений
- MPI_COMM_WORLD
  - включает все процессы
  - коммуникатор по-умолчанию

# MS - MPI

Имлементация от Microsoft
#include <mpi.h>
msmpi.lib

# Структура параллельноей программы на MPI

```
#include <mpi.h>

int _tmain(int argc, _TCHAR* argv[])
{
    /// программный код без использования MPI
    MPI_Init(&argc, &argv);
    /// программыний код с использованием MPI
    MPI_Finalize();
    /// программный код без использования MPI
    return 0;
}
```

# Hello World MPI

```
int _tmain(int argc, _TCHAR* argv[])
{

    int rank, size, len;
    char host[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_processor_name(host, &len);

    printf("Hello world!. I'a %d of %d on %s\n", rank, size, host);

    MPI_Finalize();

    return 0;
}
```

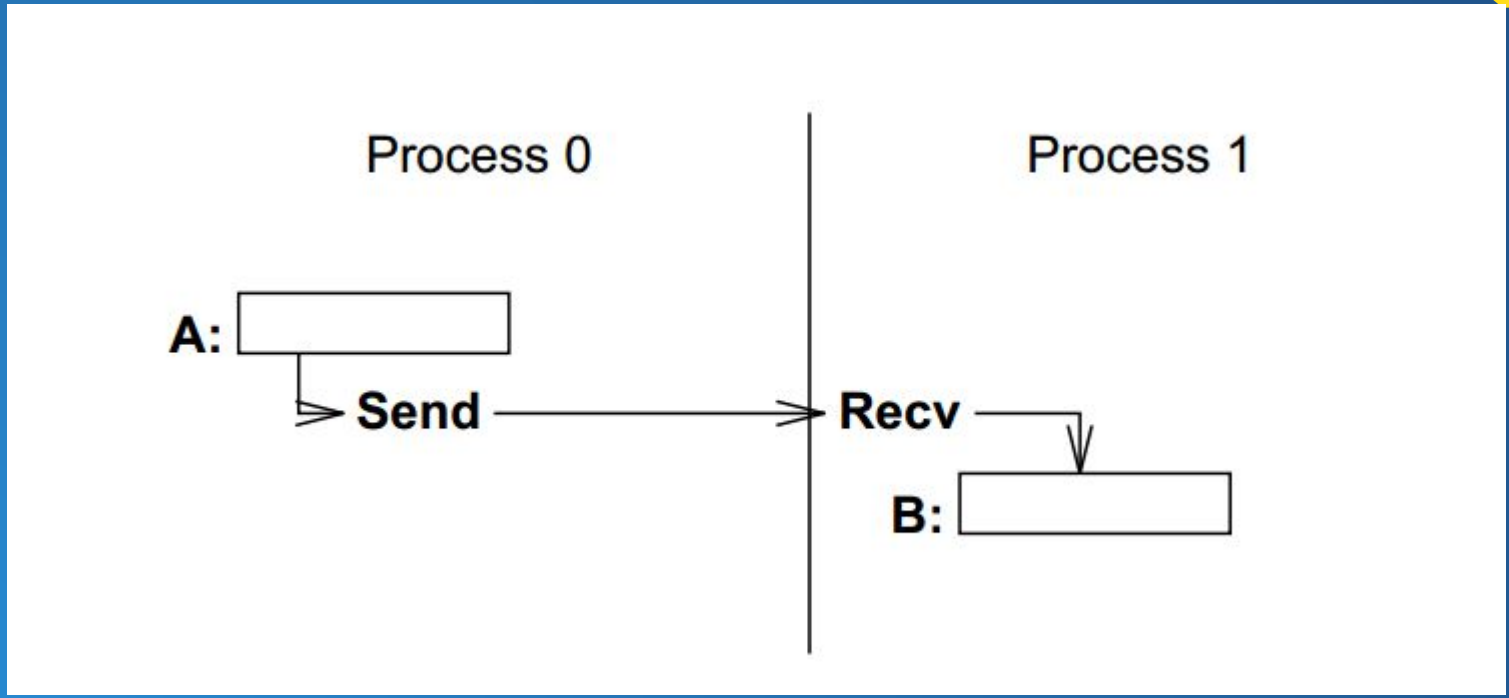
# Запуск с MS-MPI

- CMD (PowerShell)
  - mpiexec -n process_count HelloWorld.exe
- VisualStudio
  - Tools -> mpiexec (требуте добавления и настройки)
- В других имплементациях - по-другому

# Hello world!

```
PS D:\Work\Itmo\Paralle\MPI\Debug> mpiexec -n 4 .\HelloWorld.exe
Hello world. I'a 2 of 4 on alexandr-pc
Hello world. I'a 3 of 4 on alexandr-pc
Hello world. I'a 1 of 4 on alexandr-pc
Hello world. I'a 0 of 4 on alexandr-pc
```

# Передача сообщений



# Структура сообщения

- Заголовок
  - Идентификаторы отправителя и получателя
  - Тег сообщения
  - Коммуникатор

- Данные
  - Буфер
  - Число элементов в буфере
  - Тип одного элемента

# Посылка сообщения

int MPI_Send ( void *buf, int count, MPI_Datatype type, int dest,
int tag, MPI_Comm comm )

buf - адрес буфера памяти
count - количество элементов ддных
datatype - тип данных
dest - идентификатор получателя
tag - тег сообщения или MPI_ANY_TAG
comm коммуникатор или MPI_COMM_WORLD

# Прием сообщения

int MPI_Recv ( void *buf, int count, MPI_Datatype datatype, int source,
int tag, MPI_Comm comm , MPI_Status * status )

buf - адрес буфера памяти
count - количество элементов данных
datatype - тип данных
source - идентификатор отправителя или MPI_ANY_SOURCE
tag - тег сообщения или MPI_ANY_TAG
comm коммуникатор или MPI_COMM_WORLD
status - указатель на структуру с результатом выполнения операции приема

# Типы данных

| MPI datatype | C datatype | C++ datatype |
|---|---|---|
| MPI::CHAR | char | char |
| MPI::WCHAR | wchar_t | wchar_t |
| MPI::SHORT | signed short | signed short |
| MPI::INT | signed int | signed int |
| MPI::LONG | signed long | signed long |
| MPI::SIGNED_CHAR | signed char | signed char |
| MPI::UNSIGNED_CHAR | unsigned char | unsigned char |
| MPI::UNSIGNED_SHORT | unsigned short | unsigned short |
| MPI::UNSIGNED | unsigned int | unsigned int |
| MPI::UNSIGNED_LONG | unsigned long | unsigned long int |
| MPI::FLOAT | float | float |
| MPI::DOUBLE | double | double |
| MPI::LONG_DOUBLE | long double | long double |
| MPI::BOOL | | bool |
| MPI::COMPLEX | | Complex<float> |
| MPI::DOUBLE_COMPLEX | | Complex<double> |
| MPI::LONG_DOUBLE_COMPLEX | | Complex<long double> |
| MPI::BYTE | | |
| MPI::PACKED | | |

C++ names for the MPI C and C++ predefined datatypes, and their corresponding C/C++ datatypes.

# Hello world 2!

```
int _tmain(int argc, _TCHAR* argv[])
{

    int rank, size, len;
    char host[MPI_MAX_PROCESSOR_NAME];
    int recvRank;
    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_processor_name(host, &len);
    if (rank == 0){
        int i;
        MPI_Status status;

        printf("Hello world!. I'a %d of %d on %s\n", rank, size, host);
        for (int i = 1; i < size ; i++){
            MPI_Recv(&recvRank, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
            printf("Message from %d: \n", recvRank);
        }
    }
    else{
        MPI_Send(&rank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

# Функции MPI

- Определены в mpi.h
- Имеют префикс MPI_
- Возвращают код выполнения операции
  - MPI_SUCCESS или код ошибки
  - кроме MPI_Wtime и MPI_Wtick
- Другие результаты через аргументы

# Время

double MPI_Wtime(void)

текущее время

```
double t1, t2, dt;
t1 = MPI_Wtime();
t2 = MPI_Wtime();
dt = t2 - t1;
```

double MPI_Wtick(void)

# Информирование о сообщениях

int MPI_Probe ( int source, int tag, MPI_Comm comm, MPI_Status * status )

Получение информации о структуре ожидаемого сообщения

---

int MPI_Get_count ( MPI_Status *status, MPI_Datatype datatype, int * count )

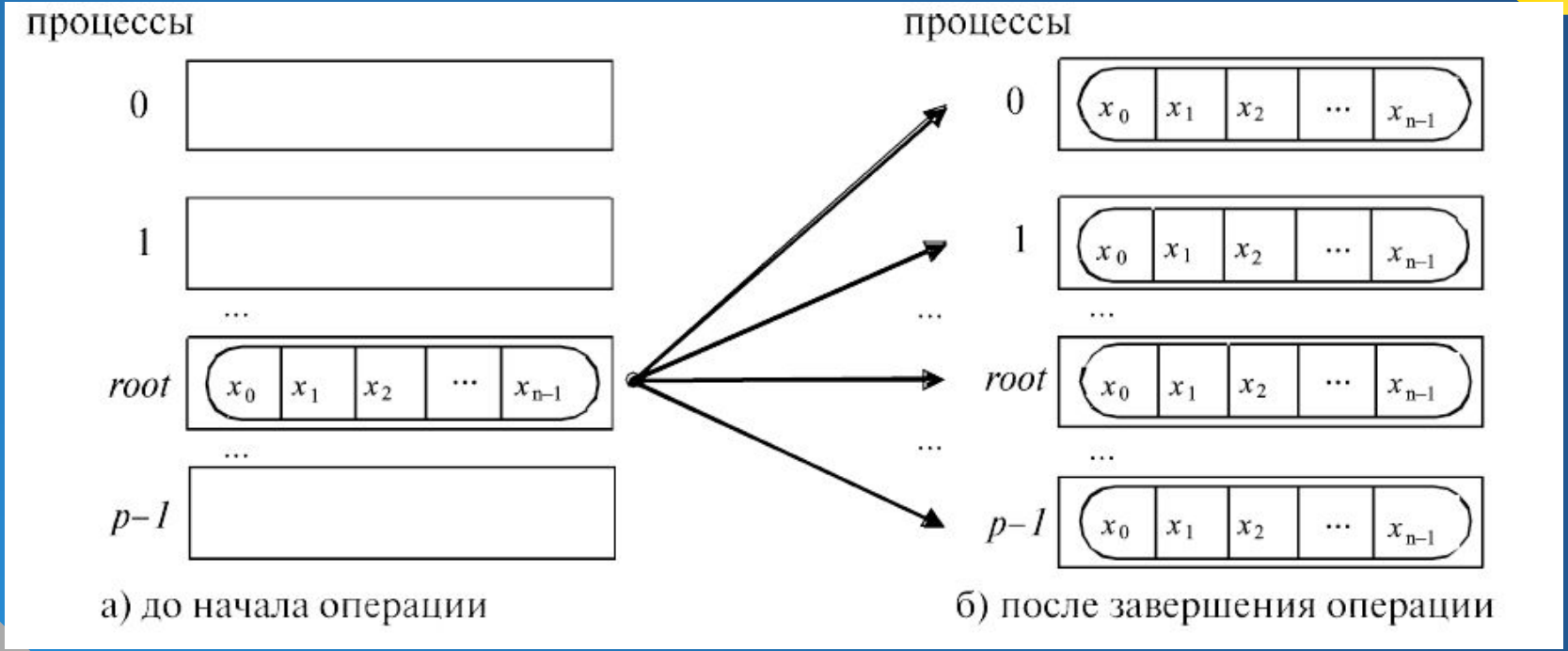
Записывате в count число принятых (посл MPI_Rcv) или принимаемых (после MPI_Probe) сообщений сообщения

# Broadcast

```
int MPI_Bcast(void *buf, int count, MPI_Datatype type,
                                    int  root, MPI_Comm comm)
```

buf, count, type - буффер с сообщением
root - идентификатор отправителя
comm - комуникатор

# Broadcast


процессы
0
1
...
root
$x_0$ $x_1$ $x_2$ ... $x_{n-1}$
...
p–1
а) до начала операции
процессы
0
$x_0$ $x_1$ $x_2$ ... $x_{n-1}$
1
$x_0$ $x_1$ $x_2$ ... $x_{n-1}$
...
root
$x_0$ $x_1$ $x_2$ ... $x_{n-1}$
...
p–1
$x_0$ $x_1$ $x_2$ ... $x_{n-1}$
б) после завершения операции

# Пример.

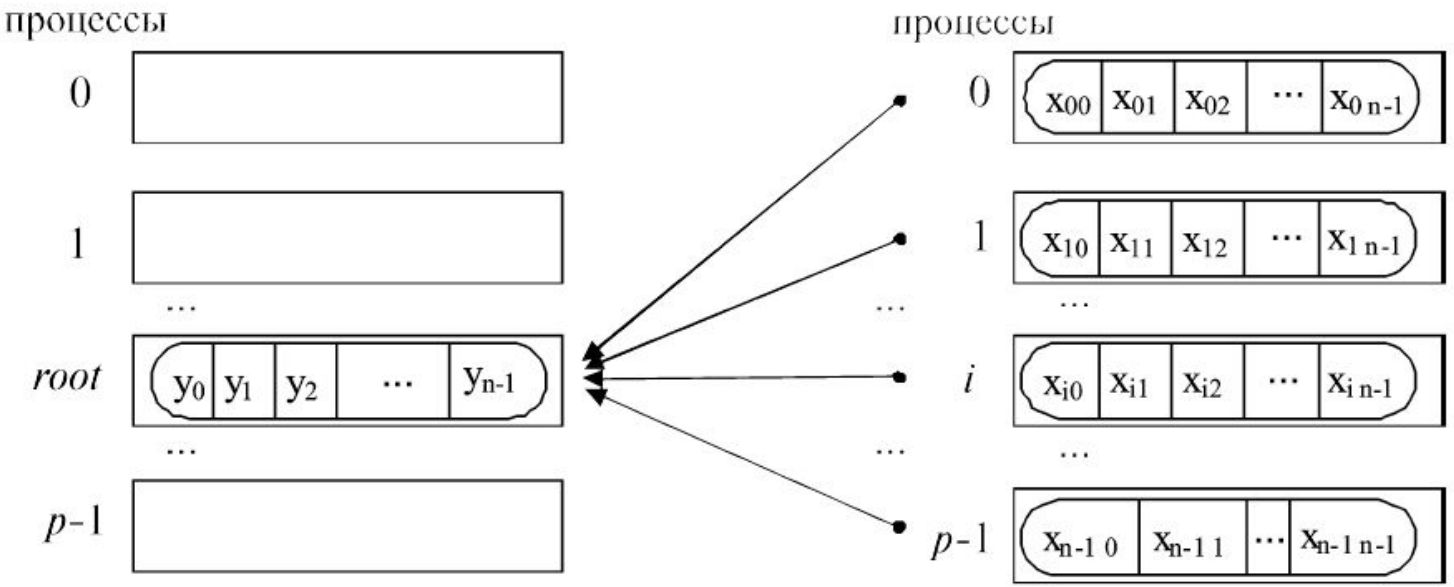
```
MPI_Bcast(x, N, MPI_DOUBLE, 0, MPI_COMM_WORLD);

int step = N / size;
int a = step*rank;
int b = step*(rank + 1);

for (int i = a; i < b; ++i)
    ProcSum += x[i];

if (rank == 0){
    MPI_Status status;
    totalSumm = ProcSum;
    for (int i = 1; i < size; ++i){
        MPI_Recv(&ProcSum, 1, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
        totalSumm += ProcSum;
    }
}
else{
    MPI_Send(&ProcSum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
}
```

# Редукция

int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype type, MPI_Op op, int root, MPI_Comm comm)

sendbuf - буффер отпраляемого собщения
recvbuf - буффер принимамого сообщеия
op - требуемая операция над данными (может быть разной )
root - идентификатор процесса на котором требуется собрать результат

# Редукция




а) после завершения операции

б) до начала операции

# Операции редукции

| Операция | Описание |
|---|---|
| MPI_MAX | Определение максимального значения |
| MPI_MIN | Определение минимального значения |
| MPI_SUM | Определение суммы значений |
| MPI_PROD | Определение произведения значений |
| MPI_LAND | Выполнение логической операции «И» над значениями сообщений |
| MPI_BAND | Выполнение битовой операции «И» над значениями сообщений |
| MPI_LOR | Выполнение логической операции «ИЛИ» над значениями сообщений |
| MPI_BOR | Выполнение битовой операции «ИЛИ» над значениями сообщений |
| MPI_LXOR | Выполнение логической операции исключающего «ИЛИ» над значениями сообщений |
| MPI_BXOR | Выполнение битовой операции исключающего «ИЛИ» над значениями сообщений |
| MPI_MAXLOC | Определение максимальных значений и их индексов |
| MPI_MINLOC | Определение минимальных значений и их индексов |

# Пример

```
MPI_Bcast(x, N, MPI_DOUBLE, 0, MPI_COMM_WORLD);

int step = N / size;
int a = step*rank;
int b = step*(rank + 1);

for (int i = a; i < b; ++i)
    ProcSum += x[i];

MPI_Reduce(&ProcSum, &totalSumm, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

# Синхронизация

int MPI_Barrier ( MPI_Comm comm )

Блокирует работу процессов, вызвавших данную функцию, до того момента как все процессы группы тоже ее не выполнят.

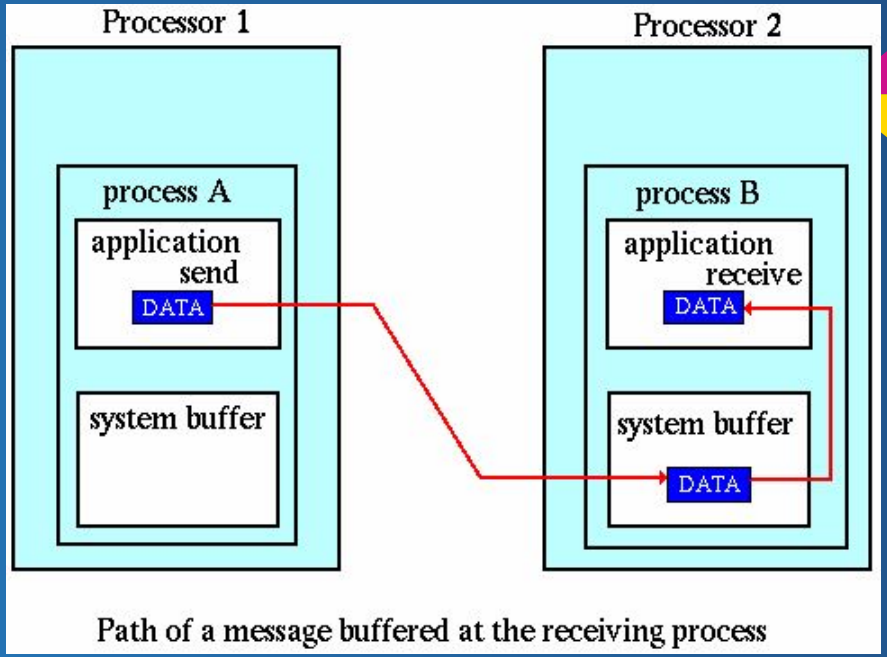
# Аварийное завершение

int MPI_Abort(MPI_Comm comm, int errorcode)

Завершает работу всех процессов группы

# Парный обмен

- MPI_Send
  - вызывающий процесс блокируется
  - после завершения буфер можно использовать
  - состояние сообщения может быть различным

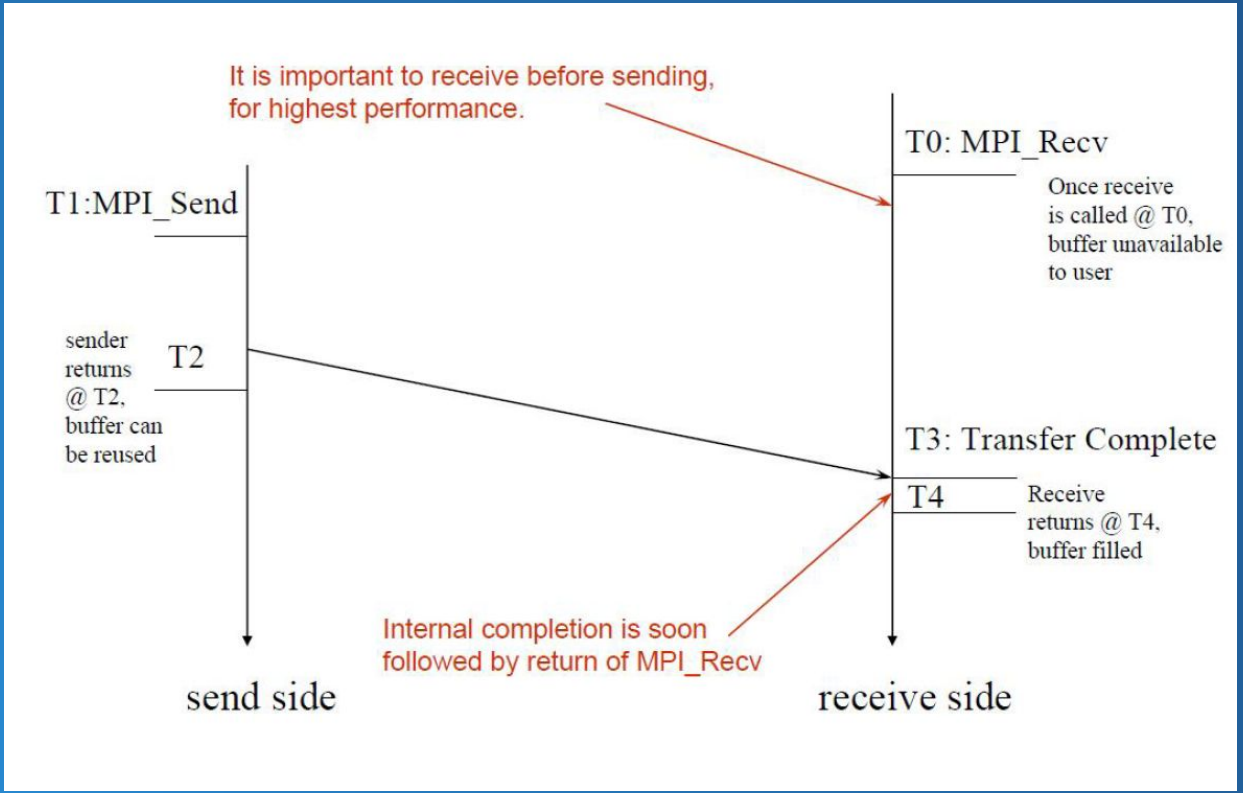


# Блокирующий обмен

Способы передачи

- стандартный (MPI_Send)
- синхронный (MPI_SSend)
- буферизированный (MPS_BSend)
- режим передачи по готовности (MPS_RSend)

# Блокирующий обмен



# Неблокирующий обмен

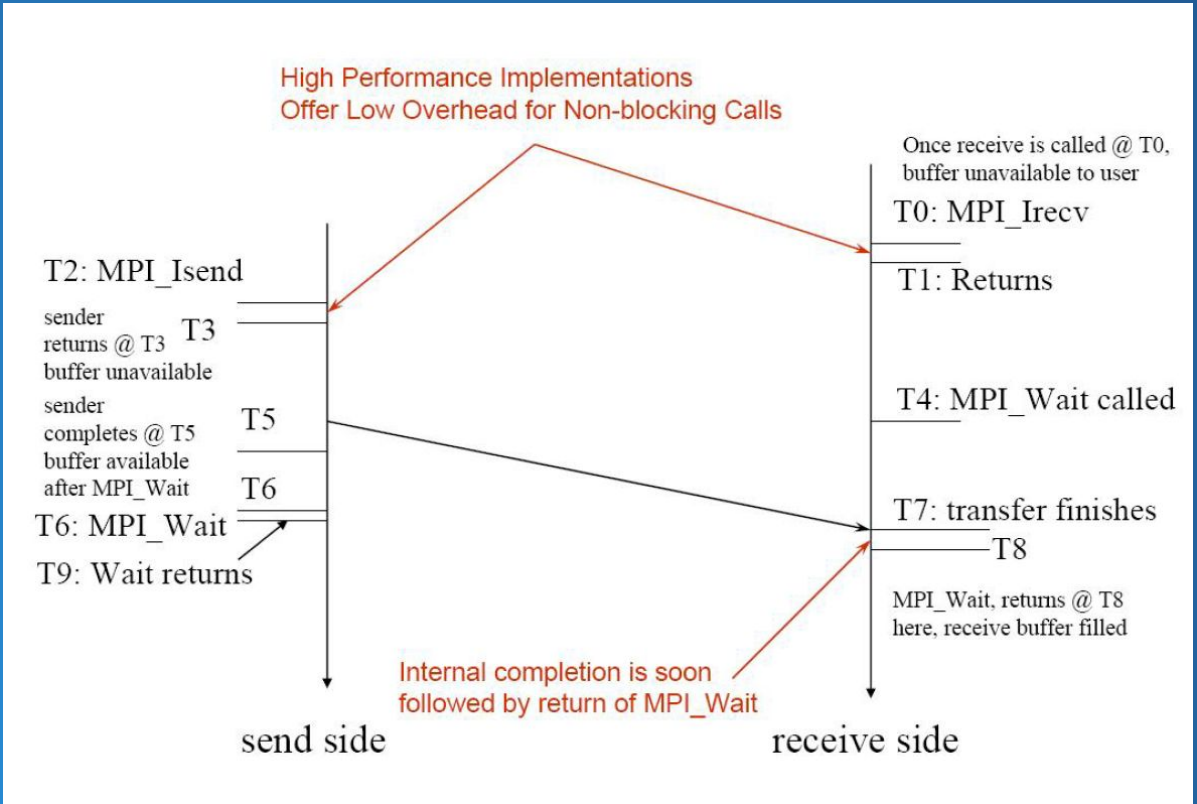
- int MPI_Isend(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm, MPI_Request *request)
- int MPI_Issend(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm, MPI_Request *request)
- int MPI_Ibsend(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm, MPI_Request *request)
- int MPI_Irsend(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm, MPI_Request *request)
- int MPI_Irecv(void *buf, int count, MPI_Datatype type, int source, int tag, MPI_Comm comm, MPI_Request *request).

# Проверка получения

- Неблокирующие
  - int MPI_Test(MPI_Request *request, int *flag, MPI_status *status)
  - MPI_Testall, MPI_Testany,MPI_Testsome

- Блокирующие
  - int MPI_Wait(MPI_Request *request, MPI_status *status)
  - MPI_Waitall, MPI_Waitany, MPI_Waitsome

# Неблокирующий прием



# Неблокирующий прием

```
int prev = (rank + size - 1) % size;
int next = (rank + 1) % size;


MPI_Irecv(&in_message1, 50, MPI_CHARACTER, prev, tag1, MPI_COMM_WORLD, &req[0]);
MPI_Irecv(&in_message2, 50, MPI_CHARACTER, next, tag2, MPI_COMM_WORLD, &req[1]);

_snprintf(out_message1, 50, "Hello, next. I'm %d of %d on %s", rank, size, host);
_snprintf(out_message2, 50, "Hello, prev. I'm %d of %d on %s", rank, size, host);

MPI_Isend(out_message1, 50, MPI_CHARACTER, next, tag1, MPI_COMM_WORLD, &req[2]);
MPI_Isend(out_message2, 50, MPI_CHARACTER, prev, tag2, MPI_COMM_WORLD, &req[3]);

MPI_Waitall(4, req, stats);

printf("%d, Message from %d: '%s'\n", rank, stats[0].MPI_SOURCE, in_message1);
printf("%d, Message from %d: '%s'\n", rank, stats[1].MPI_SOURCE, in_message2);
```

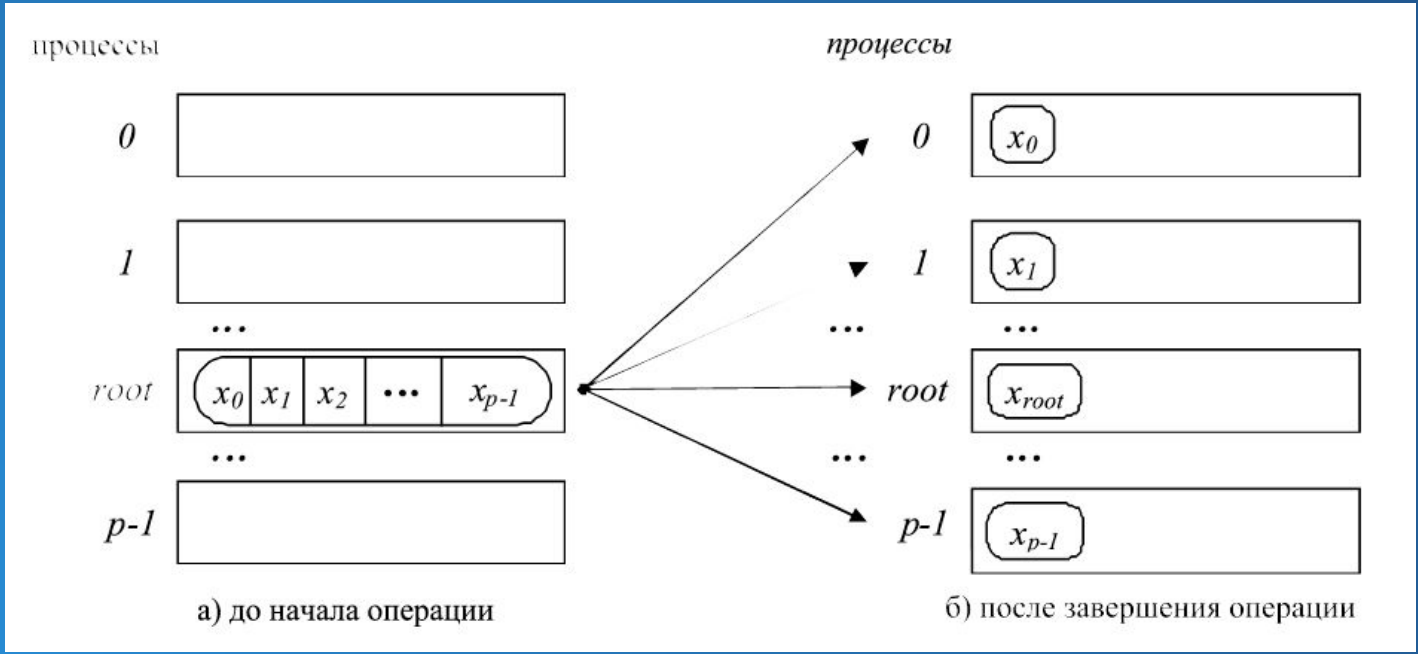
# Приемо-передача

```
int MPI_Sendrecv(void *sbuf,int scount,MPI_Datatype stype, int dest,
                 int stag, void *rbuf,int rcount,MPI_Datatype rtype,
                 int source,int rtag, MPI_Comm comm, MPI_Status *status),
```

```
int MPI_Sendrecv_replace(void *buf, int count, MPI_Datatype type,
                         int dest, int stag, int source, int rtag,
                         MPI_Comm comm, MPI_Status* status),
```

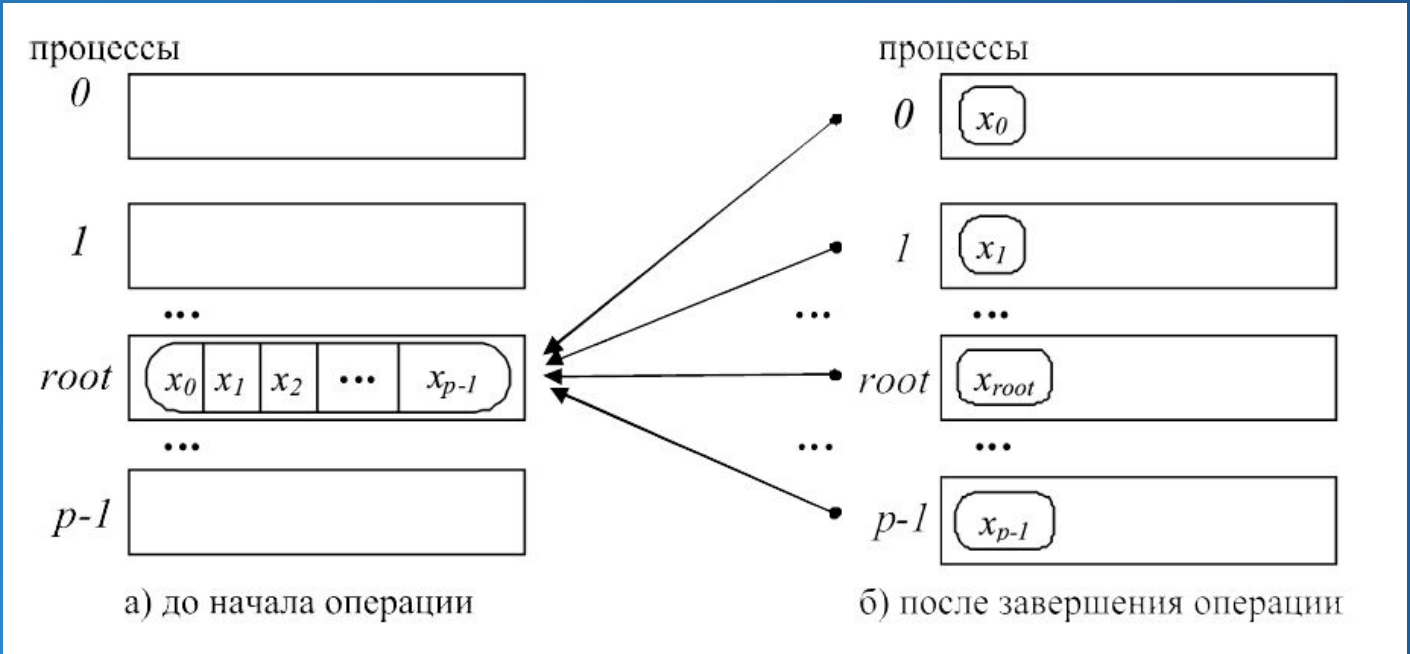
# Обмен данными

int MPI_Scatter(void *sbuf, int scount, MPI_Datatype stype, void *rbuf,
int rcount, MPI_Datatype rtype, int root,MPI_Comm comm)



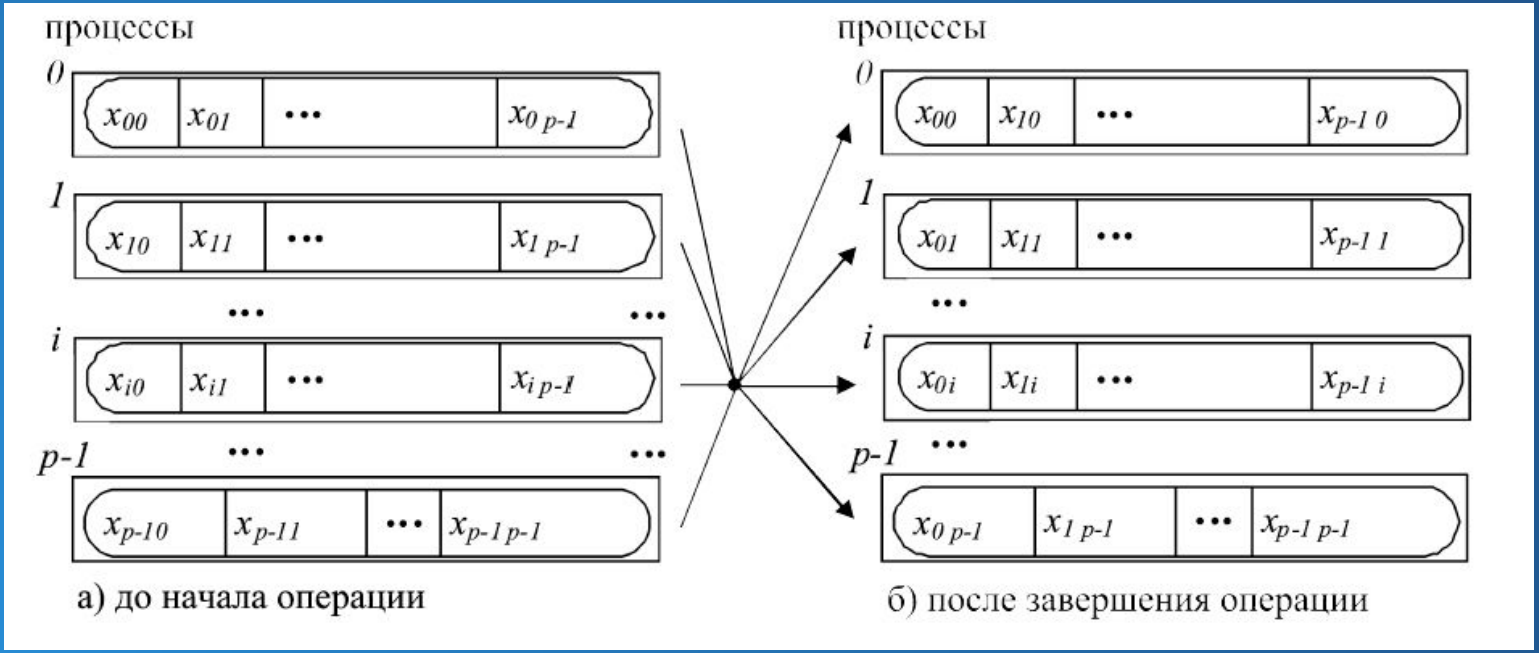
# Обмен данными

int MPI_Gather(void *sbuf, int scount, MPI_Datatype stype, void *rbuf,
int rcount, MPI_Datatype rtype, int root, MPI_Comm comm),



# Обмен данными

int MPI_Alltoall(void *sbuf,int scount,MPI_Datatype stype, void *rbuf,
int rcount,MPI_Datatype rtype,MPI_Comm comm)



# Обмен данными

MPI_Allreduce
MPI_Scan

# Пример с ПИ

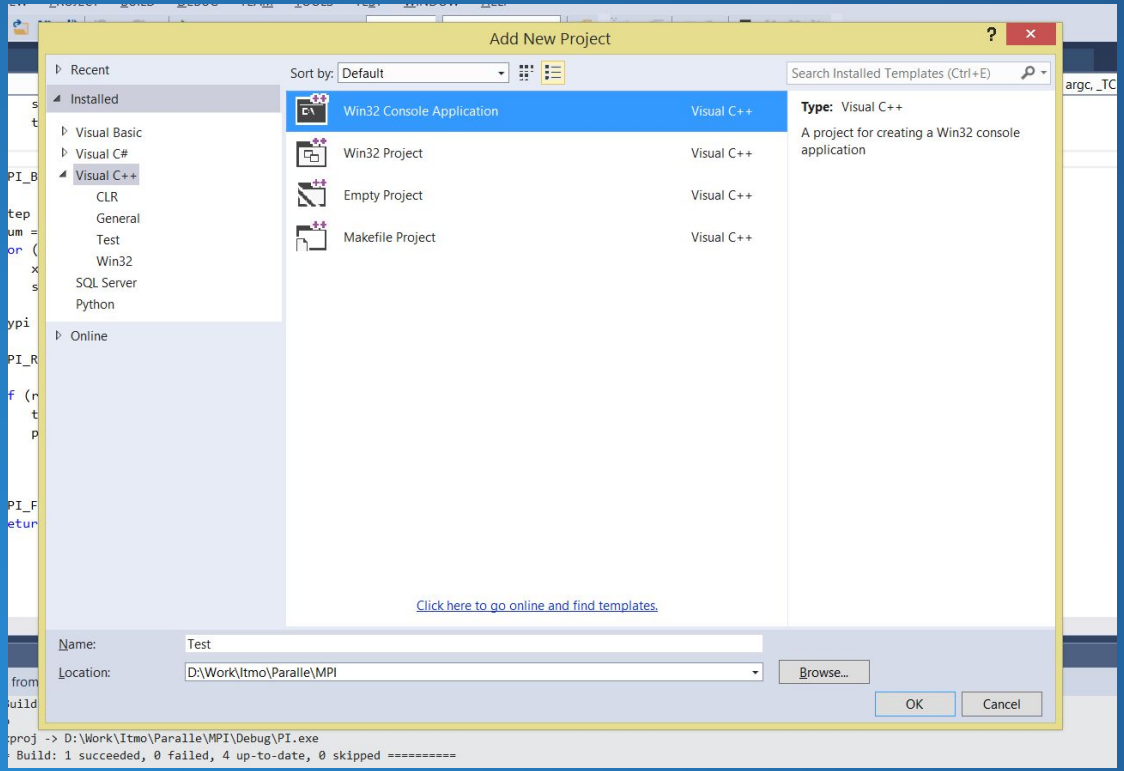
```
if (rank == 0) {
    printf("Enter the number of intervals:");
    fflush(stdout);
    scanf("%d", &n);
    time = MPI_Wtime();
}

MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

h = 1.0 / (double)n;
sum = 0.0;
for (int i = rank+ 1; i <= n; i += size) {
    x = h * ((double)i - 0.5);
    sum += 4.0 / (1.0 + x*x);
}
mypi = h * sum;

MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,   MPI_COMM_WORLD);

if (rank == 0){
    time = MPI_Wtime() - time;
    printf("pi is approximately %.16f, Error is %.16f, Run time is %fs\n",pi, fabs(pi - PI25DT), time);
}
```

# Другое

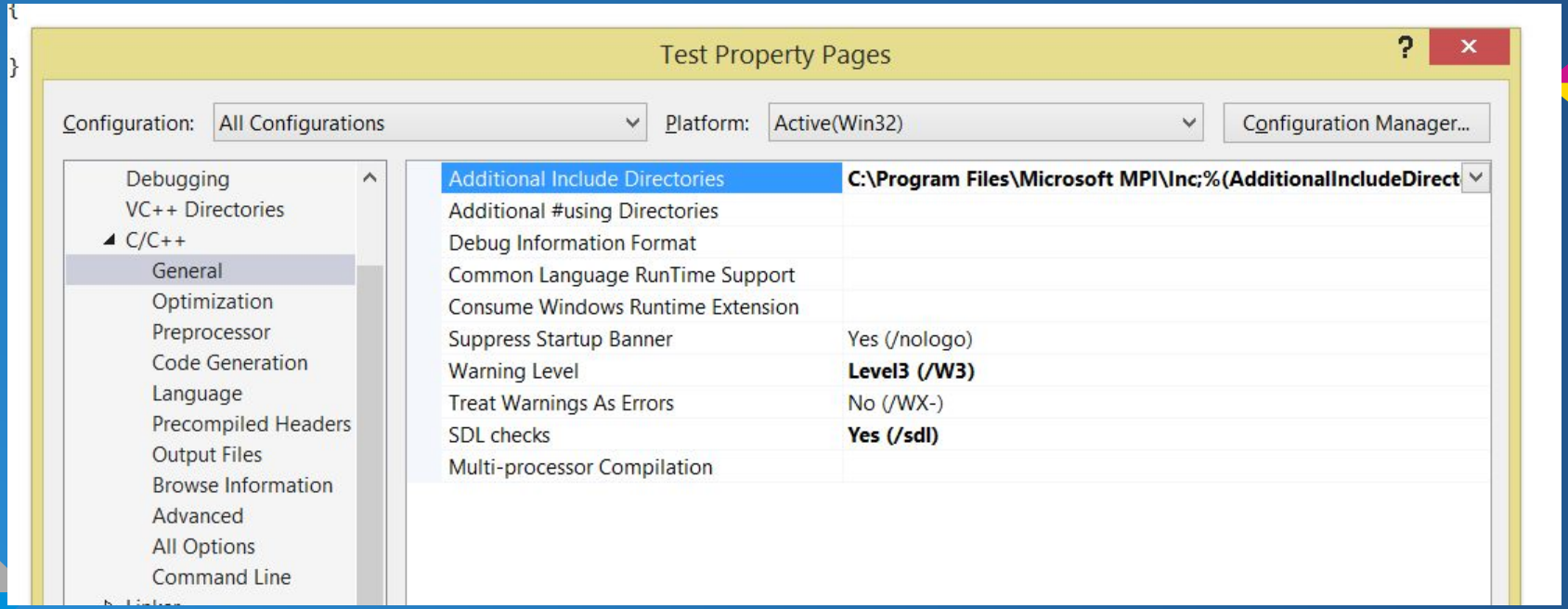
Пользовательские типы данных
Упаковка данных
Работы с группами и коммуникаторами
MPI 2.0

# Преимущества и недостатки

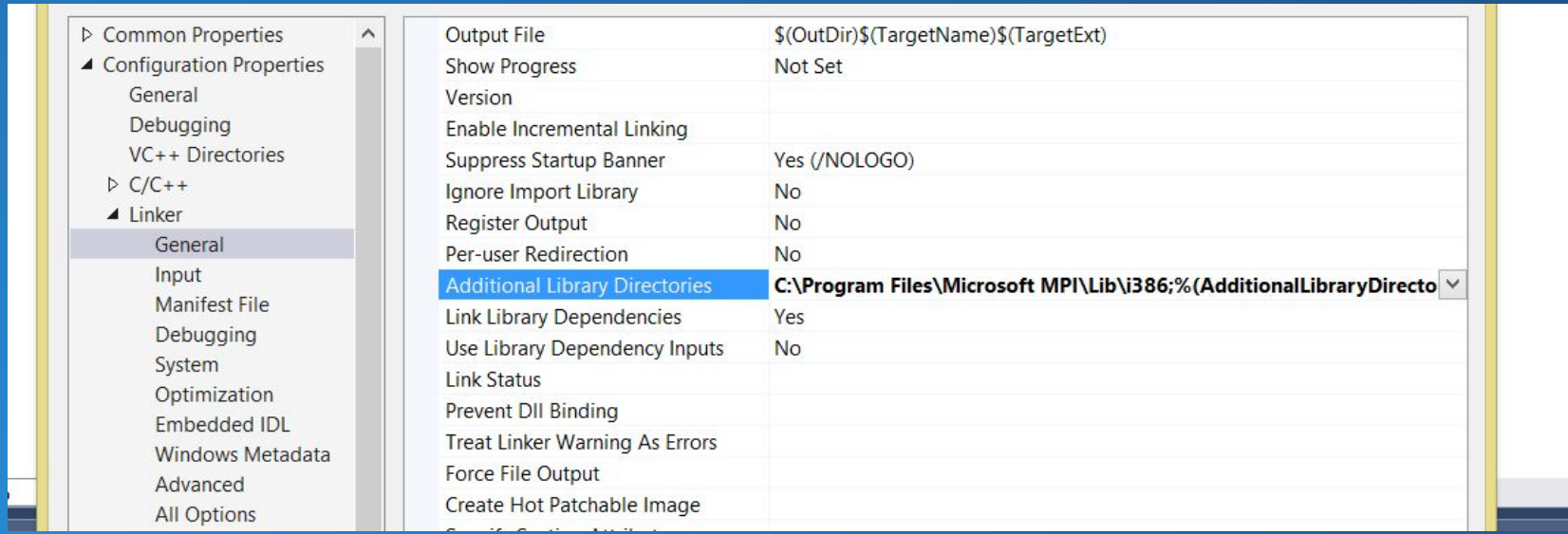
# Создание приложений с использованием MS-MPI



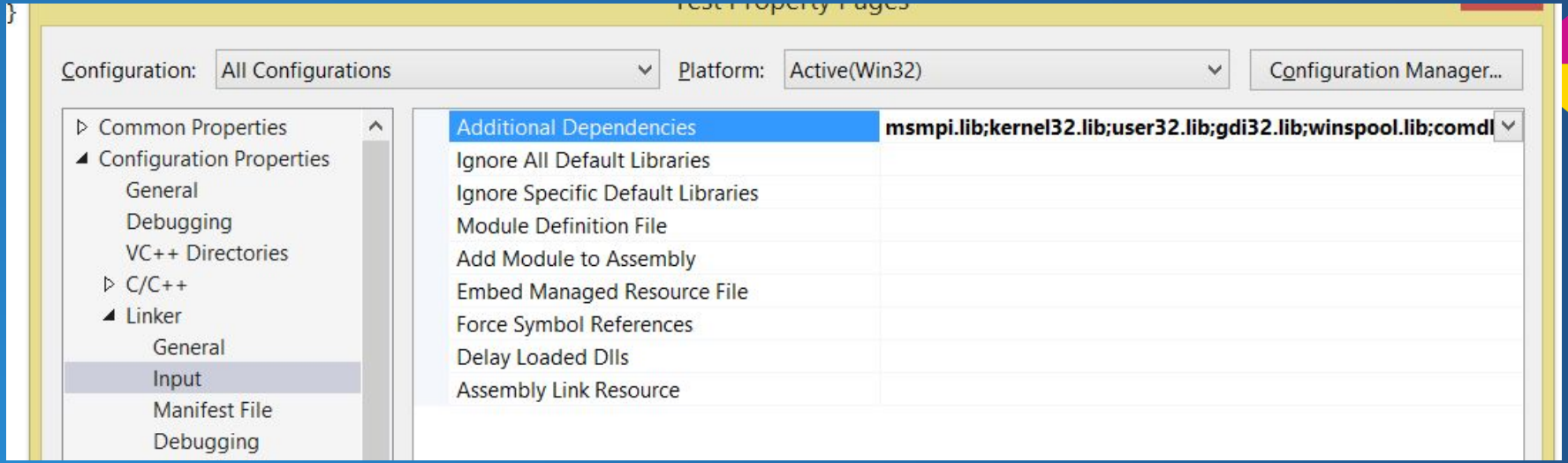
# MS-MPI


Test Property Pages
Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
All Options
Command Line
Additional Include Directories
C:\Program Files\Microsoft MPI\Inc;%(AdditionalIncludeDirect
Additional #using Directories
Debug Information Format
Common Language RunTime Support
Consume Windows Runtime Extension
Suppress Startup Banner
Yes (/nologo)
Warning Level
Level3 (/W3)
Treat Warnings As Errors
No (/WX-)
SDL checks
Yes (/sdl)
Multi-processor Compilation

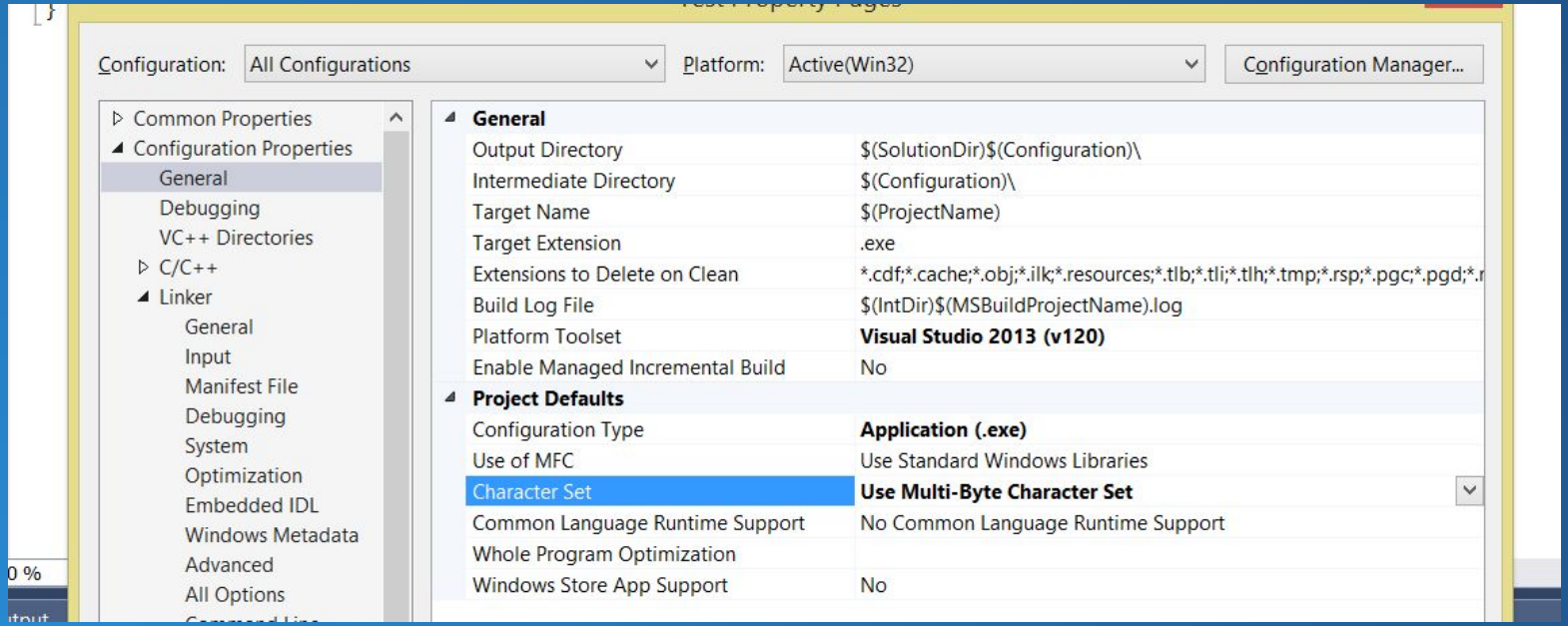
# MS-MPI


Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Output File
$(OutDir)$(TargetName)$(TargetExt)
Show Progress
Not Set
Version
Enable Incremental Linking
Suppress Startup Banner
Yes (/NOLOGO)
Ignore Import Library
No
Register Output
No
Per-user Redirection
No
Additional Library Directories
C:\Program Files\Microsoft MPI\Lib\i386;%(AdditionalLibraryDirecto
Link Library Dependencies
Yes
Use Library Dependency Inputs
No
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output
Create Hot Patchable Image

# MS-MPI


Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
Additional Dependencies
msmpi.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdl
Ignore All Default Libraries
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded Dlls
Assembly Link Resource

# MS-MPI


Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
General
Output Directory $(SolutionDir)$(Configuration)\
Intermediate Directory $(Configuration)\
Target Name $(ProjectName)
Target Extension .exe
Extensions to Delete on Clean *.cdf;*.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;*.pgc;*.pgd;*.
Build Log File $(IntDir)$(MSBuildProjectName).log
Platform Toolset Visual Studio 2013 (v120)
Enable Managed Incremental Build No
Project Defaults
Configuration Type Application (.exe)
Use of MFC Use Standard Windows Libraries
Character Set Use Multi-Byte Character Set
Common Language Runtime Support No Common Language Runtime Support
Whole Program Optimization
Windows Store App Support No