Лекция 7
Введение в MPI

Параллельное программирование

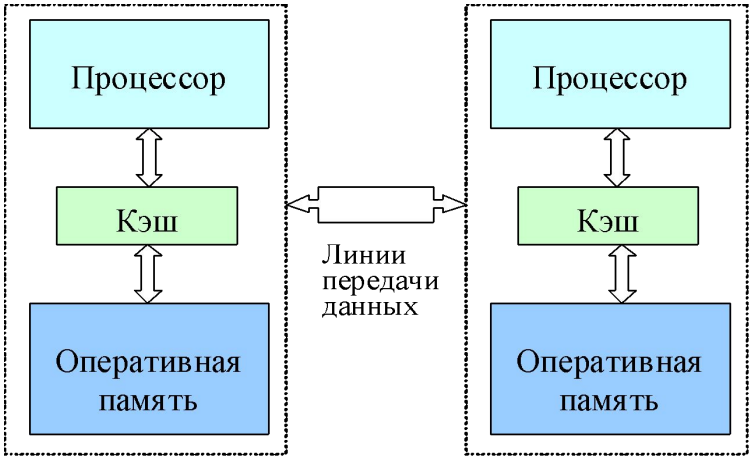
доцент М.А. Сокольская

План.

1. MPI: основные понятия и определения
2. Введение в MPI
   a) Инициализация и завершение MPI программ
   b) Определение количества и ранга процессов
   c) Прием и передача сообщений
   d) Определение времени выполнения MPI программы
7. Пример: первая программа с использованием MPI

Понятие MPI

MPI используется в вычислительных системах с распределенной памятью, в которых процессоры работают независимо друг от друга.



Для организации параллельных вычислений в таких системах необходимо уметь:

- *распределять* вычислительную нагрузку,
- *организовать* информационное взаимодействие (*передачу данных*) между процессорами.

Решение всех перечисленных вопросов обеспечивает ***MPI - интерфейс передачи данных*** *(message passing interface)*

Стандарт MPI

1993 г. – объединение нескольких групп в MPI Forum для создания единых требований к средствам программирования многопроцессорных систем с распределённой памятью .

Результат – стандарт MPI 1.0 в 1994 г.

Основные положения стандарта:

1. Реализации стандарта должны быть через подключаемые библиотеки или модули, без создания новых компиляторов или языков.
2. Библиотеки должны реализовывать все возможные типы обменов данными между процессорами (вычислительными узлами)

Стандарт MPI

1997 г – стандарт MPI 2.0

Возможности, заложенные в стандарт превышают возможности самих машин, поэтому версия 2.0 широко применяется только сейчас.

- В рамках MPI для решения задачи разрабатывается **одна программа**, она запускается на выполнение **одновременно на всех** имеющихся процессорах
- Для организации различных вычислений на разных процессорах:
  - Есть возможность подставлять разные данные для программы на разных процессорах,
  - Имеются средства для идентификации процессора, на котором выполняется программа

Такой способ организации параллельных вычислений обычно именуется как *модель "одна программа множество процессов"* (*single program multiple processes* or *SPMP*)

- В MPI существует *множество операций передачи данных*:
  - обеспечиваются разные способы пересылки данных;
  - реализованы практически все основные коммуникационные операции.

Эти возможности являются наиболее сильной стороной MPI (об этом, в частности, свидетельствует и само название MPI)

Библиотека MPI

Существует для двух языков:

- Fortran
- ***C/C++***

Представляет собой реализацию общих положений стандарта под тот или иной язык.

Мы рассматриваем реализацию **mpi.h** для C/C++

Работа на кластере

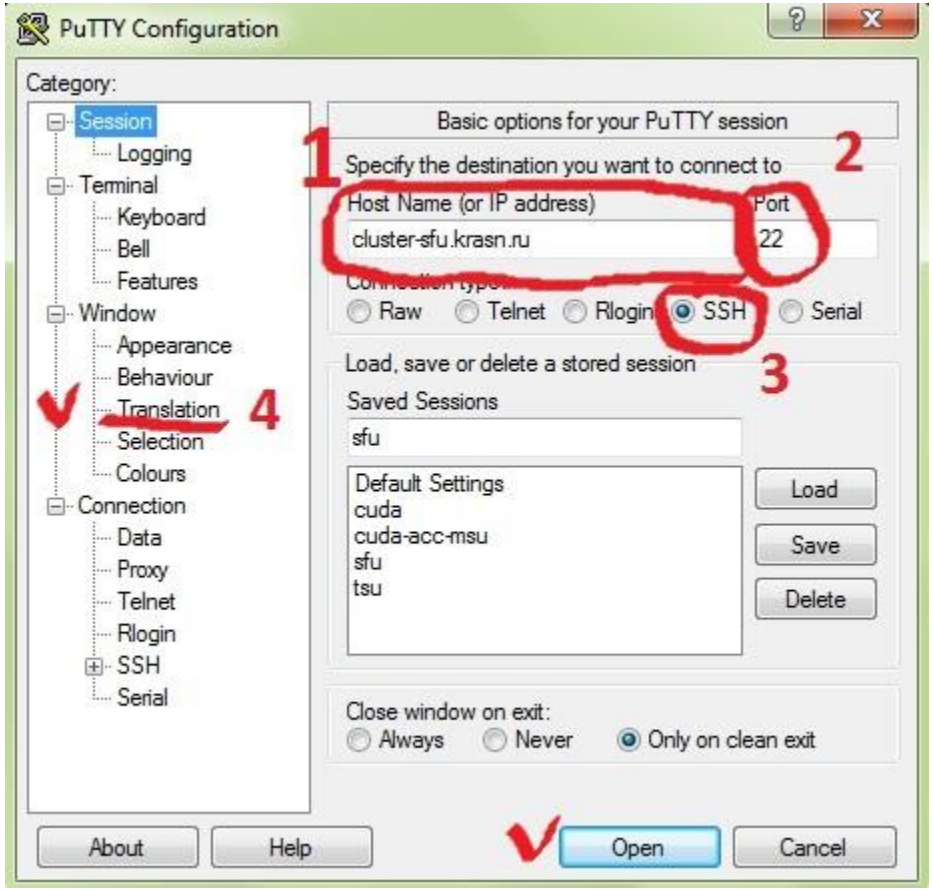
Необходимое ПО устанавливается (как правило) на кластерных системах. Доступ к кластеру КГПУ осуществляется удалённо.

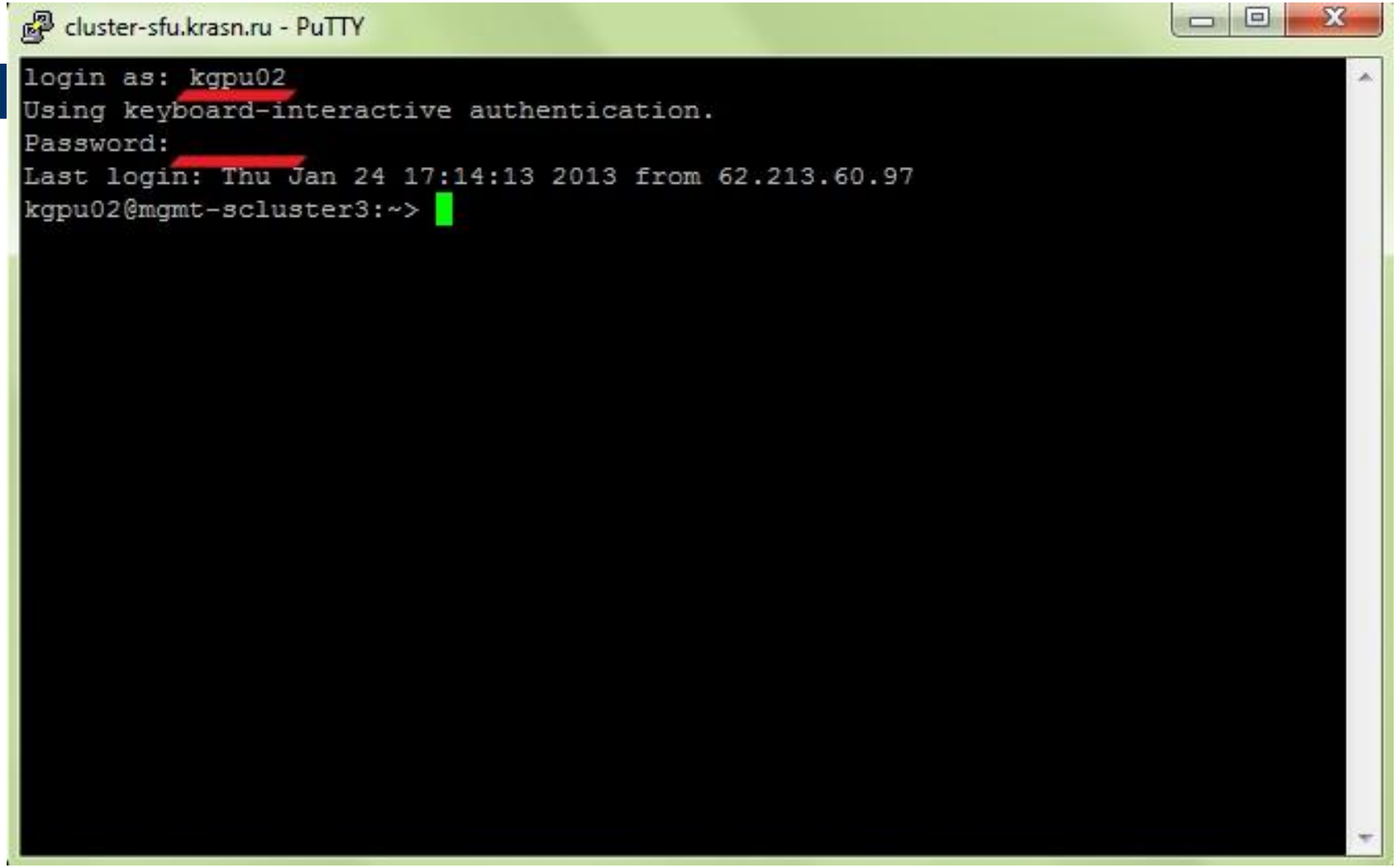
Программы (для ОС Windows):

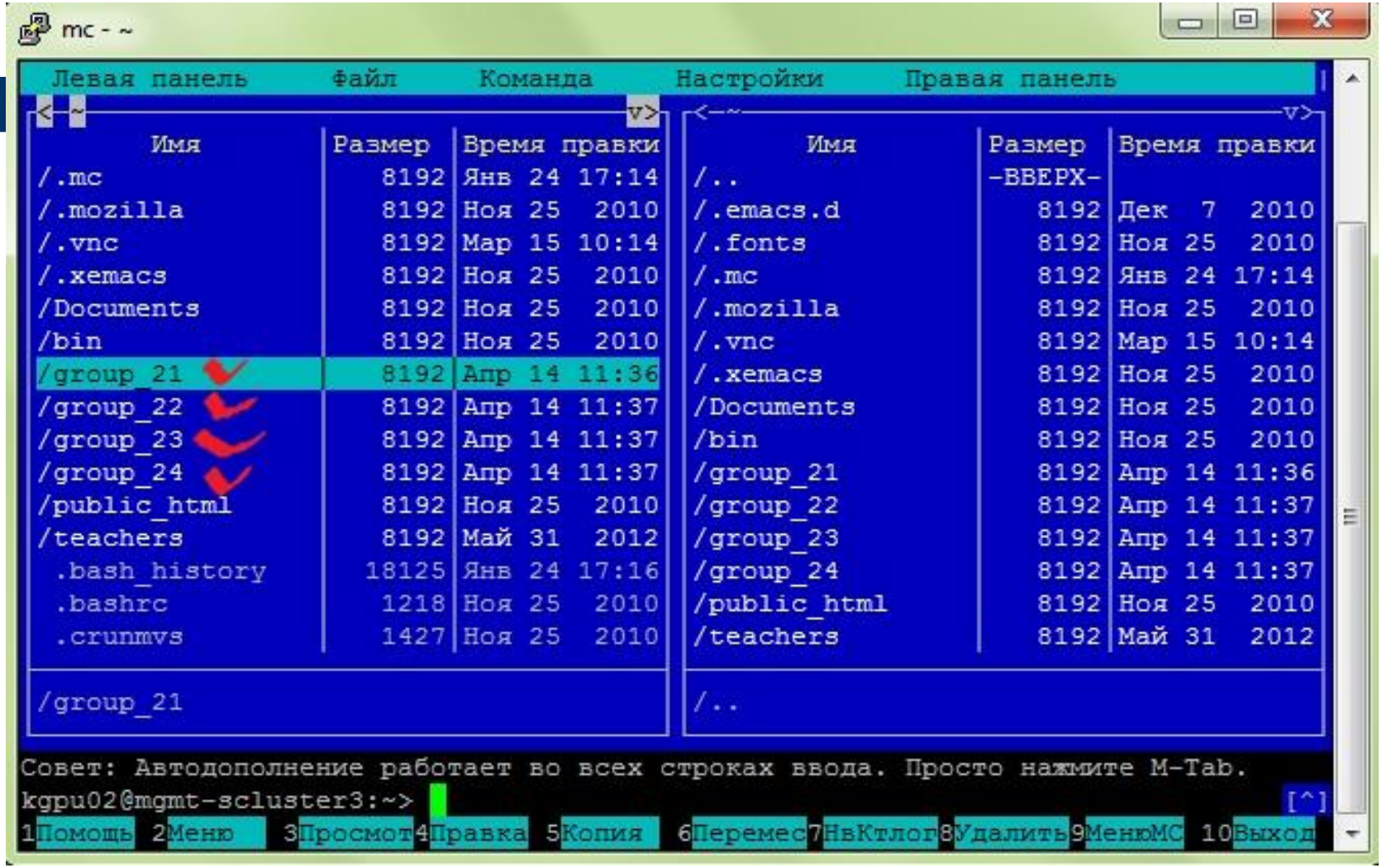
- *putty*: для доступа к кластеру, запуска и компиляции программ;
- *WinSCP*: для обмена файлами между кластером и удалённой машиной.

Для ОС Linux – доступ на кластер с командной строки

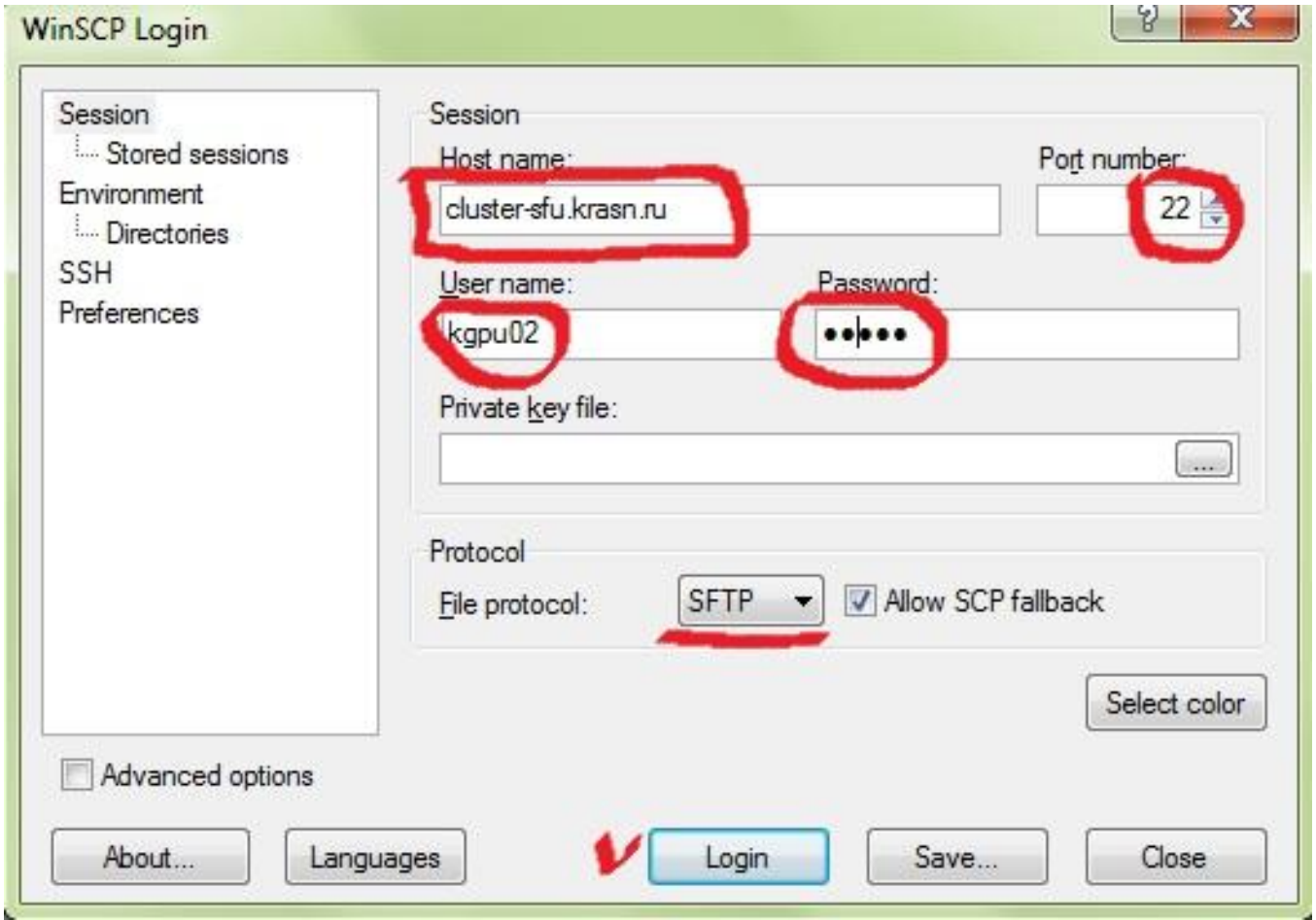
Использование Putty

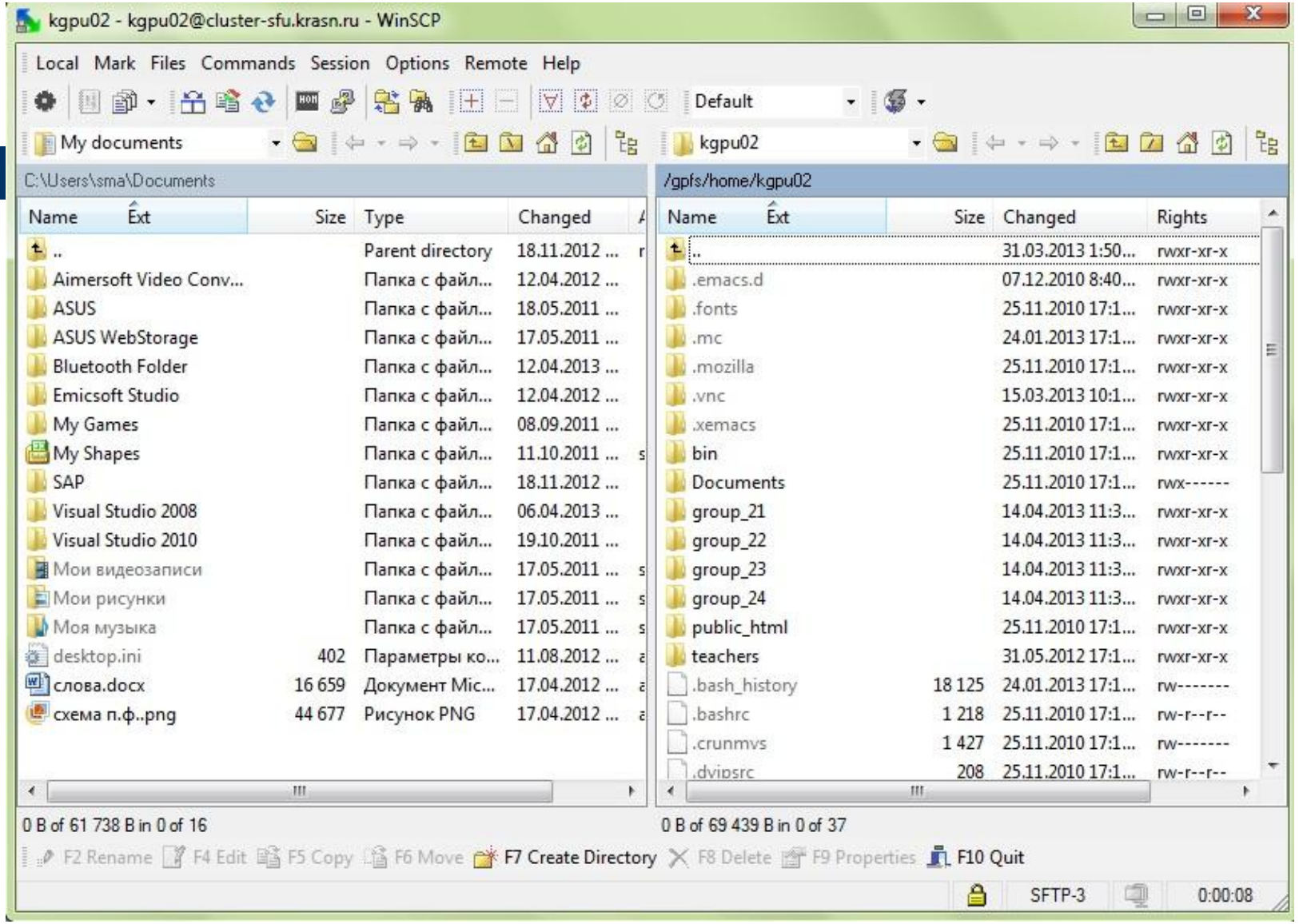



cluster-sfu.krasn.ru - PuTTY
login as: kgpu02
Using keyboard-interactive authentication.
Password:
Last login: Thu Jan 24 17:14:13 2013 from 62.213.60.97
kgpu02@mgmt-scluster3:~>


mc - ~
Левая панель Файл Команда Настройки Правая панель
Имя Размер Время правки
/.mc 8192 Янв 24 17:14
/.mozilla 8192 Ноя 25 2010
/.vnc 8192 Мар 15 10:14
/.xemacs 8192 Ноя 25 2010
/Documents 8192 Ноя 25 2010
/bin 8192 Ноя 25 2010
/group_21 8192 Апр 14 11:36
/group_22 8192 Апр 14 11:37
/group_23 8192 Апр 14 11:37
/group_24 8192 Апр 14 11:37
/public_html 8192 Ноя 25 2010
/teachers 8192 Май 31 2012
.bash_history 18125 Янв 24 17:16
.bashrc 1218 Ноя 25 2010
.crunmvs 1427 Ноя 25 2010
/group_21
Имя Размер Время правки
/.. -ВВЕРХ-
/.emacs.d 8192 Дек 7 2010
/.fonts 8192 Ноя 25 2010
/.mc 8192 Янв 24 17:14
/.mozilla 8192 Ноя 25 2010
/.vnc 8192 Мар 15 10:14
/.xemacs 8192 Ноя 25 2010
/Documents 8192 Ноя 25 2010
/bin 8192 Ноя 25 2010
/group_21 8192 Апр 14 11:36
/group_22 8192 Апр 14 11:37
/group_23 8192 Апр 14 11:37
/group_24 8192 Апр 14 11:37
/public_html 8192 Ноя 25 2010
/teachers 8192 Май 31 2012
/..
Совет: Автодополнение работает во всех строках ввода. Просто нажмите M-Tab.
kgpu02@mgmt-scluster3:~>
1Помощь 2Меню 3Просмот 4Правка 5Копия 6Перемес 7НвКтлог 8Удалить 9МенюМС 10Выход

Использование WinSCP




kgpu02 - kgpu02@cluster-sfu.krasn.ru - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
My documents
kgpu02
C:\Users\sma\Documents
/gpfs/home/kgpu02
Name Ext Size Type Changed
.. Parent directory 18.11.2012 ...
Aimersoft Video Conv... Папка с файл... 12.04.2012 ...
ASUS Папка с файл... 18.05.2011 ...
ASUS WebStorage Папка с файл... 17.05.2011 ...
Bluetooth Folder Папка с файл... 12.04.2013 ...
Emicsoft Studio Папка с файл... 12.04.2012 ...
My Games Папка с файл... 08.09.2011 ...
My Shapes Папка с файл... 11.10.2011 ...
SAP Папка с файл... 18.11.2012 ...
Visual Studio 2008 Папка с файл... 06.04.2013 ...
Visual Studio 2010 Папка с файл... 19.10.2011 ...
Мои видеозаписи Папка с файл... 17.05.2011 ...
Мои рисунки Папка с файл... 17.05.2011 ...
Моя музыка Папка с файл... 17.05.2011 ...
desktop.ini 402 Параметры ко... 11.08.2012 ...
слова.docx 16 659 Документ Mic... 17.04.2012 ...
схема п.ф..png 44 677 Рисунок PNG 17.04.2012 ...
Name Ext Size Changed Rights
.. 31.03.2013 1:50... rwxr-xr-x
.emacs.d 07.12.2010 8:40... rwxr-xr-x
.fonts 25.11.2010 17:1... rwxr-xr-x
.mc 24.01.2013 17:1... rwxr-xr-x
.mozilla 25.11.2010 17:1... rwxr-xr-x
.vnc 15.03.2013 10:1... rwxr-xr-x
.xemacs 25.11.2010 17:1... rwxr-xr-x
bin 25.11.2010 17:1... rwxr-xr-x
Documents 25.11.2010 17:1... rwx------
group_21 14.04.2013 11:3... rwxr-xr-x
group_22 14.04.2013 11:3... rwxr-xr-x
group_23 14.04.2013 11:3... rwxr-xr-x
group_24 14.04.2013 11:3... rwxr-xr-x
public_html 25.11.2010 17:1... rwxr-xr-x
teachers 31.05.2012 17:1... rwxr-xr-x
.bash_history 18 125 24.01.2013 17:1... rw-------
.bashrc 1 218 25.11.2010 17:1... rw-r--r--
.crunmvs 1 427 25.11.2010 17:1... rw-------
.dvipsrc 208 25.11.2010 17:1... rw-r--r--
0 B of 61 738 B in 0 of 16
0 B of 69 439 B in 0 of 37
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:00:08

MPI: основные понятия и определения…

Понятие параллельной программы

- Под *параллельной программой* в рамках MPI понимается множество одновременно выполняемых *процессов*:
  - процессы могут выполняться на разных процессорах; вместе с этим, на одном процессоре могут располагаться несколько процессов,
  - Каждый процесс параллельной программы порождается на основе *копии одного и того же программного кода* (модель SPMP).

MPI: основные понятия и определения…

Количество процессов определяется *в момент запуска* параллельной *программы* средствами среды исполнения MPI программ.

Все процессы программы **последовательно перенумерованы.**

Определение:

Номер процесса именуется ***рангом*** процесса.

MPI: основные понятия и определения…

В основу MPI положены четыре основных понятия:

- Тип операции передачи сообщения
- Тип данных, пересылаемых в сообщении
- Понятие коммуникатора (*группы процессов*)
- Понятие виртуальной топологии

MPI: основные понятия и определения…

Операции передачи данных

Основу MPI составляют операции передачи сообщений.

- Среди предусмотренных в составе MPI функций различаются:
  - парные (*point-to-point*) операции между двумя процессами,
  - коллективные (*collective*) операции для одновременного взаимодействия нескольких процессов.

MPI: основные понятия и определения…

Понятие коммуникаторов

Определение:

Коммуникатор в MPI - специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (*контекст)*:

- парные операции передачи данных выполняются для процессов, принадлежащих одному и тому же коммуникатору,
- коллективные операции применяются одновременно для всех процессов одного коммуникатора.

Указание коммуникатора является обязательным для **всех** операций передачи данных в MPI.

MPI: основные понятия и определения…

В ходе вычислений могут создаваться новые и удаляться существующие коммуникаторы.

Один и тот же процесс может принадлежать разным коммуникаторам.

Все имеющиеся в параллельной программе процессы входят в состав создаваемого *по умолчанию коммуникатора* с идентификатором **MPI_COMM_WORLD**.

Типы данных MPI

При выполнении операций передачи сообщений для определения передаваемых или получаемых данных в функциях MPI необходимо указывать тип пересылаемых данных.

MPI содержит большой набор *базовых типов данных*, во многом совпадающих с типами данных в языках C/C++ и Fortran.

В MPI можно создавать *новые производные типы данных* для более точного и краткого описания содержимого пересылаемых сообщений.

Инициализация и завершение MPI программ

Первой вызываемой функцией MPI должна быть функция:

int **MPI_Init** (int *argc, char ***argv);

(служит для *инициализации среды* выполнения MPI программы; параметрами функции являются количество аргументов в командной строке ОС и текст самой командной строки.)

Последней вызываемой функцией MPI обязательно должна являться функция:

int **MPI_Finalize** (void);

Инициализация и завершение MPI программ

Структура параллельной программы, разработанная с использованием MPI, должна иметь следующий вид:

```
#include "mpi.h"
int main ( int argc, char *argv[] ) {
  <программный код без использования MPI
функций>
  MPI_Init ( &agrc, &argv );
    <программный код с использованием MPI
функций >
  MPI_Finalize();
  <программный код без использования MPI
функций >
  return 0;
```

Определение количества и ранга процессов

Определение *количества процессов* в выполняемой параллельной программе осуществляется при помощи функции:

int **MPI_Comm_size** (MPI_Comm comm, int *size);

Для определения *ранга процесса* используется функция:

int **MPI_Comm_rank** (MPI_Comm comm, int *rank);

Определение количества и ранга процессов…

Как правило, вызов функций *MPI_Comm_size* и *MPI_Comm_rank* выполняется сразу после *MPI_Init*:

```
#include "mpi.h"
int main ( int argc, char *argv[] ) {
    int ProcNum, ProcRank;
    <программный код без использования MPI функций>
    MPI_Init (&agrc, &argv);
    MPI_Comm_size (MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank (MPI_COMM_WORLD, &ProcRank);
    <программный код с использованием MPI функций >
    MPI_Finalize();
    <программный код без использования MPI функций >
    return 0;
}
```

Определение количества и ранга процессов

Коммуникатор *MPI_COMM_WORLD* создается по умолчанию и представляет все процессы выполняемой параллельной программы;

Ранг, получаемый при помощи функции *MPI_Comm_rank*, является рангом процесса, выполнившего вызов этой функции, и, тем самым, переменная *ProcRank* будет принимать *различные значения в разных процессах*.

Работа функции MPI_Comm_rank()

Парная передача сообщений

Для *передачи сообщения* процесс-отправитель должен выполнить функцию:

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

где:

buf – адрес буфера памяти, в котором располагаются данные отправляемого сообщения,

count – количество элементов данных в сообщении,

type - тип элементов данных пересылаемого сообщения,

dest - ранг процесса, **которому** отправляется сообщение,

tag - значение-тег, используемое для идентификации сообщений,

comm - коммуникатор, в рамках которого выполняется передача данных.

Передача сообщений…

- Отправляемое сообщение определяется через указание блока памяти (*буфера*), в котором это сообщение располагается. Используемая для указания буфера триада (***buf, count, type***) входит в состав параметров практически всех функций передачи данных,
- Процессы, между которыми выполняется передача данных, обязательно должны принадлежать коммуникатору, указываемому в функции *MPI_Send*,
- Параметр *tag* используется только если нужно различать передаваемые сообщения, в противном случае в качестве значения параметра может быть использовано произвольное целое число.

Типы данных для передачи и приёма сообщений

Базовые типы данных MPI для языка C/C++

MPI_Datatype	C Datatype
MPI_BYTE	
MPI_CHAR	signed char
MPI_DOUBLE	double
MPI_FLOAT	float
MPI_INT	int
MPI_LONG	long
MPI_LONG_DOUBLE	long double
MPI_PACKED	
MPI_SHORT	short
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long
MPI_UNSIGNED_SHORT	unsigned short

Прием сообщений

Для *приема сообщения* процесс-получатель должен выполнить функцию:

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

где

- **buf, count, type** – буфер памяти для приема сообщения
- **source** - ранг процесса, от которого должен быть выполнен прием сообщения,
- **tag** - тег сообщения, которое должно быть принято для процесса,
- **comm** - коммуникатор, в рамках которого выполняется передача данных,
- **status** – указатель на структуру данных с информацией о результате выполнения операции приема данных.

Прием сообщений…

- Буфер памяти **должен быть достаточным для приема** сообщения, а тип элементов передаваемого и принимаемого сообщения должны совпадать; при нехватке памяти часть сообщения будет потеряна и в коде завершения функции будет зафиксирована ошибка переполнения,
- При приеме сообщения от любого процесса-отправителя для параметра *source* может быть указано значение *MPI_ANY_SOURCE*,
- При приеме сообщения с любым тегом для параметра *tag* может быть указано значение *MPI_ANY_TAG*,

Прием сообщений…

Параметр *status* позволяет определить ряд характеристик принятого сообщения:

status.MPI_SOURCE – ранг процесса-отправителя принятого сообщения,

status.MPI_TAG - тег принятого сообщения.

Функция

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

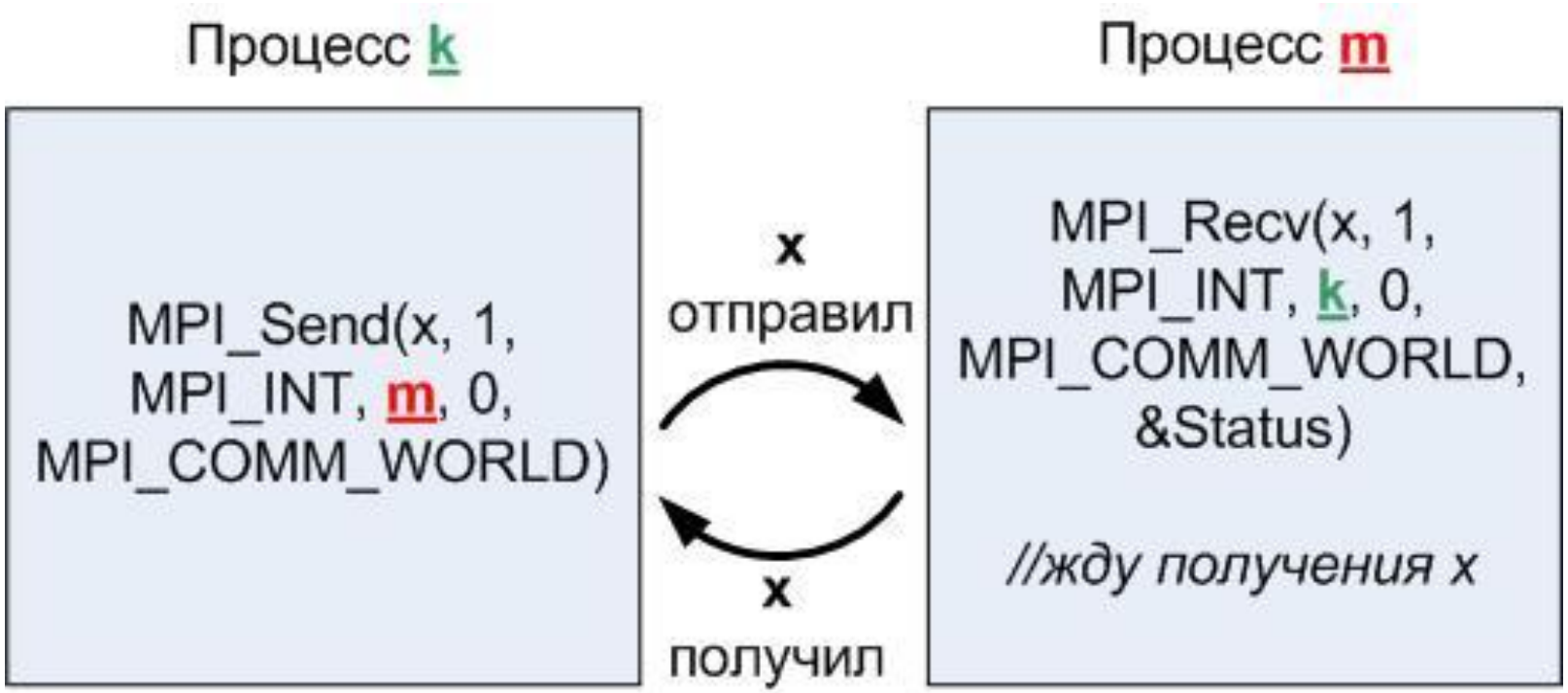
возвращает в переменной *count* количество элементов типа *type* в принятом сообщении.

Прием сообщений…

Функция *MPI_Recv* является *блокирующей* для процесса-получателя, т.е. его выполнение приостанавливается до завершения работы функции.

Таким образом, **если** по каким-то причинам **ожидаемое** для приема **сообщение будет отсутствовать**, **выполнение** параллельной **программы будет блокировано**.

Парные функции приема-передачи сообщений



Первая параллельная программа с использованием MPI: “Hello, world!!!”

```
#include " mpi.h"
int main(int argc, char* argv[])
{
  int ProcNum, ProcRank, RecvRank;
  MPI_Status Status;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
  MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
  if ( ProcRank == 0 )
  { // Действия для процесса 0
     printf ("\n Hello from process %3d", ProcRank);
     for ( int i=1; i < ProcNum; i++ )
     {
```

```
    MPI_Recv(&RecvRank, 1, MPI_INT,
   MPI_ANY_SOURCE, MPI_ANY_TAG,
   MPI_COMM_WORLD, &Status);
     printf("\n Hello from process %d", RecvRank);
    }
 }
 else // Действия для всех остальных процессов
    MPI_Send(&ProcRank,1,MPI_INT,0,0,MPI_COMM_WO
   RLD);
 MPI_Finalize(); // Завершение работы
 return 0;
}
```

Первая параллельная программа с использованием MPI…

- Каждый процесс определяет свой ранг, после чего действия в программе разделяются (разные процессы выполняют различные действия),
- Все процессы, кроме процесса с рангом 0, передают значение своего ранга нулевому процессу,
- Процесс с рангом 0 сначала печатает значение своего ранга, а далее последовательно принимает сообщения с рангами процессов и также печатает их значения,

Первая параллельная программа с использованием MPI…

Порядок приема сообщений заранее не определен и зависит от условий выполнения параллельной программы (более того, этот порядок может изменяться от запуска к запуску).

Если это не приводит к потере эффективности, следует обеспечивать однозначность расчетов и при использовании параллельных вычислений:

Самостоятельно:

Подумать, как обеспечить вывод приветствий от процессов в порядке нумерации процессов.

Итоги

Мы рассмотрели:

Основные определения и понятия MPI, основные функции MPI и их применение.