Введение в параллельные вычисления

д.т.н. Мокрова Наталия Владиславовна

асп. Морунов Егор, Сырко Денис

пятница	ауд. 118, 119	
12:50 – 14:20	Лекция	
	1 неделя	2 неделя
14:35 - 17:55	Лаб – КС44	Лаб - КС40

Выбор модели программирования

Системы с общей памятью

- C++11 threads (pthreads)
- OpenMP

Системы с распределенной памятью

- MPI = message passing interface

Гетерогенные системы (графические процессоры)

- CUDA

Лекция 11. Системы с распределенной памятью. MPI

Стандарты MPI http://www.mpi-forum.org спецификация MPI-1.1

Обмен данными с использованием MPI.

- http://habrahabr.ru/company/intel/blog/251357/

Лекции и семинары

- http://www.slideshare.net/Aleximos/mpi-9793227

Технологии

- http://parallel.ru/tech/tech_dev/mpi.html
- http://parallel.ru/tech/tech_dev/MPI/examples/ (примеры)

Примеры из учебника "Технологии параллельного программирования MPI и OpenMP"

- http://parallel.ru/tech/tech_dev/MPI%26OpenMP/examples/

Литература

- Богачёв К.Ю. Основы параллельного программирования. М.: БИНОМ. Лаборатория знаний, 2003.
- Немнюгин С.А. Средства программирования для многопроцессорных вычислительных систем. Спб., 2007.
- Антонов А.С. Параллельное программирование с использованием технологии MPI. М.: Изд-во МГУ, 2004.
- Шпаковский Г.И. , Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. Мн.: БГУ, 2002.

Учебные материалы по MPI, доступные в Internet

- Лекция об MPI в курсе "Параллельная обработка данных" (Вл.В.Воеводин).
- Вычислительный практикум по технологии MPI (А.С.Антонов).
- Глава об MPI(link is external) в книге Яна Фостера "Designing and Building Parallel Programs".
- Учебные материалы по MPI(link is external) на сервере MHPCC.
- MPI: The Complete Reference(link is external). Авторы: Marc Snir, Steve Otto, Steve Huss-Lederman, David Walker, Jack Dongarra(link is external).
- Tutorial on MPI: The Message Passing Interface(link is external). Автор: Bill Gropp(link is external).
- Writing Message-Passing Parallel Programs with MPI(link is external) – учебный курс по MPI, созданный в EPCC (The University of Edinburgh). Авторы: Neil MacDonald, Elspeth Minty, Mario Antonioletti, Joel Malard, Tim Harding, Simon Brown.

С программой что-то не так

1. Пример: процессор второго компьютера, в отличие от первого, имел лишь два канала связи с памятью, а не три, как предписано стандартной конфигурацией.
2. Для исследования эффективности программного окружения конкретного компьютера предложена система тестов, позволяющая определить реальную скорость передачи сообщений, скорость срабатывания различных функций систем с передачей сообщений, время задержки на барьерах, эффективность обменов параллельных процессов в различных логических топологиях и др. (http://parallel.ru/testmpi).
3. Структура программы – можно ли не изменяя алгоритма улучшить эффективность? (http://parallel.ru/v-ray)
4. Заменить отдельные алгоритмические компоненты на другие, свойства которых лучше соответствуют особенностям архитектуры целевого компьютера.

АНАЛИЗ КОНФИГУРАЦИИ КОМПЬЮТЕРА

↓

АНАЛИЗ ЭФФЕКТИВНОСТИ СИСТЕМНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

↓

АНАЛИЗ СТРУКТУРЫ ПРОГРАММЫ

↓

АНАЛИЗ АЛГОРИТМИЧЕСКОГО ПОДХОДА

http://parallel.ru/tech/opt/

Message Passing Interface

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 – наиболее распространённый стандарт интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров.

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++.

MPI ориентирован на системы с распределенной памятью, когда затраты на передачу данных велики.

Технологии OpenMP и MPI могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

Стандарты MPI

Первая версия MPI разрабатывалась в 1993 – 1994 году, и MPI 1 вышла в 1994.

Большинство современных реализаций MPI поддерживают версию 1.1. Стандарт MPI версии 2.0 поддерживается большинством современных реализаций, однако некоторые функции могут быть реализованы не до конца.

Примеры реализаций MPI

- MPICH – самая распространённая свободная реализация, работает на UNIX-системах и Windows NT
- Intel MPI – коммерческая реализация для Windows / Linux
- Oracle HPC ClusterTools – бесплатная реализация для Solaris SPARC/x86 и Linux на основе Open MPI
- MPJ – MPI for Java
- MPJ Express – MPI на Java и др.

Функционирование интерфейса

Базовым механизмом связи между MPI процессами является передача и приём сообщений.

Сообщение несёт передаваемые данные и информацию, позволяющую принимающей стороне осуществлять их выборочный приём:

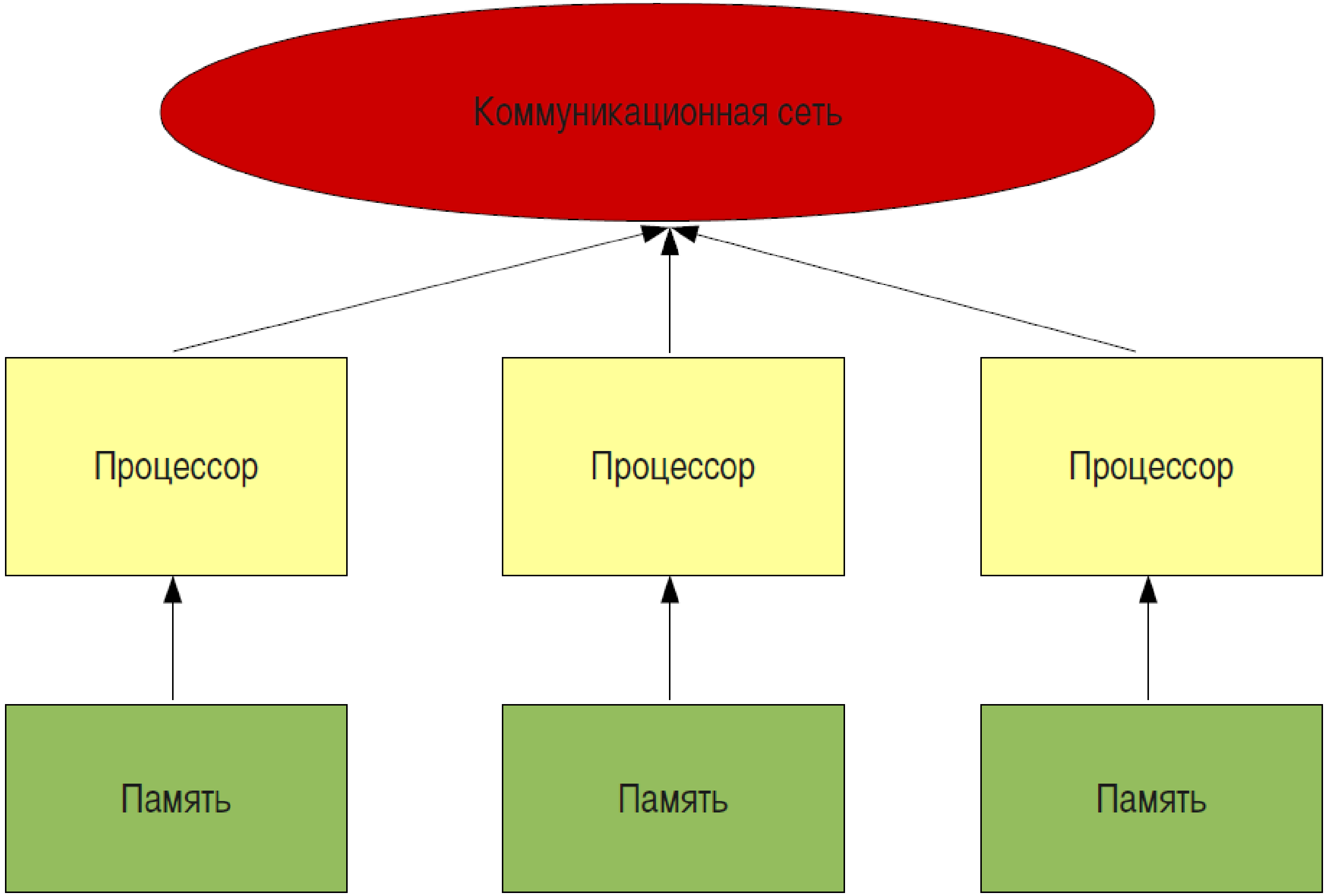
- отправитель – ранг (номер в группе) отправителя сообщения;
- получатель – ранг получателя;
- признак – может использоваться для разделения различных видов сообщений;
- коммуникатор – код группы процессов.

Операции приёма и передачи могут быть блокирующимися и неблокирующимися. Для неблокирующихся операций определены функции проверки готовности и ожидания выполнения.

Другим способом связи является удалённый доступ к памяти (RMA), позволяющий читать и изменять область памяти удалённого процесса. Локальный процесс может переносить область памяти удалённого процесса (внутри указанного процессами окна) в свою память и обратно, а также комбинировать данные, передаваемые в удалённый процесс с имеющимися в его памяти данными (например, путём суммирования). Все операции удалённого доступа к памяти не блокирующиеся, однако, до и после их выполнения необходимо вызывать блокирующиеся функции синхронизации.

Полная версия интерфейса содержит описание более 125 процедур и функций.

Модель передачи сообщений



Коммуникационная сеть

Процессор

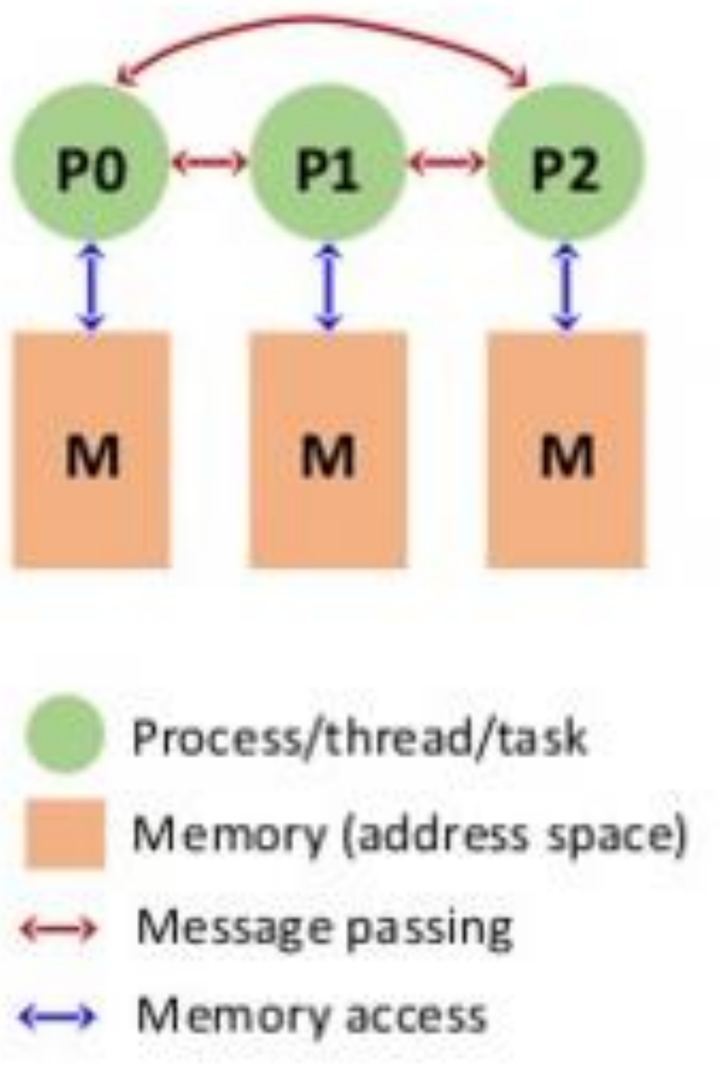
Процессор

Процессор

Память

Память

Память



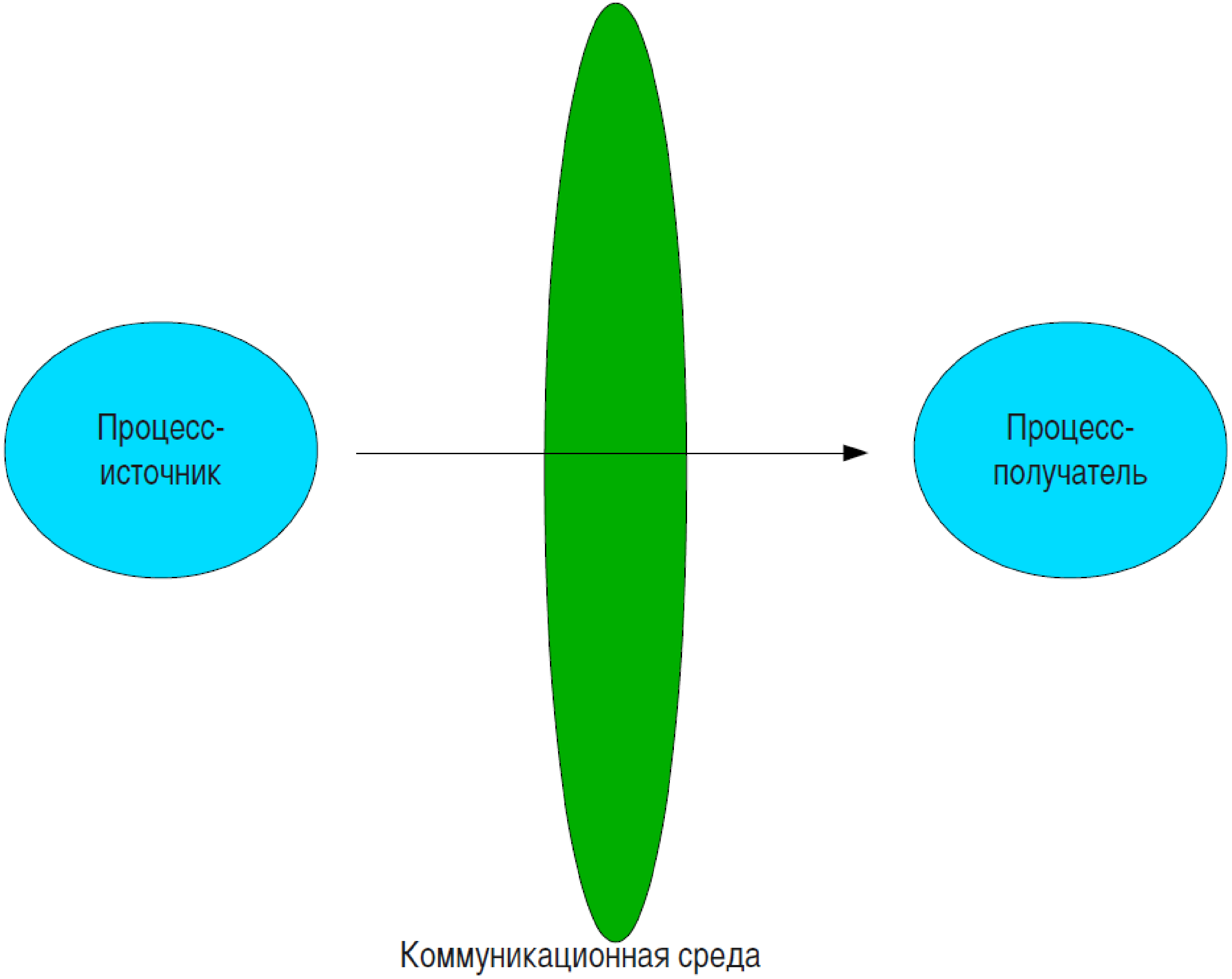
Программа состоит из N параллельных процессов, которые порождаются при запуске (MPI-1) или м.б. динамически созданы во время выполнения (MPI-2).

Передача сообщения – способ взаимодействия

Каждый процесс имеет уникальный идентификатор и изолированное адресное пространство.

Общих переменных или данных в MPI нет.

Процессы могут образовывать группы для реализации коллективных операций обмена информацией.



Терминология и обозначения

MPI – библиотека функций, предназначенная для поддержки работы параллельных процессов в терминах передачи сообщений.

Номер процесса – целое неотрицательное число, являющееся уникальным атрибутом каждого процесса.

Атрибуты сообщения – номер процесса-отправителя, номер процесса-получателя и идентификатор сообщения.

Заведена структура *MPI_Status*, содержащая поля:

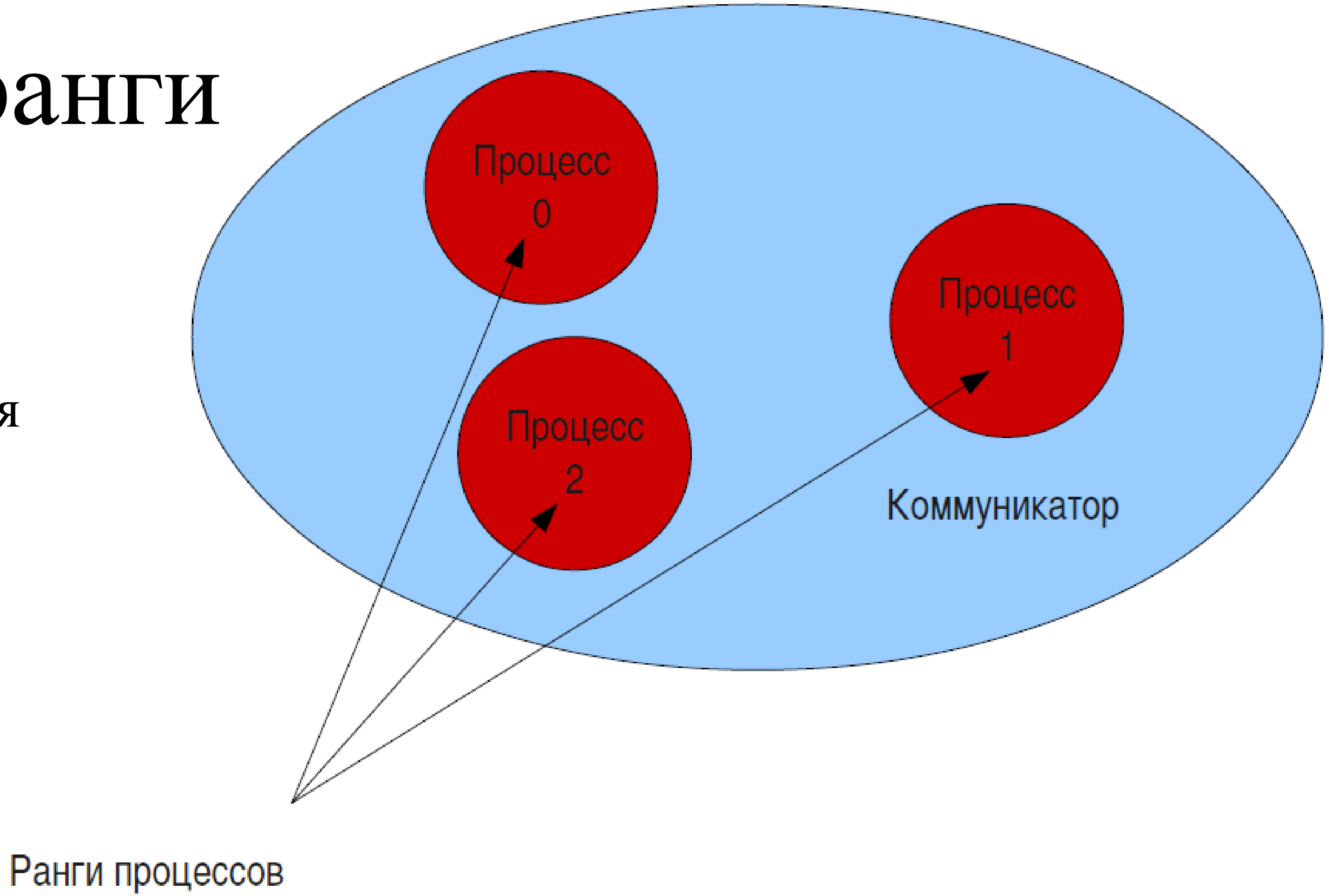
- *MPI_Source* (номер процесса отправителя),
- *MPI_Tag* (идентификатор сообщения),
- *MPI_Error* (код ошибки); могут быть и добавочные поля.

Идентификатор сообщения (msgtag) – атрибут сообщения, являющийся целым неотрицательным числом, лежащим в диапазоне от 0 до 32767.
Процессы объединяются в *группы*, внутри группы все процессы перенумерованы.
С каждой группой ассоциирован свой *коммуникатор*.
Процесс имеет два основных атрибута – *коммуникатор и номер в коммуникаторе*.

Коммуникатор и ранги

Коммуникатор (communicator) – множество процессов, образующих логическую область для выполнения коллективных операций (обменов информацией и др.)

Состав групп произволен. Группы могут совпадать, входить одна в другую, не пересекаться или пересекаться частично. Процессы могут взаимодействовать только внутри некоторого коммуникатора, сообщения в разных коммуникаторах не пересекаются.



Каждый процесс имеет специальный идентификатор – ранг (rank).
Каждый процесс в рамках одного коммуникатора имеет уникальный ранг.

Локализация и коммуникаторы

Коммуникатор – среда общения.

Коммуникаторы имеют предопределенный тип – MPI_Comm.

При старте программы все порожденные процессы работают в рамках коммуникатора MPI_COMM_WORLD.

Он существует всегда и служит для взаимодействия всех запущенных процессов MPI-программы.

При старте программы имеется коммуникатор MPI_COMM_SELF, содержащий только один текущий процесс, и коммуникатор MPI_COMM_NULL, не содержащий ни одного процесса.

При пересылке необходимо указать идентификатор группы, внутри которой производится эта пересылка. Все процессы содержатся в группе с предопределенным идентификатором MPI_COMM_WORLD.

Установка MPI

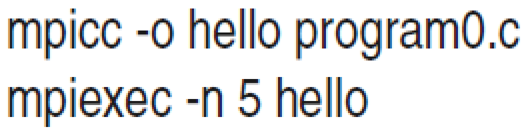
Одна из самых распространённых реализаций MPI – MPICH (MPI Chameleon).

Установка в Ubuntu: `sudo apt-get install mpich2`
Компиляция: `mpic++ -o test ./test.cpp`
Запуск: `mpirun -np 4 test`

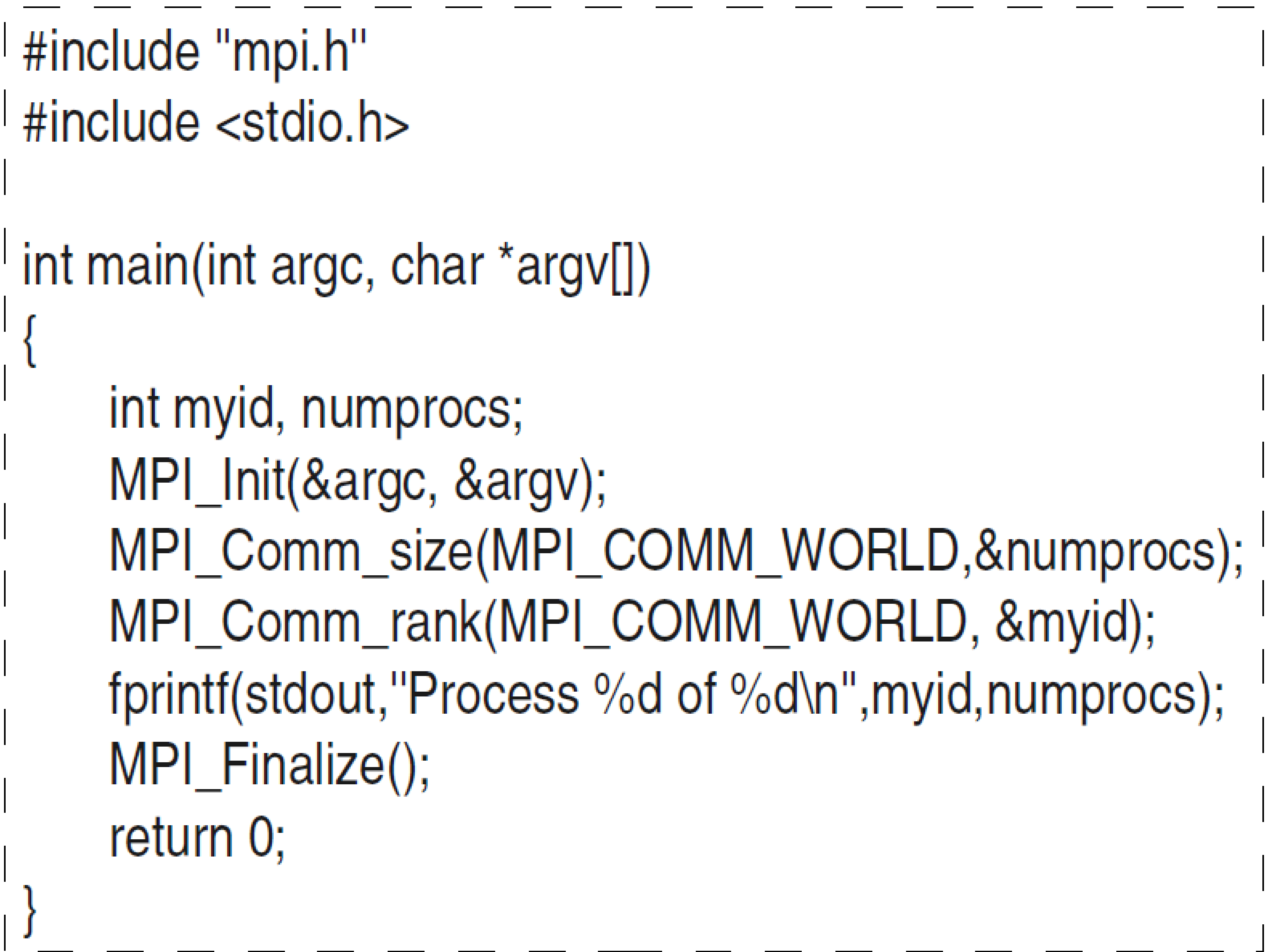
Официальный сайт: http://www.mpich.org/
Скачайть библиотеку MPICH2:
`wget http://www.mpich.org/static/downloads/3.0.4/mpich-3.0.4.tar.gz`
Распаковать tar-архив: `tar xzvf mpich-3.0.4.tar.gz`

Общий вид MPI программы

Параллельная программа с точки зрения MPI – набор процессов, запущенных на разных вычислительных узлах. Каждый процесс порождается на основе одного и того же программного кода.



```
mpicc -o hello program0.c
mpiexec -n 5 hello
```



```
#include "mpi.h"
#include <stdio.h>

int main(int argc, char *argv[])
{
    int myid, numprocs;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    fprintf(stdout,"Process %d of %d\n",myid,numprocs);
    MPI_Finalize();
    return 0;
}
```

Общие процедуры MPI

Общие процедуры необходимы в каждой параллельной программе

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

MPI_Init – инициализация параллельной части (реальная инициализация для каждого приложения происходит не более одного раза, если повторно, то действия не выполняются и происходит возврат из подпрограммы).

Все MPI-процедуры могут быть вызваны только после вызова *MPI_Init*.

Процедура возвращает: в случае успешного выполнения – *MPI_SUCCESS*, иначе – код ошибки.

int MPI_Finalize(void)

MPI_Finalize – завершение параллельной части приложения.

К моменту вызова *MPI_Finalize* некоторым процессом все действия, требующие его участия в обмене сообщениями, должны быть завершены.

Сложный тип аргументов *MPI_Init* предусмотрен, чтобы передавать всем процессам аргументы *main*:

```
int main(int argc, char** argv)
{
        MPI_Init(&argc, &argv);
                ...
        MPI_Finalize();
}
```

Общие процедуры MPI

int MPI_Comm_size(MPI_Comm comm, int* size)

Определение общего числа параллельных процессов в группе *comm*.

comm – идентификатор группы; OUT *size* – размер группы.

int MPI_Comm_rank(MPI_Comm comm, int* rank)

Определение номера процесса в группе *comm*.

Значение, возвращаемое по адресу *&rank*, лежит в диапазоне от 0 до *size_of_group-1*.

comm – идентификатор группы;

OUT *rank* – номер вызывающего процесса в группе *comm*.

double MPI_Wtime(void)

Функция возвращает астрономическое время в секундах (вещественное число), прошедшее с некоторого момента в прошлом.

Гарантируется, что этот момент не будет изменен за время существования процесса.

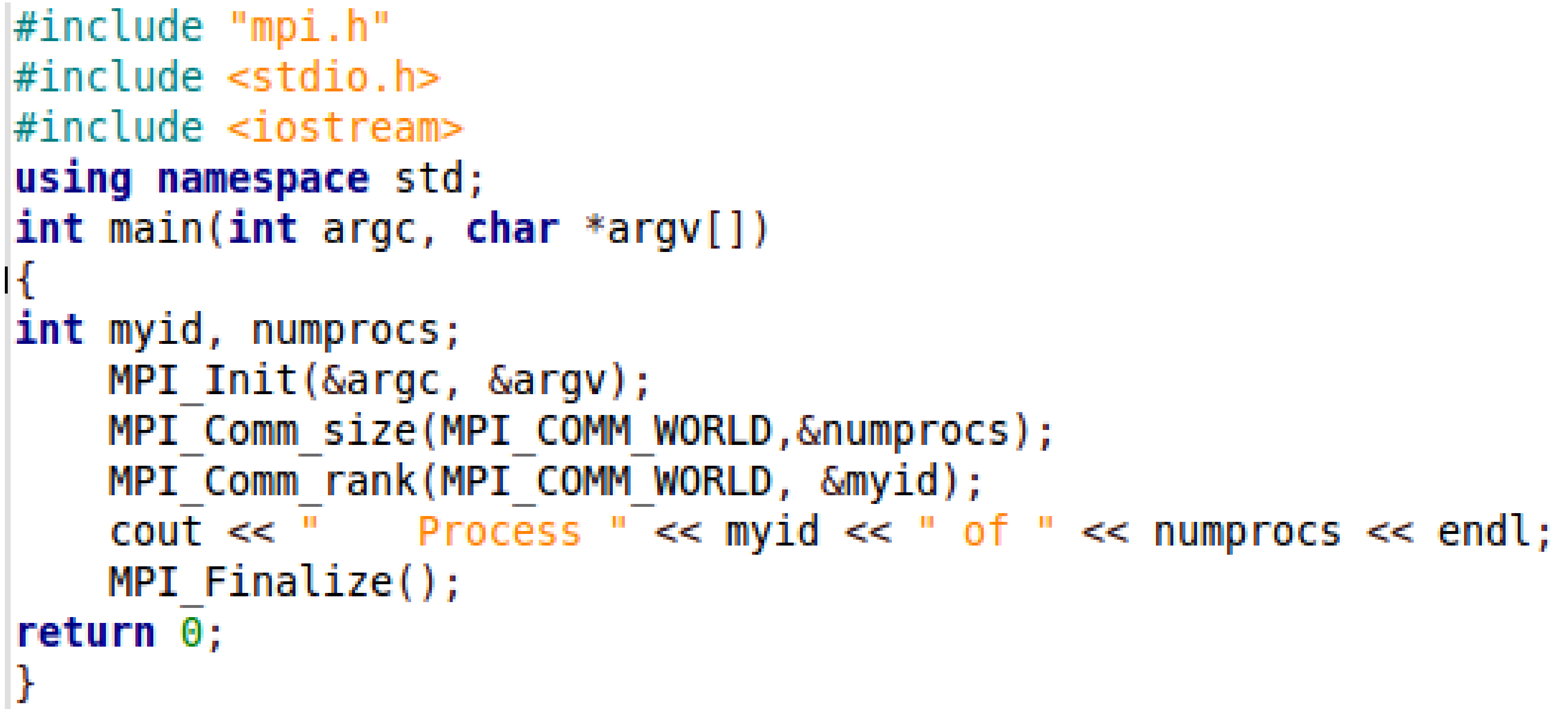
Пример Process N

Каждый запущенный процесс печатает свой уникальный номер в коммуникаторе MPI_COMM_WORLD и число процессов в данном коммуникаторе.

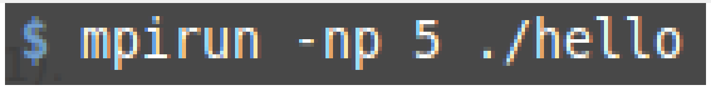
Строка вывода выведена столько раз, сколько процессов порождено при запуске.

Порядок строк не определен.

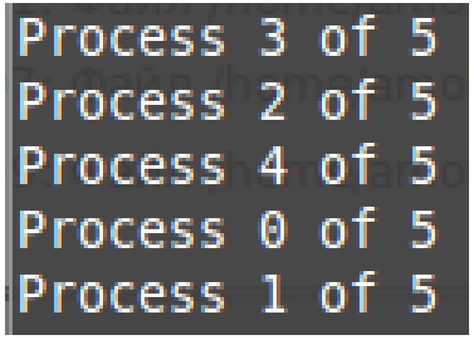
(см. код Си)



```cpp
#include "mpi.h"
#include <stdio.h>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
int myid, numprocs;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    cout << "    Process " << myid << " of " << numprocs << endl;
    MPI_Finalize();
return 0;
}
```



```
$ mpirun -np 5 ./hello
```



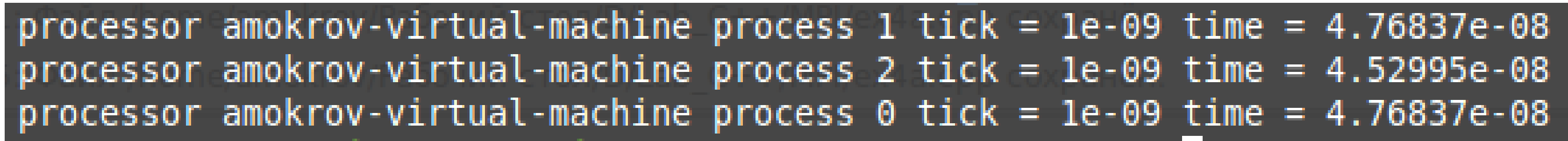
```
Process 3 of 5
Process 2 of 5
Process 4 of 5
Process 0 of 5
Process 1 of 5
```

Пример PROSSESSOR_NAME

```
#define NTIMES 100
int main(int argc, char **argv)
{
  double time_start, time_finish, tick;
  int rank, i; int len;
  char *name;
  name = (char*)malloc(MPI_MAX_PROCESSOR_NAME*sizeof(char));
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(name, &len);
  tick = MPI_Wtick();
  time_start = MPI_Wtime();
  for (i = 0; i<NTIMES; i++)
    time_finish = MPI_Wtime();
  cout << " processor " << name << " process " << rank << " tick = ";
  cout << tick << " time = " << (time_finish-time_start)/NTIMES << endl;
  MPI_Finalize();
}
```

MPI_Wtick() – разрешение таймера на вызвавшем процессоре в сек.

MPI_Get_processor_name (name, &len) – name – имя узла, на котором запущен вызвавший процесс; len – количество символов в имени.



```
processor amokrov-virtual-machine process 1 tick = 1e-09 time = 4.76837e-08
processor amokrov-virtual-machine process 2 tick = 1e-09 time = 4.52995e-08
processor amokrov-virtual-machine process 0 tick = 1e-09 time = 4.76837e-08
```

Передача/прием сообщений

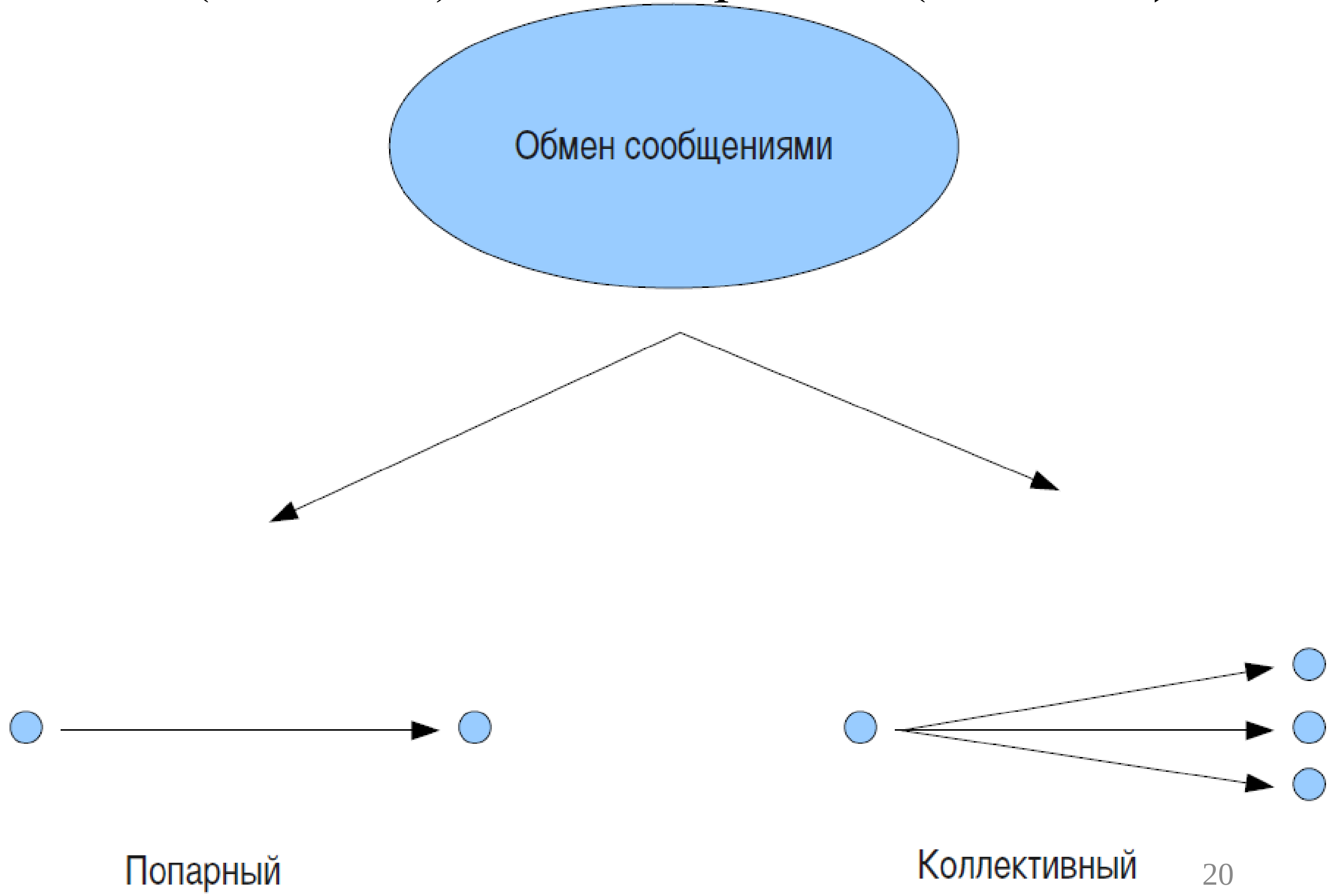
Основная операция в MPI – передача сообщений.
В MPI реализованы практически все основные коммуникационные шаблоны: двухточечные (point-to-point), коллективные (collective) и односторонние (one-sided).

Программы содержат средства порождения и завершения параллельных процессов и средства взаимодействия запущенных процессов между собой.

Процедуры передачи:

Индивидуальные типа точка-точка.

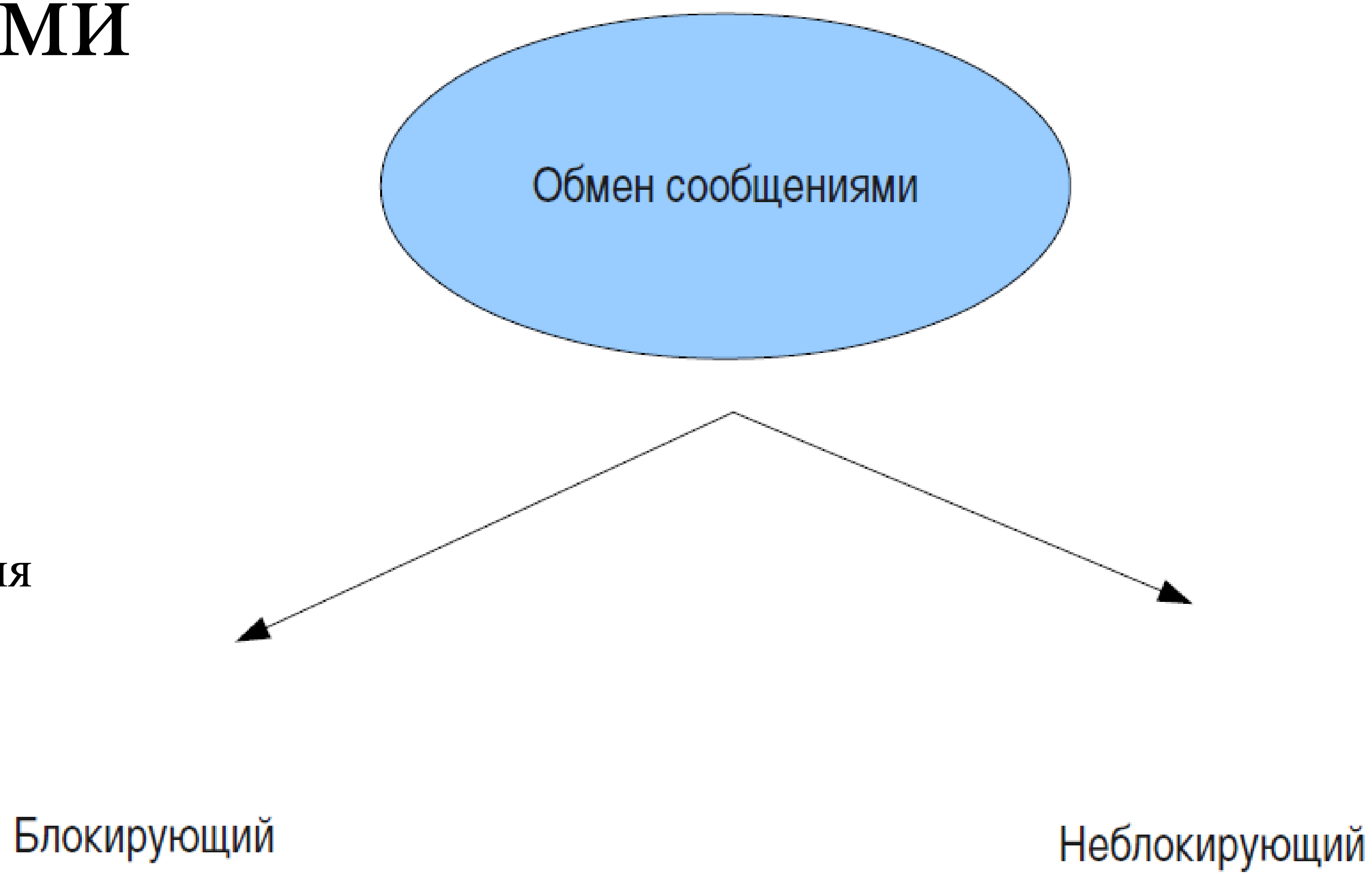
Коллективные – в операцию вовлечены все процессоры коммуникатора.



Обмен сообщениями

Все процедуры делятся на классы: процедуры с блокировкой (с синхронизацией) и без блокировки (асинхронные).

Процедуры обмена с блокировкой приостанавливают работу до выполнения условия. Возврат из асинхронных происходит немедленно после инициализации коммуникационной операции.


Обмен сообщениями

Блокирующий

Неблокирующий

Неаккуратность с блокировкой приводит к тупиковым ситуациям.

Использование асинхронных операций требует аккуратного использования массивов данных.

Суперкомпьютерные дни в России

26-27 сентября 2016 г. Московская международная конференция

Провели Суперкомпьютерный консорциум университетов России и Федеральное агентство научных организаций России.

http://russianscdays.org/

- **Опубликован фотоальбом конференции**
- **Опубликованы презентации некоторых пленарных докладов**
- **На сайте опубликован сборник трудов конференции**
- **Объявлена 23-я редакция списка Top50 самых мощных компьютеров СНГ.**

В 2015 году объединены международная суперкомпьютерная конференция "Научный сервис в сети Интернет", международная конференция "Высокопроизводительные параллельные вычисления на кластерных системах", конференция Russian Supercomputing Conference.

Тематика конференции – все аспекты суперкомпьютерных технологий: разработка аппаратного и программного обеспечения, решение больших задач, использование суперкомпьютерных технологий в промышленности, проблемы экзафлопсных вычислений, суперкомпьютерное образование и многие другие.

Top50

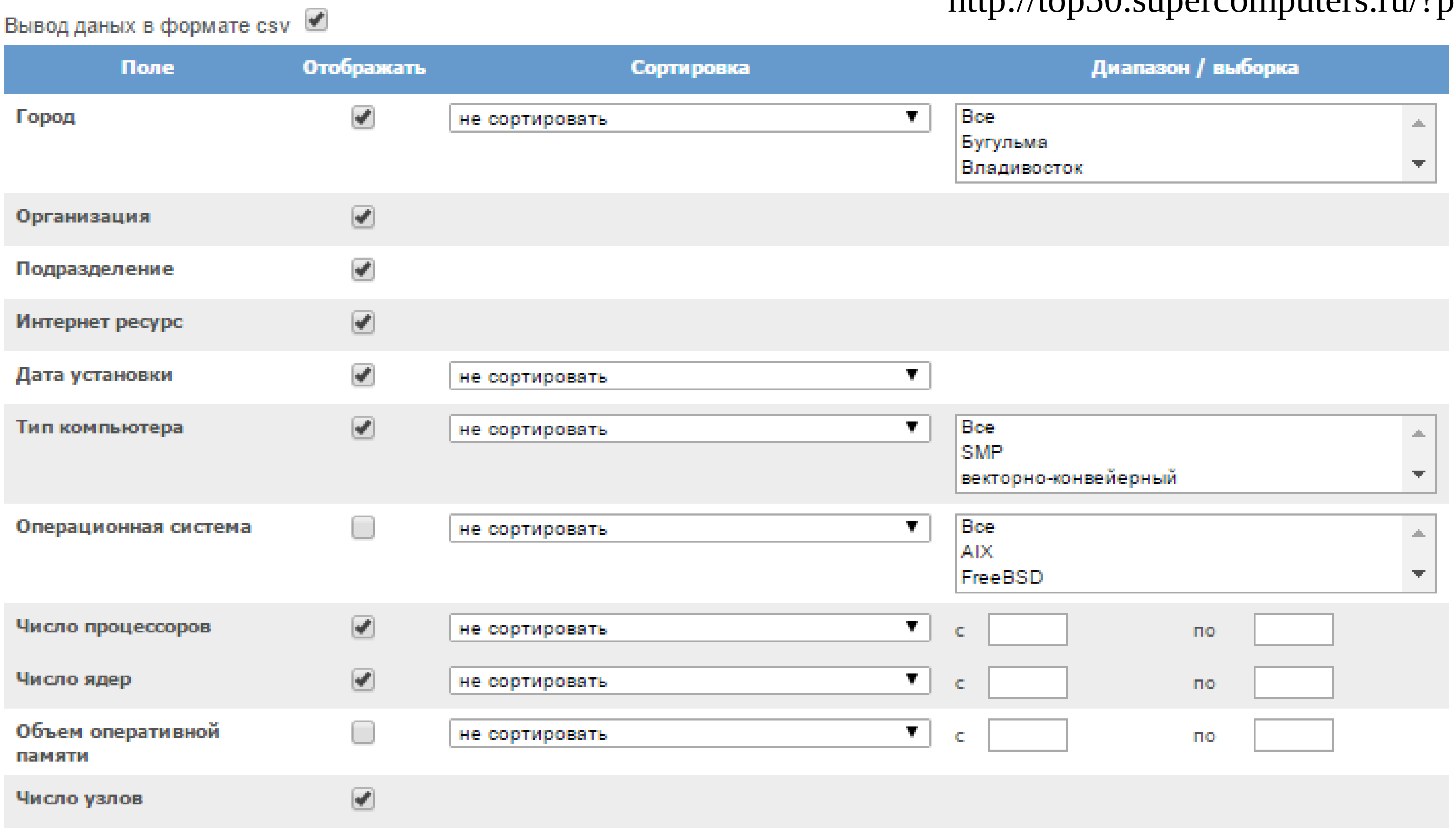
N	Место	Тип	Число CPU	Число ядер	Область применения	Произв-ть Linpack
1 2 29	Московский государственный университет имени М.В. Ломоносова 2014 г.	массивно-параллельный	1280	37120	Наука и образование	1,849.00
3	Санкт-Петербургский политехнический университет" 2014г.	кластер	1424	19936	Наука и образование	658.11
4	Москва Российская академия наук 2012 г.	кластер	416	28704	Наука и образование	375.70
5 6	Москва НИЦ "Курчатовский институт" 2015 г.	кластер	774	11082	Наука и образование	374.13
7	Нижний Новгород Нижегородский государственный университет 2014 г.	кластер	360	30760	Наука и образование	289.50

N	Место	Тип	Число CPU	Число ядер	Область применения	Произв-ть Linpack
8 15	Челябинск Южно-Уральский государственный университет 2013 г.	кластер	768	28032	Наука и образование	288.20
13	Екатеринбург ИММ УрО РАН 2008 г.	кластер	92	6184	Наука и образование	109.88
17	Москва Т-Нано 2013 г.	кластер	640	5564	Коммерческий сектор	93.14
18	Санкт-Петербург Институт прикладной астрономии РАН 2014 г.	кластер	80	1680	Исследования	85.34
22	Долгопрудный Московский физико-технический институт государственный университет 2012 г.	кластер	448	3584	Исследования	70.12

N	Место	Тип	Число CPU	Число ядер	Область применения	Произв-ть Linpack
28, 31, 46, 49	Москва Государственный сектор 2011 г.	кластер	1462	8772	не указано	49.43
32, 33, 38	Москва Банковский сектор 2011 г.	кластер	1320	7920	Финансы	44.63
35, 50	Москва Промышленность 2014 г.	кластер	230	2300	Промышленность	41.63
40	Нижний Новгород ОАО ЦНИИ Буревестник 2012 г.	кластер	512	3520	Промышленность	36.88
44	Нижегородская обл., Дивеевский р-н, пос. Сатис ООО "ЦКО" 2014 г.	кластер	168	1680	Промышленность	35.31
45	Алматы КазНТУ им. К.И.Сатпаева 2014 г.	кластер	44	2904	Наука и образование	34.71
- 46 (2015)	Москва Национальный исследовательский технологический университет "МИСиС" 2014 г.	кластер	200	1600	Наука и образование	28.14

Формирование произвольной таблицы

http://top50.supercomputers.ru/?page=table



Вывод даных в формате csv ☑

Поле	Отображать	Сортировка	Диапазон / выборка
Город	☑	не сортировать ▼	Все Бугульма Владивосток
Организация	☑		
Подразделение	☑		
Интернет ресурс	☑		
Дата установки	☑	не сортировать ▼	
Тип компьютера	☑	не сортировать ▼	Все SMP векторно-конвейерный
Операционная система	☐	не сортировать ▼	Все AIX FreeBSD
Число процессоров	☑	не сортировать ▼	с по
Число ядер	☑	не сортировать ▼	с по
Объем оперативной памяти	☐	не сортировать ▼	с по
Число узлов	☑		