Основы параллельного программирования с использованием MPI

Лекция 1

Немнюгин Сергей Андреевич

Санкт-Петербургский государственный университет

физический факультет

кафедра вычислительной физики

Интернет-Университет
Суперкомпьютерных Технологий

High-Performance Computing University

План лекции

- Обзор курса
- Модели программирования последовательная и параллельная
- Программные инструменты параллельного программирования
- Литература и другие источники информации

Обзор курса

Аннотация

Курс «Основы параллельного программирования с использованием MPI» посвящен основам разработки параллельных программ с использованием интерфейса обмена сообщениями (MPI - Message Passing Interface). В курсе рассматриваются основные понятия и концепции модели передачи сообщений, «архитектура» MPI. Обсуждаются способы организации обменов разного типа: двухточечных, коллективных, блокирующих и неблокирующих, и т. д. Рассматриваются вспомогательные средства, такие как пользовательские типы, виртуальные топологии, другие вопросы. В курсе сочетаются теоретические и практические занятия. Приводятся примеры, предусмотрено выполнение лабораторных работ и домашних заданий, а также тестирование.

Обзор курса

Темы курса:

- краткий обзор моделей параллельного программирования и программных реализаций этих моделей;
- архитектура MPI, привязки к языкам программирования C/C++ и Fortran;
- структура MPI-программы, настройка среды, компиляция и выполнение MPI-программ;
- двухточечные обмены, блокирующие и неблокирующие, буферизованные, двух- и односторонние, и т. д.;
- коллективные обмены – широковещательная рассылка, распределение и сбор данных, операции редукции, синхронизация и т. д.;
- работа с группами процессов, интра- и интеркоммуникаторы;
- пользовательские типы данных;
- виртуальные топологии.

О лекторе

Немнюгин Сергей Андреевич

Кандидат физико-математических наук, доцент кафедры вычислительной физики Санкт-Петербургского государственного университета. Научные интересы связаны с использованием методов статистического моделирования для решения задач квантовой физики, моделированием процессов распространения заряженных частиц и жесткого электромагнитного излучения в веществе, математическим моделированием в экономике, а также применением методов высокопроизводительных и распределенных вычислений для решения задач вычислительной физики. Активно занимается педагогической деятельностью в области вычислительной физики и высокопроизводительных вычислений.

Электронная почта:
parallel-g112@yandex.ru

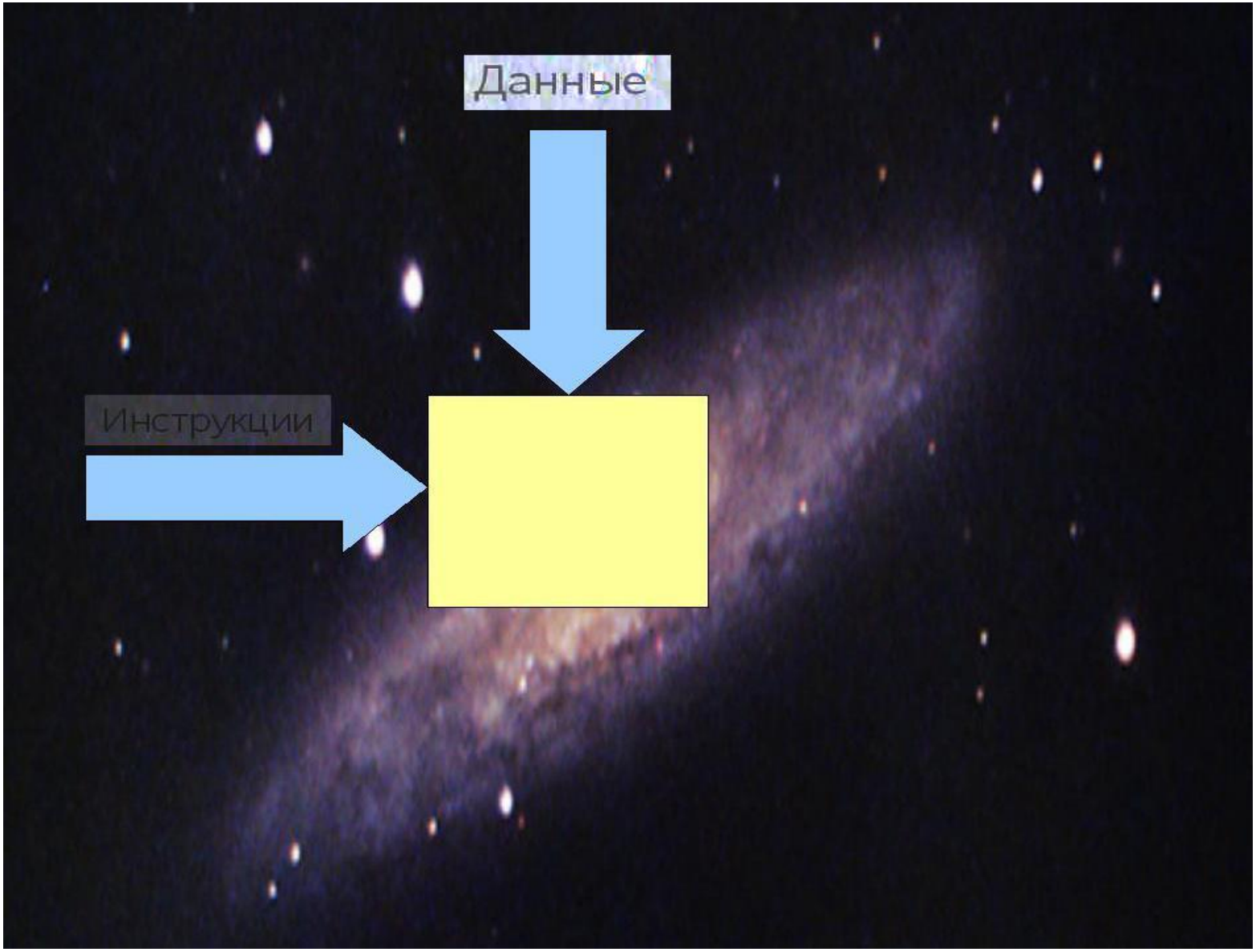
Лекция 1

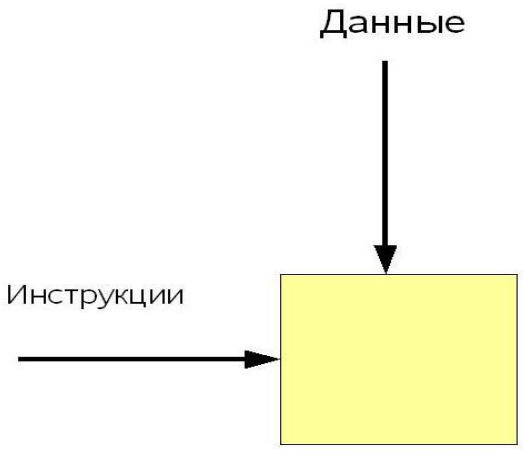
Аннотация

В первой лекции обсуждаются последовательная и параллельная модели программирования. Рассматриваются различные парадигмы параллельного программирования и их программные реализации (POSIX Threads, Microsoft Windows API, OpenMP, MPI, PVM и т. д.).

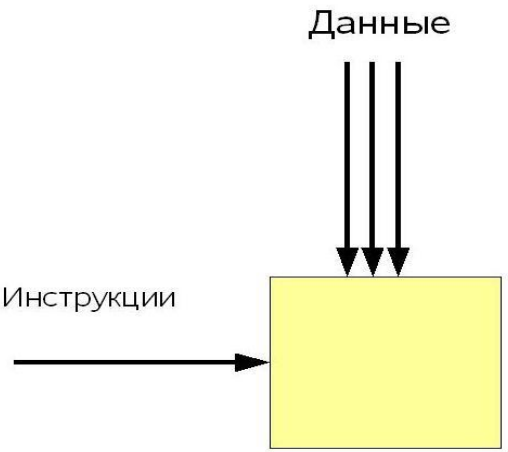
Последовательная и параллельная модели программирования

Идея Флинна

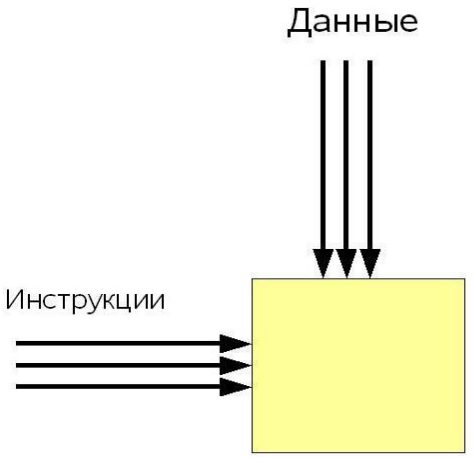




SISD-архитектура

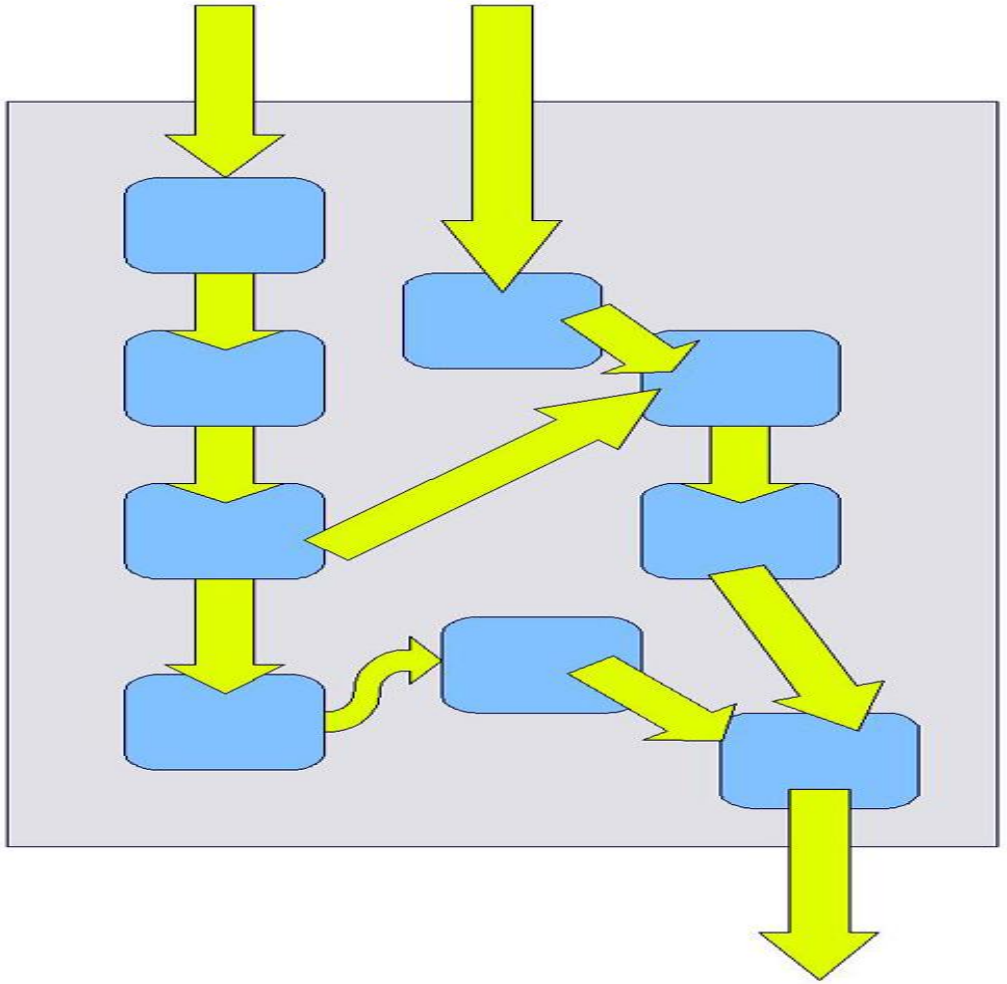


SIMD-архитектура

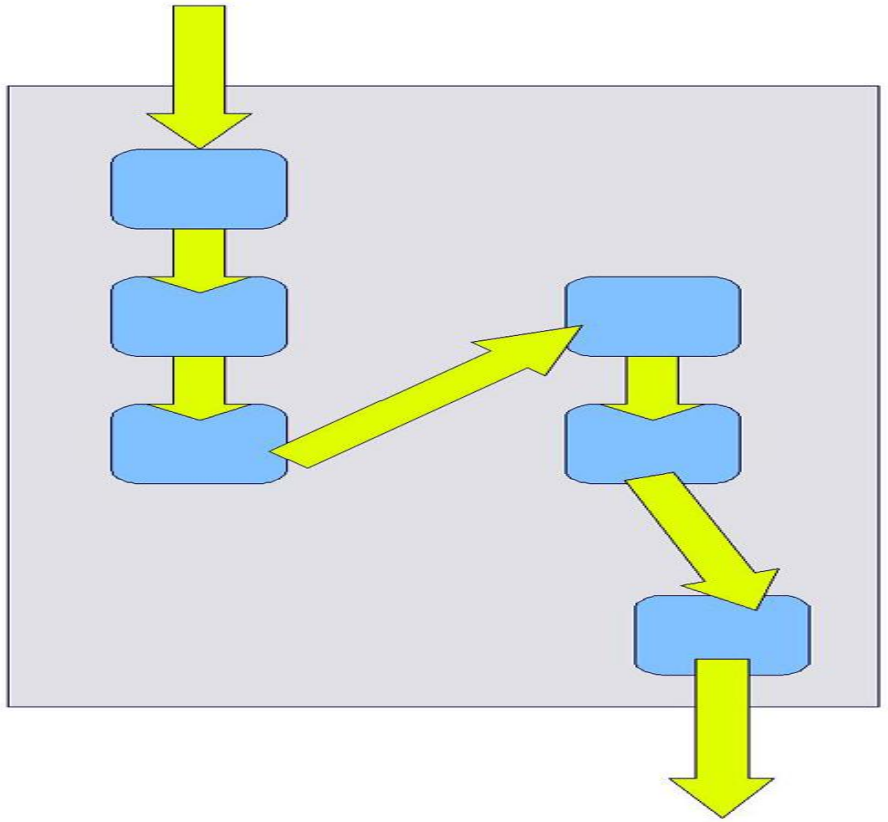


MIMD-архитектура

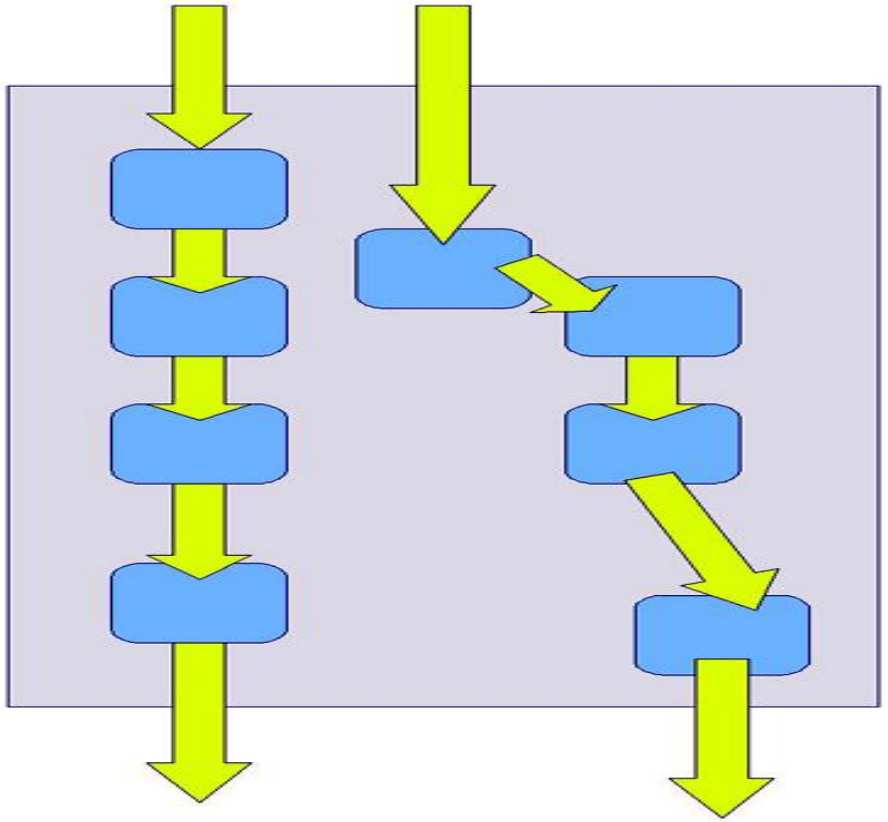
Что происходит с данными внутри программы



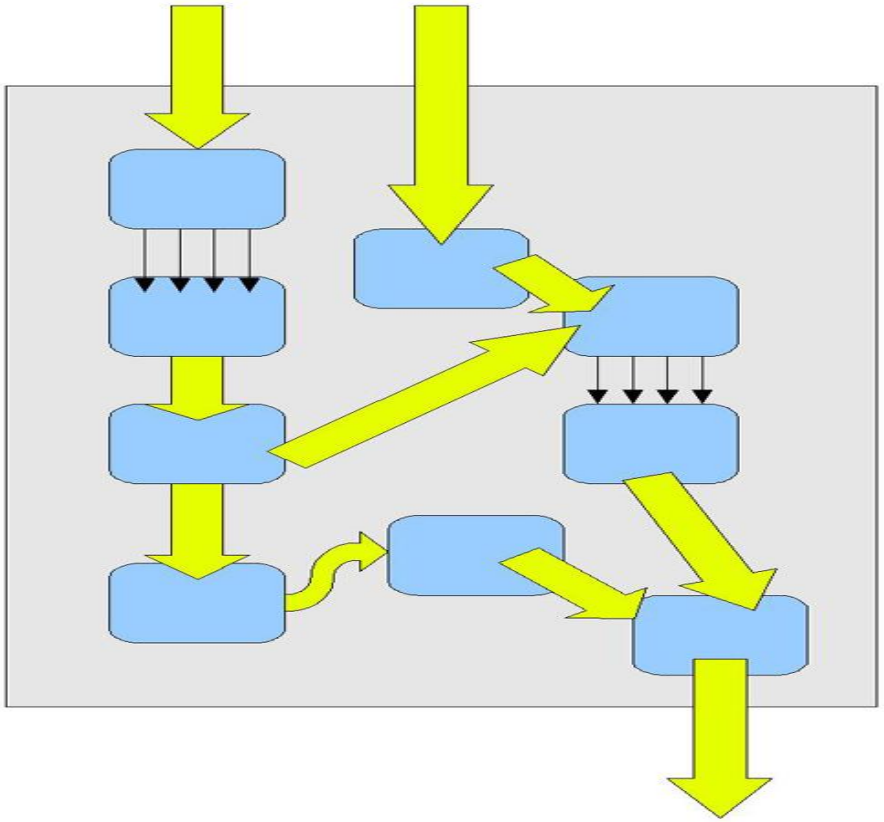
Последовательная программа



Параллельная программа



Общий случай



Последовательный+распараллеливаемый компоненты

Последовательная модель программирования

Последовательную модель программирования характеризуют:

- невысокая производительность;
- применение стандартных языков программирования;
- хорошая переносимость.

Параллельная модель программирования

Параллельную модель программирования характеризуют:

- возможность добиться более высокой производительности;
- применение специальных приемов и инструментов программирования;
- повышенная трудоемкость программирования;
- проблемы с переносимостью программ.

Требования к параллельным программам:

- достаточная степень параллелизма;
- хорошая масштабируемость;
- детерминизм;
- эффективность.

Параллельная модель программирования

Повышенная трудоёмкость параллельного программирования связана с тем, что программист должен заботиться:

- об управлении работой множества процессов;
- об организации межпроцессных пересылок данных;
- о вероятности тупиковых ситуаций (взаимных блокировках);
- о нелокальном и динамическом характере ошибок;
- о возможной утрате детерминизма («гонки за данными»);
- о масштабируемости;
- о сбалансированной загрузке вычислительных узлов.

Параллельная модель программирования

Две парадигмы параллельного программирования

1. параллелизм данных;
2. параллелизм задач.

Параллельная модель программирования

Параллелизм данных

Подход, основанный на параллелизме данных, характеризуется тем, что:

- одна операция применяется сразу к нескольким элементам массива данных. Различные фрагменты такого массива обрабатываются на векторном процессоре или на разных процессорах параллельной машины;
- обработкой данных управляет одна программа;
- пространство имен является глобальным;
- параллельные операции над элементами массива выполняются одновременно на всех доступных данной программе процессорах.

Параллельная модель программирования

Параллелизм данных

От программиста требуется:

- задание опций векторной или параллельной оптимизации транслятору;
- задание директив параллельной компиляции;
- использование специализированных языков параллельных вычислений, а также библиотек подпрограмм, специально разработанных с учетом конкретной архитектуры компьютера и оптимизированных для этой архитектуры.

Параллельная модель программирования

Параллелизм задач

Параллелизм данных характеризуется тем, что:

- вычислительная задача разбивается на несколько относительно самостоятельных подзадач. Каждая подзадача выполняется на своем процессоре (ориентация на архитектуру MIMD);
- для каждой подзадачи пишется своя собственная программа на обычном языке программирования (чаще всего это Fortran или C);
- подзадачи должны обмениваться результатами своей работы, получать исходные данные. Практически такой обмен осуществляется вызовом процедур специализированной библиотеки. Программист при этом может контролировать распределение данных между различными процессорами и различными подзадачами, а также обмен данными.

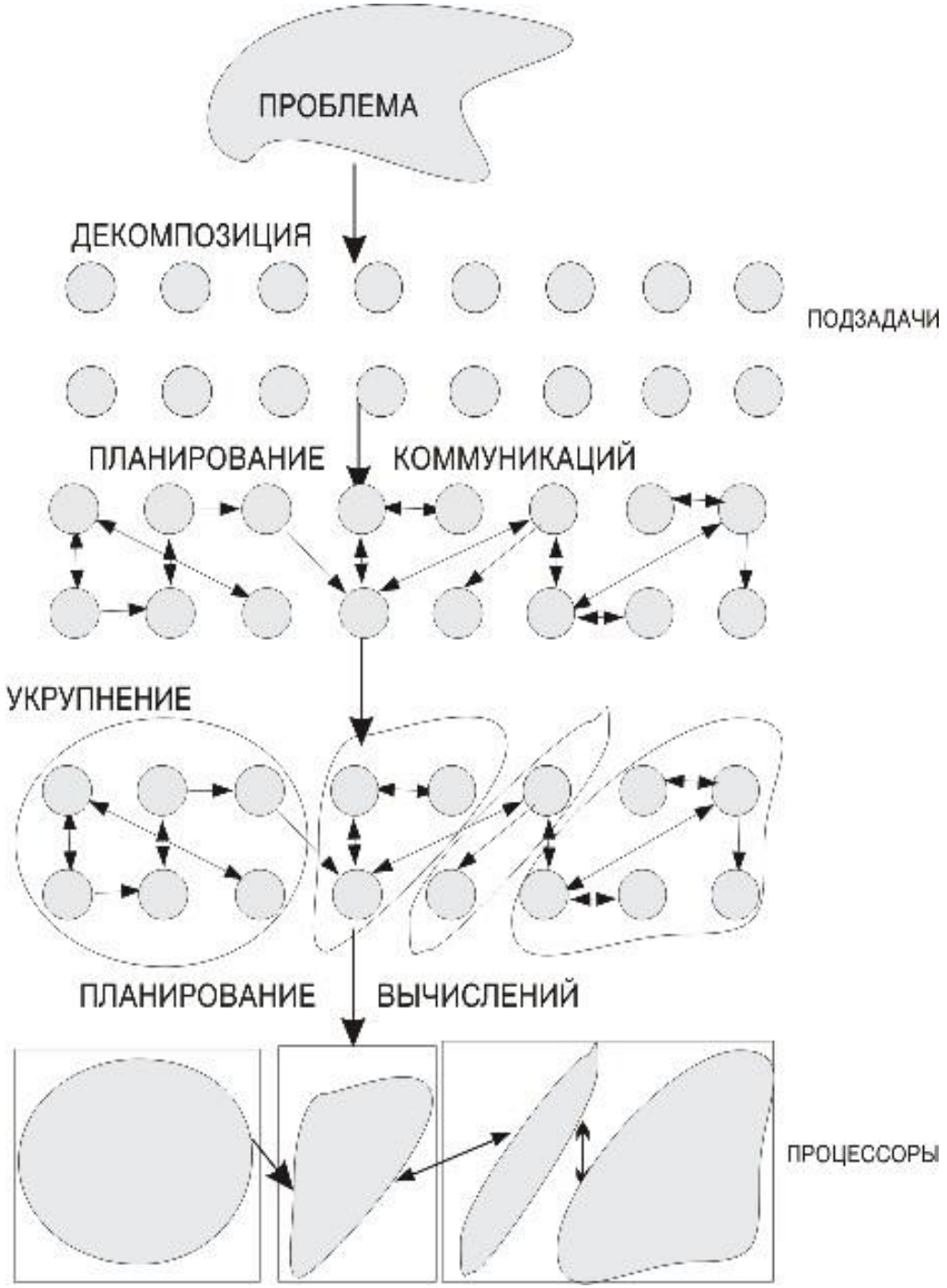
Параллельная модель программирования

Параллелизм задач

Достоинства подхода:

- большая гибкость и большая свобода, предоставляемая программисту в разработке программы, эффективно использующей ресурсы параллельного компьютера;
- возможность достижения максимального быстродействия.

Разработка параллельной программы/алгоритма


ПРОБЛЕМА
ДЕКОМПОЗИЦИЯ
ПОДЗАДАЧИ
ПЛАНИРОВАНИЕ КОММУНИКАЦИЙ
УКРУПНЕНИЕ
ПЛАНИРОВАНИЕ ВЫЧИСЛЕНИЙ
ПРОЦЕССОРЫ

Разработка параллельной программы/алгоритма

Декомпозиция

На этом этапе выполняются анализ, оценка возможности распараллеливания. Задача и связанные с ней данные разделяются на более мелкие части подзадачи и фрагменты структур данных. Особенности архитектуры конкретной вычислительной системы на данном этапе могут не учитываться.

Требования

- количество подзадач после декомпозиции должно быть достаточно большим;
- следует избегать лишних вычислений и пересылок данных;
- подзадачи должны быть примерно одинакового размера.

Разработка параллельной программы/алгоритма

Декомпозиция

Независимость

- ***независимость по данным -*** данные, обрабатываемые одной частью
  программы, не модифицируются другой ее частью;
- ***независимость по управлению -*** порядок выполнения частей программы может быть определен только во время выполнения программы (наличие зависимости по управлению предопределяет последовательность выполнения);
- ***независимость по ресурсам -*** обеспечивается достаточным количеством компьютерных ресурсов (объемом памяти, количеством функциональных узлов и т. д.);
- ***зависимость по выводу -*** возникает, если две подзадачи производят
  запись в одну и ту же переменную, а зависимость по вводу-выводу, если операторы ввода/вывода двух или нескольких подзадач обращаются к одному файлу (или переменной).

Разработка параллельной программы/алгоритма

Проектирование коммуникаций

Типы коммуникаций:

- ***локальные*** - каждая подзадача связана с небольшим набором других подзадач;
- ***глобальные*** - каждая подзадача связана с большим числом других подзадач;
- ***структурированные*** - каждая подзадача и подзадачи, связанные с ней, образуют регулярную структуру (например, с топологией решетки);
- ***неструктурированные*** - подзадачи связаны произвольным графом;
- ***синхронные*** - отправитель и получатель данных координируют обмен;
- ***асинхронные*** - обмен данными не координируется.

Разработка параллельной программы/алгоритма

Проектирование коммуникаций

Рекомендации по проектированию коммуникаций:

- количество коммуникаций у подзадач должно быть примерно одинаковым, иначе приложение становится плохо масштабируемым;
- там, где это возможно, следует использовать локальные коммуникации.

Разработка параллельной программы/алгоритма

Укрупнение

На этапе укрупнения учитывается архитектура вычислительной системы, при этом часто приходится объединять (укрупнять) задачи, полученные на первых двух этапах, для того чтобы их число соответствовало числу процессоров.

Требования

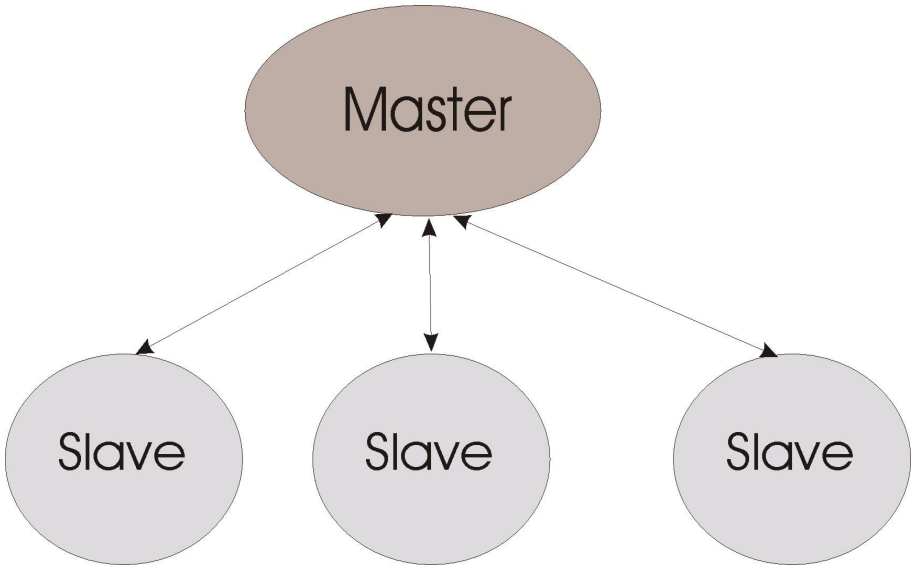
- снижение затрат на коммуникации;
- если при укрупнении приходится дублировать вычисления или данные, это не должно приводить к потере производительности и масштабируемости программы;
- результирующие (под)задачи должны иметь примерно одинаковую трудоемкость;
- сохранение масштабируемости.

Разработка параллельной программы/алгоритма

Организация параллельной программы

Простая схема хозяин/работник (Master/slave)

- *главная задача* отвечает за размещение подчиненных задач;
- *подчиненная задача* получает исходные данные для обработки от главной задачи и возвращает ей результат работы.



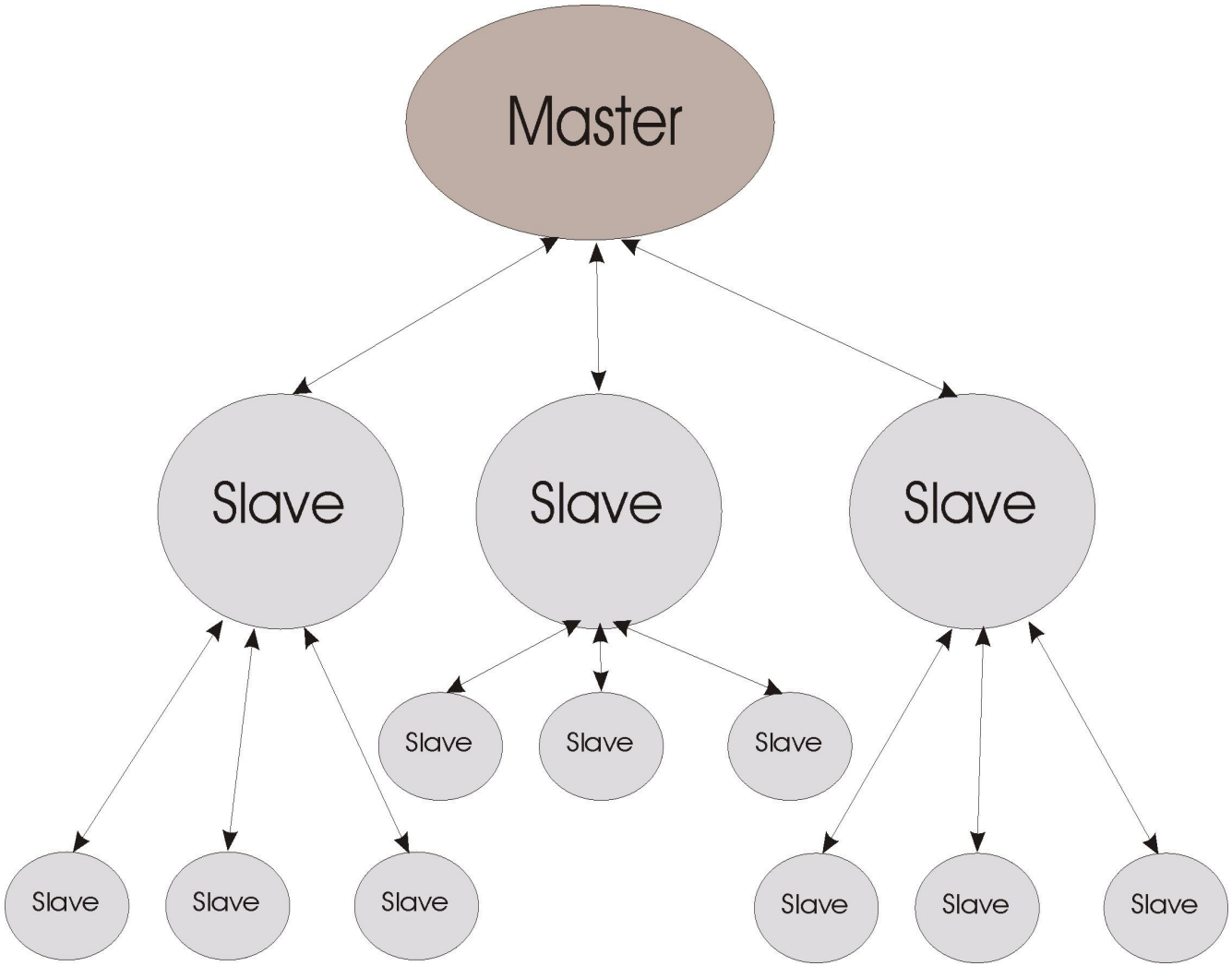
Разработка параллельной программы/алгоритма

Организация параллельной программы

Иерархическая схема хозяин/работник (Master/slave)

Подчиненные задачи разделены на непересекающиеся подмножества и у каждого из этих подмножеств есть своя главная задача. *Главные задачи* подмножеств управляются одной "самой главной" задачей.

Разработка параллельной программы/алгоритма



Многопоточные программы

Поток (нить) представляет собой последовательный поток управления (последовательность команд) в рамках одной программы. При создании процесса порождается главный поток, выполняющий инициализацию процесса. Он же начинает выполнение команд.

Поток и процесс соотносятся следующим образом

- процесс имеет главный поток, инициализирующий выполнение команд процесса;
- любой поток может порождать в рамках одного процесса другие потоки;
- каждый поток имеет собственный стек;
- потоки, соответствующие одному процессу, имеют общие сегменты кода и данных.

Многопоточные программы

При разработке многопоточных приложений возникают следующие проблемы:

- гонки за данными;
- блокировки;
- несбалансированность загрузки.

Гонки за данными (data races) - являются следствием зависимостей, когда несколько потоков модифицируют содержимое одной и той же области памяти. Наличие гонок за данными не всегда является очевидным.

Многопоточные программы

Блокировка возникает, если поток ожидает выполнение условия, которое не может быть выполнено. Обычно возникновение тупиковой
ситуации является следствием конкуренции потоков за ресурс, который удерживается одним из них.

Условия возникновения блокировки:

- доступ к ресурсу эксклюзивен (возможен только одним потоком);
- поток может удерживать ресурс, запрашивая другой;
- ни один из конкурирующих потоков не может освободить запрашиваемый ресурс

Программные инструменты параллелизма

Open Multi-Processing (OpenMP)

OpenMP - стандарт программного интерфейса приложений для параллельных систем с общей памятью. Поддерживает языки C, C++, Fortran.

Официальный сайт OpenMP
http://openmp.org

Open Multi-Processing (OpenMP)

Корпорация Intel предлагает **Intel ® Cluster OpenMP** кластерный вариант OpenMP, предназначенный для программирования для систем с распределенной памятью

Официальный сайт
http://www.intel.com

POSIX Threads

POSIX Threads - стандарт **POSIX** реализации потоков (нитей) выполнения, определяющий API для создания и управления ими.

Ссылка
http://www.opengroup.org/onlinepubs/009695399/basedefs/pthread.h.html

Windows API

В Microsoft Windows имеется возможность разработки многопоточных приложений на C++.

Ссылка
http://msdn.microsoft.com

Parallel Virtual Machine (PVM)

PVM (Parallel Virtual Machine) - позволяет объединить разнородный набор компьютеров, связанных сетью, в общий вычислительный ресурс, который называют Параллельной Виртуальной Машиной.

Официальный сайт PVM
http://www.csm.ornl.gov/pvm

Message Passing Interface (MPI)

Message Passing Interface (MPI) - *Интерфейс Передачи Сообщений*, спецификация разработанная в 1993—1994 годах группой MPI Forum, в состав которой входили представители академических и промышленных кругов. Она стала первым стандартом систем передачи сообщений.

Официальный сайт MPI
http://www.mpi-forum.org

Message Passing Interface (MPI)

MPICHameleon (**MPICH**) является свободно распространяемой реализацией MPI.

Официальный сайт MPICH
http://www-unix.mcs.anl.gov/mpi/mpich1

Заключение

В этой лекции мы рассмотрели:

- особенности последовательной и параллельной моделей программирования;
- наиболее известные инструменты параллельного программирования.

Вопросы для обсуждения

- В чём вы видите ограничения последовательной модели программирования?
- Какие достоинства и недостатки у подходов, основанных на параллелизме данных и параллелизме задач?
- В чём достоинства и недостатки различных видов коммуникаций?

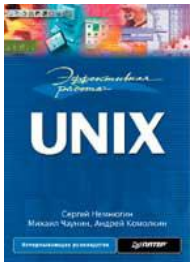
Темы для самостоятельной работы

- Следуя приведённым в лекции ссылкам, познакомьтесь с основными особенностями наиболее известных инструментов разработки параллельных и многопоточных программ.
- Составьте, по возможности, наиболее полный список реализаций интерфейса MPI.

Литература

С.Немнюгин, О.Стесик. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002.

В.П.Гергель. Теория и практика параллельных вычислений. Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007 г., 424 стр.



С.Немнюгин, М.Чаунин, А.Комолкин. Эффективная работа:UNIX. – СПб.: Питер, 2001.

Литература

С.Немнюгин, О.Стесик. Современный Фортран. Самоучитель. – СПб.: БХВ-Петербург, 2004.

В.В.Воеводин, Вл.В. Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.

Портал PARALLEL.RU

Материалы к курсу «Средства программирования для многопроцессорных и многоядерных вычислительных систем». Сайт физического факультета СПбГУ (раздел «Библиотека»):
http://www.phys.spbu.ru

Тема следующей лекции

Архитектура MPI