Технологии решения сложных задач на кластерных вычислительных системах с использованием средств MPI

Программа обучения

Решение инженерных задач на вычислительных системах

- Основные понятия вычислительной математики
- Основные концепции и инструменты программирования
- Введение в C/C++

Параллельные технологии решения сложных задач

- Понятие параллелизма в программировании и формы его представления.

Использование MPI

- Стандарт MPI и его реализации.
- Компиляция и запуск MPI программ в Linux и Windows.
- Синхронные и асинхронные операции обмена сообщениями MPI.
- Коллективные операции MPI обмена сообщениями.
- Определяемые пользователем типы данных.
- Использование топологий в MPI программах.
- Многопоточное программирование в MPI

Вычислительные задачи и их реализация на MPI

Отладка, оптимизация MPI программ

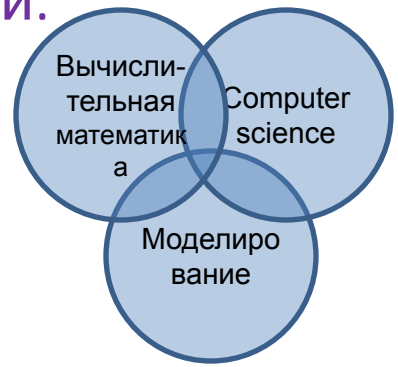
- Вспомогательные инструменты при работе с MPI
  - Jump shot
  - Intel Trace Analyzer
- Общие подходы к оптимизации программ.

Введение: Моделирование как методология

- Подход «cut-and-try» устарел
- Современная наука:

1.Построить модель (выбрать представление)
2. Разработать алгоритм
3. Эффективно выполнить вычисления
4. Оценить погрешность
5. Выполнить визуализацию

- Появляются дисциплины, продвижение в которых возможно только лишь с помощью моделирования и использования масштабных вычислений:
  - Нанотехнологии
  - Микробиология: поиск новых лекарств
  - Науки о материалах

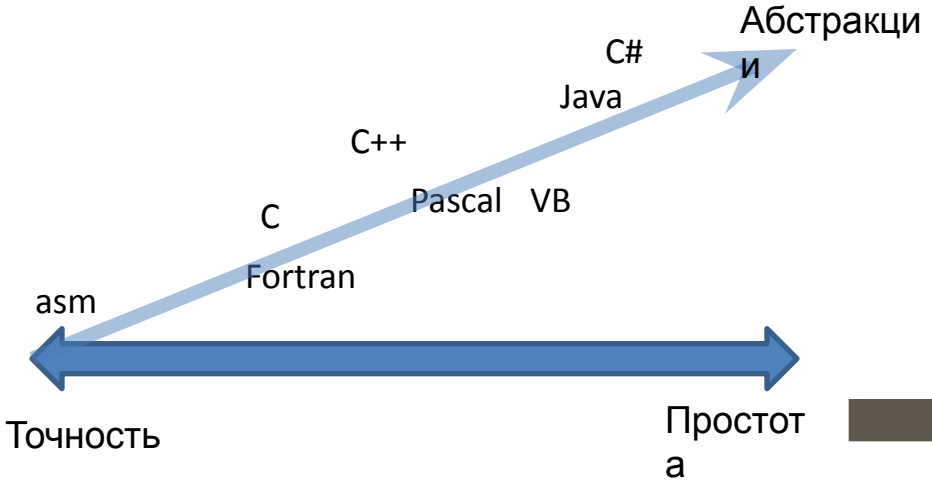


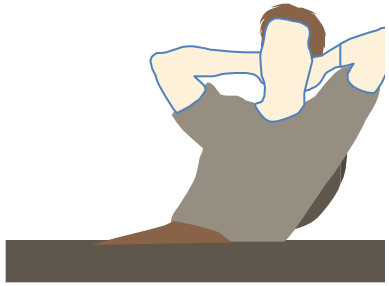
Введение: Языки программирования

- Сила 0 и 1

c = a + b

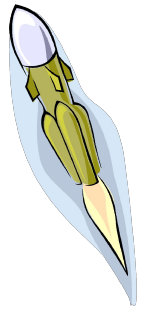
0101100101000100011101010100010
0101100101110010001010010100001
1
0011101010100010011101010010010
1
0101110101010101010100000111101





Основные понятия вычислительной математики

- Вектор $a^T=(a_1,a_2,\ldots,a_n)$
- Матрица A (m x n), A[i,j]
- Функция
- O(f(n))



Округление!

Системы счисления

$126_{10} = 1 \times 10^2 + 2 \times 10^1 + 6 \times 10^0$

Двоичная система:

$01111110_2 = 0\times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1\times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

Числа с плавающей точкой:

$+ \quad .126 \quad \times \quad 10^3$

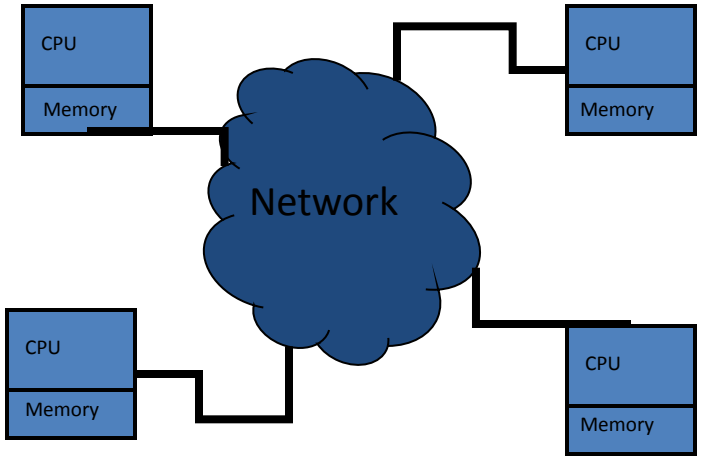
знак мантисса показатель

Параллелизм в задачах

Четыре шага при создании параллельных программ:

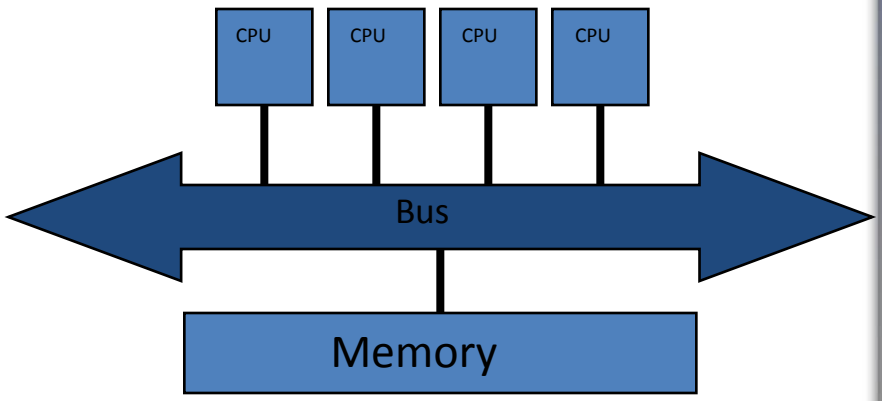
1. Разделение
   - Разделение данных и задач на фрагменты
2. Взаимодействие
   - Распределение данных между задачами и установление зависимостей
3. Агломерация
   - Задачи группируются для повышения эффективности использования ресурсов
4. Распределение
   - Распределение задач по «вычислителям»

«Designing and Building Parallel Programs», Ян Фостер



Модель передачи сообщений

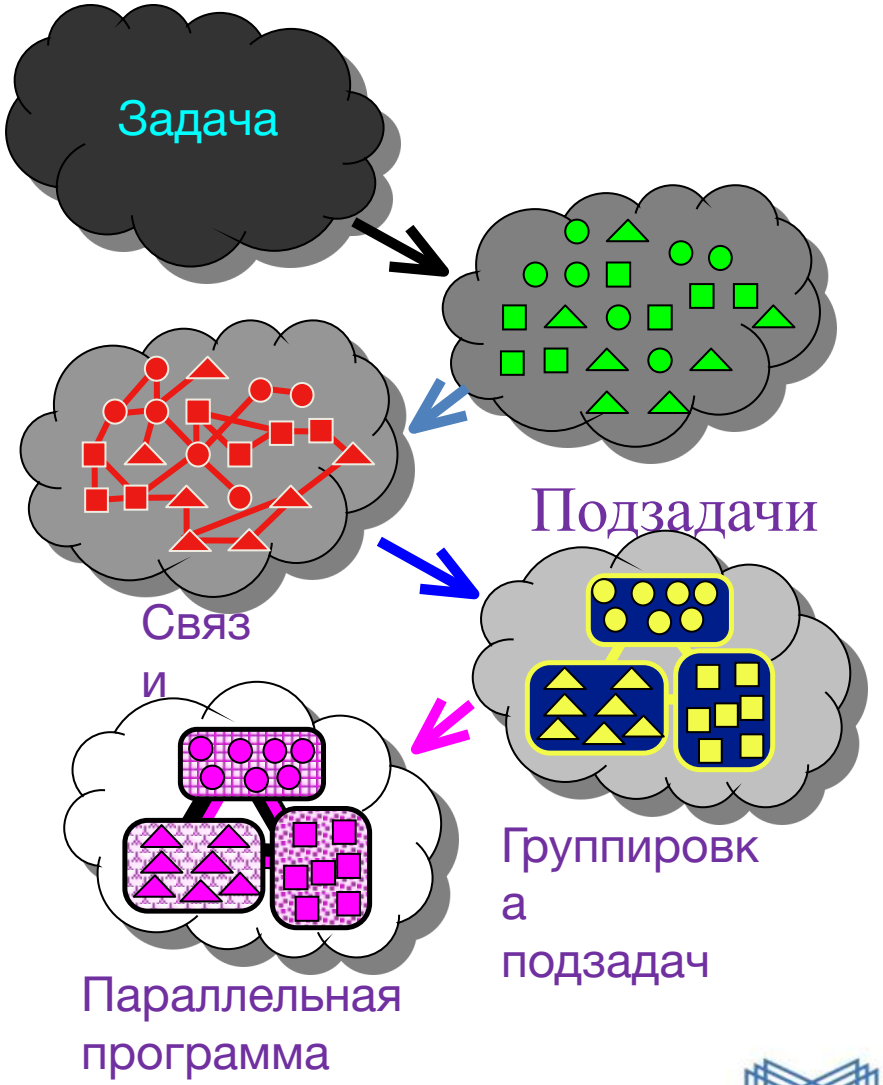
- Множество процессов
- Распределение данных с помощью посылки сообщений (MPI)



Потоки (Threads)

- Один процесс
  - Конкурентное выполнение
- Разделяемые данные и ресурсы (Потоки ОС, OpenMP)

- Разделение
  - Разделить задачу на фрагменты (подзадачи)
- Взаимодействие
  - Определить тип и способ взаимодействия
- Агломерация
  - Группировать фрагменты задачи
- Распределение
  - Распределить группы задач по процессорам



Разделение.

Выделить как можно больше параллельных фрагментов

- Независимые вычисления и/или данные
- Количество элементарных действий должно быть максимальным

Методы декомпозиции:

- Функциональная декомпозиция
- Декомпозиция по данным

Где нет параллелизма?

Подпрограммы, обрабатывающие объекты, имеющие «состояние»

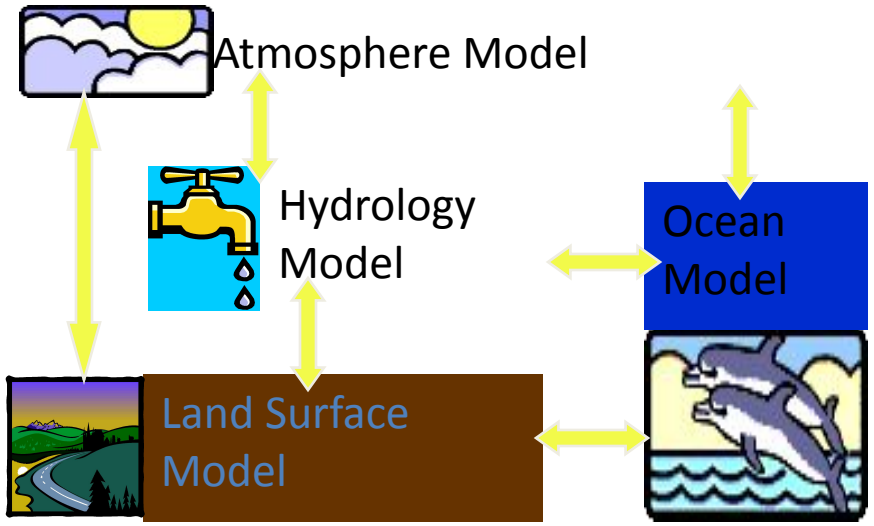
- Выделение памяти
- Ввод-вывод из файлов

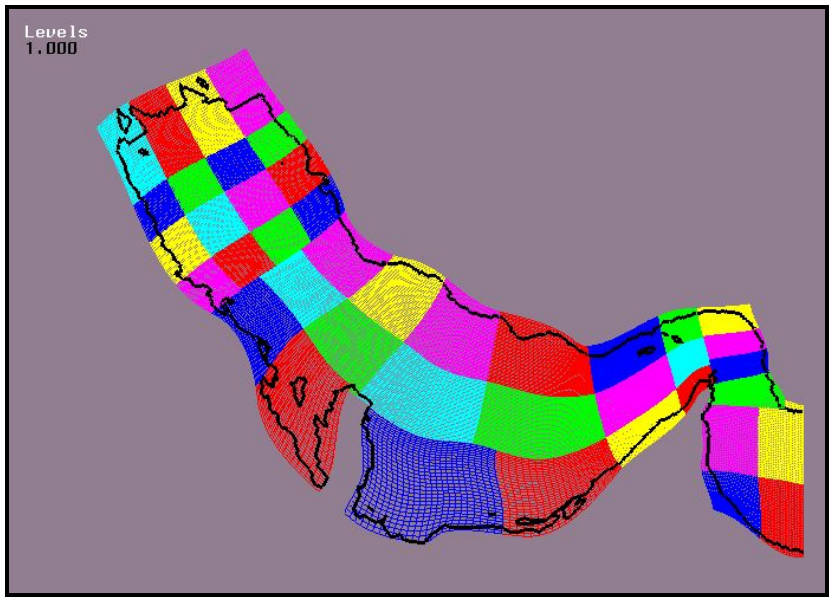
Циклы с зависимостями по данным

- Переменная записывается в одной итерации, а считывается в другой – простой проверкой может быть инвертирование цикла

Функциональная декомпозиция

–Анализируется процесс вычислений





- Декомпозиция по данным
  - Анализируются массивные или часто используемые данные
  - Одна и та же операция применяется ко всем данным

Взаимодействие

Определить тип и способ взаимодействия подзадач, определить, какие данные должны быть разделяемыми.

Точка-точка
- одна подзадача передает данные другой

Коллективное использование
- группа подзадач работает с одной копией данных

Агломерация

Группировка элементарных действий производится для того, чтобы:
- повысить производительность и зернистость
- локализовать связи
- сохранить масштабируемость
- упростить программирование

ГОНКИ ДАННЫХ!

Обычно проявляют себя в случаях, когда множество параллельно выполняющихся задач обращаются к общим данным.

Порядок выполнения мог подразумеваться программистом, но не гарантируется в ходе вычислений.

Сложно предугадать

- Недетерминированное выполнение
- В процессе отладки могут не проявляться

Ускорение

Коэффициент ускорения при параллельном выполнении программ при ее выполнении на N процессорах определяется следующим образом:

$$k = \frac{1}{\alpha + \frac{1-\alpha}{N}}$$

В этой формуле

α – участок программы, выполняющийся полностью последовательно

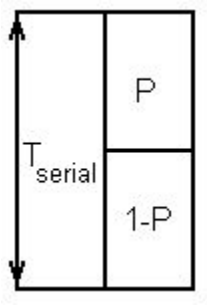
1 – α - полностью параллельный участок программы

N – количество «вычислителей»

Ясно, что при $N \to \infty$, $K \to 1 / \alpha$

Закон Амдала

Также часто используется другая формулировка коэффициента ускорения в форме закона Амдала:



T – время выполнения последовательного варианта программы,

P – участок программы, который можно распараллелить,

1 – P – участок программы, который распараллелить нельзя,

N – количество «вычислителей»

$$T_{parallel} = \left\{(1-P) + \frac{P}{N}\right\} T_{serial}$$

И коэффициент ускорения будет определяться следующим образом:

$$k = \frac{T_{serial}}{T_{parallel}}$$