

Билет 36. Методы организация параллельных вычислений при суперкомпьютерном решении сеточных задач

Топология - это механизм сопоставления процессами некоторого коммуникатора альтернативной схемы адресации. В *MPI* топологии виртуальны. Может использоваться системой для оптимизации распределения процессов по физическим процессорам.

Декартова топология - это:

- логическая топология определена многомерной решеткой;
- обобщение линейной и матричной топологий на произвольное число измерений.

Пример. Рассмотрим конкретную задачу: уравнение Лапласа на прямоугольнике.

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \\ x \in [0, 1], \\ u(x, 0) = \sin(\Pi x), \\ u(x, 1) = \sin(\Pi x)e^{-x}, \\ u(0, y) = u(1, y) = 0. \end{cases}$$

Рассмотрим 5-точечный шаблон:

$$u_{i,j}^{n+1} \cong \frac{u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n}{4}.$$

Создается декартова топология (механизм сопоставления процессами некоторого коммуникатора альтернативной схемы адресации; виртуальный). Необходим обмен данными на границах, будем использовать следующие функции *MPI*.

Основные функции для реализации:

1. Создание топологии: *int MPI_Cart_create*;
2. Определение декартовых координат процесса по его рангу *int MPI_Cart_coords*;
3. Определение ранга процесса по его декартовым координатам: *int MPI_Cart_rank*;

4. Разбиение решетки на подрешетки меньшей размерности: *int MPI_Cart_sub*;
5. Поддержку процедуры последовательной передачи данных по одному из измерений решетки: *int MPI_Cart_shift*;
6. Получение информации о топологии по коммуникатору и числу измерений: *MPI_Cart_get*.

Блокирующие передачи:

1. *int MPI_Send*;
2. *int MPI_Recv*.

Коллективные операции:

1. *int MPI_Bcast* - рассылка сообщения от процесса source всем процессам, включая рассылающий процесс;
2. *int MPI_Gather* - сборка данных со всех процессов группы в один массив;
3. *int MPI_AllReduce* - выполнение глобальных операций с возвратом результатов во всех процессах в одном буфере;
4. *int MPI_Reduce* - Функция аналогична предыдущей, но результат будет записан в буфер только у одного процесса.