Опыт создания вычислительного метакластера на базе кластерных систем Томского научного центра[1]

А. В. Старченко, Г. М. Ибраев
Томский государственный университет, Томск

В настоящее время основное направление развития высокопроизводительных вычислительных ресурсов связано с разработкой технологий, позволяющих объединять возможности вычислительных комплексов, принадлежащих разным организациям, в том числе, территориально удаленным друг от друга. За этим направлением закрепилось название «Grid-технология». Под Grid-технологией понимают инфраструктуру, делающую возможным совместное использование высокопроизводительных компьютеров, сетей, баз данных и исследовательских систем, принадлежащих различным организациям. Grid-приложения часто предназначены для обработки очень больших распределенных объемов данных, которые должны перемещаться между компонентами Grid-инфраструктуры, что требует обеспечения определенных мер защиты этих данных, что не всегда легко достижимо при использовании распространенных сегодня протоколов сети Интернет [1].

Решение проблем, связанных с использованием Grid-технологий, является предметом нескольких исследовательских проектов, выполняющихся в различных организациях. Одним из наиболее известных является проект Globus Allians [2]. В рамках этого проекта ведется разработка сетевых протоколов и приложений, предназначенных для реализации Grid-технологии в сфере обеспечения вычислительных потребностей фундаментальных и прикладных научных исследований. Работы по данному проекту выполняются совместно в целом ряде организаций США и Западной Европы. Крупными корпоративными партнерами являются компании IBM и Microsoft. Основное внимание разработчиками проекта уделяется вопросам управления вычислительными ресурсами, механизмам доступа и управления данными, созданию средств разработки Grid-приложений,

[1] Работа выполнена при поддержке РФФИ, грант № 07-05-01126, и программы «СКИФ-ГРИД».

информационному обеспечению и безопасности. В результате работ в рамках проекта создан набор библиотек и прикладных программ, предназначенных для реализации Grid-технологии, а также поддержки и разработки Grid-приложений.

В качестве более простого по настройке и применению пакета для организации распределенных параллельных вычислений на метакомпьютере, созданном на основе ресурсов географически удаленных вычислительных кластеров, может быть рассмотрено разработанное в Суперкомпьютерном центре г. Штутгарта специализированное программное обеспечение PACX-MPI (в дальнейшем PACX — PArallel Computer eXtension) [3—5]. С технической точки зрения библиотека PACX является прослойкой между MPI-приложением и нижележащей MPI-библиотекой и может эффективно использоваться для решения сильносвязанных (с большим числом межпроцессорных обменов) MPI-программ на ресурсах удаленных кластерных систем. Программы такого рода получаются обычно при численном решении конечно-разностными методами задач механики сплошных сред или физики атмосферного пограничного слоя.

Целью данной работы является построение многопроцессорной вычислительной установки (метакластера) на базе ресурсов вычислительных кластеров ТГУ и ИОА СО РАН и программного обеспечения PACX-MPI, ее апробация на задаче переноса примеси, поступающей из приподнятого над землей точечного источника, в атмосфере.

1. Библиотека для организации параллельных вычислений PACX-MPI

Предположим, у нас имеется вычислительные кластерные системы с распределенной памятью, которые используются для параллельных вычислений на базе библиотеки MPI, но расчеты на них можно производить лишь по отдельности, т. е. вычислительные ресурсы данных кластеров нельзя использовать совместно для запуска одной параллельной MPI-программы. Для преодоления этой ограниченности в Суперкомпьютерном центре г. Штутгарта была создана библиотека PACX-MPI.

Библиотека PACX обладает следующими преимуществами:

- не требуется изменение кода MPI-приложения, достаточно лишь его откомпилировать с библиотекой PACX;

- осуществляется сжатие данных перед отправкой межкластерных сообщений;
- производится буферизация передаваемых данных;
- имеется поддержка аппаратно-программной гетерогенности кластеров;
- для внешней коммуникации реализована поддержка стандартных протоколов TCP, ATM, SSL;
- для внутрикластерных сообщений используются стандартные (оптимизированные) MPI-вызовы нижележащей MPI библиотеки.

Концептуальная схема взаимодействия кластеров PACX проиллюстрирована на рис. 1 [3]. На каждом кластере для коммуникации с другими кластерами используется два процесса. Один — PACX In-Server — фоновый процесс для отправки исходящих сообщений, другой — PACX Out-Server — фоновый процесс для приема входящих сообщений. PACX In-Server каждого кластера устанавливает связь с PACX Out-Server'ами остальных кластеров, которые планируется объединить в метакластер.

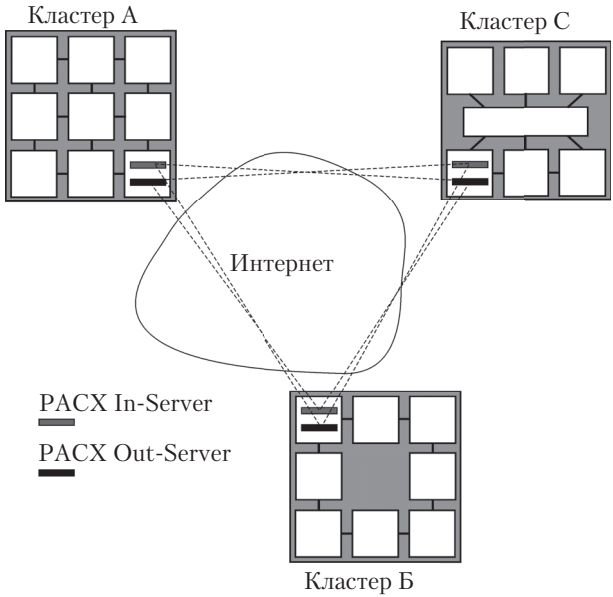

Рис. 1. Концептуальная схема взаимодействия кластеров PACX-MPI

В библиотеке PACX для организации межкластерных сообщений каждый вычислительный узел имеет два номера: локальный номер и глобальный номер. Глобальный номер идентифицирует узел как узел создаваемого метакомпьютера, а локальный номер является внутреннем идентификатором в соответствующем кластере — компоненте метакомпьютера. Для понимания необходимости использования двойной нумерации рассмотрим пример передачи сообщения типа «точка-точка» из глобального узла 2 в глобальный узел 7 на рис. 2.

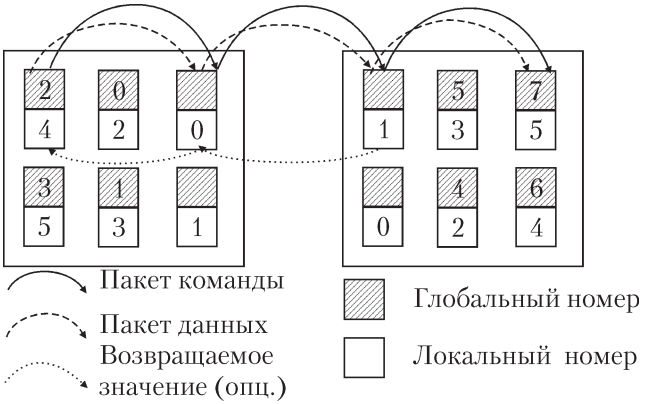

Рис. 2. Пример передачи сообщения типа «точка-точка»

Как видно на рис. 2, каждый кластер для коммуникации использует два *выделенных коммуникационных узла*. Эти коммуникационные узлы прозрачны для пользователя и, соответственно, они не имеют глобальных номеров. При отправке сообщения узел с глобальным номером 2 обнаруживает, что его адресат находится на втором кластере, поэтому он передает сообщение своему коммуникационному узлу (процесс PACX Out-Server). Коммуникационный узел первого кластера, в свою очередь, упаковывает сообщение и передает его коммуникационному узлу второго кластера (процесс PACX In-Server). Коммуникационный узел второго кластера распаковывает сообщение, сопоставляет глобальный номер узла (7) с локальным номером (5), затем передает сообщение с помощью MPI-вызова нижележащей библиотеки MPI узлу 5.

2. Задача о переносе примеси и численный метод ее решения

Предполагается, что в определенный момент времени из приподнятого на высоте h над поверхностью точечного источника начался выброс примеси (рис. 3), и необходимо рассчитать распределение концентрации примеси в различных точках рассматриваемой области через некоторое время.

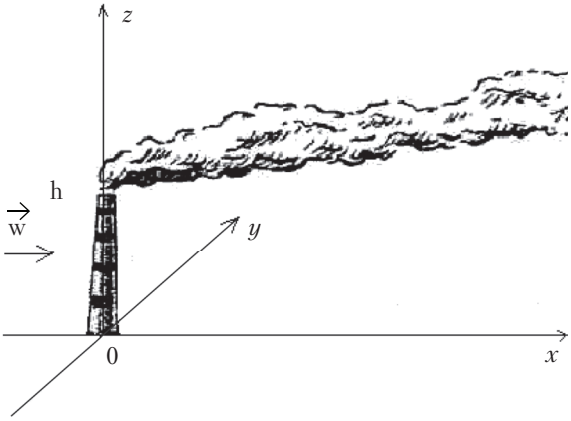

Рис. 3. Система координат

Известными являются локальные метеорологические параметры в приземном слое атмосферы (компоненты вектора скорости ветра и коэффициенты турбулентной диффузии) и начальное распределение концентрации примеси в рассматриваемой области. Интенсивность выброса примеси является переменной величиной и зависит от времени t.

2.1. Математическая постановка задачи

Математически процесс переноса примеси от точечного источника описывается краевой задачей для адвективно-диффузионного уравнения следующего вида [6]:

$$\frac{\partial C}{\partial t}+u\frac{\partial C}{\partial x}+v\frac{\partial C}{\partial y}+w\frac{\partial C}{\partial z}= \\ =\frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right)+\frac{\partial}{\partial y}\left(D\frac{\partial C}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial C}{\partial z}\right)+S(t,x,y,z) \qquad (1)$$

где u, v, w — проекции вектора скорости ветра на оси OX, OY, OZ соответственно; D, K_z — коэффициенты горизонтальной и вертикальной турбулентной диффузии; $S = f(t)\delta(x)\delta(y)\delta(z-h)$ — источник примеси, который располагается в точке с координатами $(0;0;h)$; $C = C(t, x, y, z)$ — концентрация примеси.

В начальный момент времени известно распределение концентрации примеси $C_0(x,y,z)$ в рассматриваемой области $\left\{-L_x/2 \dots L_x/2; -L_y/2 \dots L_y/2; 0 \dots L_z\right\}$, а на границах выполняются следующие граничные условия:

$$\frac{\partial C}{\partial x} = 0 \text{ при } x = \frac{-L_x}{2}, \ \frac{\partial C}{\partial x} = 0 \text{ при } x = \frac{L_x}{2};$$

$$\frac{\partial C}{\partial y} = 0 \text{ при } y = \frac{-L_y}{2}, \ \frac{\partial C}{\partial y} = 0 \text{ при } y = \frac{L_y}{2}; \qquad (2)$$

$$\frac{\partial C}{\partial z} = 0 \text{ при } z = 0, \ \frac{\partial C}{\partial z} = 0 \text{ при } z = L_z,$$

где L_x, L_y, L_z — размеры области исследования по осям OX, OY, OZ соответственно.

Решение данной краевой задачи состоит в нахождении концентрации примеси в рассматриваемой области $\left\{0 \dots L_x; 0 \dots L_y; 0 \dots L_z\right\}$ через требуемый промежуток времени T. Аналитическое решение данного дифференциального уравнения может быть получено только в простейших случаях [7]. Поэтому для решения таких задач в основном используются численные методы решения [8].

2.2. Метод решения

Для решения поставленной задачи используется метод конечных разностей или метод сеток [8]. Суть данного метода заключается в следующем. Область непрерывного изменения аргументов заменяется конечным множеством дискретных узлов (сеткой). Вводятся сеточные функции, определяемые значениями функций непрерывных аргументов в узлах сетки. Входящие в уравнение частные производные заменяются *разностными соотношениями*. После такой замены решение задачи сводится к решению системы разностных алгебраических уравнений.

Определим конечно-разностную сетку следующим образом:

$$\varpi_h = \left\{\left(t^n, x_i, y_j, z_k\right)\right\},$$

где

$$t^n = n \cdot \tau,\ n = 0 \ldots N_T,\ \tau = T / N_T$$

$$x_i = i \cdot h_x,\ i = 0 \ldots N_x,\ h_x = L_x / N_x$$

$$y_j = j \cdot h_y,\ j = 0 \ldots N_y,\ h_y = L_y / N_y$$

$$z_k = k \cdot h_z,\ k = 0 \ldots N_z,\ h_z = L_z / N_z$$

Таким образом, будем искать приближенное решение задачи (1)—(2) в узлах конечной разностной сетки $C^n_{i,j,k} \approx C\left(t^n, x_i, y_j, z_k\right)$, где $n = 0...N_T$, $i = 0...N_x$, $j = 0...N_y$, $k = 0...N_z$. N_T, N_x, N_y, N_z — количество разбиений по времени и по осям OX, OY, OZ соответственно.

При замене производных по x, y, z, входящих в уравнение (1), во внутренних узлах сетки воспользуемся центрально-разностными аппроксимациями и правыми разностями — для производной по времени. Граничные условия (2) будем аппроксимировать с помощью разностных соотношений первого порядка точности.

В результате получим явную разностную схему следующего вида (для случая постоянных коэффициентов):

$$\begin{cases} \dfrac{C^{n+1}_{i,j,k} - C^n_{i,j,k}}{\tau} + u\dfrac{C^n_{i+1,j,k} - C^n_{i-1,j,k}}{2h_x} + v\dfrac{C^n_{i,j+1,k} - C^n_{i,j-1,k}}{2h_y} + w\dfrac{C^n_{i,j,k+1} - C^n_{i,j,k-1}}{2h_z} = \\ = D\left(\dfrac{C^n_{i+1,j,k} + C^n_{i-1,j,k} - 2C^n_{i,j,k}}{h_x^{\,2}} + \dfrac{C^n_{i,j+1,k} + C^n_{i,j-1,k} - 2C^n_{i,j,k}}{h_y^{\,2}}\right) + K_z\dfrac{C^n_{i,j,k+1} + C^n_{i,j,k-1} - 2C^n_{i,j,k}}{h_z^{\,2}} + \\ + S^n_{i,j,k}, \quad i = 1,...,N_x - 1, j = 1,...,N_y - 1, k = 1,...,N_z - 1, n > 0; \\ C^0_{i,j,k} = C0(x_i, y_j, z_k), i = 0,...,N_x, j = 0,...,N_y, k = 0,...,N_z; \\ C^{n+1}_{0,j,k} = C^{n+1}_{1,j,k}; C^{n+1}_{N_x,j,k} = C^{n+1}_{N_x-1,j,k}; j = 1,...,N_y - 1, k = 1,...,N_z - 1, n = 0,...,N_T - 1; \\ C^{n+1}_{i,0,k} = C^{n+1}_{i,1,k}; C^{n+1}_{i,N_y,k} = C^{n+1}_{i,N_y-1,k}; i = 1,...,N_x - 1, k = 1,...,N_z - 1, n = 0,...,N_T - 1; \\ C^{n+1}_{i,j,0} = C^{n+1}_{i,j,1}; C^{n+1}_{i,j,N_z} = C^{n+1}_{i,j,N_z-1}; i = 1,...,N_x - 1, j = 1,...,N_y - 1, n = 0,...,N_T - 1; \end{cases}$$

Эта разностная схема имеет первый порядок аппроксимации по времени и координатам. Она является условно устойчивой и монотонной при выполнении неравенств

$$\tau < \frac{1}{2D\left(\frac{1}{h_x^{\,2}} + \frac{1}{h_y^{\,2}}\right) + \frac{2K_z}{h_z^{\,2}}},\ h_x < \frac{2D}{u}, h_y < \frac{2D}{v}, h_z < \frac{2K_z}{w}.$$

3. Параллельная реализация численного метода и результаты

Для распараллеливания рассматриваемой задачи существуют различные способы декомпозиции вычислений по активным процессам. Одним из них является декомпозиция по данным (data decomposition) [9]. Суть данного способа декомпозиции заключается в равномерном распределении данных для однородных вычислений по имеющимся процессам.

При решении задачи методом конечных разностей на каждом шаге по времени необходимо приближенно рассчитать значения $C(t, x, y, z)$ в точках сетки ϖ_h, покрывающей рассматриваемую область, т. е. фактически необходимо найти $N_x \cdot N_y \cdot N_z$значений. При декомпозиции необходимо учесть, что при вычислении значения $C_{i,j,k}^{n+1}$ используются значения сеточной функции в соседних по осям Ox, Oy, Oz узлах сетки, полученные на предыдущем шаге времени.

Известно, что время передачи одного числа между вычислительными узлами на современных вычислительных кластерах с распределенной памятью в среднем на два-три порядка превышает время одной арифметической операции. Поэтому при декомпозиции расчетной области необходимо минимизировать количество узлов сеточного шаблона, принадлежащих различным процессам. С учетом этого требования ниже рассмотрены следующие способы декомпозиции расчетной области: одномерная и двухмерная.

3.1. Распределение обрабатываемых данных между вычислительными узлами

В данной работе разбиение при одномерной декомпозиции производится по одному из координатных направлений — оси OY. Если визуально всю расчетную область представить в виде параллелепипеда, то в результате одномерной декомпозиции параллелепипед по одной координатной оси разбивается на p

равных параллелепипедов (рис. 4), где p — число активированных копий параллельной MPI-программы.

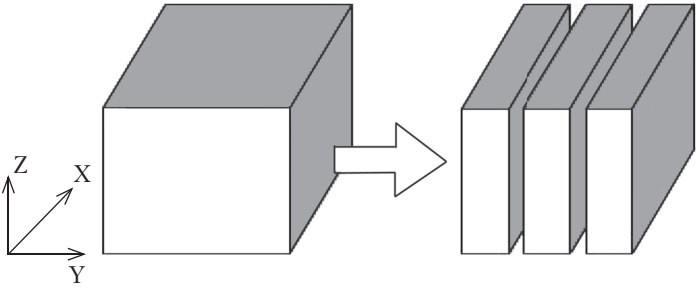

Рис. 4. Одномерная декомпозиция расчетной области

В результате разбиения на соприкасающихся границах полученных параллелепипедов для вычисления значений концентраций по выбранному шаблону одно требуемое значение попадает на соседний параллелепипед. Поэтому на каждом шаге соседним процессам необходимо обмениваться значениями для обеспечения следующего $n+1$-го шага вычисления. На рис. 5 изображен обмен результатами вычислений между процессами при одномерной декомпозиции расчетной области. Показаны узлы сетки, в которых вычисляются значения концентрации на каждом процессе. Более темным цветом выделены граничные значения концентрации, которые передаются на соседний процесс на каждом шаге по времени.

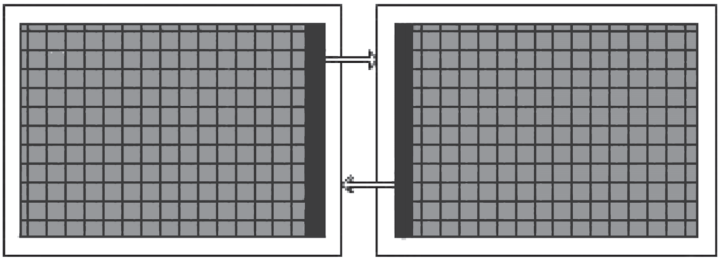

Рис. 5. Обмен результатами вычислений между соседними процессами

Разбиение при двухмерной декомпозиции производится по двум направлениям (например, по осям OY, OZ). В отличие от одномерной декомпозиции разбиение производится сначала

в направлении оси OY, зачем получившиеся параллелепипеды разбиваются на такое же количество областей в другом направлении — оси OZ (рис. 6).

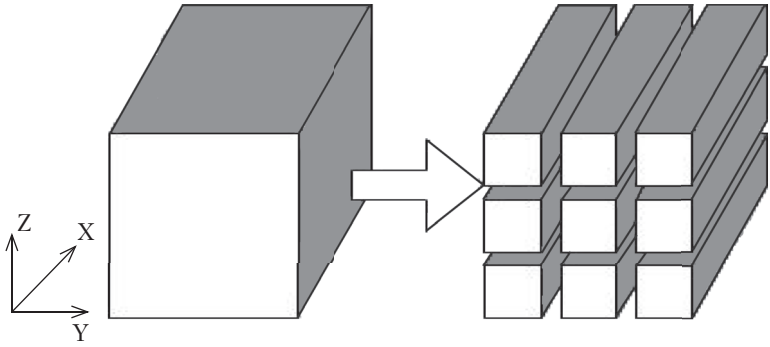

Рис. 6. Двухмерная декомпозиция расчетной области

В результате такого разбиения каждый получившийся параллелепипед граничит с не более чем с четырьмя соседними параллелепипедами. На рис. 7 изображен обмен результатами вычислений при двухмерной декомпозиции расчетной области для разбиения $p = 4$. Ограничение данной декомпозиции заключается в том, что число разбиений должно быть равно квадрату целого числа, т. е. $p = 1, 4, 9, 16, 25, ...$

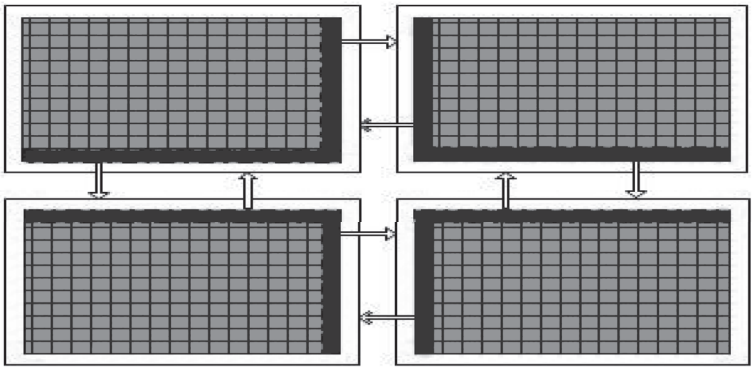

Рис. 7. Обмен результатами вычислений
с соседними процессами для разбиения $p = 4$

При одномерной и двухмерной декомпозиции задачи переноса примеси на эффективность параллельной программы значительное влияние оказывает способ передачи сообщений между процессами. В соответствии с [10] наиболее приемлемым способом при одномерной и двухмерной декомпозиции является передача и прием сообщений типа «точка-точка».

Библиотека MPI предоставляет широкий набор функций для передачи и приема сообщений типа «точка-точка» [10]. В одномерной и двухмерной декомпозиции использованы следующие способы передачи и приема сообщений:

MPI_SendRecv — блокирующая операция приема-передачи объединяет в едином вызове передачу сообщения одному процессу и прием сообщения от того же процесса;

Неблокирующие прием и передача состоят из двух этапов: инициализация передачи (MPI_Isend) и приема (MPI_Irecv); блокирующая проверка завершения обмена (MPI_Wait).

Первый метод передачи и приема сообщений выгоден с точки зрения используемой памяти, но влечет за собой простаивание процессоров. В свою очередь если кластерная сеть высокоскоростная, то это простаивание процессов может оказывать незначительное влияние на общее время вычислений. Второй метод приема и передачи является полной противоположностью, вероятность простаивания процессов очень мала, но свою очередь требуется больше объема оперативной памяти. В итоге выбор использования того или иного способа во многом обусловлен спецификой задачи, которую во многих случаях трудно определить без проведения экспериментальных опытов на многопроцессорной системе.

3.2. Теоретическая оценка ускорения и эффективности

Для оценки производительности параллельной реализации решаемой задачи применяются следующие характеристики: *ускорение* и *эффективность* [11]. Ускорение на p процессорах, полученное при решении задачи с размерностью N (для удобства проведения оценок принимаем, что $N_x = N_y = N_z = N-1$), рассчитывается по следующей формуле:

$$S(N,p) = \frac{T(N,1)}{T(N,p)}$$

т. е. через отношение времени решения задачи на одном процессоре ко времени ее решения на p-процессорной системе.

Для рассматриваемых подходов декомпозиции общее время выполнения параллельной программы состоит из времени, затрачиваемого на арифметические операции, и времени, требуемого на обмен значениями между процессами. Предполагается, что процессоры обладают одинаковой вычислительной производительностью и вычислительная нагрузка сбалансирована. Поэтому общее время на арифметические операции при использовании одномерной декомпозиции $\left(T_a^{1D}\right)$ обратно пропорционально количеству активных процессов. Если процессы осуществляют обмен данными одновременно, то общее время на обмен значениями $\left(T_o^{1D}\right)$ не будет зависеть от количества процессов. В итоге

$$T^{1D} = T_a^{1D} + T_o^{1D} \approx t_a \frac{N^3}{p} + 2N^2 t_o$$

где t_a — время выполнения арифметических операций для вычисления значения C_{ijk}^{n+1}, t_o — время передачи одного значения между процессами, p — количество процессов

Таким образом, теоретическая оценка ускорения для одномерной декомпозиции:

$$S(N,p) = \frac{t_a N^3}{t_a \frac{N^3}{p} + 2N^2 t_o} = \frac{p}{1 + 2\frac{t_o}{t_a}\frac{p}{N}} \qquad (3)$$

Для анализа ускорения параллельного алгоритма в случае одномерной декомпозиции были проведены серии экспериментов для $p = 1 \ldots 20$, $N = 189$ ($189 \times 189 \times 189$) на кластере ИОА СО РАН (рис. 8).

Использование блокирующих вызов при одномерной декомпозиции дает лучшее ускорение, чем неблокирующих. Это объясняется тем, что внутренние механизмы управления посылкой асинхронных сообщений не совсем эффективны, когда размер сообщения длинный (при $N = 189$) ~ 300 Kb.

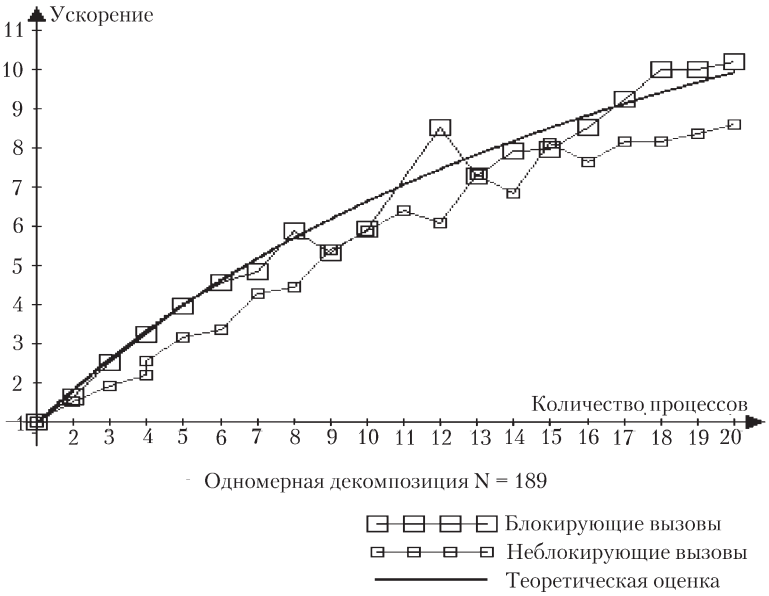

Рис. 8. Ускорение параллельной программы при одномерной декомпозиции

При двухмерной декомпозиции общее время выполнения параллельной программы состоит из времени, затраченного на арифметические операции, и времени на обмен значениями между процессами по двум координатным направлениям (OY, OZ):

$$T^{2D} = T_a^{2D} + T_o^{2D} = t_a \frac{N^3}{p} + 4\frac{N^2}{\sqrt{p}} t_o$$

Тогда теоретическая оценка ускорения для двумерной декомпозиции будет иметь вид:

$$S(N,p) = \frac{t_a N^3}{t_a \dfrac{N^3}{p} + 4t_o \dfrac{N^2}{\sqrt{p}}} = \frac{p}{1 + 4\dfrac{t_o}{t_a}\dfrac{\sqrt{p}}{N}} \qquad (4)$$

Для анализа ускорения параллельного алгоритма в случае двухмерной декомпозиции были проведены серии экспериментов для $p = 1, 4, 9, 16$, $N = 189$ $(189 \times 189 \times 189)$ на кластере ИОА СО РАН (рис. 9).

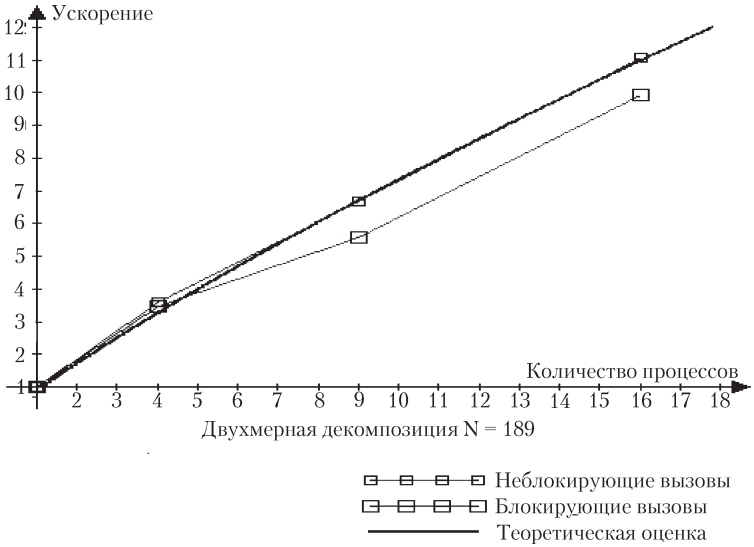

Рис. 9. Ускорение параллельной программы при двухмерной декомпозиции

Как видно, в случае двухмерной декомпозиции по мере уменьшения размера сообщения (при 16 процессах размер сообщения ~75 Kb) неблокирующие вызовы дают лучшее ускорение, чем синхронные вызовы. Таким образом, асинхронные вызовы уместнее использовать, когда межпроцессорные сообщения достаточно небольшого размера.

В результате проведенных экспериментов показано, что двухмерная декомпозиция является более эффективной по сравнению с одномерной. На шестнадцати процессорах двухмерная декомпозиция дает такое же ускорение, которое при одномерной декомпозиции достигается только на двадцати процессорах.

4. Тестирование метакомпьютера на примере задачи переноса примеси

Тестирование сконфигурированной многопроцессорной системы проводилось на примере задачи переноса примеси для одномерной декомпозиции с использованием блокирующего вызова (MPI_SendRecv) при размерности $N = 200$. Запуск MPI-приложения с высокой долей обменных операций осуществлялся на

10 процессорах метакомпьютера, состоящего из вычислительных узлов кластеров ТГУ и ИОА СО РАН (см. табл. 1). В качестве канала связи между кластерами рассматривались общественный Интернет г. Томска и выделенная оптоволоконная линия.

Таблица 1

Технические характеристики кластеров ТГУ и ИОА СО РАН

	Кластер ИОА СО РАН	**Кластер ТГУ**
Число узлов	10	9
Конфигурация узла	2 ЦП Pentium 3 1 GHz, RAM 1 Gb, бездисковый	2 ЦП Pentium 3 650 MHz, RAM 256 Mb, бездисковый
Конфигурация сервера	2 ЦП Pentium 3 1 Ghz, RAM 1 Gb, 3 SCSI HDD 18Gb	2 ЦП Pentium 3 650 MHz, RAM 512Mb, 36 Gb
Коммуникационное оборудование	Gigabit Ethernet на коммутаторе 3Com SuperStack 4900	100 Mbit Ethernet на коммутаторе Cisco Catalyst 2924
Программное обеспечение	Linux Kernel 2.4.28 , MPICH 1.2.4, Intel C++ & Fortran Compiler 7.1	Linux Kernel 2.6.9-22, MPICH 1.2.6, Intel C++ & Fortran Compiler 9.0
Реальная производительность на тесте LINPACK	10.5 GFlops	5.5 GFlops
Пиковая производительность	20 GFlops	11,7 GFlops

Гетерогенность метакомпьютера, построенного на базе разнородных кластеров, можно характеризовать 3 видами сообщений: внутрикластерные сообщения кластера ТГУ, внутрикластерные сообщения кластера ИОА СО РАН, межкластерные сообщения. Эффективность передачи таких сообщений зависит от латентности и пропускной способностью коммуникационной среды, по которой они передаются.

В случае использования канала связи «Интернет» (см. табл. 2) скорость передачи межкластерных и внутрикластерных сообщений имеет существенные различия, что не дает никакого преимущества для использования метакомпьютера для сильносвязанных MPI-приложений с интенсивными блокирующими сообщениями. В этом случае актуальным является использование метакомпьютера для задач пакетной обработки.

Таблица 2

Результаты тестирования метакомпьютера

Суммарная пиковая мощность в GFlops	Количество ЦП (CPU)		Время работы в минутах			
	Кластер ТГУ	Кластер ИОА СО РАН	MPI запуск	PACX-MPI запуск	Совместный PACX-MPI запуск, канал связи Интернет	Совместный PACX-MPI запуск, канал связи Оптоволокно
6,5	10		39,45	40,25		
10		10	18,43	19,53		
7,5	7	3			120,39	34,11
8,95	3	7			79,32	31,10
8,25	5	5			93,58	32,19

В случае использования выделенного канала связи «Оптоволокно» конфигурация (7,3) имеет примерно такую же производительность, что и конфигурация (10,0). Причем, возрастание числа более производительных процессоров кластера ИОА СО РАН в составе вычислительных узлов метакомпьютера все в меньшей степени влияет на увеличение производительности метакомпьютера.

Для оптимального использования метакомпьютера при решении сильносвязанных параллельных задач в его конфигурацию необходимо включать в большом количестве узлы менее производительного кластера, чем узлы более производительного кластера. Кроме того, поскольку на каждом кластере необходимо выделять два коммуникационных узла, то суммарная пиковая производительность полученного метакомпьютера может составить 28,4 GFlops. Это на 8,4 GFlops больше, чем пиковая производительность кластера ИОА СО РАН.

Заключение

Для запуска сильносвязанных параллельных MPI-программ построен метакомпьютер на базе двух вычислительных кластеров Томского научного центра (ТГУ и Института оптики атмосферы СО РАН). В качестве программной инфраструктуры, создающей единый высокопроизводительный вычислительный ресурс, использовалось программное обеспечение PACX-MPI, разработанное в Суперкомпьютерном центре г. Штутгарта. Этот пакет был выбран в силу относительной простоты в установке, настройке и конфигурировании. Кроме того, он позволяет без какой-либо дополнительной доработки использовать созданные для вычислительных кластеров с распределенной памятью MPI-программы.

Для тестирования созданного метакомпьютера была разработана программа решения трехмерного нестационарного адвективно-диффузионного уравнения конечно-разностными методами с использованием явных разностных схем; получена параллельная реализация этого метода с использованием одномерной и двумерной декомпозиции; исследовано влияние способа организации межпроцессорных обменов (блокирующих и неблокирующих) на основе стандарта MPI; выполнены теоретические оценки ускорения разработанной параллельной программы, сделано сравнение с результатами вычислительных экспериментов на кластере ИОА СО РАН.

На основе разработанной параллельной программы для переноса примеси с использованием блокирующих межпроцессорных обменов (MPI_SendRecv) исследовано влияние конфигурации гетерогенного метакомпьютера на его реальную производительность; показана перспективность рассматриваемого способа объединения кластеров с точки зрения увеличения производительности; продемонстрированы повышенные требования к высокоскоростной сети, связывающей компоненты метакомпьютера.

Литература

1. **Эндрюс Г. Р.** Основы многопоточного, параллельного и распределенного программирования. — М. : Вильямс, 2003. — 512 с.

2. [Электронный ресурс]. — Режим доступа: http://www.globus.org , свободный.

3. **Keller R., Muller M.** PACX-MPI Home Page [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi , свободный.

4. **Gabriel E., Resch M., Beizel T., Keller R.** Distributed Computing in Heterogeneous Computing Environment EuroPVMM-PI'98 Liverpool/UK, 1998 [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi.

5. **Trams M.** Feasibility of PACX-MPI for use in a Cluster of Clusters Environment [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi.

6. **Старченко А. В., Беликов Д. А.** Численная модель для оперативного контроля уровня загрязнения городского воздуха // Оптика атмосферы и океана. — 2003. — Т. 16. — № 7. — С. 655—667.

7. **Берлянд М. Е.** Прогноз и регулирование загрязнения атмосферы. — Л. : Гидрометиздат, 1985. — 272 с.

8. **Самарский А. А.** Введение в численные методы. — СПб. : Лань, 2005. — 228 с.

9. **Minty E., Davey R., Simpson A., Henty D.** Decomposing the Potentially Parallel — The Edinburgh Parallel Computing Centre [Электронный ресурс]. — Режим доступа : http://www.epcc.ed.ac.uk.

10. **Немнюгин С. А., Стесик О. Л.** Параллельное программирование для многопроцессорных вычислительных систем. — СПб. : БХВ-Петербург, 2002. — 400 с.

11. **Воеводин В. В.** Распределенная обработка данных // Вторая Сибирская школа-семинар по параллельным вычислениям. — Томск: Изд-во Том. ун-та, 2004. — С. 3—9.