Оценка эффективности алгоритмов математической физики для компьютеров с распределенной памятью

И. Н. Коньшин[1,2]

[1] Вычислительный центр им. А.А. Дородницына, ФИЦ ИУ РАН, Москва 119333
[2] Институт вычислительной математики им. Г.И. Марчука Российской академии наук, Москва 119333

igor.konshin@gmail.com

Аннотация В работе представлены несколько моделей выполнения параллельных программ на компьютерах с распределенной памятью. Предсказание параллельной эффективности алгоритма основано на оценках вычислительной и коммуникационной сложности алгоритма. Дле некоторых алгоритмов вычислительной математики для явных схем решения уравнения теплопереноса получены оценки ускорения, проведены численные эксперименты для сравнения реального и теоретически предсказанного ускорения.

Keywords: математическая физика, параллельные вычисления, оценка параллельной эффективности, ускорение

1 Введение

Существует множество работ и интернет рессурсов, в которых описываются свойства вычислительных алгоритмов, модели параллельных вычислений, а также даются рекомендации по написанию наиболее эффективных приложений [4–6]. В данной работе сделана попытка создания конструктивной модели параллельных вычислений, на основе которой можно предсказывать параллельную эффективность реализации алгоритма на конкретной вычислительной системе.

При работе на компьютерах с общей памятью, можно строить такую модель на основе закона Амдала (см. [1,2]), на компьютерах с распределенной памятью необходимо учитывать свойства параллельности как реализуемого алгоритма, так и самой вычислительной системы. В данной работе будет проведен детальный анализ скорости передачи сообщений в зависимости от длины сообщения. При игнорировании в модели времени инициализации сообщений можно получить более компактные формулы оценки параллельной эффективности, а при ее учитывании несколько более сложные, но также конструктивные и содержательные.

В данной работе мы сосредоточим свое внимание на оценке параллельной эффективности для задач математической физики. В литературе имеются

описания огромного количества результатов по достигнутой параллельной эффективности для задач математической физики, но нет теоретических оценок о том какой эффективности можно было бы достигнуть, а таким образом и сравнения теоретической и практической эффективности.

Данная работа имеет следующую структуру. В первом и втором разделах приводятся оценки параллельной эффективности алгоритмов и их применение к задачам математической физики. В третьем разделе кратко описывается конфигурация используемого вычислительного кластера. В четвертом и пятом разделе анализируются численные эксперименты по проведению обменов на фоне вычислений, а также детально исследуется скорость передачи сообщений в зависимости от длины сообщения. В шестом разделе проводится уточнение оценок на основе полученных результатов, а также их применение для алгоритмов матемаческой физики. В седьмом разделе дается описание результатов численных экспериментов, а в заключении приводятся краткие выводы по работе.

2 Оценка параллельной эффективности алгоритмов

Для получения оценки времени работы параллельного алгоритма ключевым моментом является оценка времени передачи сообщения. Воспользуемся для этого широко используемой формулой

$$T_c = \tau_0 + \tau_c L_c, \tag{1}$$

где τ_0 – время инициализации сообщения, τ_c – скорость передачи сообщений (т.е. время передачи сообщения единичной длины), а T_c – время передачи сообщения длины L_c. Детальное изучение величин τ_0 и τ_c мы проведем несколько позже в разделе 6, а сейчас нам будет достаточно считать, что в исследуемых алгоритмах длина сообщений достаточно велика и для простоты можно считать что $\tau_0 = 0$. Тогда формула (1) для оценки времени передачи сообщения существенно упрощается:

$$T_c = \tau_c L_c. \tag{2}$$

Оценим ускорение, которое можно получить при использовании p процессоров при работе некоторого параллельного алгоритма. Пусть $T(p)$ – время решения задачи на p процессорах, тогда ускорение расчета, которое можно получить, используя данный алгоритм, будет выражаться формулой:

$$S = T(1)/T(p).$$

Следуя [1, 2], для проведения дальнейших оценок обозначим через L_a общее количество арифметических операций алгоритма, а через τ_a время выполнения одной арифметической операции. Аналогично, пусть L_c – общая длина всех сообщений, а τ_c – уже введенное время передачи сообщения единичной длины. Тогда, время выполнения всех арифметических операций

можно выразить формулой $T_a = \tau_a L_a$, а время передачи всех сообщений $T_c = \tau_c L_c$.

Дополнительно, можно ввести величину

$$\tau = \tau_c/\tau_a, \tag{3}$$

которая выражает характеристику "параллельности" используемого компьютера, другими словами означающую, сколько арифметических операций можно успеть выполнить за время передачи одного числа на другой процессор. Аналогично, введем величину

$$L = L_c/L_a, \tag{4}$$

которая выражает характеристику "параллельности" исследуемого алгоритма, т.е. является обратной величиной к количеству арифметических операций выполняемых при работе алгоритма на передачу одного числа на другой процессор.

Теперь при оценке ускорения мы можем записать:

$$\begin{aligned} S = S(p) = T(1)/T(p) = T_a/(T_a/p + T_c/p) = pT_a/(T_a + T_c) \\ = p/(1 + T_c/T_a) = p/(1 + (\tau_c L_c)/(\tau_a L_a)) = p/(1 + \tau L). \end{aligned} \tag{5}$$

Оценка эффективности работы алгоритма при этом будет выглядеть следующим образом:

$$E = S/p = 1/(1 + \tau L). \tag{6}$$

В работах [1,2] можно найти детальное обсуждение применимости полученных оценок (5)–(6), мы же ограничимся пояснением на первый взгляд странного факта, что в формулу оценки эффективности (6) не входит количество используемых процессоров p. На самом деле, величина L зависит от общей длины обменов L_c, которое, в свою очередь, зависит от количества процессоров p.

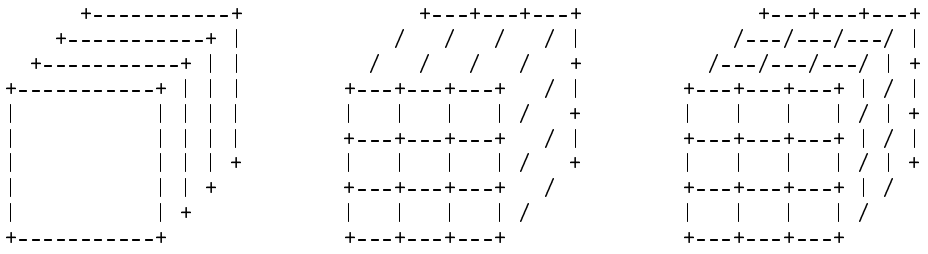
3 Оценки для алгоритмов математической физики

Теперь, следуя [8], можно приступить к оценке параллельной эффективности алгоритмов математической физики. Для простоты ограничим наше рассмотрение явными схемами, примененными для нестационарных задач, описываемыми некоторыми конечно-разностными уравнениями.

Будем рассматривать задачу в d-мерном кубе ($d = 1, 2, 3$) со стороной n ячеек, общее количество d-мерных кубических ячеек будет равно $N = n^d$. Пусть через V обозначено количество неизвестных функций на расчетную ячейку (например, $V = 5$ при рассмотрении трех скоростей u, v, w, давления и температуры). Пусть для вычисления значений на новом временном слое в каждой ячейке требуется знать значения в ближайших соседних ячейках, что означает использование $(2d+1)$-точечного d-мерного шаблона дискретизации (более сложные дискретизации, особенно для уравнений с перекрестными производными, могут содержать в шаблоне до 3^d точек). Обозначим

через C среднее количество арифметических операций на ячейку при вычислении решения на новом временном слое. Теперь, почти все готово для оценки ускорения при решении описанной задачи на p процессорах, но получаемое ускорение будет существенно зависеть от способа распределения ячеек по процессорам.

На рис. 1 показаны три различных способа распределения данных по процессорам, $D = 1, 2, 3$, используя по r слоев в одном, двух и трех направлениях, соответственно. Общее количество процессоров при этом будет равно $p = r^D$.



$D = 1$,	$D = 2$,	$D = 3$,
$p = r$,	$p = r \times r$,	$p = r \times r \times r$,
$L_a = CN/p$,	$L_a = CN/p$,	$L_a = CN/p$,
$L_c \approx 2Vn^2$,	$L_c \approx 4Vn^2/r$,	$L_c \approx 6Vn^2/r^2$,
$L \approx 2Vr/(Cn) \sim r/n$.	$L \approx 4Vr/(Cn) \sim r/n$.	$L \approx 6Vr/(Cn) \sim r/n$.

Рис. 1: (a) Распределение данных по процессорам для трехмерной задачи математической физики по r слоев в одном, двух и трех направлениях.

Естественно полагая, что $D \le d$, можно приступить к оценкам вычислительных и коммуникационных затрат.

Арифметические затраты, в пересчете на 1 процессор, для всех рассматриваемых случаев составят $L_a = CN/p = Cn^d/r^D$. Коммуникационные затраты будут зависеть от количества граничных ячеек, которые нужно принять (отправить) на каждом из процессоров, и составят $L_c = (2 - 2/r)DVn^{d-1}/r^{D-1}$. Заметим, что $L_c = 0$ при $r = 1$.

Таким образом, характеристика параллельности алгоритма (4) будет вычисляться следующим образом:

$$L = L_c/L_a = (2 - 2/r)DVn^{d-1}r^D/(Cn^2r^{D-1}) = (2 - 2/r)DVC^{-1} \cdot r/n \sim r/n. \tag{7}$$

Эта величина обратно пропорциональна количеству ячеек данного процессора в направлении разбиения по процессорам.

Подставляя найденное выражение для L в (6), получаем

$$E = 1/(1 + (2 - 2/r)DVC^{-1}\tau \cdot r/n), \qquad S = pE \tag{8}$$

p	$D=1$	$D=2$	$D=3$
1	1.00	1.00	1.00
10	0.97	0.98	0.98
64	0.82	0.95	0.96
729	0.29	0.85	0.92

Таблица 1: Теоретическая оценка параллельной эффективности (9) при параметрах (10).

или, чтобы прояснить также и зависимость от размерности исходной задачи d:

$$E(d,D) = 1/(1 + (2 - 2/p^{1/D})DVC^{-1}\tau \cdot p^{1/D}/N^{1/d}), \qquad S = pE(d,D). \quad (9)$$

Чтобы проиллюстрировать применение формулы 9, приведем короткую таблицу 1, где показаны теоретические оценки зависимости параллельной эффективности от количества процессоров p и типа декомпозиции области по процессорам D для трехмерной задачи при следующем выборе параметров:

$$d = 3, \quad N = n^3, \quad n = 1000, \quad V = 5, \quad C = 30, \quad D = 1, 2, 3, \quad \tau = 10. \quad (10)$$

Полученные значения эффективности показывают важность выбора правильного типа декомпозиции D, особенно при использовании большого количества процессоров.

4 Вычислительный кластер

Численные эксперименты проводились на кластере ИВМ РАН [9]. Характеристики вычислительных узлов из очереди “x6core”, используемых для проведения расчетов:

- Compute Node Asus RS704D-E6;
- 12 ядер (два 6-ядерных процессора Intel Xeon X5650@2.67 ГГц);
- Оперативная память: 24 Гб.;
- Операционная система: SUSE Linux Enterprise Server 11 SP1 (x86_64);
- Коммутационная сеть: Mellanox Infiniband QDR 4x.

Для сборки кода использовался компилятор Intel языка C версии 4.0.1, с поддержкой MPI версии 5.0.3.

5 Проведение обменов на фоне счета

Наличие в стандарте MPI возможности проведения ассинхронных обменов позволяет, после инициализирования обменов, не дожидаясь их окончания,

N_{drops}	$p = 1$	$p = 2$	$p = 13$
1	15.42	17.18	18.52
8	15.43	15.58	15.78
64	15.43	15.63	15.77

Таблица 2: Время работы теста по проведению обменов на фоне счета в очереди “x6core”.

сразу приступить к выполнению тех вычислений, для которых на данном процессоре уже имеются все необходимые данные. Идеей проведения обменов на фоне вычислений пронизаны все руководства по параллельности. Однако эффект от перекрытия вычислений и обменов напрямую зависит от конкретной архитектуры и реализации MPI. Попробуем разобраться с этим вопросом подробнее.

Была разработана специальная программа, реализующая тест 1 из работы [7]. На фоне интенсивных вычислений на массивом чисел, происходила отправка порций посчитанных значений на другой процессор, на котором на следующей итерации эти данные использовались в его вычислениях. Ассинхронные обмены выполнялись с помощью функций MPI_Isend и MPI_Irecv, а для ожидания завершения обменов перед следующей итерацией использовадась функция MPI_Waitall. Длина массива была выбрана равной $M = 2^{25}$, а количество итераций было взято равным 10. Количество порций, на которые разбивался массив перед его отправкой на другой процессор, выбиралось равным 1, 8 и 64. Вычисления проводились на кластере ИВМ РАН [9] (см. раздел 4) в очереди “x6core”.

Результаты численных экспериментов по проведению обменов на фоне счета приведены в табл. 2. Отклонения при замерах времени были незначительными и составляли от 1 до 5 процентов. Рассмотрим результаты расчетов более подробно. Вычисления и обмены проводились на первом и последнем из используемых процессов. Параметр p означает количество используемых процессоров, таким образом, во второй колонке $p = 1$ приведены результаты расчетов без проведения обменов. Замеры времени в этой колонке практически одинаковы, мы будем использовать их контрольную точку наших дальнейших наблюдений. В первой строке при $N_{\text{drops}} = 1$ мы имеем ситуацию, когда данные отправляются одной порцией, таким образом, обмены получаются синхронными и перекрытия с вычислениями не происходит. Колонка $p = 2$ содержит данные, когда два обменивающихся процессов физически расположены на одном вычислительном узле и, несмотря на использование библиотеки MPI, обмены фактически проводятся в рамках общей памяти. Можно заметить, что за счет выполнения синхронных обменов ($N_{\text{drops}} = 1$) общее время счета возросло на около 10%. Колонка $p = 13$ содержит данные, когда два обменивающихся процесса физически расположены на различных вычислительном узле. Один вычислительный узел очереди “x6core”состоит из двух 6-и ядерных процессоров, а так как в тесте вычисления и обмены проводятся на первом и последнем процессах, то

в итоге обмены проводятся через комминикационную сеть между узлами. За счет выполнения синхронных обменов ($N_{\text{drops}} = 1$) общее время счета в этом случае возросло на около 20%. С ростом количества порций N_{drops} общее время счета уменьшается, стремясь ко времени счета на 1 процессоре.

Полученные результаты позволяют сделать вывод о том, что на данном вычислительном устройстве эффект от проведения обменов на фоне счета наблюдается, но не является значительным и определяющим.

Следует также заметить, что теоретический учет вклада ассинхронных обменов в общее время решения является чрезвычайно сложной задачей, ввиду того, что, полагая время задержки за счет обменов пренебрежимо малым, ускорение всех алгоритмов следовало бы считать линейным (что очень далеко от действительности даже при идеально равномерной загруженности процессоров). Нашей же целью является разработка простых и конструктивных оценок эффективности параллельных приложений для задач математической физики.

6 Детальный анализ времени передачи сообщений

Чтобы уточнить оценки (5)–(6) и (9) необходимо провести детальный анализ скорости передачи сообщений с учетом времени инициализации сообщения τ_0 из формулы (1). Опишем тест 2 из работы [7], который исследует эту зависимость.

Описываемый численный эксперимент состоял в передаче сообщения общей длиной в $L_c = M = 2^m$ слов типа “double”, причем сообщение разбивалось на $n_{c,i} = 2^i$ порций каждая длины $L_{c,i} = L_c/n_{c,i} = 2^{m-i}$, $i = 0, ..., m$. Для большей статистической достоверности тест повторялся несколько раз. Таким образом было получено $m+1$ значение $T_{c,i}$ общего времени передачи сообщения.

Выпишем теоретические оценки величины $T_{c,i}$, учитывая время инициализации сообщений τ_0 в формуле (1):

$$T_c(L_c) = T_c n_c = (\tau_0 + \tau_c L_c) n_c, \quad n_c = M/L_c. \tag{11}$$

Для проведения численных экспериментов было выбрано $m = 25$, $M = 2^m = 33554432$, и для очереди “x6core” были получены значения $T_c(L_{c,i})$, для $L_{c,i} = 2^i$, $i = 0, ..., m$. В качестве измеренного времени инициализации сообщения τ_0 в формуле 1 можно взять величину

$$\tau_0 = \max_{i=0,...,m} T_c(L_{c,i})/M = T_c(1)/M = 10.0/M \approx 3.0 \cdot 10^{-7}, \tag{12}$$

а в качестве средней скорости передачи одного числа τ_c можно использовать величину

$$\tau_c = \min_{i=0,...,m} T_c(L_{c,i})/M = 0.10/M \approx 3.0 \cdot 10^{-9}. \tag{13}$$

Значения констант 10.0 и 0.10 были получены из описанного численного эксперимента, проведенного в очереди “x6core”, для простоты, значения констант были округлены. Можно заметить, что для полученных значений отношение τ_0/τ_c составляет 100. Другими словами, остюда можно сделать вывод, что время $T_c(0)$ передачи пустого сообщения будет всего в 2 раза меньше, чем время $T_c(100)$ передачи 100 чисел типа “double”.

На рис. 2 представлены результаты сравнения полученных экспериментальных данных с теоретическими, рассчитанными по формуле (11). Данные измерений являются статистически достоверными: среднестатистическое отклонение для большинства измерений оказывается менее 1-5% для 10 повторений работы описанного теста. На левой части рис. 2 видно, что теоретическая кривая практически совпадает с экспериментальными данными, что свидетельствует о разумности сделанных предположений относительно скорости обменов, несмотря на значительное округление констант (12)–(13).

Интересно заметить, что на правой части рис. 2 заметно некоторое замедление обменов при использовании порции слишком большой длины. Вероятно это связано с размером внутреннего буфера библиотеки MPI, который выбирается при ее инсталляции. Кроме того, при небольших длинах сообщений $L_c < 14$ скорость передачи данных на одном вычислительном узле в большинстве случаев оказывается несколько больше чем на двух, а при $L_c \geq 14$, наоборот. В среднем, различие составляет около 10%. Вероятно, этот эффект связан с особенностями конкретной реализации MPI [3] и архитектурой узлов из очереди “x6core”.

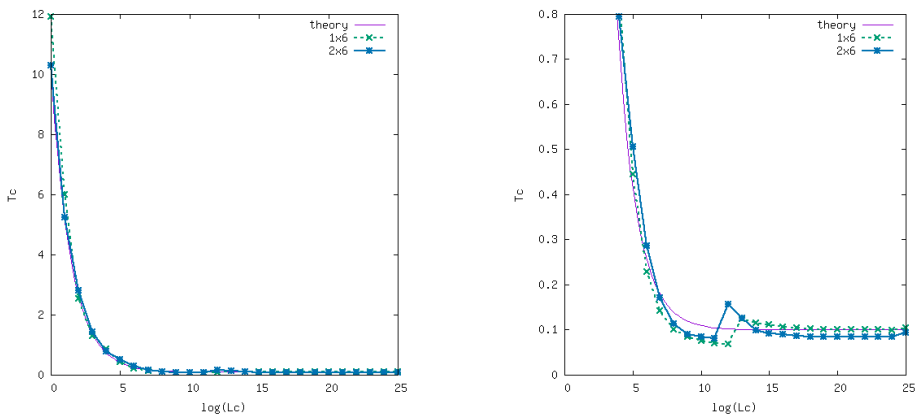


Рис. 2: Общее время пересылки сообщения длиной 2^{25} слов типа “double” порциями длины L_c. Теоретическая оценка из (11) и расчеты в очереди “x6core”.

7 Уточнение оценок и их применение для алгоритмов матемаческой физики

С учетом формулы (1), выпишем время, затрачиваемое на проведение обменов данными:

$$T_c = n_c\tau_0/q + \tau L_c,$$

где L_c, как и раньше, обозначает общую длину всех обменов (в пересчете на один процессор), q – количество слоев перекрытия подобластей (при этом необходимые обмены выполняются один раз на q шагов по времени), а n_c – общее количество обменов на каждом из процессоров. Заметим, что в большинстве алгоритмов n_c обычно не изменяется с ростом количества процессоров p.

Время, затраченное на арифметические операции запишем следующим образом:

$$T_a = (1+Q)\tau_a L_a, \tag{14}$$

где L_a, как и раньше, обозначает общее количество арифметических операций (так же в пересчете на один процессор), новый параметр Q выражает долю увеличения количества арифметических операций, если в алгоритме, для экономии количества или длинны обменов было решено дублировать некоторые операции, выполняемые на других процессорах. Если, как в ранее рассмотренных алгоритмах, дублирования не происходит, то $Q = 0$ и формула (14) переходит в ранее используемую формулу $T_a = \tau_a L_a$, а если дублирование вычислений происходит, то Q принимаем некоторое малое положительное значение, зависящее от особенностей алгоритма.

Используя новые формулы для T_a и T_c при оценке ускорения алгоритма, получим

$$\begin{aligned} S = S(p,\tau_0,Q) &= T(1)/T(p) = T_a(1)/(T_a(p)+T_c(p)) \\ &= \tau_a L_a/((1+Q)\tau_a L_a/p + n_c\tau_0/q + \tau_c L_c/p) \\ &= p/(1+\tau L + Q + pn_c\tau_0/(q\tau_a L_a)). \end{aligned} \tag{15}$$

Здесь как и ранее в формулах (3) и (4) величины τ и L означают характеристики параллельности компьютера и алгоритма, соответственно, а их произведение в знаменателе характеризует снижение ускорения за счет обменов данными; Величина Q, если она отлична от 0, выражает потерю ускорения за счет дублирования вычислений; а отношение задержек на инициализацию n_c сообщений $(n_c\tau_0/q)$ ко времени выполнения арифметических операций на каждом процессоре $(\tau_a L_a/p)$ вносит вклад в снижение ускорения за счет инициализации сообщений.

Величина τ_0, входящая в оценку (15) может быть вычислена достаточно точно, как это сделано в (12). Обратимся снова к алгоритмам математической физики, чтобы оценить остальные неизвестные, входящие в (15). Количество обменов n_c на одну итерацию по времени будет равно или пропорционально количеству подобластей, с которыми граничит конкретный

процессор. В простейшем случае, в обозначениях раздела 3, в зависимости от способа декомпозиции области D, можно положить $n_c = 2D$.

Предположим теперь, что вместо одного слоя соседних ячеек, мы хотим обмениваться сразу q слоями ячеек, для того чтобы выполнять обмены в q раз реже. Это позволит нам сократить время расчета вследствие уменьшения времени инициализации обменов, но приведет к некоторому дублированию вычислений. Общая длина каждого из обменов при этом возрастет в q раз до qL_c, хотя суммарная длина обменов за q итераций по времени останется практически неизменной. Заметим, что дублирующие вычисления будут производиться на $(q-1)$ шагах по времени в точности в тех ячейках, которые участвовали в обменах, при этом количество дополнительных арифметических операций составит $L_c q(q-1)/2$. Таким образом, получена оценка последней неизвестной величины в (15):

$$Q = \frac{q(q-1)}{2}\frac{L_c}{L_a} = \frac{q(q-1)}{2}L. \tag{16}$$

Заметим, что в традиционном случае без дублирования вычислений (т.е. при $q = 1$) мы получаем $Q = 0$. Для дальнейшего анализа этой формулы предположим, что пользователь желает избежать двукратного дублирования вычислений ($Q = 1$) и связанного с этим двукратного падения эффективности вычислений, тогда ему достаточно выбирать значения q не более чем $\sqrt{2/L}$. Вспоминая формулу (6), можно заметить, что такое же падение эффективности можно наблюдать при $\tau L = 1$. Поэтому предыдущее ограничение можно записать еще и таким образом: $q < \sqrt{2\tau}$. Встречающиеся автору компьютеры обычно характеризуются параметром τ из диапозона от 10 до 30 (иногда до 100), поэтому рекомендуется выбирать $q < 5$.

Возвращаясь к оценке ускорения и подставляя полученные величины в (15), получим оценку

$$S = p/(1+(\tau_{ca}+q(q-1)/2)(2-2/p^{1/D})DVC^{-1}p^{1/D}N^{-1/d}+2DC^{-1}q^{-1}pN^{-1}\tau_{0a}), \tag{17}$$

где обозначено $\tau_{ca} = \tau = \tau_c/\tau_a$ и $\tau_{0a} = \tau_0/\tau_a$.

Влияние на ускорение различных величин уже обсуждалось ранее, заметим только, что во второй член знаменателя свой вклад вносит компьютерозависимая величина $\tau = \tau_c/\tau_a$, а в третий член – величина τ_0/τ_a. Отметим также, что второй член возрастает с ростом q, а третий – убывает.

В заключении данного раздела можно заметить, что оценки (9) и (17) легко обобщаются также на случай области в виде прямоугольника или параллелепипеда. В случае областей сложной формы и произвольного распределения данных по процессорам все неизвестные величины $\tau = \tau_c/\tau_a$ и $L = L_c/L_a$ могут быть легко подсчитаны, например, непосредственно в создаваемом приложении перед проведением расчета.

8 Результаты численных экспериментов

В разделе 4 приведено описание вычислительного кластера, используемого для всех сделанных в данной работе расчетов. Перейдем к описанию рашаемой модельной задачи.

В качестве модельной задачи было выбрано решение уравнения теплопроводности в d-мерной кубической области, $d \leq 3$, с одинаковым количеством ячеек в каждом из направлений. Ячейки области были распределены по процессорам в D направлениях, $D \leq d$. Рассматривались перекрытия подобластей в q слоев, что позволяло выполнять только один этап обменов на q итераций по времени. Рассматривалась стандартная конечно-разностная дискретизация уравнения теплопроводности с явной схемой по времени.

На Рис. 3 приведены результаты расчетов для модельной задачи размерности $100p \times 100 \times 100$, $d = 3$, $D = 1$, $p = 64$, с использованием перекрытия подобластей размера $q = 1, ..., 6$. Было выполнено 120 шагов по времени. Можно заметить, что оптимальным по времени счета является использование варианта $q = 3$, что находится в полном соответствии с теоретическими оценками из раздела 7.

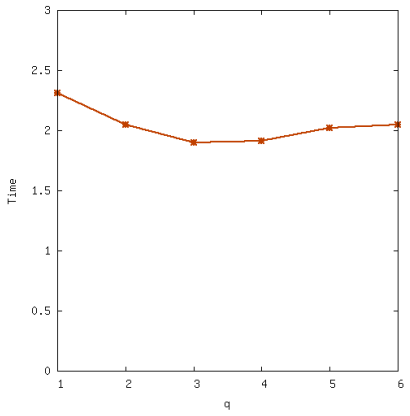


Рис. 3: Время решения в зависимости от размера перекрытия подобластей $q = 1, ..., 6$.

На Рис. 4 и 5 рассматривались задачи размерности $432 \times 432 \times 432$ и $512 \times 512 \times 512$, соответственно. Исследовались все возможные одно-, двух- и трехмерные ($D = 1, 2, 3$) распределения по процессорам, для всех вариантов соотношения процессоров, при которых все подобности получаются одинакового размера. Рассматривался классический вариант с минимальным перекрытием $q = 1$, при этом делалось 120 шагов по времени. Теоретические оценки по формуле (17) дают несколько более оптимистичекий прогноз уско-

рения алгоритма, однако хорошо видно, что использование распределения большей размерности в большинстве случаев обеспечивает большее ускорение, как это и следует из теоретических оценок, полученных в разделах 3 и 7.

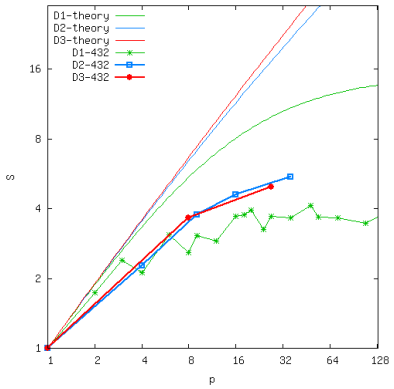


Рис. 4: Ускорение при $D = 1, 2, 3$ для задачи $432 \times 432 \times 432$.

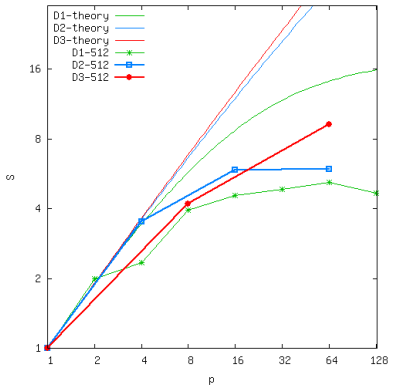


Рис. 5: Ускорение при $D = 1, 2, 3$ для задачи $512 \times 512 \times 512$.

9 Заключение

Получены конструктивные оценки для параллельной эффективности алгоритма, которые включают в себя характеристики параллельности компьютера и параллельности самого алгоритма. Получены также оценки учитывающие время инициализации вычислений, а также возможности группировки сообщений. Для явных схем аппроксимации в задачах математической физики получены оценки, которые кроме геометрических параметров задачи и типа распределения ячеек по процессорам, включают также количество неизвестных функций на ячейку, а также количество арифметических операций, приходящихся на ячейку на одном шаге по времени. Детально проанализирована зависимость скорости передачи сообщений от длины сообщения, а также целесообразность проведения вычислений на фоне ассинхронных обменов. Для модельной задачи теплопереноса проведено прямое сравнение результатов эксперимента с теоретическими оценками параллельной эффективности. Подтвержден вывод о целесообразности использование типа распределения данных по процессорам большей размерности.

Благодарности. Теоретическая часть данной работы была выполнена при финансовой поддержке гранта РНФ 14-11-00190. Экспериментальная часть была частично поддержана грантом РФФИ 17-01-00886.

Список литературы

1. Коньшин И.Н., Модели параллельных вычислений для оценки реального ускорения исследуемого алгоритма. Russian Supercomputing Days: Proc. of the Int. Conf. (September 26-27, 2016, Moscow, Russia). Moscow State University, Moscow, 2016, 269-280. http://2016.russianscdays.org/files/pdf16/269.pdf
2. Konshini I., Parallel computational models to estimate an actual speedup of analyzed algorithm. In: Vol. 687 of Communications in Computer and Information Science (Ed. Vl. Voevodin), Springer, 2017, 304–317
3. MPI: The Message Passing Interface standard. http://www.mcs.anl.gov/research/projects/mpi/ (дата обращения: 15.04.2018)
4. AlgoWiki: Открытая энциклопедия свойств алгоритмов. URL: http://algowiki-project.org (дата обращения: 15.04.2018)
5. Воеводин В.В., Воеводин Вл.В.: Параллельные вычисления. БХВ-Петербург, Петербург, 2002
6. Гергель В.П., Стронгин Р.Г.: Основы параллельных вычислений для многопроцессорных вычислительных систем. Изд-во Нижегородского госуниверситета, Нижний Новгород, 2003
7. Байдин Г.В.: О некоторых стереотипах параллельного программирования. Вопросы атомной науки и техники, Серия: Математическое моделирование физических процессов, 2008, No. 1, 67–75
8. Коньшин И.Н., Параллелизм в вычислительной математике. Международная летняя суперкомпьютерная академия. Трек: Параллельные алгоритмы алгебры и анализа и опыты суперкомпьютерного моделирования. 2012. URL: http://academy2012.hpc-russia.ru/files/lectures/algebra/0704_1_ik.pdf (дата обращения: 15.04.2018)
9. Кластер ИВМ РАН. URL: http://cluster2.inm.ras.ru (дата обращения: 15.04.2018)