Реализация параллельного варианта алгоритма вычисления двумерного быстрого преобразования Фурье по аналогу алгоритма Кули-Тьюки

В.С. Тутатчиков[1]

[1]Сибирский федеральный университет, пр. Свободный 79, Красноярск, Россия, 660041

Аннотация. В настоящее время в цифровой обработке сигналов широко применяется двумерное быстрое преобразование Фурье (БПФ). Обычно оно вычисляется при помощи комбинации одномерных БПФ сначала для каждой строки, затем для каждого столбца. В этом случае алгоритм легко адаптируется для параллельных вычислений. В работе представлен способ распараллеливания алгоритма вычисления двумерного БПФ по аналогу алгоритма Кули-Тьюки, требующего меньшее количество комплексных операций сложения и умножения, чем стандартный способ. Главная сложность распараллеливания указанного алгоритма заключается в том, что двумерный аналог алгоритма Кули-Тьюки выполняется итерационно за s проходов при размере исходного сигнала 2^s*2^s. При этом в каждой из s итераций происходит пересчет всего набора используемых данных. Описаны способы решения данной проблемы при распараллеливании в случае реализации алгоритма на языке программирования C++ с использованием библиотек параллельных вычислений OpenMP и MPI. Приведены результаты численного эксперимента, в которых показано, что при распараллеливании достигается ускорение вычислений до 4 раз по сравнению со стандартным способом вычисления двумерного БПФ.

1. Введение

Широкое распространение в цифровой обработке сигналов получило двумерное быстрое преобразование Фурье (БПФ). Наиболее распространенный способ его вычисления – это применение одномерного БПФ, например алгоритма Кули-Тьюки, сначала для каждой строки двумерного сигнала, потом для каждого столбца [1]. Также существуют способы ускорения вычислений за счет факторизации матрицы преобразования Фурье [2] или перехода в гиперкомплексную алгебру [3, 4]. В данной работе будем рассматривать декомпозицию двумерного массива по обеим координатам по аналогу алгоритма Кули-Тьюки [5]. В [6] показан способ расширения данного алгоритма на n-мерный случай. В двумерном случае аналог алгоритма Кули-Тьюки требует меньшее количество операций комплексного умножения и сложения, чем стандартный способ вычисления БПФ по строкам и столбцам [7].

В работе показаны два способа распараллеливания указанного алгоритма для систем с общей и распределенной памятью, реализованных на языке программирования C++ с использованием библиотек параллельных вычислений OpenMP и MPI, показаны их преимущества и недостатки.

2. Описание последовательного алгоритма

В качестве основного алгоритма возьмем реализацию алгоритма Кули-Тьюки с прореживанием по времени и предварительной двоичной рекурсивной перестановкой [8]. В двумерном случае пусть нам дан сигнал $f(x,y)$ со значениями в комплексном пространстве, $x,y=0,1,...,N-1$, $N=2^s$. Тогда дискретное преобразование Фурье данного сигнала можно задать в виде

$$F(u,v)=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp\left(\frac{2\pi i(ux+vy)}{N}\right) \quad (1)$$

где $u,v=0,1,...,N-1$.

Преобразуем (1) разбиением обоих координат *x* и *y* на четные и нечетные компоненты:

$$F(u,v)=F_{00}(u,v)+F_{10}(u,v)\cdot\exp\left(\frac{2\pi iu}{N}\right)+F_{01}(u,v)\cdot\exp\left(\frac{2\pi iv}{N}\right) \quad (2)$$
$$+F_{11}(u,v)\cdot\exp\left(\frac{2\pi i(u+v)}{N}\right),$$

где

$$F_{ij}(u,v)=\sum_{x_1=0}^{N/2-1}\sum_{y_1=0}^{N/2-1} f(2x_1+i,2y_1+j)\exp\left(\frac{2\pi i(ux_1+vy_1)}{N/2}\right) \quad (3)$$

представляют собой ДПФ подсигнала $f(x,y)$, *i* и *j* принимают значение 0 в случае четного индекса и 1 в случае нечетного индекса координаты *(x,y)*.

Следует заметить, что в (2) мы сделали перестановку элементов $f(x,y)$ для удобства вычисления, разбив их на четыре группы: $F_{00}(u,v)$ - попали все четные по обеим координатам элементы, $F_{10}(u,v)$ - попали все четные по первой координате и нечетные по второй координате, $F_{01}(u,v)$ - все нечетные по первой координате и четные по второй координате, $F_{11}(u,v)$ - все нечетные по обеим координатам.

Выражение (2) можно упростить, если разбить координаты *u* и *v* в оставшихся экспонентах $\exp\left(\frac{2\pi iu}{N}\right)$ и $\exp\left(\frac{2\pi iuv}{N}\right)$ на две группы таким образом, чтобы разница индексов между связанными элементами в группах составляла $N/2$ [9]:

$$F(u,v)=F(u_1,v_1)+F(u_1+N/2,v_1)+F(u_1,v_1+N/2)+F(u_1+N/2,v_1+N/2), \quad (4)$$

где $u_1,v_1=0,1,...,N/2-1$.

Тогда экспоненты в (2) для элементов $u_1+N/2$ и $v_1+N/2$ примут вид:

$$\exp\left(\frac{2\pi i\cdot(u_1+N/2)}{N}\right)=\exp\left(\frac{2\pi iu_1}{N}\right)\cdot\exp(\pi i)=-\exp\left(\frac{2\pi iu_1}{N}\right),$$
$$\exp\left(\frac{2\pi i\cdot(v_1+N/2)}{N}\right)=\exp\left(\frac{2\pi iv_1}{N}\right)\cdot\exp(\pi i)=-\exp\left(\frac{2\pi iv_1}{N}\right). \quad (5)$$

Теперь выражение (2) с учетом (4) – (5) примет вид:

$$F(u_1,v_1)=F_{00}(u_1,v_1)+F_{10}(u_1,v_1)\cdot\exp\left(\frac{2\pi iu_1}{N}\right)+F_{01}(u_1,v_1)\cdot\exp\left(\frac{2\pi iv_1}{N}\right)$$
$$+F_{11}(u_1,v_1)\cdot\exp\left(\frac{2\pi i(u_1+v_1)}{N}\right),$$
$$F(u_1+N/2,v_1)=F_{00}(u_1+N/2,v_1)-F_{10}(u_1+N/2,v_1)\cdot\exp\left(\frac{2\pi iu_1}{N}\right)$$

$$+ F_{01}(u_1 + N/2, v_1) \cdot \exp\left(\frac{2\pi i v_1}{N}\right) - F_{11}(u_1 + N/2, v_1) \cdot \exp\left(\frac{2\pi i (u_1 + v_1)}{N}\right),$$

$$F(u_1, v_1 + N/2) = F_{00}(u_1, v_1 + N/2) + F_{10}(u_1, v_1 + N/2) \cdot \exp\left(\frac{2\pi i u_1}{N}\right) \quad (6)$$

$$- F_{01}(u_1, v_1 + N/2) \cdot \exp\left(\frac{2\pi i v_1}{N}\right) - F_{11}(u_1, v_1 + N/2) \cdot \exp\left(\frac{2\pi i (u_1 + v_1)}{N}\right),$$

$$F(u_1 + N/2, v_1 + N/2) = F_{00}(u_1 + N/2, v_1 + N/2) - F_{10}(u_1 + N/2, v_1 + N/2) \cdot \exp\left(\frac{2\pi i u_1}{N}\right)$$

$$- F_{01}(u_1, v_1) \cdot \exp\left(\frac{2\pi i v_1}{N}\right) + F_{11}(u_1, v_1) \cdot \exp\left(\frac{2\pi i (u_1 + v_1)}{N}\right).$$

Формула (6) задает бабочку разбиения двумерного быстрого преобразования Фурье по аналогу алгоритма Кули-Тьюки. Она связывает между собой четыре элемента сигнала $f(x, y)$ с определенной четностью и четыре получающихся элемента спектра Фурье, разбитые на парные индексы *u* и *v* относительно половины длины сигнала *N*.

Используемая схема разбиения (2) – (6) исходного сигнала $f(x, y)$ на четыре подсигнала $F_{ij}(u, v)$, $i, j = 0,1$ представляет лишь один шаг рекурсии в аналоге алгоритма Кули-Тьюки. На следующем этапе мы берем полученные подсигналы $F_{ij}(u, v)$ и применяем к каждому из них еще один шаг рекурсии до тех пор, пока не останется подсигнал $F_{ij}^{k}(u, v)$, состоящий из 2×2 элементов, дискретное преобразование Фурье которого вычисляем непосредственно.

Число операций сложения и умножения комплексных чисел, необходимое для вычисления двумерного БПФ сигнала $f(x, y)$ с $N \times N$, $N = 2^s$ отсчетами приведено в таблице 1. Из нее видно, что описанный алгоритм требует меньшее количество операций умножения комплексных чисел, чем стандартный способ вычисления двумерного (2D) БПФ комбинацией одномерных алгоритмов Кули-Тьюки сначала по строкам, затем по столбцам [10].

Таблица 1. Сравнение числа операций комплексных чисел.

Операция	2D БПФ по строкам и столбцам	2D БПФ по аналогу алгоритма Кули-Тьюки
Умножение	$N^2 \log_2 N$	$\frac{3}{4} N^2 \log_2 N$
Сложение	$2N^2 \log_2 N$	$2N^2 \log_2 N$

В приведенном алгоритме можно перейти от рекурсивного построения алгоритма к итеративному, в котором первый шаг итерации вычисления БПФ соответствует последнему шагу рекурсивного разложения (6), а последний шаг итерации соответствует первому шагу рекурсии [6]. На каждом шаге итерации происходит вычисление двумерных бабочек преобразования Фурье для разного набора элементов.

Рассмотрим подробнее перестановку элементов $f(x, y)$ при разбиении на четные и нечетные элементы на примере сигнала, состоящего из 4×4 отсчетов (рисунок 1). Большой квадрат слева представляет собой исходный сигнал, в котором каждый отсчет пронумерован двойным индексом *i* - номер строки, *j* – номер столбца.

При первом шаге разбиения сигнал делится на четыре квадрата 2×2 отсчета каждый (разбиение отмечено двумя длинными синими линиями в левой части рисунка). При этом в верхний левый квадрат помещаются элементы с четными индексами по обеим координатам (в нашем случае индексы 0 и 2); в верхний правый помещаются элементы четные по первой координате и нечетные по второй (*i* = 0,2, *j* = 1,3); в нижний левый помещаются нечетные по

первой координате и четные по второй (i = 1,3, j = 0,2); в нижний правый помещаются нечетные по обеим координатам (с индексами 1 и 3). Результат разбиения представлен на рисунке 1 в центре. На этом первый шаг перестановки заканчивается.

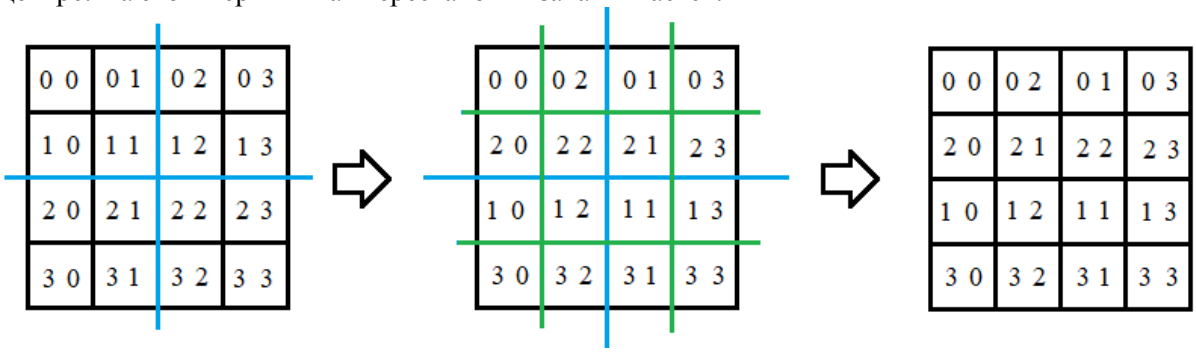

Рисунок 1. Схема разбиения на четные и нечетные элементы.

На втором шаге рекурсии каждый из полученных на прошлом этапе квадратов тоже делится на четыре квадрата меньшего размера с 1×1 отсчетами (деление обозначено выступающими зелеными линиями на центральном квадрате). Затем в каждом из квадратов будет происходить разбиение на четные и нечетные индексы (по новой нумерации в пределах полученного квадрата 2×2), но в данном случае их элементы уже стоят в правильном порядке. В этом случае завершаем перестановку.

В общем случае полученную перестановку можно задать следующим образом [2]. Пусть j – целое число из множества $J=\{0,1,2,...,2^s-1\}$ представимо в двоичное системе в виде $j_{s-1}2^{s-1}+...+j_1 2+j_0$ где $j_i=0,1$ для всех $i=0,...,s-1$. Вектор $(j_{s-1},...,j_1,j_0)_2$ будем называть двоичным кодом числа j. Сопоставим с числом j число $j^1\in J$, которое задается двоичным кодом $(j_0,j_1,...,j_{s-1})_2$. Перестановка $rev_s(j)=j^1$ множества J называется реверсной:

$$rev_s(j)=j_0 2^{s-1}+...+j_{s-2}2+j_{s-1} \quad (7)$$

В ходе выполнения разложения (6) элементы в строках и столбцах $f(x,y)$ будут переставлены в соответствии с рекурсивной перестановкой (7), следовательно, для получения корректного сигнала $F(u,v)$ следует применить перестановку (7) для индексов элементов строк и столбцов. В силу ее обратимости $rev_s(rev_s(j))=j$ можно предварительно применить указанную перестановку к элементам $f(x,y)$ [6]. Тогда при вычислении первой итерации аналога алгоритма Кули-Тьюки бабочка (6) использует соседние элементы, на второй итерации использует элементы, стоящие через один друг от друга. На k-ом шаге итерации бабочка использует четверку элементов, отстоящих друг от друга на 2^{k-1} относительно левого верхнего выбранного алгоритма. В каждом шаге итерации мы перебираем все возможные четверки элементов построчно. Например, если $f(x,y)$ содержит $2^3\times2^3$ отсчетов, то тогда схему выбора элементов двумерной бабочки преобразования Фурье можно представить на рисунке 2, где левый квадрат – это первый шаг итерации, а правый – третий.

Данный вариант построения последовательного алгоритма двумерного БПФ позволяет перейти к схеме распараллеливания для систем с общей памятью.

3. Параллельный вариант для систем с общей памятью

Переход от рекурсивного разложения к циклу с итерациями по $k=1,...,s$ при вычислении двумерного БПФ по аналогу алгоритма Кули-Тьюки позволяет эффективно распараллелить вычисления внутри каждой итерации.

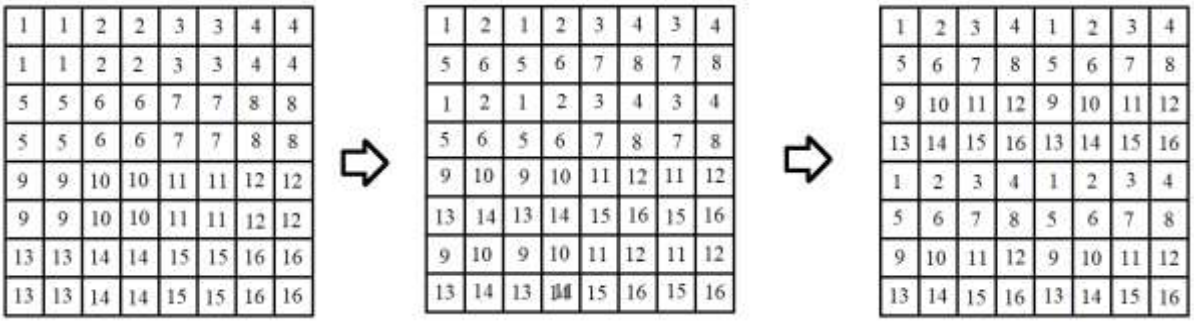

Рисунок 2. Схема разбиения элементов для вычисления бабочки внутри итерации.

При старте главный поток получает данные из файла, выполняет их реверсивную перестановку (7) и записывает результат в массив $f(x, y)$. Данный массив поступает в общую память для всех потоков, затем начинается параллельная секция. Главный процесс создает дополнительные потоки, присваивает им идентификационные номера. Далее в зависимости от номера и шага итерации каждый процесс определяет свой непересекающийся набор строк $f(x, y)$, в пределах которых он будет вычислять двумерные бабочки для выбранных данных, а затем записывать результат итерации в соответствующие строки массива $f(x, y)$. По окончании обработки всех строк массива внутри итерации происходит остановка всех процессов для того, чтобы получить актуальную общую версию массива $f(x, y)$ на данном шаге. После этого происходит переход к следующему шагу из s возможных. После завершения всех итераций получаем Фурье-образ исходного сигнала в массиве $f(x, y)$ [11]. Результаты численного эксперимента реализации такого алгоритма приведены в разделе 5, в котором достигается ускорение до 4 раз по сравнению с последовательным вариантом.

Для примера возьмем сигнал $f(x, y)$, состоящий из $2^3 \times 2^3$ отсчетов, следовательно, двумерное БПФ можно вычислить за 3 итерации. Схема разбиения данных на первой итерации представлена на рисунке 3 слева, для последней итерации – правая часть рисунка. При этом синим цветом показаны строки, которые обрабатывает главный процесс, а белым остались строки, которые обрабатывает дополнительный поток при разбиении на два параллельных потока.

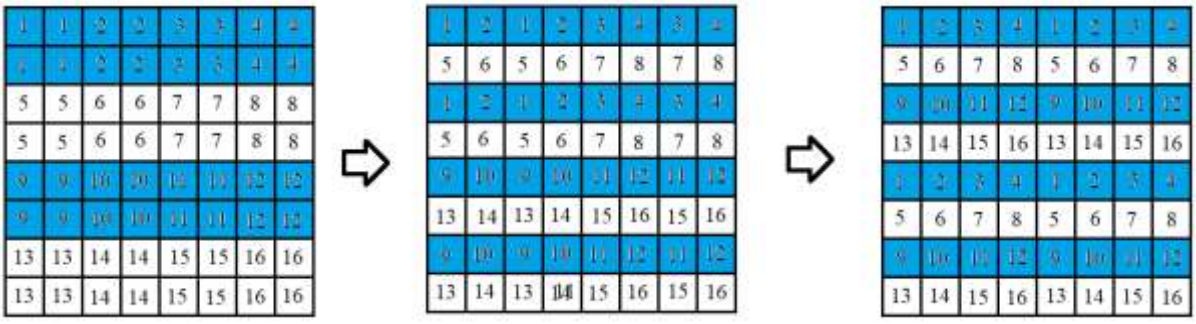

Рисунок 3. Схема распараллеливания данных для систем с общей памятью.

Как видно из рисунка, потоки получают данные по мере выполнения вычислений: сначала главный поток получил строки данными для бабочки 1, а второй поток строки с данными для бабочки 5. Потом, например, главный поток закончил свои вычисления быстрее и получил следующий набор строк с бабочкой 9. Дополнительный поток забрал оставшиеся строки в первой итерации. Аналогичный способ разделения данных присутствует и в двух оставшихся итерациях, но номера связанных строк будут другие. При этом результат вычислений каждый поток запишет в обрабатываемые строки, таким образом будут вычислены двумерные бабочки БПФ для всех строк массива без пересечений.

Преимуществом данного способа распараллеливания является простота реализации. Но при этом происходят дополнительные временные издержки при обращении каждого потока в общую память и синхронизацию данных после завершения очередной итерации. От этих недостатков можно избавиться, если проанализировать процесс вычислений внутри каждого потока. Например, на первом шаге каждый поток получает бабочку для соседних элементов, на втором шаге для элементов, стоящих через один друг от друга. И только на последнем шаге каждый потом получает элементы, стоящие через половину длины сигнала *s* друг от друга.

Можно оптимизировать распределение данных для двух потоков, если первую половину строк отнести к главному потоку, а вторую половину строк к дополнительному. Тогда в первых двух шагах итерации каждый из потоков сможет работать со своим набором данных независимо друг от друга. Следовательно, не надо будет делать синхронизацию потоков после окончанию итераций, а сами отправляемые данные можно будет передать в виде локальной переменной, что тоже увеличит скорость доступа к данным. В данном случае только перед началом третьей итерации потребуется произвести синхронизацию потоков и снова собрать результат в глобальный массив $f(x, y)$. А дальше можно продолжить распараллеливание старым способом, либо произвести перестановку строк таким образом, чтобы используемые в итерации данные попадали в два независимых набора локальных переменных, выполнить шаг итерации, а потом произвести обратную перестановку, чтобы получить корректные значения Фурье-образа. Модернизированная схема распараллеливания для систем с общей памятью поможет ускорить вычисления при больших значениях числа итераций *s*.

3. Параллельный вариант для систем с распределенной памятью

В случае распараллеливания систем с распределенной памятью применяется аналогичный алгоритм распараллеливания. Но при этом появляются существенные отличия. Во-первых, в данном случае отсутствует общая память, следовательно весь массив $f(x, y)$ хранится только в главном процессе. При старте параллельной секции главный процесс отсылает всем остальным две связанные строки данных. Затем каждый процесс вычисляет двумерные бабочки БПФ в них и отправляет результат главному процессу. При отправке и получении данных происходит синхронизация между потоками, то есть в этот момент вычисления останавливаются. Затем главный процесс собирает данные потоков, обновляет массив $f(x, y)$ и либо отправляет потокам следующий набор строк, либо переходит к следующей итерации [12, 13].

В данном случае на время работы программы начинает оказывать влияет время пересылки данных. Например, если производить вычисления на одном процессоре с несколькими потоками, то время пересылки будет ограничено только скоростью доступа к оперативной памяти и в целом совпадает с результатами распараллеливания для систем с общей памятью (смотри раздел 4). Однако если запустить вычисления на двух разных процессорах с разными потоками, то появляются существенные временные задержки при пересылке данных по локальной сети или шине данных между процессорами.

Можно частично скомпенсировать эти задержки модернизацией алгоритма распараллеливания аналогично случаю систем с общей памятью. Для примера возьмем сигнал $f(x, y)$ с $2^3 \times 2^3$ отсчетами и два параллельных процессора. В первых двух итерациях разобьем набор строк пополам, оставим верхнюю половину для главного процесса и отправим нижнюю во второй процесс. Тогда первые два шага итерации вычисления двумерного БПФ по аналогу алгоритма Кули-Тьюки процессы будут выполнять вычисления независимо друг от друга без задержек на синхронизацию и пересылку данных. Затем после завершения второй итерации главный процесс получает данные от второго, формирует новое значение $f(x, y)$ и переходит к третьему шагу итерации. При этом главный процесс делает частичную перестановку данных таким образом, чтобы разбить $f(x, y)$ на два блока строк, каждый их которых будет вычисляться независимо, отправляет второй блок вспомогательному потоку (рисунок 4).

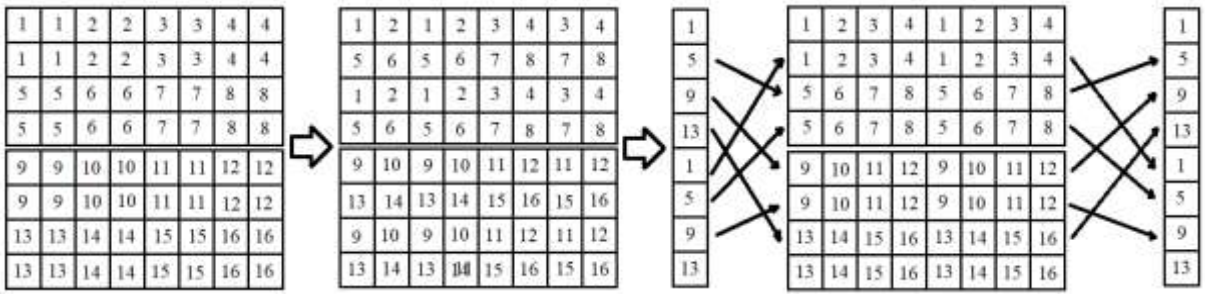

Рисунок 4. Схема распараллеливания данных для систем с распределенной памятью.

Далее выполняется шаг итерации, после чего дополнительный процесс отправляет результат главному потоку, который выполняет обратную частичную перестановку и получает результат в виде Фурье-образа $F(u,v)$. Таким образом вместо постоянно пересылки данных в виде пары строк во время работы осуществляется только четыре соединения между процессами, что позволит заметно ускорить вычисления для систем с распределенной памятью, выполняемые на разных процессорах.

В следующем разделе рассмотрим результаты сравнения работы указанных алгоритмов.

4. Результаты численного эксперимента

Для тестирования последовательного алгоритма вычисления двумерного БПФ по аналогу алгоритма Кули-Тьюки была написана программа на языке программирования C++. Параллельный вариант алгоритма был разработан при помощи библиотеки параллельных вычислений OpenMP для систем с общей памятью и библиотеки MPI для систем с распределенной памятью. Тестирования проводилось на одном и нескольких узлах кластера суперкомпьютера СФУ с характеристиками IBM HS21 XM Xeon Quad core E5450 3.0 GHz, 64 Gb оперативной памяти [14]. Средние результаты тестирования в секундах приведены в таблице 2.

Таблица 2. Результаты численного эксперимента в секундах.

Размер сигнала	Число процессов	2D БПФ по строкам и столбцам	2D БПФ КТ с общей памятью	2D БПФ КТ с распред. памятью, 1 узел	2D БПФ КТ с распред. памятью, 2 узла	2D БПФ КТ с распред. памятью, 4 узла
1024*1024	1	0,490	0,230	0,310	0.310	0.300
	2	0,310	0,153	0,310	1.170	1.150
	4	0,220	0,126	0,270	0.830	1.010
	8	0,180	0,129	0,270	0.630	0.840
	16	0,180	0,148	0,350	96.300	0.730
2048*2048	1	2,300	0,920	1,330	1.340	1.320
	2	1,550	0,885	1,260	3.740	3.720
	4	1,060	0,657	0,950	2.800	3.610
	8	0,840	0,490	0,850	2.280	3.170
	16	0,850	0,657	1,000	-	71.360
4096*4096	1	9,880	4,496	5,850	5.840	5.840
	2	6,240	4,490	4,590	13.020	12.970
	4	4,370	3,187	3,590	10.500	14.040
	8	3,430	2,578	3,060	36.800	120.610
	16	3,740	2,195	3,390	-	189.220
8192*8192	1	43,210	16,003	25,190	25.300	26.530
	2	26,990	14,413	19,160	49.810	52.320
	4	18,550	12,708	14,870	41.810	88.310
	8	14,520	9,194	13,130	36.800	228.750
	16	14,090	12,345	11,700	-	-

В таблице 2 колонка 7 обозначает результат тестирования двумерного БПФ по аналогу алгоритма Кули-Тьюки для систем с распределенной памятью при запуске на одном физическом узле и создании параллельных потоков на нем же. Колонки 8 и 9 показывают время работы алгоритма при запуске на двух и четырёх физических узлах, передача данных между которыми происходит по сети.

5. Заключение

По таблице 7 можно сделать вывод, что алгоритм вычисления двумерного БПФ по аналогу алгоритма Кули-Тьюки без распараллеливания выполняется примерно в 1,7 раз быстрее стандартного алгоритма вычисления 2D БПФ по строкам и столбцам. В случае распараллеливания для систем с общей памятью ускорение вычислений достигает до 4 раз. При запуске алгоритма на системах с распределенной памятью на одном узле ускорение вычислений в некоторых случаях достигает до 4 раз. Но при этом при запуске алгоритма на разных узлах заметно существенно понижение скорости работы до трёх раз. Это объясняется тем, что в первоначальной версии алгоритма распараллеливания для систем с общей памятью присутствует по парная рассылка строк для вычислений между процессами. Для этой рассылки приходится каждый раз устанавливать связь между главным процессом и вспомогательными, а затем передавать данные по сети между узлами, что значительно медленнее непосредственного обращения в оперативную память. В работе предложен способ модификации алгоритма, заключающийся в увеличении размера блока пересылаемых данных и значительного снижении числа пересылок между процессами.

Предложенный алгоритм может широко применяться в задачах обработки снимков дистанционного зондирования Земли [15] или теории антенных решеток [16].

6. Благодарности

Автор выражает благодарность за финансовую поддержку Красноярскому краевому фонду поддержки научной и научно-технической деятельности и Институту космических и информационных технологий Сибирского федерального университета. Также хочу выразить благодарность своему научному руководителю Носкову Михаилу Валериановичу за ценные советы при планировании исследования и рекомендации по оформлению статьи.

7. Литература

[1] Dudgeon, D.E. Multidimensional Digital Signal Processing / D.E. Dudgeon, R.M. Mersereau – Prentice Hall, 1983. – 400 p.
[2] Малоземов, В.Н. Основы дискретного гармонического анализа / В.Н. Малоземов, С.М. Машарский – Издательство «Лань», 2012. – 304 с.
[3] Алиев, М.В. Алгоритмы двумерного гиперкомплексного дискретного преобразования Фурье / М.В. Алиев, А.М. Белов, А.В. Ершов, М.А. Чичева // Компьютерная оптика. – 2004. – Т. 26. – С.101-104.
[4] Алиев, М.В. Многомерное гиперкомплексное ДПФ: параллельный подход / М.В. Алиев, М.А. Чичева // Компьютерная Оптика. – 2005. – Т. 27. – С. 135-137.
[5] Блейхут, Р. Быстрые алгоритмы цифровой обработки сигналов – М.: Мир, 1989. – 448 с.
[6] Старовойтов, А.В. О многомерном аналоге алгоритма Кули-Тьюки // Вестник Сибирского государственного аэрокосмического университета. – 2010. – Т. 1, № 27. – С. 69-72.
[7] Tutatchikov, V.S. Two-dimensional fast Fourier transform // Proceeding of 11th International Forum on Strategic Technology (IFOST), 2016. – P. 495-498.
[8] Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер – Москва: Техносфера, 2006. – 856 с.
[9] Noskov, M.V. Modification of a Two-Dimensional Fast Fourier Transform by the Analog of th Cooley-Tukey Algorithm for a Rectangular Signal / M.V. Noskov, V.S. Tutatchikov // Pattern Recognition and Image Analysis. – 2015. – Vol. 25(1). – P. 81-83.

[10] Tutatchikov, V.S. Calculating the n-Dimensional Fast Fourier Transform / V.S. Tutatchikov, O.I. Kiselev, M.V. Noskov // Pattern Recognition and Image Analysis. – 2013. – Vol. 23(3). – P. 429-433.
[11] Noskov, M.V. Parallel Version n-Dimensional Fast Fourier Transform Algorithm Analog of the Cooley-Tukey Algorithm / M.V. Noskov, V.S. Tutatchikov // Proceeding of the 5TH International Workshop on Image Mining. Theory and Applications – Berlin, Germany, 2015. – P. 114-117.
[12] Носков, М.В. Применение параллельного аналога алгоритма Кули–Тьюки двумерного быстрого преобразования Фурье для цифровой обработки снимков дистанционного зондирования земли / М.В. Носков, В.С. Тутатчиков, М.П. Лапчик, М.И. Рагулина // Региональные проблемы дистанционного зондирования Земли: Материалы V Междунар. науч. конф. – Красноярск: Сиб. федер. ун-т, 2018. – С. 153-156.
[13] Noskov, M. Application of parallel version two-dimensional fast Fourier transform algorithm analog of the Cooley-Tukey algorithm for digital image processing of satellite data / M. Noskov, V. Tutatchikov, M. Lapchik, M. Ragulina, T. Yamskikh // E3S Web of Conferences. – 2019. – Vol. 75. – P. 01012.
[14] Суперкомпьютерный комплекс СФУ [Электронный ресурс]. Режим доступа: http:// cluster.sfu-kras.ru/.
[15] Носков, М.В. Применение аналога алгоритма Кули–Тьюки двумерного быстрого преобразования Фурье для частотной фильтрации снимков дистанционного зондирования земли в фиксированном формате 4К / М.В. Носков, В.С. Тутатчиков // Региональные проблемы дистанционного зондирования Земли: Материалы VI Междунар. науч. конф. – Красноярск: Сиб. федер. ун-т, 2018. – С. 131-133.
[16] Кашкин, В.Б. О конфигурациях узлов кубатурных формул в теории антенных решеток / В.Б. Кашкин, О.И. Киселев, М.В. Носков, В.С. Тутатчиков // Известия высших учебных заведений. Физика. – 2012. – Т. 55, № 9-2. – С. 68-69.

Implementation of a parallel version of the algorithm for calculating a two-dimensional FFT using an analog of the Cooley-Turkey algorithm

V.S. Tutatchikov[1]

[1]Siberian Federal University, Svobodny pr. 79, Krasnoyarsk, Russia, 660041

Abstract. Currently, two-dimensional fast Fourier transform (FFT) is widely used in digital signal processing. It is usually calculated using a combination of one-dimensional FFTs, first for each row, then for each column. In this case, the algorithm is easily adapted for parallel computing. The paper presents a method of parallelizing the algorithm for calculating a two-dimensional FFT using an analogue of the Cooley-Tukey algorithm, which requires fewer complex operations of addition and multiplication than the standard method. The main difficulty in parallelizing this algorithm is that the two-dimensional analog of the Cooley-Tukey algorithm is iteratively performed in s passes with the size of the initial signal 2 ^ s * 2 ^ s. Moreover, in each of s iterations, the entire set of used data is recalculated. Methods for solving this problem during parallelization are described in the case of implementing an algorithm in the C ++ programming language using OpenMP and MPI parallel computing libraries. The results of a numerical experiment are presented, in which it is shown that parallelization achieves an acceleration of computations up to 4 times in comparison with the standard method for calculating two-dimensional FFT.