ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №1(32), 2017, с. 47–62

УДК 519.68

Д. Ю. Князьков

Эффективный расчет двумерного БПФ на однородном или гетерогенном вычислительном кластере

Аннотация. Рассмотрена задача осуществления двумерного БПФ матрицы на суперкомпьютере. Исследована зависимость времени выполнения БПФ от размера матрицы для суперкомпьютеров МВС-100К, МВС-10П и HybriLIT. Описан метод балансировки вычислительной нагрузки между вычислениями на процессоре и видеокарте при использовании гетерогенного кластера. На примере видеокарты TESLA K40 показано, что время, необходимое для перемещения данных, близко времени, требуемому для осуществления двумерного БПФ на графическом вычислителе, а само время расчета в 48 раз меньше времени счета на двухпроцессорном узле.

Ключевые слова и фразы: HPC-вычисления, суперкомпьютерные вычисления, быстрое преобразование Фурье, вычисления на графических процессорах.

Введение

Алгоритм быстрого преобразования Фурье широко применяется в области обработки сигналов, различных вычислительных методах и приложениях — практически везде, где используется преобразование Фурье, по сравнению с которым он позволяет снизить количество требуемых арифметических операций с N^2 до $N\log(N)$ [**1**, **2**]. В настоящей работе изучается вопрос быстродействия алгоритма двумерного быстрого преобразования Фурье (БПФ) при изменении размера матрицы от небольшого (по сравнению с доступной для вычислительного ядра памяти) до такого размера, когда эта матрица занимает оперативную память всех доступных для счета узлов суперкомпьютера. При этом считается, что в случае, если на узел помещается более одной матрицы интересующего нас размера, обработка таких матриц происходит одновременно, причем требуется оптимизировать эффективность расчета, а не время работы алгоритма. Выполнение последнего условия дает

Работа выполнена при поддержке РФФИ, проект №16-31-60096 мол-а-дк.

возможность использовать полученные данных производительности для оценки времени работы алгоритмов, где БПФ является элементарным шагом, и требуется выполнять большое количество однотипных двумерных БПФ. Такие требования естественны, например, для алгоритмов, использующих секционирование свертки, когда требуется на расчетной сетке с очень большим количеством узлов вычислить свертку

$$I(i,j) = \sum_{k,l=0}^{N-1} \varphi(k,l) S(i-k, j-l), \quad i,j = 0, ..., H-1.$$

Если соответствующие матрицы не помещаются в ОЗУ вычислителя, область вычисления БПФ разделяется на отдельные блоки, осуществляется БПФ этих блоков, а затем результаты объединяются [3]. Для вычисления каждой отдельной свертки используется БПФ [2]:

$$\mathbf{I} = FFT^{-1}[FFT(\varphi) \otimes FFT(\mathbf{K})]. \tag{1}$$

В качестве примера рассмотрим алгоритм большого пиксела расчета распространения электромагнитных полей [4, 5], с помощью которого вычисление поля $I(x,y)$ от объекта $\varphi(\xi,\eta)$ удается свести к расчету суммы

$$I(x,y) = \iint\limits_{\mathrm{O}} \varphi(\xi,\eta) K(x,y,\xi,\eta) d\xi d\eta = \sum_{k,l=0}^{N-1} \varphi(k,l) \iint\limits_{\square_{kl}} K(x,y,\xi,\eta) d\xi d\eta,$$

что позволяет снизить количество требуемых для расчета операций с

$$\Upsilon_{\text{БПФ}} = 30(1+k)^2 10^2 \frac{a^2}{\lambda^2} \log_2 \frac{10(1+k)a}{\lambda} \tag{2}$$

до

$$\Upsilon_{\text{БП}} = 30(1+k)^2 \frac{a^2}{\sigma^2} \log_2 \frac{(1+k)a}{\sigma}, \tag{3}$$

где a — размер излучающего объекта, λ — длина волны излучения, k — отношение размера области, в которой излучение регистрируется к размеру излучающего объекта, σ — размер расчетного «пиксела». Способ секционирования свертки (3), приводящий к вычислительной сложности

$$\Upsilon_{\text{БПЗ}} = (80k^2 + 40) \frac{a^2}{\sigma^2} \log_2 \frac{2a}{\sigma} \tag{4}$$

был предложен и реализован в [4]. Метод сдвига расчетных сеток для случая, когда области задания и расчета поля имеют различный шаг, был предложен в [5] и исследован в [6].

Типовой расчет по алгоритму (4) занимал около 100 минут на 640 вычислительных ядрах (80 вычислительных узлах) суперкомпьютера МВС-100К МСЦ РАН [4]. При этом осуществлялось 201 двумерное БПФ распределенно хранящейся матрицы размером 79999×79999. Каждое такое БПФ занимало 29 секунд, а количество используемых вычислительных узлов было минимальным, чтобы в их ОЗУ могла бы разместиться такая матрица. Расчеты для таких и больших размеров расчетных сеток актуальны для нужд микролитографии [7].

Формулы для количества операций при вычислении секционированной свертки имеют сходную структуру и основной множитель, отвечающий за операцию двумерного БПФ, когда нужно сделать БПФ для всех N строк матрицы размером $N \times N$, транспонировать матрицу и еще раз сделать БПФ для всех N строк, равен $CN^2 \log(N)$. Поскольку здесь необходимо транспонирование, время работы программы будет зависеть уже не столько от количества требуемых операций, сколько от скорости работы памяти и межузловой коммуникационной сети суперкомпьютера. В настоящей работе предлагается определить эмпирическую зависимость времени работы двумерного БПФ от размера обрабатываемой на суперкомпьютере матрицы для получения адекватных оценок времени работы алгоритмов, использующих двумерное БПФ, например, алгоритмов типа (2)–(4).

Следует отметить, что использование спецвычислителей, например, представляющих из себя специальные микросхемы [8, 9] или реконфигурируемые системы [10], позволяет существенно сократить время выполнения БПФ, однако такие устройства малодоступны и дороги. Все расчеты в настоящей работе ведутся на общедоступных системах кластерного типа. Не рассматриваются возможности, связанные с использованием жестких дисков и других запоминающих устройств — считается, что все данные находятся в оперативной памяти узлов кластера.

1. Двумерное распределенное БПФ на однородном вычислительном кластере

Целью настоящей работы является исследование времени работы алгоритма БПФ для различных размеров обрабатываемого массива:

когда все данные находятся в границах памяти вычислительного ядра, так и когда данные хранятся распределенно на нескольких вычислительных узлах. Поскольку БПФ рассматривается как элементарная операция в рамках какого-либо внешнего алгоритма, считается, что одновременно на всех доступных узлах вычислительного комплекса проводятся такие же БПФ. Поэтому нас будет в первую очередь интересовать эффективность проводимых вычислений, а не время работы алгоритма, тогда как понятно, что (как правило) увеличение количества используемых вычислительных мощностей будет приводить к уменьшению времени работы.

Ранее была проанализирована зависимость эффективности расчета двумерного БПФ от количества используемых вычислительных ядер суперкомпьютера МВС-100К МСЦ РАН [4]. Было показано, что эффективность расчета сильно падает при увеличении количества используемых вычислительных ядер (ВЯ) от 1 до 8 в рамках одного вычислительного узла (ВУ), далее, при увеличении количества ВУ до 8, эффективность падает, но гораздо медленнее, а при дальнейшем увеличении количества используемых узлов скорость падения эффективности опять увеличивалась. При этом эффективность монотонно падает на всем исследованном участке.

Один из этапов двумерного БПФ — это транспонирование обрабатываемой матрицы — операция, требующая больших объемов межпроцессорной коммуникации. Поскольку, как правило, время обмена между ВУ гораздо больше времени осуществления вычислений, при увеличении размера задачи следует хранить вычислительный массив на одном ВУ до тех пор, пока он не займет всю доступную на узле память. Таким образом, суммируя вышесказанное, можно сформулировать два условия эффективного использования вычислительных ресурсов:

(*1*) матрица должна храниться на минимально возможном количестве ВУ;
(*2*) когда несколько матриц размещены в ОЗУ одного ВУ, для их обработки необходимо испрользовать минимально возможное количество ВЯ.

Обозначим N_n — количество ВУ, N_c — количество ВЯ на одном ВУ, M_0 — емкость памяти вычислительного ядра (рассматривается случай хранения комплексных чисел двойной точности), N_p — общее количество вычислительных ядер, W — количество одновременно работающих групп процессов на одном ВУ, $N \times N$ — размер одной матрицы, для которой выполняется БПФ.

В соответствии с приведенными выше условиями эффективности, в зависимости от размера обрабатываемой матрицы, необходимо использовать один из трех сценариев выполнения БПФ:

(*1*) Размер матрицы таков, что количество матриц, помещающихся в оперативной памяти вычислительного узла, больше количества вычислительных ядер этого узла:

$$0 < N \leq \sqrt{\frac{M_0}{N_c}}.$$

В этом случае каждая матрица обрабатывается одним ВЯ: $p = 1$, $W = N_c$. Среднее время одного БПФ составит

$$T_{mean} = \frac{T(N,1)}{N_c N_n},$$

где $T(N,p)$ — время осуществления одного двумерного БПФ матрицы размера $N \times N$ при использовании для вычислений p вычислительных ядер, а T_{mean} — среднее время БПФ одной матрицы, то есть если необходимо сделать БПФ для L матриц, где L достаточно велико, общее время работы составит

$$T_{all} = T_{mean} \cdot L.$$

(*2*) Размер матрицы таков, что количество матриц, находящихся в оперативной памяти одного ВУ, уже меньше количества ВЯ одного узла N_c, и некоторые (или все) матрицы обрабатываются двумя или более ВЯ:

$$\sqrt{\frac{M_0}{N_c}} < N < \sqrt{M_0}.$$

Пусть W — количество одновременно обрабатываемых двумерных матриц, или, что то же самое, количество одновременно выполняющих БПФ групп процессов:

$$W = \left[\frac{M_0}{N^2}\right],$$

где $[d]$ — округление до ближайшего целого, не превосходящего d.

В случае, когда на одном вычислительном узле запущено несколько групп вычисляющих процессов ($W > 1$), время в настоящем исследовании замеряется по самой медленной группе[1].

[1]Вообще говоря, иногда в случае $W > 1$ и когда N_c не делится нацело на W, возможно несколько повысить эффективность вычислений и снизить время счета, если лучше балансировать вычислительную нагрузку.

Когда N_c не делится нацело на W, на $W - (N_c - \left[\frac{N_c}{W}\right] W)$ матрицах БПФ осуществляется с помощью $\left[\frac{N_c}{W}\right]$ процессов, а на $N_c - \left[\frac{N_c}{W}\right] W$ матрицах БПФ осуществляется с помощью $\left[\frac{N_c}{W}\right] + 1$ процессов.

При этом общий MPI-коммуникатор MPI_COMM_WORLD разделяется на W коммуникаторов $COMM_i$, $i = 0, \ldots, W-1$ так, что процесс с номером r попадает в коммуникатор с номером

$$i_r = \begin{cases} \frac{r}{\left[\frac{N_c}{W}\right]+1}, & r < C_1, \\ \frac{r-C_1}{\left[\frac{N_c}{W}\right]} + N_c - \left[\frac{N_c}{W}\right] W, & r \geq C_1 \end{cases}$$

и имеет внутри этого коммуникатора номер

$$\tilde{r} = \begin{cases} r - i_r(\left[\frac{N_c}{W}\right] + 1), r < C_1, \\ r - C_1 - \left[\frac{N_c}{W}\right](i_r - (N_c - \left[\frac{N_c}{W}\right] W)), r \geq C_1. \end{cases}$$

Среднее время одного БПФ составит

$$T_{mean} = \frac{T(N, [\frac{N_c}{W}])}{W N_n}.$$

В рамках рассматриваемого случая можно отдельно выделить такие размеры матрицы N, при которых на каждом ВУ помещается уже не более одной матрицы размером $N \times N$ и все N_c вычислительных ядер ВУ осуществляют двумерное БПФ этой матрицы:

$$\sqrt{\frac{M_0}{2}} < N < \sqrt{M_0}.$$

(3) Случай, когда матрица лежит в ОЗУ нескольких (точнее, $\widetilde{N}_n = \{\frac{N^2}{M_0}\}$) узлов:

$$N \geq \sqrt{M_0}.$$

Тогда БПФ каждой матрицы осуществляется всеми ВЯ всех $\widetilde{N}_n$ узлов, на которых размещена матрица. Среднее время одного БПФ составит

$$T_{mean} = \frac{T(N, N_c \widetilde{N}_n)}{\left[\frac{N_n}{\widetilde{N}_n}\right]}.$$

1.1. Время выполнения двумерного БПФ на кластерах МВС-100К, МВС-10П и HybriLIT

Описанная выше методика применялась для определения времени выполнения двумерного БПФ на кластерах МВС-100К, МВС-10П и HybriLIT. Два первых суперкомпьютера установлены в МСЦ РАН, последний — в ЛИТ ОИЯИ, г. Дубна. Каждому из суперкомпьютеров в зависимости от его архитектуры соответствует свой набор значений параметров: M_0 — общий объем доступной оперативной памяти на одном ВУ; N_c — количество вычислительных ядер на одном ВУ; $N_{-1} = \sqrt{\frac{M_0}{N_c}}$ — в точности N_c матриц размера $N_{-1} \times N_{-1}$ помещается на одном ВУ, и каждая такая матрица обрабатывается в точности одним вычислительным ядром (если $N < N_{-1}$ — на ВУ лежит более N_c матриц размера $N \times N$, и каждый вычислительный процесс обрабатывает последовательно несколько таких матриц, а если $N > N_{-1}$ — уже несколько групп процессов, где по крайней мере одна из них состоит из более, чем одного процесса, обрабатывают один раз каждая свою матрицу); $N_0 = \sqrt{\frac{M_0}{2}}$ — предельный размер, при котором в оперативной памяти узла помещается уже только одна матрица, и все вычислительные процессы совместно осуществляют ее БПФ; $N_i = \sqrt{iM_0}$, $i = 1, 2, \ldots, N_n$ — предельный размер, при котором матрица размера $N \times N$ еще помещается в оперативной памяти i узлов кластера.

Суперкомпьютер МВС-100К состоит из 1275 вычислительных узлов с несколько различающейся конфигурацией. Большая часть ВУ (990 узлов) состоит из 2 процессоров CPU Intel Xeon E5450 с тактовой частотой 3 ГГц и 8 ГБайт ОЗУ. 19 узлов этого кластера имеют в своем составе восемь GPU Nvidia Tesla M2090 и 192 ГБайт ОЗУ. Пиковая производительность суперкомпьютера составляет 227,94 TFlops, на всех ВУ установлены жёсткий диск объёмом не менее 36 ГБайт, используется коммуникационная сеть Infiniband DDR.

Суперкомпьютер МВС-10П имеет пиковую производительность 523.8 ТФЛОПС и состоит из 207 вычислительных узлов. Каждый вычислительный узел имеет в своем составе 2 процессора Xeon E5-2690, 64 ГБ оперативной памяти, два сопроцессора Intel Xeon Phi 7110X. Коммуникационную и транспортную сеть кластера построена на базе FDR Infiniband.

Таблица 1. Параметры используемых кластеров

Название	M_0	N_c	N_{-1}	N_0	N_1	N_2
МВС-100К	$5 \cdot 10^8$	8	7900	15800	22300	31600
МВС-10П	$4 \cdot 10^9$	16	15800	44700	63200	89400
HybriLIT	$8 \cdot 10^9$	24	18000	63200	89400	126400

Кластер HybriLIT состоит из 7 вычислительных узлов, на каждом из которых установлены два процессора Intel Xeon E5-2695 v2 2.4 ГГц, 128 Гб ОЗУ. Также узлы этого кластера имеют в своем составе видеовычислители Tesla K20X, Tesla K40 и сопроцессоры XeonPhi 5110P, XeonPhi 7120P. Всего на 168 процессорных вычислительных ядрах достигается 3.2 ТФлопс пиковой производительности, на 37248 GPU-ядрах производительность составляет 18.47 ТФлопс на двойной точности, а на 182 ядрах сопроцессоров — 3 ТФлопс на двойной точности.

Значения интересующих нас параметров M_0, N_c, N_{-1}, N_0, N_1, N_2 для этих, описанных выше кластеров приведены в таблице 1.

В соответствии с описанным выше подходом на языке программирования C++ была написана программа (ее исходный код доступен по адресу https://bitbucket.org/Jclash/fft), осуществляющая эффективное БПФ двумерных матриц размера $N \times N$ на кластерной системе и осуществляющая замеры времени выполнения БПФ. Для реализации БПФ использовалась библиотека FFTW [11] (сайт проекта www.fftw.org). Каждому N соответствует некоторое количество процессов $p(N)$, при котором обработка большого количества матриц размера $N \times N$ будет наиболее эффективна. В случае, когда БПФ осуществляется группами с разным количеством вычислительных процессов, в качестве $p(N)$ можно взять среднее количество ВЯ, приходящихся на одну матрицу. Таким образом, можно рассмотреть зависимость времени обработки одной матрицы $T(N, p(N))$ от размера N. Эта зависимость была исследована для трех суперкомпьютеров: МВС-100К и МВС-10П МСЦ РАН, HybriLIT ЛИТ ОИЯИ. Результаты этого исследования приведены на рис. 1.

При вычислениях, показанных на рис. 1 на суперкомпьютере МВС-100К использовалось до 16 ВУ, на суперкомпьютере МВС-10П — до 2 ВУ, а все вычисления на кластере HybriLIT велись в рамках одного узла. Этим объясняется немонотонность роста времени счета

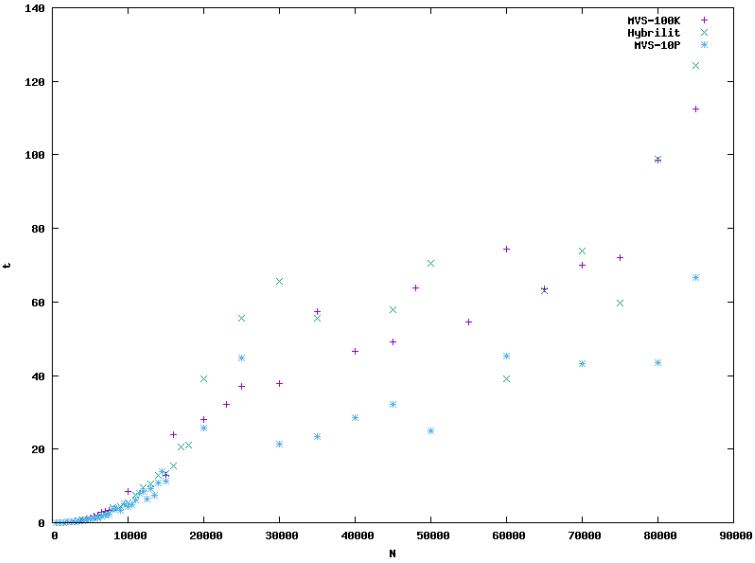

Рис. 1. Зависимость времени выполнения двумерного БПФ $T(N, p(N))$ распределенно хранящейся матрицы размера $N \times N$ от ее линейного размера N на суперкомпьютерах МВС-100К, МВС-10П и HybriLIT

для кластеров HybriLIT и МВС-10Р при размере матрицы, лежащим в диапазоне от 20000 до 80000, так как при изменении размера матрицы изменялось и количество обрабатывающих матрицу процессов, а в случае, когда количество процессов в группах различалось, замер времени производился по самой медленной группе. При небольших размер матрицы (когда БПФ каждой матрицы осуществляется одним ВЯ) время выполнения одного БПФ на всех трех кластерах оказывается достаточно близким. Когда матрица лежит в ОЗУ нескольких ВУ и обрабатывается ими (график рис. 1 для кластера МВС-100К и размера матрицы большего 20000), рост времени осуществления БПФ становится монотонным. В целом же видно, что время на обработку матрицы растет, оно сравнимо для кластеров МВС-100К и HybriLIT и несколько меньше для кластера МВС-10П. Например, при максимальном (из рассмотренных) размере матрицы 85000×85000 быстрее всего (67 секунд) двумерное БПФ было выполнено на суперкомпьютере МВС-10П с использованием 2 ВУ, на МВС-100К этот расчет был

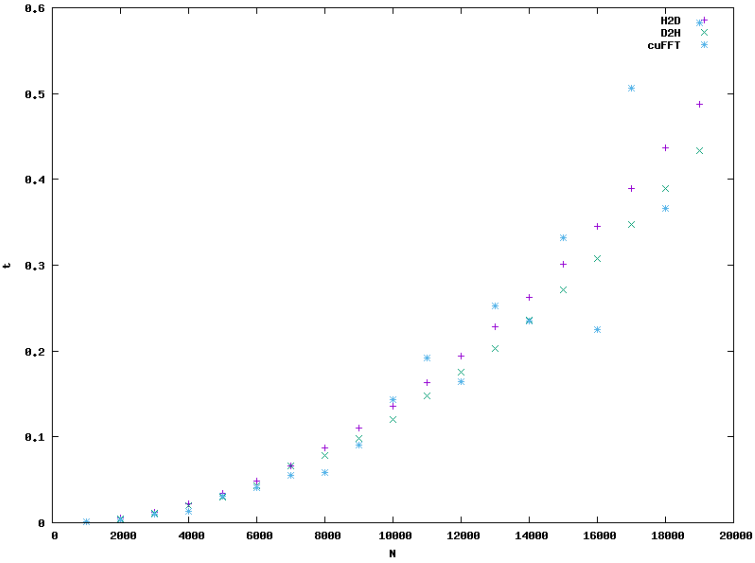

Рис. 2. Время выполнения операции копирования из ОЗУ ВУ на видеокарту (H2D), время копирования из памяти видеокарты в ОЗУ ВУ (D2H) и время выполнения двумерного БПФ для матрицы размера $N \times N$ на видеовычислителе TESLA K40 (cuFFT), при изменении N от 1000 до 19000

сделан на 10 ВУ за 112 секунд и на одном ВУ кластера HybriLIT — за 124 секунды.

2. Двумерное БПФ на герерогенном кластере

Было проведено исследование возможности ускорения расчета за счет использования графических ускорителей NVIDIA TESLA. Также была исследована зависимость скорости выполнения БПФ и пересылки данных при использовании GPU (см. рис. 2). Размер матриц изменялся от 1000×1000 до 19000×19000 — максимально возможного размера для использовавшегося видеоадаптера. Для ускорения пересылок CPU-GPU, в ОЗУ ВУ для хранения матриц использовалась фиксированная память (pinned memory). Для осуществления двумерного БПФ использовалась библиотека NVIDIA cuFFT. Сравнение времени счета с результатами рис. 1 показало, что использование

видеоускорителя дает примерно 48-и кратное ускорение счета по сравнению с использованием CPU (точнее двух CPU, так как на ВУ всех рассмотренных кластеров установлено два процессора).

Из графика рис. 2 видно, что время вычисления двумерного БПФ матрицы практически совпадает со временем ее записи в память GPU или чтения результатов обратно из памяти GPU в ОЗУ ВУ. Таким образом, так как три процесса — перемещение исходных данных в память ускорителя, осуществление вычислений на ускорителе и перемещение результатов из памяти ускорителя в память узла — могут на современных графических ускорителях выполняться одновременно, можно считать, что время выполнения БПФ для N_{GPU} матриц размера $N \times N$ составит

$$T_{GPU}(N) \cdot (N_{GPU} + 2),$$

где $T_{GPU}(N)$ — время выполнения двумерного БПФ матрицы размером $N \times N$ (или, что по времени практически одно и то же, время копирования матрицы в память устройства или обратно, в ОЗУ ВУ) графическим ускорителем.

Использование GPU оправдано в случае, если за время выполнения всего расчета на GPU можно успеть вычислить БПФ хотя бы одной матрицы:

$$T_{mean} \cdot (L - 1) > 3T_{GPU}(N).$$

Если последнее условие выполнено, рассмотрим подходы к балансировке нагрузки между CPU и GPU в двух случаях:

(*1*) Соотношение времен выполнения БПФ на CPU и GPU таково, что за время, пока делается БПФ для N_c матриц на CPU, GPU не успеет сделать БПФ для всех оставшихся матриц:

$$\left(\frac{M_0}{N^2} - N_c + 2\right) T_{GPU}(N) \geq T(N, 1).$$

В этом случае каждое из N_c ВЯ CPU осуществляет БПФ одной из матриц, всего $N_r \geq 1$ раз. Остальные $\frac{M_0}{N^2} - N_c N_r$ БПФ осуществляются на GPU. Оптимальное N_r можно вычислить следующим образом:

$$N_r = \frac{1}{N_c}\left(\frac{M_0}{N^2} + 2 - \frac{T(N, 1)}{T_{GPU}(N)}\right).$$

(2) Соотношение времен выполнения БПФ на CPU и GPU таково, что за время, пока делалось бы одно БПФ для N_c матриц на CPU, на устройстве GPU было бы сделано БПФ для количества матриц, превышающее количество всех оставшихся (не обработанных CPU) матриц:

$$\left(\frac{M_0}{N^2} - N_c + 2\right) T_{GPU}(N) < T(N,1).$$

В этом случае БПФ матриц размера $N \times N$ на CPU выполняются одновременно p вычислительными ядрами. Время работы в таком случае составит

$$T^{hybr}(p) = \max\left\{\left(\frac{M_0}{N^2} - \frac{N_c}{p} + 2\right) T_{GPU}, T(N,p)\right\}.$$

Для балансировки вычислительной нагрузки между CPU и GPU следует сначала найти такое первое p, для которого выполнено

$$\left(\frac{M_0}{N^2} - \frac{N_c}{p}\right) T_{GPU} \geq T(N,p).$$

Обозначим такое p как p^*. Если для p^* выполняется точное равенство, в качестве искомого оптимального $\tilde{p}$ следует взять $\tilde{p} = p^*$. Если точного равенства не достигается, в качестве $\tilde{p}$ возьмем

$$\tilde{p} = \operatorname{argmin}\{T^{hybr}(p^*), T^{hybr}(p^* - 1)\}.$$

Тогда $\frac{N_c}{\tilde{p}}$ БПФ будет выполнено на CPU и $N_{GPU} = \frac{M_0}{N^2} - \frac{N_c}{\tilde{p}}$ на GPU.

3. Заключение

В настоящей работе получены времена выполнения двумерного распределенного БПФ для однородных вычислительных кластеров МВС-100К МСЦ РАН, МВС-10П МСЦ РАН, HybriLIT ЛИТ ОИЯИ. Эти результаты могут быть использованы для уточнения оценок времени работы алгоритмов, использующих двумерное БПФ, например, алгоритмов секционирования свертки.

Разработана программа, осуществляющая распределение вычислительной нагрузки между ВЯ суперкомпьютера и оценивающая время работы при выполнении двумерного БПФ на кластере однородной архитектуры. Исходный код программы доступен по адресу https://bitbucket.org/Jclash/fft.

На примере видеокарты TESLA K40 показано, что время, необходимое для перемещения данных из ОЗУ ВУ в память видеокарты либо обратно, близко времени, требуемому для осуществления двумерного БПФ на графическом вычислителе. При этом ускорение счета составило примерно 48 раз по сравнению с использованием двух процессоров CPU. Это дает основание заключить, что аппаратная структура современных видеовычислителей хорошо подходит для решения задачи осуществления большого количества однотипных двумерных БПФ на гетерогенном кластере. При этом следует использовать предложенный алгоритм балансировки вычислительной нагрузки между центральным процессором и видеокартой.

Благодарности. *Автор выражает глубокую признательность руководству и сотрудникам ЛИТ ОИЯИ, предоставившим возможность и техническую поддержку расчетов на кластере HybriLIT.*

Список литературы

[1] I. Muhammad, O. Khan. “Performance analysis of Fast Fourier Transform on Field Programmable Gate Arrays and graphic cards”, *Proc. of the 2016 International Conference on Computing, Electronic and Electrical Engineering*, ICE Cube (Quetta, Pakistan, 11–12 April, 2016), IEEE, 2016, pp. 158–162. ↑ 47

[2] Г. Нуссбаумер. *Быстрое преобразование Фурье и алгоритмы вычисления сверток*, Радио и связь, М., 1985, 248 с. ↑ 47,48

[3] Д. Даджион, Г. Мерсеро. *Цифровая обработка многомерных сигналов*, Мир, М., 1988, 488 с. ↑ 48

[4] Д.Ю. Князьков. «Эффективные методы расчета электромагнитных полей», *Вычислительные методы и программирование*, **13**:1 (2012), с. 181–188. ↑ 48,49,50

[5] A. Shamaev, D. Knyazkov. “An Effective Method of Electromagnetic Field Calculation”, *Numerical Analysis and Its Applications*. V. II, 5th International Conference NAA 2012 (Lozenetz, Bulgaria, June 15–20, 2012), Lecture Notes in Computer Science, vol. **8236**, Springer, Berlin–Heidelberg, 2013, pp. 487–494. ↑ 48,49

[6] П. А. Михеев. «Применение быстрого преобразования фурье при расчете сегментированной свёртки», *Доклады Академии наук*, **464**:2 (2015), с. 152–155. ↑ 49

[7] М. В. Борисов, В. А. Боровиков, А. А. Гавриков, Д. . Князьков, В. И. Раховский, Д. А. Челюбеев, А. С. Шамаев. «Методы создания и коррекции качества голографических изображений геометрических объектов с элементами субволновых размеров», *Доклады Академии Наук*, **434**:3 (2010), с. 332–336. ↑ 49

[8] О. Ю. Сударева. *Эффективная реализация алгоритмов быстрого преобразования Фурье и свертки на микропроцессоре КОМДИВ128-РИО*, ред. В. Б. Бетелин, НИИСИ РАН, М., 2014, 266 с. ↑ 49

[9] А. А. Бурцев. «Применение векторного сопроцессора для ускорения операции быстрого преобразования Фурье», Национальный Суперкомпьютерный Форум 2015 (Переславль-Залесский, Россия, 24–27 ноября 2015), URL: http://2015.nscf.ru/TesisAll/5_Prikladnoe_PO/05_459_BurtsevAA.pdf ↑ 49

[10] И. А. Каляев, И. И. Левин, Е. А. Семерников, В. И. Шмойлов. *Реконфигурируемые мультиконвейерные вычислительные структуры*, ЮНЦ РАН, Ростов-на-Дону, 2008, 320 с. ↑ 49

[11] M. Frigo, S. G. Johnson. “The design and implementation of FFTW3”, *Proceedings of the IEEE*, **93**:2 (2005), pp. 216–231. ↑ 54

Рекомендовал к публикации *Программный комитет*
Пятого национального суперкомпьютерного форума ***НСКФ-2016***

Пример ссылки на эту публикацию:

Д. Ю. Князьков. «Эффективный расчет двумерного БПФ на однородном или гетерогенном вычислительном кластере», *Программные системы: теория и приложения*, 2017, **8**:1(32), с. 47–62.
URL: http://psta.psiras.ru/read/psta2017_1_47-62.pdf

Об авторе:

Дмитрий Юрьевич Князьков
К.ф.-м.н., н.с. ИПМех РАН. Область научных интересов: прямые и обратные задачи дифракции, математическая теория усреднений, суперкомпьютерные вычисления, алгоритмы быстрой свертки и БПФ, радиометрия океана, сейши, голография, тепломассообмен. Автор около 30 научных публикаций и 4 патентов.
e-mail: knyaz@ipmnet.ru

Dmitri Knyazkov. *Effective computation of two-dimensional FFT on a homogeneous or heterogeneous cluster.*

Abstract. The paper considers performing two-dimensional FFT on a supercomputer. It investigates a dependance of FFT computation time from a matrix size for MVS-100K, MVS-10P and HybriLIT supercomputers. A method of CPU-GPU load balance for a heterogeneous cluster is proposed. For a TESLA K40 card it is shown, that two-dimensional FFT computation time is almost equal to data transferring time. The computation itself is 48 times faster when using GPU comparing to two-processors node. (*In Russian*).

Key words and phrases: HPC, supercomputer computations, fast Fourier transform, FFT, GPU computations.

References

[1] I. Muhammad, O. Khan. "Performance analysis of Fast Fourier Transform on Field Programmable Gate Arrays and graphic cards", *Proc. of the 2016 International Conference on Computing, Electronic and Electrical Engineering*, ICE Cube (Quetta, Pakistan, 11–12 April, 2016), IEEE, 2016, pp. 158–162.

[2] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer Series in Information Sciences, vol. **2**, 2nd edn., Springer, Berlin–Heidelberg, 1982, xii+276 p.

[3] D. E. Dudgeon, R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall Signal Processing Series, 2nd edn., Prentice-Hall, Englewood Cliffs, 1984, xv+400 p.

[4] D. Yu. Knyazkov. "Efficient Numerical Methods for the Analysis of Electromagnetic Fields", *Vychislitel'nyye metody i programmirovaniye*, **13**:1 (2012), pp. 181–188 (in Russian).

[5] A. Shamaev, D. Knyazkov. "An Effective Method of Electromagnetic Field Calculation", *Numerical Analysis and Its Applications*. V. II, 5th International Conference NAA 2012 (Lozenetz, Bulgaria, June 15–20, 2012), Lecture Notes in Computer Science, vol. **8236**, Springer, Berlin–Heidelberg, 2013, pp. 487–494.

[6] P. A. Mikheyev. "Application of the Fast Fourier Transform to Calculating Pruned Convolution", *Doklady Mathematics*, **92**:2 (2015), pp. 630–633.

[7] M. V. Borisov, V. A. Borovikov, A. A. Gavrikov, D. . Knyazkov, V. I. Rakhovskiy, D. A. Chelyubeyev, A. S. Shamayev. "Methods of the Development and Correction of the Quality of Holographic Images of Geometry Objects with Subwave-Size Elements", *Doklady Physics*, **55**:9 (2010), pp. 436–440.

[8] O. Yu. Sudareva. *Effective realization of fast Foureier transform and convolution algorithms on a COMDIV128-RIO microprocessor*, ed. V. B. Betelin, NIISI RAN, M., 2014 (in Russian), 266 p.

[9] A. A. Burtsev. "Using a Vector Co-Processor for Fast Fourier Transform Operation Speed-Up", Natsional'nyy Superkomp'yuternyy Forum 2015 (Pereslavl'-Zalesskiy, Rossiya, 24–27 noyabrya 2015) (in Russian), URL: http://2015.nscf.ru/TesisAll/5_Prikladnoe_PO/05_459_BurtsevAA.pdf

[10] I. A. Kalyayev, I. I. Levin, Ye. A. Semernikov, V. I. Shmoylov. *Reconfigurable multi-conveyer computational structures*, YuNTs RAN, Rostov-na-Donu, 2008 (in Russian), 320 p.

[11] M. Frigo, S. G. Johnson. "The design and implementation of FFTW3", *Proceedings of the IEEE*, **93**:2 (2005), pp. 216–231.

Sample citation of this publication:

Dmitri Knyazkov. "Effective computation of two-dimensional FFT on a homogeneous or heterogeneous cluster", *Program systems: Theory and applications*, 2017, **8**:1(32), pp. 47–62. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2017_1_47-62.pdf