Высопроизводительные алгоритмы обращения матриц на GPU

Н.С. Недожогин, А.С. Сармакеева, С.П. Копысов

Институт механики УрО РАН

Международная научная конференция
Параллельные вычислительные технологии (ПаВТ) 2014
Ростов-на-Дону, 1 — 4 апреля 2014 г.

Содержание

Задачи:

- цифровая обработка изображений;
- решение задач линейного программирования, структурного анализа, теории цепей;
- теории графов, генетики, социологии и т.д;
- решения систем уравнений;
- вычисление дополнения Шура в методах декомпозиции области;
- построение предобуславливателей.

Матрицы:

невырожденные;

разреженные;

плохо обусловленные.

Метод Csanky

В работе Ксанки (Csanky L.)[1] был построен теоретически наилучший параллельный алгоритм обращения матриц.
Вычислительные затраты алгоритма Ксанки для обращения матрицы $(N \times N)$ составляют $\mathcal{O}(\log^2 N)$ операций с использованием $N^4/(\log N)$ процессоров.

Метод Ксанки является отличным теоретическим алгоритмом, но мало пригоден при создании параллельных программ:

- Метод крайне чувствителен к ошибкам округления, возникающих при вычислении следов матриц.
- Требуется N^4 процессоров.

[1] Csanky L. Fast Parallel Matrix Inversion Algorithms, SIAM J. Comput. 1976. Vol 5. P.618-623.

Метод Ньютона

В 1985 г. в работе Pan V., Reif J.[2] были предложены устойчивые параллельные алгоритмы обращения для хорошо обусловленных матриц с оптимальными затратами, основанный на итерационном методе Ньютона.

Затраты по времени составляют $\mathcal{O}(\log^2 N)$ операций и $\log N$ процессоров.

Применять метод Ньютона для обращения матриц общего вида не совсем целесообразно:

- На каждом шаге требуется два матричных умножения;
- Сходимость метод Ньютона достаточно медленная для плохо обусловленных матриц.

[2] Pan V., Reif J. Fast and efficient parallel solution of dense linear systems // Computers and Mathematics with Applications. 1989. Vol. 17, №11. P.1481-1491.

Метод LU факторизации

Для заданной невырожденной матрицы A существует разложение $A = LU$, где U - верхне-треугольная, L - нижне треугольная матрицы.
Если матрицы A, U, L обратимы, то справедливо следующее утверждение: $A^{-1} = U^{-1}L^{-1}$.

- Факторизация проводилась для матриц хранящихся в формате CSR.
- Рассматривалась только последовательная реализация LU-факторизации на CPU, показывающая наилучшие результаты для малых и средних матриц.
- Особенности данного алгоритма не позволяют эффективно использовать все возможности массивно-параллельной архитектуры GPU.
- В работе Li R., Saad Y. [3] также отмечается максимальное GPU-ускорение порядка трех.

[3] Li R., Saad Y. GPU-accelerated preconditioned iterative linear solvers // The Journal of Supercomputing. 2013. Vol. 63. Issue 2. P. 443-466.

Метод Гаусса-Жордана

- Все операции выполняются на центральном процессоре CPU.
- Так как преобладающие вычисления в данном методе содержат матричные операции, то в дальнейшем можно ожидать высокую производительность варианта для гибридной архитектуры при совместном использовании CPU+GPU.
- В работе Ezzatti P.[4] показано, что максимально получаемое трехкратное ускорение метода обращения Гаусса-Жордана достигается лишь при вычислениях с одинарной точностью и разделением операций выполняемых на CPU и GPU.

[4] Ezzatti P., Quintana-Orti E.S., Remon A. Using graphics processors to accelerate the computation of the matrix inverse // J. of Supercomputing. – 2011. V.58. – P.429-437.

Метод сопряженных градиентов

Рассматриваемый в работе алгоритм вычисления обратной матрицы состоит из решений матричной системы вида $AX = E$, где E — единичная матрица. Система эффективно решается на GPU предобусловленным алгоритмом сопряженных градиентов[5] .

- В тестах использовался диагональный предобуславливатель, так как он обеспечивает минимальные вычислительные затраты (количества требуемой памяти, времени построения предобуславливателя и времени его решения).
- Данный алгоритм удобен в методе дополнения Шура. Если заменить матрицу E в правой части на произвольную матрицу B, то результатом будет матрица $X = A^{-1}B$
- Эффективен только для хорошо обусловленных матриц больших размерностей.

[5] Kopysov S. P., Kuzmin I. M., Nedozhogin N. S., Novikov A. K., Sagdeeva Y. A. Hybrid Multi-GPU solver based on Schur complement method // Lecture Notes in Computer Science – 2013. – vol. 7979. – pp. 65-79.

Метод Шермана-Моррисона-Вудбери

За основу построения A^{-1} берут матрицу B той же размерности, что и A, но с известной обратной матрицей.
Для обратимости матрицы A вида: $A = B - UV^T$, где A — невырожденная матрица $N \times N$, а U, V — матрицы $(N \times k)$, необходимо и достаточно, чтобы была обратимой матрица k-го порядка $P = I_k - V^T B^{-1} U$. При этом

$$A^{-1} = B^{-1} + B^{-1} U P^{-1} V^T B^{-1}. \quad (1)$$

В работе He X., Holm M., Neytcheva M.[6] был исследован алгоритм (1) для обращения плотных матриц, в том числе и на GPU, в котором требуется достаточно затратное разделение матриц на блоки. Заметного ускорения на GPU получено не было.

[6] He X., Holm M., Neytcheva M. Efficiently parallel implementation of the inverse Sherman-Morrison algorithm //Lecture Notes in Computer Science. – 2013. – V. 7782. — P. 206-219.

Метод Шермана-Моррисона

В частном случае при $k = 1$ формула Шермана-Моррисона-Вудберри приводит к следующей теореме:

Теорема[7]. Пусть B — невырожденная матрица и вектора $\mathbf{u}$ и $\mathbf{v}$ такие, что $r = 1 + \mathbf{v}^T B^{-1}\mathbf{u} \neq \mathbf{0}$. Тогда матрица $A = B^{-1} + \mathbf{v}^T\mathbf{u}$ является обратимой

$$A^{-1} = B^{-1} - r^{-1}B^{-1}\mathbf{u}\mathbf{v}^T B^{-1}. \tag{2}$$

Полагая, что $B^{-1}\mathbf{e}_k = \mathbf{b}_k^*$, $\mathbf{b}_k^*$ — k-ый столбец матрицы B^{-1}. Тогда столбец $\mathbf{a}_j^*$

$$\mathbf{a}_j^* = \mathbf{b}_j^* - \frac{\mathbf{v}^T\mathbf{b}_j^*}{(1 + \mathbf{v}^T\mathbf{b}_j^*)}\mathbf{b}_k^*, \quad j = 1, 2, \ldots, N. \tag{3}$$

По (3) вычисляем A^{-1}, принимая что $B = E$ и вычисляя на k шаге $\mathbf{u} = \mathbf{e}_k$, $\mathbf{v}^T = \mathbf{v}^k = \mathbf{a}^k - \mathbf{e}_k^T$, исходя из $A = B + \sum_{k=1}^{N}\mathbf{v}^k$. Столбец матрицы на k шаге вычисляется из соотношения

$$\mathbf{a}_j^{(k)} = \mathbf{a}_j^{(k-1)} - \frac{\mathbf{v}^T\mathbf{a}_j^{(k-1)}}{(1 + \mathbf{v}^T\mathbf{a}_k^{(k-1)})}\mathbf{a}_k^{(k-1)}, \quad j = 1, 2, \ldots, N, \tag{4}$$

где $A^{(0)} = E$ и для $k = N$ находится $\mathbf{a}_j^{(N)} = \mathbf{a}_j^*, j = 1, 2, \ldots, N$.

[7] Sherman J., Morrison W.J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix // Ann. Math. Statistics. – 1950. – V. 21. №1. – P. 124-127.

Метод Шермана-Моррисона

Алгоритм 1 Параллельный алгоритм Шермана-Моррисона

Require: $\mathbf{a}, \mathbf{v} \in \mathbb{R}^N$ {вектора хранятся в памяти GPU}

$\mathbf{v}^T = \mathbf{a}^k - \mathbf{e}_k^T$ { вектор $\mathbf{e}_k \in \mathbb{R}^N$, k элемент которого равен 1, а все остальные 0}

for $k = 0 \to N$ **do**

  for $i = 0 \to N$ **do**

    for $j = 0 \to N$ **do**

      $\beta_1 \leftarrow (\mathbf{v}_k^T, \mathbf{a}_j^{(k-1)})$ {вычисляется с помощью функции cublasDdot}

      $\beta_2 \leftarrow 1 + (\mathbf{v}_k^T, \mathbf{a}_k^{(k-1)})$ {cublasDdot}

      $\beta_3 \leftarrow \beta_1 / \beta_2$

      $\mathbf{a}_j^{(k)} \leftarrow \mathbf{a}_j^{(k-1)} - \beta_3$ {cublasDaxpy}

    end for

  end for

end for

Тестовые матрицы

Численные эксперименты были проведены на тестовых матрицах:

- Matrix Market: http://math.nist.gov/MatrixMarket/ (дата обращения: 29.08.2013)
- Tim Davis: University of Florida Sparse Matrix Collection: sparse matrices from a wide range of applications: http://www.cise.ufl.edu/research/sparse/matrices/ (дата обращения: 29.08.2013)
- Матрицы полученные при решении задачи напряженно-деформированного состояния винтовой пружины методом дополнения Шура[8].

Все матрицы симметричные и положительно-определённые. Размеры матриц варьируются от 276 до 18000, число ненулевых элементов — от 1666 до 6897316, числа обусловленности χ — от 10 до 10^{10}, а все результаты получены с двойной точностью.

Для вычислений использовался узел с четырехядерным процессором Intel Xeon CPU E5430 частотой 2667 МГц и графическим ускорителем NVIDIA GeForce GTX 580. Количество ядер CUDA 512, объем памяти 3 ГБ, частота ядра/памяти 772 МГц/4008 МГц.

[8] Kopysov S. P., Kuzmin I. M., Nedozhogin N. S., Novikov A. K., Sagdeeva Y. A. Hybrid Multi-GPU solver based on Schur complement method // Lecture Notes in Computer Science – 2013. – vol. 7979. – pp. 65-79.

Оценка точности обращения

Для проверки точности, полученных результатов обращения матриц возьмем тестовые матрицы разной размерности и с разным числом обусловленности χ. Для вычисления погрешности используем матрицу $R = E - AA^{-1}$. Вычислим максимальную норму и норму Фробениуса :

$$||R||_\infty = \max_i \sum_j |R_{ij}|, \quad ||R||_2 = \sqrt{\sum_{i,j} R_{ij}^2}. \tag{5}$$

Таблица 1: Максимальная норма и норма Фробениуса

Матрица	N dim.	χ cond.	Метод LU $\|\|R\|\|_\infty$	Метод LU $\|\|R\|\|_2$	Метод JG $\|\|R\|\|_\infty$	Метод JG $\|\|R\|\|_2$
494_BUS	494	3.9e+06	1.2e-11	4.7e-11	2.3e-11	8.2e-11
1138_BUS	1138	1e+02	1.1e-10	4.1e-10	1.2e-10	3.0e-10
BCSSTK15	3948	8e+09	1.7e-10	1.4e-10	8.1e-10	2.9e-10

Матрица	N dim.	χ cond.	Метод SM $\|\|R\|\|_\infty$	Метод SM $\|\|R\|\|_2$	Метод CG $\|\|R\|\|_\infty$	Метод CG $\|\|R\|\|_2$
494_BUS	494	3.9e+06	1.1e-11	2.7e-11	1.1e-07	1.4e-07
1138_BUS	1138	1e+02	7.3e-11	3.6e-10	1.7e-07	2.8e-07
BCSSTK15	3948	8e+09	1.6e-10	1.7e-10	2.8e-07	5.8e-07

Сравнение алгоритмов

Таблица 2: Время обращения матриц (сек.)

Матрица	N/Nnz dim./non-zero	χ cond.	GJ CPU	LU CPU	CG GPU	SM CPU	SM GPU
Schur276	276/7488	1.2e+02	0.1	0.1	1.1	0.1	4.1
494_BUS	494/1666	3.9e+06	0.3	0.7	23.9	1.31	12.1
1138_BUS	1138/4054	8.0e+09	9.5	11.9	133.1	23.0	62.5
Schur1236	1236/40806	1.5e+01	12.6	18.6	8.8	34.5	74.5
BCSSTK11	1473/34241	5.3e+08	21.4	33.9	246.6	57.9	104.8
BCSSTK15	3948/117816	8.0e+09	426.0	649.1	329.1	1182.9	751.1
Shur5688	5688/203238	2.9e+01	1262.8	1822.5	84.8	3171.6	1572.2
ND3K	9000/3279690	1.6e+07	5009.6	7175.4	34801.3	12960.3	3918.7
msc10848	10848/1229776	9.9e+09	8718.8	12446.8	15259.6	21663.5	5741.0
Dubcova1	16129/253009	9.9e+02	29868.9	40859.7	617.3	75472.2	13006.5
bodyy4	17546/121550	8.1e+02	40400.6	—	376.4	—	15566.6
ND6K	18000/6897316	1.5e+07	41782.0	35674.5	—	—	16386.0

Выводы:

- для матриц небольшого размера — последовательные алгоритмы GJ реализованные на CPU,
- с увеличением размеров матриц, эффективнее использовать алгоритмы на GPU,
- для хорошо обусловленных матриц — MultiGPU-реализация метода сопряженных градиентов,
- в случае плохообусловленных матриц — метод Шермана-Моррисона, показывающий минимальное трехкратное GPU-ускорение.

Развитие:

- Разработка параллельной реализации на MultiGPU.
- Параллельный вариант метода Шермана-Моррисона учитывающий симметричность матриц.
- Построение явных предобуславливателей на основе модификаций алгоритма Шермана-Моррисона.

Благодарности:
Работа выполнена при финансовой поддержке РФФИ (грант 14-01-31066-мол_а,14-01-00055-а, 13-01-00101-а) и программы Президиума РАН № 18 при поддержке УрО РАН (проект 12-П-1-1005).