Анализ эффективности использования средств распределенных вычислений для систем с общей памятью при моделировании течений вязкой жидкости методом граничных элементов*

М.А. Пономарева

Томский государственный университет, Томск

Представлены результаты применения средств распределенных вычислений для систем с общей памятью для параллельной реализации алгоритма решения задач динамики вязкой жидкости при малых числах Рейнольдса методом граничных элементов.

Введение

С появлением многопроцессорной вычислительной техники актуальной становится разработка эффективных параллельных реализаций существующих вычислительных алгоритмов решения научно-технических задач, в том числе задач динамики жидкости. В то же время известно, что использование многих средств организации распределенных вычислений сопряжено со значительным преобразованием программного кода (до 100% в случае использования GPU) и ограниченностью использования полученной параллельной версии программы при отсутствии вычислительных систем с параллельной архитектурой [1–3]. В некоторых случаях добиться значительного повышения эффективности работы алгоритма возможно, частично избежав упомянутых трудностей. Для этого достаточно задействовать вычислительный потенциал наиболее широко распространенной, многопроцессорной SMP-машины, воспользовавшись средствами распределенных вычислений для систем с общей памятью (например, OpenMP), в том числе существующими оптимизированными низкоуровневыми математическими библиотеками, имеющими параллельную реализацию (например, Intel® Math Kernel Library). В настоящей работе такой подход

*Исследование выполнено при поддержке гранта Президента РФ (МК-3687.2014.1) и РФФИ в рамках научного проекта №14-08-31579 мол_а и проекта № 12-08-00313а.

использован для оптимизации программы расчета, реализующей непрямой метод граничных элементов (НМГЭ) для моделирования плоских течений вязкой жидкости [4]. НМГЭ относится к классу численных методов расчета динамики жидкости с явным выделением границы, что делает его применение весьма эффективным в случаях, когда требуется максимально точное выполнение граничных условий, например при наличии свободной поверхности и учете межфазного взаимодействия (поверхностного натяжения, смачиваемости и т.п.). К преимуществам данного подхода можно также отнести снижение размерности задачи на единицу и непрерывное моделирование полей в области решения. Результаты расчетов с использованием многопоточных вычислений методом граничных элементов (МГЭ) (включая расчеты на кластерных системах и графических процессорах) приведены в работах К.Е. Афанасьева, Н.А. Гумерова, A.J. Davies, Y.Y. Wu, M.S. Ingber, Y.J. Liu, S.R. Subia, D. Li и др. для случаев идеальной жидкости, многофазных течений и, конечно же, в теории упругости, где МГЭ получил более широкое распространение. То, что алгоритм, реализующий МГЭ, имеет степень параллелизма, близкую к идеальной, является несомненным преимуществом гранично-интегрального подхода и также положительно сказывается в случае реализации параллельных вычислений для моделирования течений вязкой жидкости.

Постановка задачи

В качестве тестовой выбрана задача о стабилизированном течении вязкой жидкости в плоском канале (рис. 1), имеющая аналитическое решение. Задача решается в безразмерных переменных. В качестве характерных масштабов длины, скорости и давления выбраны полуширина канала R, средняя скорость течения U, $R/\mu U$ (μ – коэффициент динамической вязкости). Система уравнений (уравнения движения совместно с уравнением неразрывности) и граничные условия имеют вид:

$$\begin{cases} \dfrac{\partial \Pi_{ij}}{\partial x_j} = 0, \quad i, j = 1, 2 \\ \dfrac{\partial u_i}{\partial x_i} = 0. \end{cases} \tag{1}$$

$$u_1 = 3/2\left(1-x_2^2\right), \quad u_2 = 0, \quad x \in \Gamma_1$$
$$u_i = 0, \quad x \in \Gamma_2 \qquad (2)$$
$$t_1 = 0, \quad u_2 = 0, \quad x \in \Gamma_3$$

где $\Pi_{ij} = -p\delta_{ij} + 2\dot{e}_{ij}$ – тензор напряжений, $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ – тензор скоростей деформаций, p – давление, u_i – компоненты вектора скорости, $t_i = \Pi_{ij} n_j$ – усилия, n_i – компоненты внешней нормали.

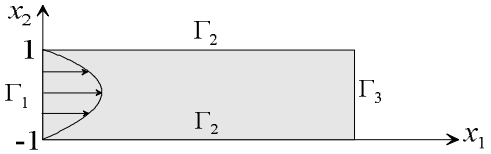


Рис. 1. Область решения

Метод решения

Система (1) заменяется эквивалентной системой граничных интегральных уравнений [4]. Дискретный аналог последней, в случае, если граница аппроксимируется N постоянными элементами, представляет собой систему линейных алгебраических уравнений (СЛАУ):

$$\underbrace{\begin{bmatrix} \Delta U_{ij}^{pq} \\ \Delta T_{ij}^{pq} \end{bmatrix}}_{2N\times 2N} \times \underbrace{\left[\varphi_j^q\right]}_{2N} = \underbrace{\begin{bmatrix} u_i^p \\ t_i^p \end{bmatrix}}_{2N}, \qquad (3)$$

коэффициенты которой

$$\Delta U_{ij}^{pq} = \int_{\Delta\Gamma} U_{ij}(x_0^p, \xi^q) d\Gamma(\xi^q), \quad \Delta T_{ij}^{pq} = \int_{\Delta\Gamma} T_{ij}(x_0^p, \xi^q) d\Gamma(\xi^q) \qquad (4)$$

являются интегралами от фундаментальных сингулярных решений линеаризованной системы уравнений Навье – Стокса для скоростей U_{ij} и усилий T_{ij} соответственно, по элементам границы области решения $\Delta\Gamma$; p и q – номера элементов, $x_0^p \in \Gamma$ – точка приложения нагрузки, $\xi^q \in \Gamma$ – точка наблюдения, u_i^p, t_i^p – значения скоростей и

усилий, известные на границе. Матрица системы (3) является несимметричной, полностью заполненной с диагональным преобладанием (рис. 2). В результате решения СЛАУ (3) находится вектор-столбец φ_j^q, свертка которого с соответствующими фундаментальными решениями дает значения скоростей и напряжений в любой точке области. В рассматриваемой задаче решением является профиль скорости u_1 на Γ_3.

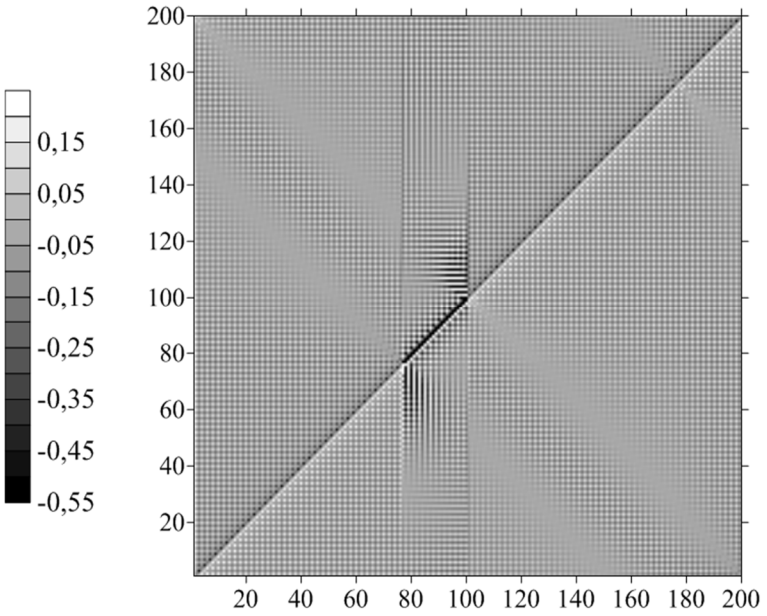


Рис. 2. Портрет матрицы при N=100

Алгоритм, в соответствии с которым работает программа расчета, выглядит следующим образом:

1. Инициализация данных: чтение параметров расчета, задание области решения и граничных условий;

2. Формирование матрицы СЛАУ: вычисление коэффициентов.

3. Решение СЛАУ в два этапа: факторизация матрицы и прямое решение СЛАУ с факторизованной матрицей;

4. Расчет профиля скорости на выходе из канала, вычисление отклонения от аналитического решения.

Результаты работы последовательного алгоритма

В последовательном алгоритме для решения СЛАУ использовались стандартные последовательные процедуры, реализующие метод Гаусса с выбором главного элемента [5]. Расчеты проведены на суперкомпьютере СКИФ «Cyberia». Для каждого расчета задействован узел, включающий 2 6-ядерных процессора Intel® Xeon™ 3670 2.99 ГГц. Время выполнения алгоритма в секундах увеличивается с ростом числа граничных элементов как $t_p \approx 5 \cdot 10^{-9} N^{2.87}$. Отношения времен выполнения отдельных блоков алгоритма ко времени вычислений показаны на рис. 3.

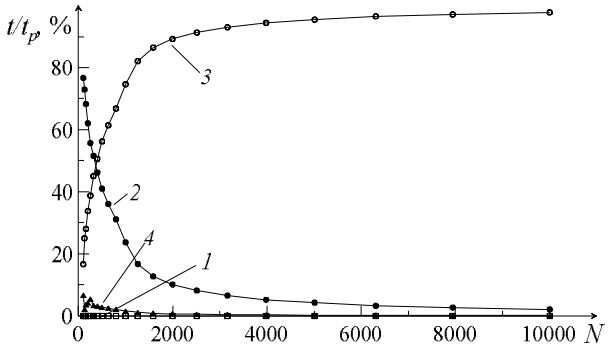


Рис. 3. Зависимость относительных времен работы отдельных блоков алгоритма (*1* – инициализация данных, *2* – формирование матрицы, *3* – решение СЛАУ, *4* – расчет профиля на выходе)

Из рис. 3 следует, что основное время тратится на формирование матрицы и решение СЛАУ.

Результаты работы параллельного алгоритма

На этапе выполнения двух наиболее ресурсоемких частей алгоритма реализованы параллельные вычисления с использованием средств распределенных вычислений для систем с общей памятью, таким образом, что для последовательного и параллельного расчета использовался один и тот же код. Для решения СЛАУ использован модуль линейной алгебры LAPACK из библиотеки Intel® Math Kernel Library [6], входящей в Intel® Visual Fortran Composer XE 2011, его последовательная и параллельная

реализации. Решение СЛАУ осуществляется посредством последовательного вызова двух библиотечных процедур. Для процедуры формирования матрицы СЛАУ использована технология OpenMP.

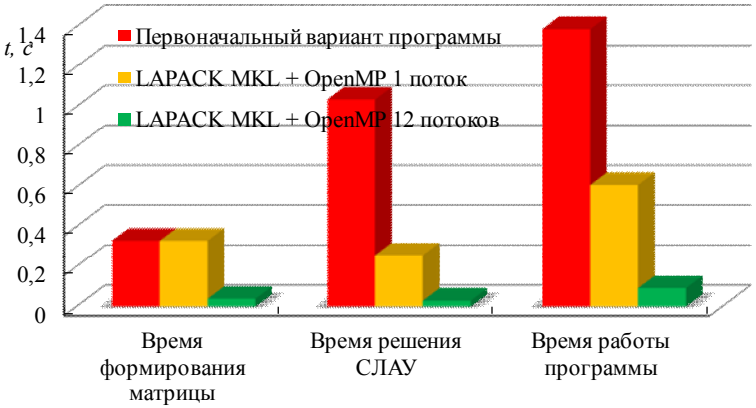


Рис. 4. Время работы программы и ее отдельных блоков при $N = 1000$

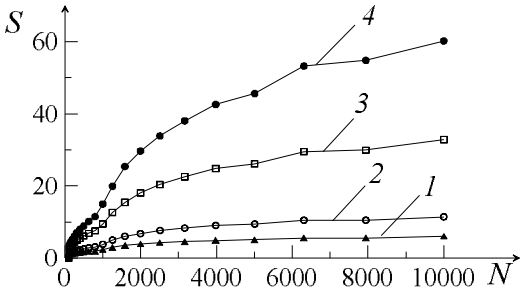


Рис. 5. Ускорение вычислений по сравнению с первоначальным вариантом программы (*1* – 1 поток, *2* – 2 потока, *3* – 6 потоков, *4* – 12 потоков)

Итерации цикла, в котором рассчитываются значения коэффициентов матрицы СЛАУ, распределяются между доступными потоками, для этого используется всего несколько директив OpenMP. Вычислительный эксперимент показал, что оптимальным является статическое распределение итераций

соответствующего цикла (до 50% эффективнее динамического распределения). Количество потоков выбиралось равным числу задействованных ядер процессоров. В случае превышения числа потоков над числом доступных ядер время расчетов не увеличивалось. Результаты представлены на рис. 4–6.

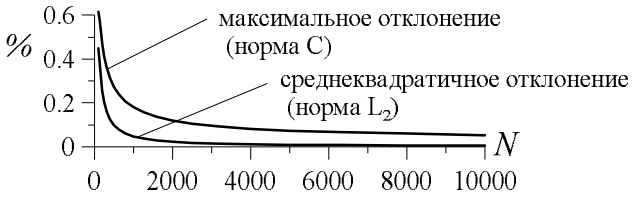


Рис. 6. Сравнение с аналитическим решением

Таким образом, применение параллельных вычислений для рассматриваемого алгоритма позволило добиться значительного ускорения проведения расчетов с использованием многоядерной архитектуры системы с общей памятью, не прибегая к значительному изменению программного кода. Так, использование модуля LAPACK [7] даже в однопоточном режиме ускоряет время решения СЛАУ приблизительно в 4–5 раз (рис. 5, кривая *1*). Суммарно полученное ускорение в случае совместного использования модуля LAPACK для решения СЛАУ и OpenMP для формирования матрицы на указанной системе достигает нескольких десятков раз.

Литература

1. *Гергель В.П.* Современные языки и технологии параллельного программирования: Учебник / Предисл.: В.А. Садовничий. М.: Изд-во МГУ, 2012. 408 с.
2. *Антонов А.С.* Технологии параллельного программирования MPI и OpenMP: Учеб. пособие / Предисл.: В.А. Садовничий. М.: Изд-во МГУ, 2012. 344 с.
3. *Беликов Д.А., Говязов И.В., Данилкин Е.А. и др.* Высокопроизводительные вычисления на кластерах: Учеб. пособие/ под ред. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2008. – 198 с.
4. *Пономарева М.А., Якутенок В.А.* Метод граничных элементов для решения уравнений Стокса. Моделирование плоских

течений вязкой жидкости со свободной поверхностью. LAMBERT Academic Publishing, 2013. 160 с.
5. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. М.: Мир, 1980.
6. http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_lapack_examples