Эффективность параллельной реализации метода конечных элементов для задачи распространения поверхностных волн*

Е.В. Дементьева[1], Е. Д. Карепова[1,2]

[1] Институт вычислительного моделирования СО РАН,
[2] Институт математики и фундаментальной информатики СФУ,
Красноярск

Проведено исследование эффективности нескольких параллельных реализаций метода конечных элементов для алгоритма численного решения начально-краевой задачи для уравнений мелкой воды, выполненных с помощью технологий MPI, OpenMP и MPI+OpenMP.

Введение

Численное моделирование поверхностных волн в больших акваториях проводилось с учетом сферичности Земли и ускорения Кориолиса на основе уравнений мелкой воды [1]. В [2] для этой задачи построен метод конечных элементов (МКЭ), и дифференциальная задача сведена к векторно-матричной форме. Полученная система линейных алгебраических уравнений решается итерационным методом Якоби, который обладает хорошим параллелизмом, а диагональное преобладание для его сходимости обеспечивается выбором шага по времени [3].

Использование МКЭ при пространственной дискретизации задачи дает ряд преимуществ, основное из которых – возможность использовать неструктурированные, неравномерные сетки для вычислительных областей сложной формы. В то же время МКЭ, безусловно, имеет большую вычислительную сложность [4] по сравнению с методом конечных разностей, что делает актуальным использование высокопроизводительных вычислительных систем для его реализации [5–11].

В настоящей работе для начально-краевой задачи для уравнений мелкой воды проводится исследование эффективности нескольких

* Работа выполнялась в рамках гранта РФФИ № 11-01-00224-а, интеграционного проекта СО РАН № 130, проекта Президиума РАН № 18.2.

параллельных реализаций метода конечных элементов, выполненных с помощью технологий MPI, OpenMP и MPI+OpenMP.

Параллельная реализация на SMP-узловом кластере

Для численного моделирования распространения длинных волн в большой акватории использовался метод конечных элементов с линейными треугольными конечными элементами [2].

Для исследования ускорения нескольких параллельных реализаций алгоритма рассматривалась модельная задача для «квадрата» на сфере с твердыми границами [12, 13]. В расчетной области использовалась равномерная квадратная сетка с согласованной триангуляцией. В обсуждаемых вычислительных экспериментах использовалась сетка 1600×1600 шагов по пространству, и делалось 200 шагов по времени. В нашем случае таких параметров сетки было достаточно для хорошей масштабируемости задачи при использовании до ста вычислительных устройств.

Самой вычислительно трудоемкой операцией в МКЭ является сборка невязки на основе локальных матриц жесткости элементов. Существует, по крайней мере, два способа обхода триангуляции при сборке невязки:

1) по элементам (традиционный способ, реализующий наиболее выгодное распределение памяти при хранении информации о триангуляции);

2) по точкам сетки (требует размещения в памяти дополнительных, в общем случае, нерегулярных структур, отвечающих за хранение информации о триангуляции).

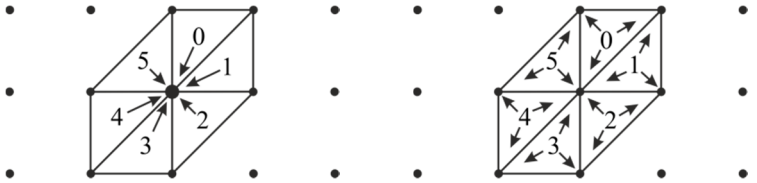


Рис. 1. Схема сборки невязки по точкам сетки (слева) и по элементам (справа)

Целью работы было исследование эффективности нескольких возможных параллельных реализаций МКЭ для решения задачи на высокопроизводительных кластерах с узлами на общей памяти (SMP-узловых кластерах). Расчеты проводились на высокопроизводительных комплексах СФУ и ССКЦ СО РАН.

Отметим, что для компиляции последовательных версий использовался набор оптимизирующих ключей компилятора, который давал наименьшее время выполнения программы. Именно это время использовалось для вычисления ускорения параллельных версий. Перечислим основные выводы, которые можно сделать на основе теоретического и численного анализа (рис. 2, 3).

1. Последовательные реализации. Численные эксперименты показали, что время выполнения последовательной программы при сборке невязки по элементам в 1,5 раза меньше, чем при сборке по точкам сетки, что можно объяснить более выгодным распределением памяти в первом случае.

2. Реализации для вычислительных систем (ВС) с общей памятью на основе технологии OpenMP. Поскольку вклад в невязку в точке дают несколько треугольников (рис. 1), то при параллельной реализации поэлементной сборки на общей памяти будут существовать точки сетки, которые обрабатываются разными нитями, причем, возможно, одновременно. Следовательно, в этом случае необходимы дополнительные затраты на синхронизацию нитей, которая занимает до 40% времени выполнения основного цикла, эффективность распараллеливания около 25% (рис. 2). При сборке невязки по точкам сетки дополнительной синхронизации нитей не требуется, что дает явные преимущества этого подхода. Эффективность распараллеливания составляет около 90% при использовании до 30 нитей и около 80% при использовании более 30 нитей (одна нить на ядро).

3. Реализация для ВС с распределенной памятью на основе технологии MPI. Поскольку используется подход, связанный с декомпозицией вычислительной области без теневых граней (перекрытий), то при обоих способах сборки невязки на каждой итерации неизбежно возникают следующие накладные расходы [12, 13]:

1) время, затрачиваемое на обмены типа точка-точка между соседними процессами для сборки полной невязки на разрезах вычислительной области;

2) время на коллективную операцию глобальной редукции для расчета критерия останова итерационного процесса.

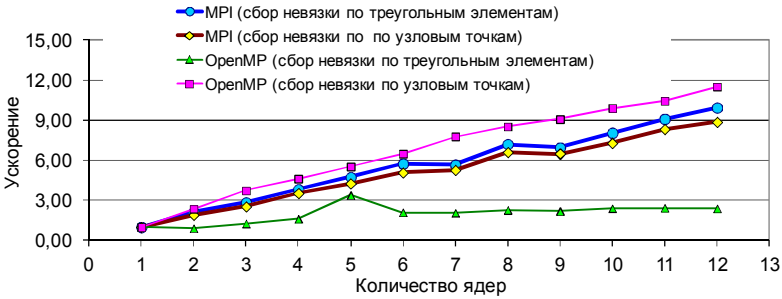


Рис. 2. График зависимости ускорения вычислений от количества используемых ядер для MPI- и OpenMP-версий программ

Численные эксперименты показали преимущество сборки невязки по элементам с эффективностью параллельной реализации около 80%. Рис. 3(в) демонстрирует, что эта версия программы выполняется быстрее, чем MPI-версия при сборке невязки по узловым точкам, что объясняется более выгодным распределением памяти в первом случае. Особо следует отметить, что эта версия MPI-программы по времени выполнения выигрывает и по сравнению с самой быстрой OpenMP-версией.

4. Совмещение технологий MPI и OpenMP для SMP-узлового кластера (рис.3). В численных экспериментах запускались два MPI-процесса на узел, по пять OpenMP-нитей на каждый MPI-процесс. Из рис. 3 видно, что при сборке невязки по узловым точкам ускорение вычислений MPI+OpenMP-версии программы практически совпадает с линейным, а эффективность составляет около 100%. В случае сборки невязки по элементам эффективность параллельной реализации составляет всего 40%, что ожидаемо ввиду затрат на синхронизацию нитей в OpenMP.

Результаты исследований показали, что наиболее эффективной из рассмотренных параллельных реализаций алгоритма оказалась MPI+OpenMP-версия программы при сборке невязки по точкам сетки. Следует отметить, что она является в то же время наиболее сложной в реализации (требует создания, хранения и обработки дополнительных структур). При этом наименьшее время выполнения показывает MPI-версия со сборкой невязки по треугольным элементам (рис. 3).

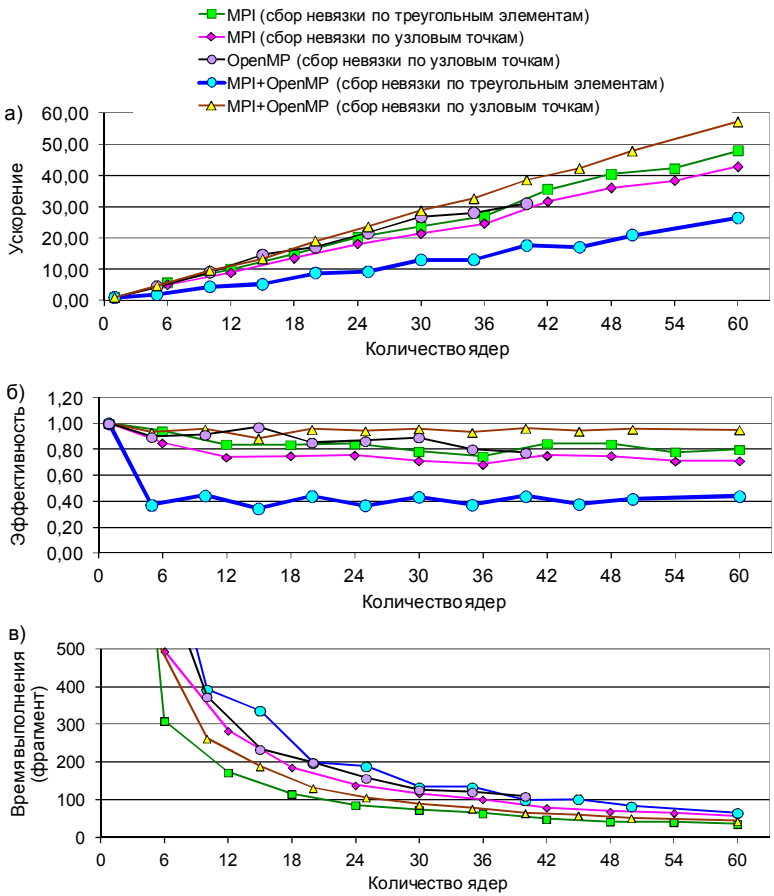


Рис. 3. Графики зависимости ускорения вычислений (а), эффективности (б) и времени выполнения (в) от количества используемых ядер для MPI-, OpenMP- и MPI+OpenMP-версий программ

Литература

1. *Agoshkov V.I.* Inverse problems of the mathematical theory of tides: boundary-function problem // Russ. J. Numer. Anal. Math. Modelling. 2005. Vol. 20, №1. P. 1–18.

2. *Kamenshchikov L.P., Karepova E.D., Shaidurov V.V.* Simulation of surface waves in basins by the finite element method // Russian J. Numer. Anal. Math. Modelling. 2006. Vol. 21(4). P. 305–320.

3. *Карепова Е.Д., Шайдуров В.В.* Параллельная реализация МКЭ для начально-краевой задачи мелкой воды // Вычислительные технологии. 2009. Т.14, № 6. С. 45–57.

4. *Ильин В.П.* Методы и технологии конечных элементов. Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.

5. *Smith I. M., Griffits D. V.* Programming the finite element method. Wiley, Chichester, 2004.

6. *Vollaire C., Nicolas L., Nicolas A.* Parallel computing for the finite element method // The European Physical Journal Applied Physics. 1998. V. 1(3). P. 305–314.

7. *Jimack P. K., Touheed N.* Developing Parallel Finite Element Software Using MPI // High Performance Computing for Computational Mechanics. 2000. P.15–38.

8. *Pantale O.* Parallelization of an object-oriented FEM dynamics code: influence of the strategies on the Speedup // Advances in Engineering Software. 2005. V. 36(6). P. 361–373.

9. *Mahinthakumar G., Saied F.* A Hybrid MPI-OpenMP Implementation of an Implicit Finite-Element Code on Parallel Architectures // International Journal of High Performance Computing Applications. 2002. V. 16(4). P. 371–393,

10. *Vargas-Felix M., Botello-Rionda S.* Solution of Finite Element Problems Using Hybrid Parallelization with MPI and OpenMP // Acta Universitaria. 2012. V. 22(7). P.14–24.

11. *Ильин В. П.* Параллельные алгоритмы для больших прикладных задач: проблемы и технологии // Автометрия. 2007. Т. 43, № 2. С. 3– 21.

12. *Karepova E., Shaidurov V., Dementyeva E.* The numerical solution of data assimilation problem for shallow water equations // Int. J. of Num. Analysis and Modeling, Series B. 2011. V.2, № 2–3. P.167–182.

13. *Дементьева Е.В., Карепова Е.Д., Малышев А.В.* Эффективность численного моделирования на кластерных системах распространения поверхностных волн // Вестник НГУ. Серия: Информационные технологии. 2011. Т. 9, № 1. С. 11–20.