2. EclipseLink Developer Guide [Электронный ресурс]. – https://www.eclipse.org/eclipselink/releases/2.6.php.

А.А. Столбова, С.В. Востокин, С.Н. Попов

ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ

(Самарский национальный исследовательский университет имени академика С.П. Королёва)

В настоящее время для анализа сигналов широко используется вейвлет-преобразование [1]:

$$W_{\psi}(a_i, b_j) = \frac{1}{\sqrt{a_i}} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-b_j}{a_i}\right)dt,$$

где f(t) – случайный процесс, ψ(t) – выбранный анализирующий вейвлет, a≠0 – параметр масштаба, b≥0 – параметр сдвига.

Одной из прикладных задач вейвлет-преобразования является анализ акустических сигналов при тестировании деталей на прочность. Часто такие сигналы имеют большой объем и, следовательно, решение данной задачи при помощи непрерывного вейвлет-преобразования занимает много времени.

Целью данной работы является демонстрация возможностей применения высокопроизводительных вычислений на кластерных системах для уменьшения времени проведения вычислительных экспериментов вейвлет-анализа сигналов.

Для проведения вычислительных экспериментов был выбран базовый вейвлет Гаусса 8 порядка [2, 3]:

$$\psi(\tau) = \left(\tau^8 - 28\tau^6 + 210\tau^4 - 420\tau^2 + 105\right)\exp\left(-\frac{\tau^2}{2}\right),$$

где $\tau = \frac{t-b}{a}$.

В качестве анализируемого сигнала использовался акустический сигнал с интервалом дискретизации $\Delta t = 5 \cdot 10^{-7}$ и числом отсчетов N=1500000.

Для распараллеливания использовалась технология OpenMP [4], предназначенная для вычислительных систем с общей памятью. Для того чтобы получить параллельную версию программы, необходимо определить ресурс её параллелизма, то есть найти в ней участки, которые могут выполняться независимо. Структура численного метода, реализующего непрерывное вейвлет-преобразование такова, что он имеет большой ресурс параллелизма. Каждый отсчет функции W_{Ψ} может вычисляется независимо и одновременно с другими отсчетами. Данное свойство позволило легко преобразовать исходный последовательный алгоритм в параллельный, выделив в нем параллельно выполняющиеся циклы с использованием директив *parallel* и *for* OpenMP.

Вычислительные эксперименты проводились на кластере «Сергей Королёв» с использованием системы TempletWeb [5]. В системе автоматизированы процессы запуска и мониторинга заданий, имеется возможность совместного редактирования кода и данных, а также визуализации результатов экспериментов. Для оценки эффективности распараллеливания полученная программа запускалась на четырёх типах узлов кластера «Сергей Королёв», отличающихся производительностью и количеством вычислительных ядер. Результаты экспериментов представлены в таблице 1.

Таблица 1 – Ускорение в параллельном алгоритме вейвлет-преобразования для разных типов узлов кластера «Сергей Королёв»

Количество процессов на узел	Время выполнения последовательной программы, с	Время выполнения параллельной программы, с	Ускорение
8	5508	731	7,53
12	5294	460	11,51
16	4146	267	15,53
20	3576	184	19,43

Результаты экспериментов позволяют сделать следующие выводы. Полученное ускорение практически совпадает с количеством процессов на узел, что говорит об эффективном распараллеливании программы вейвлет-преобразования. Отношение времени выполнения последовательной программы на маломощном процессоре кластера, аналогичном процессорам современных настольных систем, ко времени выполнения параллельной программы на наиболее производительном узле равно 29,93. Это говорит о том, что применение высокопроизводительной техники реализации непрерывного вейвлет-преобразования является обоснованным и необходимым, так как позволяет получить значительное уменьшение времени счета. Имеющиеся технические средства позволяют значительно снизить трудоёмкость организации вычислительных экспериментов.

Выражаем благодарность за предоставленные данные лаборатории спектрального анализа Испытательного центра Тольяттинского государственного университета.

Литература

1. Витязев, В.В. Вейвлет-анализ временных рядов: учебное пособие / В.В. Витязев. – СПб: Изд-во С.-Петерб. ун-та, 2001. – 58 с.

2. Яковлев, А.Н. Введение в вейвлет-преобразования: Учеб. Пособие / А.Н. Яковлев. – Новосибирск: Издательство НГТУ, 2003. – 104 с.

3. Прохоров, С.А. Паттерновое проектирование при создании комплекса программ для проведения вейвлет-анализа / С.А. Прохоров, А.А. Столбова // Известия СНЦ РАН. – Самара, 2015. – т. 17, № 2(5). – С. 1092-1096.

4. OpenMP [Электронный ресурс] // OpenMP Homepage. – Режим доступа: http://www.openmp.org.

5. Артамонов, Ю.С. Инструментальное программное обеспечение для разработки и поддержки исполнения приложений научных вычислений в кластерных системах / Ю.С.Артамонов, С.В. Востокин // Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки, 19:4 (2015), 785–798.

М.В. Терёхин, Е.И. Чигарина

ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ РАНЖИРОВАНИЯ В СИСТЕМАХ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

(Самарский университет)

В системах баз данных одной из основных задач является задача сокращения времени выполнения запросов при работе с данными с использованием минимального объема памяти, то есть задача эффективного выполнения запросов. В данной работе выполнен анализ эффективности реализации запросов с использованием функций ранжирования по сравнению с традиционными запросами без них.

Функции ранжирования возвращают ранг каждой записи внутри «окна». В общем случае рангом является число, отражающее положение или «вес» записи относительно других записей в том же наборе. Формируется «окно» с помощью группировки. Однако, поскольку результат работы функций ранжирования зависит от порядка обработки записей, то обязательно должен быть указан порядок записей внутри «окна» посредством конструкции ORDER BY. Функции ранжирования являются недетерминированными, то есть при одних и тех же входных значениях они могут возвращать разный результат [1].

Существует четыре функции ранжирования:

- ROW_NUMBER() – функция, выполняющая нумерацию записей в указанном порядке внутри «окна»;
- RANK() – функция, раздающая такие номера записям, что если встретятся несколько записей с одинаковым значением, то этим записям будет присвоен одинаковый ранг. Следующая запись с новым значением получит такой ранг, как будто бы предыдущие записи получили свой уникальный номер, то есть образуется пропуск в нумерации;
- DENSE_RANK() – функция, такая же, что и RANK(), но без пропусков ранга;
- NTILE() – функция, разделяющая записи на указанное количество групп.

Общая структура функций ранжирования имеет вид:

ФУНКЦИЯ_РАНЖИРОВАНИЯ ( )