ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОИСКА ПОХОЖИХ ПОДПОСЛЕДОВАТЕЛЬНОСТЕЙ ВРЕМЕННОГО РЯДА ДЛЯ СОПРОЦЕССОРА INTEL XEON PHI

А.В. Мовчан, М.Л. Цымблер

Южно-Уральский государственный университет (Челябинск)

Введение. *Временной ряд (time series)* представляет собой совокупность вещественных значений, каждое из которых ассоциировано с последовательными отметками времени. Задача *поиска похожих подпоследовательностей (subsequence matching)* определяется следующим образом [4]. Пусть дан временной ряд T, его подпоследовательности мы обозначаем как T_{ij}, где i<j – номера членов ряда; пусть задан *запрос Q* – временной ряд с длиной, не превышающей длину ряда T; имеется *функция схожести* $D(t_1,t_2)$, определяющая расстояние между двумя рядами. Необходимо найти подпоследовательности T_{ij}, имеющие длину, равную длине запроса, для которых значение функции $D(T_{ij},Q)$ минимально.

Задача поиска похожих подпоследовательностей временного ряда возникает в широком спектре научных и прикладных задач из различных предметных областей: медицина [10], финансы [17], анализ видео [2] и др. В соответствии с современными тенденциями развития аппаратного обеспечения актуальным подходом к решению данной задачи является разработка параллельных алгоритмов, совместно использующих центральный процессор и многоядерные ускорители. В данной работе предлагается параллельный алгоритм поиска похожих подпоследовательностей для многоядерного сопроцессора Intel Xeon Phi, который в настоящее время активно применяется для решения различных вычислительно сложных научных задач [14, 16].

Статья организована следующим образом. В разделе 1 представлен обзор существующих алгоритмов поиска похожих подпоследовательностей временного ряда. Раздел 2 содержит описание предлагаемого алгоритма. В разделе 3 рассмотрены результаты вычислительных экспериментов, в которых исследуется эффективность предложенного алгоритма. В заключении суммируются полученные результаты и рассматриваются направления дальнейших исследований в данной области.

1. Обзор работ в области интеллектуального анализа временных рядов. В настоящее время существует большое количество алгоритмов поиска похожих подпоследовательностей, которые различаются используемым подходом к определению схожести временных рядов [5].

В одном из первых исследований [4] временной ряд рассматривается как набор многомерных прямоугольников, по которым строится пространственный индекс, а поиск подпоследовательностей заключается в выполнении пространственных запросов к этому индексу. На основе данной работы другие исследователи улучшили производительность поиска похожих подпоследовательностей. Например, алгоритм DualMatch [6], заключается в разделении временного ряда на непересекающиеся окна и разделении запроса на скользящие окна для составления индекса. В данных алгоритмах используется дискретное преобразование Фурье в качестве оценки евклидовой метрики и используют построение индекса для поиска похожих подпоследовательностей.

Алгоритмы поиска похожих подпоследовательностей, в которых схожесть определяется с помошью алгоритма *динамической трансформации шкалы времени (Dynamic Time Warping, DTW)* [13], используют суффиксные деревья для построения индекса [7], индексы на основе префикса запросов [9], оптимизации последовательного сравнения всех возможных подпоследовательностей [11] и др. Подход, основанный на оптимизациях, менее требователен к памяти, что позволяет осуществлять поиск в очень больших временных рядах, а использование оптимизаций позволяет добиться высокой производительности. Например, алгоритм, описанный в работе [11] и представляющий собой точный последовательный поиск с оптимизациями, позволяет осуществлять поиск во временных рядах длиной в триллион точек данных за приемлемое время.

В работе [15] рассматривается создание многопоточного алгоритма поиска похожих подпоследовательностей. Для определения схожести временных рядов данный алгоритм использует DTW.

В работе [12] рассматривается применение GPU и FPGA для ускорения поиска похожих подпоследовательностей. Однако в данном алгоритме используется полный перебор без оптимизаций.

Алгоритм UCR-DTW [11] является на сегодня, по-видимому, самым быстрым алгоритмом поиска похожих подпоследовательностей среди алгоритмов, использующих алгоритм динамической трансформации шкалы времени для определения схожести временных рядов. Данный алгоритм заключается в переборе всех возможных подпоследовательностей. Перед сравнением к временным рядам применяется *Z-нормализация* (преобразование последовательности, обеспечивающее приблизительное равенство нулю среднего арифметического, и приблизительное равенство единице среднеквадратичного отклонения) [6]. Для ускорения вычислений в данном алгоритме применяется широкий набор оптимизаций (использование квадрата

расстояния, использование оценок DTW, заблаговременная отмена вычислений DTW и Z-нормализации, переупорядочивание вычислений оценки LB_{Keogh}, обмен ролей запроса и данных в LB_{Keogh} и каскадное применение оценок).

В следующем разделе мы покажем, как можно ускорить поиск похожих подпоследовательностей, используя сопроцессор Intel Xeon Phi.

2. Параллельный алгоритм поиска похожих подпоследовательностей для сопроцессора Intel Xeon Phi. Разработка параллельного алгоритма поиска похожих подпоследовательностей для сопроцессора Intel Xeon Phi происходила в два этапа: сначала был разработан параллельный алгоритм на основе технологии OpenMP, а затем разработанный параллельный алгоритм был адаптирован для сопроцессора Intel Xeon Phi [1].

Далее мы рассмотрим параллельный алгоритм, основанный на технологии OpenMP. Параллельный алгоритм использует параллелизм по данным, разбивая временной ряд на равные участки, для обработки каждого из них отдельной OpenMP-нитью. Диаграмма деятельности данного алгоритма приведена на рис. 1. UCR-DTW представляет собой подпрограмму, реализующую последовательный алгоритм.

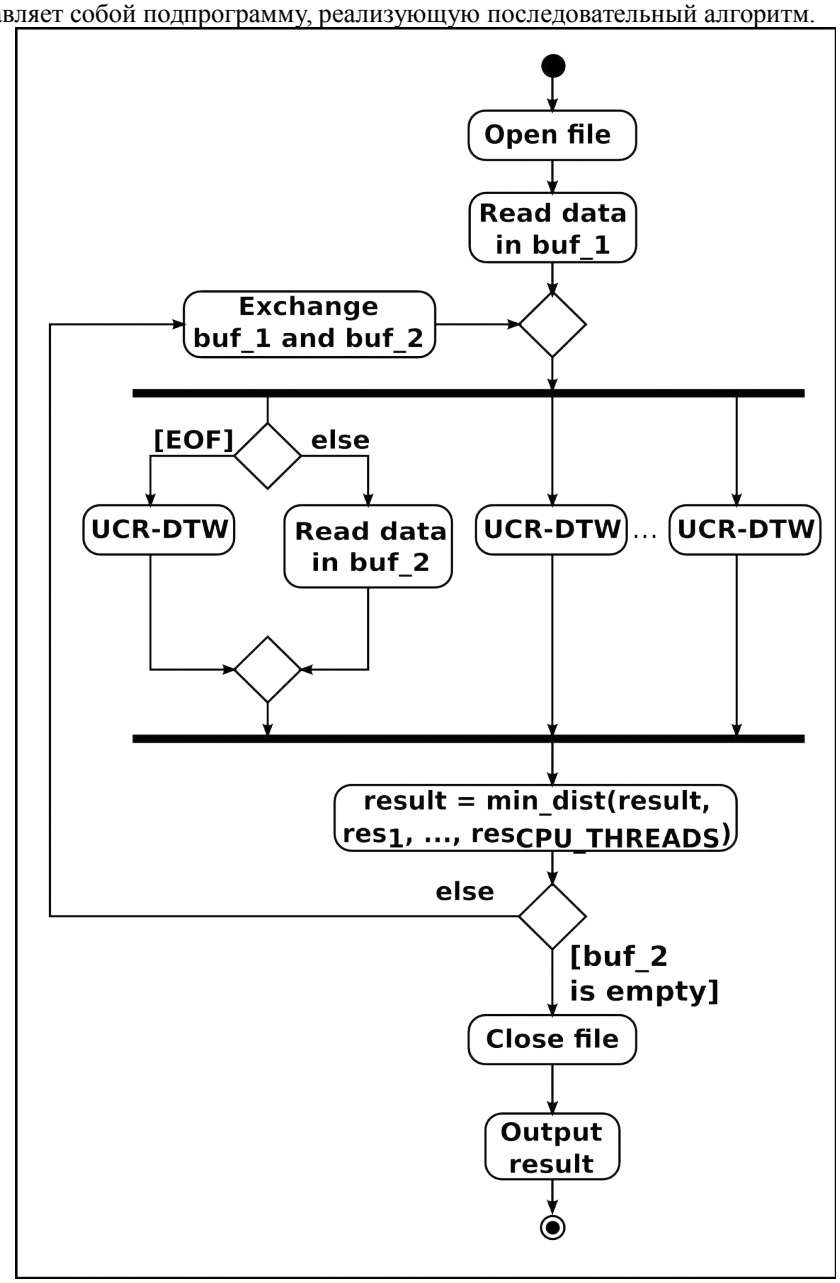

Рис. 1. Диаграмма деятельности параллельного алгоритма

В подпрограмме UCR-DTW используется общая (shared) переменная, которая хранит информацию о расстоянии до ближайшей подпоследовательности. Эта информация позволяет каждой нити отбрасывать неподходящие подпоследовательности, используя оценки, и реже вычислять расстояние с использованием DTW. Нить-мастер выполняет считывание новых данных из файла одновременно с обработкой уже считанных данных, что позволяет уменьшить последовательную часть программы.

Далее мы рассмотрим параллельный алгоритм для сопроцессора Intel Xeon Phi (см. диаграмму деятельности алгоритма на рис. 2). Для организации взаимодействия между процессором и сопроцессором используется *режим выгрузки (offload mode)*. В данном режиме программа выполняется на центральном процессоре, а некоторый код и данные выгружаются для выполнения на сопроцессоре. Работа организуется следующим образом.

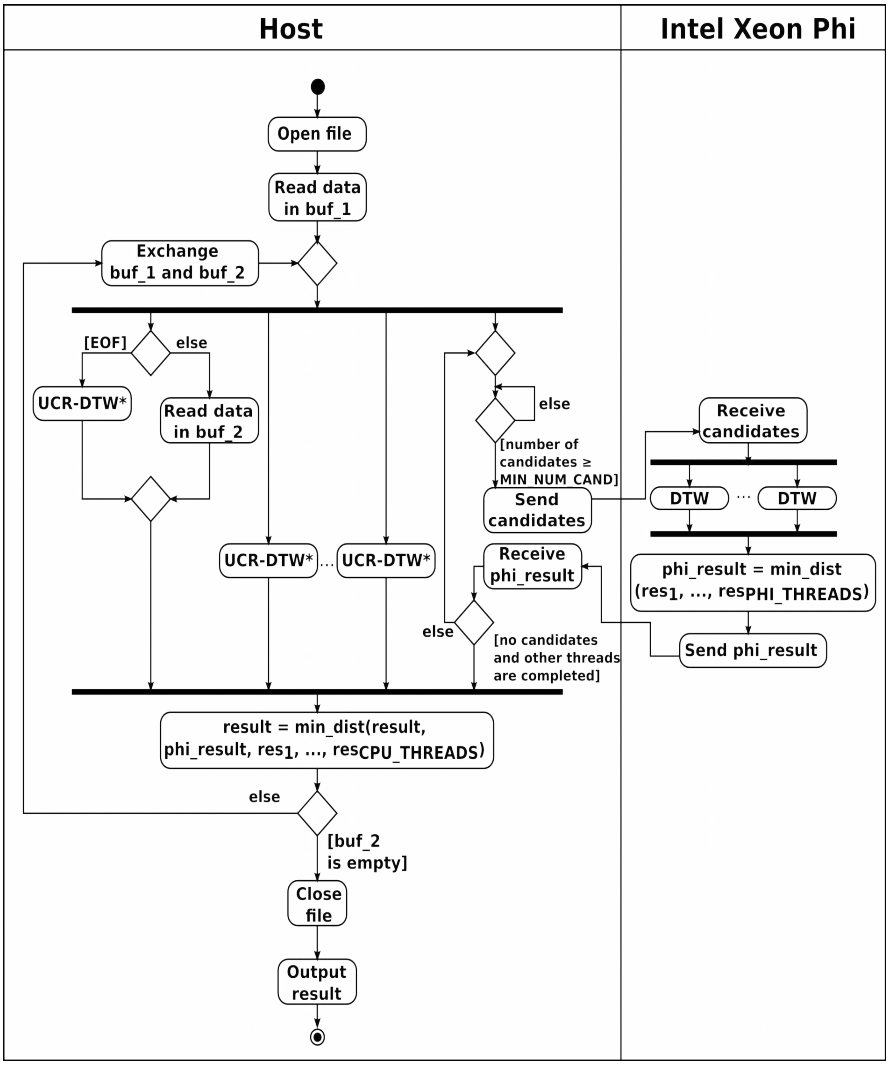

Рис. 2. Диаграмма деятельности параллельного алгоритма для сопроцессора Intel Xeon Phi

Одна из нитей процессора управляет работой сопроцессора, а остальные нити центрального процессора выполняют подпрограмму UCR-DTW* для выделенного им участка временного ряда. Подпрограмма UCR-DTW* параллельного алгоритма для сопроцессора является модифицированной версией подпрограммы UCR-DTW параллельного алгоритма для процессора. Подпрограмма UCR-DTW* также реализует последовательный алгоритм, но при необходимости произвести расчет DTW либо производит расчет, либо добавляет подпоследовательность в очередь для последующего вычисления DTW на сопроцессоре.

Поток, управляющий работой сопроцессора, следит за количеством подпоследовательностей в очереди. Если количество подпоследовательностей превышает `MIN_NUM_CAND`, то `MIN_NUM_CAND` подпоследовательностей из очереди отправляются на сопроцессор, где для каждой подпоследовательности выполняется расчет DTW.

Рассмотрим работу подпрограммы UCR-DTW*, диаграмма деятельности которой приведена на рис. 3. В данной подпрограмме осуществляется перебор всех возможных подпоследовательностей, длина которых равна длине запроса. Для каждой подпоследовательности производится каскадное вычисление оценок. Если одна из оценок превышает `bsf` (расстояние до самой похожей подпоследовательности), то подпоследовательность отбрасывается. Если ни одна из оценок не отбросила подпоследовательность, то проверяется количество подпоследовательностей в очереди. Если количество подпоследовательностей превышает `MAX_NUM_CAND`, то вычисление DTW выполняется на процессоре, иначе подпоследовательность добавляется в очередь для последующего вычисления на сопроцессоре.

Кроме информации о подпоследовательности, на сопроцессор пересылается массив, содержащий информацию об оценке LB_{Keogh} для данной подпоследовательности. Эта информация используется для заблаговременного отмена расчета DTW.

Важным инструментом ускорения вычислений для сопроцессора является векторизация. Векторизация позволяет выполнять набор однотипных операций над несколькими элементами массива одновременно. Поэтому очень важно применять векторизацию во всех циклах, в которых это возможно. В связи с этим функция вычисления DTW была модифицирована так, чтобы максимизировать использование векторных инструкций. Для этого внутренний цикл функции был разделен на два цикла, один из который векторизуется.

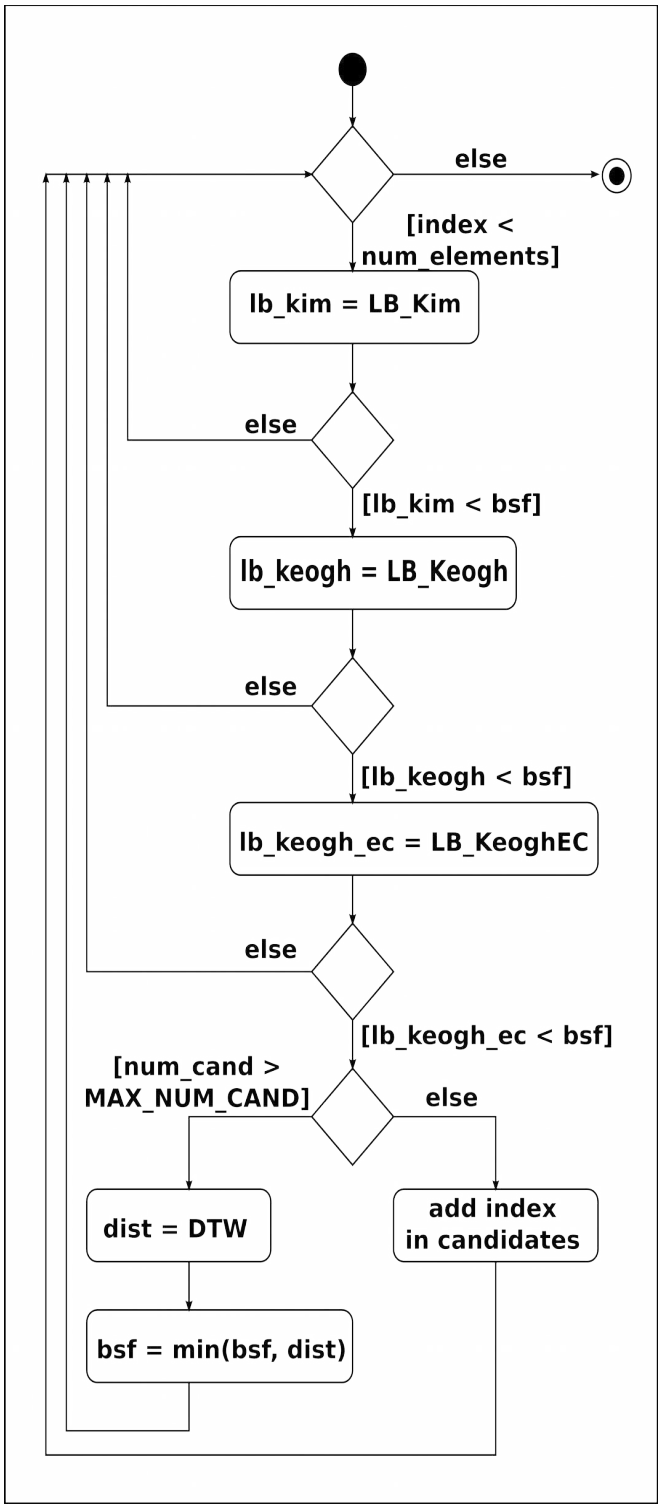

Рис. 3. Диаграмма деятельности подпрограммы UCR-DTW*

Это позволило увеличить скорость вычислений функции DTW.

3. Вычислительные эксперименты. Для исследования эффективности предложенного алгоритма нами проведена серия вычислительных экспериментов, в которых измерялось время выполнения поиска подпоследовательностей при варьируемой длине запроса. В качестве аппаратной платформы экспериментов использовался вычислительный узел суперкомпьютера «Торнадо ЮУрГУ», характеристики которого приведены в табл. 1.

Табл. 1. Аппаратные характеристики вычислительного узла суперкомпьютера «Торнадо ЮУрГУ»

Характеристики	Процессор	Сопроцессор
Модель	Intel Xeon X5680	Intel Xeon Phi SE10X
Количество ядер	6	61
Частота ядер, GHz	3.33	1.1
Количество потоков на ядро	2	4
Производительность, Тфлопс	0.371	1.076

На рис. 4 приведены результаты экспериментов на синтетических данных, которые были получены с помощью *модели случайных блужданий (random walk)*. Случайное блуждание представляет собой математическую модель процесса случайных изменений как шагов в дискретные моменты времени. При этом предполагается, что изменение на каждом шаге не зависит от предыдущих и от времени. Чтобы сформировать одномерный временной ряд, мы использовали одномерное дискретное случайное блуждание, которое начинается с 0 и на каждом шаге увеличивается или уменьшается на определенное значение с одинаковой вероятностью.

В данных экспериментах временной ряд состоял из 100 миллионов элементов. Время выполнения приведено без учета времени загрузки данных в память, поскольку в данном случае типичным сценарием работы будет однократная загрузка данных в память и последующее многократное выполнение поиска похожих подпоследовательностей. Результаты показывают, что параллельный алгоритм для сопроцессора более эффективен при большой длине запроса. При небольшой длине запроса время выполнения параллельного алгоритма для сопроцессора практически равно скорости выполнения параллельного алгоритма для процессора.

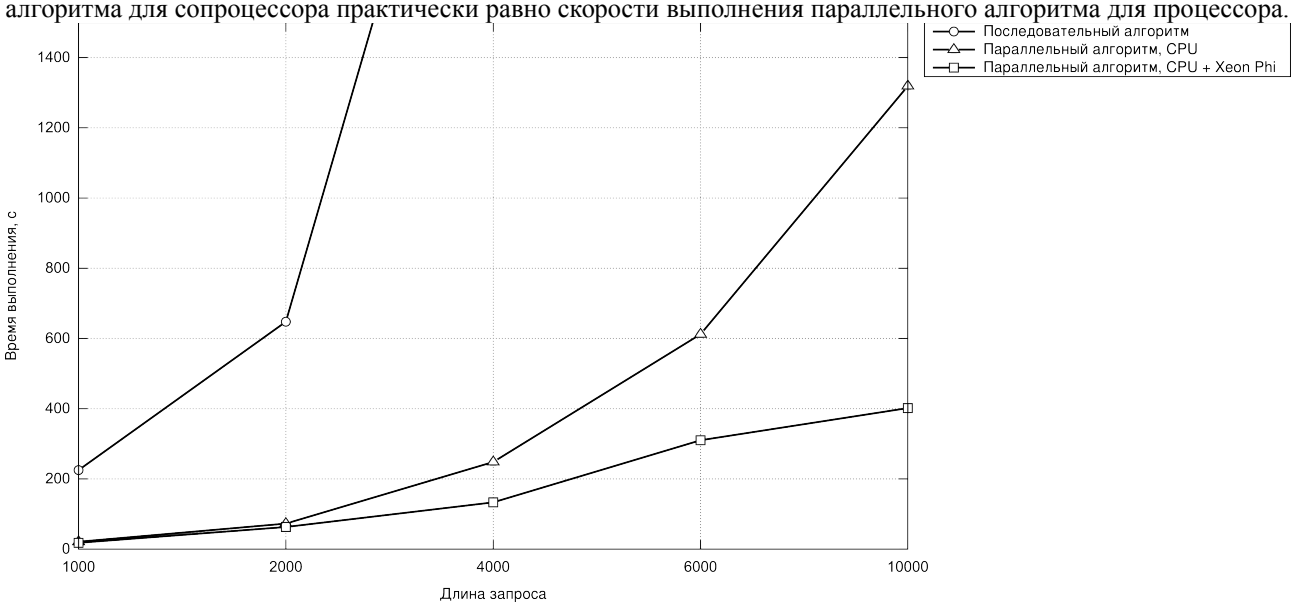

Рис. 4. Сравнение производительности разработанных алгоритмов на синтетических данных

На рис. 5 приведены результаты экспериментов на реальных данных, в качестве которых использовались данные электрокардиограммы человека. Длина временного ряда, используемого в этих экспериментах, составляет более 20 млн. элементов, что соответствует примерно 22 час. при частоте снятия показателей ЭКГ 250 Гц. Использование сопроцессора позволило уменьшить время выполнения в 3 раза по сравнению с параллельным алгоритмом, использующим только центральный процессор.

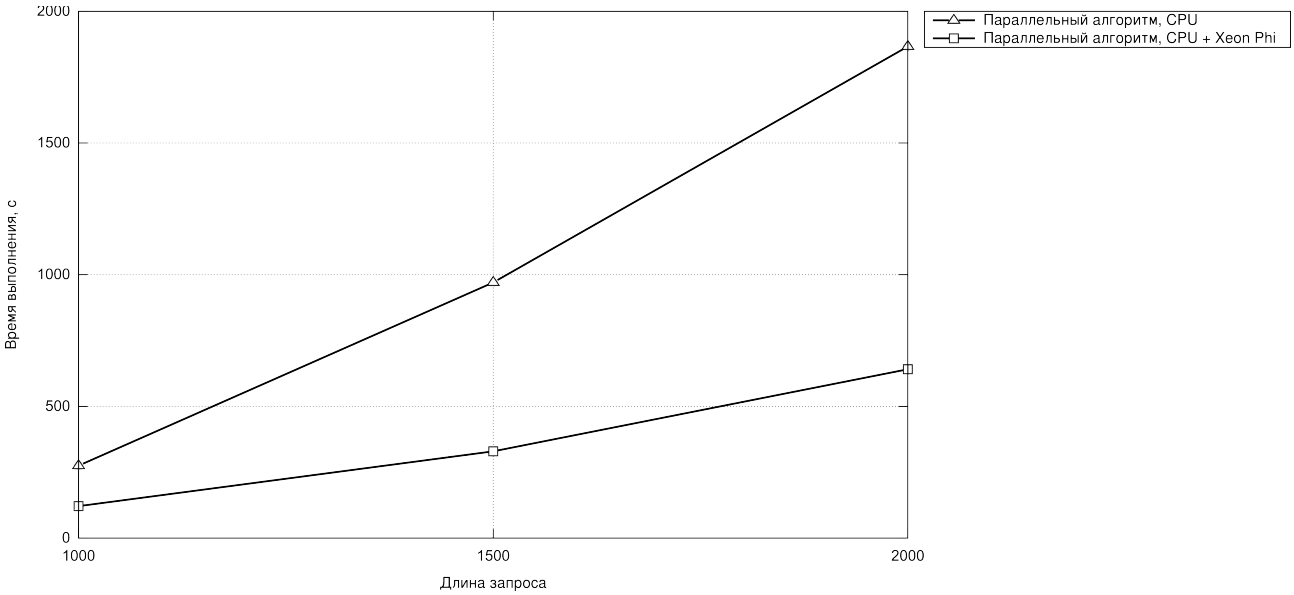

Рис. 5. Сравнение производительности разработанных алгоритмовна реальных данных

Заключение. В статье рассмотрена разработка параллельного алгоритма поиска похожих подпоследовательностей для сопроцессора Intel Xeon Phi. Для реализации параллельного алгоритма использовался параллелизм по данным, технология параллельного программирования OpenMP, а взаимодействие между центральным процессором и сопроцессором Intel Xeon Phi осуществлялось с помощью режима выгрузки. Для увеличения производительности параллельного алгоритма использована векторизация циклов. Представлены результаты вычислительных экспериментов с синтетическими и реальными данными, подтверждающие лучшую эффективность предложенного алгоритма по сравнению с параллельным алгоритмом, использующим только центральный процессор. Использование сопроцессора Intel Xeon Phi дает наилучшие результаты при большой длине запроса.

В качестве возможного направления дальнейших исследований интересными представляются две задачи: модернизация разработанного алгоритма для случая вычислительного узла с несколькими сопроцессорами Intel Xeon Phi и расширение данного алгоритма для кластерной системы, вычислительные узлы которой оснащены сопроцессорами Intel Xeon Phi.

Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта № 12-07-00443-а.

ЛИТЕРАТУРА:

1. Мовчан А.В., Цымблер М.Л. Разработка параллельного алгоритма поиска похожих подпоследовательностей временного ряда для сопроцессора Intel Xeon Phi // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1-3 апреля 2014 г., г. Ростов-на-Дону). Челябинск: Издательский центр ЮУрГУ, 2014. С. 372.
2. Chen Y., Chen G., Chen K., Ooi B. C. Efficient Processing of Warping Time Series Join of Motion Capture Data // IEEE International Conference on Data Engineering, Shanghai, 29 March - 2 April, 2009. IEEE, 2009. P. 1048-1059.
3. Ding H., Trajcevski G., Scheuermann P., Wang X., Keogh E. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures // Proceedings of the VLDB Endowment, 2008. Vol. 1, No. 2. P. 1542-1552.
4. Faloutsos C., Ranganathan M., Manolopoulos Y. Fast Subsequence matching in Time-series Databases // The 1994 ACM SIGMOD international conference on Management of data, New York, NY, 24-27 May, 1994. ACM, 1994. P. 419-429.
5. Fu T.-C. A Review on Time Series Data Mining // Engineering Applications of Artificial Intelligence, 2011. Vol. 24, No. 1. P. 164-181.
6. Goldin D., Kanellakis P. On Similarity Queries for Time-Series Data: Constraint Specification and Implementation // The First International Conference on Principles and Practice of Constraint Programming, Cassis, France, 19-22 September, 1995. Springer-Verlag, 1995. P. 137-153.
7. Kim S.W., Yoon J., Park S., Kim T.H. Shape-based Retrieval of Similar Subsequences in Time-series Databases // The 2002 ACM symposium on Applied computing, Madrid, Spain, 10-14 March, 2002. ACM, 2002. P. 438-445.
8. Moon Y.-S., Whang K.-Y., Loh W.-K. Duality-Based Subsequence Matching in Time-Series Databases // The 17th International Conference on Data Engineering, Washington, DC, USA, 2-6 April, 2001. IEEE Computer Society, 2001. P. 263-272.

9. Park S., Kim S.W., Cho J.S., Padmanabhan S. Prefix-querying: an Approach for Effective Subsequence Matching under Time Warping in Sequence Databases // The 10th international conference on Information and knowledge management, Atlanta, Georgia, USA, 5-10 November, 2001. ACM, 2001. P. 255-262.
10. Raghavendra B.S., Bera D., Bopardikar A.S., Narayanan R. Cardiac Arrhythmia Detection Using Dynamic Time warping of ECG beats in e-healthcare systems // World of Wireless, Mobile and Multimedia Networks (WoWMoM), Lucca, Italy, 20-24 June, 2011. IEEE, 2011. P. 1-6.
11. Rakthanmanon T., Campana B., Mueen A., Batista G., Westover B., Zhu Q., Zakaria J., Keogh E. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping // The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Beijing, China, 12-16 August, 2012. ACM, 2012. P. 262-270.
12. Sart D., Mueen A., Najjar W., Keogh E., Niennattrakul V. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs // The 10th IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13-17 December, 2010. IEEE, 2010. P. 1001-1006.
13. Bellman R., Kalaba R. On adaptive control processes // IEEE Transactions on Automatic Control, 1959. Vol. 4, No. 2. P. 1-9.
14. Sinha P., Misra G., Vikranman D., Das A. Biosequence Analysis Using Intel® Xeon Phi // 7th UKSim/AMSS European Modelling Symposium, Manchester, UK, 20-22 November, 2013. IEEE, 2013. P. 497-499.
15. Srikanthan S., Kumar A., Gupta R. Implementing the Dynamic Time Warping Algorithm in Multithreaded Environments for Real Time and Unsupervised Pattern Discovery // Computer and Communication Technology (ICCCT), Allahabad, India, 15-17 September, 2011. IEEE Computer Society, 2011. P. 394-398.
16. Woop S., Feng L., Wald I., Benthin C. Embree Ray Tracing Kernels for CPUs and the Xeon Phi Architecture // International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, Anaheim, CA, USA, 21-25 July, 2013. ACM, 2013. P. 44.
17. Wu H., Salzberg B., Zhang D. Online Event-driven Subsequence Matching over Financial Data Streams // The 2004 ACM SIGMOD international conference on Management of data, Paris, France, 13-18 June, 2004. ACM , 2004. P. 23-34.