УДК 004.942

А.М. Кориков, В.В. Симонов

Гибридная архитектура параллельных вычислительных систем

Предложена гибридная архитектура организации высокопроизводительных параллельных вычислений. На этой основе разработана эффективная с точки зрения быстродействия гибридная параллельная версия функции SSTEBZ из библиотеки LAPACK для поиска собственных чисел трехдиагональных симметричных матриц методом деления отрезков. Использование графического процессора в качестве ускорителя параллельных вычислений позволило обеспечить значительный прирост быстродействия гибридного алгоритма по сравнению со стандартной реализацией. Представленный гибридный параллельный алгоритм обладает вычислительной устойчивостью.

Ключевые слова: гибридная архитектура, собственные числа, ускоритель параллельных вычислений, гибридный параллельный алгоритм.

Эволюция средств вычислительной техники (СВТ) требует развития новых подходов к программированию высокопроизводительных вычислений и, в частности, эта эволюция привела к возникновению и развитию GPGPU (General-Purpose computing on Graphics Processing Units – вычисления общего назначения на графических процессорах) [1] – семейства технологий для ускорения вычислений общего назначения с помощью графических процессоров. Анализ достоинств и недостатков данных технологий ускорения вычислений доказывает необходимость разработки гибридных параллельных программ для центрального и графического процессоров, так как начиная с 2004 г. частота процессоров практически перестала расти. Производительность отдельно взятого ядра процессора ограничена частотой, поэтому скорость выполнения последовательных программ также ограничена, и для достижения высокой производительности в научных расчетах требуется разрабатывать параллельные программы.

К настоящему времени налажено массовое производство процессоров, имеющих 2, 4 и даже 6 ядер, их количество будет только расти. Компания Intel представила экспериментальный чип MIC с 80 интегрированными x86 процессорами, что подтверждает отмеченную тенденцию.

Ранее созданные программы не могут применить даже половину вычислительной мощности современных процессоров, так как не используют параллельные вычисления, а потенциал увеличения производительности отдельно взятого процессорного ядра практически исчерпан. Быстродействие ранее созданных алгоритмов на современных компьютерах повышается медленнее роста сложности задач, что приведет в итоге к неэффективности систем, использующих эти алгоритмы.

Согласно закону Амдала–Уэра, последовательные участки параллельной программы также ограничивают рост скорости их выполнения с ростом количества вычислителей. Джин Амдал сформулировал закон в 1967 г., обнаружив простое по существу, но непреодолимое по содержанию ограничение на рост производительности при распараллеливании вычислений: «В случае, когда задача разделяется на несколько частей, суммарное время её выполнения на параллельной системе не может быть меньше времени выполнения самого длинного фрагмента» [2]. Согласно этому закону ускорение выполнения программы за счёт распараллеливания её инструкций на множестве вычислителей ограничено временем, необходимым для выполнения её последовательных инструкций.

Так как частота процессоров не растет, все будущие улучшения производительности СВТ могут быть за счет увеличения параллельности, а ограничения по рассеиваемой мощности требуют использования большого числа простых процессоров, работающих на низкой частоте, что еще более осложняет проблему.

1. Гибридные параллельные вычислительные системы: постановка задачи

К настоящему времени арсенал теоретических разработок и методов организации параллельных вычислений весьма ограничен. Поэтому необходим поиск новых методов распараллеливания алгоритмов и, в частности, разработка методов выделения шагов, способных выполняться параллельно. Современные GPU (Graphics Processing Unit – графические процессоры) содержат сотни

простых, эффективных в плане потребления энергии ядер, оптимизированных для обработки интенсивных потоков данных [3]. Многоядерные x86 процессоры содержат несколько ядер, оптимизированных для быстрого выполнения одного потока, но потребляют намного больше энергии для выполнения одной операции.

Для улучшения отношения производительности приложений к потребляемой мощности вполне естественно перенести интенсивную обработку данных на оптимизированные для этого процессоры GPU, а CPU использовать для оставшихся последовательных вычислений. В этом суть предлагаемой нами гибридной архитектуры. Так как программист может переносить интенсивные параллельные вычисления на GPU, а последовательные на CPU, то можно сосредоточиться на создании более энергоэффективных GPU, а уделом CPU останется оптимизация последовательных вычислений.

Компания Intel предложила похожий подход в проекте MIC (Many Integrated Core – англ. много интегрированных ядер) [4]. Экспериментальный чип от компании Intel содержит 80 простых ядер архитектуры Pentium, имеющих по 2 блока для операций с плавающей точкой и 5 портов для обмена сообщениями с другими ядрами на чипе. При производительности в 1 TFLOP, при частоте 3,16 ГГц и напряжении питания 0,95 В чип потребляет всего 62 Вт мощности. Компания Intel не сделала свой аналог GPU, она использовала дизайн процессора Pentium примерно 15-летней давности, чтобы процессор был простым, и добавила поддержку векторных операций. В итоге получен чип, значительно превосходящий процессор XEON по энергоэффективности. Одним из достоинств MIC является поддержка x86 инструкций, т.е. для ускорения вычислений не обязательно перекомпилирование существующих программ. Однако подход, предложенный компанией Intel, порождает проблемы, суть которых излагается ниже.

Современные высокопроизводительные приложения используют MPI или OpenMP для распараллеливания вычислений.

С точки зрения MPI MIC чип представляет собой кластер из 80 процессоров в одном чипе. На практике это может породить следующие проблемы. MPI не позволяет процессам пользоваться разделяемой памятью, поэтому при выполнении 80 процессов с неразделяемой памятью в системе быстро возникнет нехватка памяти. Кроме того, остается в силе закон Джина Амдала, ограничивающий ускорение параллельных программ. В данном случае, производительность ограничена производительностью простых Pentium-подобных ядер [5].

При использовании OpenMP [8] ситуация немного лучше – отпадают проблемы с неразделяемой памятью и очень интенсивным обменом MPI-сообщениями. Возможно, самое большое ограничение – это то, что многие программы не предполагают такой уровень распараллеливания, чтобы загрузить работой более 50 ядер. OpenMP-программа из-за высоких накладных расходов на переключение потоков может эффективно работать или на 50^{+}-ядерном процессоре, или на 4–6-ядерном, но не на обоих сразу.

В 2011 г. компания AMD представила так называемые APU (Accelerated Processing Unit – ускоритель вычислений) – гибридные ускорители вычислений. APU представляет собой чип, на котором размещены сразу и CPU, и GPU. Доступ к возможностям APU может осуществляться с помощью AMD APP или OpenCL.

Изложенное выше подтверждает перспективность разработки гибридных вычислительных систем. Эту перспективу далее раскроем на частной, но важной для многих приложений задаче поиска собственных чисел для симметричной трехдиагональной матрицы. Эта задача возникает при управлении стохастическими системами, имитационном моделировании сложных систем, обучении и т.п., когда требуется получить набор случайных векторов с нормальным распределением, заданным ковариационной матрицей. Для генерации случайных векторов необходимы собственные числа и собственные векторы матрицы ковариации. Существует множество алгоритмов для решения данной задачи [6].

Большинство методов поиска собственных чисел для симметричной матрицы работает с трехдиагональными симметричными матрицами. Трехдиагональная матрица – матрица, в которой заданы главная диагональ и по одной диагонали выше и ниже главной, остальные элементы равны нулю. Симметричная трехдиагональная матрица размера $n \times n$ может быть описана двумя векторами: вектором главной диагонали – d длиной n и вектором диагонали под/над главной – e длиной $n - 1$.

2. Алгоритм бисекции для поиска собственных чисел

В 1954 г. Воллес Гивенс (Wallace Givens) предложил алгоритм бисекции для поиска собственных чисел для вещественной симметричной матрицы. Этот алгоритм основан на вычислении для

некоторого числа x с помощью некоторой функции count(x) собственных чисел меньше x. При этом, собственные векторы могут быть вычислены методом обратной итерации за $O(n^2)$ операций, если собственные числа хорошо разделены и в худшем случае за $10 \cdot O(n^3)$ итераций, если разделены плохо и требуется дополнительная ортогонализация собственных векторов. Алгоритм бисекции позволяет вычислять каждое собственное число независимо от других, что делает его подходящим для параллельных машин. Это его свойство позволяет добиться высокого быстродействия алгоритма на современных машинах.

Отмеченные выше ускорители вычислений (FFT-процессоры, графические процессоры и др.) позволяют ускорить вычисления, параллельные на данных, в десятки и даже сотни раз. При этом ускорители тратят гораздо меньше энергии на одну операцию, чем центральный процессор.

Далее представлена реализация гибридного алгоритма поиска собственных чисел методом бисекции, использующая центральный процессор для управления вычислениями, и OpenCL-ускоритель вычислений [8]. Также приведено сравнение реализации с аналогами по точности и быстродействию.

3. Сравнение реализации алгоритма бисекции с аналогами

Предлагаемый параллельный алгоритм поиска собственных чисел для трехдиагональной симметричной матрицы был реализован на C++ и OpenCL и имеет интерфейс такой же, как у функции SSTEBZ из библиотеки LAPACK [9], что позволяет использовать его совместно с ней. Алгоритм повторения бисекции интервалов реализован на C++ и выполняется центральным процессором, а OpenCL-ядро для бисекции выполняется параллельно для каждого интервала на выбранном пользователем OpenCL-устройстве. В качестве OpenCL-ускорителя могут выступать многоядерный центральный процессор, графический процессор или ускоритель вычислений вроде FFT-процессора. Назовем предложенный алгоритм Hybrid SSTEBZ. В настоящее время Hybrid SSTEBZ реализован только для вещественных чисел с одинарной точностью.

Измерение времени выполнения вычислений производилось на машине с процессором Intel Core 2 Quad 2.83 ГГц, 4 ГБ DDR II ОЗУ и видеокартой ATI Radeon HD7850 2 Гб 920 МГц. Измерения времени производились на поиске собственных чисел трехдиагональной матрицы, заполненной случайными числами из интервала (–1; 1). Для каждого из алгоритмов матрица заполнялась одинаково. Измерения производились по 50 раз. Далее на рис. 1 и 4 приведено среднее время вычисления.

На рис. 1 приведено сравнение времени выполнения предложенного алгоритма с рассмотренными выше аналогами для различных размеров матрицы при использовании только центрального процессора. Это значит, что реализация LAPACK всегда использует только одно ядро центрального процессора, а AMD EigenValues [7, 10] и Hybrid SSTEBZ используют центральный процессор в качестве OpenCL-устройства.

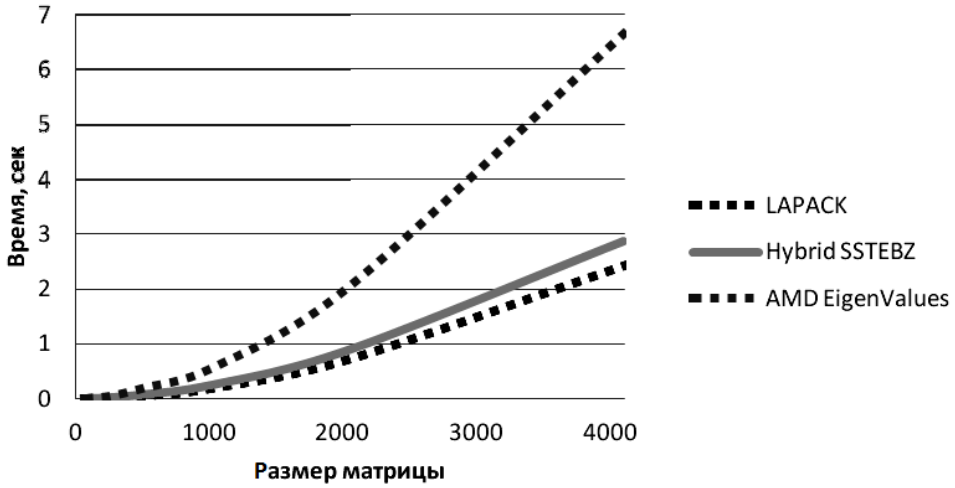

Рис. 1. Сравнение времени вычисления собственных чисел на центральном процессоре для различных размеров матрицы

Из рис. 1 видно, что реализация из библиотеки LAPACK при использовании только центрального процессора (CPU) справляется с вычислениями быстрее своих конкурентов. Даже несмотря на то, что и Hybrid SSTEBZ, и AMD EigenValue используют все четыре процессорных ядра и векториза-

цию [11], а LAPACK – только одно процессорное ядро. Происходит это потому, что алгоритм LAPACK при бисекции интервалов исключает достаточно узкие сошедшиеся интервалы из рассмотрения. Таким образом, алгоритм LAPACK производит примерно в 8 раз меньше арифметических операций, чем Hybrid SSTEBZ, который вынужден рассматривать узкие интервалы снова и снова. Hybrid SSTEBZ работает от 1,2 до 2,3 раза быстрее, чем AMD EigenValues, благодаря исключению лишних вычислений функции count. Отмеченное преимущество предложенного алгоритма Hybrid SSTEBZ перед алгоритмом AMD EigenValue наглядно иллюстрируют рис. 2 и 3.

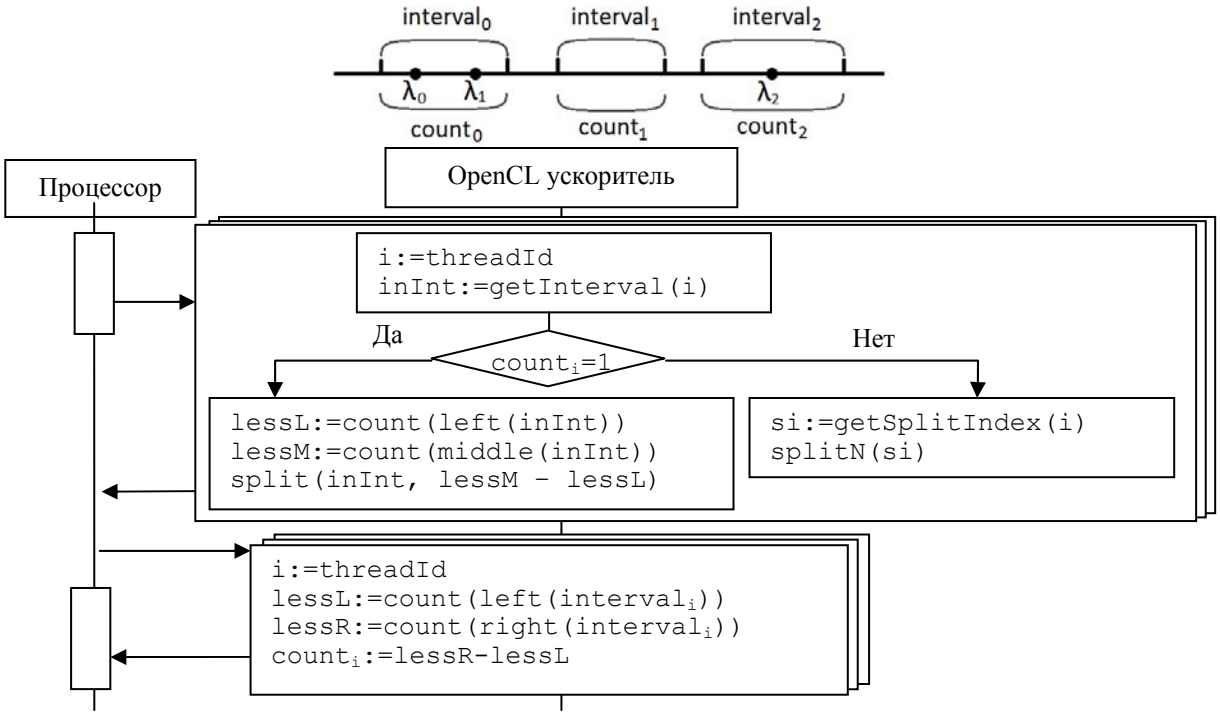

Рис. 2. Алгоритм AMD EigenValue

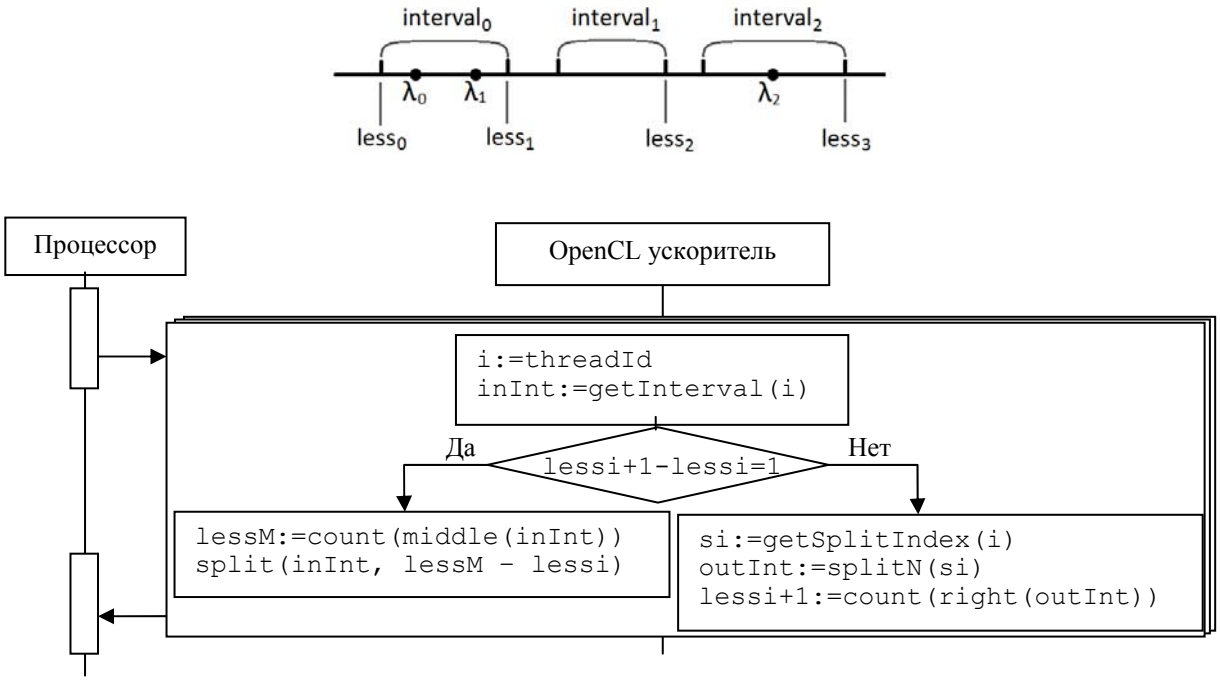

Рис. 3. Алгоритм Hybrid SSTEBZ

AMD EigenValues и Hybrid SSTEBZ были разработаны специально для массивного распараллеливания. Сравним на рис. 4 время выполнения ими вычислений на GPU (Graphics Processing Unit – графический процессор) с реализацией из LAPACK.

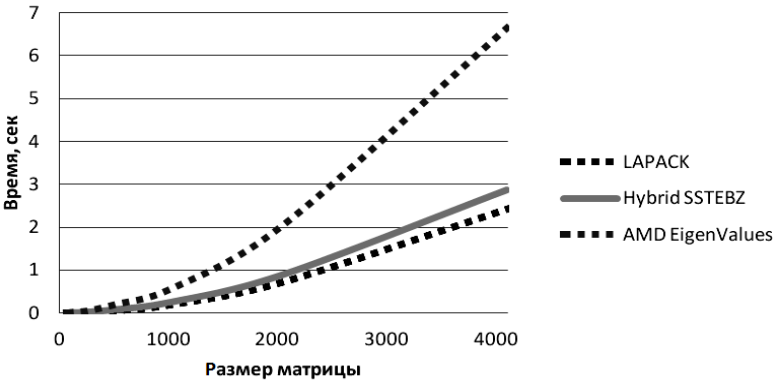

Рис. 4. Сравнение времени вычисления собственных чисел на графическом процессоре ATI HD7850 против LAPACK на центральном процессоре Intel Core 2 Duo 2,83 ГГц

Из рис. 4 видно, что параллельные алгоритмы AMD EigenValues и Hybrid SSTEBZ вычисляют собственные числа на GPU в десятки раз быстрее, чем LAPACK на центральном процессоре. Реализация предложенного алгоритма проигрывает LAPACK в скорости на задачах размерности меньше 128, при размерности 512 Hybrid SSTEBZ уже выигрывает по скорости в 4 раза, а при размерности матрицы 4096 – в 44 раза! При вычислениях на GPU, Hybrid SSTEBZ – от 1,3 до 3 раз быстрее, чем AMD EigenValues.

Также Hybrid SSTEBZ проверен на точность. За эталон точности взята реализация SSTEBZ из LAPACK. Сравнивалось отклонение найденных собственных чисел от AMD EigenValues и Hybrid SSTEBZ от LAPACK. Принималось, что ответ не совпадает с эталонным, если $|a–b| > \max(|a|, |b|)\cdot\text{eps}$, где a и b – эталонный ответ и полученный ответ соответственно, а eps было принято 10^{-6}. Матрицы для проверки генерировались случайным образом, значения элементов выбирались из равномерного случайного распределения в интервале (–1; 1).

AMD EigenValues уже при размере матрицы 128×128 выдавала 15% ошибок на 1000 попыток, и с ростом размера количество ошибок только увеличивалось. Результаты Hybrid SSTEBZ совпадали с эталонными вплоть до размерности матрицы 2048×2048, а при размерности 4096×4096 было обнаружено 15 ошибок на 1000 примеров, что составляет 1,5% от общего числа больших матриц, или 0,2% от общего числа тестов. Причиной этого могло послужить то, что при делении интервала на n интервалов нет проверки count на монотонность.

Заключение. В статье представлена эффективная с точки зрения быстродействия гибридная параллельная версия функции SSTEBZ из библиотеки LAPACK для поиска собственных чисел методом деления отрезков. Использование графического процессора в качестве ускорителя параллельных вычислений позволило обеспечить прирост быстродействия гибридного алгоритма в 44 раза по сравнению со стандартной реализацией. Представленная реализация является устойчивой и выдает точный результат в 99,8% случаев. Несмотря на то, что она выигрывает у последовательного алгоритма по скорости вычислений и масштабируемости, она имеет большую вычислительную сложность. Исследование гибридного параллельного алгоритма Hybrid SSTEBZ подтверждает эффективность и перспективность предлагаемой авторами гибридной архитектуры.

Исследование поддержано грантом по ФЦП «Научные и научно-педагогические кадры инновационной России» (госконтракт № 14.740.11.0398) и проектом 7.701.2011 (НИР 1/12 темплана ТУСУРа) по госзаданию Министерства образования и науки.

Литература

1. Сайт, посвящённый вопросам организации вычислений на GPU [Электронный ресурс]. – Режим доступа: http://gpgpu.org/, свободный (дата обращения: 02.10.2012).

2. Свободная интернет-энциклопедия Wikipedia [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Закон_Амдала, свободный (дата обращения: 02.10.2012).

3. No free lunch for Intel MIC (or GPU's) [Электронный ресурс]. – Режим доступа: http://blogs.nvidia.com/2012/04/no-free-lunch-for-intel-mic-or-gpus/, свободный (дата обращения: 02.10.2012).

4. Many Integrated Core (MIC) Architecture – Advanced [Электронный ресурс]. – Режим доступа: http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html, свободный (дата обращения: 02.10.2012).

5. HPC's Future [Электронный ресурс]. – Режим доступа: http://www.scientificcomputing.com/HPC-Future.aspx, свободный (дата обращения: 02.10.2012).

6. Справочник по теории автоматического управления / под ред. А.А. Красовского. – М.: Наука, 1987. – 712 с.

7. Dhillon I.S. A New O(N^2) Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem / I.S. Dhillon. University of California (Berkeley) – 1997. – 195 с.

8. Открытый стандарт параллельных вычислений OpenCL [Электронный ресурс]. – Режим доступа: http://www.khronos.org/opencl/, свободный (дата обращения: 02.10.2012).

9. LAPACK [Электронный ресурс]. – Режим доступа: http://www.netlib.org/lapack/, свободный (дата обращения: 02.10.2012).

10. AMD Accelerated Parallel Processing SDK [Электронный ресурс]. – Режим доступа: http://developer.amd.com/sdks/AMDAPPSDK/Pages/default.aspx, свободный (дата обращения: 02.10.2012).

11. Autovectorization in Intel® OpenCL SDK 1.5 [Электронный ресурс]. – Режим доступа: http://software.intel.com/en-us/blogs/2011/09/26/autovectorization-in-intel-opencl-sdk-15/, свободный (дата обращения: 02.10.2012).

Кориков Анатолий Михайлович
Д-р техн. наук, зав. каф. автоматизированных систем управления (АСУ) ТУСУРа
Тел.: 8 (382-2) 41-42-79, 8 (382-2) 70-15-36
Эл. почта: korikov@asu.tusur.ru

Симонов Василий Владимирович
Аспирант каф. АСУ
Эл. почта: simonov.vasiliy@gmail.com

Korikov A.M., Simonov V.V.

Hybrid architecture of parallel computation systems

In the paper we offer the hybrid architecture for high performance parallel computations. Based on this architecture, effective in terms of performance, implementation of LAPACK's SSTEBZ algorithm founding eigenvalues for symmetric tridiagonal matrix has been developed. Usage of graphic processing unit as numeric coprocessor allowed to gain high performance of implementation in comparison to a standard one. Algorithm implementation described in this paper has high numeric stability.

Keywords: hybrid architecture, eigenvalues, graphics processing unit, hybrid parallel algorithm.
