Раздел II. Математическое и программное обеспечение суперкомпьютеров

УДК 004.272.32, 519.683.4, 519.683.8 DOI 10.18522/2311-3103-2016-11-2739

В.В. Гетманский, Е.О. Мовчан, А.Е. Андреев

УСКОРЕНИЕ РАСЧЕТА ДИНАМИЧЕСКОГО НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ С ПОМОЩЬЮ НАБОРОВ ВЕКТОРНЫХ ИНСТРУКЦИЙ*

Рассмотрен один из способов ускорения расчета динамики систем тел за счет оптимизации под архитектуру процессора с векторными инструкциями. Задача является актуальной в связи с необходимостью проведения большого числа вычислительных экспериментов за ограниченное время. Приведено математическое описание метода и вычислительный алгоритм. Разработан эффективный код для определения динамического напряженно-деформированного состояния тел в сложных механизмах, выполняемый на процессорах, поддерживающих векторные расширения SSE, AVX, FMA и KNC (архитектура Xeon Phi Knights Corner). В расчете использован метод дискретных элементов, являющийся одной из реализаций метода моделирования динамики систем тел со связями. Основными вычислительно сложными компонентами расчета являются расчет правых частей системы с использованием матрично-векторных преобразований в пространстве, расчет углов Эйлера и вычисление матриц поворота, а также численное интегрирование уравнений с помощью явного метода Рунге-Кутты 4-го порядка. Вычислительный алгоритм имеет ограниченную масштабируемость при использовании параллельных вычислений из-за сильной зависимости по данным, в связи с чем оптимизация алгоритма является важным направлением исследования для ускорения расчетов. Для эффективной векторизации вычислений рассмотрен специальный формат хранения данных матриц и векторов в памяти. Для матриц и векторов, превышающих по размерности длину векторных регистров процессора, разработаны варианты операций блочного умножения под каждый набор инструкций. Для матриц и векторов размерностью меньше длины регистров (в случае AVX для одинарной точности представления вещественных чисел и в случае KNC для двойной точности) разработаны микроалгоритмы для упаковки нескольких строк матриц и нескольких векторов в один векторный регистр с перераспределением элементов. Микроалгоритмы реализованы с помощью функций, реализующих векторные инструкции для каждого набора векторных регистров. Достигнуто ускорение расчета до 3-х раз за счет векторизации и проведено сравнение времени расчета, векторизованного с помощью авторского алгоритма, с результатами, полученными путем автовекторизации компилятором.

Автовекторизация; оптимизация кода; векторные регистры; динамика системы тел; SIMD.

* Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов №№ 16-07-00534, 15-01-04577, 15-07-06254, при финансовой поддержке РФФИ и Администрации Волгоградской области в рамках научного проекта № 16-47-340385.

V.V. Getmanskiy, E.O. Movchan, A.E. Andreev

DYNAMICAL STRESS-STRAIN SIMULATION SPEEDUP USING SIMD INSTRUCTIONS

One way to speedup a multibody system dynamics simulation by code optimization for CPU architecture with vector registers is considered. The problem is up to date because of necessity to perform a lot of computations in a short amount of time. Mathematical formulation of problem and computational algorithm are described. The efficient code development for dynamic stress-strain solver for bodies in complex mechanism is proposed. The code runs on processors supporting SIMD operations of SSE, AVX, FMA and KNC (Xeon Phi Knights Corner) instruction set extensions. The discrete elements method is used as one of the implementations of multibody system dynamics simulation. The most computational expensive parts of the code are right-hand side calculation using 3-dimensional matrix-vector transformations, Euler angles and rotation matrices calculation, numerical integration using Runge-Kutta 4-th order method. Computational algorithm has a limited scalability in case of using parallel computing because of strong data dependency between parallel code branches. Therefore the optimization of code is an important study for achieving speedup of computations. Special data format for storing matrices and vectors in memory and efficient vectorization of matrix-vector operations is considered. Block multiplication of matrices and vectors with greater dimension than the vector register length is developed. In case of dimension of matrices and vectors lower than vector register length (single precision floating point data for AVX and double precision for KNC) special microalgorithms for packing several matrix rows and vectors with elements permutation are developed. The microalgorithms are implemented using intrinsic functions for each vector instructions set. Speedup of up to 3 times is achieved using vectorization. The computation time of intrinsic-implemented algorithm is compared with compiler auto-vectorization feature. The microalgorithms are implemented using intrinsic functions for each vector instructions set. Speedup of up to 3 times is achieved using vectorization. The computation time of intrinsic-implemented algorithm is compared with compiler auto-vectorization feature.

Auto-vectorization; code optimization; intrinsic; vector registers; multibody dynamics; SIMD.

Введение. Зачастую в процессе проектирования механической системы или механизма появляется необходимость оценить напряжения и деформации, возникающие в узле под действием внешних сил. Для решения этой задачи можно использовать метод дискретных элементов [1].

Вычислительный эксперимент позволяет принимать конструкторские решения без дорогостоящих натурных испытаний, поэтому в ряде производств, в частности, в процессе проектирования автомобилей специального назначения, его используют многократно на протяжении всего цикла проектирования изделия. Один вычислительный эксперимент с фиксированным набором параметров по определению напряжений выполняется от нескольких часов до нескольких дней. Ввиду изменения в процессе проектирования конструкторской геометрии и необходимости проведения серий вычислительных экспериментов с разными параметрами, ускорение расчета для ускорения всего процесса проектирования в целом является весьма актуальной задачей.

Рассматриваемый метод расчета напряженно-деформированного состояния применяется в системе инженерного анализа для разработки машин (пакет ФРУНД) и предполагает анализ напряженно-деформированного состояния узлов конструкции. Этот анализ возможен только при получении входных данных в виде динамических реакций, которые могут приходить либо с датчиков при проведении натурных испытаний, либо из полной модели машины.

Расчет полной модели машины методом анализа динамики систем тел со связями происходит существенно быстрее, чем анализ динамического напряженно-деформированного состояния. Таким образом, при проведении совместного анали-

за динамики для получения реакций и анализа напряжений в отдельных деталях вычислительно сложным является именно анализ напряжений, следовательно, его необходимо ускорять. Необходимость ускорения процесса расчета моделей машин связана с многократным проведением вычислительного эксперимента при корректировке параметров в зависимости от предыдущих результатов. Корректировкой занимается инженер, поэтому для него важно не тратить большое количество времени на получение очередного результата. Кроме того, некоторые результаты могут требовать модификации конструкторской геометрии, что тоже занимает некоторое время.

Таким образом, ускорение метода расчета динамического напряженно-деформированного состояния отдельных деталей машин, как самого вычислительно сложного, а следовательно, медленного в общем процессе моделирования динамики машины, существенно ускоряет вычислительный эксперимент.

Предлагаемый метод заключается в моделировании динамики системы тел со связями, приближенных дискретными элементами, расположенными в ячейках трехмерной регулярной ортогональной сетки. Каждый элемент представляет собой АТТ (абсолютно твердое тело) заданного размера и массы, имеет 6 степеней свободы и от 1 до 6 связей с соседними элементами. Наличие или отсутствие связей между элементами задает начальное положение и форму тела в пространстве.

Масштабируемость такого алгоритма ограничена – по результатам исследований по распараллеливанию расчета на системах с общей и распределенной памятью (с использованием технологий OpenMP и MPI) [1–4] было достигнуто максимальное ускорение в 10 раз на 24 узлах без сопроцессоров, при дальнейшем увеличении числа узлов увеличение ускорения не наблюдается.

Следующим шагом по ускорению расчета является оптимизация кода. В данной работе для этого используется векторизация [5] как актуальный и эффективный способ повышения производительности вычислений на современных процессорах. Данный подход к решению подобных задач рассмотрен в работах [6–12]. Для рассматриваемой задачи применение векторных инструкций обусловлено также тем, что положение и движение тел в пространстве задано с помощью векторов и матриц, в связи с чем обрабатываемые данные легко соотносятся с векторными регистрами. Таким образом, для данной работы научной новизной является разработка алгоритма вычислений, эффективно использующего векторные регистры и инструкции современных процессоров и ускоряющего вычисления при расчете деформаций и напряжений методом дискретных элементов.

С развитием технологий производительность ядер процессоров традиционной архитектуры повышается благодаря увеличению разрядности векторных регистров, поэтому направление настоящей работы является перспективным для следующего поколения процессоров.

Метод дискретных элементов. В данной работе рассматривается решатель, выполняющий расчет динамического напряженно-деформированного состояния системы тел со связями. Подобные решатели систем многотельной динамики нашли широкое применение при проектировании и инженерном анализе механических конструкций, где они используются, к примеру, для определения оптимальной массы деталей, необходимой жесткости соединительных элементов, наиболее подходящей геометрической формы конструкции в целом.

Динамический анализ позволяет оценить характеристики движения всей механической системы в целом, для его проведения используется метод моделирования динамики системы абсолютно твёрдых тел со связями. В таком подходе в качестве основной модели используется модель динамики системы тел со связями, а вспомогательные модели описывают физические процессы в отдельных телах.

В ходе расчета НДС (напряженно-деформированного состояния) необходимо смоделировать совокупность внутренних напряжений и деформаций, возникающих в теле в результате действия на него некоторых внешних сил, то есть получить тензоры напряжений ($\mathbf{T}_\sigma$) и деформации ($\mathbf{T}_\varepsilon$) в некоторый момент времени [13].

Общее описание модели решения. В методе дискретных элементов для расчета динамического НДС в упругом теле оно представляется как совокупность дискретных элементов кубической формы со стороной, равной шагу пространственной дискретизации. Эти элементы формируют в расчетной области геометрию модели.

Матрица масс в такой постановке задачи является диагональной, что значительно упрощает расчет. Таким образом, движение каждого дискретного элемента описывается обыкновенным дифференциальным уравнением следующего вида:

$$\mathbf{M}\ddot{\mathbf{y}} = \mathbf{q}(\dot{\mathbf{y}}, \mathbf{y}, t) - \mathbf{M}\mathbf{a}(t) + \mathbf{s}(\dot{\mathbf{y}}, \mathbf{y}), \quad (1)$$

где $\mathbf{y}$ – координаты дискретных элементов, $\mathbf{M}$ – матрица инерции, $\mathbf{a}$ – вектор ускорений твердого тела, входящего в полную динамическую модель, $\mathbf{s}(\dot{\mathbf{y}}, \mathbf{y})$ – силы стабилизации.

Функция $\mathbf{q}(\dot{\mathbf{y}}, \mathbf{y}, t)$ описывает силы, возникающие из-за связей между дискретными элементами. Случайные возмущения, передаваемые от динамической модели, могут вызывать смещение дискретных элементов, поэтому для их привязки к опорному телу введены силы стабилизации $\mathbf{s}(\dot{\mathbf{y}}, \mathbf{y})$. Уравнение (1) записано в системе координат опорного тела, поэтому правая часть (1) также содержит силы инерции $\mathbf{Ma}(t)$ тела из динамической модели.

Для граничных узлов в правые части уравнений добавляются силы реакций, возникающие в связях между элементами. Уравнение для граничного элемента с использованием равномерного распределения внешних сил по граничным узлам имеет вид:

$$\mathbf{M}_i\ddot{\mathbf{y}}_i = \mathbf{f}(t)/|F| + \mathbf{q}_i(\dot{\mathbf{y}}, \mathbf{y}, t) - \mathbf{M}_i\mathbf{a}_i(t) + \mathbf{s}_i(\dot{\mathbf{y}}, \mathbf{y}),\ i \in F, \quad (2)$$

где $|F|$ – количество узлов, входящих в границу, к которой приложена сила $\mathbf{f}$.

Уравнение (1) интегрируется явным методом Рунге-Кутта 4 порядка [14]. Таким образом, данные, полученные из динамической модели, позволяют определить силы инерции и силы, возникающие в связях.

Пересчет матриц поворота. В описании модели используется два вида систем координат – глобальная, описывающая всю модель в целом, и локальные, описывающие положение каждого отдельного тела, ориентируясь по центру масс тела и центральным осям инерции. Введем следующие обозначения: глобальную СК обозначим за O_0, локальную СК опорного тела – O_1, дискретного элемента – O_2. Взаимное расположение данных систем координат и тел в них представлено на рис. 1.

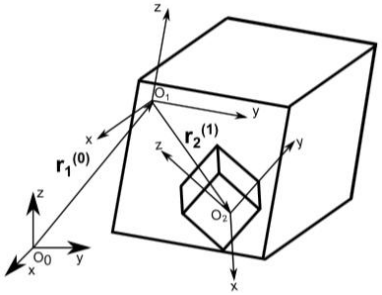


Рис. 1. Взаиморасположение глобальной и локальных систем координат

Так как все данные для расчета хранятся в глобальной системе координат, необходимо ввести преобразование положения тела [15] из локальной СК в глобальную. Для этого используются матрицы поворота – квадратные матрицы, с помощью которых можно определить положение тела, описанного в одной системе координат, в другой СК.

Повороты выполняются на основе углов Эйлера (ψ, θ, и φ) [16], позволяющих однозначно определить положение тела в данной системе координат. Новые значения углов можно получить, численно проинтегрировав кинематические уравнения Эйлера, определяющие проекции угловых скоростей тела с помощью углов Эйлера по следующим формулам:

$$\begin{cases} \dot{\psi} = \frac{\omega_1 \sin\varphi + \omega_2 \cos\varphi}{\sin\theta}, \\ \dot{\varphi} = \omega_3 - (\omega_1 \sin\varphi + \omega_2 \cos\varphi)\mathrm{ctg}\varphi, \\ \dot{\theta} = \omega_1 \cos\varphi - \omega_2 \sin\varphi \end{cases} \quad (3)$$

Эти операции необходимы для того, чтобы выполнить пересчет матриц поворота для каждого дискретного элемента. Данные матрицы имеют следующую форму:

$$A = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\theta \\ \cos\varphi\sin\psi + \sin\varphi\sin\theta\cos\psi & \cos\varphi\cos\psi - \sin\varphi\sin\theta\sin\psi & -\sin\varphi\cos\theta \\ \sin\varphi\sin\psi - \sin\varphi\cos\theta\sin\psi & \sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi & \cos\varphi\cos\theta \end{bmatrix} \quad (4)$$

Расчет деформаций и напряжений. Расчет правых частей уравнений (1) для каждого дискретного элемента осуществляется в несколько шагов. Сначала вычисляется матрица поворота для перевода дискретного элемента в систему координат опорного тела по формуле:

$$\mathbf{A}_{21} = \mathbf{A}_{02}^{\mathrm{T}} \mathbf{A}_{01}, \quad (5)$$

где первая цифра индекса означает номер СК, в которую необходимо перевести положение тела, вторая цифра – текущую СК.

С использованием параметров дискретных элементов и полученных по (4) матриц поворота рассчитываются деформации компонент:

$$\mathbf{d}_{\mathrm{sl}}^{(1)} = \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}\mathbf{C}_2^{(2)} - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{P}_2 - \mathbf{P}_1)^{(0)}, \quad (6)$$

$$\mathbf{d}_{vl}^{(1)} = \mathbf{W}_1^{(1)} \times \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}(\mathbf{W}_2^{(2)} \times \mathbf{C}_2^{(2)}) - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{V}_2 - \mathbf{V}_1)^{(0)}, \quad (7)$$

$$\mathbf{d}_{sa}^{(0)} = \mathbf{R}_1^{(0)} - \mathbf{R}_2^{(0)}, \quad (8)$$

$$\mathbf{d}_{va}^{(0)} = \mathbf{W}_1^{(0)} - \mathbf{W}_2^{(0)}, \quad (9)$$

где $\mathbf{C}_1^{(1)}$ и $\mathbf{C}_2^{(2)}$ – вектора точек связи, записанные в локальной системе координат первого и второго тела (верхний индекс), $\mathbf{P}_1^{(0)}$ и $\mathbf{P}_2^{(0)}$ – координаты центров масс дискретных элементов в глобальной СК, $\mathbf{W}_1^{(1)}$ и $\mathbf{W}_2^{(2)}$ – угловые скорости, $\mathbf{V}_1^{(0)}$ и $\mathbf{V}_2^{(0)}$ – линейные скорости дискретных элементов.

Выражение (6) задает линейный сдвиг дискретного элемента, (7) – изменение линейной скорости, (8) – угловой сдвиг и (9) – изменение угловой скорости. На основе данных выражений вычисляются итоговые значения деформаций, с помощью которых обновляются данные о текущем состоянии модели – пересчет линейных и угловых компонент, обновление сил и моментов и др. По обновленным параметрам выполняется очередной шаг интегрирования явным методом Рунге-Кутты 4-го порядка точности.

Векторизация решателя динамического НДС. Из формул (1) и (6–9) видно, что основные вычисления составляют матрично-векторные операции. В данном случае удобно применить векторизацию к расчету правых частей и численному интегрированию, так как эти операции выполняются независимо над разными наборами данных.

Вычисления преобразований координат производятся с помощью матриц поворота (4), размерность которых 3x3, а не полных матриц преобразования в трехмерном пространстве размерностью 4x4 из-за особенностей вычисления неизвестных в механической системе. Для каждого дискретного элемента процедуры вычисления поступательных и вращательных степеней свободы принципиально отличаются, сходство имеется только в процедуре численного интегрирования. При этом каждая из поступательных степеней свободы вычисляется одинаково так же, как и каждая из вращательных.

В процедуре численного интегрирования операции вычисления сумм и произведений производятся над полными векторами неизвестных, состоящих из всех степеней свободы дискретного элемента. При этом вычисленные вектора неизвестных скоростей и перемещений используются на следующей итерации для пространственных преобразований, поэтому целесообразно хранить вектора сгруппированными по координатам, то есть тройками (x, y, z) для поступательных степеней свободы и (rx, ry, rz) – для вращательных.

Векторные регистры процессора имеют разрядность, кратную степени двойки, поэтому, на первый взгляд, можно было бы эффективно векторизовать операции с полной матрицей пространственных преобразований в однородных координатах, но ввиду отмеченных выше причин для всей процедуры вычислений и минимизации операций выделения и копирования памяти использован способ хранения векторов последовательно по координатам с пропусками в 1 элемент для кратности вектора разрядности используемого регистра.

Описание способа организации памяти. В данном методе важную роль играет организация памяти. Практически каждая операция расчета включает загрузку данных из памяти и выгрузку в нее результатов. Поэтому работа с памятью должна быть максимально эффективной.

По умолчанию используется невыровненное выделение памяти, при котором блоки данных не умещаются в границах машинных слов или сдвинуты относительно этих границ. Работа с невыровненными данными на многих операциях осуществляется существенно дольше, чем с выровненными, а для некоторых систем и команд она недопустима [17]. В частности, при использовании векторных инструкций существуют как команды для выровненной загрузки данных в регистры, так и для невыровненной (за исключением набора инструкций KNC), но при этом невыровненная загрузка примерно в 3 раза дольше выровненной. В связи с этим, первым шагом для оптимизации кода является выровненное выделение памяти под массивы данных, для чего используются команды вида:

```
_var = (double*)_aligned_malloc(varSize, 32);
```

С использованием векторных регистров связана также проблема размерности матриц, связанная с необходимостью загрузки в регистр на 2, 4 или 8 вещественных элементов двойной точности данных, сгруппированных по три элемента. Для эффективной загрузки в регистр размерность матриц и векторов должна быть 3x4 и 1x4 соответственно, чтобы загружать их по строкам.

Этого можно достичь, если дополнить каждую строку фиктивным четвертым элементом, равным 0, который фактически не участвует в операциях и не влияет на результат.

После этого строка матрицы либо целиком помещается в векторный регистр (для AVX), либо занимает два полных регистра меньшего размера (для SSE), можно также загружать в регистр две строки одновременно (для KNC).

Таким образом, хранение векторов для координат n дискретных элементов в выровненной памяти имеет вид

$$(x_0, y_0, z_0, 0, rx_0, ry_0, rz_0, 0, x_1, y_1, z_1, 0, \dots, x_n, y_n, z_n, 0, rx_n, ry_n, rz_n),$$

а матриц поворота, соответственно:

$$(a_{00}, a_{01}, a_{02}, 0, a_{10}, a_{11}, a_{12}, 0, a_{20}, a_{21}, a_{22}, 0, \dots),$$

где a_{ij} – элемент матрицы поворота дискретного элемента.

Анализ эффективности автовекторизации при оптимизации решателя. Современные компиляторы используют алгоритмы анализа кода, позволяющие проводить оптимизацию на этапе трансляции в машинные команды. Основная операция оптимизации, дающая существенное ускорение – это автовекторизация [18].

В табл. 1 представлены данные о времени выполнения расчета над одной и той же моделью на разных процессорах. Для сравнения использованы встроенный компилятор Microsoft C++ Compiler версии 18.0 (Visual Studio 2013) и компилятор фирмы Intel C++ версии 15.0 с ключами оптимизации по скорости (-O2) и полной оптимизации (-O3) и включенной поддержкой векторных команд.

Таблица 1

Результаты выполнения расчета с разными вариантами автовекторизации

Процессор	Уровень оптимизации	Без автовекторизации	SSE	AVX	AVX2
Intel Core i7 CPU 3537U	Без оптимизации	11.741 с	11.834 с	11.830 с	11.466 с
	По скорости (-O2)	7.135 с	7.060 с	7.341 с	7.074 с
	Полная (-O3)	8.410 с	7.555 с	7.329 с	7.288 с
Intel Xeon CPU E5-2650 v3	Без оптимизации	17.548 с	17.217 с	14.001 с	12.561 с
	По скорости (-O2)	12.825 с	12.599 с	9.261 с	8.913 с
	Полная (-O3)	12.729 с	12.472 с	9.199 с	8.876 с

Таким образом, максимальное ускорение, полученное с помощью автовекторизации, для данного решателя составляет около 1,3 раза при компиляции с полной оптимизацией и поддержкой расширения набора команд AVX2. Это значение может меняться в зависимости от используемого компилятора.

Описание векторизованного алгоритма численного интегрирования. Алгоритм численного интегрирования с помощью явного метода Рунге-Кутты 4-го порядка точности предполагает вычисление следующего приближения в 4 шага. На каждом шаге вычисляются коэффициенты, которые используются на следующих шагах, что требует обновления данных в памяти после каждого этапа.

Рассмотрим векторизацию данного метода на примере одного из шагов вычислений. Невекторизованный фрагмент программы, рассчитывающей 3-й шаг, выглядит так:

```
for (int j = 0; j < _nVariables; j++) {
        _hDDX3[j] = _varDDX[j] * _timeStep;
        _varX[j] = _initX[j] + (_initDX[j] + _hDDX2[j] * 0.5) *
_timeStep;
        _varDX[j] = _initDX[j] + _hDDX3[j];
```

Этот же фрагмент с помощью AVX-инструкций можно записать так:

```
for (int j = 0; j < _nVariables; j += 4) {
        Xtmp = _mm256_load_pd(_initX + j);
        DXtmp = _mm256_load_pd(_initDX + j);
        DDXtmp = _mm256_load_pd(_varDDX + j);

        hDDX3 = _mm256_mul_pd(DDXtmp, timeStep);
        _mm256_store_pd(_hDDX3 + j, hDDX3);
        tmp = _mm256_fmadd_pd(hDDX2, constant, DXtmp);
        tmp = _mm256_fmadd_pd(tmp, timeStep, Xtmp);
        _mm256_store_pd(_varDX + j, _mm256_add_pd(DXtmp, hDDX3));
```

Для других регистров меняются только названия инструкций и шаг цикла, так, для KNC:

```
    for (int j = 0; j < _nVariables; j += 8){
        Xtmp = _mm512_load_pd(_initX + j); //. . .
```

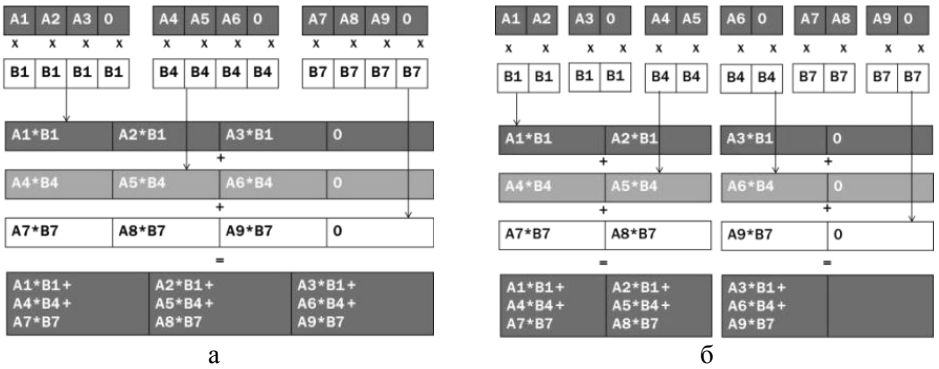
Здесь также использованы инструкции FMA [19], выполняющие совмещенное сложение-умножение за одну машинную команду. В идеале ускорение данной функции должно быть равно размеру использованных векторных регистров – 8 раз для KNC, 4 раза для AVX и 2 для SSE. Но на практике из-за большого числа операций загрузки и выгрузки данных достигаемый эффект не так значителен – 2,5, 2,1 и 1,7 раза, соответственно, на KNC, AVX и SSE с помощью FMA-инструкций.

Описание векторизованного алгоритма вычисления правых частей. Наибольшие временные затраты при расчете НДС вызывает вычисление правых частей. Все операции в данной функции являются матричными или векторными по выровненной схеме, описанной выше.

Рассмотрим векторизацию матричных операций на примере транспонированного перемножения матриц:

```
Mat3x4 matA21 = matA02.Tmul(matA01);
```

Для векторизации с помощью векторных инструкций применяется схема, представленная на рис. 2. На схеме приведен пример вычисления 1-й строки результирующей матрицы для SSE и AVX-инструкций.



а б

Рис. 2. Схема векторизованного перемножения матриц: а – с помощью регистров AVX; б – с помощью регистров SSE

Для данной функции максимальное ускорение ограничивается размерностью матриц – 3 раза, и это ускорение обуславливает общее ускорение программы, так как операция является самой трудоемкой. В случае SSE-инструкций используется два регистра на каждый вектор, поэтому ожидаемое ускорение – около 2 раз, однако за счет более эффективной работы с памятью был получен результат в примерно 2,9 раза. Полученное ускорение за счет использования регистров и векторных инструкций AVX – примерно 3,3 раза. Для регистров KNC при условии одновременной обработки двух матриц/векторов ожидается ускорение порядка 4–4,5 раз.

Результаты тестирования векторизованной версии полного расчета динамического НДС представлены в табл. 2.

Таблица 2

Ускорение выполнения расчета с разными вариантами векторизации

Размерность	Автовекторизация	SSE	AVX	AVX2+FMA
50x50x50	1,30230827	2,41034483	2,36737589	2,94864865
60x60x60	1,30606992	2,44652406	2,37803092	2,8045977
70x70x70	1,24943246	2,5406809	2,43049645	2,96465028
80x80x80	1,19765739	2,48066717	2,39676386	2,85617792
90x90x90	1,32642016	2,50839014	2,51498423	2,91983511
100x100x100	1,26203184	2,5763678	2,35296192	3,04633645
110x110x110	1,39702253	2,43249238	2,415591	2,94208315
120x120x120	1,30596896	2,36457357	2,41409739	3,05859375
130x130x130	1,30607769	2,40815483	2,41968989	3,11012604
140x140x140	1,3709803	2,39247239	2,37123791	2,8801424
150x150x150	1,34033363	2,54172619	2,29560906	3,08522767

Следует отметить, что в настоящее время использование SSE-инструкций не дает существенного результата. Это связано с тем, что большинство современных процессоров поддерживает более новые расширения AVX, KNC и другие, а регистры SSE эмулируются через регистры большего размера, что ведет к потерям производительности.

Обработка матриц с помощью KNC-инструкций. При имеющейся структуре хранения данных без изменения алгоритма при выполнении матричных операций используются только 4 элемента двойной точности для каждой строки матрицы или вектора. Как известно, регистры KNC способны вместить 8 чисел типа double, а это значит, что половина каждого регистра в ходе расчета не используется, что существенно снижает производительность программы. В связи с этим возникает необходимость модификации алгоритма перемножения матриц для обработки одновременно нескольких пар матриц и векторов, например, как представлено на рис. 3.

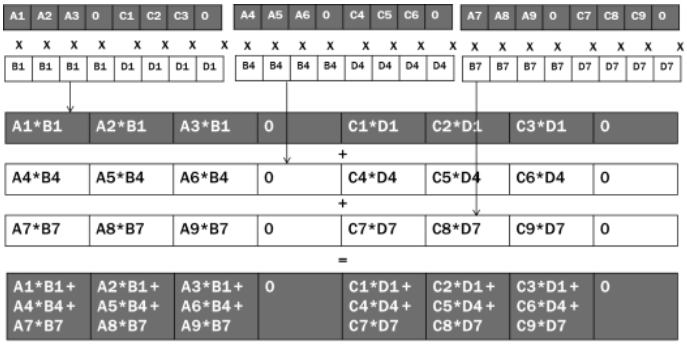


Рис. 3. Примерная схема векторизованного перемножения матриц с помощью регистров KNC

Один из подходов к решению данной проблемы предложен в работе [20], где желаемое содержимое регистров получается путем последовательных операций размножения и смешивания элементов внутри векторного регистра:

```
zmm08=_mm512_extload_ps(vec+ 0, _MM_UPCONV_PS_NONE,
_MM_BROADCAST_1X16, _MM_HINT_NONE);
zmm09=_mm512_extload_ps(vec+ 4, _MM_UPCONV_PS_NONE,
_MM_BROADCAST_1X16, _MM_HINT_NONE);
zmm10=_mm512_extload_ps(vec+ 8, _MM_UPCONV_PS_NONE,
_MM_BROADCAST_1X16, _MM_HINT_NONE);
zmm11=_mm512_extload_ps(vec+ 12, _MM_UPCONV_PS_NONE,
_MM_BROADCAST_1X16, _MM_HINT_NONE);


zmm06 = _mm512_mask_blend_ps(mask32_0, zmm08, zmm09);
zmm07 = _mm512_mask_blend_ps(mask32_1, zmm10, zmm11);
zmm04 = _mm512_mask_blend_ps(mask32_2, zmm06, zmm07);
```

В [20] рассмотрен также еще один способ организации пересылок данных между регистрами, поддерживаемый векторным расширением KNC, с помощью команды переставления и размножения элементов в регистре swizzle:

```
zmm04 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_AAAA);
zmm05 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_BBBB);
zmm06 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_CCCC);
zmm07 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_DDDD);
```

Очевидно, в данном случае второй вариант является более предпочтительным, так как он позволяет получить желаемый результат за меньшее число шагов и, следовательно, получить больший выигрыш по времени. Поэтому следующим этапом векторизации решателя будет доработка KNC-версии программы с использованием рассмотренной схемы перемножения нескольких пар матриц одновременно.

Выводы и перспективы дальнейших исследований. Разработаны векторизованные версии алгоритма расчета напряженно-деформированного состояния, использующие векторные регистры разного размера (128, 256, 512 бит) и соответствующие им базовые операции.

Векторизация кода решателя позволила ускорить вычисления:

- в среднем в 1,3 раза с помощью автовекторизации;
- в среднем в 2,4 раза с использованием команд SSE;

- в среднем в 2,8 раз с использованием команд AVX;
- до 3 раз за счет использования FMA-инструкций.

Дальнейшая оптимизация алгоритмов для высокопроизводительных кластеров возможна за счет векторизации под новые архитектуры, например, поддерживающие AVX-512, и переноса кода пространственных преобразований для определения положения тел на графические вычислители и сопроцессоры.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Гетманский В.В., Горобцов А.С., Измайлов Т.Д.* Распараллеливание расчёта напряжённо-деформированного состояния тела в многотельной модели методом декомпозиции расчётной области // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". – 2013. – Вып. 16, № 8 (111). – С. 5-10.
2. *Getmanskiy V.V., Gorobtsov A.S., Sergeev E.S., Ismailov T.D., Shapovalov O.V.* Concurrent simulation of multibody systems coupled with stress-strain and heat transfer solvers // Journal of Computational Science. – 2012. – No. 3 (6). – P. 492-497.
3. *Сергеев Е.С., Гетманский В.В., Горобцов А.С.* Перенос системы многотельной динамики на вычислительный кластер // Научно-технические ведомости Санкт-Петербургского гос. политехн. ун-та. – 2010. – Вып. 101. – С. 93-99.
4. *Горобцов А.С., Гетманский В.В., Андреев А.Е., Doan D.T.* Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings: ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015. – P. 379-391.
5. *Andreev A., Nasonov A., Novokschenov A., Bochkarev A., Kharkov E., Zharikov D., Kharchenko S., Yuschenko A.* Vectorization algorithms of block linear algebra operations using SIMD instructions // Communications in Computer and Information Science. – 2014. – Vol. 535. – P. 323-340.
6. *Mulansky M.* Optimizing Large-Scale ODE Simulations // SIAM Journal of Scientific Computing. – 2014. – 18 p.
7. *Bialas P., Kowal J., Strzelecky A.* GPU-accelerated and CPU SIMD optimized Monte Carlo simulation // Computing and Informatics. – 2014. – Vol. 33. – P. 1191-1208.
8. *Kral S. Franchetti F., Lorenz J. and Ueberhuber C.W.* SIMD Vectorization of Straight Line FFT Code. – P. 251-260.
9. *Jeong H., Kim S., Lee W. and Myung S.-H.* Performance of SSE and AVX Instruction Sets // The 30th International Symposium on Lattice Field Theory (June 24 – 29, 2012 Cairns, Australia): Proceedings. – P. 249-258.
10. *Заранек С.В., Чоу Б., Шарма Г., Зарринкуб Х.* Ускорение алгоритмов и приложений MATLAB. – URL: http://matlab.ru/articles/Matlab_acceleration.pdf. (дата обращения: 22.09.2016).
11. *Лемешевский С.В.* Численные методы решения уравнений в частных производных. – URL: http://slemeshevsky.github.io/python-num-pde/term2/build/latex/FDMforPDE.pdf. (дата обращения: 12.08.2016).
12. *Алексеев В.А., Головашкин Д.Л.* Векторизация метода распространяющегося пучка и его реализация по технологии CUDA // Компьютерная оптика. – 2010. – Т. 34, № 2. – С. 225-230.
13. *Иванов К.А. [и др.].* Прикладная теория пластичности. – М.: Политехника, 2009. – 376 с.
14. *Горелов Ю.Н.* Численные методы решения обыкновенных дифференциальных уравнений (метод Рунге – Кутта): учеб. пособие. – Самара: Изд-во «Самарский университет», 2006. – 48 с.
15. *Яглом И.М.* Геометрические преобразования. Ч. 1. Движения и преобразования подобия. – М.: Государственное издательство технико-теоретической литературы, 1956. – 280 с.
16. *Кузнецов Е.Б.* Об одном подходе к интегрированию кинематических уравнений Эйлера // Вычислительная математика и математическая физика. – 1998. – Т. 38, № 11. – С. 1806-1813.
17. *Конищев Д.* Что такое выравнивание и как оно влияет на работу ваших программ. – URL: http://webcache.googleusercontent.com/search?q=cache:rb4kHpT7ansJ:konishchevdmitry.blogspot.com/2010/01/blog-post.html+&cd=1&hl=en&ct=clnk&gl=ru. (дата обращения: 12.08.2016)..

18. *Ермолицкий А.Е.* Методы повышения эффективности векторизации в оптимизирующем компиляторе // Вопросы радиоэлектроники. Сер. ЭВТ. – 2010. – Вып. 3. – С. 41-50.
19. *Muller J.-M., Brisebarre N.* The Fused Multiply-Add Instructions // Handbook of Floating-point Arithmetic. – 2009. – P. 151-179.
20. *Zumbusch G.* Vectorized Higher Order Finite Difference Kernels // State-of-the-Art in Scientific and Parallel Computing (PARA). – 2012. – P. 343-357.

REFERENCES

1. *Getmanskiy V.V., Gorobtsov A.S., Izmaylov T.D.* Rasparallelivanie rascheta napryazhenno-deformirovannogo sostoyaniya tela v mnogotel'noy modeli metodom dekompozitsii raschetnoy oblasti [Parallelization of the calculation of the stress-strain state of bodies in a multibody model decomposition method the computational region], *Izvestiya VolgGTU. Seriya "Aktual'nye problemy upravleniya, vychislitel'noy tekhniki i informatiki v tekhnicheskikh sistemakh"* [Izvestia Volgograd State Technical University. Series Actual Problems of Management, Computing Hardware and Informatics in Engineering Systems"], 2013, Issue 16, No. 8 (111), pp. 5-10.
2. *Getmanskiy V.V., Gorobtsov A.S., Sergeev E.S., Ismailov T.D., Shapovalov O.V.* Concurrent simulation of multibody systems coupled with stress-strain and heat transfer solvers, *Journal of Computational Science*, 2012, No. 3 (6), pp. 492-497.
3. *Sergeev E.S., Getmanskiy V.V., Gorobtsov A.S.* Perenos sistemy mnogotel'noy dinamiki na vychislitel'nyy klaster [The transfer of multibody system dynamics on the compute cluster], *Nauchno-tekhnicheskie vedomosti Sankt-Peterburgskogo gos. politekhn. un-ta* [Scientific-technical Bulletin of Saint-Petersburg, gosudarstvenno Polytechnic University], 2010, Issue 101, pp. 93-99.
4. *Gorobtsov A.S., Getmanskiy V.V., Andreev A.E., Doan D.T.* Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach, *Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings: ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015*, pp. 379-391.
5. *Andreev A., Nasonov A., Novokschenov A., Bochkarev A., Kharkov E., Zharikov D., Kharchenko S., Yuschenko A.* Vectorization algorithms of block linear algebra operations using SIMD instructions, *Communications in Computer and Information Science*, 2014, Vol. 535, pp. 323-340.
6. *Mulansky M.* Optimizing Large-Scale ODE Simulations, *SIAM Journal of Scientific Computing*, 2014, 18 p.
7. *Bialas P., Kowal J., Strzelecky A.* GPU-accelerated and CPU SIMD optimized Monte Carlo simulation, *Computing and Informatics*, 2014, Vol. 33, pp. 1191-1208.
8. *Kral S. Franchetti F., Lorenz J. and Ueberhuber C.W.* SIMD Vectorization of Straight Line FFT Code, pp. 251-260.
9. *Jeong H., Kim S., Lee W. and Myung S.-H.* Performance of SSE and AVX Instruction Sets, // *The 30th International Symposium on Lattice Field Theory (June 24 – 29, 2012 Cairns, Australia): Proceedings*, 2012, pp. 249-258.
10. *Zaranek S.V., Chou B., Sharma G., Zarrinkub Kh.* Uskorenie algoritmov i prilozheniy MATLAB [Acceleration of algorithms and MATLAB applications]. Available at: http://matlab.ru/articles/Matlab_acceleration.pdf. (accessed 22 September2016).
11. *Lemeshevskiy S.V.* Chislennye metody resheniya uravneniy v chastnykh proizvodnykh [Numerical methods for solving partial differential equations]. Available at: http://slemeshevsky.github.io/python-num-pde/term2/build/latex/FDMforPDE.pdf. (accessed 12 August 2016).
12. *Alekseev V.A., Golovashkin D.L.* Vektorizatsiya metoda rasprostranyayushchegosya puchka i ego realizatsiya po tekhnologii CUDA [Vectorization method the beam propagation and its implementation on CUDA technology], *Komp'yuternaya optika* [Computer optics], 2010, Vol. 34, No. 2, pp. 225-230.
13. *Ivanov K.A. [i dr.]*. Prikladnaya teoriya plastichnosti [Applied theory of plasticity]. Moscow: Politekhnika, 2009, 376 p.
14. *Gorelov Yu.N.* Chislennye metody resheniya obyknovennykh differentsial'nykh uravneniy (metod Runge – Kutta): ucheb. posobie [Numerical methods for solving ordinary differential equations (Runge – Kutta): textbook]. Samara: Izd-vo «Samarskiy universitet», 2006, 48 p.

15. *Yaglom I.M.* Geometricheskie preobrazovaniya. Ch. 1. Dvizheniya i preobrazovaniya podobiya [The geometric transformation. Part 1. Motions and similarity transformations]. Moscow: Gosudarstvennoe izdatel'stvo tekhniko-teoreticheskoy literatury, 1956, 280 p.
16. *Kuznetsov E.B.* Ob odnom podkhode k integrirovaniyu kinematicheskikh uravneniy Eylera [About one approach to integrating the kinematic Euler equations], *Vychislitel'naya matematika i matematicheskaya fizika* [Computational mathematics and mathematical physics], 1998, Vol. 38, No. 11, pp. 1806-1813.
17. *Konishchev D.* Chto takoe vyravnivanie i kak ono vliyaet na rabotu vashikh program [What is alignment and how it affects your programs]. Available at: http://webcache.googleusercontent.com/search?q=cache:rb4kHpT7ansJ:konishchevdmitry.blogspot.com/2010/01/blog-post.html+&cd=1&hl=en&ct=clnk&gl=ru. (accessed 12 August 2016).
18. *Ermolitskiy A.E.* Metody povysheniya effektivnosti vektorizatsii v optimiziruyushchem kompilyatore [Methods of increasing the efficiency of vectorization in an optimizing compiler], *Voprosy radioelektroniki. Ser. EVT* [Questions of radio electronics. Series of EVT], 2010, Issue 3, pp. 41-50.
19. *Muller J.-M., Brisebarre N*. The Fused Multiply-Add Instructions, *Handbook of Floating-point Arithmetic*, 2009, pp. 151-179.
20. *Zumbusch G.* Vectorized Higher Order Finite Difference Kernels, State-*of-the-Art in Scientific and Parallel Computing (PARA)*, 2012, pp. 343-357.

Статью рекомендовал к опубликованию д.т.н., профессор И.И. Левин.

Гетманский Виктор Викторович – Волгоградский государственный технический университет; e-mail: victor.getmanski@gmail.com; 400005, г. Волгоград, пр. Ленина, 28; тел.: +79275065804; кафедра ЭВМ и систем; с.н.с.; к.т.н.

Мовчан Евгения Олеговна – e-mail: verborum123@mail.ru; тел.: +79197980049; бакалавр; студент.

Андреев Андрей Евгеньевич – e-mail: andan2005@yandex.ru; тел.: +79023628177; кафедра ЭВМ и систем; и.о. зав. кафедрой; к.т.н.

Getmanskiy Victor Victorovich – Volgograd State Technical University; e-mail: victor.getmanski@gmail.com; 28, Lenin av., Volgograd, 400005, Russia; phone: +79275065804; the department of computers and systems; senior researcher; cand. of eng. sc.

Movchan Evgenija Olegovna – e-mail: verborum123@mail.ru; phone: +79197980049; bachelor; student.

Andreev Andrey Evgenevich – e-mail: andan2005@yandex.ru; phone: +79023628177; the department of Computers and systems; head of department; cand. of eng. sc.

УДК 004.4 DOI 10.18522/2311-3103-2016-11-3954

А.И. Дордопуло, И.И. Левин, И.А. Каляев, В.А. Гудков, А.А. Гуленок

ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ГИБРИДНОГО ТИПА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ COLAMO*

Рассматриваются методы программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы. В качестве основы технологии программирования вычислительных систем гибридного типа предлагается язык программирования высокого уровня COLAMO с расширениями, с помощью которых можно описывать различные виды параллельных вычислений – структурную,

* Работа выполнена при частичной финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии № 14.578.21.0006 от 05.06.2014, уникальный идентификатор RFMEFI57814X0006.