КОНЕЧНО-ЭЛЕМЕНТНЫЕ ТЕХНОЛОГИИ ДЛЯ КЛАСТЕРОВ ГИБРИДНОЙ АРХИТЕКТУРЫ [1]

А.К. Новиков, С.П. Копысов, И.М. Кузьмин, Н.С. Недожогин

Институт механики УрО РАН, г. Ижевск

Международная научная конференция
Параллельные вычислительные технологии (ПаВТ'2015)
г. Екатеринбург, 30 марта – 2 апреля 2015 г.

[1] Работа выполнена при поддержке РФФИ (проекты 13-01-00101-а, 14-01-00055-а, 14-01-31066-мол_а)

Содержание

Распределение вычислений между GPU и CPU

Схема со сборкой глобальной матрицы жесткости.

- Интегрирование локальных матриц жесткости $K^e, e = \overline{1, m/n_p}$ на GPU
- Передача $K^e, e = \overline{1, m/n_p}$ на CPU.
- Сборка глобальной матрицы жесткости $K = \sum_{e=1}^{m} C_e^T K^e C_e$ на CPU.
- Задание граничных условий Дирихле на CPU.
- Решение системы $Ku = f$ на нескольких GPU.

Распределение вычислений между GPU и CPU

Схема со сборкой глобальной матрицы жесткости.

- Интегрирование локальных матриц жесткости $K^e, e = \overline{1, m/n_p}$ на GPU
- Передача $K^e, e = \overline{1, m/n_p}$ на CPU.
- Сборка глобальной матрицы жесткости $K = \sum_{e=1}^{m} C_e^T K^e C_e$ на CPU.
- Задание граничных условий Дирихле на CPU.
- Решение системы $Ku = f$ на нескольких GPU.

Схема по конечным элементам в итерационных методах подпространств Крылова.

- Интегрирование локальных матриц жесткости $K^e, e = \overline{1, m/n_p}$ на GPU
- Задание граничных условий Дирихле на уровне конечных элементов на GPU.
- Поэлементное решение системы $Ku = f$ на нескольких GPU.

Распределение вычислений между GPU и CPU

Схема со сборкой глобальной матрицы жесткости.

- Интегрирование локальных матриц жесткости $K^e, e = \overline{1, m/n_p}$ на GPU
- Передача $K^e, e = \overline{1, m/n_p}$ на CPU.
- Сборка глобальной матрицы жесткости $K = \sum_{e=1}^{m} C_e^T K^e C_e$ на CPU.
- Задание граничных условий Дирихле на CPU.
- Решение системы $Ku = f$ на нескольких GPU.

Схема по конечным элементам в итерационных методах подпространств Крылова.

- Интегрирование локальных матриц жесткости $K^e, e = \overline{1, m/n_p}$ на GPU
- Задание граничных условий Дирихле на уровне конечных элементов на GPU.
- Поэлементное решение системы $Ku = f$ на нескольких GPU.

Отображение на программные технологии

- Подмножеству из m/n_p конечных элементов ставится в соответствие один из n_p MPI процессов.
- В каждом процессе MPI создается $n_{GPU} + 1$ нить OpenMP, здесь n_{GPU} — число ускорителей в вычислительном узле.
- Мастер-нить OpenMP обеспечивает MPI коммуникации.
- Нити с номерами $1, ..., n_{GPU}$ передают данные между CPU и GPU внутри узла, и запускают kernel-функций CUDA.

Вычисление локальных матриц жесткости на графическом ускорителе

Уровни распараллеливания

1: В CUDA-нити $th \in [1, m]$ выполнить:
{I уровень}
2: **for** $i = 1$ to n_i **do**
3: **for** $j = 1$ to n_j **do**
4: **for** $k = 1$ to n_k **do**
5:
В CUDA-нити $th \in [1, m \cdot n_G]$ выполнить:
{II уровень}
6: Вычисление производных ψ_e по r, s, t.
7: Вычисление J, $detJ$ и J^{-1}.
8: Вычисление матрицы B^e.
9: **for** $\alpha = 1$ to N_e **do**
10: **for** $\beta = \alpha$ to N_e **do**
11: $\hat{K}^e_{\alpha\beta} = B^{e\,T}_{\alpha} D^e B^e_{\beta} \times$
$\times w_i w_j w_k \det J.$
12: $K^e_{\alpha\beta} \leftarrow K^e_{\alpha\beta} + \hat{K}^e_{\alpha\beta}.$
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17: **end for**

Особенности алгоритмов

AI	AII
$th \leftarrow K^e$	$th \leftarrow K^e_{\alpha\beta}(r_i, s_i, t_i)$
$N_{th} = m$	$N_{th} = m \cdot n_G$
$N_{b_x} = \frac{N_{th}}{N_{tib}} + 1$	$N_{b_x} = maxGridSize[1]$
$N_{b_y} = 1$	$N_{b_y} = \frac{N_{th}/N_{tib}}{N_{b_x}-1} + 1$

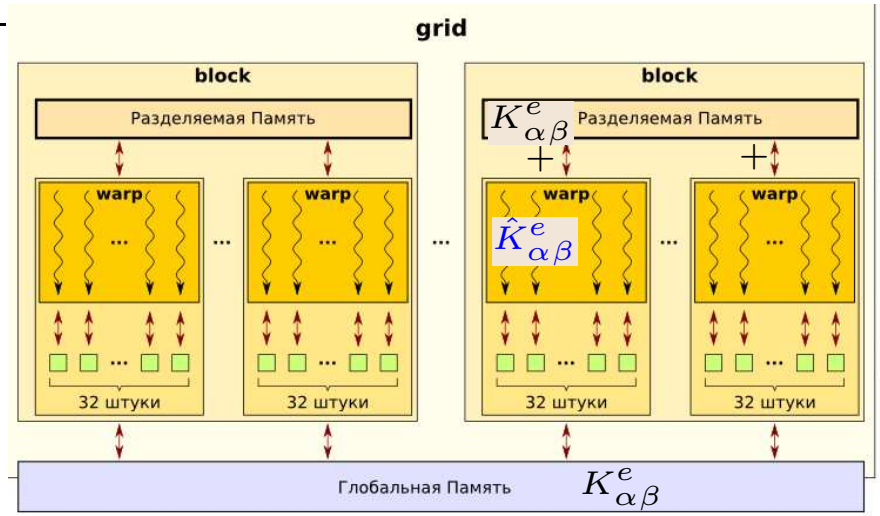

Рис. 1: Суммирование $K^e_{\alpha\beta}$ в AII

Таблица 1: Характеристика иерархических элементов 1–5-го порядка

Порядок к.э., p	Число неизвестных в к.э., N_e	Размер матрицы K_e, $N_e(N_e+1)/2$	FP64, Kb
1	24	300	>2
2	60	1830	>14
3	96	4656	>37
4	150	11325	>90
5	222	24753	>198

Таблица 2: Время вычисления одним ядром CPU (сек.) и ускорение на GPU

Шестигранник, $m = 10000$, $(N_e \times N_e)$	t_{CPU} E5430/E5-2609		$\frac{t_{CPU}}{t_{GPU}}$ GTX580/GTX680	
	-00	-03	AI	AII
$p = 1, n_G = 8, (24 \times 24)$	1.23 /1.12	0.17/0.14	0.41/0.74	0.39/0.67
$p = 2, n_G = 8, (60 \times 60)$	5.73 /5.20	0.74/0.58	1.09/1.08	0.90/1.05
$p = 3, n_G = 64, (96 \times 96)$	108.3 /97.45	13.30/10.52	2.68/1.48	7.07/6.41
$p = 4, n_G = 125, (150 \times 150)$	491.1/441.04	59.02/47.52	2.69/1.41	8.41/6.75
$p = 5, n_G = 343, (222 \times 222)$	2867.6/2598.90	337.83/293.00	2.81/1.50	5.82/5.01

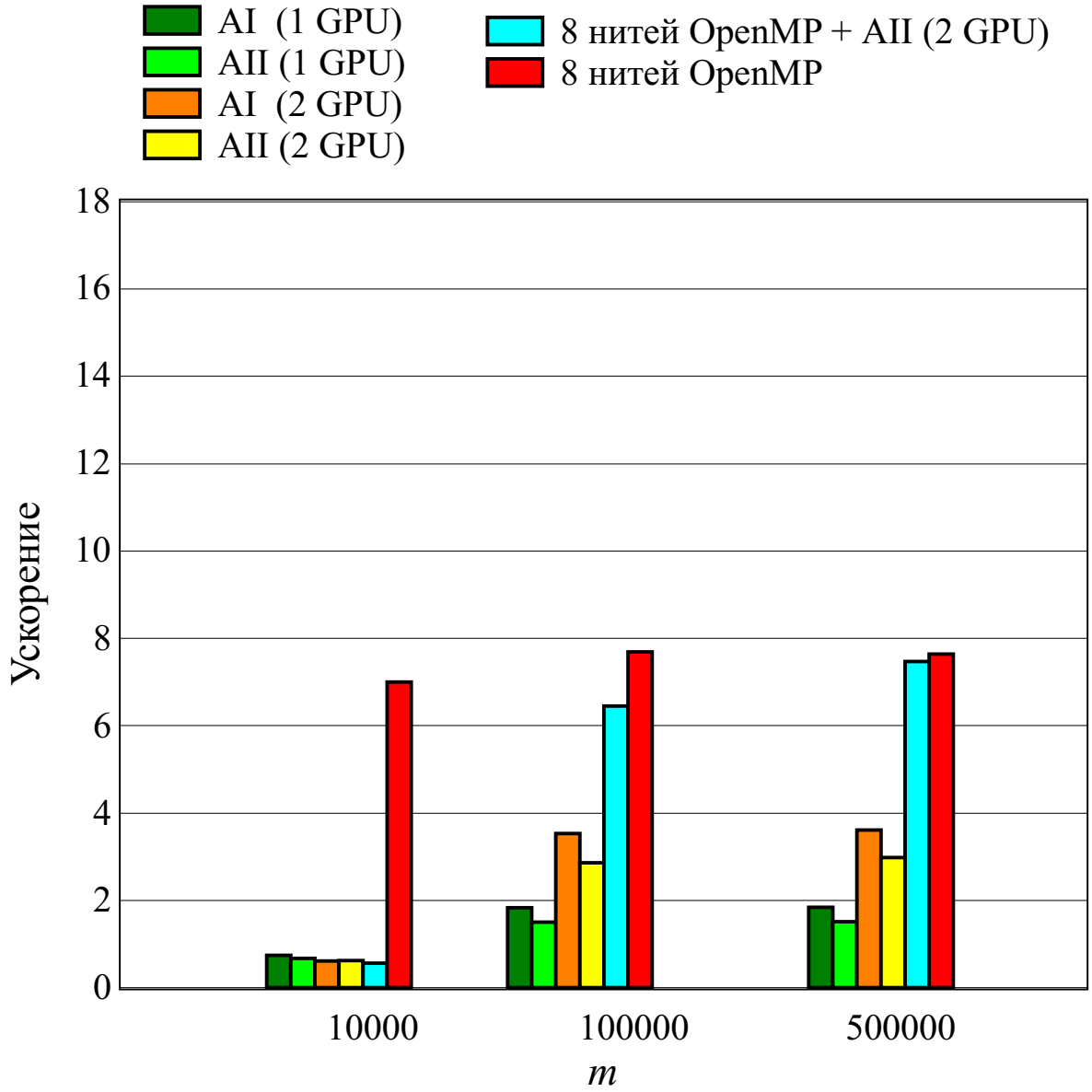

Рис. 2: Ускорение при вычислении матриц $K^e, e = \overline{1, m}$, $p = 1$.

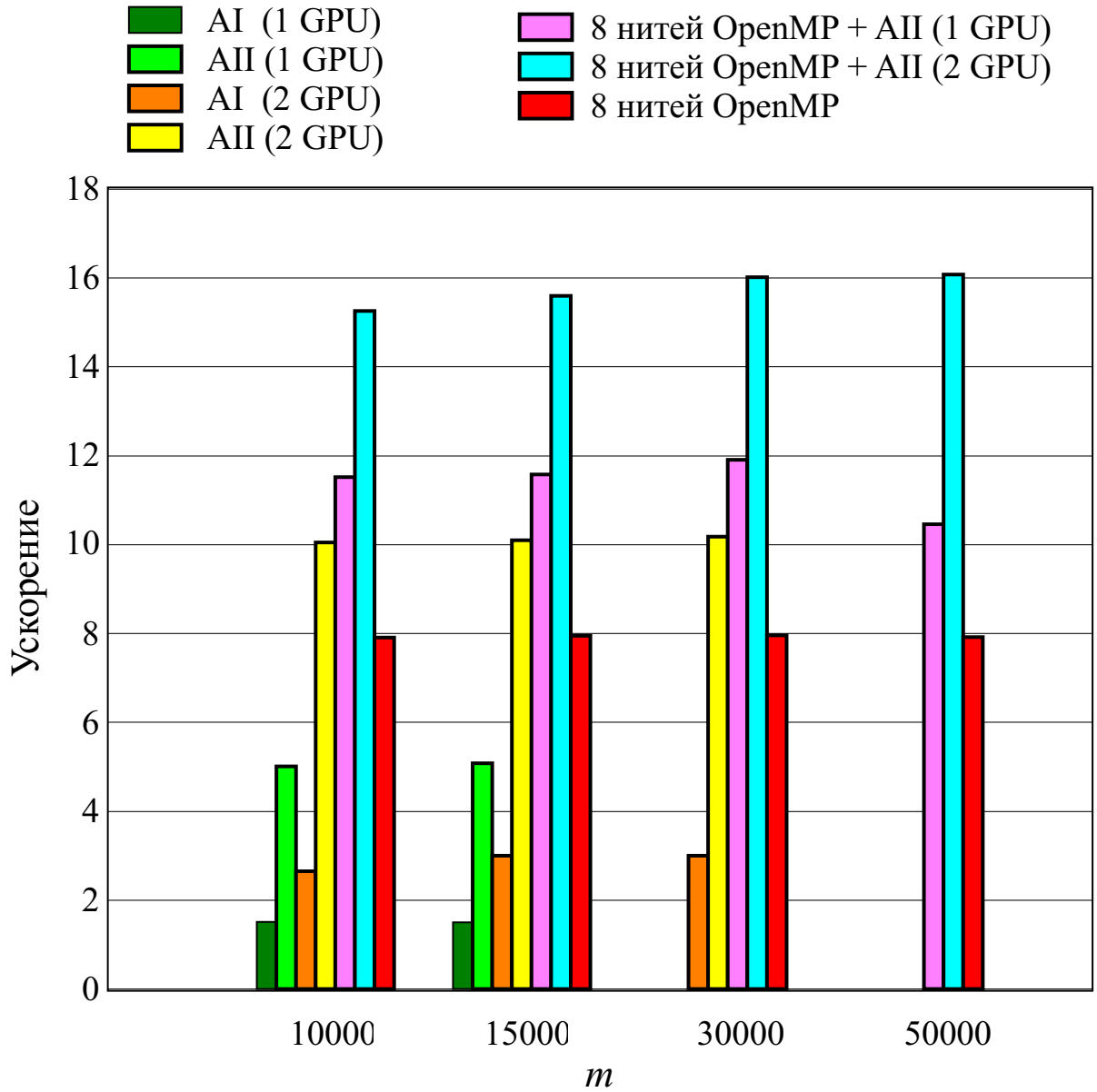

Рис. 3: Ускорение при вычислении матриц $K^e, e = \overline{1, m}$, $p = 5$.

Предобусловленный метод сопряженных градиентов (МСГ)

Require: $r_0 = f$; {cublasDcopy}
$s_0 = \mathcal{M}^{-1} r_0$; {kernel-функция собственной разработки}
$p_0 = s_0$; {cublasDcopy}
$\rho_0 = (s_0, r_0)$; {cublasDdot, опция reduction, MPI_Allreduce}
$||r_0||_2 := \sqrt{(r_0, r_0)}$; {cublasDdot, опция reduction, MPI_Allreduce}

1: **for** $i = 0, 1, \ldots$ **do**
2: $q_i := Kp_i$; {kernel-функции собственной разработки, MPI_Allgather}
3: $\gamma_i := (p_i, q_i)$; {cublasDdot, опция reduction, MPI_Allreduce}
4: $\alpha_i := -\rho_i / \gamma_i$;
5: $u_{i+1} := u_i - \alpha_i p_i$; {последовательно cublasDscal и cublasDasxpy}
6: $r_{i+1} := r_i + \alpha_i q_i$; {последовательно cublasDscal и cublasDasxpy}
7: $||r_{i+1}||_2 := \sqrt{(r_{i+1}, r_{i+1})}$; {cublasDdot, опция reduction, MPI_Allreduce}
8: **if** $(||r_{i+1}||_2 / ||r_0||_2 < \varepsilon)$ **then**
9: $\bar{u} := u_{i+1}$; {Копирование GPU -> CPU}
10: **return**
11: **end if**
12: $s_{i+1} := \mathcal{M}^{-1} r_{i+1}$; {kernel-функция собственной разработки}
13: $\rho_{i+1} := (s_{i+1}, r_{i+1})$; { cublasDdot, опция reduction, MPI_Allreduce}
14: $\beta_i := \rho_{i+1} / \rho_i$;
15: $p_{i+1} := s_{i+1} + \beta_i p_i$; {последовательно cublasDscal и cublasDasxpy}
16: $i = i + 1$;
17: **end for**

Здесь $\mathcal{M} = diag(K)$, $diag(K) = \sum_{e=1}^{m} C_e^T diag(K^e) C_e$, $\varepsilon = 10^{-6}$.

Матрично-векторное произведение $q = Kp$ на GPU

Матрица K в компактном формате (CSR)

$$q = \left[q_i\right]_{N\times 1}, \quad \underbrace{q_i = \sum_{j=1}^{N} k_{ij} \cdot p_j,}_{\text{одна нить CUDA}} \quad k_{ij} \neq 0, \; j = \overline{1, N}. \tag{1}$$

Матрично-векторное произведение $q = Kp$ на GPU

Матрица K в компактном формате (CSR)

$$q = \left[q_i\right]_{N\times 1}, \quad \underbrace{q_i = \sum_{j=1}^{N} k_{ij} \cdot p_j,}_{\text{одна нить CUDA}} \quad k_{ij} \neq 0, \; j = \overline{1, N}. \tag{1}$$

Поэлементные схемы (EBE)

- *Схема с извлечением*

$$\underbrace{p_e \leftarrow C_e p, \;\; q_e = K^e p_e,}_{\text{одна нить CUDA}} \qquad \underbrace{q \leftarrow q + C_e^T q_e,}_{\text{конфликт при записи}} \qquad e = \overline{1, m}. \tag{2}$$

Матрично-векторное произведение $q = Kp$ на GPU

Матрица K в компактном формате (CSR)

$$q = \left[q_i\right]_{N\times 1}, \quad \underbrace{q_i = \sum_{j=1}^{N} k_{ij} \cdot p_j}_{\text{одна нить CUDA}}, \quad k_{ij} \neq 0, \; j = \overline{1, N}. \tag{1}$$

Поэлементные схемы (EBE)

- *Схема с извлечением*

$$\underbrace{p_e \leftarrow C_e p, \;\; q_e = K^e p_e,}_{\text{одна нить CUDA}} \qquad \underbrace{q \leftarrow q + C_e^T q_e,}_{\text{конфликт при записи}} \qquad e = \overline{1, m}. \tag{2}$$

- *Схема с разнесением*

$$\underbrace{\bar{p} = Cp,}_{\tilde{N} \text{ нитей CUDA}} \qquad \underbrace{\tilde{q} = \tilde{K}\bar{p},}_{\tilde{N} \text{ нитей CUDA}} \qquad \underbrace{q = C^T \tilde{q}.}_{N \text{ нитей CUDA}} \tag{3}$$

Здесь матрицы $\tilde{K} = [K_{ii}]_{m\times m}$, $C = [C_i]_{1\times m}$, $K_{ii} = K^e$, $C_i = C_e^T$, $i, e = \overline{1, m}$;
векторы $p_e, q_e \in \mathbb{R}^{N_e}$, $\bar{p}, \tilde{q} \in \mathbb{R}^{\tilde{N}}$, $\tilde{N} = m \cdot N_e$, $p, q \in \mathbb{R}^N$ и $N_e \ll N$.

Сравнение схем решения конечно-элементных СЛАУ

Таблица 3: Сравнение поэлементной схемы с решением собранной системы[2]

Тип геометрии элемента	Число элементов / узлов	N/Nnz	Решение системы, сек CPU / GPU CSR	EBE
Треугольник	200000/101101	202202/1782214	180 / 1.1	160/6.1/26
	394800/397902	397902/3513958	508 / 2.6	460 / 15.5
	1195600/600495	1200990 / 10628410	2758/11.6/238	2423/60.7
Четырехугольник	100000/101101	202202/2206990	134 / 1.2	106 / 3.2
	197400/397902	397902 / 4352690	346/3.1/112	286 / 9.1
	597800/600495	1200990/13168770	1876 / 13.8	1675/39.1/43
Тетраэдр	119360/23493	70479 / 1545935	19 / 0.3	72/3.4/21
	221184/41377	124131/2867779	40 / 0.5	209 / 3.9
	647731/119007	357021 / 8269721	126 / 1.3 /97	471 / 12.1
Шестигранник	21970/25676	77028/2915736	10 / 0.3	21 / 0.6
	49130/55404	166212 / 6453944	30 / 0.6 / 50	62 / 1.3
	106480/116909	350727/13881759	83 / 1.4	175/3.2/55

[2] Копысов С.П. и др. Параллельная реализация конечно-элементных алгоритмов на графических ускорителях в программном комплексе FEStudio // Компьютерные исследования и моделирование. – 2014. – Т. 6. – № 1. – С. 79–97.

Таблица 4: Время решения (сек.) на сетке $m = 106480$, $p = 1$, $n_G = 8$, $N_e = 24$

Шаг МКЭ	CPU	GPU		CPU+GPU		
	CSR	EBE	EBE1	CSR	EBE	EBE1
Вычисление K^e, $e = \overline{1,m}$	1373.35	1.97	1.97	1373.5	0.56	0.56
Формирование K		—	—		—	—
Решение $Ku = f$	182.06	11.08	12.84	3.36	11.08	12.84
Общее время	1555.41	13.08	14.81	1376.86	11.64	13.4

Таблица 5: Время решения (сек.) на сетке $m = 106480$, $p = 1$, $n_G = 8$, $N_e = 24$

Шаг МКЭ	OpenMP+CUDA	MPI+OpenMP+CUDA	MPI+CUDA		
	2 GPU	2×2 GPU	2 GPU	2×1 GPU	2×2 GPU
Вычисление K^e, $e = \overline{1,m}$	1.01	0.5	1.02	1.03	0.54
Решение $Ku = f$	7.39	5.2	7.67	8.63	6.11
Общее время	8.4	5.7	8.69	9.66	6.65