Параллельная композиция в поэлементной схеме метода конечных элементов[1]

Копысов С.П., Новиков А.К., Пиминова Н.К.

Институт механики УрО РАН, г. Ижевск

Международная научная конференция
Параллельные вычислительные технологии (ПаВТ 2016)
Северный (Арктический) федеральный университет, Архангельск,
28 марта – 1 апреля 2016 г.

[1] Работа выполнена при поддержке РФФИ (проекты 16-01-00129-а, 14-01-00055-а).

Операция параллельной композиции

Введем параллельную композицию, как операцию, состоящую из двух этапов, взаимосвязанных, но разнесенных по времени выполнения:

- разделение с заданными свойствами (локализация данных и сбалансированность вычислительной нагрузки);
- объединение / суммирование (сборка) распределенных данных.

Применение композиции в конечно-элементных вычислениях:

- Сборка глобальной матрицы жесткости $K = \mathcal{A}^T \tilde{K} \mathcal{A}$.
- Сборка векторов в поэлементных схемах $q = \mathcal{A}^T \tilde{q}$.
- Интегрирование локальных матриц жесткости.

Задачи параллельной композиции:

- Исключения “состояние гонки“ на этапе сборки при композиции.
- Локализация данных.
- Сбалансированность вычислительной нагрузки.

Параллельная сборка в поэлементных схемах

$$\underbrace{\begin{bmatrix} \bar{p}^{(1)} \\ \bar{p}^{(2)} \\ \vdots \\ \bar{p}^{(m)} \end{bmatrix}}_{\bar{p}} \times \underbrace{\begin{bmatrix} \tilde{K}_{11} & 0 & \dots & 0 \\ 0 & \tilde{K}_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{K}_{mm} \end{bmatrix}}_{\tilde{K}} = \underbrace{\begin{bmatrix} \tilde{q}^{(1)} \\ \tilde{q}^{(2)} \\ \vdots \\ \tilde{q}^{(m)} \end{bmatrix}}_{\tilde{q}} \xrightarrow{\mathcal{A}^T} \underbrace{\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix}}_{q}.$$

Произведение $q = Kp$ разделяется на две операции:

- произведение $\tilde{q} = \tilde{K}\bar{p}$, здесь $\bar{p} = \mathcal{A}p$;
- сборку $q = \mathcal{A}^T\tilde{q}$.

Произведение $\tilde{q} = \tilde{K}\bar{p}$

```
#pragma omp parallel num_threads(procs)
{
     int thread = omp_get_thread_num();
     int ib = subloop[thread];
     int ie = subloop[thread+1];

     for(int e = ib; e < ie; e++)
     {
             MV(e, tilde_K, tilde_p, tilde_q, N_e);
     }
}
```

Сборка вектора $q = \mathcal{A}^T\tilde{q}$.

```
#pragma omp parallel num_threads(procs)
{
...
     for(int e = ib; e < ie; e++)
     {
// #pragma omp critical
          tilde_q2q(e, AT, tilde_q, q);
     }
}
```

Фрагмент тела функции tilde_q2q

```
for(int ldof=0;ldof<N_e;ldof++)
{
...
  q[gdof] += tilde_q[N_e * e + ldof];
}
```

Особенности неструктурированных сеток.

1. Отношение соседства в сетке не определяется приращением номера сеточного объекта (узла, ребра, ячейки).
2. Переменная валентность узлов, в том числе, внутри области.
3. Как правило, переменный шаг сетки.

Отношение соседства для ячеек сетки для поэлементных схем МКЭ.

Будем полагать, что ячейка e_k является соседней с ячейкой e_j, если $v(e_j) \bigcap v(e_k) \neq \emptyset$, где $v(e_j)$ и $v(e_k)$ — соответствующие ячейкам e_j и e_k множества вершин.

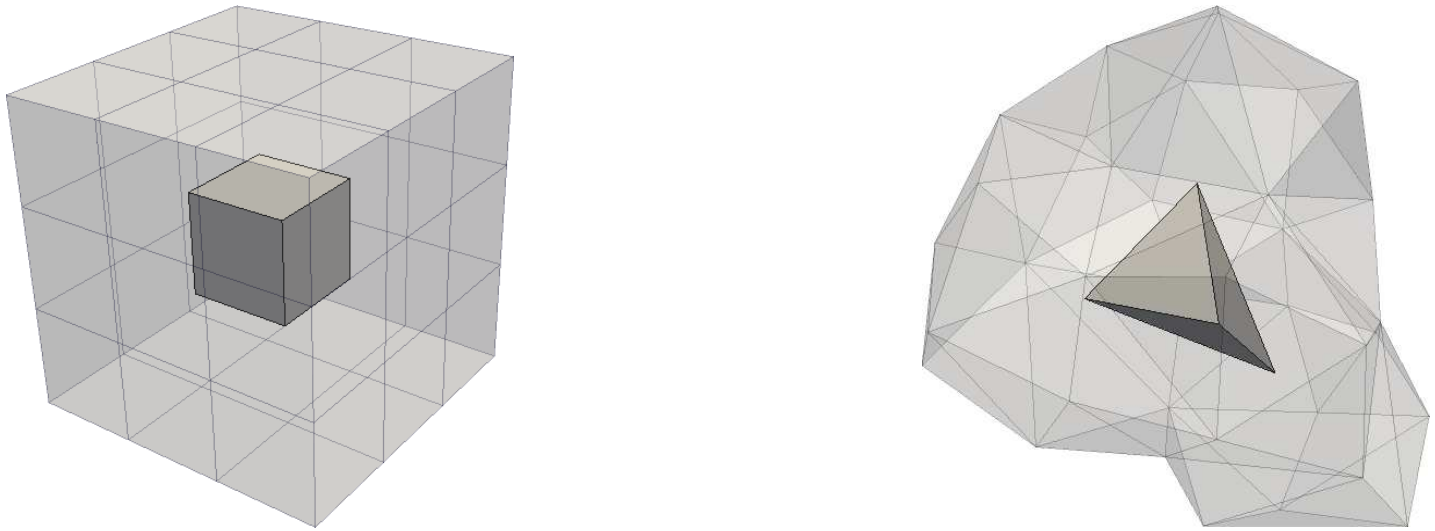

Рис. 1: Примеры ячеек с “соседями“.

Алгоритм разделения неструктурированной сетки на слои.

1. Задается начальный слой $s_i = \{e_j^{(i)}\}, i = 1, j = \overline{1, m_i}, m_i = |\{e_j^{(i)}\}|$;
2. Слои $i = 2, 3, ..., n_s$ определяются из выражения

$$s_i = \begin{cases} Adj(S_{i-1}), & i = 2; \\ Adj(S_{i-1}) \setminus S_{i-2}, & i = \overline{3, n_s}, \end{cases}$$

где $s_i = \{e_j^{(i)}\}$ — множество ячеек (конечных элементов) в слое i;

$e_j^{(i)}$ — ячейка с номером j в слое i;

$Adj(s_i) = \bigcup_{j=1}^{m_i} Adj(e_j^{(i)})$ — множество ячеек сетки соседних со слоем i;

m_i — число ячеек в слое i.

Неструктурированные сетки, разделенные на слои

Рис. 2: Сетка из шестигранных ячеек, $m = 31744$, $n_s = 64$.

Рис. 3: Сетка из четырехгранных ячеек, $m = 485843$, $n_s = 132$.

Варианты построения подмножеств ячеек.

1. Многоуровневое разделение сетки (METIS).
2. На основе объединения слоев:
   1. Блочный.
   2. По четности номеров слоев.

Многоуровневое разделение сетки (METIS).

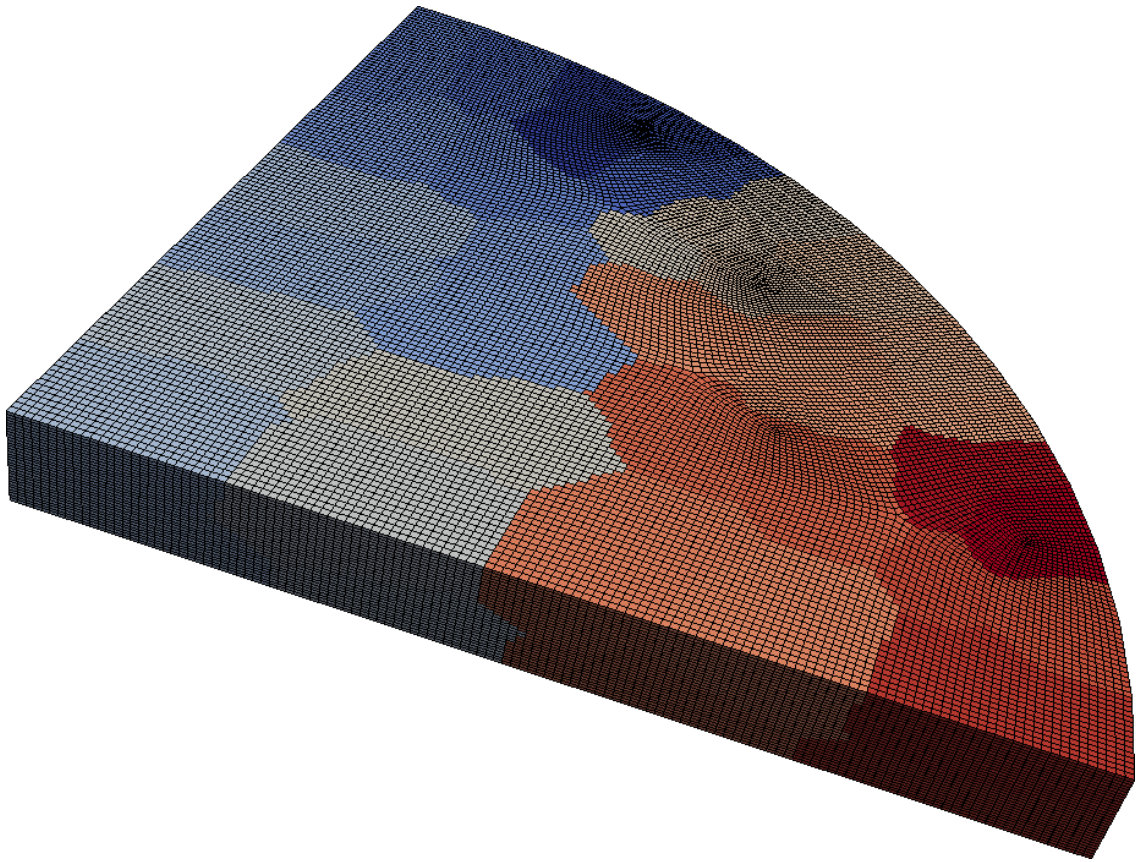

Рис. 4: Подмножества ячеек, полученные разделением сетки программой mpmetis (METIS 5.1.0, $m = 507904$, $n_p = 60$).

Блочный вариант построения подмножеств ячеек.

1. Объединение слоев $S = \bigcup_{i=1}^{n_s} s_i$.
2. Разделение S на подмножества $S_i : |S_i| = m/n_p$, $i = \overline{1, n_p}$, где n_p — число параллельных вычислительных процессов (рис. 5).

Рис. 5: Подмножества, полученные при блочном варианте построения.

Построения подмножеств ячеек по четности номеров слоев.

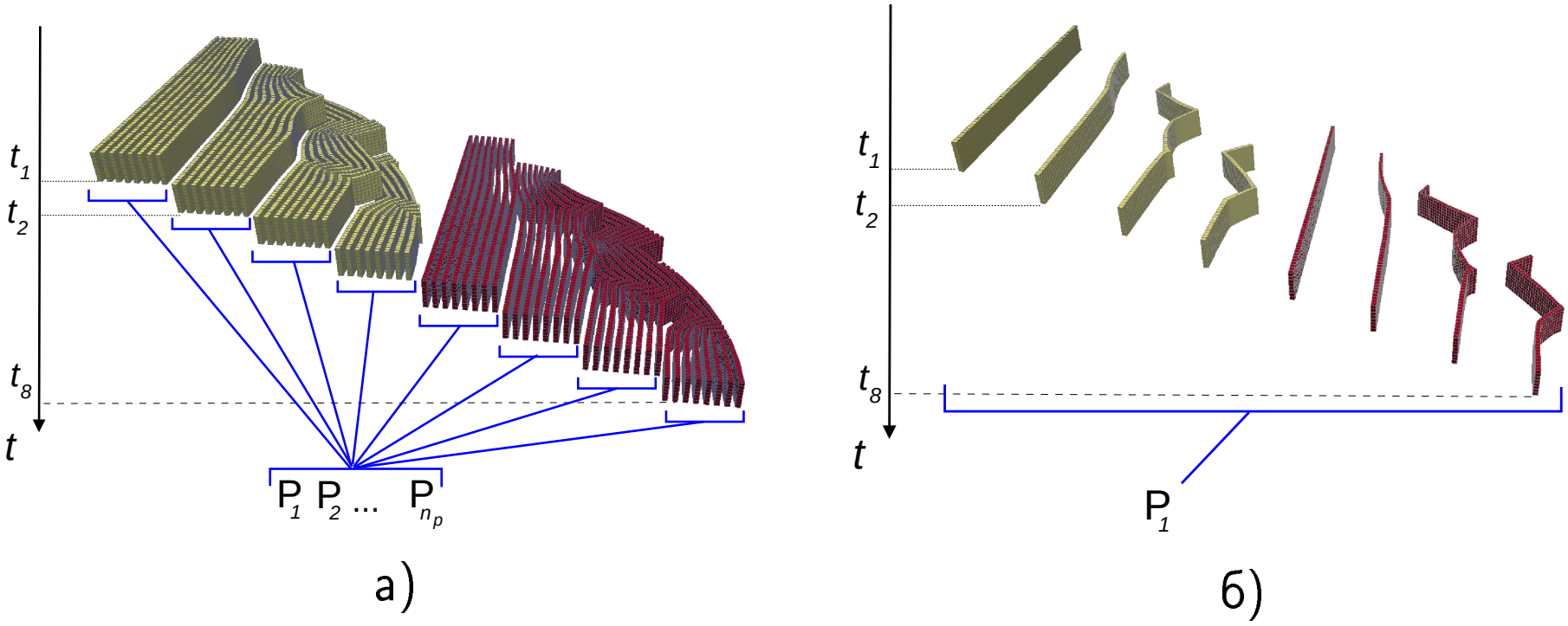

Рис. 6: Подмножества ячеек по четности номеров слоев и их распределение по процессам: а) слои, ячейки которых участвуют в параллельных вычислениях восьми процессов ($n_p = 8$); б) последовательность слоев (подмножество S_1), над которыми выполняет вычисление первый процесс.

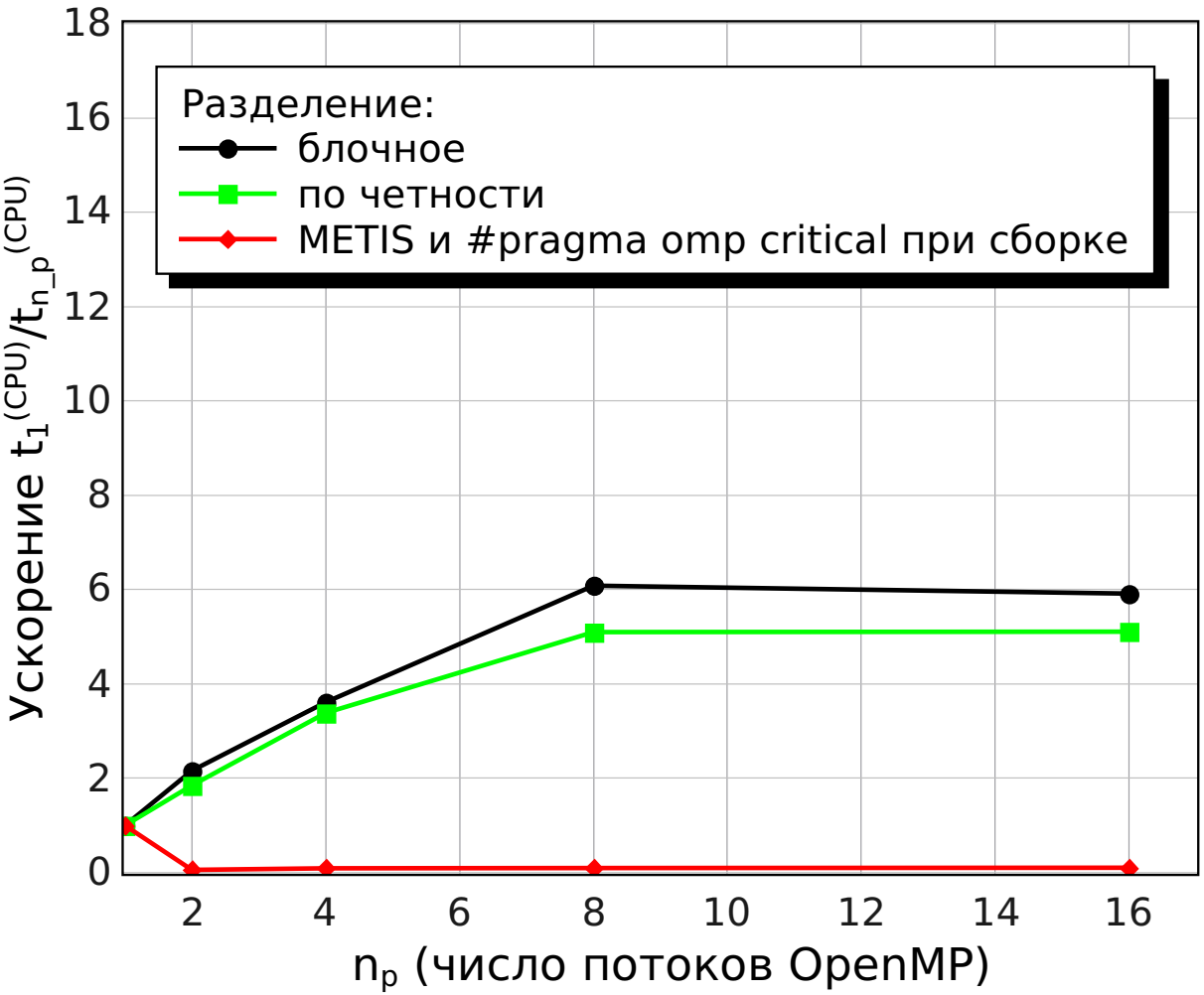

Рис. 7: Ускорение сборки вектора $q = \mathcal{A}^T \tilde{q}$ на восьмиядерном процессоре Xeon E5-2690, здесь и далее $n_p = 16$ соответствует двум Xeon E5-2690.

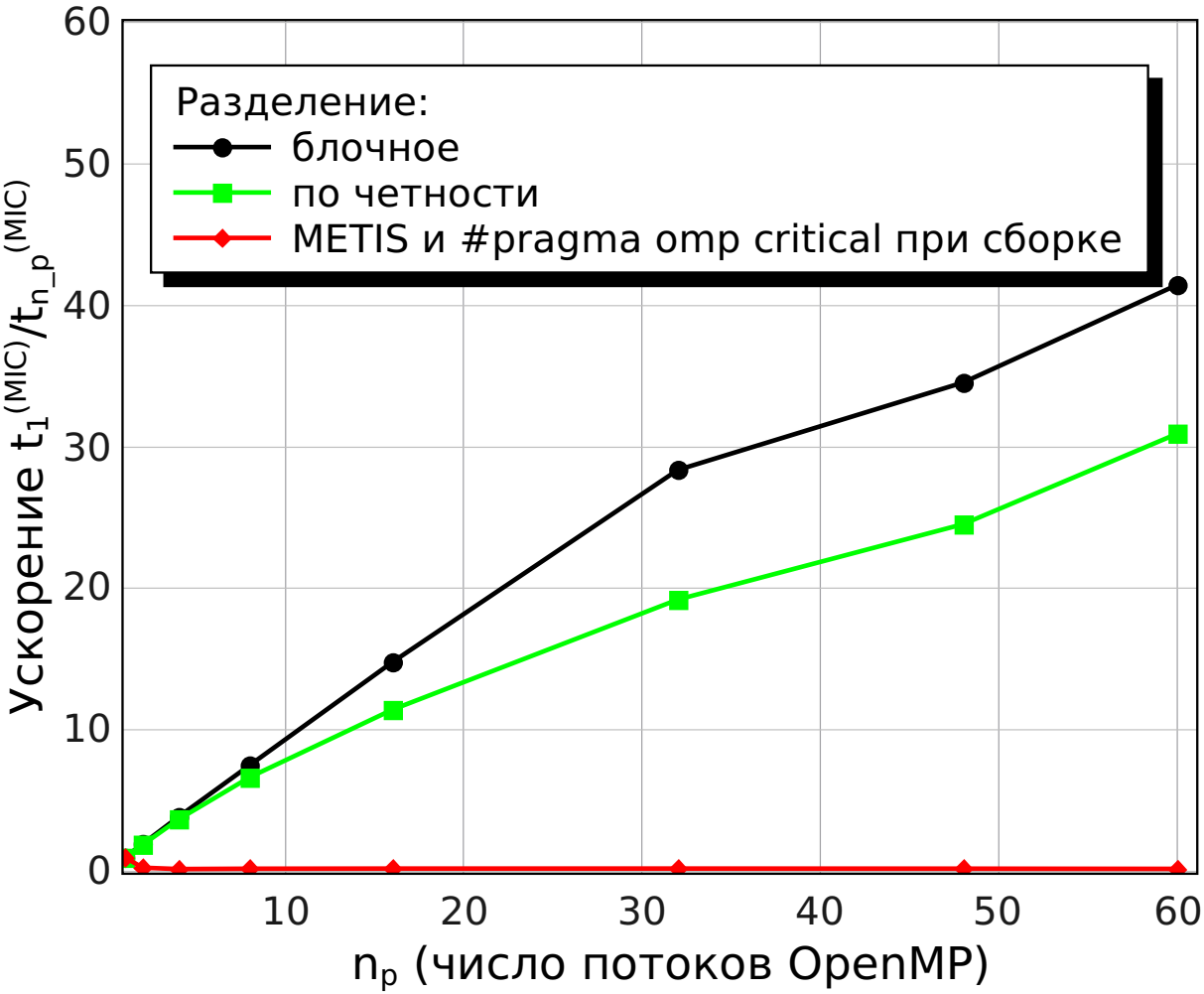

Рис. 8: Ускорение сборки вектора q на процессоре Xeon Phi 7110X

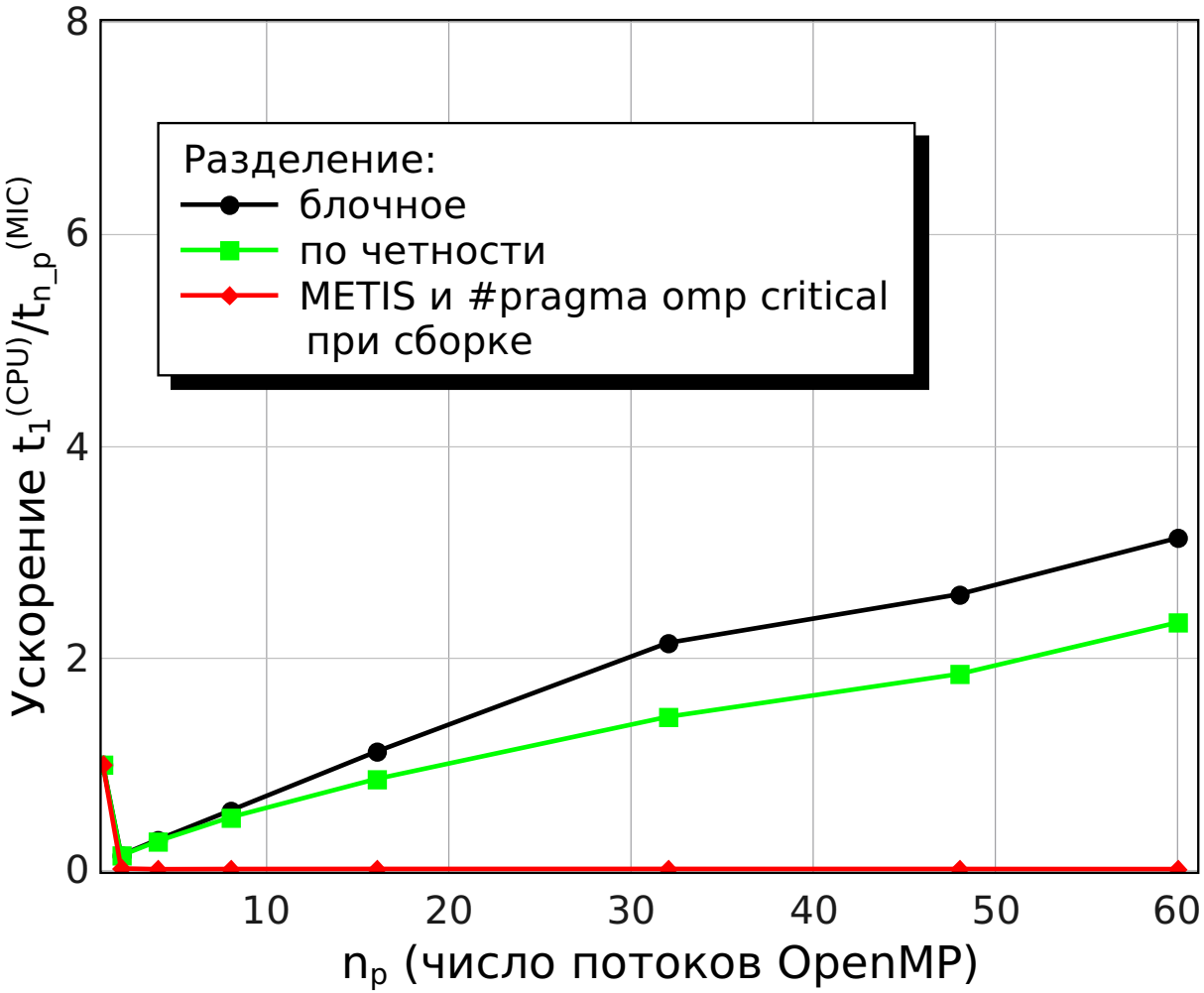

Рис. 9: Ускорение сборки вектора q на Phi 7110X относительно одного ядра E5-2690

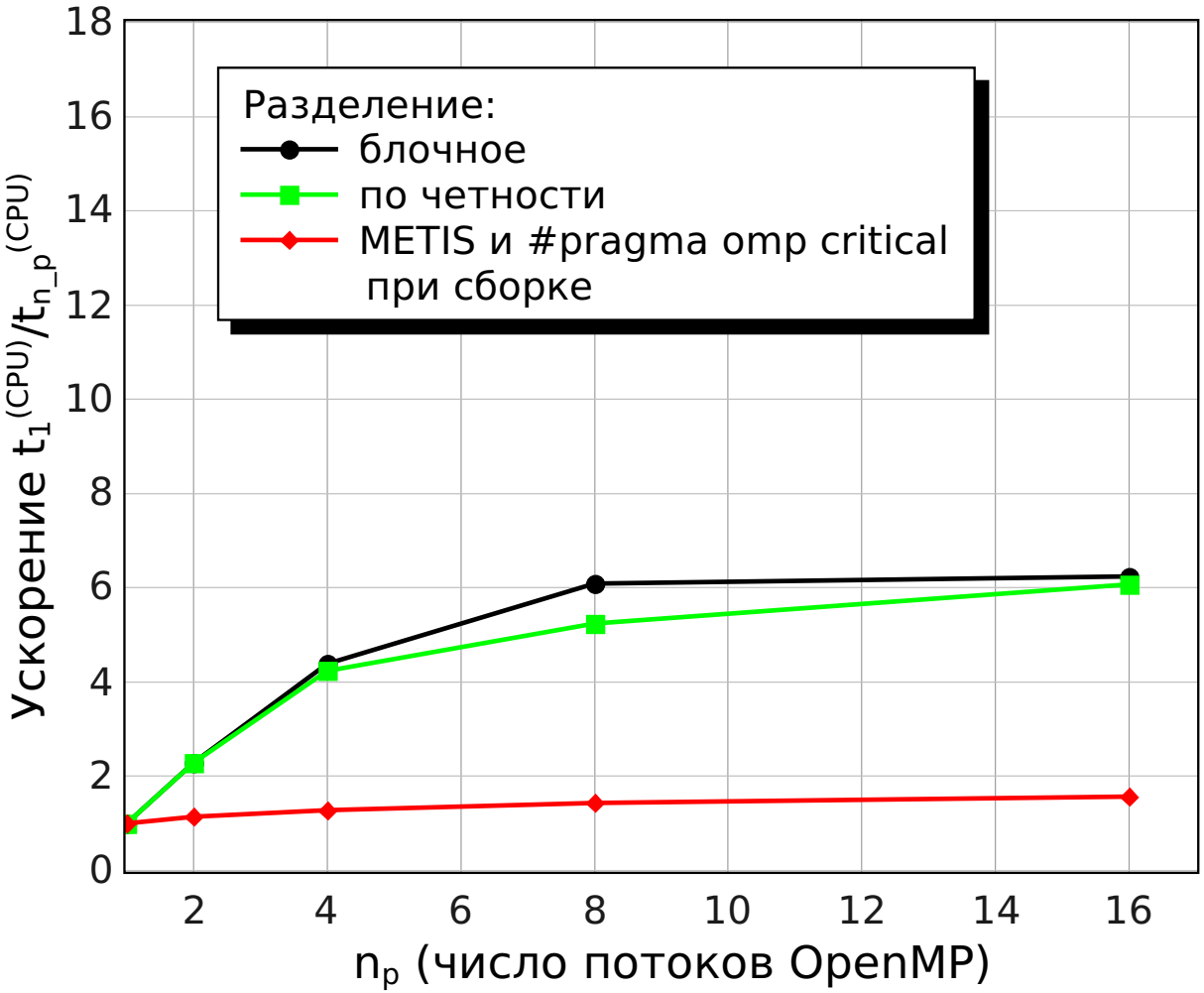

Рис. 10: Ускорение при вычислении $q = \mathcal{A}^T \tilde{K} \bar{p}$ на процессоре Xeon E5-2690

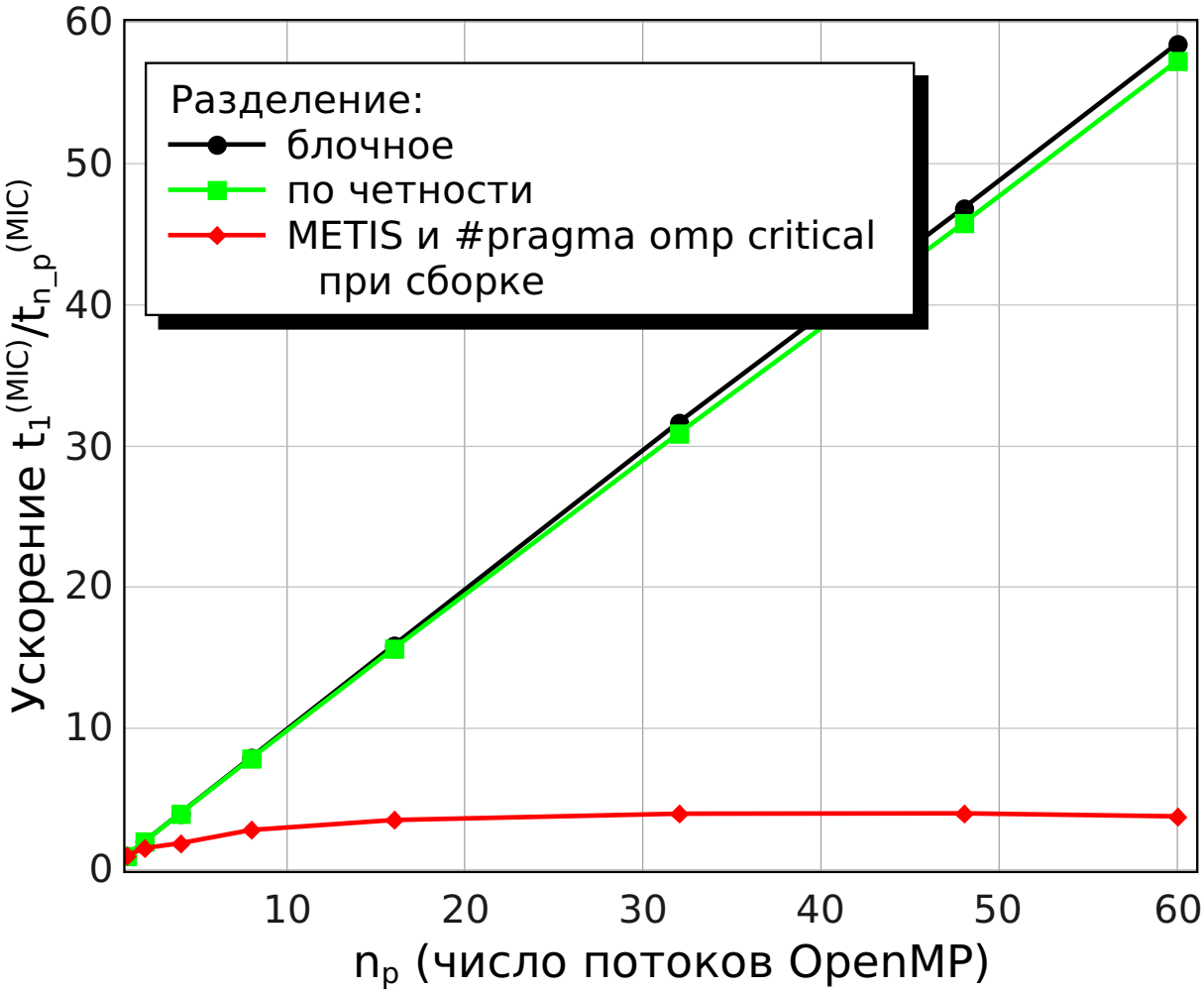

Рис. 11: Ускорение вычислений $q = \mathcal{A}^T \tilde{K} \bar{p}$ на процессоре Xeon Phi

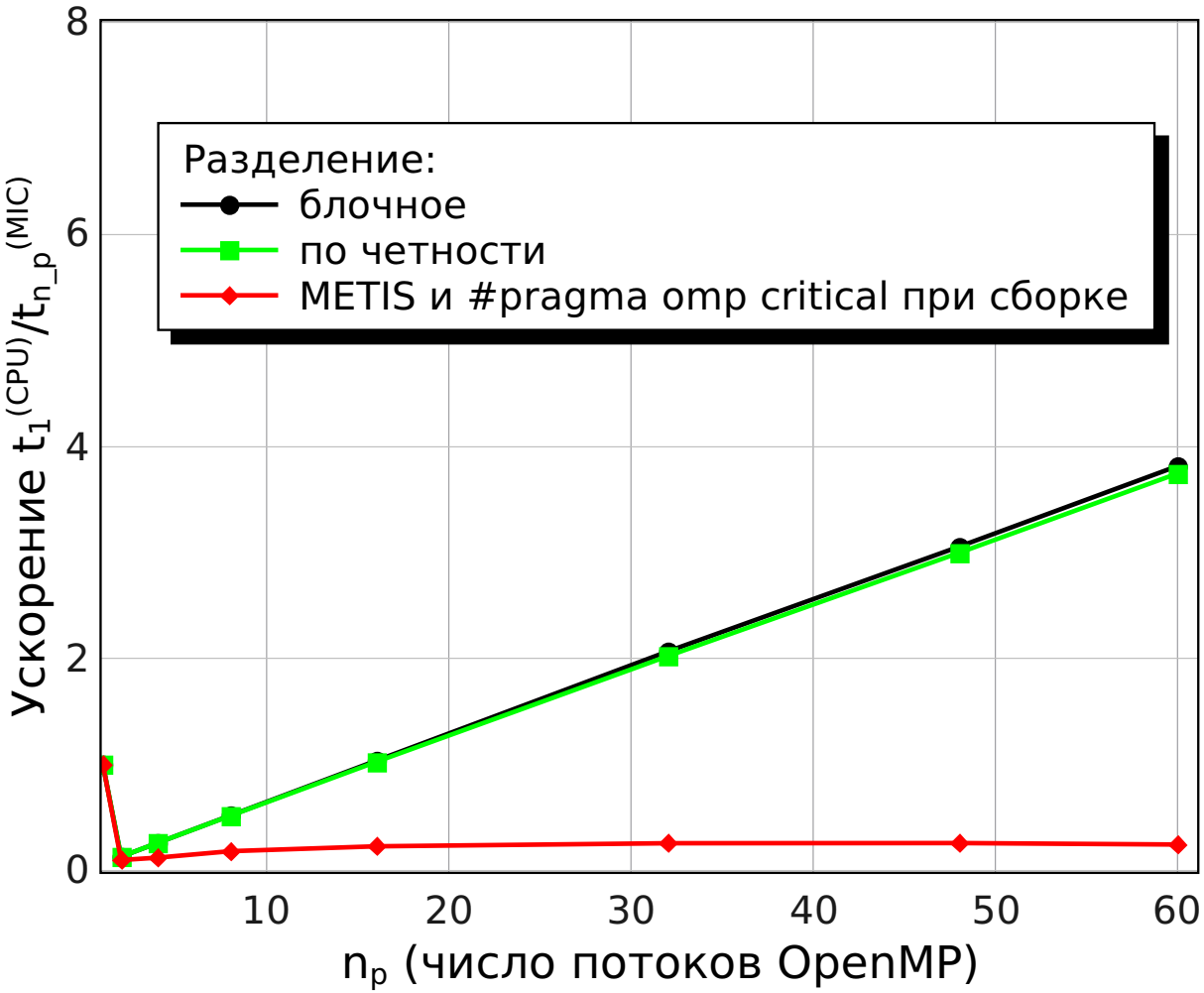

Рис. 12: Ускорение вычислений $q = \mathcal{A}^T \tilde{K} \bar{p}$ на Xeon Phi относительно Xeon E5-2690

Заключение

- Предложенные варианты разделения сетки позволили исключить критическую секцию на этапе сборки векторов и обеспечили существенное ускорение вычислений, как при сборке, так при вычислении матрично-векторного произведения в поэлементной схеме метода конечных элементов.
- Дальнейшие исследования будут связаны с оценкой масштабируемости предложенных алгоритмов послойного упорядочения и сборки в параллельной конечно-элементной композиции.