Послойная декомпозиция в конечно-элементном анализе на гибридных архитектурах

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

Институт механики УрО РАН, г. Ижевск

Международная конференция
«Суперкомпьютерные дни в России»,
г. Москва,
25 – 26 сентября 2017 г.

Введение

Проблемы параллельных алгоритмов на современных вычислительных архитектурах и их решение:

- Конфликты общей памяти.
- Неоднородность ресурсов.
- Ограничение производительности алгоритмов доступом к памяти.

Введение

Проблемы параллельных алгоритмов на современных вычислительных архитектурах и их решение:

- Декомпозиция алгоритма на независимые части.
- Неоднородность ресурсов.
- Ограничение производительности алгоритмов доступом к памяти.

Введение

Проблемы параллельных алгоритмов на современных вычислительных архитектурах и их решение:

- Декомпозиция алгоритма на независимые части.
- Распределение вычислений на ресурсы, в зависимости от их производительности.
- Ограничение производительности алгоритмов доступом к памяти.

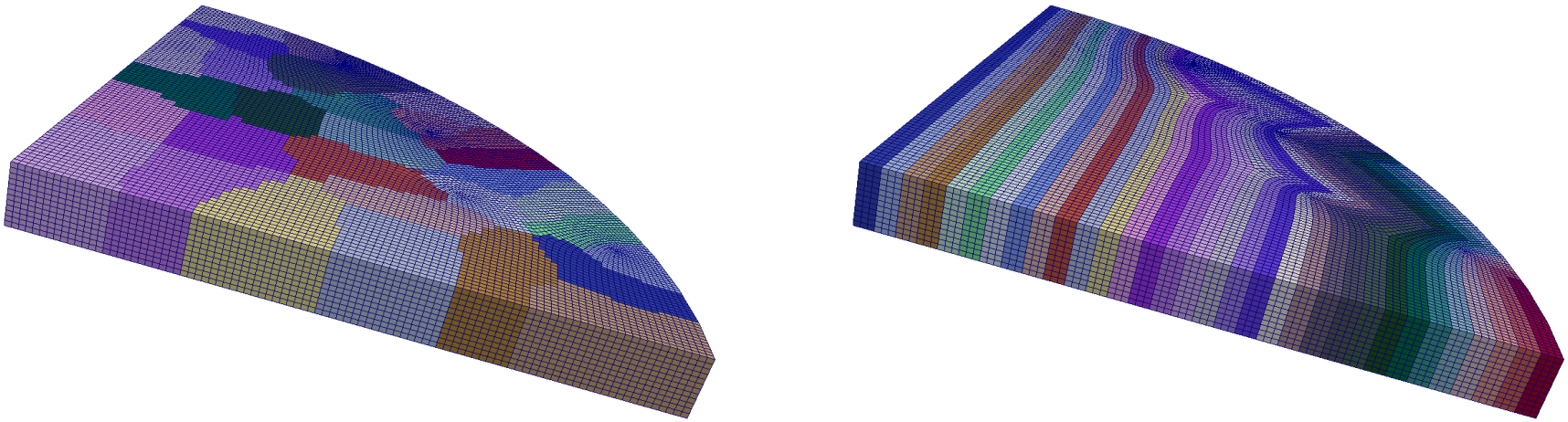
Введение

Проблемы параллельных алгоритмов на современных вычислительных архитектурах и их решение:

- Декомпозиция алгоритма на независимые части.
- Распределение вычислений на ресурсы, в зависимости от их производительности.
- Увеличение интенсивности вычислений.

Варианты декомпозиции области:

1. Разделение сетки с ветвлением (METIS).
2. Разделение сетки без ветвления на основе объединения слоёв:
   1. Блочный.
   2. По четности номеров слоев.



а) б)

Рис. 1 : Разделение 1/4 мембраны, число конечных элементов $N_{fe} = 107136$, число подобластей $n_\Omega = 32$: а) с ветвлением, б) без ветвления.

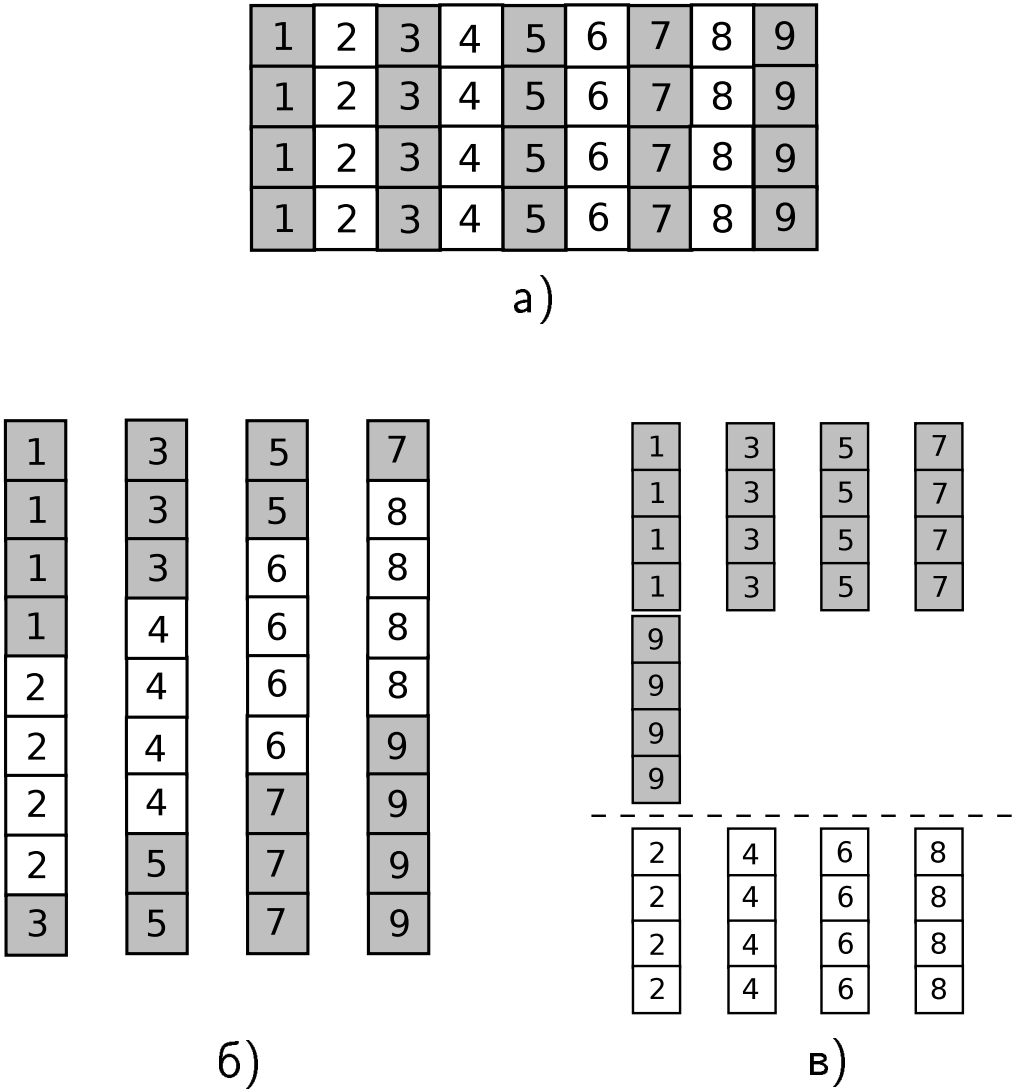


Рис. 2 : Разделение а) и варианты объединения слоёв в блоки для $n_\Omega = 4$: б) блочная схема; в) схема по чётности.

Инициализация:
$A, \bar{M} \in \mathbb{R}^{N \times N}$;
$u, r, p, q, z \in \mathbb{R}^N$;
$i \leftarrow 0$;
$u_i \leftarrow 0$;
$r_i \leftarrow f$;
$z_i \leftarrow \bar{M} r_i$;
$p_i \leftarrow z_i$;
$\rho_i \leftarrow (r_i, z_i)$;
Выполнять, пока $||r_i||_2 / ||b||_2 > \varepsilon$
{
$q_i \leftarrow A p_i$;
$\alpha_i \leftarrow (r_i, z_i)/(q_i, p_i)$;
$u_{i+1} \leftarrow u_i + \alpha_i p_i$;
$r_{i+1} \leftarrow r_i - \alpha_i q_i$;
$z_{i+1} \leftarrow \bar{M} r_{i+1}$;
$\rho_{i+1} \leftarrow (r_{i+1}, z_{i+1})$;
$\beta_{i+1} \leftarrow \rho_{i+1}/\rho_i$;
$p_{i+1} \leftarrow z_{i+1} + \beta_{i+1} p_i$;
$i \leftarrow i + 1$.
}

Инициализация:
$\tilde{A} \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$, $C \in \mathbb{Z}^{\tilde{N}\times N}$,
$\bar{M} \in \mathbb{R}^{N\times N}$;
$u, r, p, q, z \in \mathbb{R}^N$;
$i \leftarrow 0$;
$u_i \leftarrow 0$;
$r_i \leftarrow f$;
$z_i \leftarrow \bar{M} r_i$;
$p_i \leftarrow z_i$;
$\rho_i \leftarrow (r_i, z_i)$;
Выполнять, пока $||r_i||_2/||b||_2 > \varepsilon$
{
$q_i \leftarrow C^T \tilde{A} C p_i$;
$\alpha_i \leftarrow (r_i, z_i)/(q_i, p_i)$;
$u_{i+1} \leftarrow u_i + \alpha_i p_i$;
$r_{i+1} \leftarrow r_i - \alpha_i q_i$;
$z_{i+1} \leftarrow \bar{M} r_{i+1}$;
$\rho_{i+1} \leftarrow (r_{i+1}, z_{i+1})$;
$\beta_{i+1} \leftarrow \rho_{i+1}/\rho_i$;
$p_{i+1} \leftarrow z_{i+1} + \beta_{i+1} p_i$;
$i \leftarrow i + 1$.
}

Инициализация:
$\tilde{A} \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$, $C \in \mathbb{Z}^{\tilde{N} \times N}$,
$\bar{M} \in \mathbb{R}^{N \times N}$;
$u, r, p, q, z \in \mathbb{R}^N$;
$i \leftarrow 0$;
$u_i \leftarrow 0$;
$r_i \leftarrow f$;
$z_i \leftarrow \bar{M} r_i$;
$p_i \leftarrow z_i$;
$\rho_i \leftarrow (r_i, z_i)$;
Выполнять, пока $||r_i||_2 / ||b||_2 > \varepsilon$
{
$q_i \leftarrow C^T \tilde{A} C p_i$;
$\alpha_i \leftarrow (r_i, z_i)/(q_i, p_i)$;
$u_{i+1} \leftarrow u_i + \alpha_i p_i$;
$r_{i+1} \leftarrow r_i - \alpha_i q_i$;
$z_{i+1} \leftarrow \bar{M} r_{i+1}$;
$\rho_{i+1} \leftarrow (r_{i+1}, z_{i+1})$;
$\beta_{i+1} \leftarrow \rho_{i+1}/\rho_i$;
$p_{i+1} \leftarrow z_{i+1} + \beta_{i+1} p_i$;
$i \leftarrow i + 1$.
}

- Преобразование $\bar{p} \leftarrow Cp$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
{ int fe_s = N_i * fe;
   for(int i=0; i < N_i; i++)
      pbar[fe_s + i] = p[ I[fe][i] ];
}
```

Инициализация:
$\tilde{A} \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$, $C \in \mathbb{Z}^{\tilde{N}\times N}$,
$\bar{M} \in \mathbb{R}^{N\times N}$;
$u, r, p, q, z \in \mathbb{R}^N$;
$i \leftarrow 0$;
$u_i \leftarrow 0$;
$r_i \leftarrow f$;
$z_i \leftarrow \bar{M} r_i$;
$p_i \leftarrow z_i$;
$\rho_i \leftarrow (r_i, z_i)$;
Выполнять, пока $||r_i||_2/||b||_2 > \varepsilon$
{
$q_i \leftarrow C^T \tilde{A} C p_i$;
$\alpha_i \leftarrow (r_i, z_i)/(q_i, p_i)$;
$u_{i+1} \leftarrow u_i + \alpha_i p_i$;
$r_{i+1} \leftarrow r_i - \alpha_i q_i$;
$z_{i+1} \leftarrow \bar{M} r_{i+1}$;
$\rho_{i+1} \leftarrow (r_{i+1}, z_{i+1})$;
$\beta_{i+1} \leftarrow \rho_{i+1}/\rho_i$;
$p_{i+1} \leftarrow z_{i+1} + \beta_{i+1} p_i$;
$i \leftarrow i + 1$.
}

- Преобразование $\bar{p} \leftarrow Cp$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
{ int fe_s = N_i * fe;
  for(int i=0; i < N_i; i++)
    pbar[fe_s + i] = p[ I[fe][i] ];
}
```

- Произведение $\tilde{q} \leftarrow \tilde{A}\bar{p}$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
{ int fe_s = N_i * fe;
  for(int i=0; i < N_i; i++)
    for(int j=0; j < N_i; j++)
      qtilde[fe_s+i] += Atilde[fe][i][j] * pbar[fe_s+j];
}
```

Инициализация:
$\tilde{A} \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$, $C \in \mathbb{Z}^{\tilde{N}\times N}$,
$\bar{M} \in \mathbb{R}^{N\times N}$;
$u, r, p, q, z \in \mathbb{R}^N$;
$i \leftarrow 0$;
$u_i \leftarrow 0$;
$r_i \leftarrow f$;
$z_i \leftarrow \bar{M} r_i$;
$p_i \leftarrow z_i$;
$\rho_i \leftarrow (r_i, z_i)$;
Выполнять, пока $||r_i||_2/||b||_2 > \varepsilon$
{
$q_i \leftarrow C^T \tilde{A} C p_i$;
$\alpha_i \leftarrow (r_i, z_i)/(q_i, p_i)$;
$u_{i+1} \leftarrow u_i + \alpha_i p_i$;
$r_{i+1} \leftarrow r_i - \alpha_i q_i$;
$z_{i+1} \leftarrow \bar{M} r_{i+1}$;
$\rho_{i+1} \leftarrow (r_{i+1}, z_{i+1})$;
$\beta_{i+1} \leftarrow \rho_{i+1}/\rho_i$;
$p_{i+1} \leftarrow z_{i+1} + \beta_{i+1} p_i$;
$i \leftarrow i + 1$.
}

- Преобразование $\bar{p} \leftarrow Cp$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
{ int fe_s = N_i * fe;
  for(int i=0; i < N_i; i++)
    pbar[fe_s + i] = p[ I[fe][i] ];
}
```

- Произведение $\tilde{q} \leftarrow \tilde{A}\bar{p}$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
{ int fe_s = N_i * fe;
  for(int i=0; i < N_i; i++)
    for(int j=0; j < N_i; j++)
    qtilde[fe_s+i] += Atilde[fe][i][j] * pbar[fe_s+j];
}
```

- Поэлементная сборка вектора $q \leftarrow C^T \tilde{q}$

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
  for(int i=0; i < N_i; i++)
  {int fe_s = N_i * fe;
  #pragma omp critical
    q[I[fe_s+i]] += qtilde[fe_s+i];
  }
```

- Поэлементная сборка без синхронизации

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
   for(int i=0; i < N_i; i++)
   {
      q[l[fe_s+i]] += qtilde[fe_s+i];
   }
```

- Поэлементная сборка без синхронизации

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
  for(int i=0; i < N_i; i++)
  {
    q[I[fe_s+i]] += qtilde[fe_s+i];
  }
```

- Сборка по степеням свободы

```
#pragma omp parallel for
  for(int I=0; I < N; I++)
    q[I] += qtilde[ N_i * I2fe[I] + I2i[I] ];
```

- Поэлементная сборка без синхронизации

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
  for(int i=0; i < N_i; i++)
  {
    q[I[fe_s+i]] += qtilde[fe_s+i];
  }
```

- Сборка по степеням свободы

```
#pragma omp parallel for
  for(int I=0; I < N; I++)
    q[I] += qtilde[ N_i * I2fe[I] + I2i[I] ];
```

- Умножение на матрицу инциндентности

```
#pragma omp parallel for
  for(int I=0; I < N; I++)
    for(int pos=CIb[I]; pos < CIb[I+1]; pos++)
      q[I] += qtilde[CJ[pos]] * CV[pos];
```

- Поэлементная сборка без синхронизации

```
#pragma omp parallel for
for(int fe=0; fe < N_fe; fe++)
  for(int i=0; i < N_i; i++)
  {
    q[I[fe_s+i]] += qtilde[fe_s+i];
  }
```

- Сборка по степеням свободы

```
#pragma omp parallel for
  for(int I=0; I < N; I++)
    q[I] += qtilde[ N_i * I2fe[I] + I2i[I] ];
```

- Умножение на матрицу инциндентности

```
#pragma omp parallel for
  for(int I=0; I < N; I++)
    for(int pos=CIb[I]; pos < CIb[I+1]; pos++)
      q[I] += qtilde[CJ[pos]] * CV[pos];
```

- «Умножение» на матрицу инциндентности

```
#pragma omp parallel for
for(int I=0; I < N; I++)
  for(int pos = CIb[I]; pos < CIb[I+1]; pos++)
      q[I] += qtilde[CJ[pos]];
```

Таблица 1 : Время сборки в поэлементной схеме, сек, $N_{fe} = 507904$

	CPU	2 CPU	4 CPU	8 CPU	GPU
С синхронизацией					
Неупорядоченное разделение	0.017	0.2870	0.1830	0.1780	—
Без синхронизации					
Блочное разделение	0.017	0.0077	0.0040	0.0027	0.0290
Разделение по четности	0.017	0.0099	0.0052	0.0034	—
Изменение алгоритма сборки					
Умножение на матрицу инцидентности	—	—	—	—	0.0036

Особенности крупно-блочной реализации $q_i = Ap_i$

- строится граф матрицы коэффициентов системы уравнений A;
- граф матрицы $G(A)$ разделяется на подграфы $G_k(A)$;
- матрица A разбивается на блоки A_k, соответствующие $G_k(A)$;

$$A = \begin{pmatrix} A_1^{[i_1,i_1]} & A_1^{[i_1,b_1]} & \cdots & 0 & 0 & \cdots & 0 & 0 \\ A_1^{[b_1,i_1]} & A_1^{[b_1,b_1]} & \cdots & 0 & A_1^{[b_1,b_k]} & \cdots & 0 & A_1^{[b_1,b_{n_p}]} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & A_k^{[i_k,i_k]} & A_k^{[i_k,b_k]} & \cdots & 0 & 0 \\ 0 & A_k^{[b_k,b_1]} & \cdots & A_k^{[b_k,i_k]} & A_k^{[b_k,b_k]} & \cdots & 0 & A_k^{[b_k,b_{n_p}]} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & A_{n_p}^{[i_{n_p},i_{n_p}]} & A_{n_p}^{[i_{n_p},b_{n_p}]} \\ 0 & A_{n_p}^{[b_{n_p},b_1]} & \cdots & 0 & A_{n_p}^{[b_{n_p},b_k]} & \cdots & A_{n_p}^{[b_{n_p},i_{n_p}]} & A_{n_p}^{[b_{n_p},b_{n_p}]} \end{pmatrix},$$

здесь $k \neq m$, $k, m \in [1, n_p]$ — номера блоков.

- $A_k^{[i_k,i_k]}$ — матрица, элементы которой связывают внутренние вершины;
- $A_k^{[i_k,b_k]}$, $A_k^{[b_k,i_k]}$ — матрицы, связывающие внутренние вершины с граничными;
- $A_k^{[b_k,b_m]}$ — матрица, связывающая граничные вершины k-го блока с граничными вершинами m-го блока.

- произведение $q_i = Ap_i$ реализуется для матрицы A в блочной форме.

В варианте нескольких GPU $q_i = Ap_i$ вычисляется следующим образом:

1. $q_k^b = \sum_m A_k^{[b_k,b_m]} p_m^b$, $m \in [1, n_p], m \neq k$; {*Вычисление на CPU и копирование на GPU*}
2. $q_k^i = A_k^{[i_k,i_k]} p_k^i + A_k^{[i_k,b_k]} p_k^b$. {*Вычисление на GPU*}
3. $q_k^b += A_k^{[b_k,b_k]} p_k^b + A_k^{[b_k,i_k]} p_k^i$; {*Вычисление на GPU*}

Вектор p представляются в виде «граничных» и «внутренних» компонент $p^T = (p_1^i, p_1^b, \ldots, p_k^i, p_k^b, \ldots, p_{n_p}^i, p_{n_p}^b)$, так же, как векторы q, u, r, z.

Таблица 2 : Время решения СЛАУ [1] блочной схемой метода сопряжённых градиентов

Название теста	$n / \frac{nnz}{n}$	CPU	1 GPU	2 GPU	4 GPU	6 GPU	8 GPU
Inline_1	503712 / 73	1537.0	166.27	149.18	144.64	181.23	230.78
Fault_639	638802 / 44	299.5	40.07	38.06	37.93	46.14	61.06
Emilia_923	923136 / 44	706.5	94.67	77.26	69.98	80.85	101.87
Audikw_1	943695 / 82	846.5	85.33	62.33	51.29	55.84	67.64
Flan_1565	1564794 / 75	1675.5	144.60	97.70	71.31	73.16	82.51

[1] В вычислительных экспериментах использованы матрицы из «The University of Florida Sparse Matrix Collection» (`http://yifanhu.net/GALLERY/GRAPHS/search.html`).

Особенности послойного разделения для блочной схемы.

- На уровне сетки:
  - статическая балансировка в виде разделения сетки на слои ячеек, с последующим объединением слоев в подобласти без ветвления;
  - формирование блоков при конечно-элементной аппроксимации на полученных подобластях сетки.
- На уровне матрицы A:
  - в качестве первого слоя берутся вершины графа, соответствующие первой строке матрицы A;
  - условие соседства — существование вершин, связанных с текущим слоем.

- Параллельное формирование дополнение Шура:
  1. Параллельно в каждой i-ой подобласти:
     1. Ввычисляем матрицы $A'_{IB} = A_{II}^{-1} A_{IB}$, решая на GPU систему
     $$A_{II} A'_{IB} = A_{IB} \tag{1}$$
     2. Вычисляем матрицы S_{BB}, соответствующие подобласти i.
     $$S_{BB} = A_{BB} - A_{BI} A'_{IB}. \tag{2}$$
     3. Решаем на GPU систему:
     $$A_{II} f'_I = f_I. \tag{3}$$
     4. Вычисляем слагаемые вектора правых частей для границы
     $$\tilde{f}_B = f_B - A_{BI} f'_I. \tag{4}$$
  2. Формируем на CPU матрицу $\tilde{S}_{BB}$ в формате CSR и вектор правой части $\tilde{f}_B$
  $$\tilde{S}_{BB} = \sum_i S^i_{BB}, \quad \tilde{f}_B = \sum_i \tilde{f}^i_B. \tag{5}$$
- Параллельное решение:
  1. Решаем систему уравнений для границы на нескольких GPU
  $$\tilde{S}_{BB} \tilde{u}_B = \tilde{f}_B. \tag{6}$$
  2. Нахождение неизвестных для внутренних узлов
  $$u_I = f'_I - A'_{IB} \tilde{u}_B. \tag{7}$$

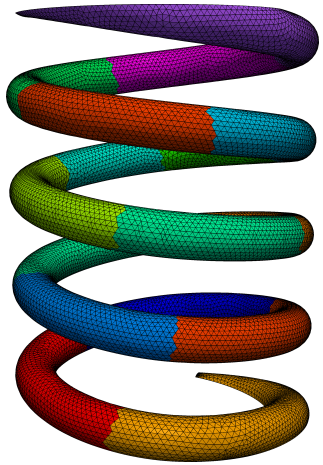


Рис. 3 : Сетка из $N_{fe} = 174264$ тетраэдров, $n_\Omega = 16$.

Особенности геометрии:

- Развитая граничная поверхность.
- Преобладание одного направления, не совпадающего с координатными.
- Существуют конечные элементы, все узлы которых принадлежат границе.
- Неравномерность по шагу сетки и валентности узлов.

Таблица 3 : Время формирование дополнения Шура, сек.

n_Ω	CPU	1 GPU	2 GPU	4 GPU	6 GPU	8 GPU
16	1583.6	1912.6	1165.5	789.9	657.6	589.4
32	480.6	1173.9	661.8	401.7	314.0	266.5
64	145.1	678.8	378.2	224.8	174.7	149.0
128	—	—	237.2	145.8	118.5	102.5
256	—	—	194.0	121.0	97.7	86.0
512	—	—	168.3	105.7	86.0	75.4
1024	18.7	233.4	144.0	92.2	77.5	68.0

Таблица 4 : Время решения системы $\tilde{S}_{BB}\tilde{u}_B = \tilde{f}_B$, сек.

n_Ω	CPU	1 GPU	2 GPU	4 GPU	6 GPU	8 GPU
16	$>$ 28800	912	421	205	281	264
32	$>$ 28800	983	630	430	317	274
64	18067	287	174	98	82	72
128	—	—	127	78	66	60
256	—	—	106	70	61	56
512	—	—	85	63	56	53
1024	6502	69	76	59	54	52

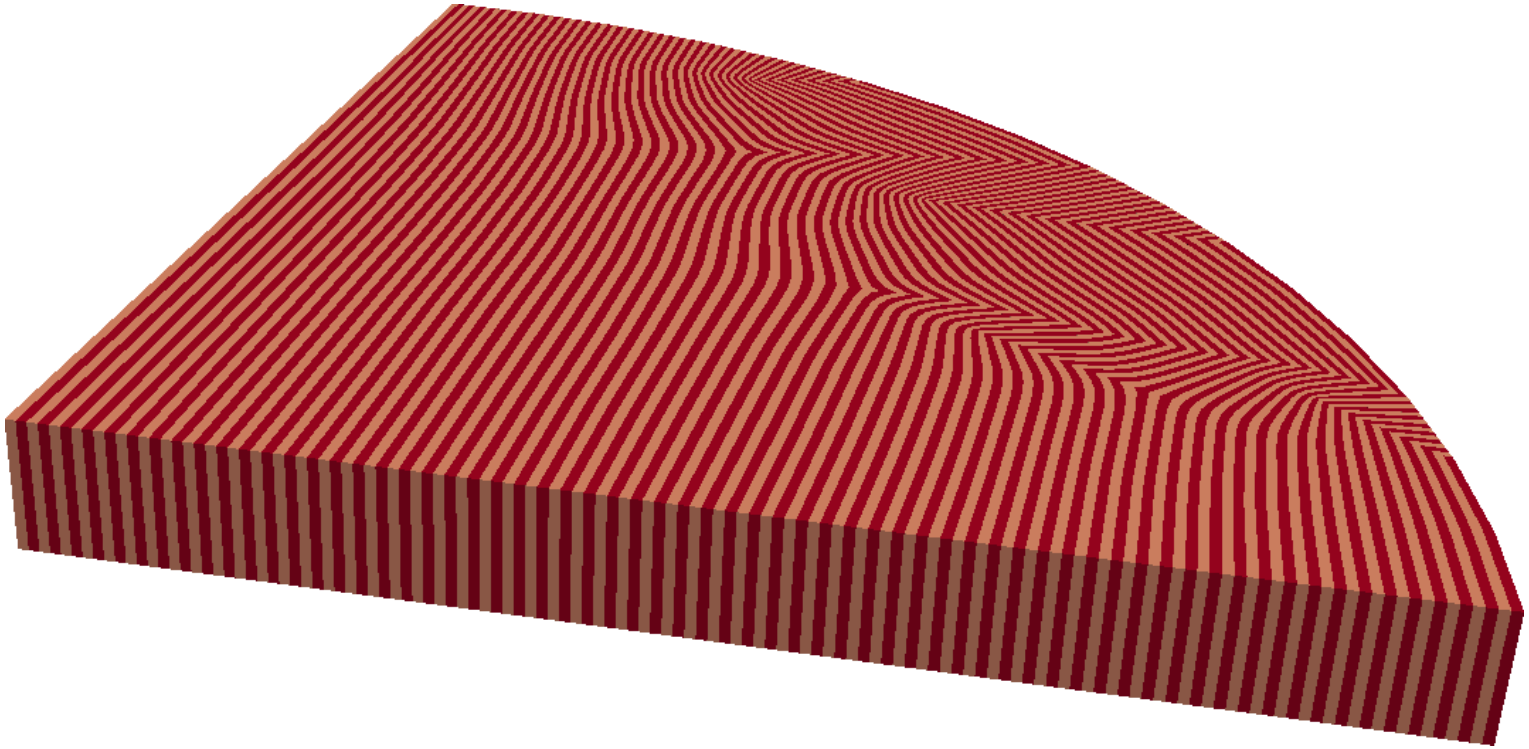


Рис. 4 : Множество слоев, $n_s = 128$.

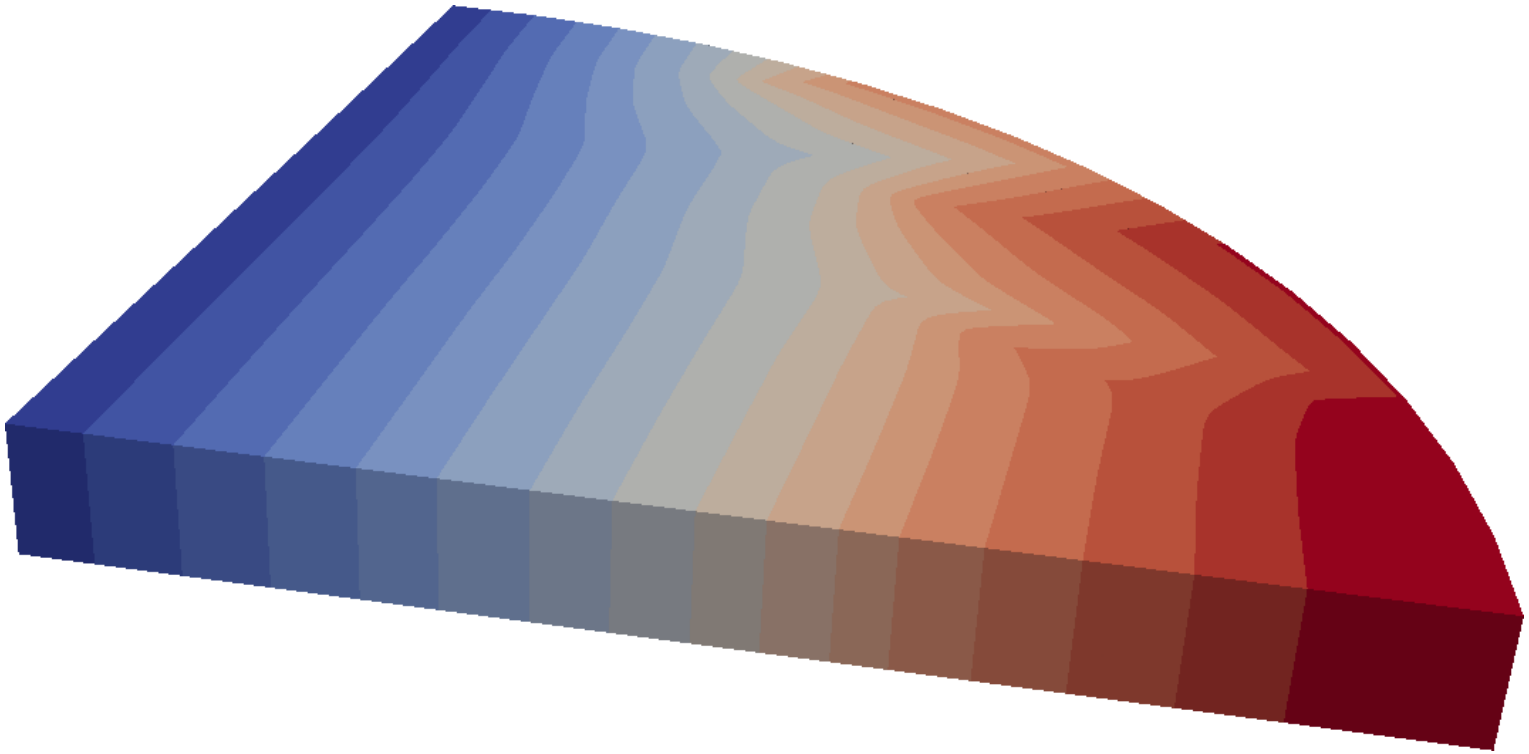


Рис. 4 : Слои объединены в 16 блоков $\omega_i, i = 1, n_\omega, n_\omega = 16$

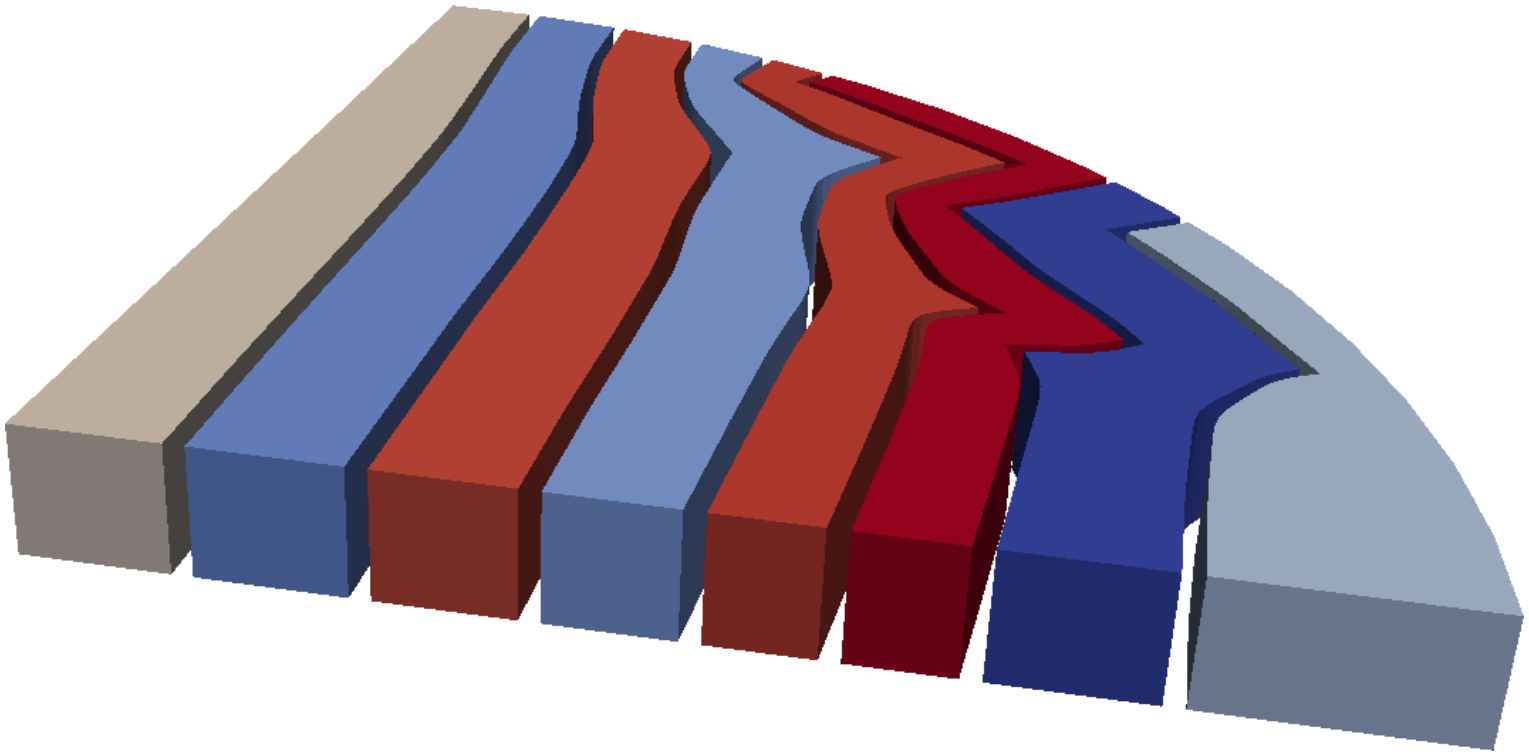


Рис. 4 : 4 уровень (8 подобластей $\Omega_j, j = 1, n_\Omega, n_\Omega = n_\omega/2 = 8$)

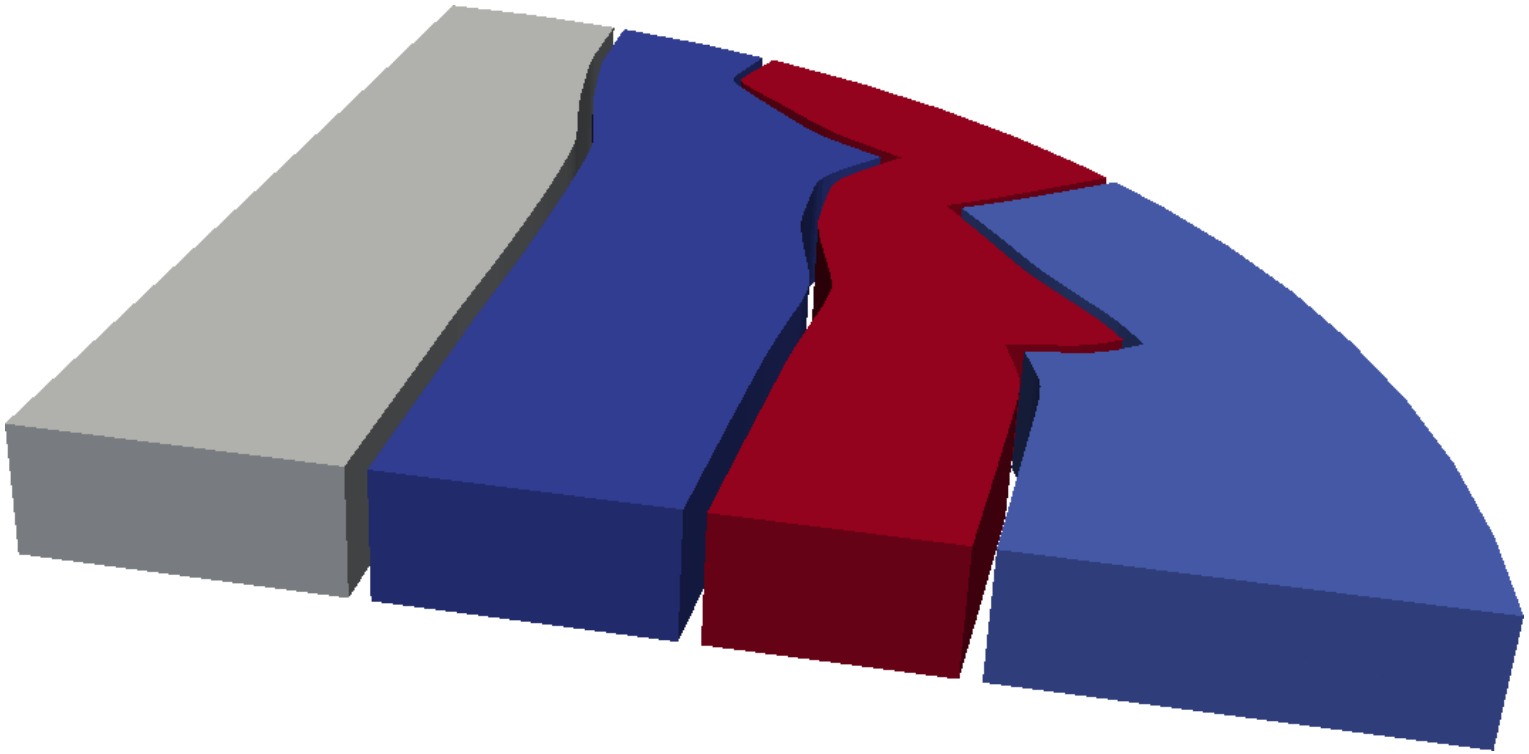


Рис. 4 : 3 уровень, $n_\Omega = 4$

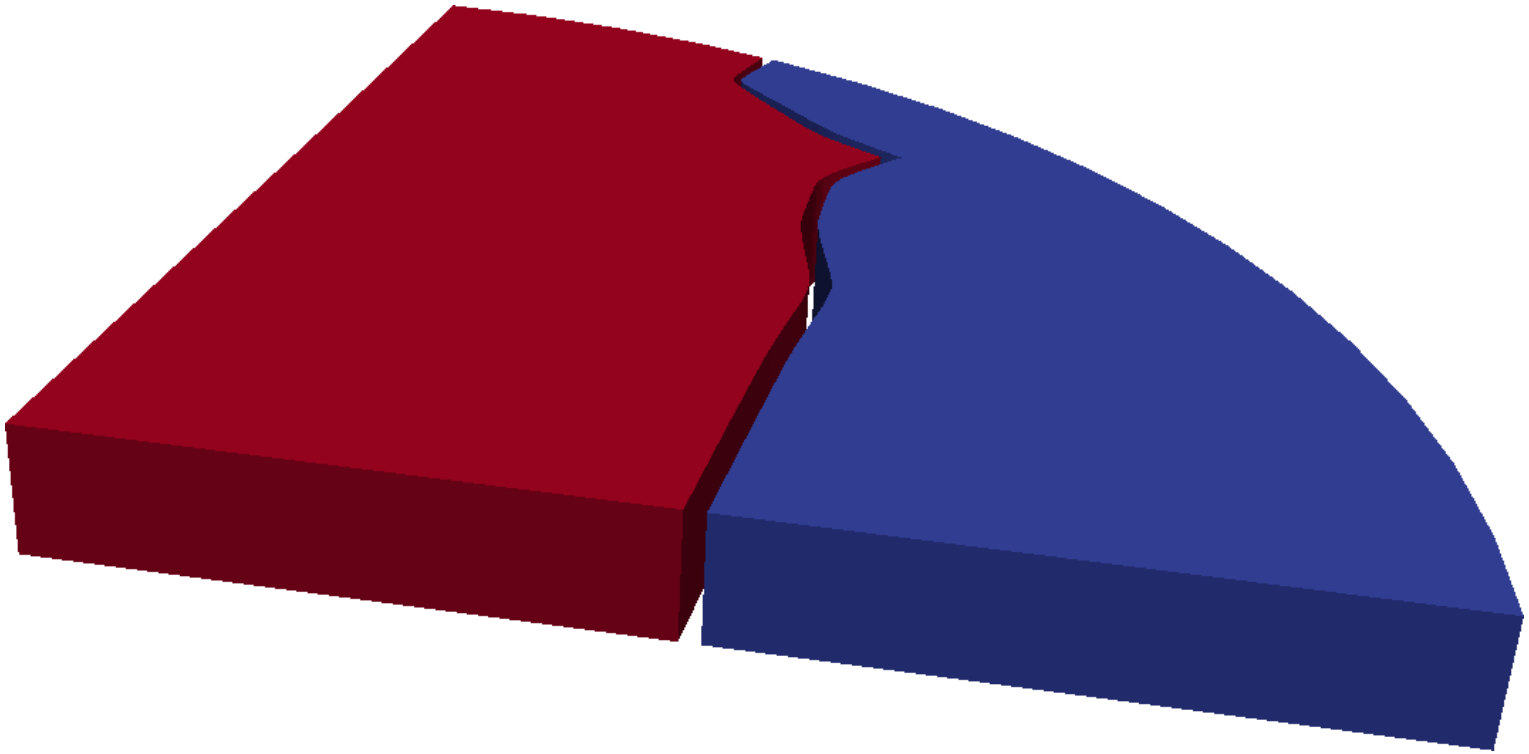


Рис. 4 : 2 уровень, $n_\Omega = 2$

- В поэлементной схеме благодаря послойному разделению (разделение без ветвления) были исключены конфликты в общей памяти при параллельной сборке вектора результата матрично-векторного произведения.
- При использовании блочного метода сопряженных градиентов представляется перспективным наследование послойного разделения сетки для разделения матрицы на блоки на этапе конечно-элементной аппроксимации.
- Применение послойного разделения сетки в методе дополнения Шура позволило: в полтора раза сократить время формирования матриц S^i_{BB} за счет перераспределения размеров матриц (уменьшения A^i_{II}, и увеличения A^i_{BI} и A^i_{IB}) и локализации доступа к данным.
- Использование блочного метода решения, позволило снизить ограничения на размер систем, решаемых на GPU.

Дальнейшие исследования:

- Объединение слоев в многосвязных сеточных областях.
- Наследование разделение сетки без ветвления для формирования блоков матрицы коэффициентов системы уравнений.
- Построение иерархического метода дополнения Шура на основе разделения по слоям.
- Послойное упорядочения в сеточных подобластях, полученных другими алгоритмами.

Работа выполнена при частичной финансовой поддержке РФФИ, гранты №17–01–00402-а, 16–01–00129-а.