1 Умножение матриц по Винограду

Наивный алгоритм перемножения матриц требует $O\left(n^3\right)$ операций. Алгоритм Штрассена улучшает асимптотику до $O\left(n^{2.807355}\right)$. Алгоритм Коммерсмита-Винограда работает за $O\left(n^{2.375477}\right)$, но на практике не используется из-за очень большой скрытой константы. На лекциях рассказывался небольшой лайфхак, который все равно работает за куб, но немного по другому.

В обычном алгоритме перемножения матриц элемент считается как

$$c_{ij} = \begin{pmatrix} a_{i1} & a_{i2} & \dots & a_{iN} \end{pmatrix} \begin{pmatrix} b_{1i} \\ b_{2i} \\ \dots \\ b_{Nj} \end{pmatrix} = \sum_{k=1}^{N} a_{ik} b_{kj}, \tag{1.1}$$

То есть всего мы сделаем N^3 умножений и $N^3 - N^2$ сложений.

Однако можно заметить, что

$$c_{ij} = \sum_{k=1}^{N/2} \left(a_{i,2k-1} + b_{2k,j}\right)\left(a_{i,2k} + b_{2k-1,j}\right) - \sum_{k=1}^{N/2} a_{i,2k-1} a_{i,2k} - \sum_{k=1}^{N/2} b_{2k-1,j} b_{2k,j}. \tag{1.2}$$

Второе и третье слагаемое можно предпосчитать и потом много раз использовать. То есть всего умножений и сложений будет

$$\frac{1}{2} N^3 + N^2 \tag{1.3}$$

и

$$\frac{3}{2} N^3 + 2N^2 - 2N \tag{1.4}$$

соответственно (конкретные чиселки зависят от того, что мы считаем сложением, лучше в коде везде раскидать комментарии, сколько где производится операций). Это немного веселее, чем в обычном алгоритме, потому что сложение дешевле, чем умножение. Приведем сам код

```
// предпосчет второго слагаемого
for i = 1:N
  row(i) = A(i, 1) * A(i, 2)
  for k = 2:N/2
    row(i) = row(i) + A(i, 2*k-1) * A(i, 2*k)
  end
end

// предпосчет третьего слагаемого
for j = 1:N
  col(j) = B(1, j) * B(2, j)
  for k = 2:N/2
    col(j) = col(k) + B(2*k-1, j) * B(2*k, j)
  end
end

// само перемножение
for i = 1:N
  for j = 1:N
    C(i, j) = -row(i) - col(j)
    for k = 1:N/2
      C(i, j) = C(i, j) + (A(i, 2 * k - 1) + B(2 * k, j)) *
(A(i, 2 * k) + B(2 * k - 1, j))
    end
  end
end
// если размерность матрицы нечетная, то надо не забыть
последние столбец/строку
if N % 2 == 1 then
  for i = 1:N
    for j = 1:N
      C(i, j) = C(i, j) + A(i, N) * B(N, j)
    end
  end
end
```

2 Умножение матриц по Штрассену

$C = AB, \quad A, B, C \in \mathbb{R}^{2^n \times 2^n}$

Если размер не степень двойки, то можно дополнить нулевыми строками/столбцами.

Разделим каждую из матриц на 4 одинаковых блока

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \tag{1.5}$$

где $A_{ij}, B_{ij}, C_{ij} \in \mathbb{R}^{2^{n-1} \times 2^{n-1}}$

Тогда C выражается как

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned} \tag{1.6}$$

Как и в обычном методе, требуется 8 умножений. Однако можно перевыразить все по-другому, определив сначала новые элементы:

$$\begin{aligned} P_1 &= \left(A_{11} + A_{22}\right)\left(B_{11} + B_{22}\right) \\ P_2 &= \left(A_{21} + A_{22}\right)B_{11} \\ P_3 &= A_{11}\left(B_{12} + B_{22}\right) \\ P_4 &= A_{22}\left(B_{21} - B_{11}\right) \\ P_5 &= \left(A_{11} + A_{12}\right)B_{22} \\ P_6 &= \left(A_{21} - A_{11}\right)\left(B_{11} + B_{12}\right) \\ P_7 &= \left(A_{12} - A_{22}\right)\left(B_{21} + B_{22}\right) \end{aligned} \tag{1.7}$$

А потом выразив

$$
\begin{aligned}
C_{11} &= P_1 + P_4 - P_5 + P_7 \\
C_{12} &= P_3 + P_5 \\
C_{21} &= P_2 + P_4 \\
C_{22} &= P_1 - P_2 + P_3 + P_6
\end{aligned}
\tag{1.8}
$$

Заметим, что таким образом нам требуется 7 умножений, а не 8. Более того, такой подход позволяет рекурсивно считать подматрицы, что хорошо прогается. Обычно такой подход продолжают до размеров матриц около 32-128, а потом используют наивный алгоритм, потому что данный метод теряет свою эффективность из-за бОльшего числа сложений, чем обычный.

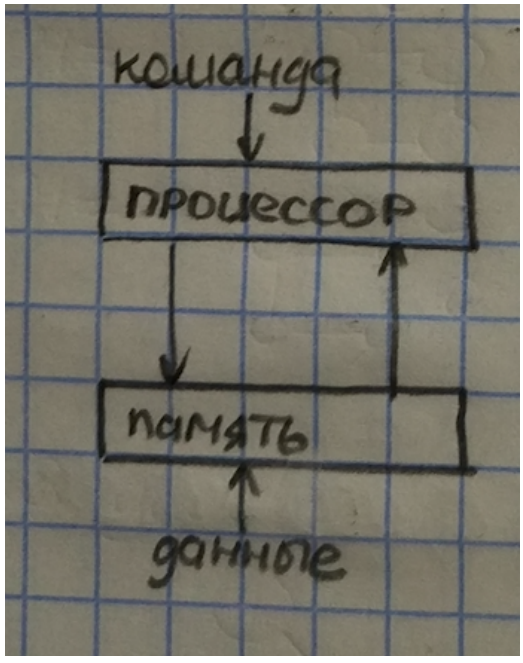
3 Классификация параллельных вычислительных систем по Флинну, модели представления параллельных алгоритмов

Вычисления **синхронные**, если вычислительный процесс определяется потоком команд.

Вычисления **асинхронные**, если вычислительный процесс определяется потоком данных.

	Один поток данных	Много потоков данных
Один поток команд	SISD Модель Неймана	SIMD Векторный процессор
Много потоков команд	MISD Конвейерный процессор	MIMD Многопроцессорная ЭВМ

3.1 SISD



Модель фон Неймана:

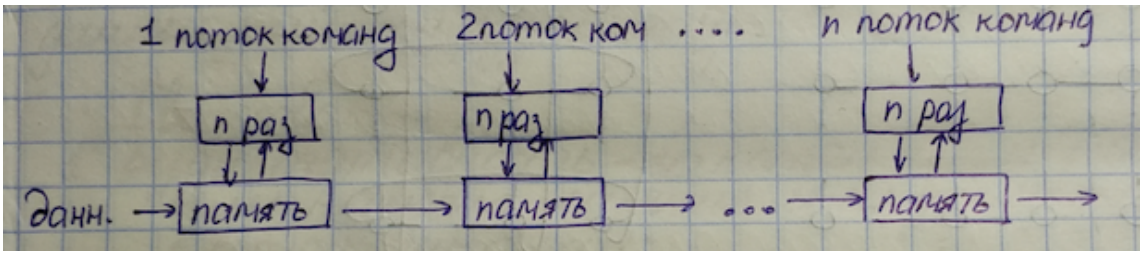
- Действия последовательны
- Данные обрабатываются последовательно

3.2 SIMD

Векторные ЭВМ

- В N раз быстрее
- Ограничение: необходимость векторизации численного метода

3.3 MISD

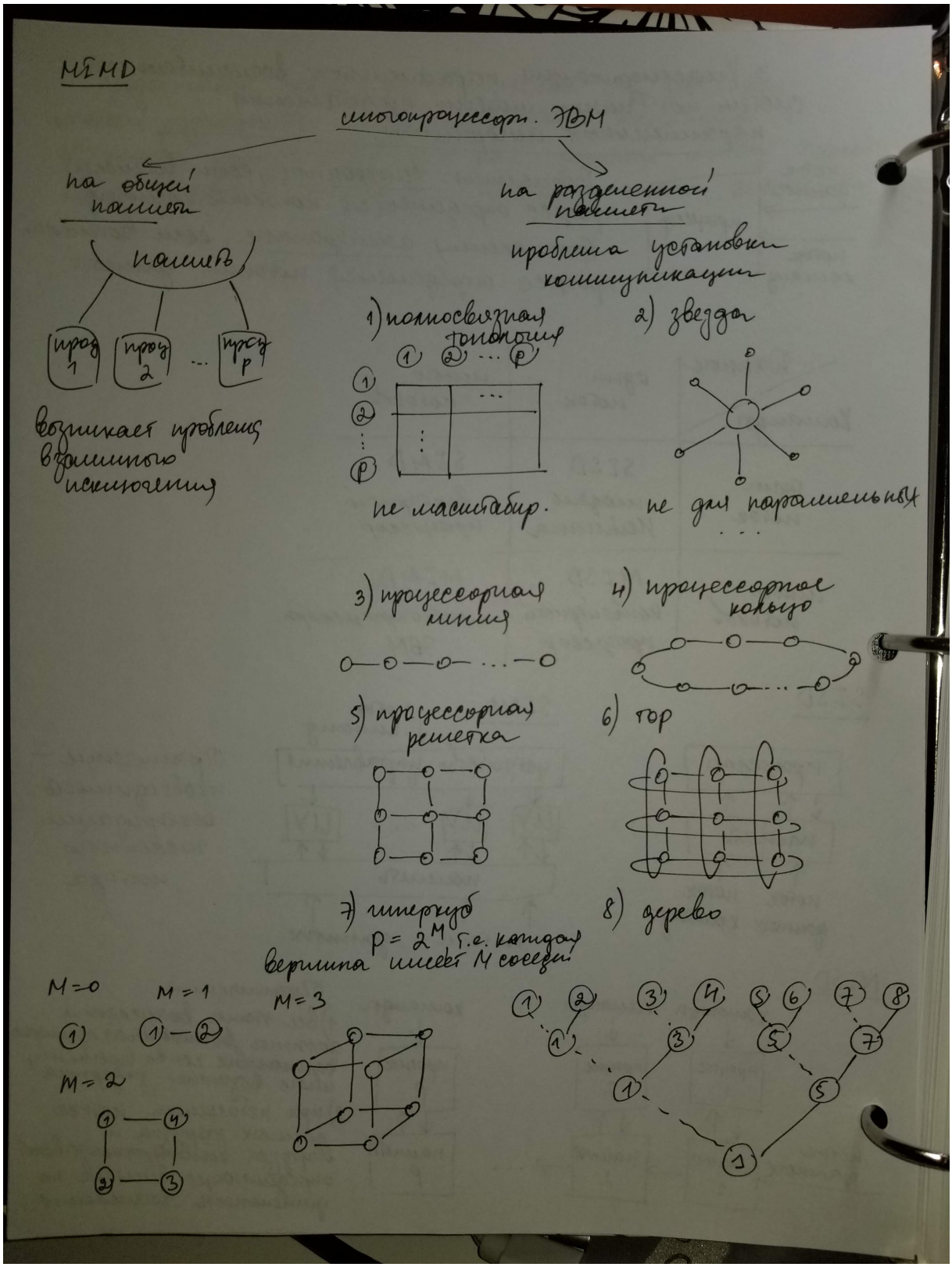


Конвейерный процессор. Конвейерный алгоритм предполагает разбиение задачи на подзадачи.

Ограничения:

- Все этапы вычисления должны выполняться примерно одинаковое количество времени, иначе возникнет задержка.
- При небольшом потоке данных загрузка и выгрузка конвейера оказывают определяющее влияние на длительность вычисления.

3.4 MIMD


MIMD
многопроцессор. ЭВМ
на общей памяти
на разделенной памяти
проблема установки коммуникации
память
возникает проблема взаимного исключения
1) полносвязная топология
2) звезда
не масштабир.
не для параллельных
3) процессорная линия
4) процессорное кольцо
5) процессорная решетка
6) тор
7) гиперкуб
p = 2^M т.е. каждая вершина имеет M соседей
8) дерево
M=0
M=1
M=2
M=3

3.5 Модели представления параллельных алгоритмов

Модель задача/канал

Задача — последовательный алгоритм и данные, необходимые для его реализации (кружочек)

Канал — указывает на коммуникации между задачами (однонаправленная стрелочка)

Коммуникация организуется по принципу «первый зашел, первый вышел». Во время вычислений задачи могут возникать и упраздняться.

Модель параллелизма данных

Ориентирована на производство векторных и матричных вычислений. Ее формализм совпадает с языком Matlab.

Модель общей памяти

Предыдущая модель дополняется операторами записи и чтения общей памяти, а также механизмами решения проблемы взаимного исключения (дедлок?).

4 Модели вычислительных процессов