Параллельные алгоритмы 20240220

Базовые параллельные алгоритмы

Якобовский Михаил Владимирович

Метод коллективного решения

Решение множества независимых друг от друга заданий, в таких прикладных областях как:

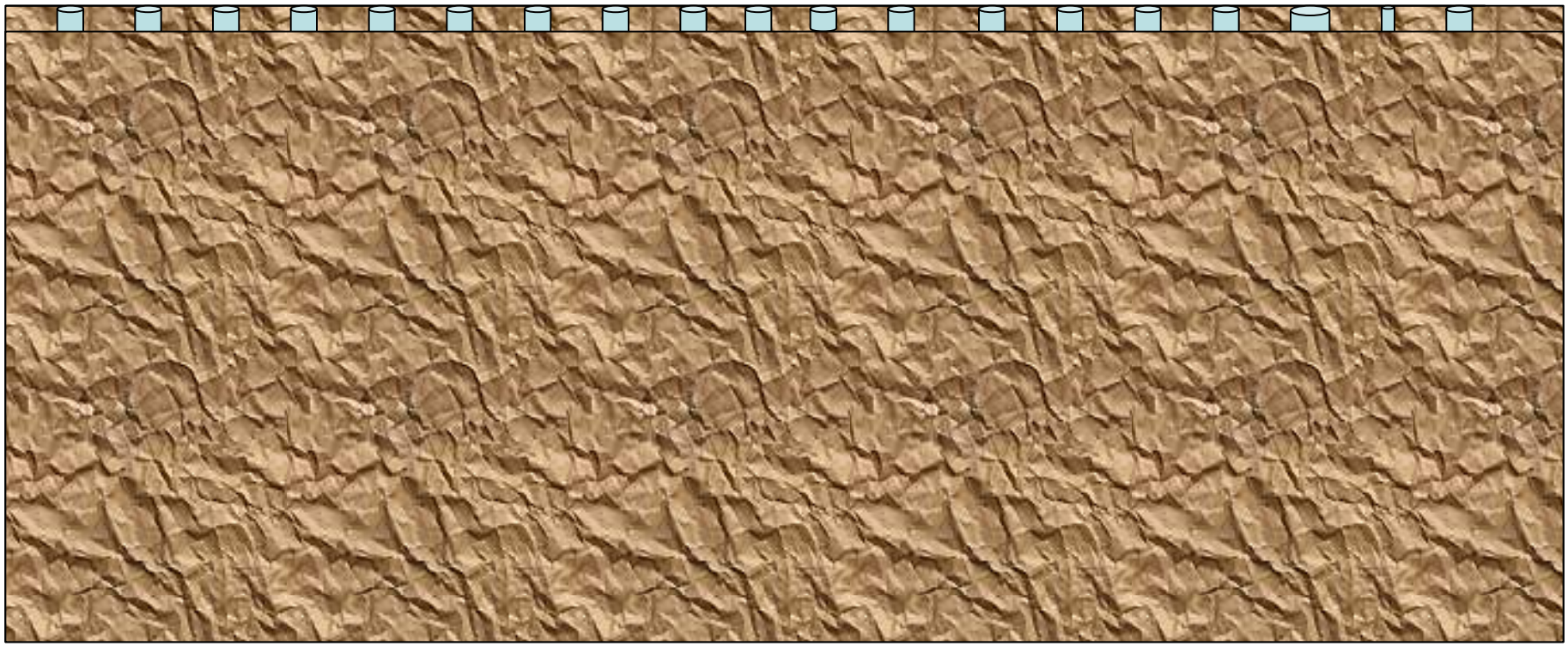
- Табулирование функций
- Решение задач методами Монте-Карло
- Численное интегрирование гладких многомерных функций
- …

Найти самую глубокую скважину







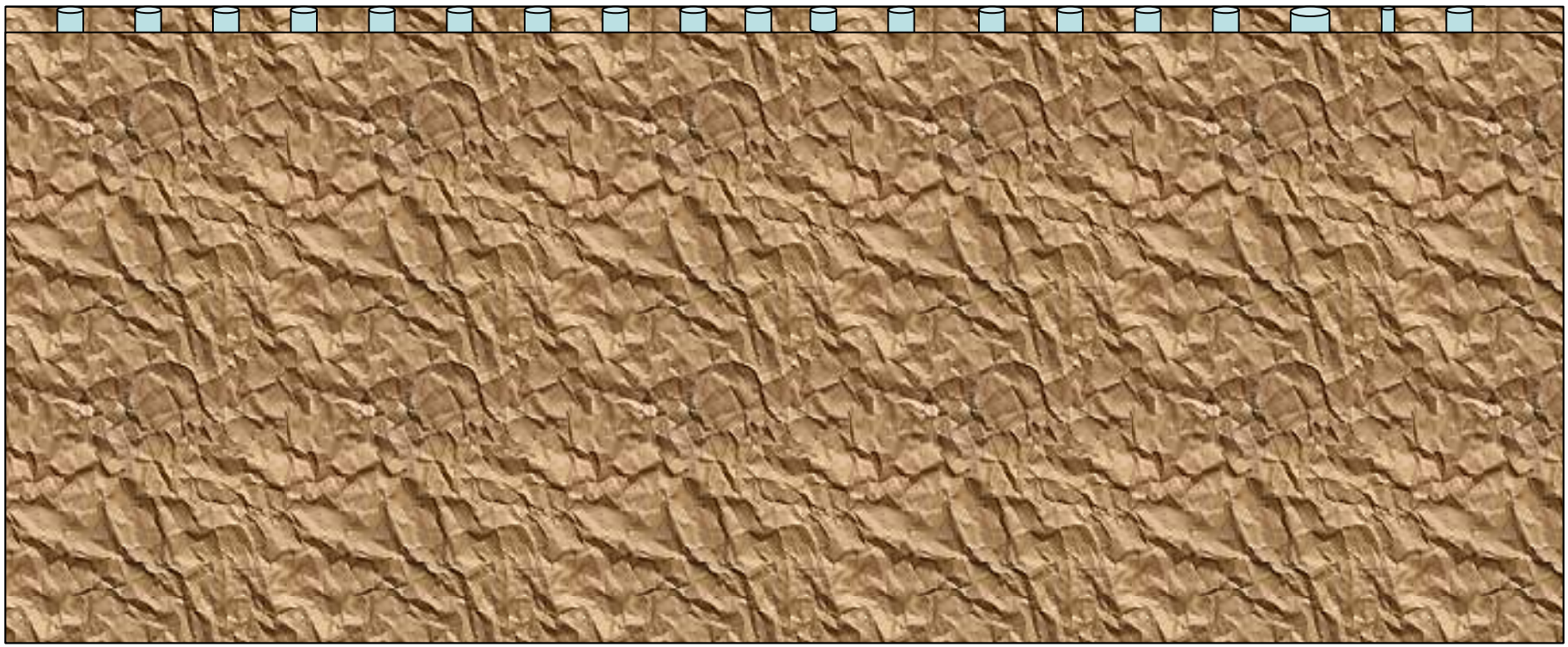


Найти самую глубокую скважину







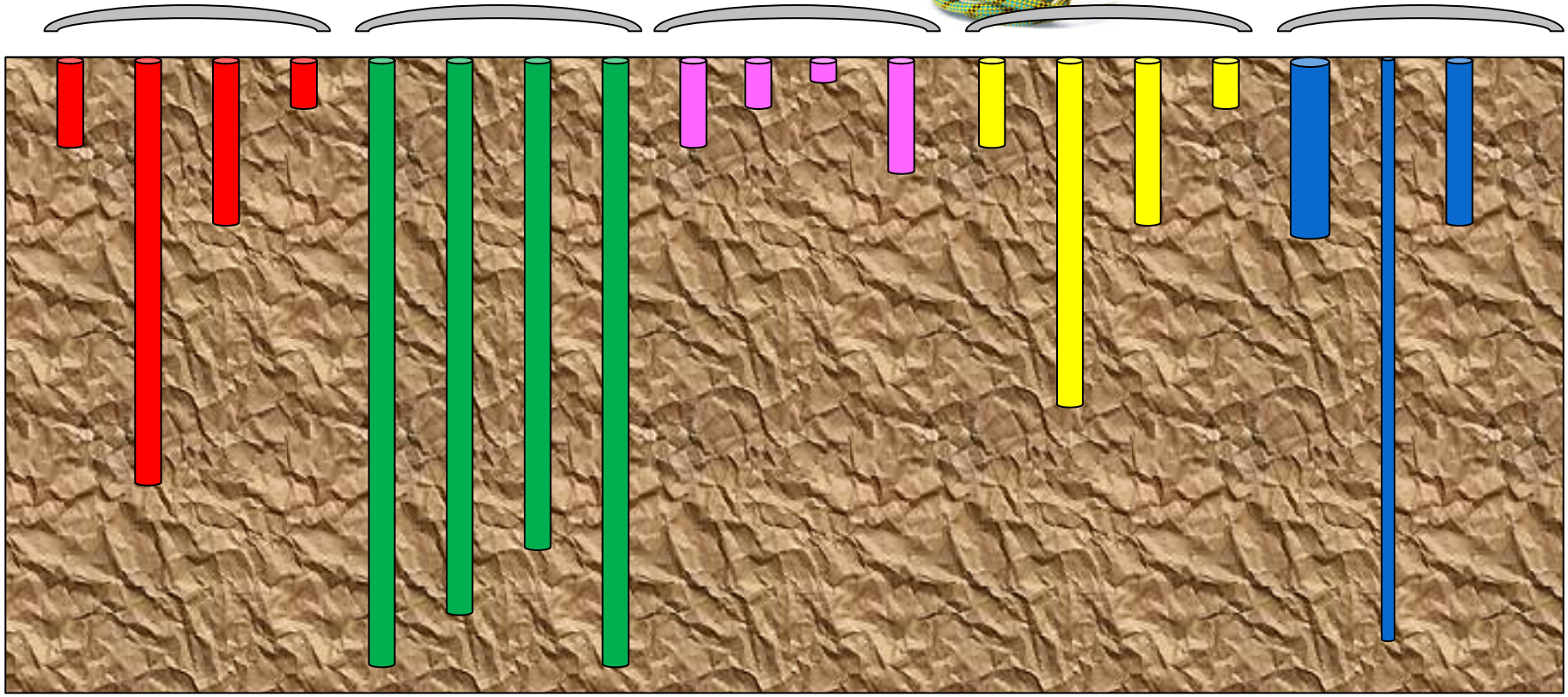


Найти самую глубокую скважину







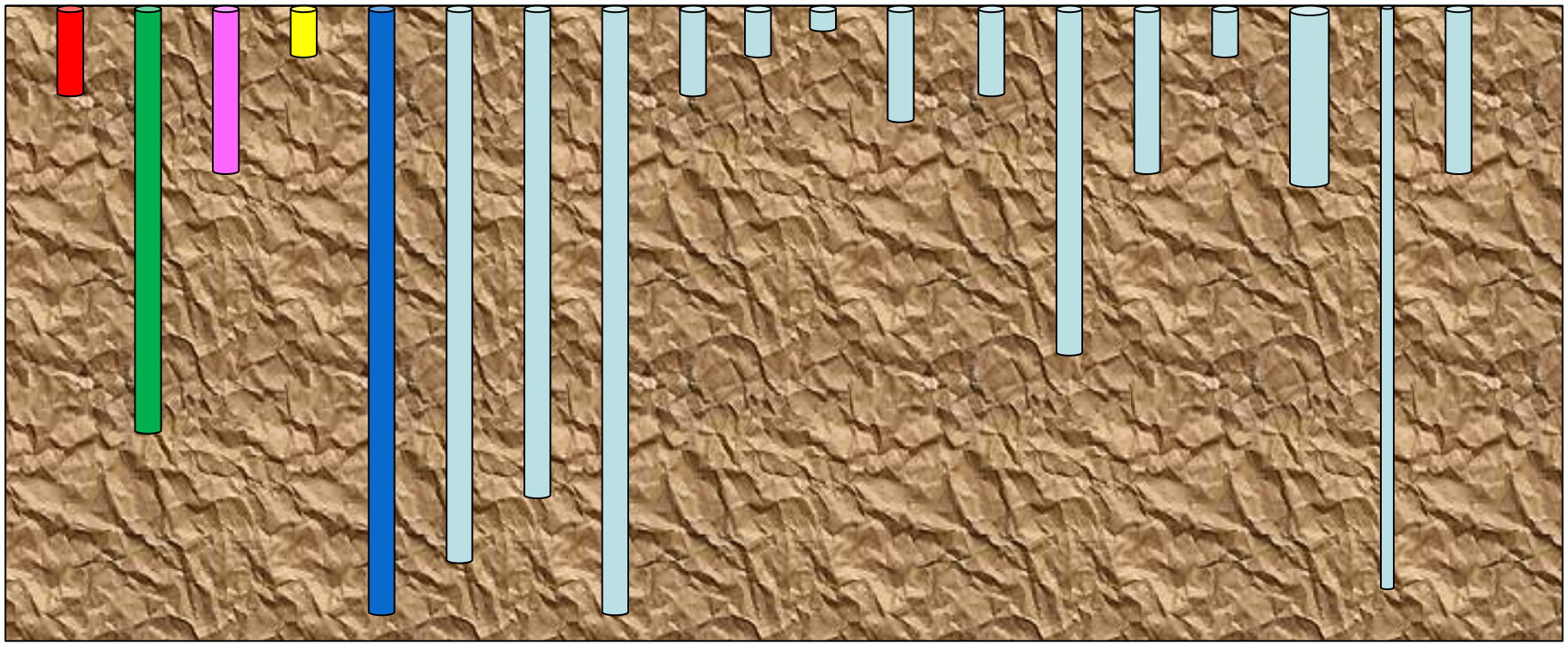


Найти самую глубокую скважину







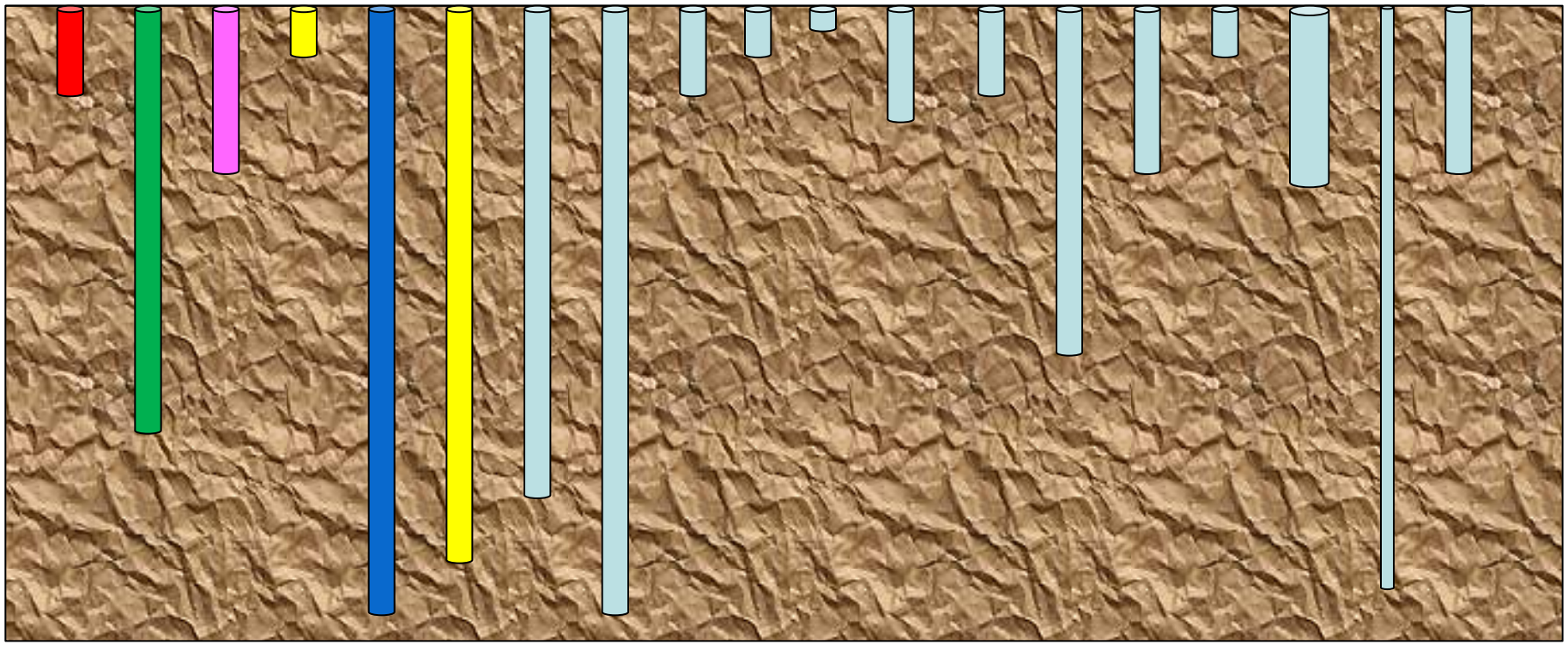


Найти самую глубокую скважину







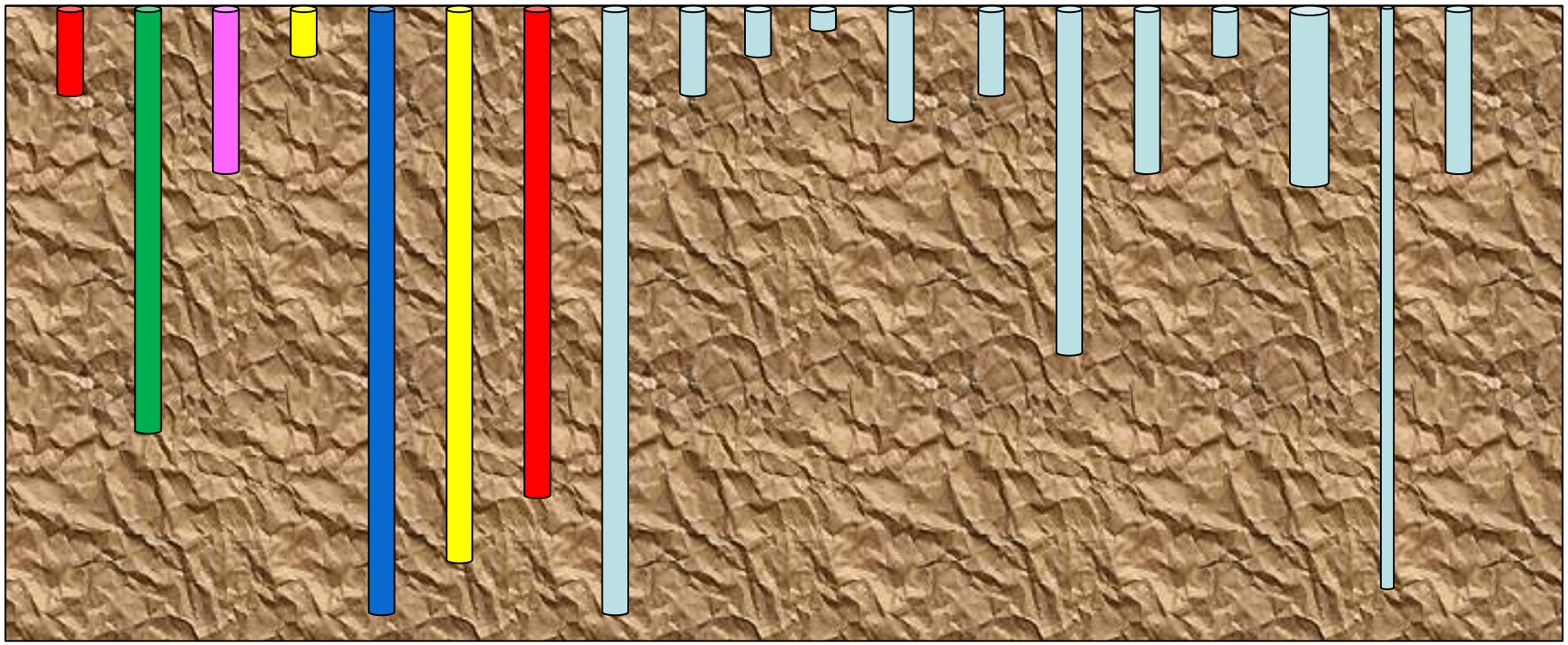


Найти самую глубокую скважину







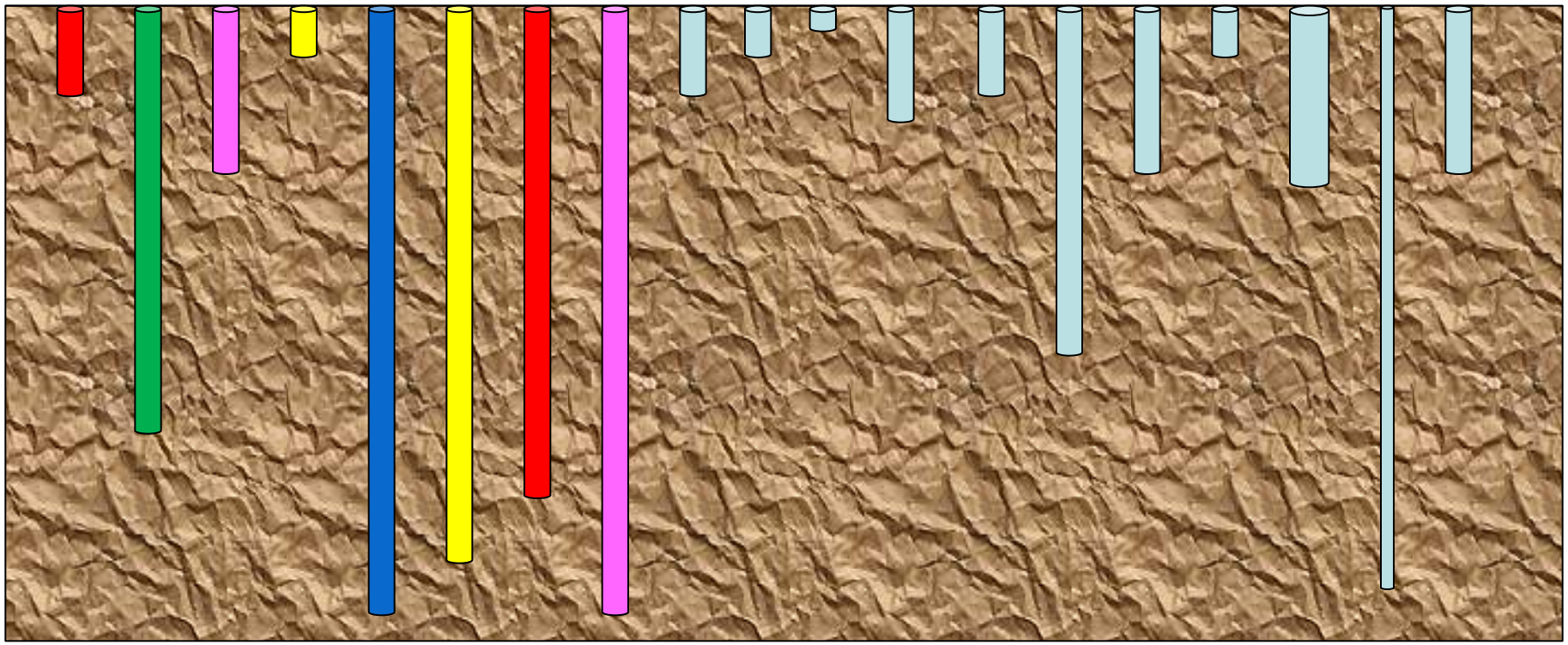


Найти самую глубокую скважину







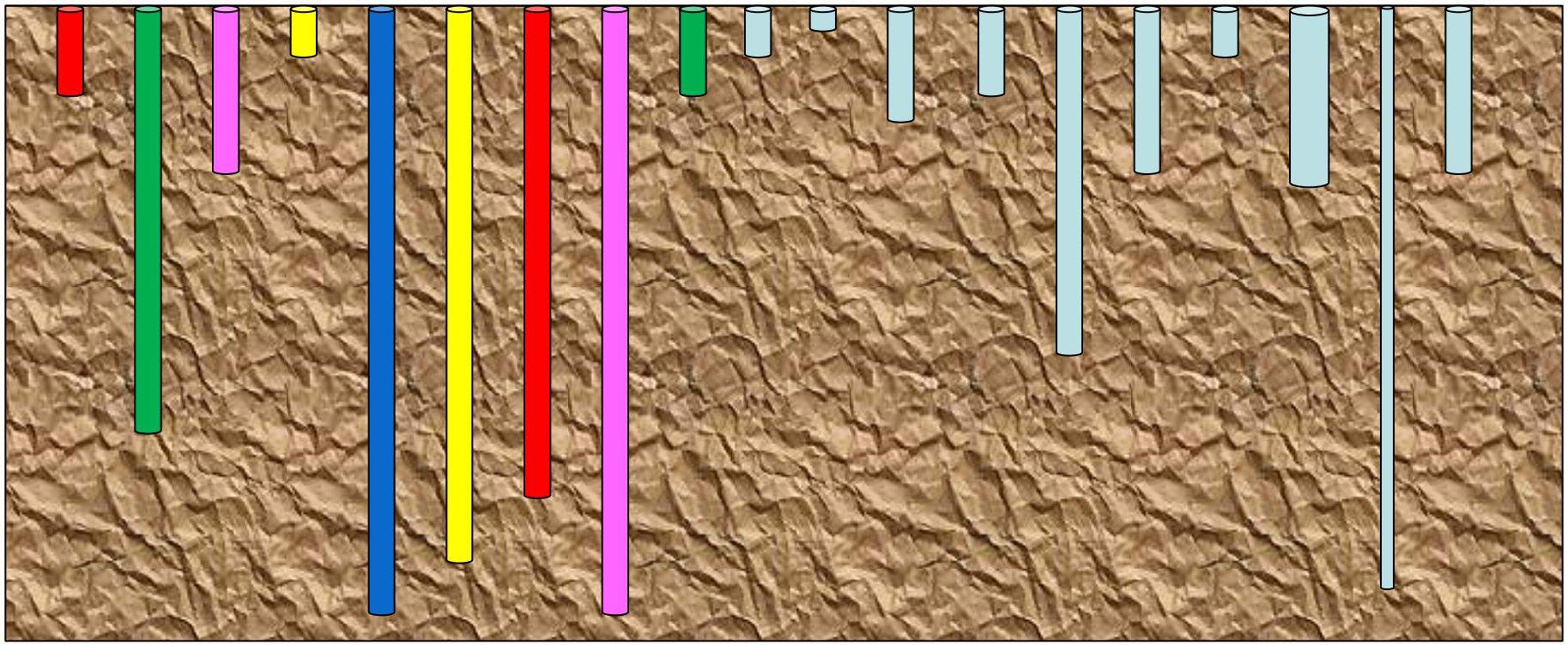


Найти самую глубокую скважину







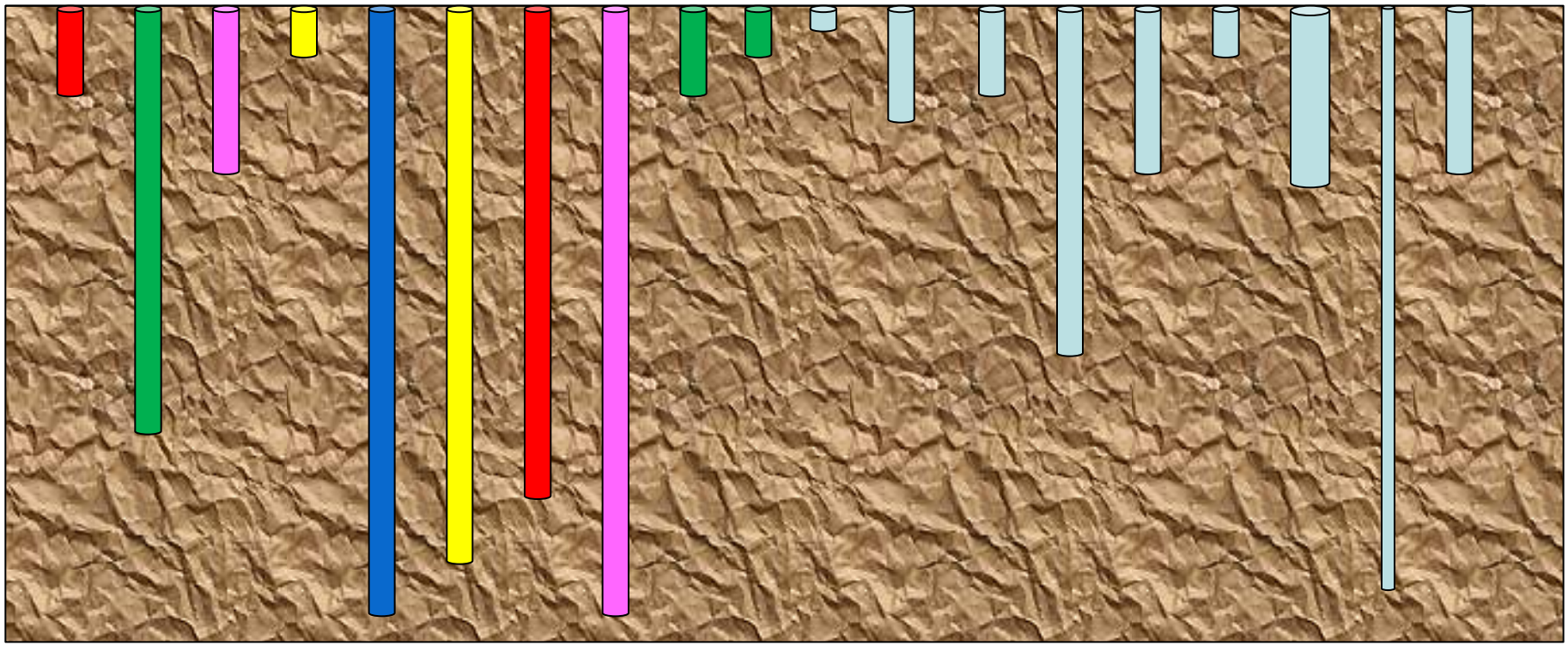


Найти самую глубокую скважину







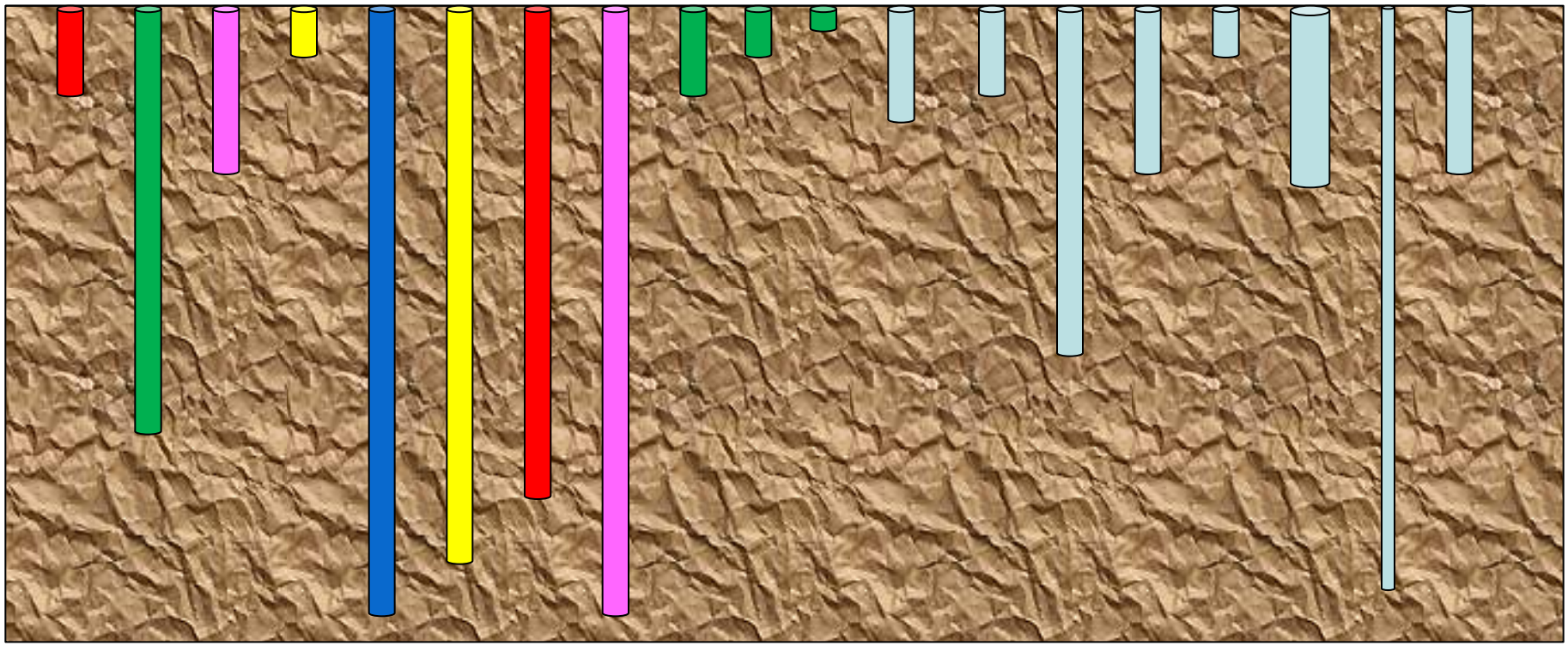


Найти самую глубокую скважину







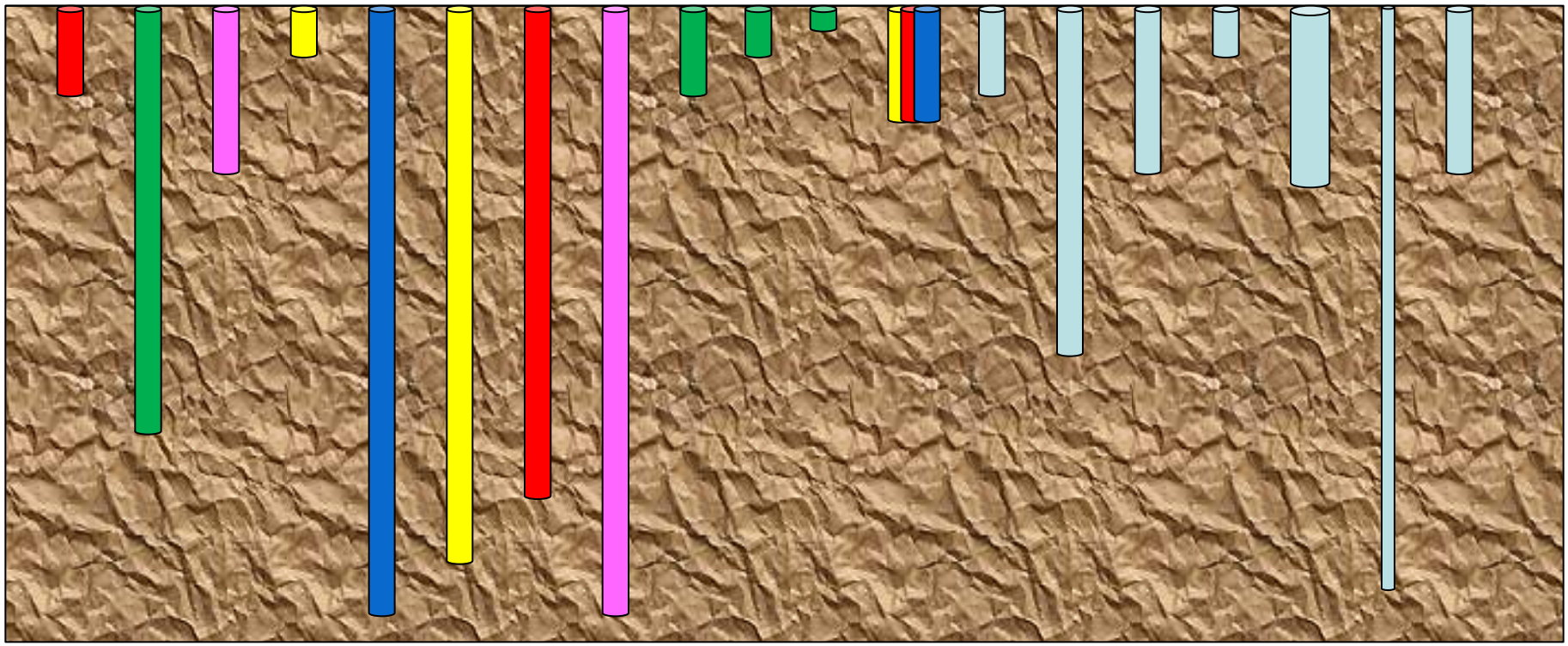


Найти самую глубокую скважину



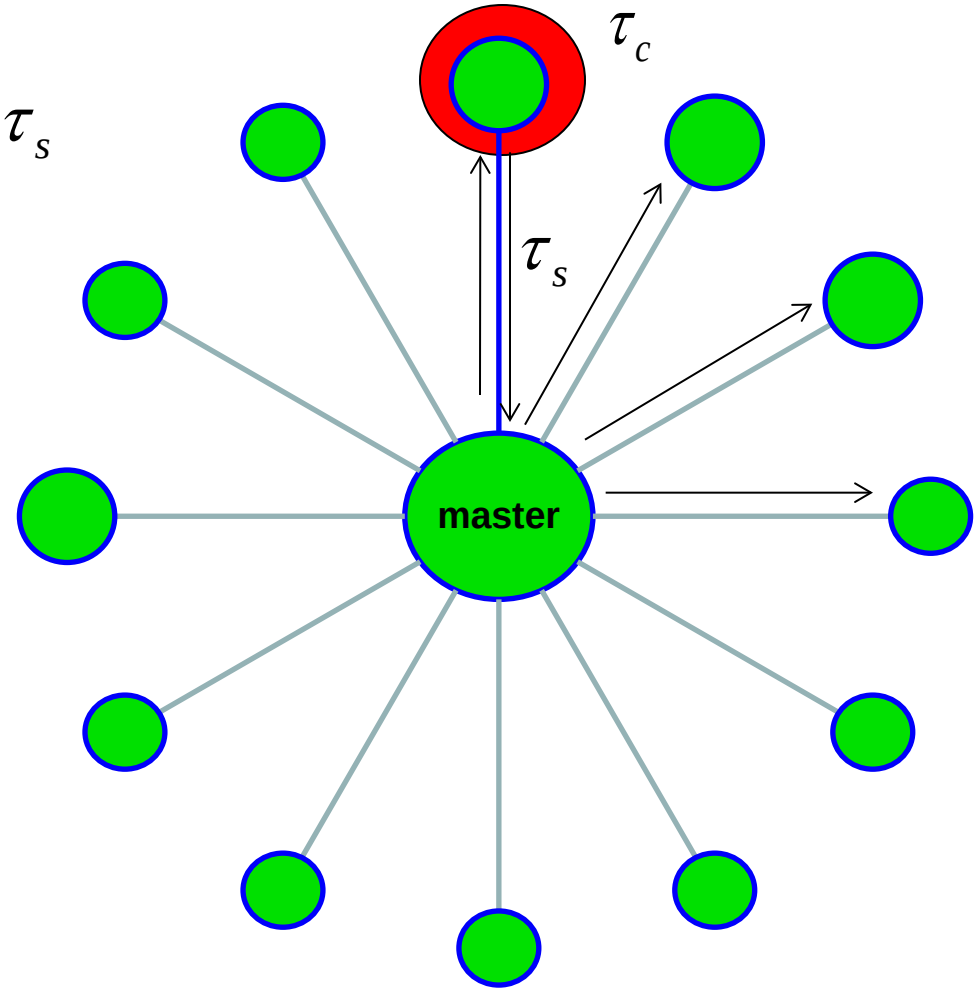






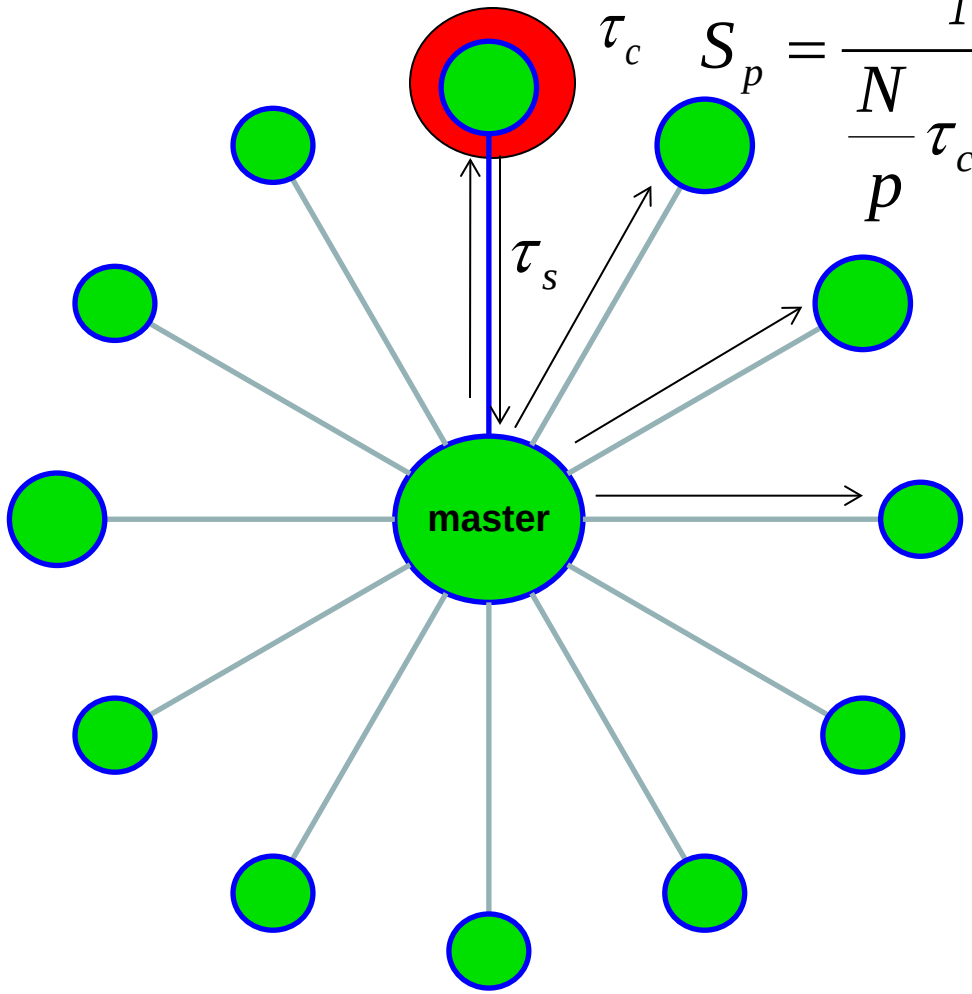
Метод коллективного решения

$$T_p = \frac{N}{p}\tau_c + N\tau_s$$



Метод коллективного решения

$$T_p = \frac{N}{p}\tau_c + N\tau_s$$



$$S_p = \frac{N\tau_c}{\frac{N}{p}\tau_c + N\tau_s} = p\frac{1}{1+p\frac{\tau_s}{\tau_c}}$$

$$p_{\max} = \frac{\tau_c}{\tau_s}$$

Пример модельной задачи: Укладка паркета

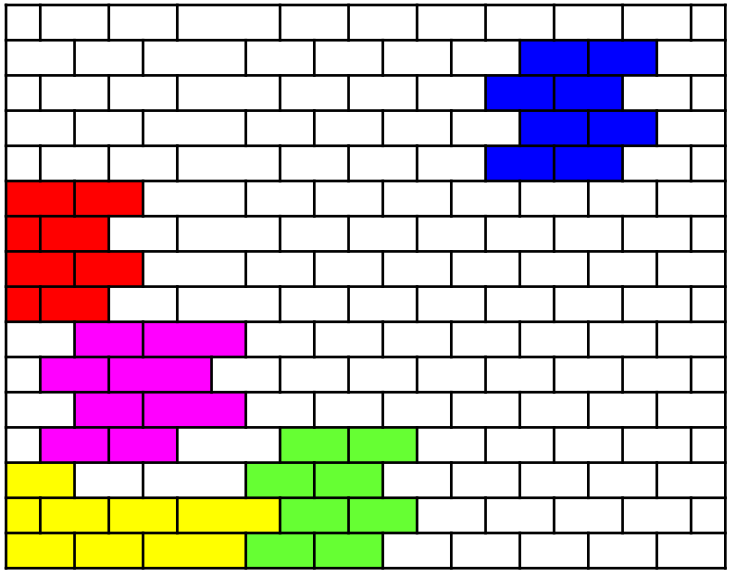
$$T_p = \frac{N}{p}\tau_c + N\tau_s$$

$$T_1(kn) = \tau_c kn$$

$$T_p(kn) = \frac{kn}{p}\tau_c + \frac{kn}{r}\tau_s = \frac{kn}{r}\left(\frac{r\tau_c}{p} + \tau_s\right)$$

$$S_{p=\frac{r\tau_c}{\tau_s}}(kn) = \frac{\tau_c kn}{\frac{kn}{r}\left(\frac{r}{p}\tau_c + \tau_s\right)} = \frac{r\tau_c}{2\tau_s} = \frac{p}{2}$$

$$S_{p=\frac{r\tau_c}{\tau_s}}(kn) == \frac{p}{2} \qquad E_{p=\frac{r\tau_c}{\tau_s}}(kn) = \frac{1}{2}$$



$$p_{\max} = \frac{r\tau_c}{\tau_s}$$

r – размер порции

Как правильно?

r – размер порции

$$p_{\max} = \frac{r\tau_c}{\tau_s} \quad \textbf{\textit{или}} \quad p_{\max} = \frac{r\tau_c}{r\tau_s} = \frac{\tau_c}{\tau_s}$$

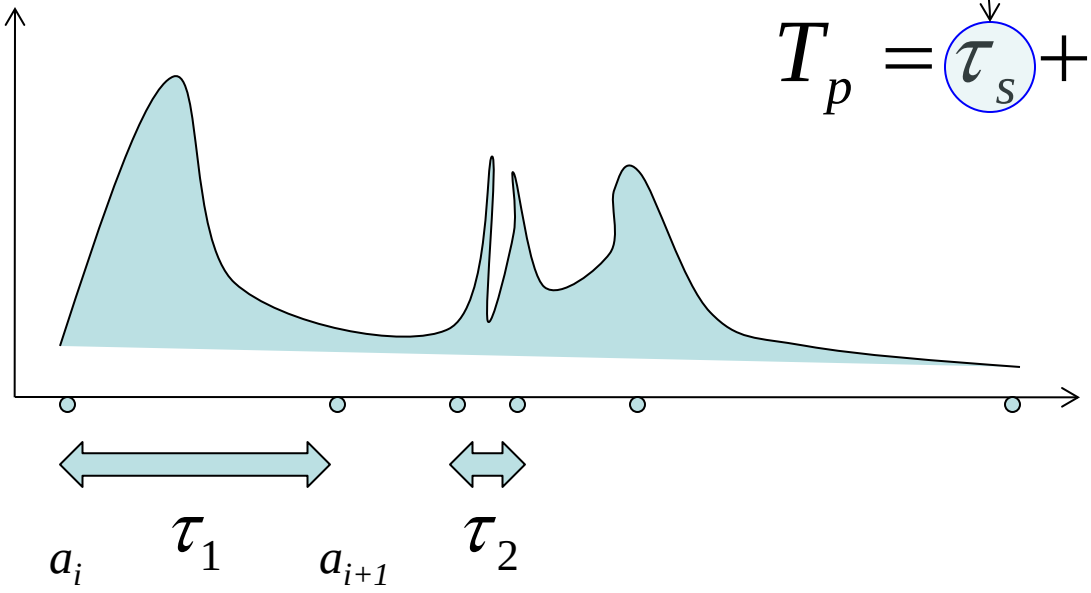
Зависит ли время передачи данных от размера порции (задания)?

Вычисление определенного интеграла

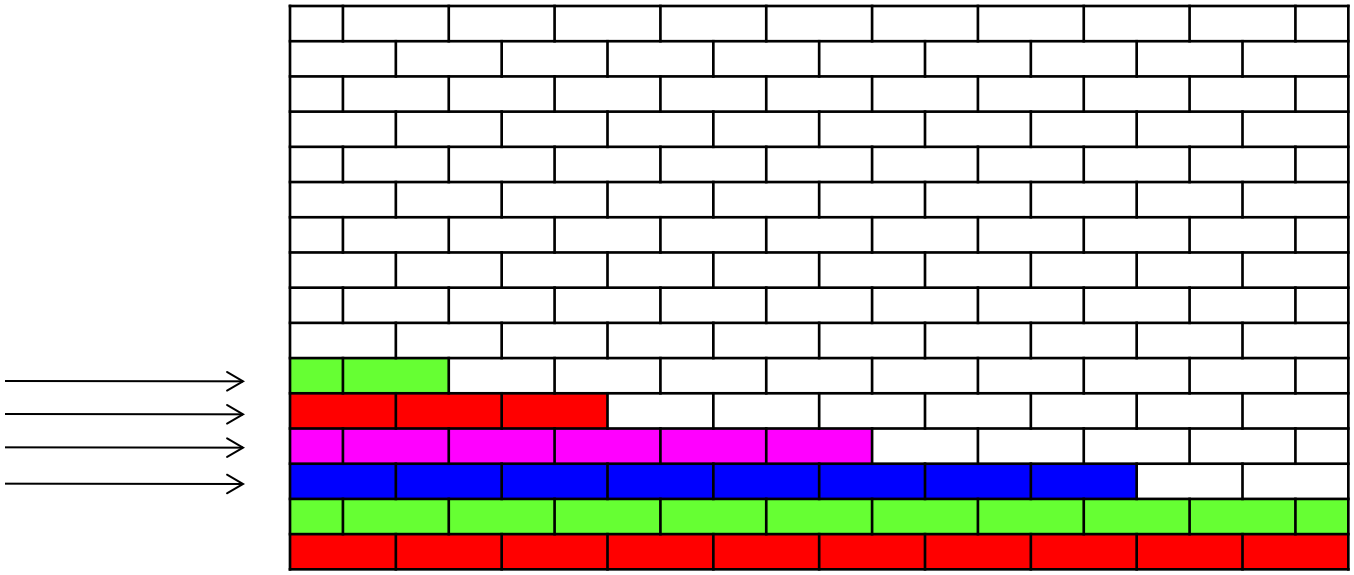
Send(a_i); Send(a_{i+1}); Recv(s);

$$I = \int_A^B f(x)dx = \sum_i \int_{a_i}^{a_{i+1}} f(x)dx$$

$$T_p = \tau_s + \max_i \tau_i$$

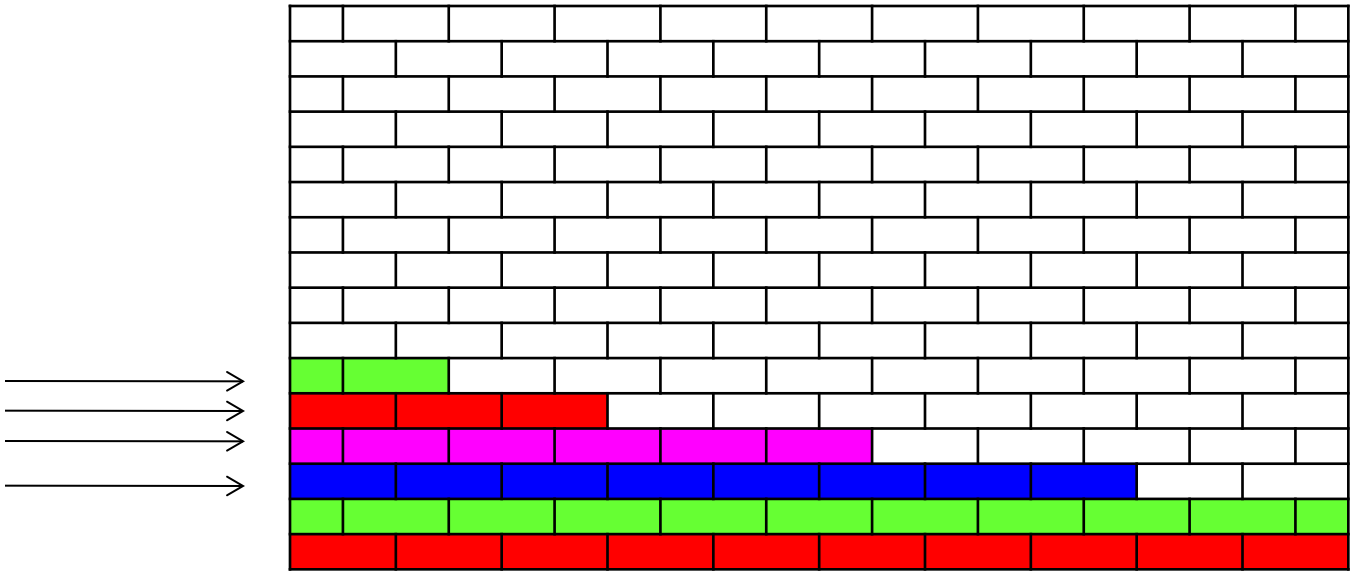


Метод конвейерного параллелизма



$$T_1(kn) = \tau_c kn \quad T_p(kn) = \tau_c \frac{kn}{p} + ?$$

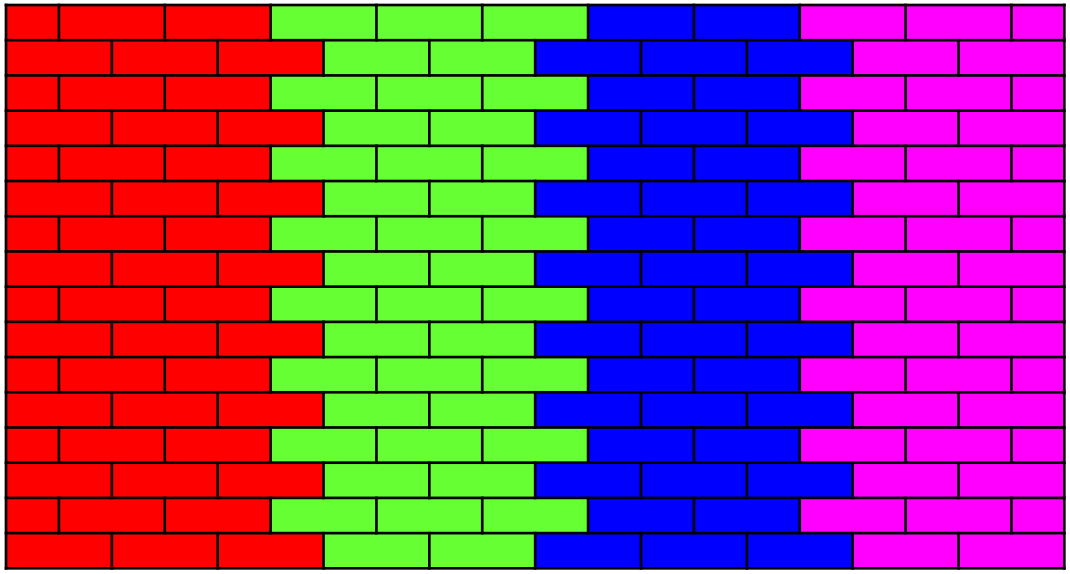
Метод конвейерного параллелизма



$$T_1(kn) = \tau_c kn \qquad T_p(kn) = \tau_c \frac{kn}{p} + k\frac{n}{p}\tau_s$$

$$S_p(kn) = p\frac{1}{1+\frac{\tau_s}{\tau_c}} \qquad E_p(kn) = \frac{1}{1+\frac{\tau_s}{\tau_c}}$$

Метод геометрического параллелизма



$$T_1(kn) = \tau_c kn \qquad T_p(kn) = \tau_c \frac{kn}{p} + 4k\tau_s$$

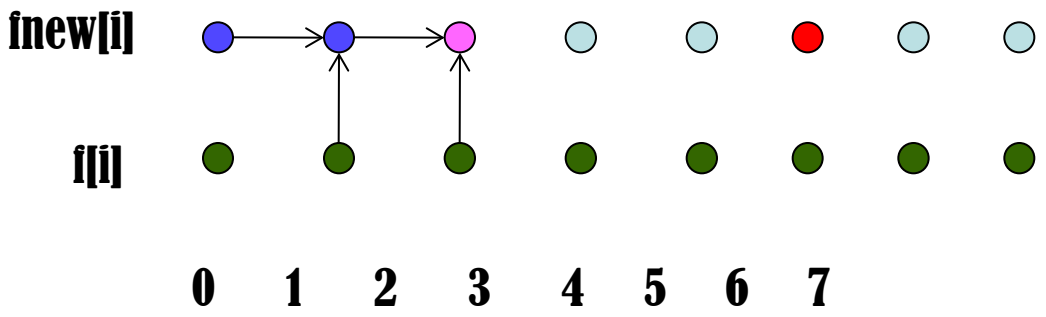
$$S_p(kn) = p \frac{1}{1 + 4\frac{p}{n}\frac{\tau_s}{\tau_c}} \qquad E_p(kn) = \frac{1}{1 + 4\frac{p}{n}\frac{\tau_s}{\tau_c}}$$

Метод конвейерного параллелизма

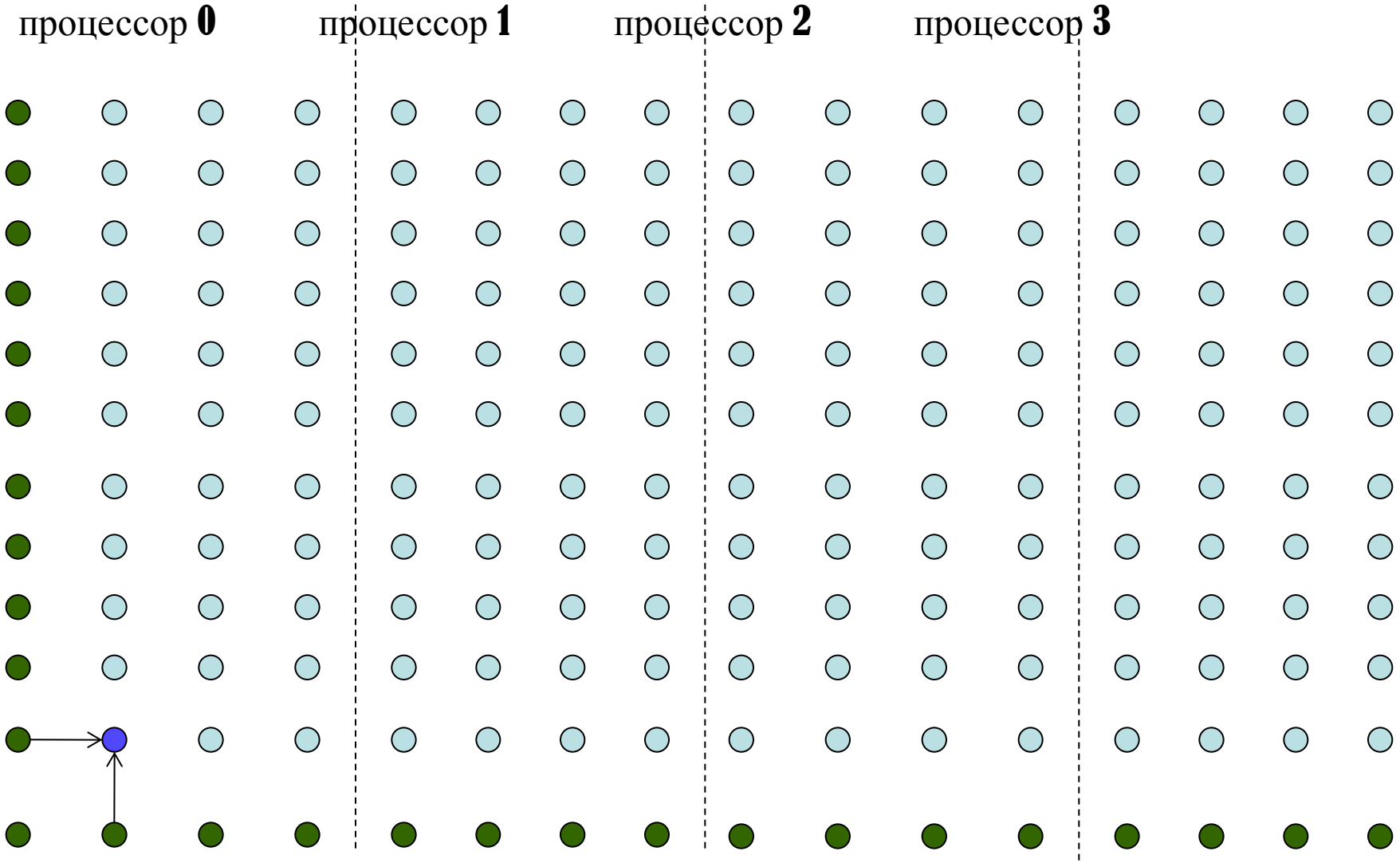
```
for(t=0; t<tmax; t+=dt)
{
        fnew[0]=g(t);

        for(i=1; i<n; i++)
                fnew[i]= fnew[i-1]+f[i]

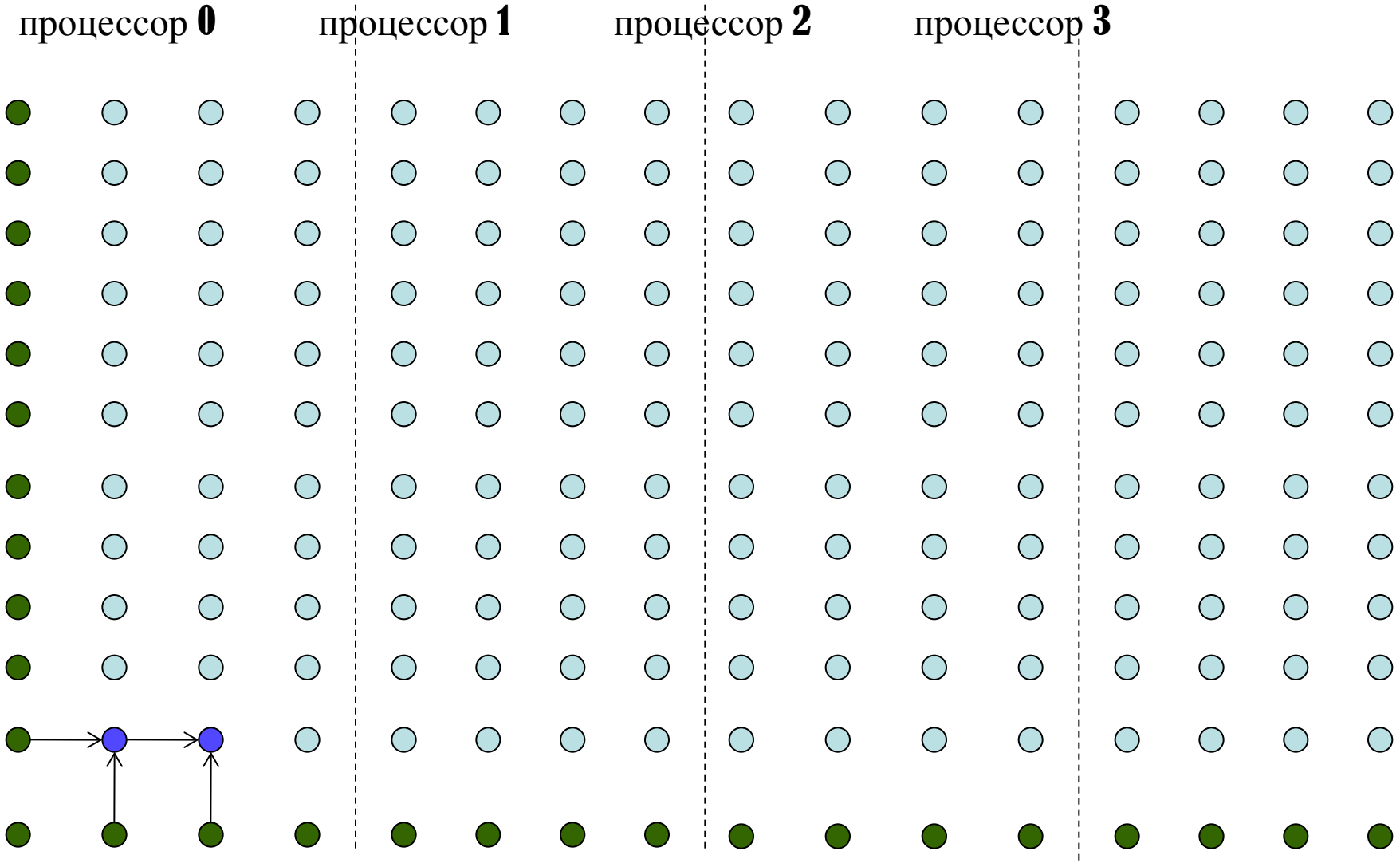
        for(i=0;i<n;i++)
                f[i] = fnew [i]
}
```



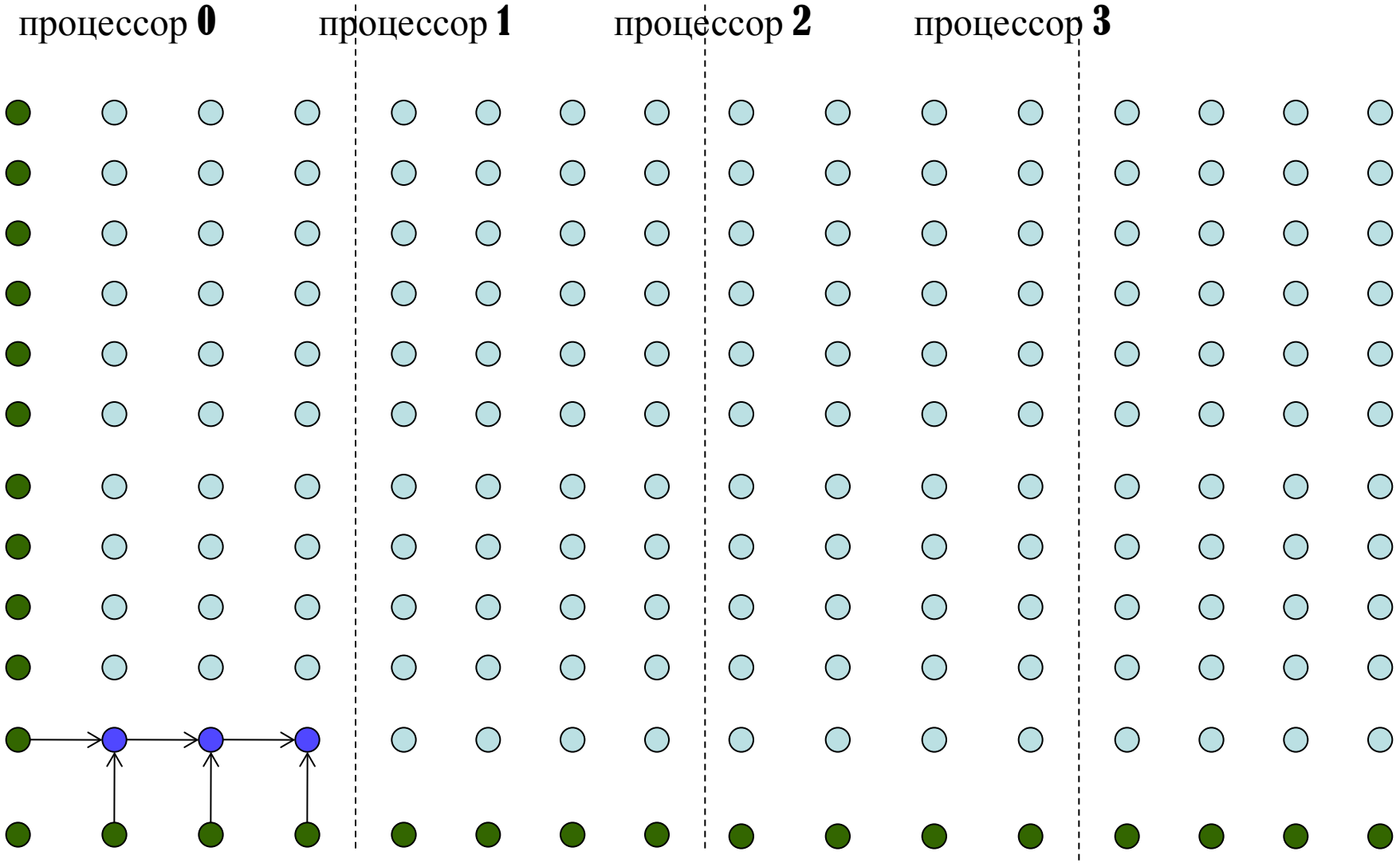
Метод конвейерного параллелизма



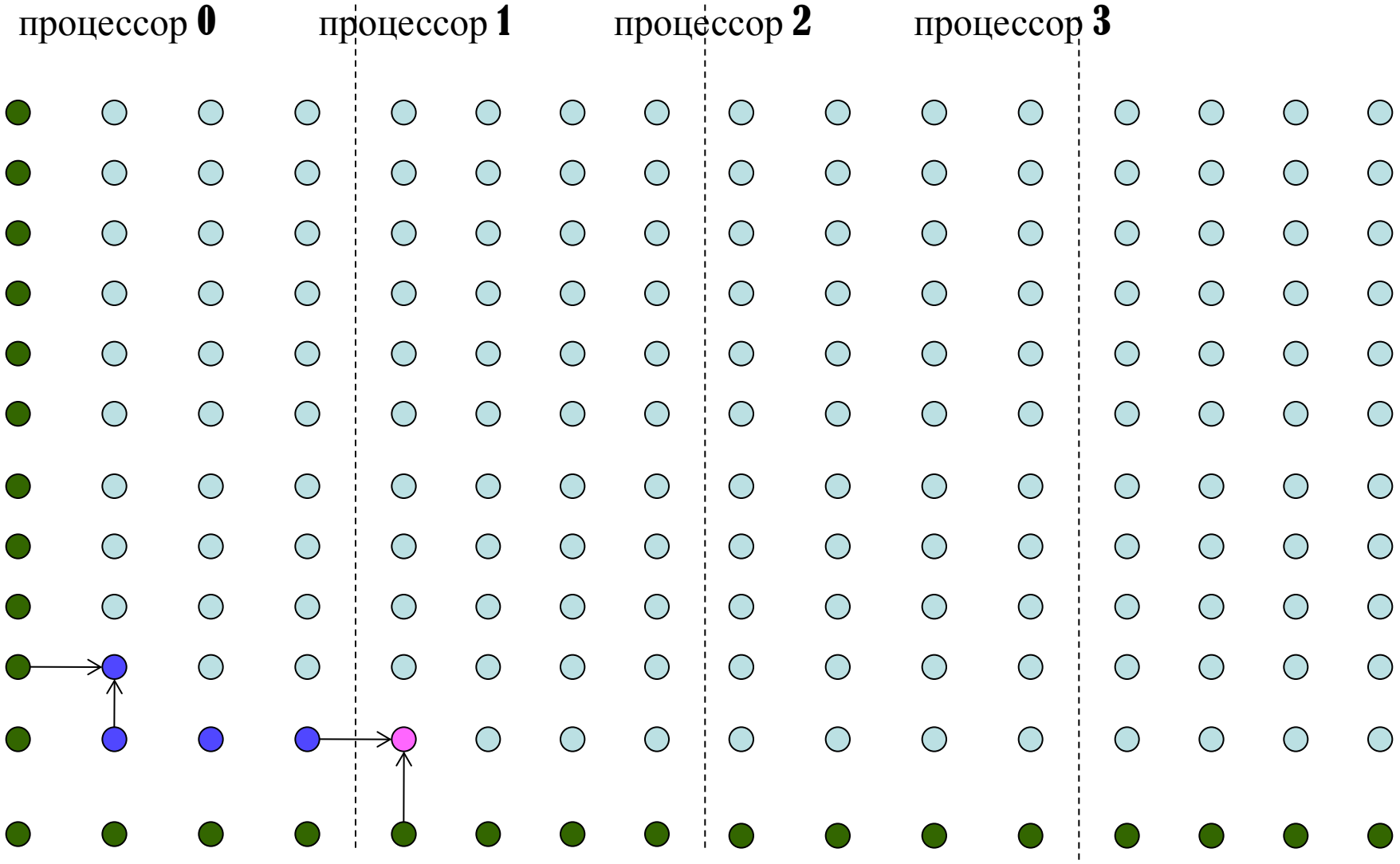
Метод конвейерного параллелизма



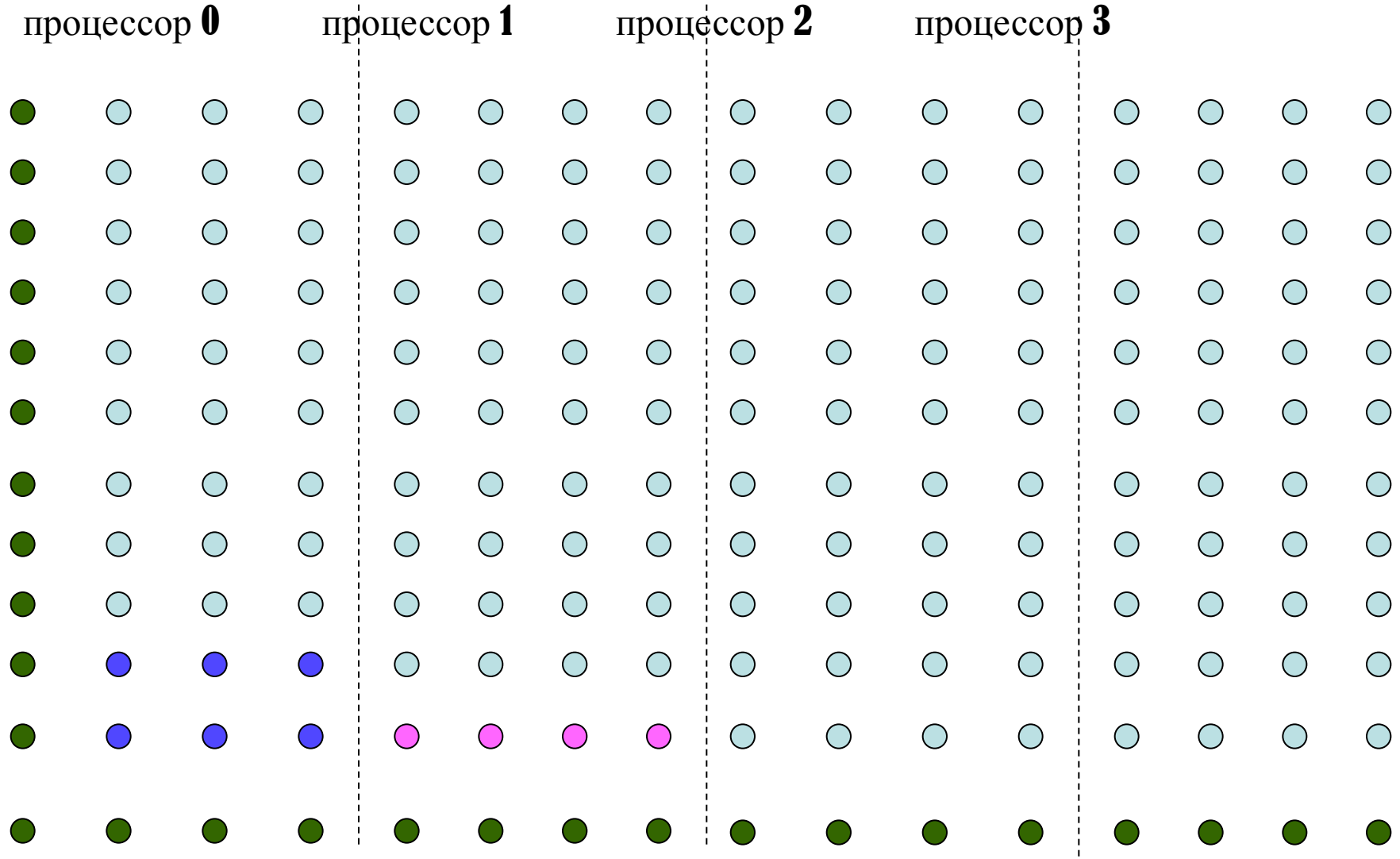
Метод конвейерного параллелизма



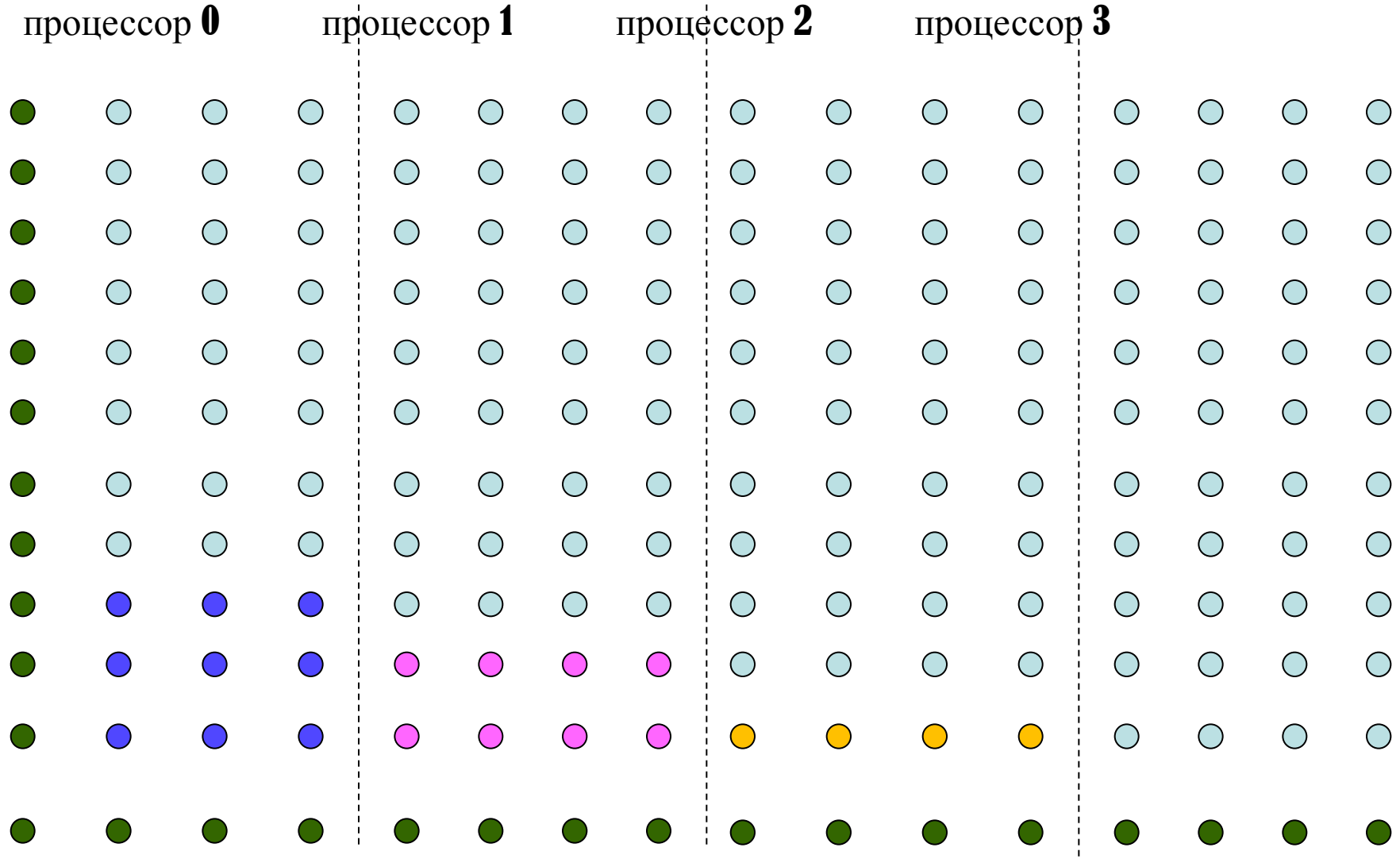
Метод конвейерного параллелизма



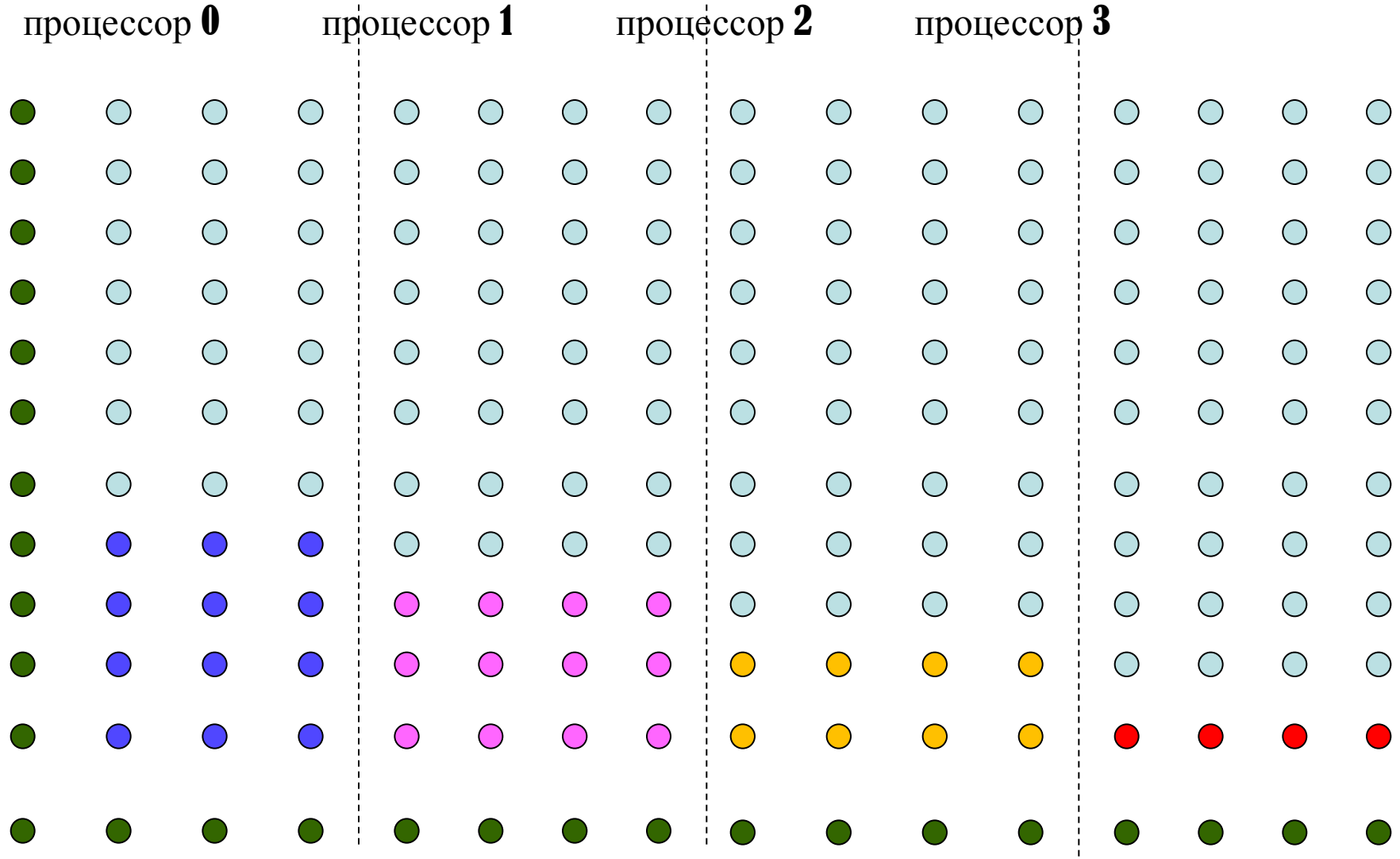
Метод конвейерного параллелизма



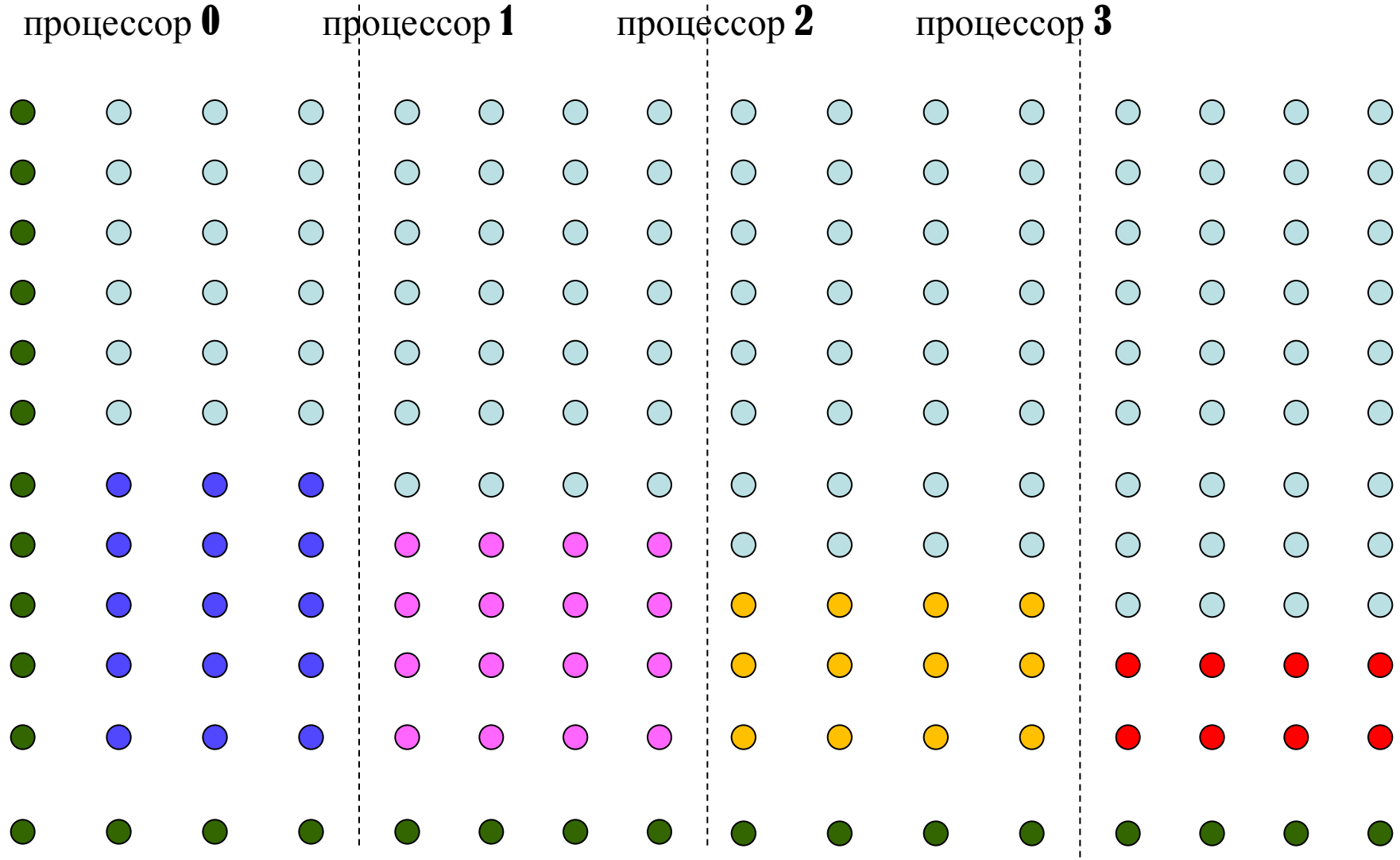
Метод конвейерного параллелизма



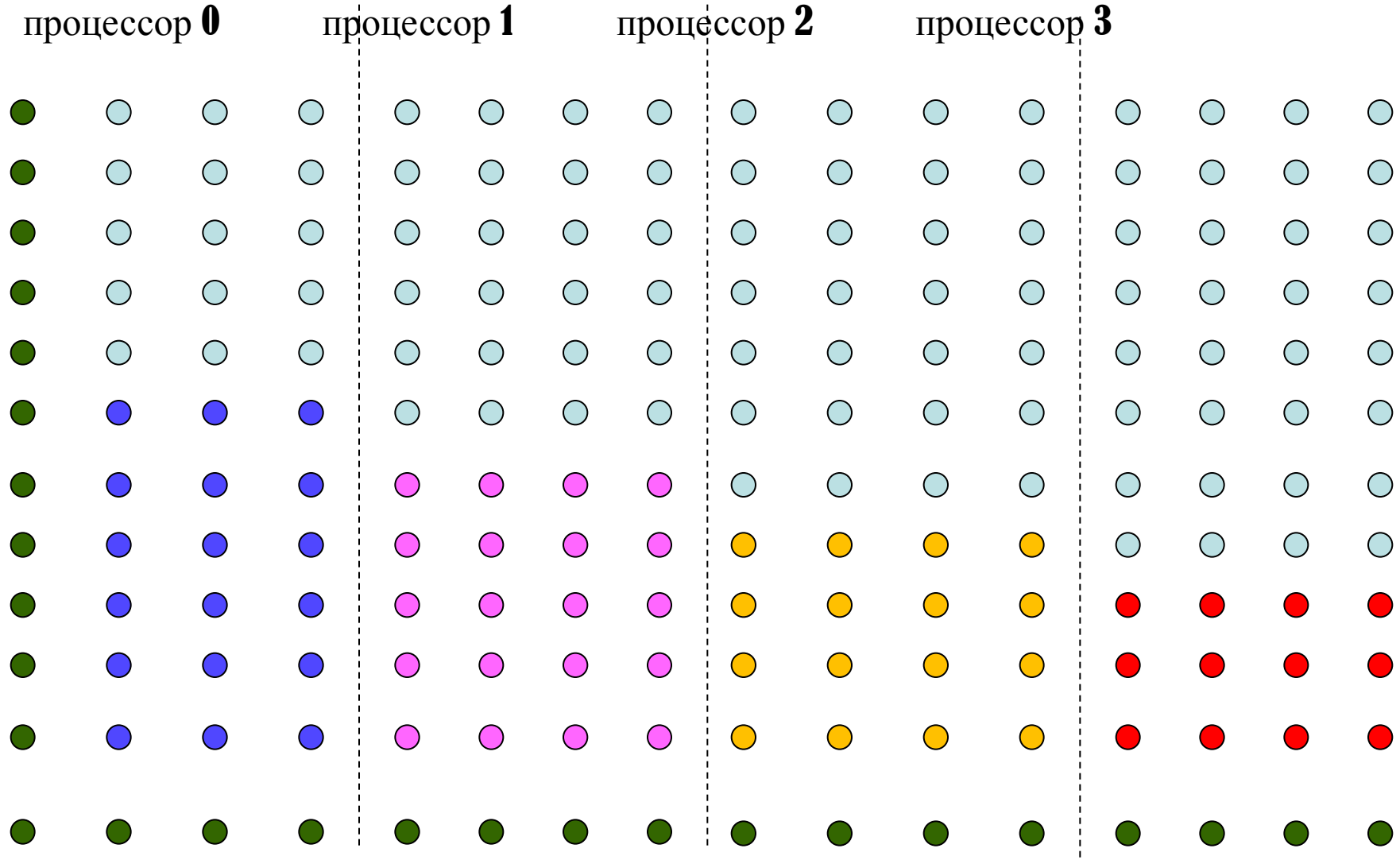
Метод конвейерного параллелизма



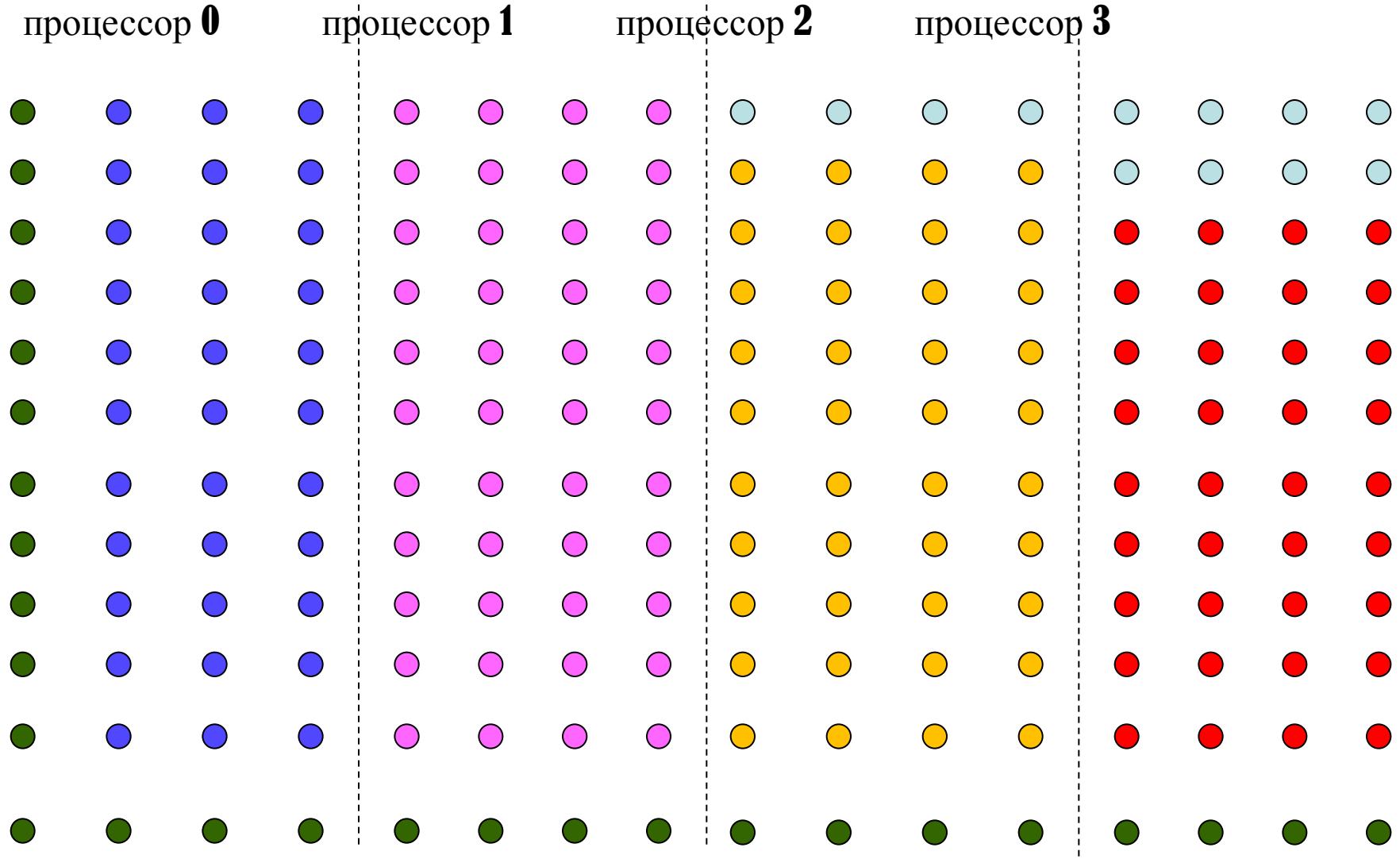
Метод конвейерного параллелизма



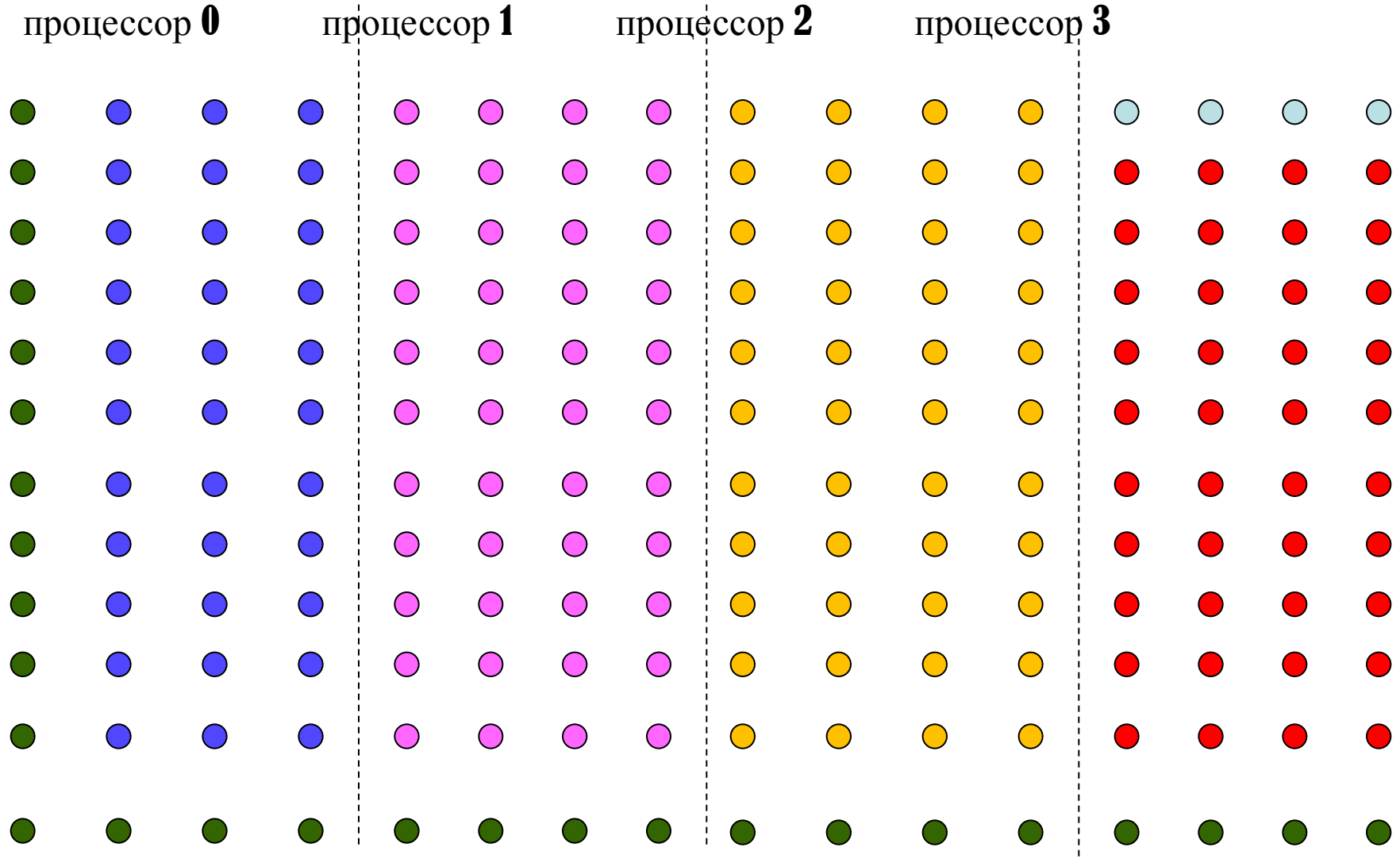
Метод конвейерного параллелизма



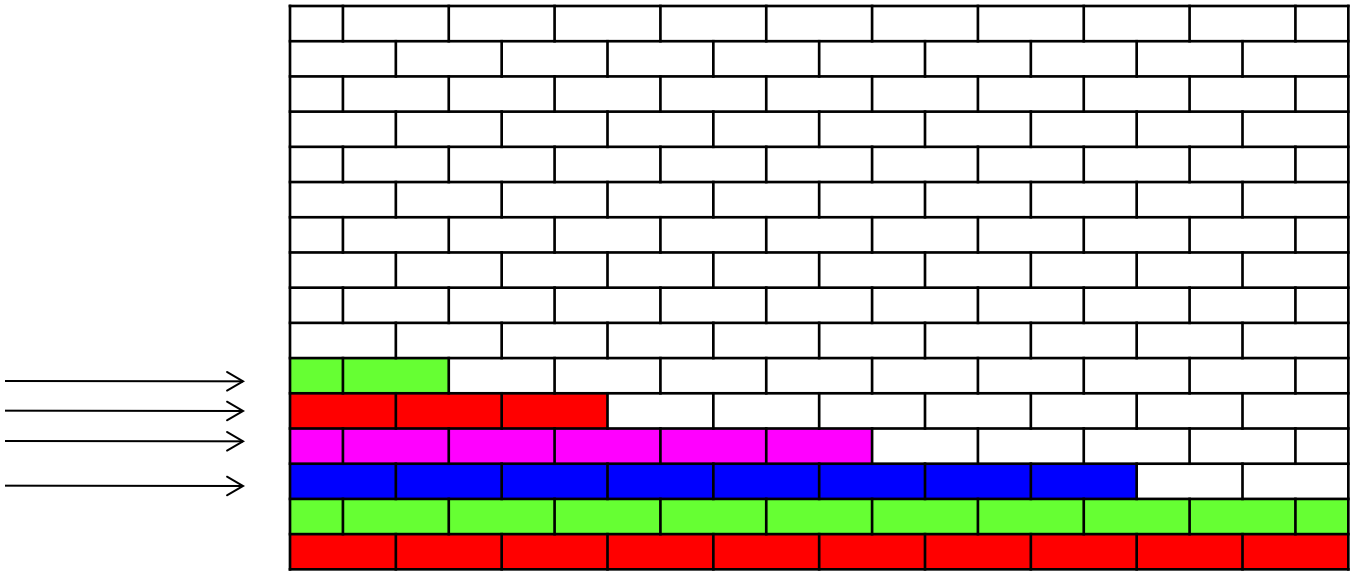
Метод конвейерного параллелизма



Метод конвейерного параллелизма



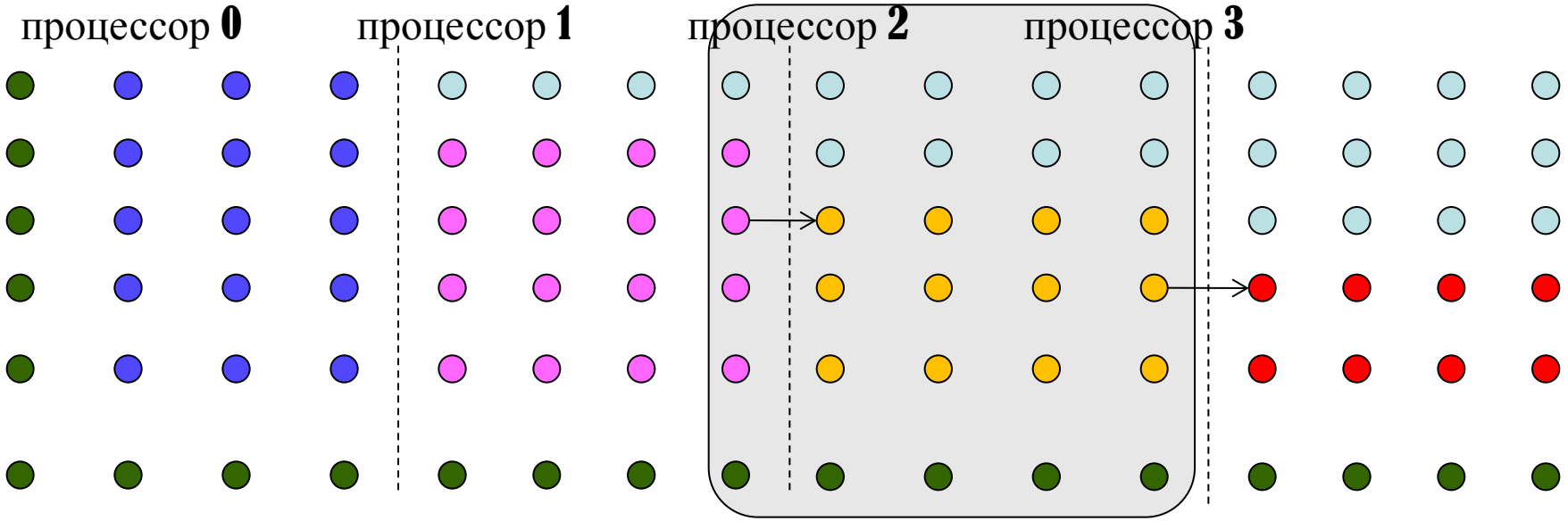
Метод конвейерного параллелизма



$$T_1(kn) = \tau_c kn \qquad T_p(kn) = \tau_c \frac{kn}{p} + k\frac{n}{p}\tau_s$$

$$S_p(kn) = p\frac{1}{1+\frac{\tau_s}{\tau_c}} \qquad E_p(kn) = \frac{1}{1+\frac{\tau_s}{\tau_c}}$$

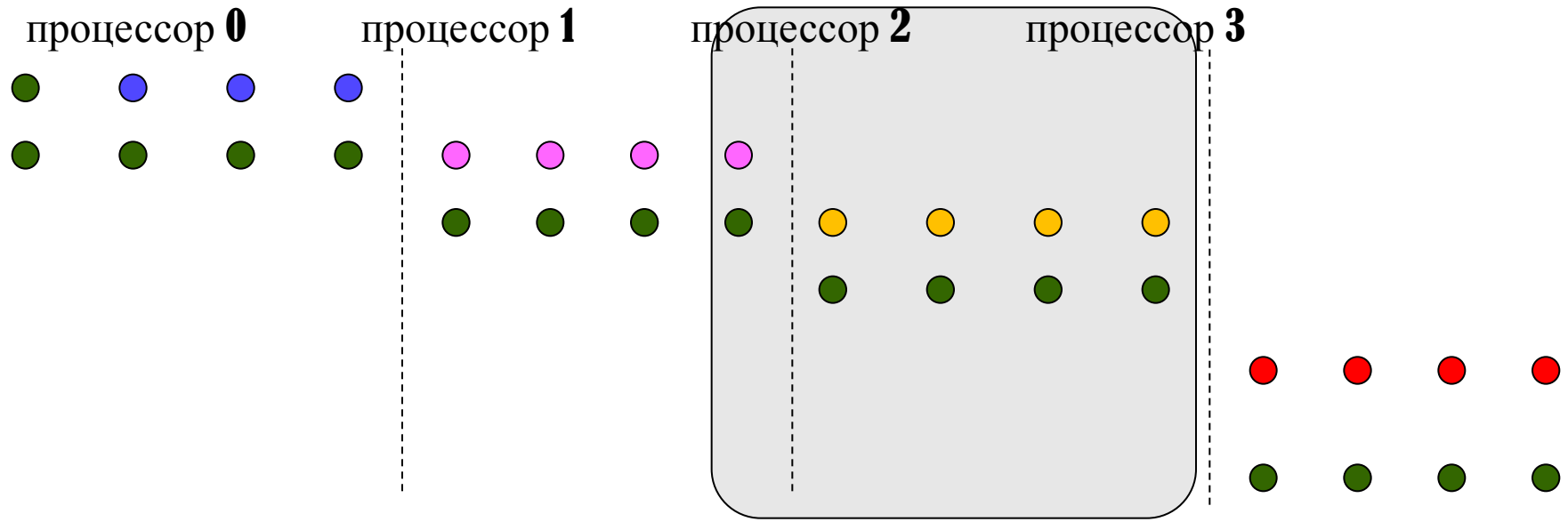
Метод конвейерного параллелизма



$$T_1(kn) = \tau_c kn \qquad T_p(kn) = \tau_c \frac{kn}{p} + 2k\tau_s$$

$$S_p(kn) = p\frac{1}{1+2\frac{p}{n}\frac{\tau_s}{\tau_c}} \qquad E_p(kn) = \frac{1}{1+2\frac{p}{n}\frac{\tau_s}{\tau_c}}$$

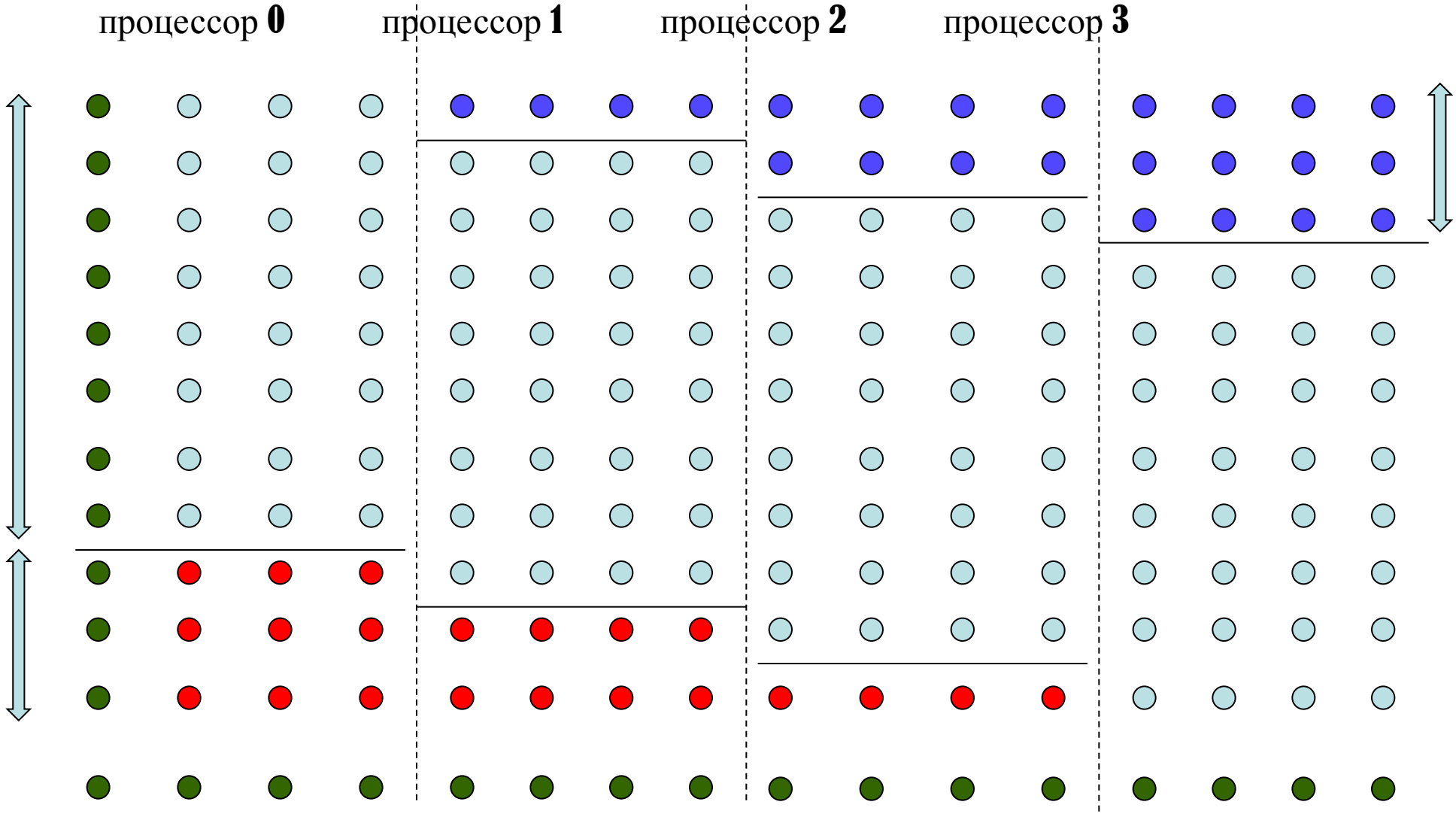
Объём хранимых данных



$$T_1(kn) = \tau_c kn \qquad T_p(kn) = \tau_c \frac{kn}{p} + 2k\tau_s$$

$$S_p(kn) = p\frac{1}{1+2\frac{p}{n}\frac{\tau_s}{\tau_c}} \qquad E_p(kn) = \frac{1}{1+2\frac{p}{n}\frac{\tau_s}{\tau_c}}$$

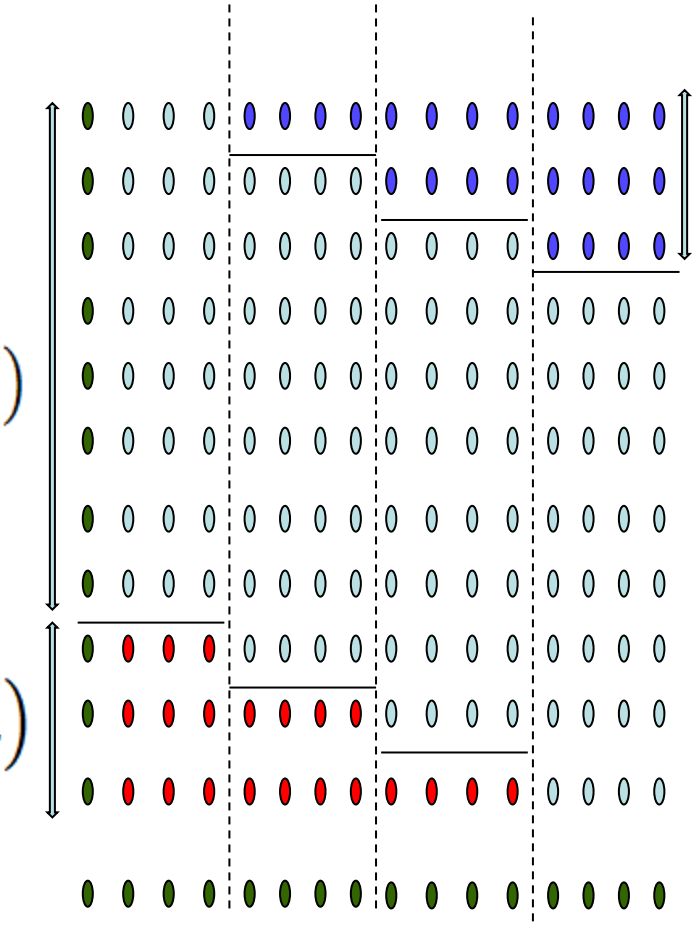
Учет стартовых и финальных затрат



Учет стартовых и финальных затрат

$k = 11$

$p = 4$



$T_{end} = (p-1)\left(\frac{n}{p}\tau_c + 2\tau_s\right)$

$T_{middle} = (k+1-p)\left(\frac{n}{p}\tau_c + 2\tau_s\right)$

$T_{start} = (p-1)\left(\frac{n}{p}\tau_c + 2\tau_s\right)$

$T_{all} = T_{start} + T_{middle} + T_end$

Учет стартовых и финальных затрат

$$T^{all} = (p + k - 1)\left(\frac{n}{p}\tau_c + 2\tau_s\right)$$

$p + k \gg 1$

$$S_p^{all} = p\frac{1}{\left(1 + \frac{p}{k}\right)\left(1 + 2\frac{p}{n}\frac{\tau_s}{\tau_c}\right)}$$

$$E_p^{all} = \frac{1}{\left(1+\frac{p}{k}\right)\left(1+2\frac{p}{n}\frac{\tau_s}{\tau_c}\right)}$$

Метод эффективен при $p \ll k$

Максимальная степень параллелизма: $\min(n, k)$

Максимальное ускорение: $\frac{pk}{p+k} \leq \frac{k}{2}$

Простые алгоритмы

- Статическая и динамическая балансировка загрузки процессоров
  - Статическая балансировка
    - метод сдваивания
    - геометрический параллелизм
    - конвейерный параллелизм
  - Динамическая балансировка
    - коллективное решение
    - диффузная балансировка

Заключение

- Отмечена важность использования простых с логической точки зрения алгоритмов
- Рассмотрены основные причины потерь времени при выполнении параллельных алгоритмов
- Рассмотрен метод геометрического параллелизма, относящийся к классу методов статической балансировки загрузки процессоров
- Рассмотрен метод коллективного решения, относящийся к классу методов динамической балансировки загрузки процессоров

Контакты

Якобовский М.В., *чл.-корр. РАН, проф., д.ф.-м.н.,*

Заместитель директора по научной работе
Института прикладной математики
им. М.В.Келдыша Российской академии наук

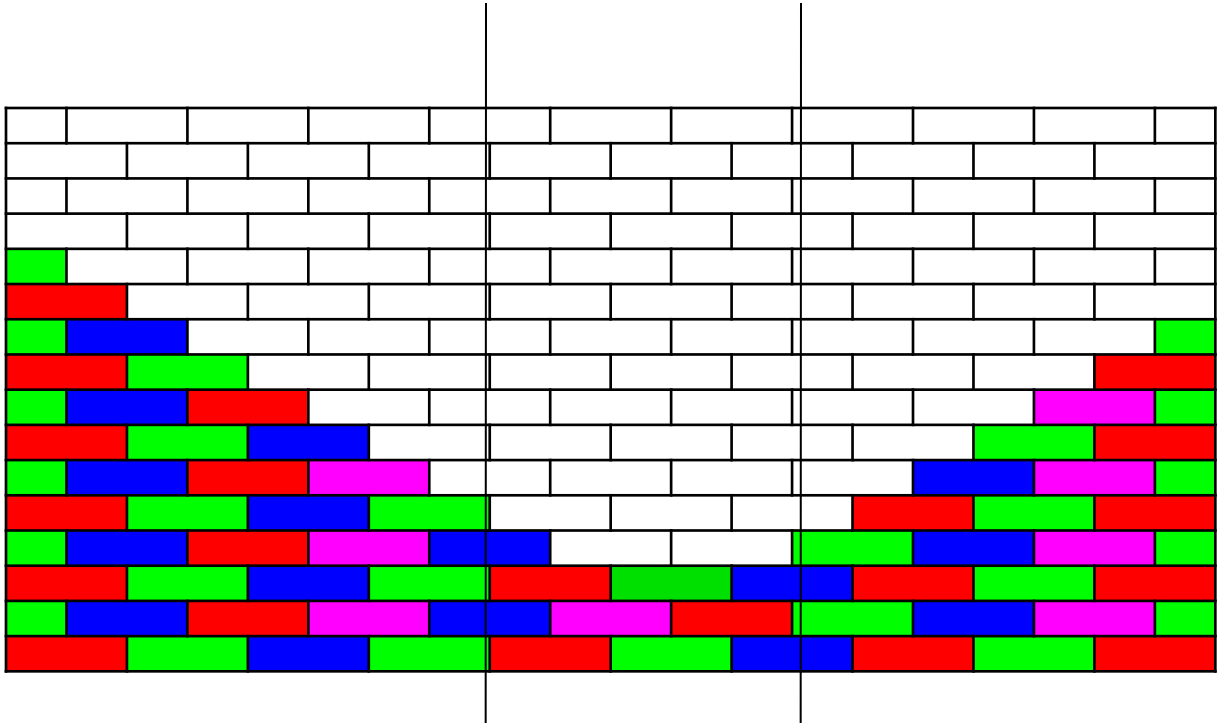
mail: lira@imamod.ru

web: http://lira.imamod.ru

Диффузная балансировка

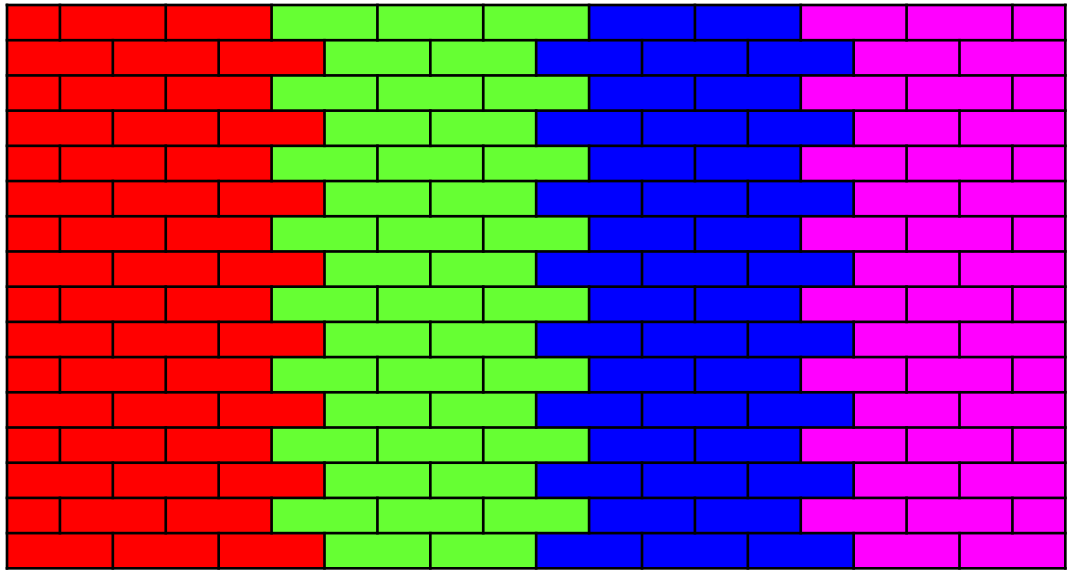
- Причины дисбаланса вычислительной нагрузки
  - Разные процессоры
  - Внешнее воздействие
  - Разная вычислительная сложность заданий
- Результат дисбаланса
  - Эффективная производительность определяется самым медленным процессором

Медленный процессор

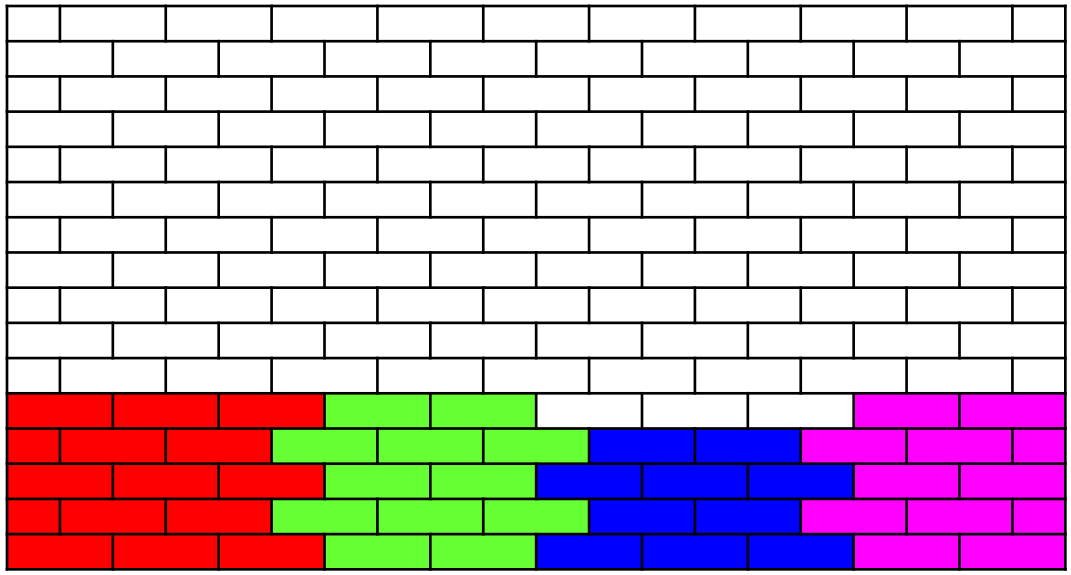


Какой объем работ забрать у среднего процессора и кому его передать?

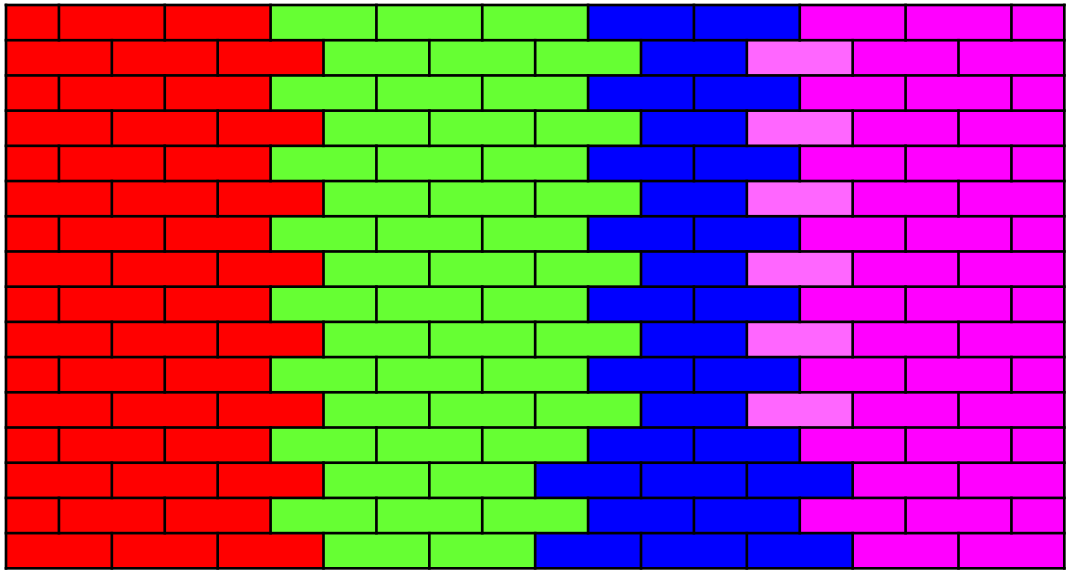
Метод геометрического параллелизма



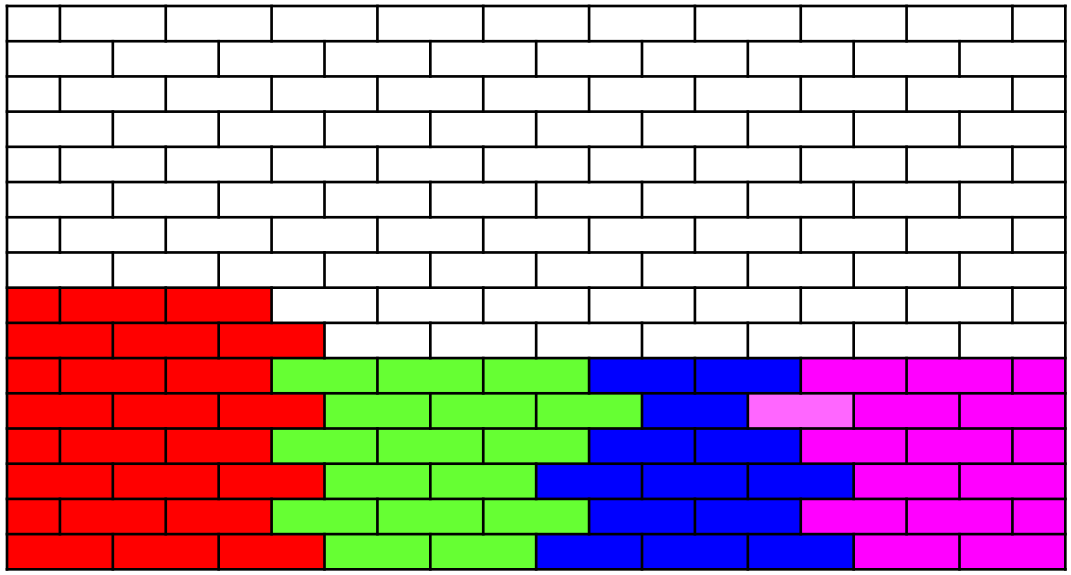
Метод геометрического параллелизма



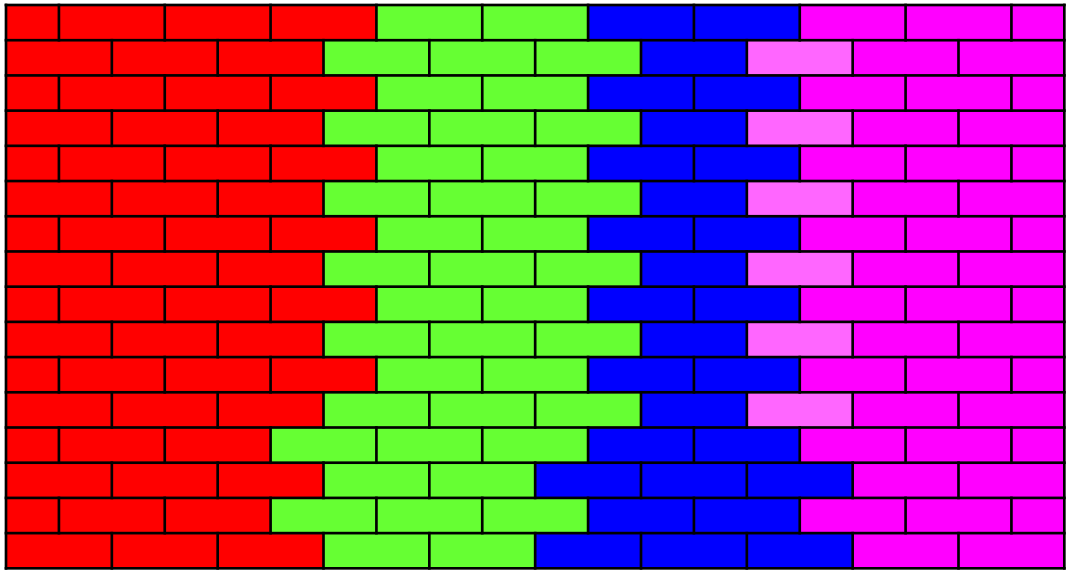
Диффузная балансировка загрузки



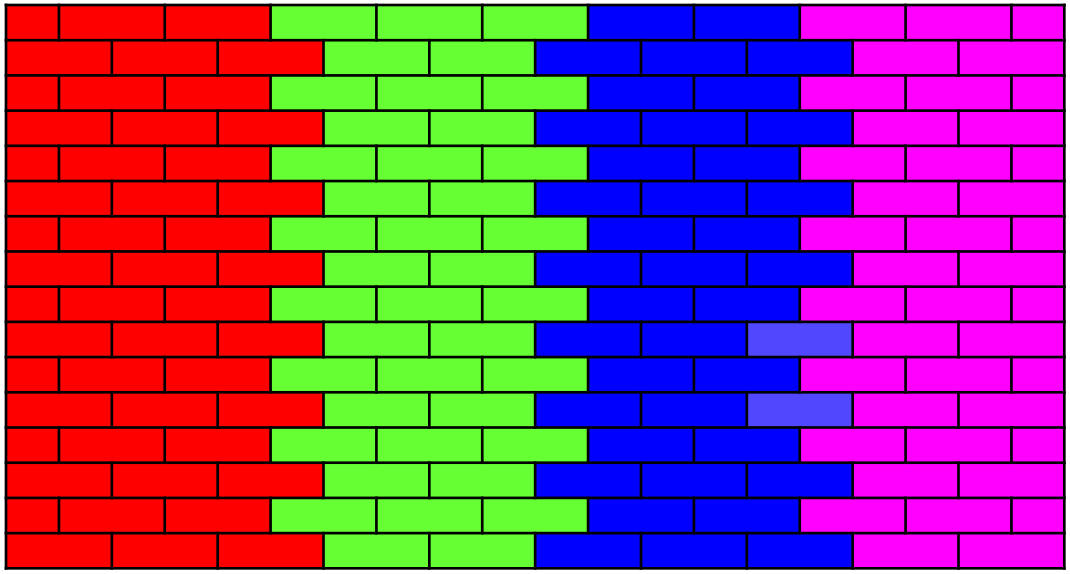
Диффузная балансировка загрузки



Диффузная балансировка загрузки



Статическое распределение



Постановка задачи диффузной балансировки

Дано:

- Количество точек – N
- Количество процессоров – p
- Процессор i обработал n_i точек за время t_i
- Для обработки любой точки требуется одинаковое число операций

Требуется:

- Найти количества точек n'_i , которое следует обработать процессорам на следующем шаге
- Определить сколько точек каждый из процессоров должен передать соседним процессорам

Диффузная балансировка

$$n_i' = N \frac{\frac{n_i}{t_i}}{\sum_{j=0}^{p-1} \frac{n_j}{t_j}}$$

В предположении одинаковой трудоёмкости обработки каждой из точек

Диффузная балансировка

$$n_i' = N \frac{\frac{n_i}{t_i}}{\sum_{j=0}^{p-1} \frac{n_j}{t_j}}$$

В предположении одинаковой трудоёмкости обработки каждой из точек