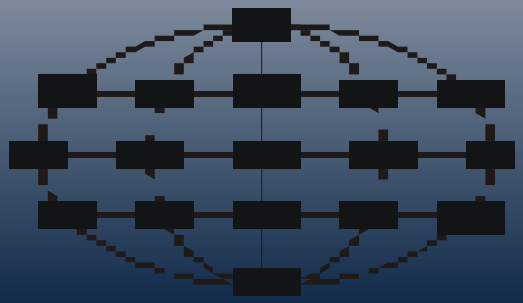
Система управления распределёнными структурами данных

Стадник Е. Г.
ФПМИ, НГТУ

Руководитель:
Городничев М.А., м.н.с. ИВМ и МГ СО РАН



Пример задачи с использованием распределённых данных

- **Уравнение Пуассона**

$$\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = f$$

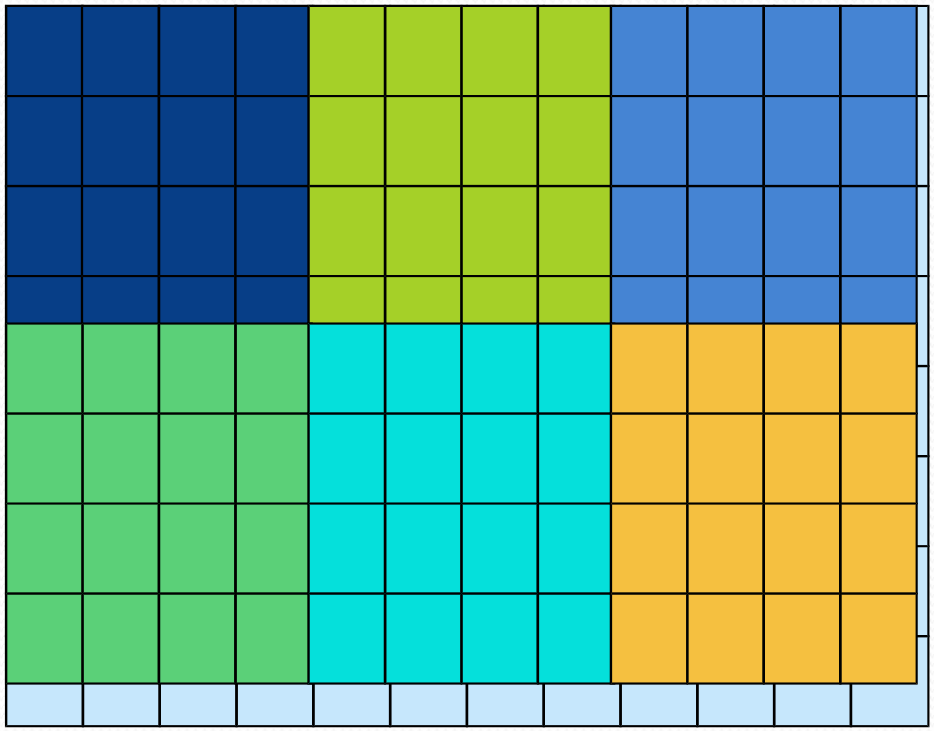
- Дискретизация
- Дискретизация

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = f_{i,j}$$

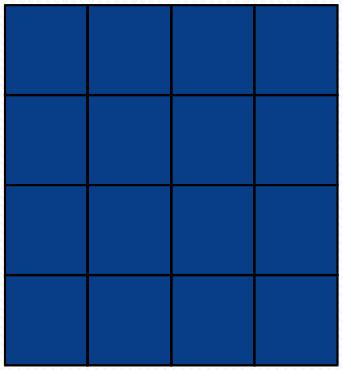
- Итеративный метод
- Итеративный метод

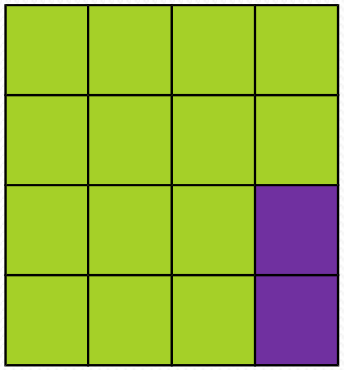
$$u_{i,j}{}^{n+1} = \frac{1}{4}\left(u_{i+1,j}{}^{n} + u_{i-1,j}{}^{n} + u_{i,j+1}{}^{n} + u_{i,j-1}{}^{n} - h^2 f_{i,j}\right)$$

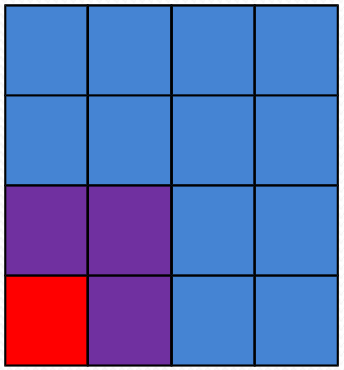
Декомпозиция данных

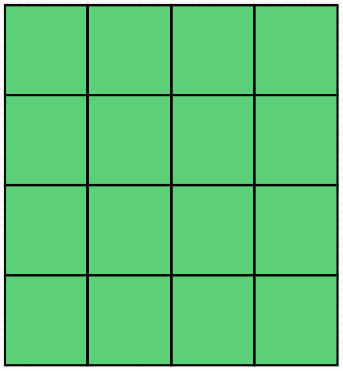


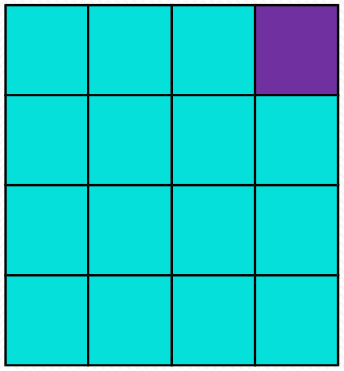
Организация коммуникаций

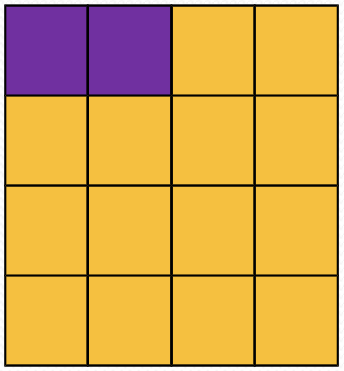




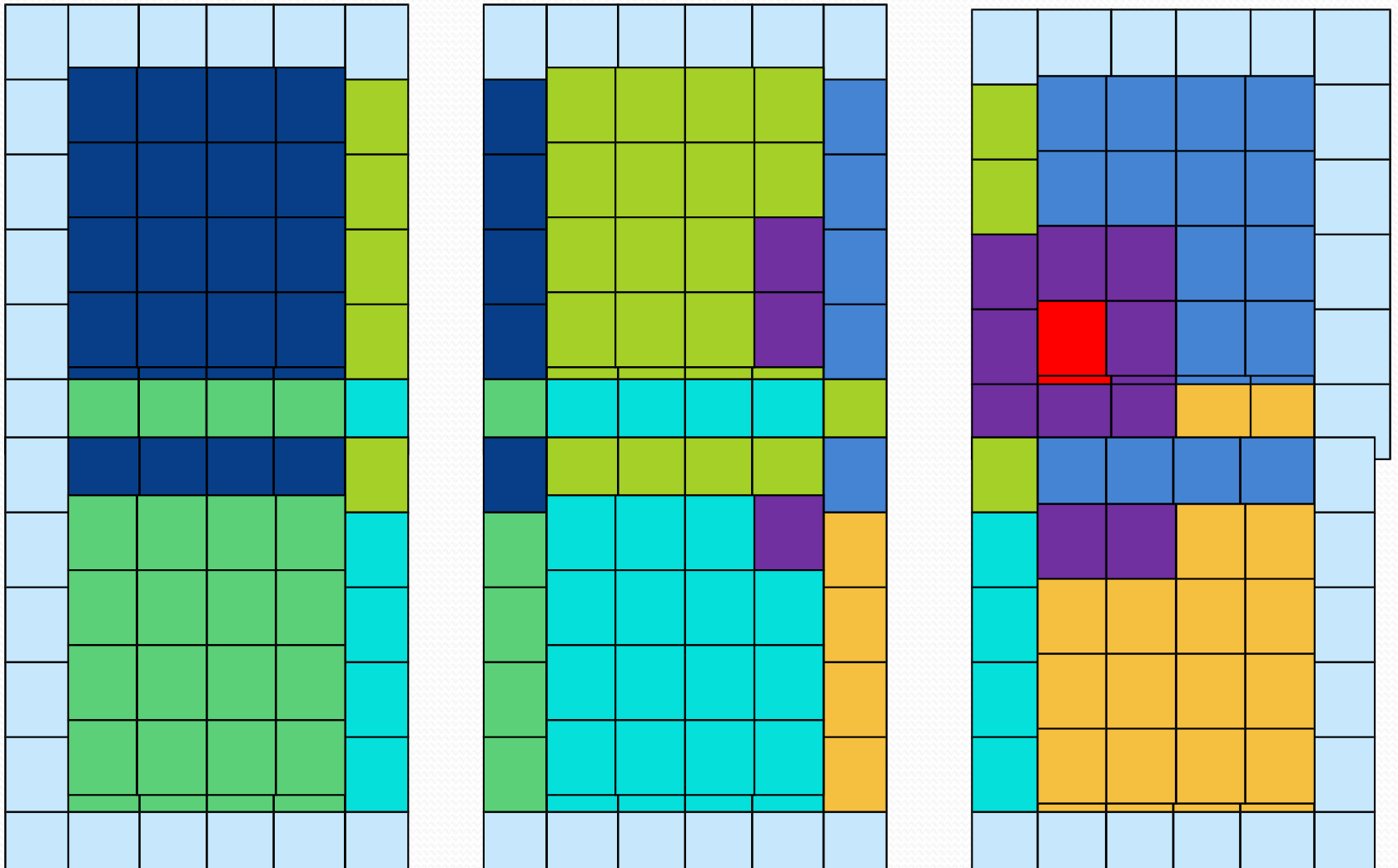




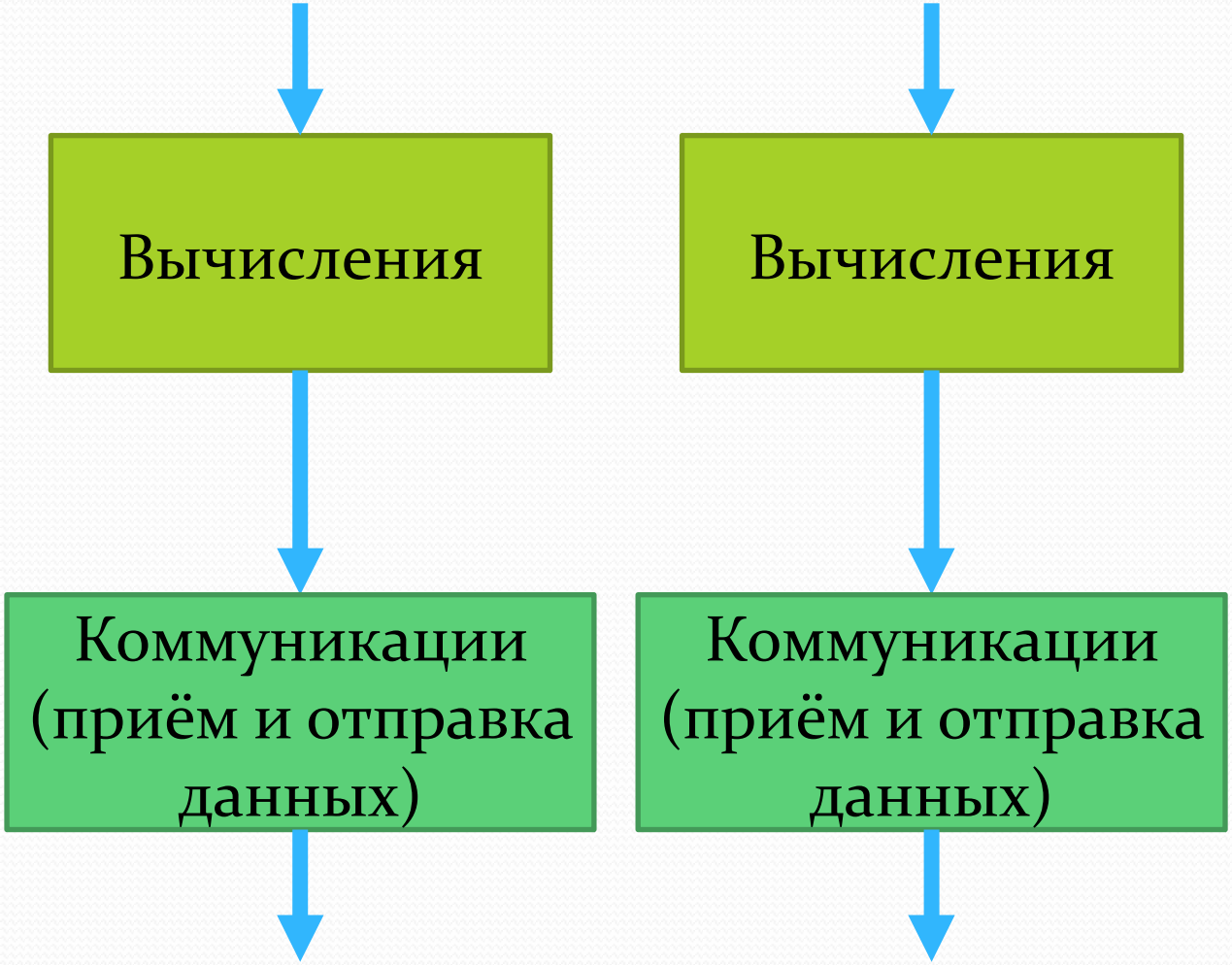




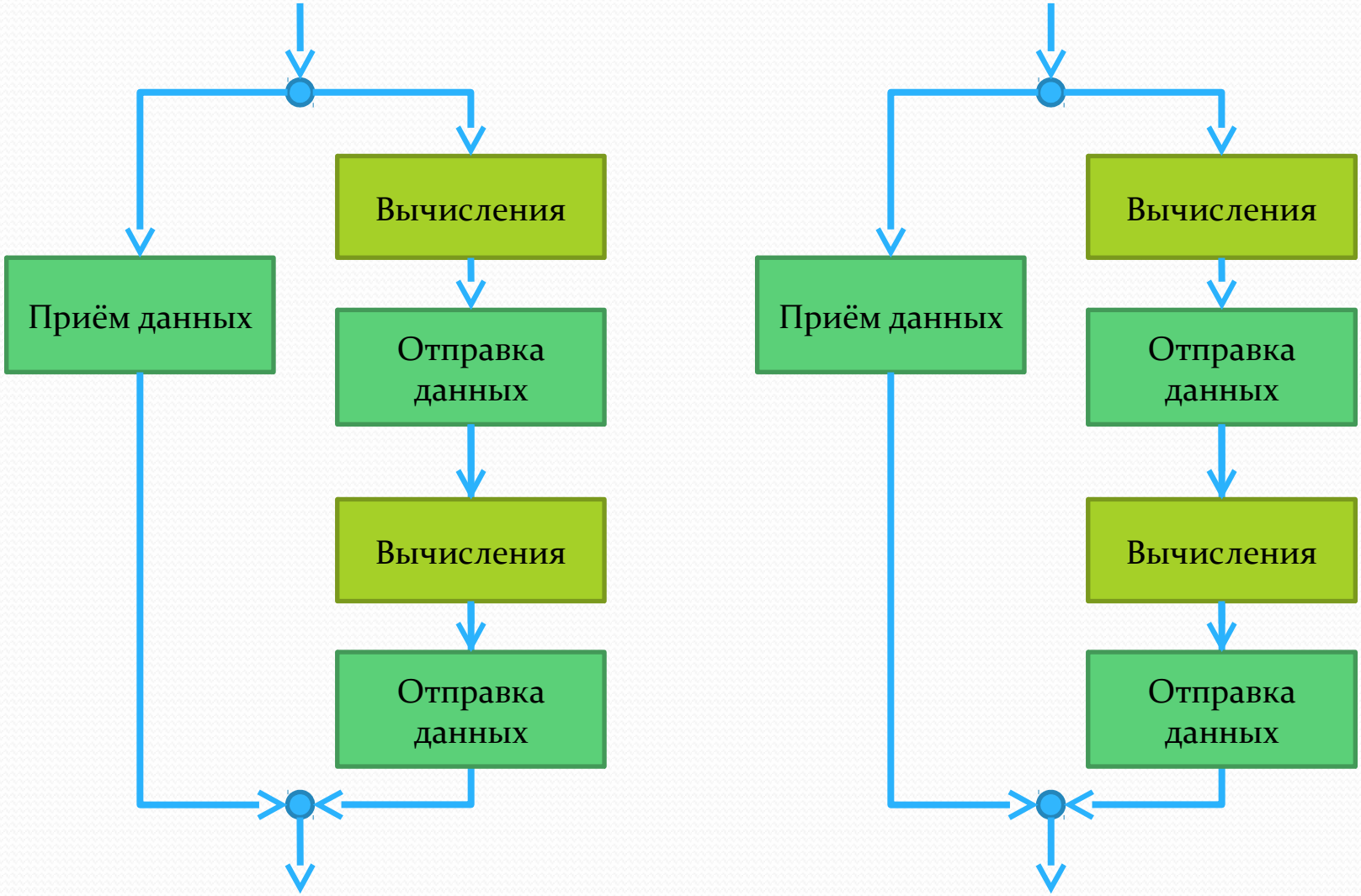
Организация коммуникаций



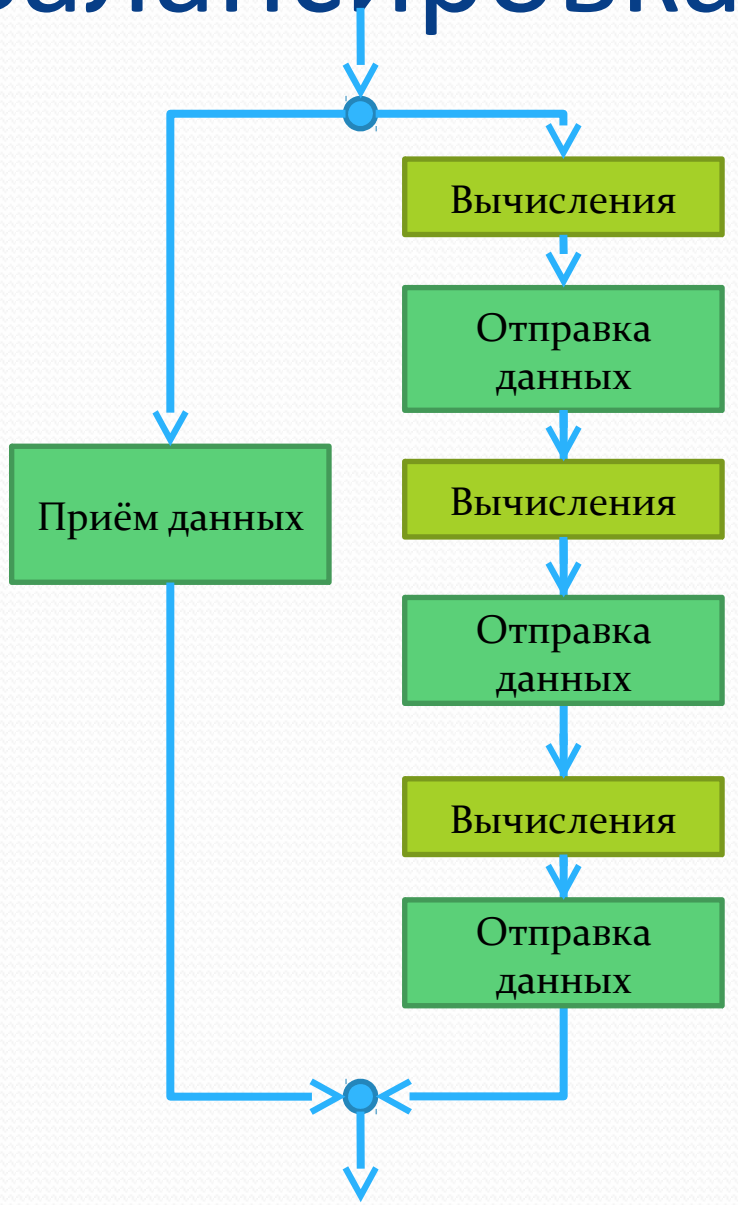
Оптимизация коммуникаций

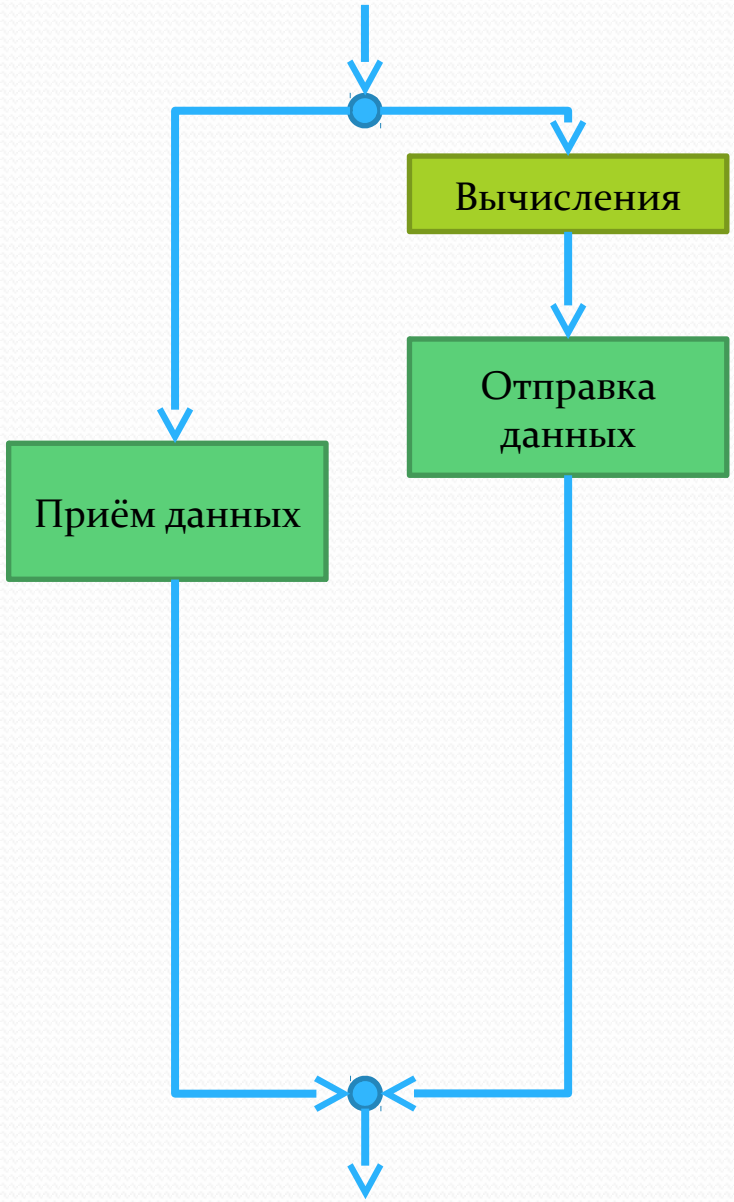


Оптимизация коммуникаций



Балансировка





Суммарные требования к фрагментам

- Регулярная структура и возможность обработки фрагментов при помощи циклов также, как и в массивах.
- Автоматическое удовлетворение всех зависимостей по данным посредством коммуникаций.
-
-
-

Анализ системы DVM

- Достоинства
  - Повышение скорости разработки параллельных программ, реализующих численные методы с регулярными структурами данных и вычислений.
  - Возможность контролировать процесс распараллеливания, добиваясь максимальной эффективности программ.
  - Имеется встроенный отладчик и профайлер
- Недостатки
  - Элементы массивов не могут иметь произвольный тип, только стандартные `int`, `long`, `float`, `double`
  - Отсутствие автоматической динамической балансировки
  - Не обеспечивается правильная обработка указателей на элементы массивов при перераспределении данных.

Библиотека DDS

Distributed data system

Структура библиотеки

Пример использования библиотеки

```
int main(int argc, char** argv)
{
    //инициализация библиотеки
    DDS_Init(&argc, &argv);
    //объявление переменных типа "распределённый массив"
    DDS_Array <double> Lay1;
    DDS_Array <double> Lay2;
    //задание параметров для распределённых переменных
    Lay2.set_all_size(bsizex, bsizey, block_count_X, block_count_Y);
    Lay2.set_parametrs(1, DA_LINE_DISTRIB);
    Lay1.set_all_size(bsizex, bsizey, block_count_X, block_count_Y);
    Lay1.set_parametrs(0, DA_LINE_DISTRIB);
    //сообщаем системе, что готовы к работе(запускается поток для приёма данных)
    DDS_Begin();
    ...
       //завершаем работу с библиотекой
    DDS_Finalize();
}
```

Пример использования библиотеки

```
for (int iter=0; iter<1000; iter++)
{
 //перебираем все локальные фрагменты
 for(DDS_Array<double>::iterator it1 = Lay1.begin(); it1<Lay1.end(); it1++)
 {
  //для текущего фрагмента считываем параметры
  it1->Get_start_element_index(&X, &Y);
  it1->Get_block_index(&global_j, &global_i);
  //получаем данные из текущего фрагмента
  double *dat=it1->Get_Data();
  //обрабатываем массив из текущего фрагмента
  for (int i=Y; i<Y+bsizey; i++, index+=2, global_i++)
  {
   for (int j=X; j<X+bsizex; j++, index++, global_j++)
    dat[index]=0.25*(dat[index+1]+dat[index-1]      +dat[index+bsizex+2]+dat[index-bsizex-2]
    -h*h*f(x,y));
   global_j-=bsizex;
  }
  //сообщаем системе, что обработка текущего фрагмента завершена
  it1->border_ready();
 }}
```

Интерфейс библиотеки

```
//инициализация библиотеки
int DDS_Init(int *argc, char ***argv);
//запуск потока для асинхронного приёма данных
int DDS_Begin();
//завершение работы с библиотекой
int DDS_Finalize();
```

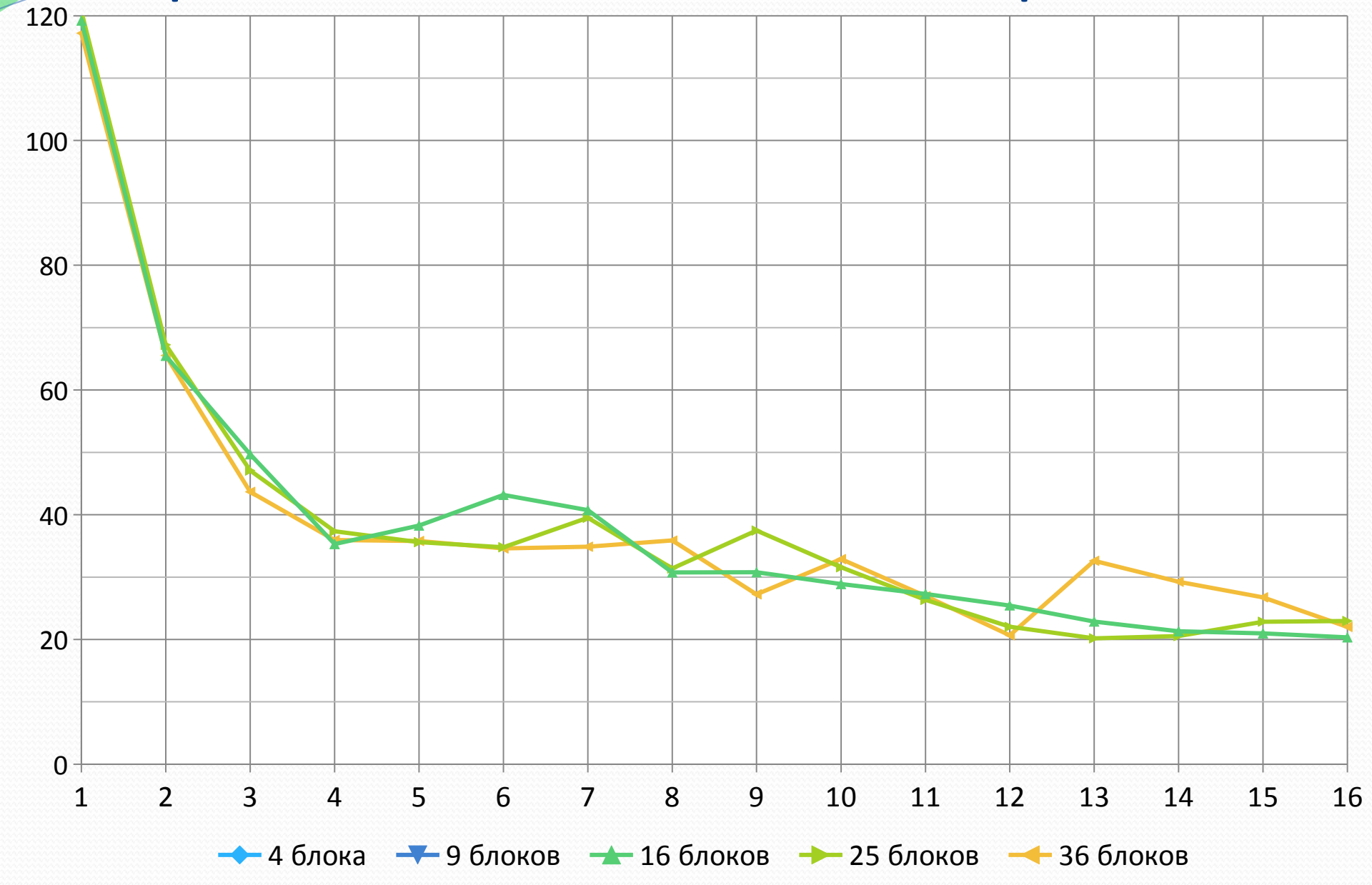
Интерфейс библиотеки
Контейнер фрагментов

```
//задаёт размеры и количество фрагментов
int set_all_size(int _Nx, int _Ny, int _kx, int _ky);
//задаёт идентификатор переменной и тип распределения
фграментов
int set_parametrs(int _id, int _distrib_type);
//возвращает итератор на первый фрагмент
iterator begin();
//возвращает итератор на последний фрагмент
iterator end();
//возвращает итератор фрагмента с индексом i
iterator operator[] (unsigned i);
```

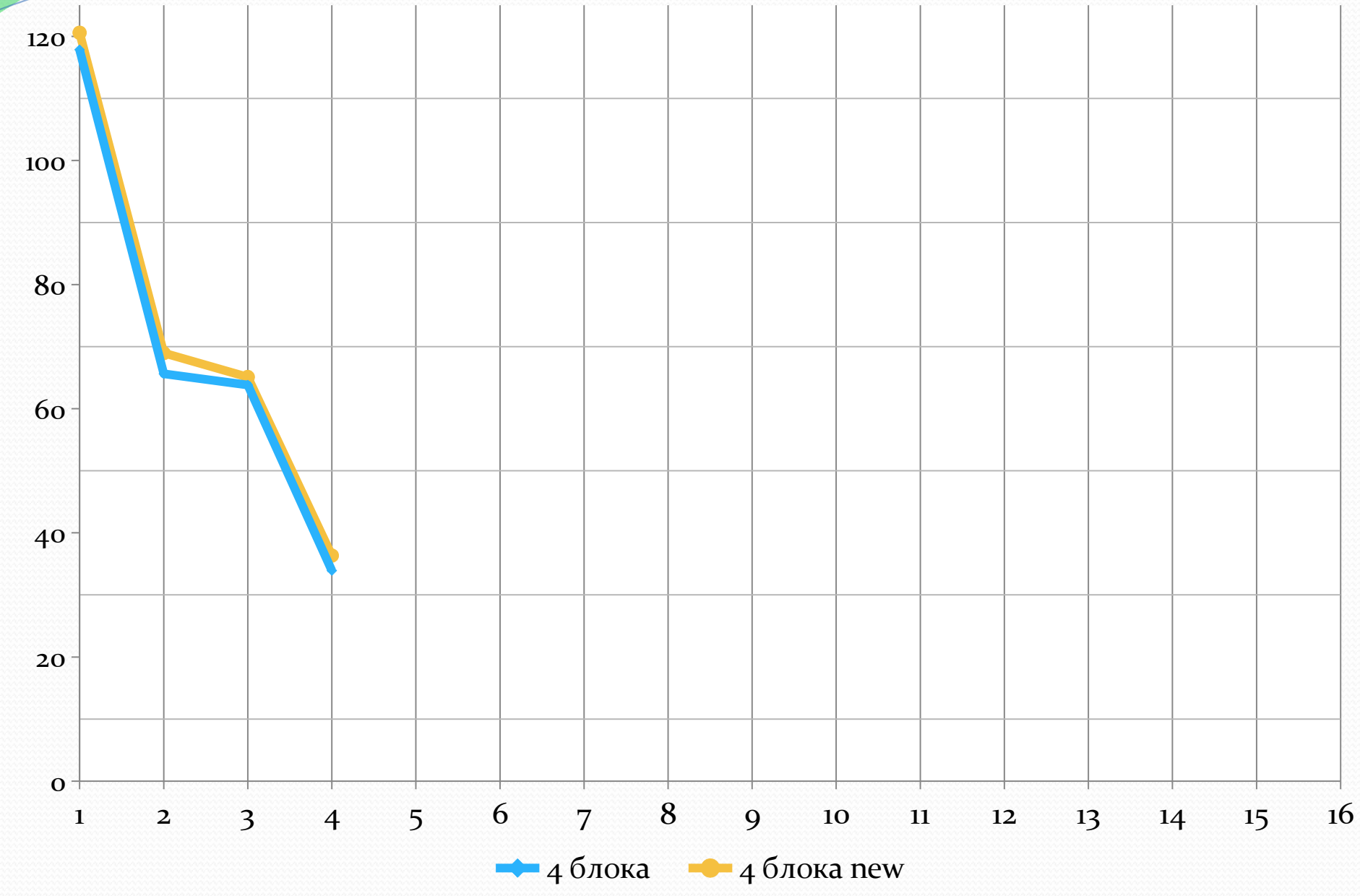
Интерфейс библиотеки
Фрагмент данных

```
//возвращает указатель на данные фрагмента
T* Get_Data();
//заменяет данные фрагмента
int Replace_Data(T* source, int delete_flag);
//возвращает координаты фрагмента
void Get_block_index(int *number_x, int *number_y);
//возвращает размеры фрагмента
void Get_block_dim(int *count_x, int *count_y);
//возвращает глобальные координаты элемента с индексом (0,0)
void Get_start_element_index(int *x, int *y);
//указывает системе, что фрагмент готов для обмена данными
int border_ready();
```

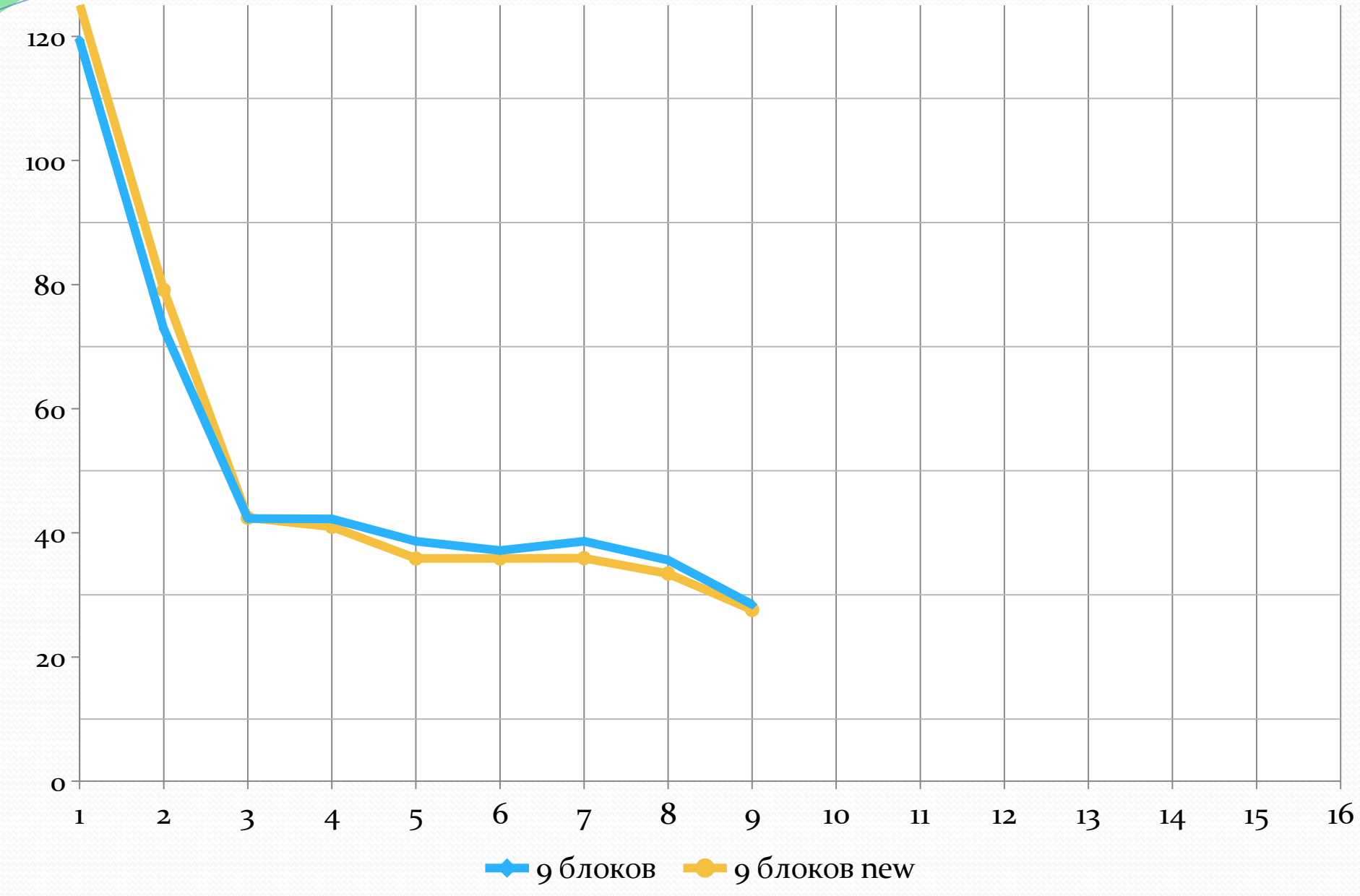
Измерение вычислительных затрат



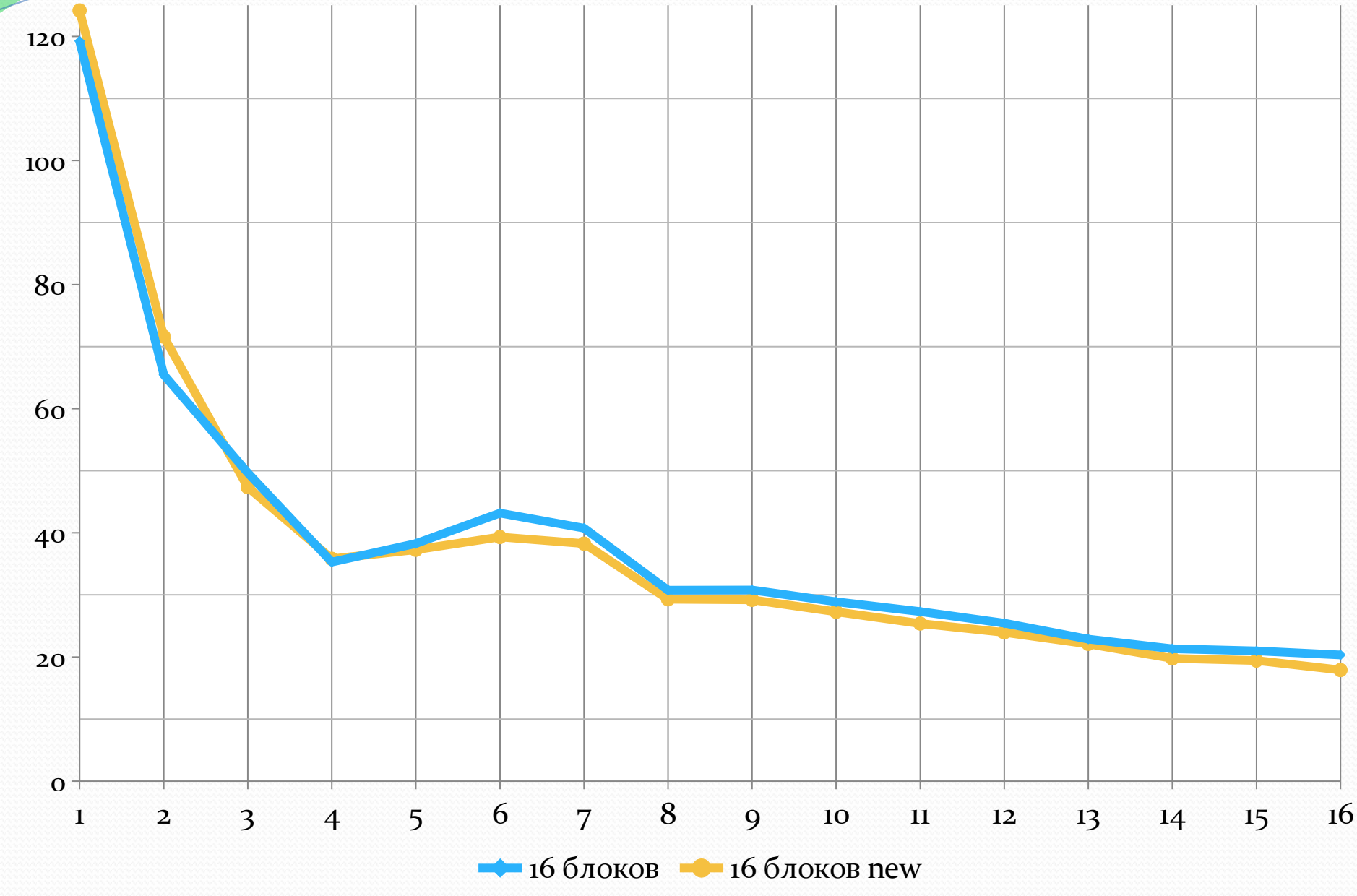
Измерение вычислительных затрат



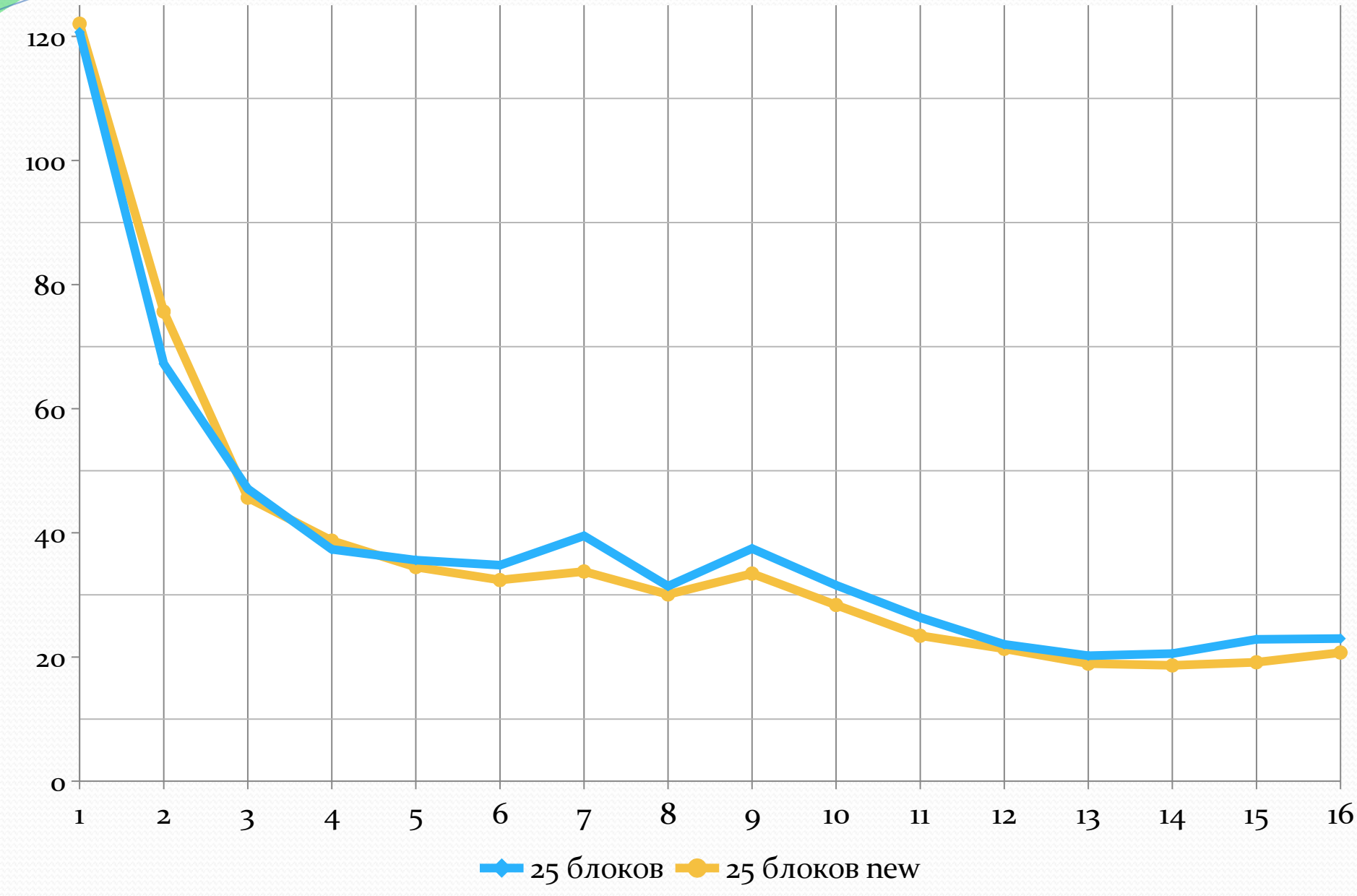
Измерение вычислительных затрат



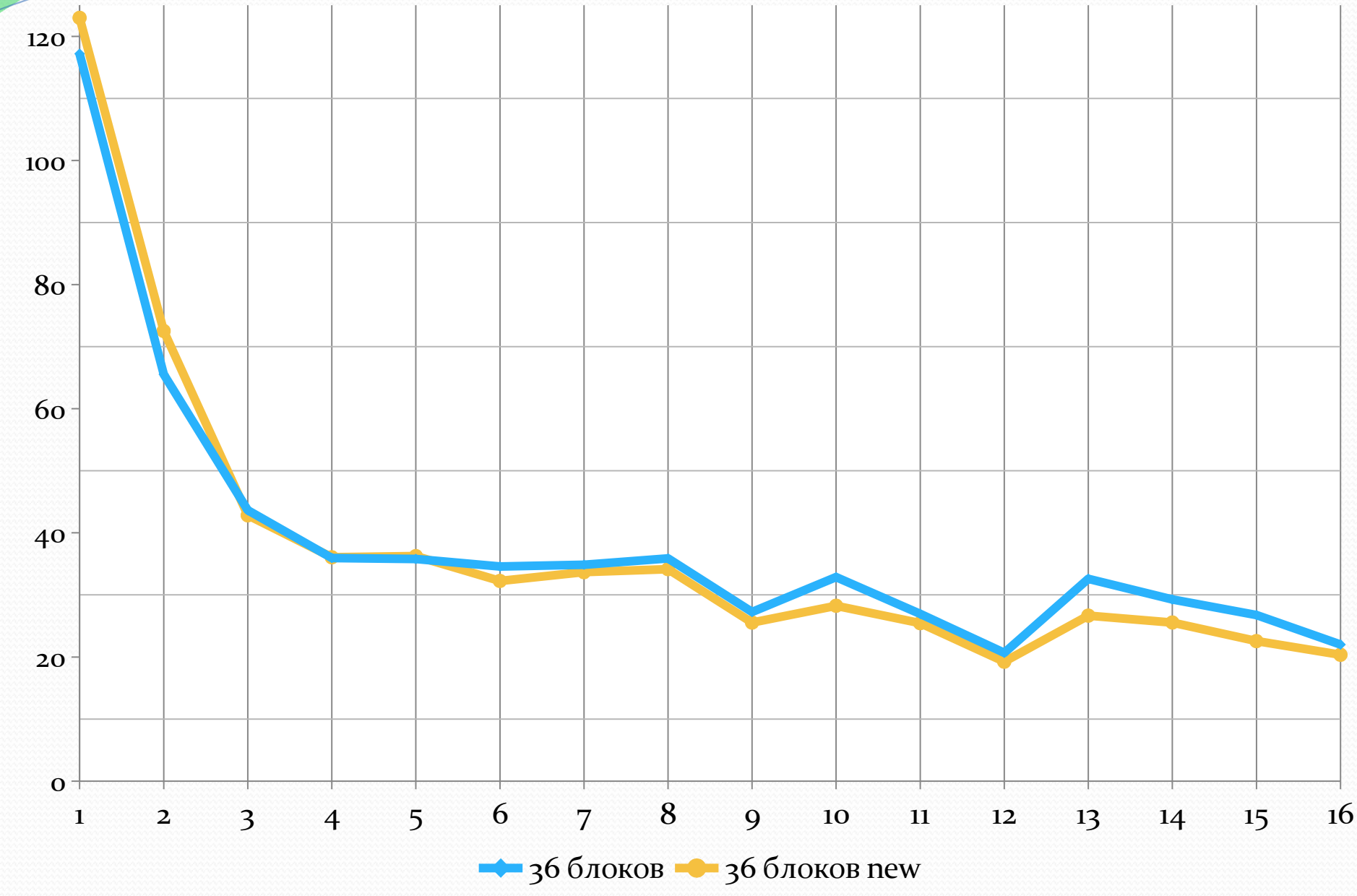
Измерение вычислительных затрат



Измерение вычислительных затрат



Измерение вычислительных затрат



Результаты работы

- Исследованы особенности работы с распределёнными данными
- Разработана библиотека для работы с распределёнными данными
- Получен прирост производительности за счёт организации очереди готовых фрагментов
- Не удалось получить существенного ускорения за счёт увеличения количества фрагментов и асинхронных коммуникаций. Требуется модификация алгоритмов приёма-передачи данных

Спасибо за внимание!