Министерство науки и образования Украины

Харьковский национальный университет им. В.Н. Каразина

Факультет компьютерных наук

Отчет

по лабораторной работе №1

на тему:

**«Применение программной системы ParaLab для организации параллельной реализации методов сортировки»**

Выполнил:

Пойманов Павел

студент группы КС-42

Харьков 2017

# Цель работы

Закрепление знаний по системе ПараЛаб, представляющей собой интегрированную среду для изучения и исследования параллельных алгоритмов решения сложных вычислительных задач.

Обучиться практическим приемам автоматической оценки и визуализации процесса выполнения эксперимента и анализа полученных результатов, позволяющих изучить эффективность использования тех или иных алгоритмов на разных вычислительных системах, сделать выводы о масштабируемости алгоритмов и определить возможное ускорение процесса параллельных вычислений.

# Задачи

1. Разработать последовательные Си-программы рассмотренных методов сортировки. Сделайте замер времени реализации программ.
2. Провести несколько вычислительных экспериментов. Изучить зависимость времени выполнения алгоритма от объема исходных данных и от количества процессоров.
3. Провести серию вычислительных экспериментов. Построить графики зависимости времени выполнения алгоритма (ускорения и эффективности) от объема исходных данных, топологии и количества процессоров.
4. Сделать выводы.

# Листинги последовательных алгоритмов сортировки:

## Сортировка пузырьком

void bubble (int \*a, int n) // функция пузырька

{

int i,j,t;

for (i = 0; i <= n-1; i++)

{

for (j = 0; j <= n-2-i; j++)

{

if (a [j] >a [j+1])

{

t = a [j] ;

a [j] = a [j+1] ;

a [j+1] = t;

}

}

}}

## Сортировка вставками

int i, j, tmp;

for (j=1; j<mass\_size; j++) {

tmp = arr[j];

for (i=0; i<mass\_size-1; i++)

if(mass[i]>mass[i+1]) {

tmp = mass[i];

mass[i] = mass[i+1];

mass[i+1] = tmp;

}

}

## Сортировка Шелла

|  |
| --- |
| void ShellSort(int n, int mass[])  {      int i, j, step;      int tmp;      for (step = n / 2; step > 0; step /= 2)          for (i = step; i < n; i++)          {              tmp = mass[i];              for (j = i; j >= step; j -= step)              {                  if (tmp < mass[j - step])                      mass[j] = mass[j - step];                  else                      break;              }              mass[j] = tmp;      }} Быстрая сортировка void quickSort(int l, int r)  {  int x = a[l + (r - l) / 2];  int i = l;  int j = r;  while(i <= j)  {  while(a[i] < x) i++;  while(a[j] > x) j--;  if(i <= j)  {  swap(a[i], a[j]);  i++;  j--;  }  }  if (i<r)  quickSort(i, r);  if (l<j)  quickSort(l, j);  } |

Таблица 1 Сравнительные показатели производительности различных методов сортировки массивов

|  |  |  |
| --- | --- | --- |
| **Метод сортировки** | **Время сортировки для размера 256, миллисекунд** | **Время сортировки для размера 512, миллисекунд** |
| Вставками (метод простых вставок) | 356 | 1444 |
| Выбором | 509 | 1956 |
| Обменом (пузырек) | 1026 | 4054 |
| Быстрая | 110 | 241 |
| Вставками (Шелла) | 127 | 349 |

# Вычислительные эксперименты в ParaLab

## Эксперимент с пузырьковой сортировкой

Выполнение на топологии Линейка

## 

Рисунок 1 Зависимость времени от объёма данных.

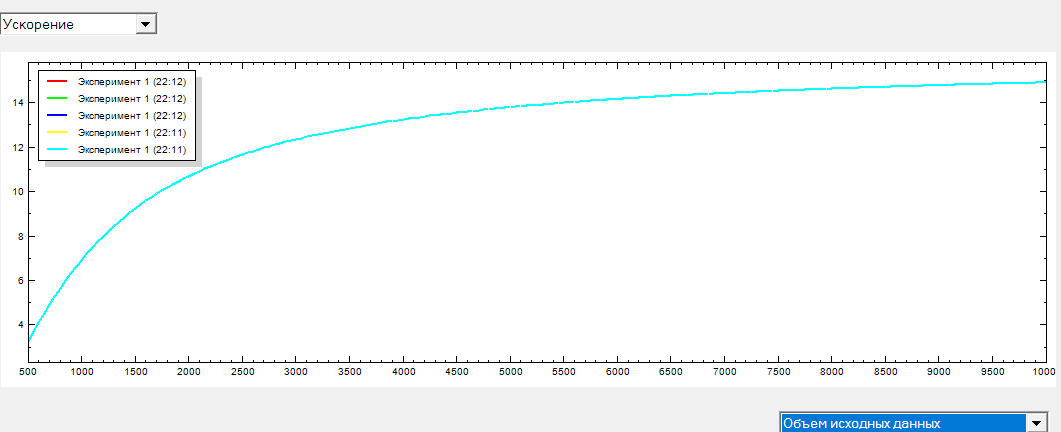


Рисунок 2 Зависимость ускорения от объёма данных

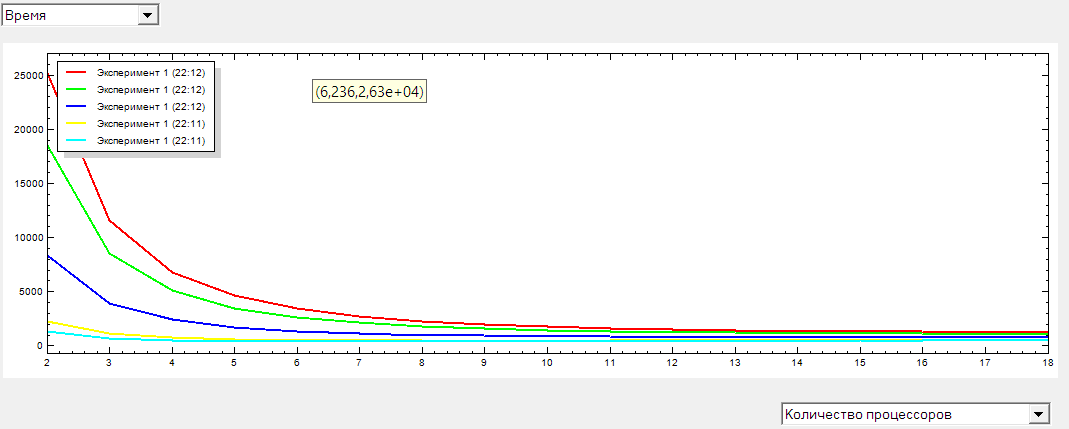


Рисунок 3 Зависимость времени выполнения от кол-ва процессоров:

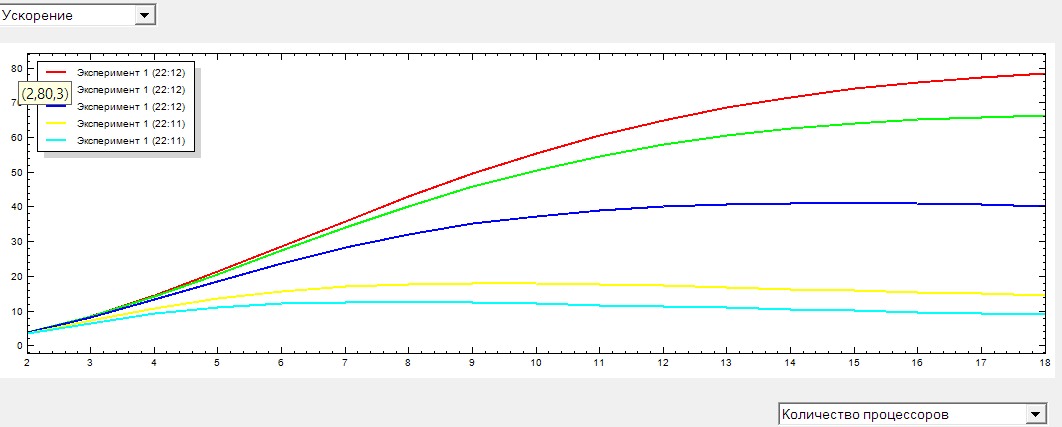


Рисунок 4 Зависимость ускорения от кол-ва процессоров:

Топология гиперкуб:

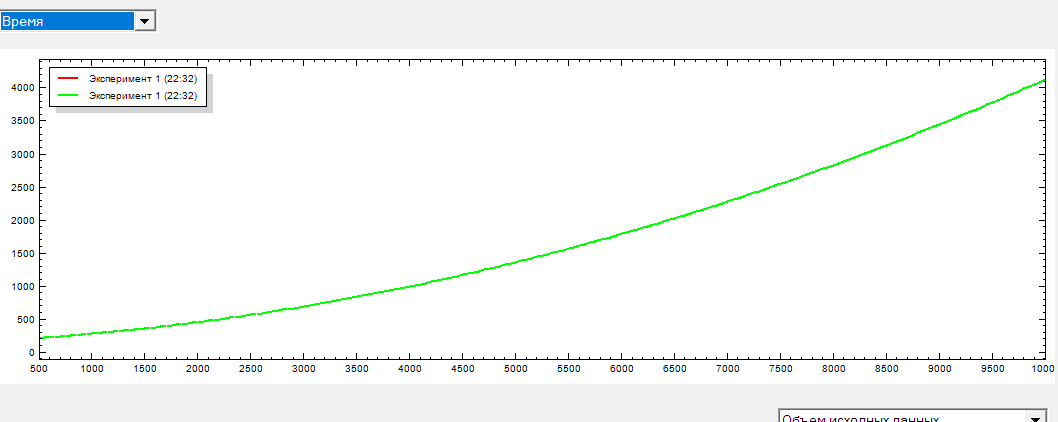


Рисунок 5 График зависимости времени выполнения от объема данных

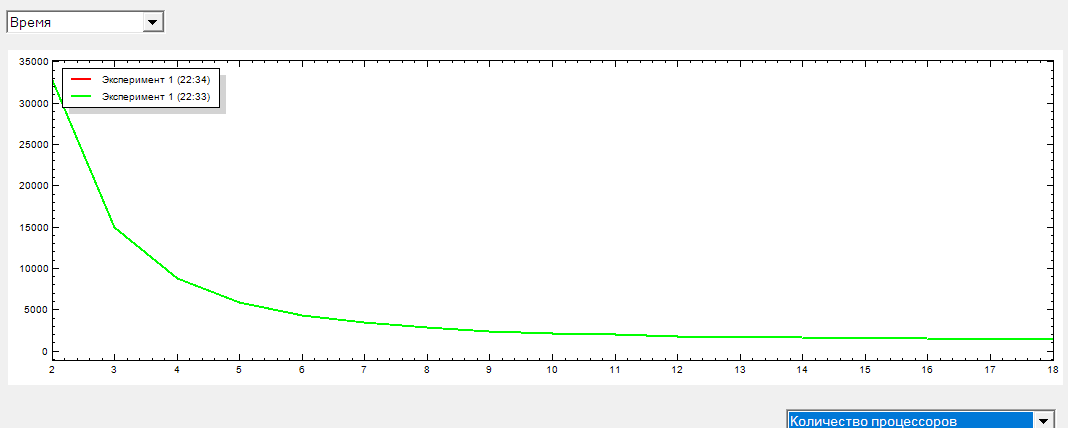


Рисунок 6 График зависимости время от количества процессоров

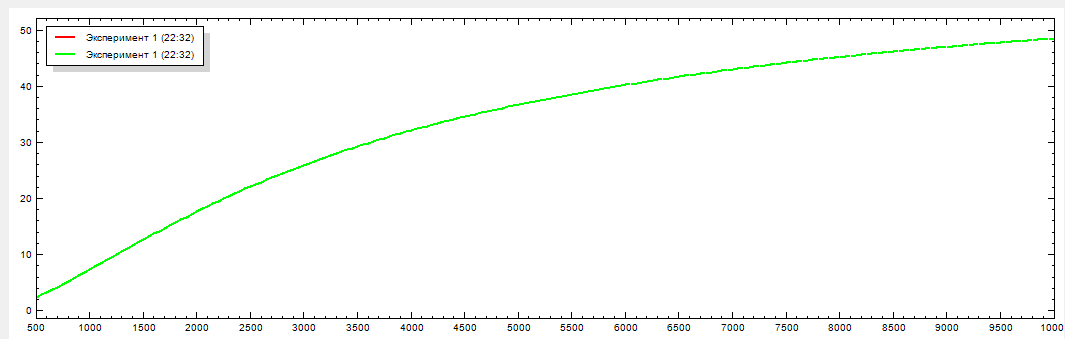


Рисунок 7 График зависимости ускорения от объема данных

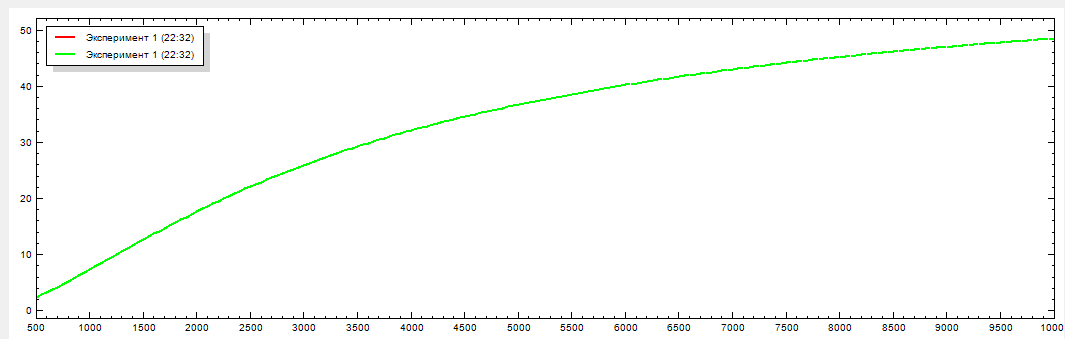


Рисунок 8 Зависимость времени выполнения от кол-ва процессоров

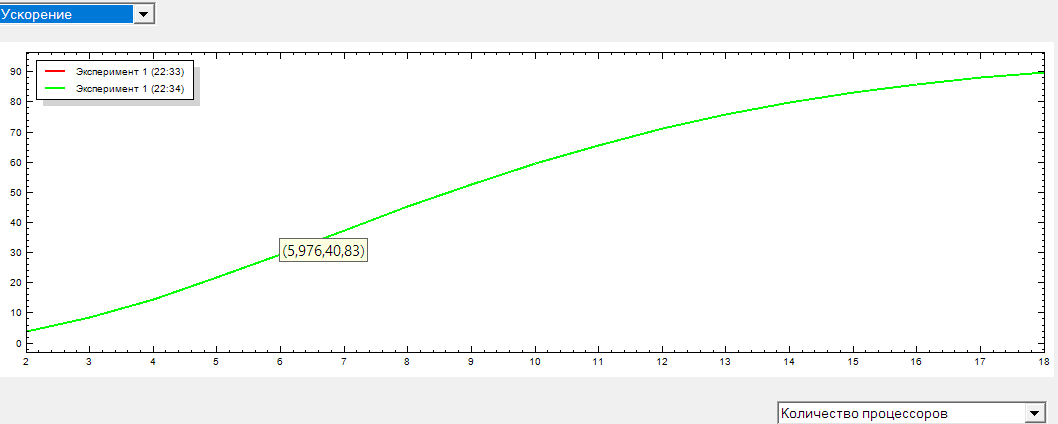


Рисунок 9 Зависимость ускорения от кол-ва процессоров

**Эксперимент с сортировкой Шелла**

Топология гиперкуб

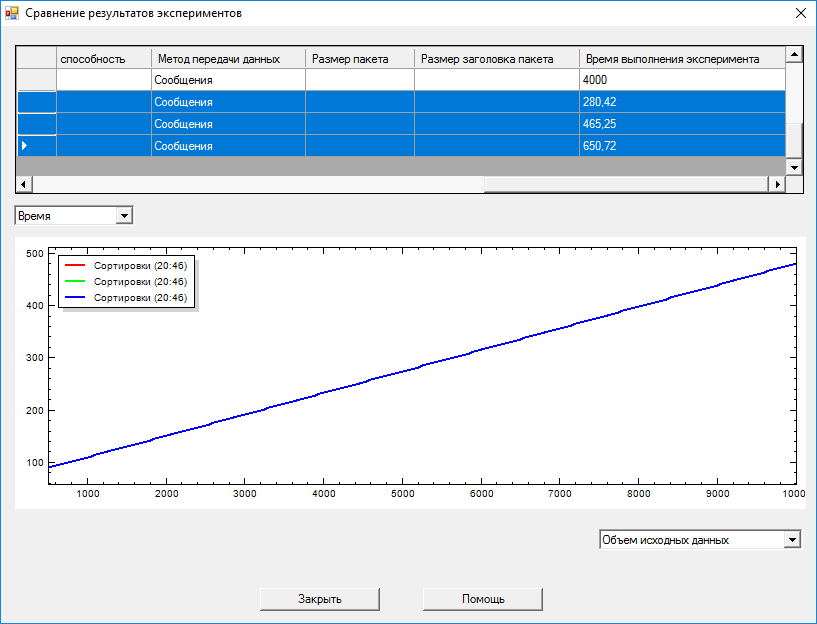


Рисунок 10 График зависимости времени выполнения от объема данных

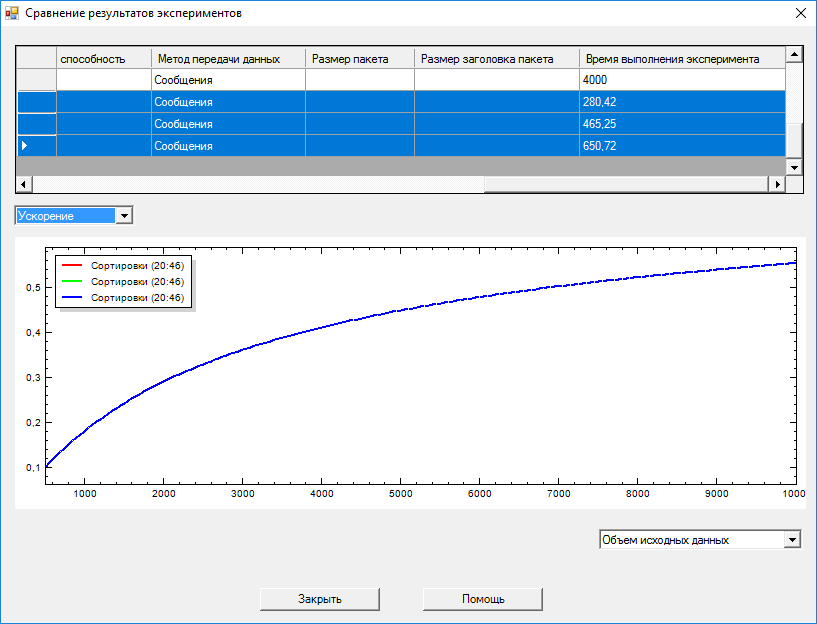


Рисунок 11 График зависимости ускорения от объема данных

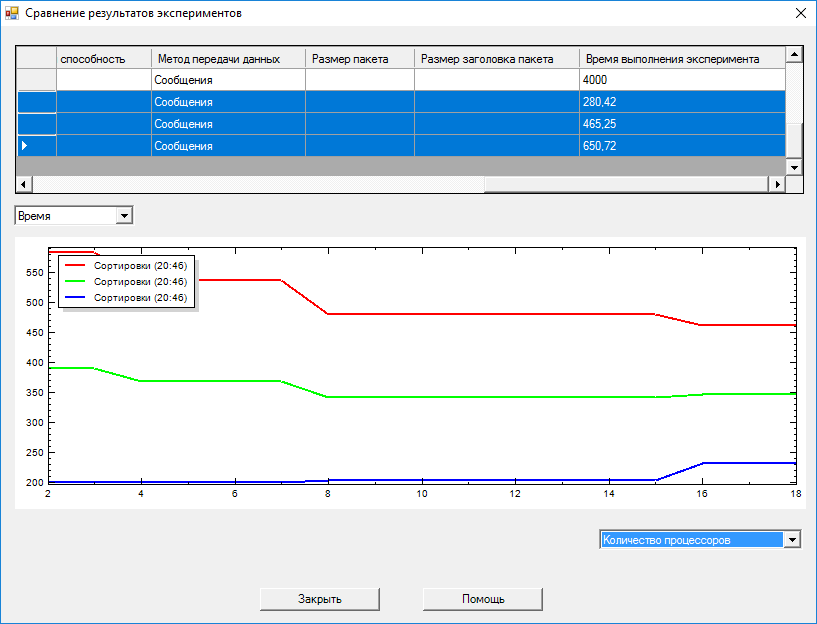


Рисунок 12 Зависимость времени выполнения от кол-ва процессоров

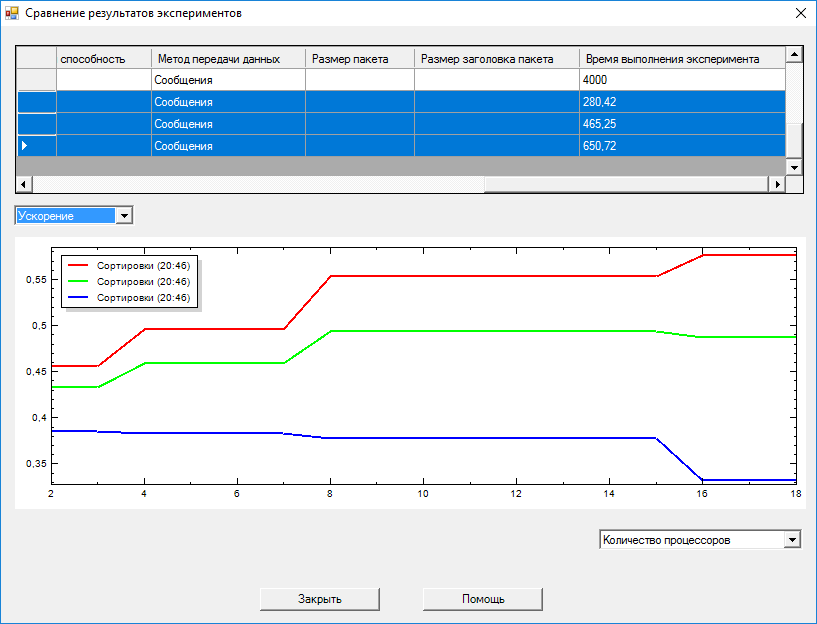


Рисунок 13 Зависимость ускорения от кол-ва процессоров:

**Эксперимент с быстрой сортировкой**

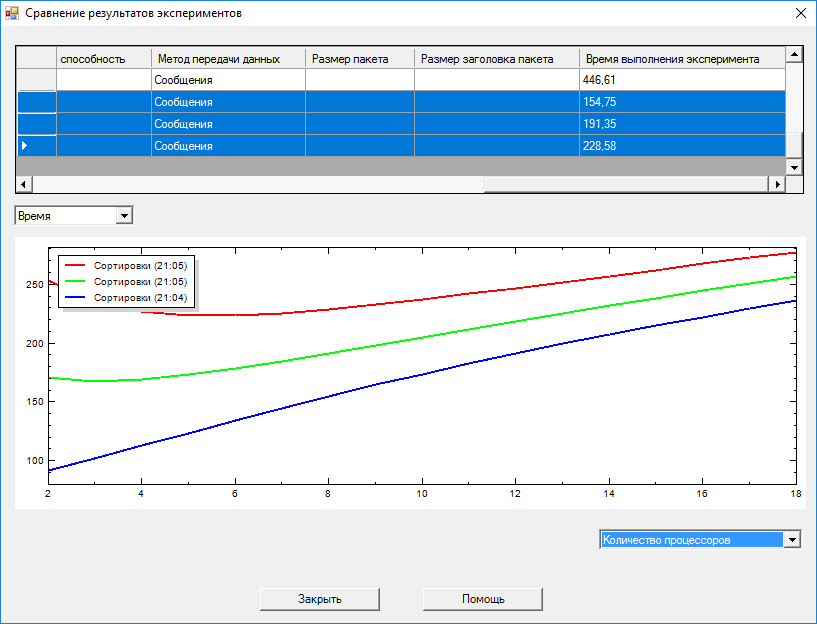


Рисунок 14 Зависимость времени выполнения от кол-ва процессоров:

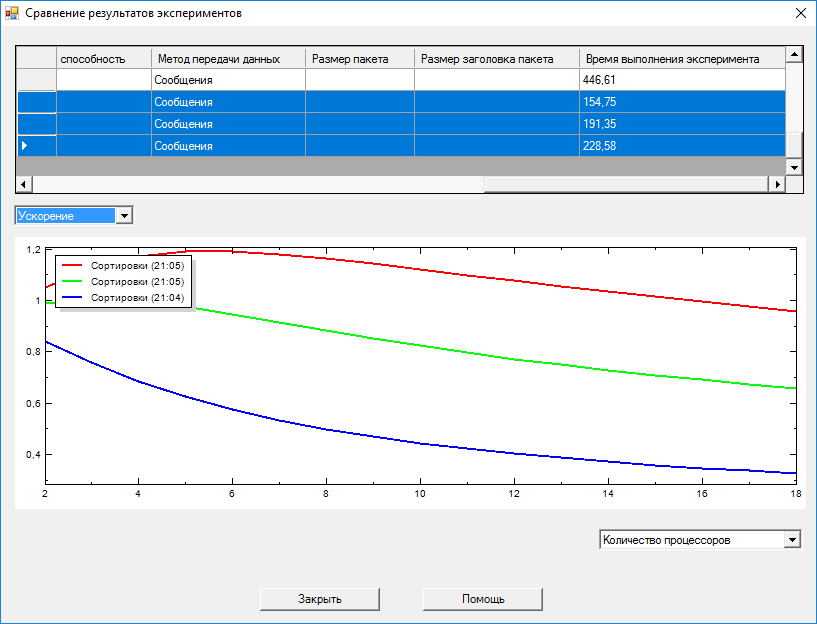


Рисунок 15 Зависимость ускорения от кол-ва процессоров:

# Выводы

После проведения большого числа эксперимента я пришел к таким выводам:

* Использование параллельных методов значительно ускоряет процесс сортировки и дает результаты в 10 раз быстрее чем последовательные аналоги.
* Не все топологии могут быть использованы в параллельных вычислениях.
* Большее количество не всегда дает результат быстрейшего выполнения алгоритма, а при большой латентности сети в некоторых случаях стоит отказаться от параллельного вычисления.