**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

«Linpack»

Содержание

[Введение 4](#_Toc179641813)

[1 Программа linpack 5](#_Toc179641814)

[1.1 Задание 5](#_Toc179641815)

[1.2 Ход работы 5](#_Toc179641816)

[1.2.1 Linpack 5](#_Toc179641817)

[1.2.2 Приоритетами задачи в планировщике 6](#_Toc179641818)

[1.2.3 Привязки к процессору 11](#_Toc179641819)

[1.2.4 Сравнение результаты по 3 сигма критериям 14](#_Toc179641820)

[1.2.5 Влияние на настройки планировщика 15](#_Toc179641821)

[Заключение 21](#_Toc179641822)

[Список использованных источников 21](#_Toc179641823)

Введение

Цель работы – познакомиться с программой linpack и протестировать ее при различных режимах работы ОС.

Для достижения поставленной цели необходимо решить следующие задачи:

* найти и скомпилировать программу linpack;
* протестировать linpack с различными приоритетами задачи в планировщике;
* протестировать linpack с наличием и отсутствием привязки к процессору;
* провести несколько тестов, сравнить результаты по 3 сигма или другим статистическим критериям;
* повлиять на настройки имеющегося планировщика.

# Программа linpack

linpack - программа, которая позволяет оценивать производительность компьютера. У линпака есть популярный тест benchmark, который выполняет решение большой системы линейных уравнений, оценивая, сколько операций с плавающей запятой (FLOPS) может выполнить компьютер за определенный промежуток времени.

FLOPS - (Floating Point Operations Per Second) - единица измерения вычислительной производительности, которая указывает, сколько операций с плавающей запятой может выполнять компьютер или вычислительная система за одну секунду.

1 FLOPS означает, что система может выполнять одну операцию с плавающей запятой в секунду.

## Задание

Cкомпилировать программу linpack и протестировать ее при различных режимах работы ОС.

## Ход работы

### Linpack

Запуск линпака для 200х200



Рисунок 1 – запуск linpack.

Запуск линпака для 1000х1000

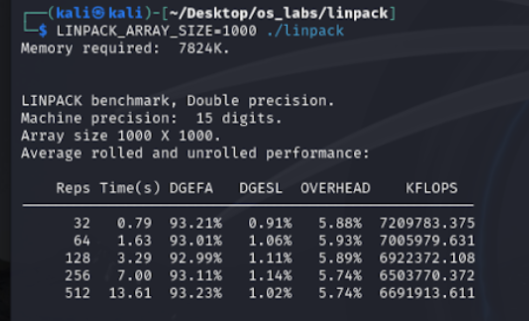


Рисунок 2 – запуск linpack.

### Приоритетами задачи в планировщике

Для того, чтобы изменить приоритеты задач используется команда «sudo nice -n x ./linpack», где х - приоритет от -20 до 19.

Запустим программу с разными приоритетами:

* максимальная приоритетность -20

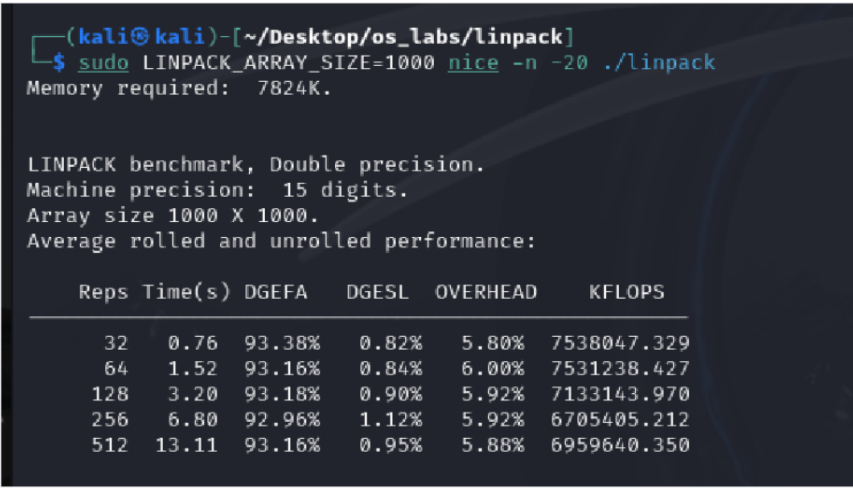


Рисунок 3 – запуск linpack с приоритетностью -20.

* приоритетность -15



Рисунок 4 – запуск linpack с приоритетностью -15.

* приоритетность -10

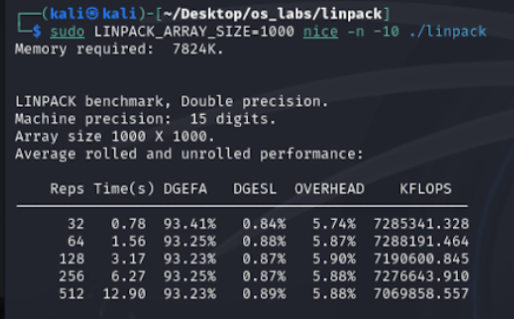


Рисунок 5 – запуск linpack с приоритетностью -10.

* приоритетность -5

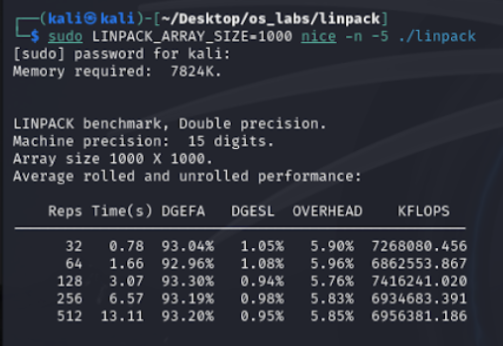


Рисунок 6 – запуск linpack с приоритетностью -5.

* приоритетность 0



Рисунок 7 – запуск linpack с приоритетностью 0.

* приоритетность 5



Рисунок 8 – запуск linpack с приоритетностью 5.

* приоритетность 10

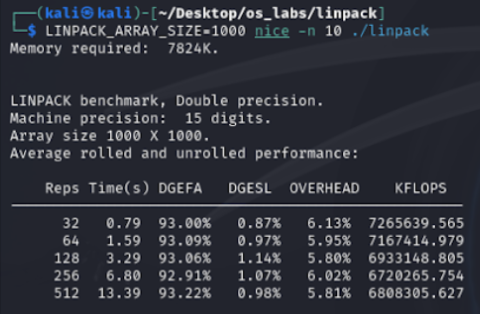


Рисунок 9 – запуск linpack с приоритетностью 10.

* приоритетность 15

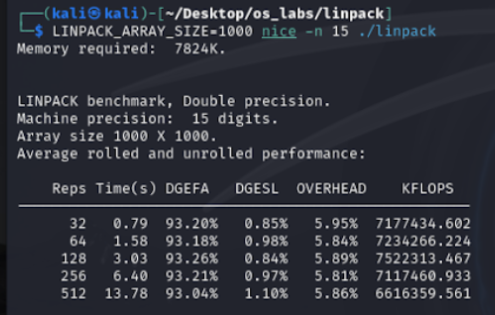


Рисунок 10 – запуск linpack с приоритетностью 15.

* минимальная приоритетность 19

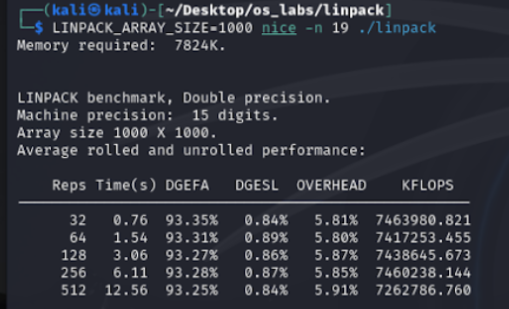


Рисунок 11 – запуск linpack с приоритетностью 19.

### Привязки к процессору

Для того, чтобы привязать или отвязать программу к процессору используется команда «taskset -c x ./linpack», где х – количество ядер.

Запустим программу с привязкой и без привязки к ядру:

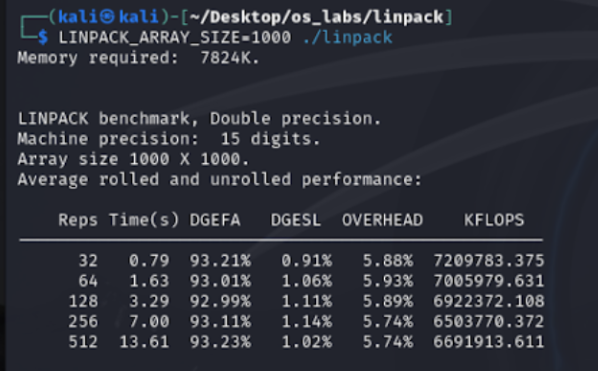


Рисунок 12 – запуск linpack без привязки к ядру.

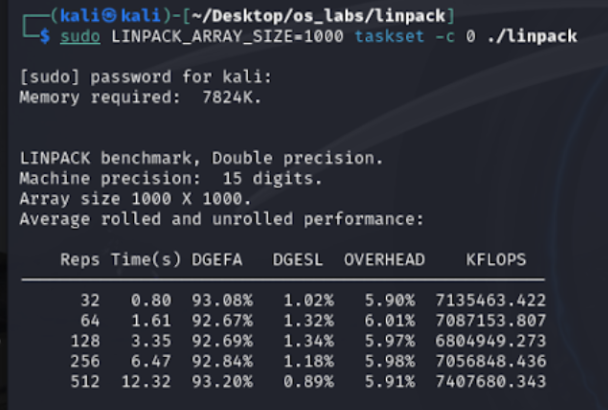


Рисунок 13 – запуск linpack с привязкой к 1 ядру.

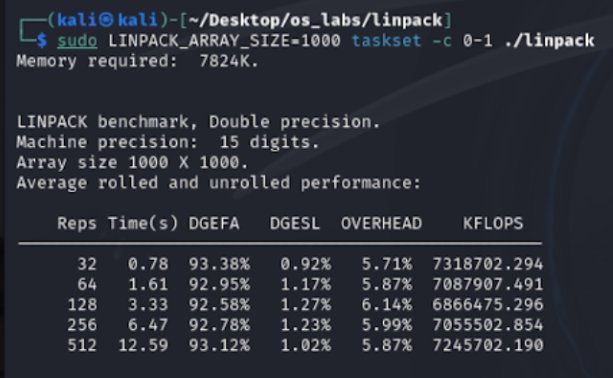


Рисунок 14 – запуск linpack с привязкой к 2 ядрам.

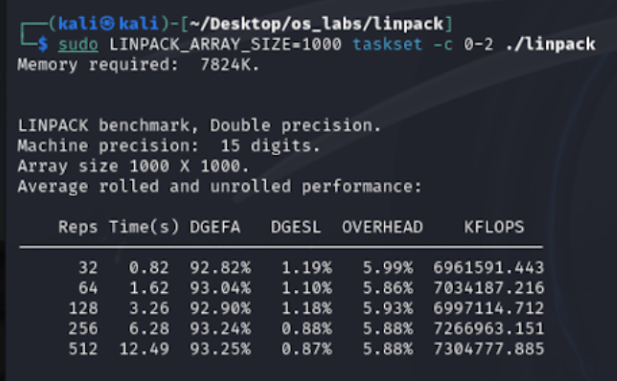


Рисунок 15 – запуск linpack с привязкой к 3 ядрам.

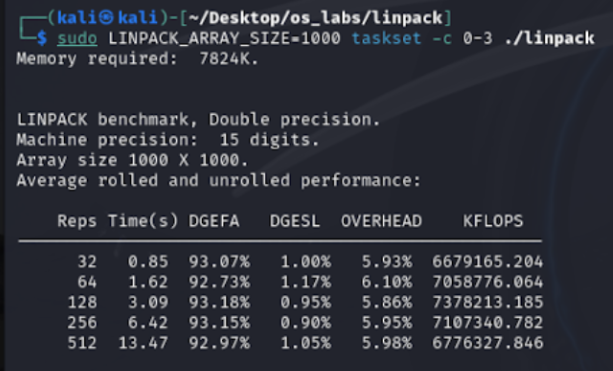


Рисунок 16 – запуск linpack с привязкой к 4 ядрам.

### Сравнение результаты по 3 сигма критериям

3 сигма — это статистический метод, который используется для оценки и анализа вариаций в данных. Он основан на концепции нормального распределения и помогает определить, насколько данные отклоняются от среднего значения.

Для этого мы с начала посчитали средние значение, а потом стандартное отклонение. В итоге мы нашли верхнюю и нижнюю границы.

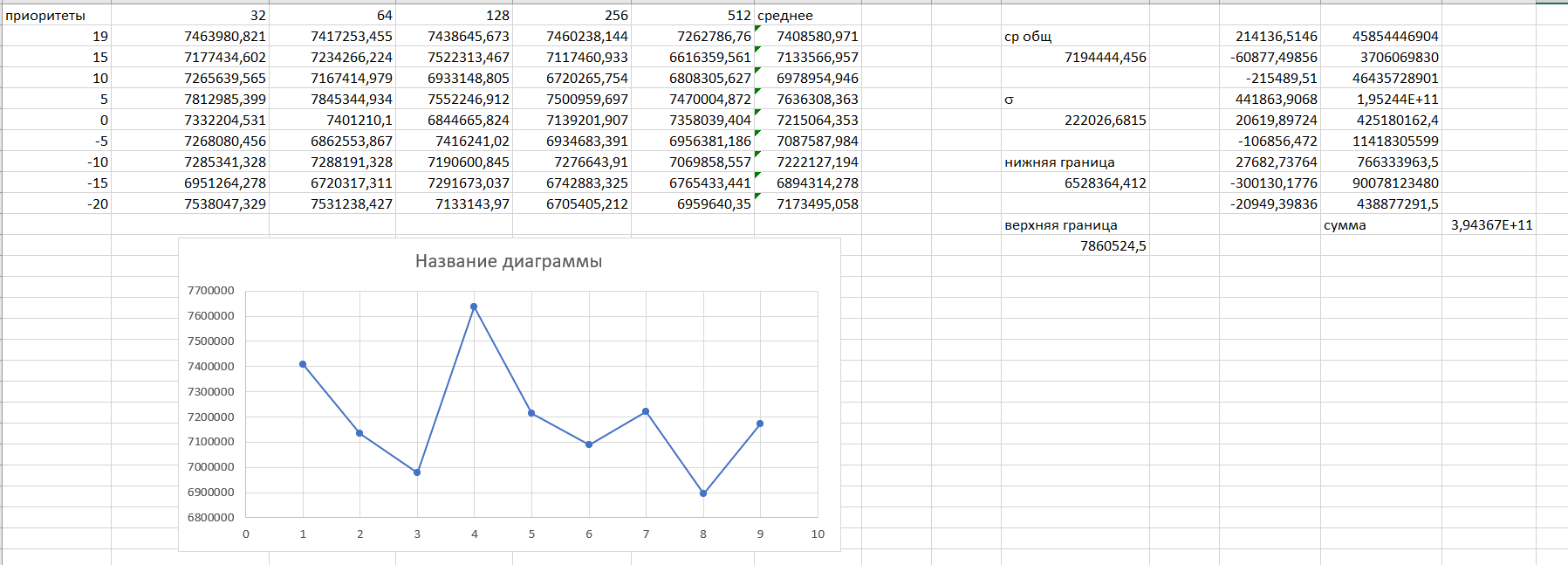


Рисунок 17 – метод 3 сигма для запуска программы с разными приоритетами.

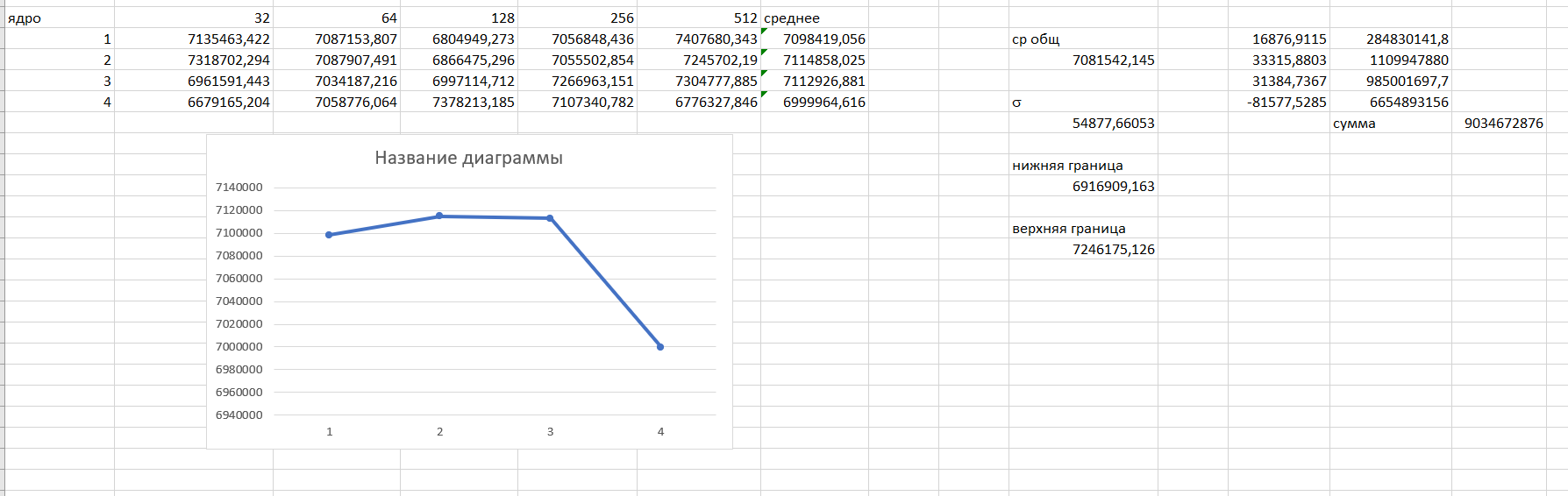


Рисунок 18 – метод 3 сигма для запуска программы с разной привязкой к процессору.

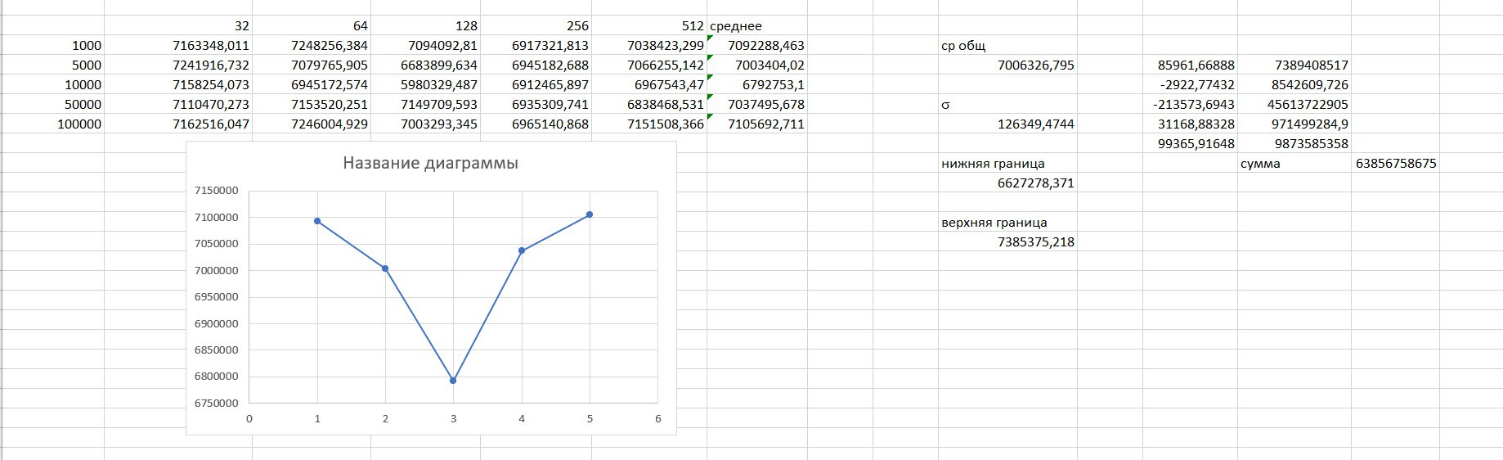


Рисунок 19 – метод 3 сигма для запуска программы с измением параметра kernel.sched\_cfs\_bandwidth\_slice\_us.

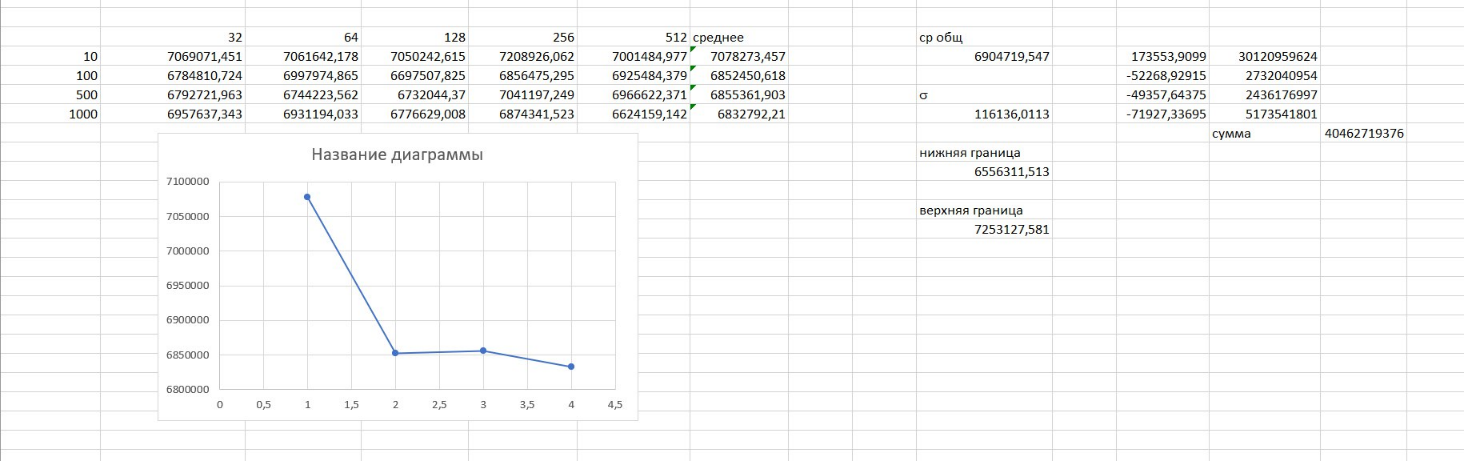


Рисунок 20 – метод 3 сигма для запуска программы с измением параметра kernel.sched\_rr\_timeslice\_ms.

По результата применения данного метода, можно сделать вывод, что все полученные нами измерения находятся в пределах нормального распределения.

### Влияние на настройки планировщика

В начале выясним, какие параметры вообще есть.

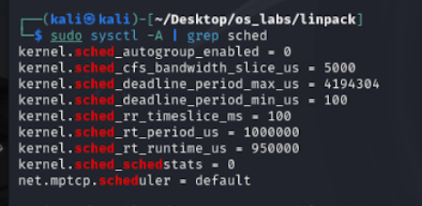


Рисунок 21 – параметры планировщика.

В рамках данной лабораторной работы были изменены параметры kernel.sched\_cfs\_bandwidth\_slice\_us и kernel.sched\_rr\_timeslice\_ms.

kernel.schedcfsbandwidthsliceus:

- Описание: этот параметр определяет максимальную длину временного слота (в микросекундах), который может быть выделен для выполнения задач в рамках планировщика CFS (Completely Fair Scheduler).

- Изменение: увеличение значения может уменьшить количество переключений контекста, что иногда полезно для задач с длительным временем выполнения.

- Границы: обычно от 1 до 10,000,000 микросекунд (1 мс — 10 секунд).

- Оптимальные настройки: 5000 микросекунд может быть хорошим компромиссом.

kernel.schedrrtimeslicems:

- Описание: длительность времени (в миллисекундах), предоставляемая задачам с приоритетом реального времени (RR - Round Robin).

- Изменение: увеличение увеличивает время работы реальных задач, что может помочь в задачах с высоким приоритетом.

- Границы: от 1 до 1000 миллисекунд.

- Оптимальные настройки: обычно 100 мс подходит для большинства приложений.

Изменим параметр kernel.sched\_cfs\_bandwidth\_slice\_us:

* 1000

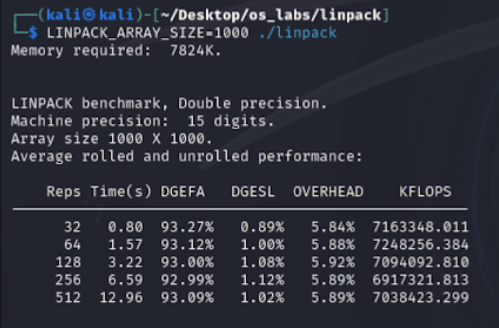


Рисунок 22 – запуск linpack с kernel.sched\_cfs\_bandwidth\_slice\_us=1000.

* 5000

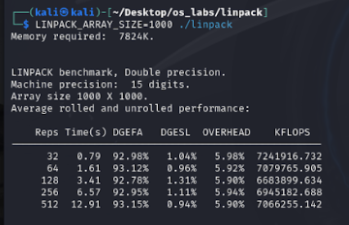


Рисунок 23 – запуск linpack с kernel.sched\_cfs\_bandwidth\_slice\_us=5000.

* 10000

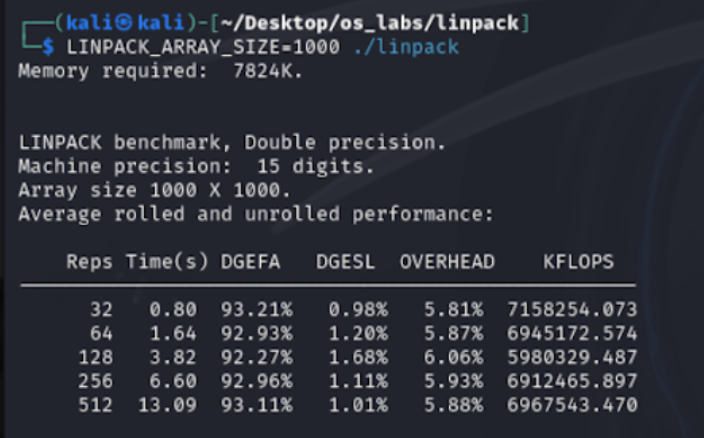


Рисунок 24 – запуск linpack с kernel.sched\_cfs\_bandwidth\_slice\_us=10000.

* 50000

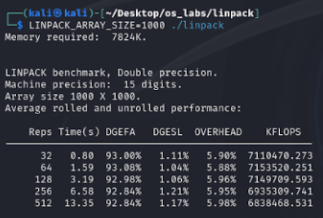


Рисунок 25 – запуск linpack с kernel.sched\_cfs\_bandwidth\_slice\_us=50000.

* 100000

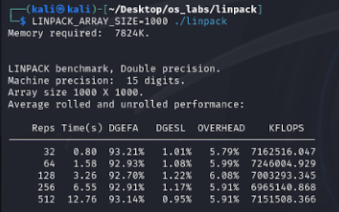


Рисунок 26 – запуск linpack с kernel.sched\_cfs\_bandwidth\_slice\_us=100000.

Изменим параметр kernel.sched\_rr\_timeslice\_ms:

* 10

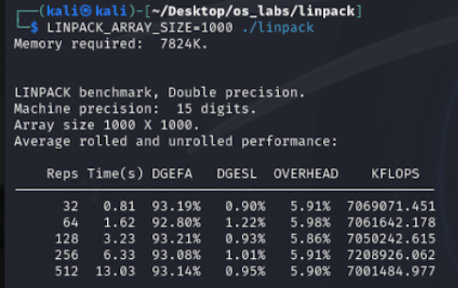


Рисунок 27 – запуск linpack с kernel.sched\_rr\_timeslice\_ms =10.

* 100

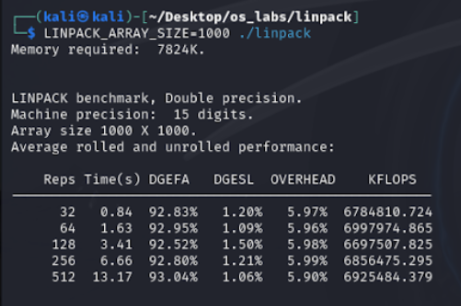


Рисунок 28 – запуск linpack с kernel.sched\_rr\_timeslice\_ms =100.

* 500

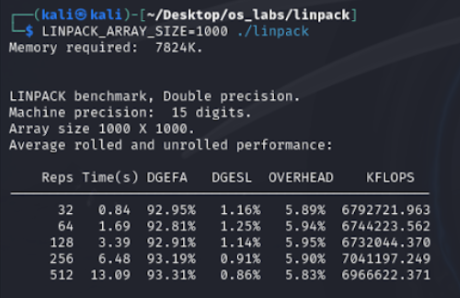


Рисунок 29 – запуск linpack с kernel.sched\_rr\_timeslice\_ms =500.

* 1000

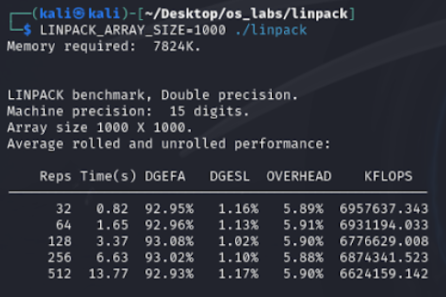


Рисунок 30 – запуск linpack с kernel.sched\_rr\_timeslice\_ms =1000.

Заключение

В ходе данной лабораторной работы мы познакомились с программой linpack. Также мы протестировали ее при различных режимах работы ОС, а именно: с различными приоритетами задач в планировщике, с наличием и отсутствием привязки к процессору, с различными настройками планировщика. Полученный результат был сравнили по 3 сигма. Это позволило закрепить полученные знания и навыки. В ходе выполнения лабораторной работы были выполнены все задачи и достигнуты поставленные цели.

Список использованных источников

1. Эндрю Таненбаум Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с.
2. Архитектура файловой системы FAT // Системный администратор. URL: <https://samag.ru/archive/article/245>.
3. Архитектура файловой системы ext2 // Системный администратор. URL: <https://samag.ru/archive/article/203>.