**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Операционные системы»**

**Тема: «Управление файловой системой»**

Студентки гр. 3312 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Половникова А.С.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимофеев А.В.

Санкт-Петербург

2025

**Цель работы.**

исследовать управление файловой системой.

**Задание 1.2.**

Копирование файла с помощью операций перекрывающегося ввода-вывода

Указания к выполнению.

1. Создайте консольное приложение, которое выполняет: −

открытие/создание файлов (функция Win32 API – CreateFile,

обязательно использовать флаги FILE\_FLAG\_NO\_BUFFERING и

FILE\_FLAG\_OVERLAPPED);

∙ файловый ввод-вывод (функции Win32 API – ReadFileEx,

WriteFileEx) блоками кратными размеру кластера;

∙ ожидание срабатывания вызова функции завершения (функция Win32

API – SleepEx);

∙ измерение продолжительности выполнения операции копирования файла

(функция Win32 API – TimeGetTime). 2. Запустите приложение и проверьте

его работоспособность на копировании файлов разного размера для ситуации

с перекрывающимся выполнением одной операции ввода и одной операции

вывода (для сравнения файлов используйте консольную команду FC).

Выполните эксперимент для разного размера копируемых блоков, постройте

график зависимости скорости копирования от размера блока данных.

Определите оптимальный размер блока данных, при котором скорость

копирования наибольшая. Запротоколируйте результаты в отчет. Дайте свои

комментарии в отчете относительно выполнения функций Win32 API.

3. Произведите замеры времени выполнения приложения для разного

числа перекрывающихся операций ввода и вывода (1, 2, 4, 8, 12, 16), не

забывая проверять работоспособность приложения (консольная команда FC).

По результатам измерений постройте график зависимости и определите число

перекрывающихся операций ввода и вывода, при котором достигается

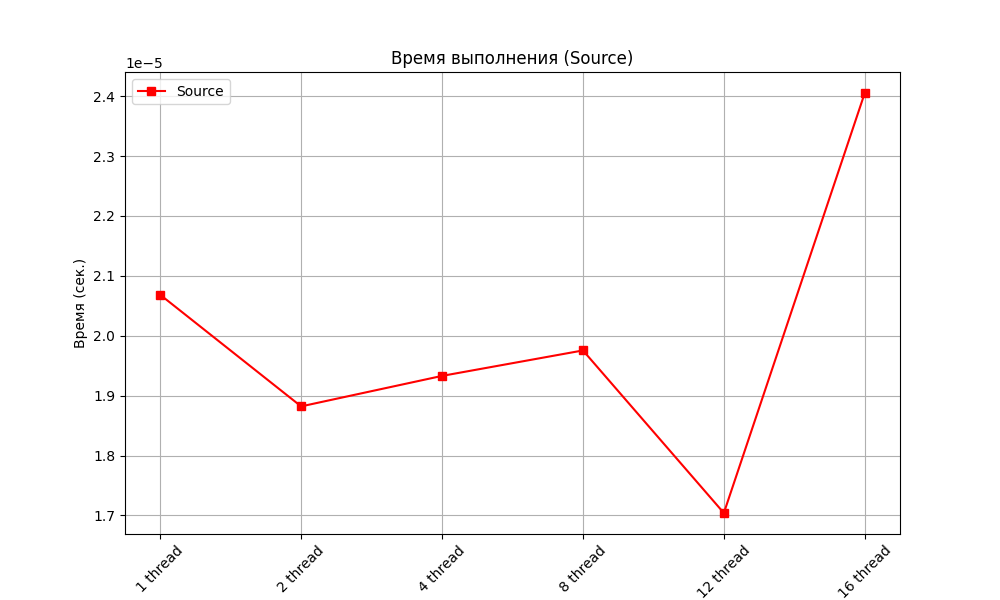
наибольшая скорость копирования файла. Запротоколируйте результаты в

отчет.

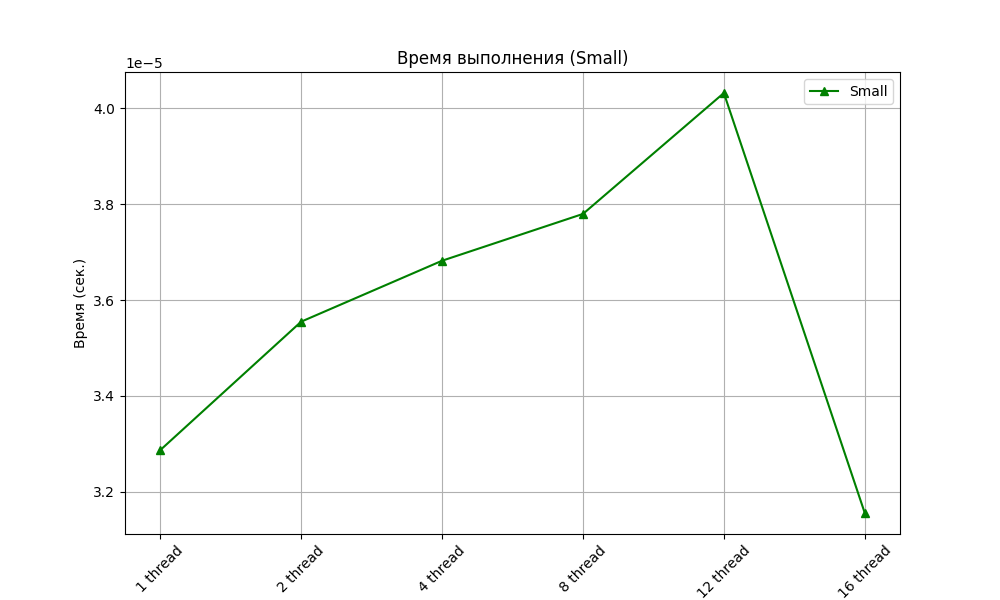
**Анализ данных по количеству потоков (source и small)**

**Данные:**

**source**

****

**small**

****

**Наблюдения:**

**Анализ**

**source:**

* Лучшее время достигается при 12 потоках.
* При увеличении с 1 до 2 потоков наблюдается небольшое улучшение.
* С 4 до 8 — время ухудшается.
* Пик эффективности — на 12 потоках.
* При 16 потоках — резкое ухудшение.

Вывод: существует оптимум в районе 12 потоков, после чего начинается деградация, вероятно, из-за оверхеда управления потоками или гиперпоточности (если ЦП поддерживает SMT).

**small:**

* В целом, увеличение потоков приводит к ухудшению времени, что нетипично.
* При 16 потоках — внезапный скачок улучшения, возможно, из-за кэширования, NUMA, или колебаний нагрузки.
* Это может быть случайной флуктуацией или некорректным измерением.

Вывод: для small — возможно, задача слишком мала, чтобы эффективно распараллеливаться. Потери на синхронизацию/управление потоками превышают выигрыши.

Общие выводы

Параллелизм эффективен только при достаточно большом объеме работы (source).

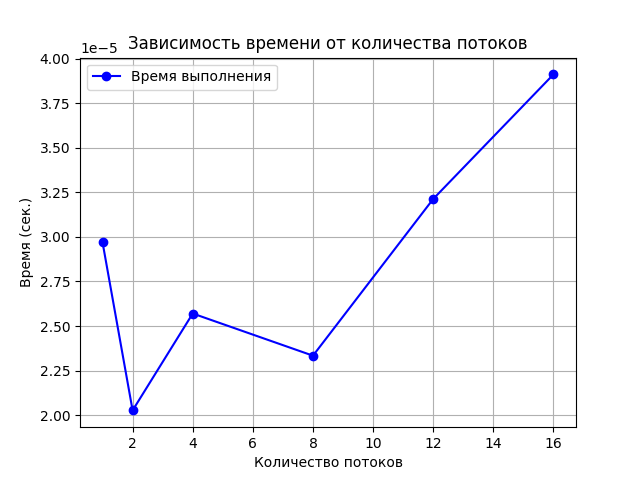
"Small" задачи могут страдать от избыточного распараллеливания.

Оптимальное количество потоков — 8 так как достигается самое выигрышное время

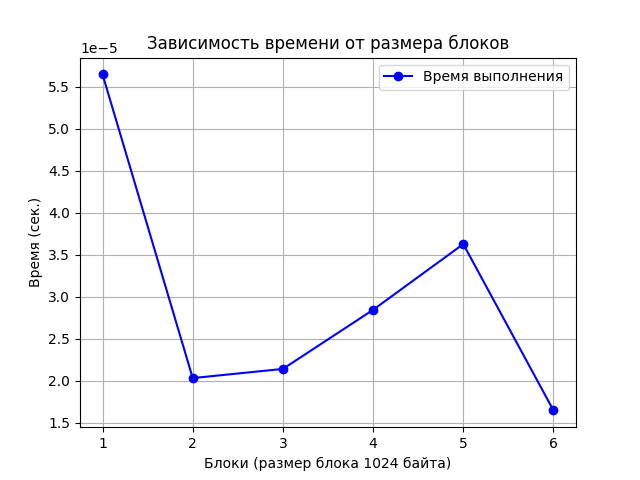
**Анализ данных по размеру блоков (sourceBlock и smallBlock)**

**Данные:**

**sourceBlock**

****

**smallBlock**



**Наблюдения:**

1. **Зависимость времени выполнения от размера блока:**

* Для **sourceBlock** минимальное время выполнения достигается при 5KB (1.7946e-05 секунд), а максимальное — при 1KB (6.5535e-05 секунд).
* Для **smallBlock** минимальное время выполнения достигается при 6KB (1.649e-05 секунд), а максимальное — при 1KB (5.6488e-05 секунд).

1. **Оптимальный размер блока:**

* Для **sourceBlock** оптимальный размер блока — 5KB.
* Для **smallBlock** оптимальный размер блока — 6KB.

1. **Аномалии:**

* В **sourceBlock** при 6KB время выполнения резко возрастает (6.3436e-05 секунд), что может быть связано с ограничениями кэша или памяти.
* В **smallBlock** при 5KB время выполнения также возрастает (3.6289e-05 секунд), что может указывать на неэффективность обработки блоков такого размера.

1. **Возможные причины:**
   * **Эффективность кэширования:** Меньшие блоки (1KB) могут неэффективно использовать кэш процессора, что приводит к увеличению времени выполнения.
   * **Накладные расходы на управление памятью:** При увеличении размера блока растут накладные расходы на выделение и освобождение памяти.
   * **Пиковая производительность:** Для каждого типа задачи существует оптимальный размер блока, при котором достигается максимальная производительность.

**Общие выводы**

**Параллелизм**

Увеличение числа потоков не всегда приводит к ускорению выполнения задачи. Оптимальное количество потоков зависит от характера задачи и доступных ресурсов системы.

Для наших данных оптимальное количество потоков — 2.

**Размер блоков**

Время выполнения зависит от размера блока. Оптимальный размер блока для **sourceBlock** — 5KB, для **smallBlock** — 6KB.

Увеличение размера блока сверх оптимального значения может привести к увеличению времени выполнения из-за накладных расходов.