Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №2по курсу

«Операционные системы»

Группа: М8О-216БВ-24

Студент: Ходаков Павел

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 19.11.25

Москва, 2025

Постановка задачи

Вариант 12.

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы. Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Наложить K раз фильтры эрозии и наращивания на матрицу, состоящую из вещественных чисел. На выходе получается 2 результирующие матрицы

Общий метод и алгоритм решения

Использованные системные вызовы:

. pthread_create()

Создание нового потока выполнения

2. pthread_join()

Ожидание завершения работы потока

3. pthread_mutex_init() / pthread_mutex_destroy()

Инициализация и уничтожение мьютекса

4. pthread_mutex_lock() / pthread_mutex_unlock()

Захват и освобождение мьютекса

5. clock()

Получение времени процессора

6. get_nprocs()

Получение количества доступных процессорных ядер

7. srand() / rand()

Инициализация генератора случайных чисел и генерация случайных значений

8. malloc() / free()

Динамическое выделение и освобождение памяти

9. write()

Вывод данных в файловый дескриптор

10. usleep()

Приостановка выполнения на микросекунды

Основной поток создает рабочие потоки через pthread_create(). Матрица делится на блоки строк для параллельной обработки. Мьютексы защищают общие данные, pthread_join() обеспечивает завершение

Код программы

https://github.com/n3tw4lk3r/MAI-OS-labs/tree/main/lab_2

Протокол работы программы

См. github

./performance_test.sh

Strace:

```
execve("./build/lab_2", ["./build/lab_2", "1000", "3", "4", "0"], 0x7ffd0175da90 /* 70 vars */) = 0
brk(NULL)                               = 0x558d97ef4000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=152743, ...}) = 0
mmap(NULL, 152743, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe514b95000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1100400, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe514b93000
mmap(NULL, 1102152, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe514a85000
mmap(0x7fe514a94000, 569344, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xf000) = 0x7fe514a94000
mmap(0x7fe514b1f000, 466944, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x9a000) = 0x7fe514b1f000
mmap(0x7fe514b91000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10b000) = 0x7fe514b91000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe514800000
mmap(0x7fe514824000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x24000) = 0x7fe514824000
mmap(0x7fe514996000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x196000) = 0x7fe514996000
mmap(0x7fe514a05000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x204000) = 0x7fe514a05000
mmap(0x7fe514a0b000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7fe514a0b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe514a82000
```

```
arch_prctl(ARCH_SET_FS, 0x7fe514a82740) = 0
set_tid_address(0x7fe514a82a10)         = 167780
set_robust_list(0x7fe514a82a20, 24)     = 0
rseq(0x7fe514a82680, 0x20, 0, 0x53053053) = 0
mprotect(0x7fe514a05000, 16384, PROT_READ) = 0
mprotect(0x7fe514b91000, 4096, PROT_READ) = 0
mprotect(0x558d63da4000, 4096, PROT_READ) = 0
mprotect(0x7fe514bfc000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\x6f\x8f\x11\x45\xe1\x00\x70\xdb", 8, GRND_NONBLOCK) = 8
munmap(0x7fe514b95000, 152743)          = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x4), ...}) = 0
brk(NULL)                               = 0x558d97ef4000
brk(0x558d97f15000)                     = 0x558d97f15000
write(1, "=== Execution Parameters ===\n", 29) = 29
write(1, "Matrix size: 1000x1000\n", 23) = 23
write(1, "Number of iterations K: 3\n", 26) = 26
write(1, "Maximum threads: 4\n", 19)    = 19
openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3
read(3, "0-11\n", 1024)                 = 5
close(3)                                = 0
write(1, "CPU cores: 12\n", 14)         = 14
write(1, "Filter 1: erosion\n", 18)     = 18
write(1, "Filter 2: dilation\n", 19)    = 19
brk(0x558d97f36000)                     = 0x558d97f36000
brk(0x558d97f57000)                     = 0x558d97f57000
brk(0x558d97f79000)                     = 0x558d97f79000
brk(0x558d97f9a000)                     = 0x558d97f9a000
brk(0x558d97fbb000)                     = 0x558d97fbb000
brk(0x558d97fdd000)                     = 0x558d97fdd000
brk(0x558d97ffe000)                     = 0x558d97ffe000
brk(0x558d9801f000)                     = 0x558d9801f000
brk(0x558d98040000)                     = 0x558d98040000
brk(0x558d98062000)                     = 0x558d98062000
brk(0x558d98083000)                     = 0x558d98083000
brk(0x558d980a4000)                     = 0x558d980a4000
brk(0x558d980c5000)                     = 0x558d980c5000
brk(0x558d980e7000)                     = 0x558d980e7000
brk(0x558d98108000)                     = 0x558d98108000
brk(0x558d98129000)                     = 0x558d98129000
brk(0x558d9814b000)                     = 0x558d9814b000
brk(0x558d9816c000)                     = 0x558d9816c000
brk(0x558d9818d000)                     = 0x558d9818d000
brk(0x558d981ae000)                     = 0x558d981ae000
brk(0x558d981d0000)                     = 0x558d981d0000
brk(0x558d981f1000)                     = 0x558d981f1000
```

brk(0x558d98212000) = 0x558d98212000
brk(0x558d98233000) = 0x558d98233000
brk(0x558d98255000) = 0x558d98255000
brk(0x558d98276000) = 0x558d98276000
brk(0x558d98297000) = 0x558d98297000
brk(0x558d982b8000) = 0x558d982b8000
brk(0x558d982da000) = 0x558d982da000
brk(0x558d982fb000) = 0x558d982fb000
brk(0x558d9831c000) = 0x558d9831c000
brk(0x558d9833e000) = 0x558d9833e000
brk(0x558d9835f000) = 0x558d9835f000
brk(0x558d98380000) = 0x558d98380000
brk(0x558d983a1000) = 0x558d983a1000
brk(0x558d983c3000) = 0x558d983c3000
brk(0x558d983e4000) = 0x558d983e4000
brk(0x558d98405000) = 0x558d98405000
brk(0x558d98426000) = 0x558d98426000
brk(0x558d98448000) = 0x558d98448000
brk(0x558d98469000) = 0x558d98469000
brk(0x558d9848a000) = 0x558d9848a000
brk(0x558d984ac000) = 0x558d984ac000
brk(0x558d984cd000) = 0x558d984cd000
brk(0x558d984ee000) = 0x558d984ee000
brk(0x558d9850f000) = 0x558d9850f000
brk(0x558d98531000) = 0x558d98531000
brk(0x558d98552000) = 0x558d98552000
brk(0x558d98573000) = 0x558d98573000
brk(0x558d98594000) = 0x558d98594000
brk(0x558d985b6000) = 0x558d985b6000
brk(0x558d985d7000) = 0x558d985d7000
brk(0x558d985f8000) = 0x558d985f8000
brk(0x558d9861a000) = 0x558d9861a000
brk(0x558d9863b000) = 0x558d9863b000
brk(0x558d9865c000) = 0x558d9865c000
brk(0x558d9867d000) = 0x558d9867d000
brk(0x558d9869f000) = 0x558d9869f000
brk(0x558d986c0000) = 0x558d986c0000
brk(0x558d986e1000) = 0x558d986e1000
brk(0x558d98702000) = 0x558d98702000
brk(0x558d98724000) = 0x558d98724000
brk(0x558d98745000) = 0x558d98745000
brk(0x558d98766000) = 0x558d98766000
brk(0x558d98787000) = 0x558d98787000
brk(0x558d987a9000) = 0x558d987a9000
brk(0x558d987ca000) = 0x558d987ca000
brk(0x558d987eb000) = 0x558d987eb000

brk(0x558d9880d000) = 0x558d9880d000
brk(0x558d9882e000) = 0x558d9882e000
brk(0x558d9884f000) = 0x558d9884f000
brk(0x558d98870000) = 0x558d98870000
brk(0x558d98892000) = 0x558d98892000
brk(0x558d988b3000) = 0x558d988b3000
brk(0x558d988d4000) = 0x558d988d4000
brk(0x558d988f5000) = 0x558d988f5000
brk(0x558d98917000) = 0x558d98917000
brk(0x558d98938000) = 0x558d98938000
brk(0x558d98959000) = 0x558d98959000
brk(0x558d9897b000) = 0x558d9897b000
brk(0x558d9899c000) = 0x558d9899c000
brk(0x558d989bd000) = 0x558d989bd000
brk(0x558d989de000) = 0x558d989de000
brk(0x558d98a00000) = 0x558d98a00000
brk(0x558d98a21000) = 0x558d98a21000
brk(0x558d98a42000) = 0x558d98a42000
brk(0x558d98a63000) = 0x558d98a63000
brk(0x558d98a85000) = 0x558d98a85000
brk(0x558d98aa6000) = 0x558d98aa6000
brk(0x558d98ac7000) = 0x558d98ac7000
brk(0x558d98ae8000) = 0x558d98ae8000
brk(0x558d98b0a000) = 0x558d98b0a000
brk(0x558d98b2b000) = 0x558d98b2b000
brk(0x558d98b4c000) = 0x558d98b4c000
brk(0x558d98b6e000) = 0x558d98b6e000
brk(0x558d98b8f000) = 0x558d98b8f000
brk(0x558d98bb0000) = 0x558d98bb0000
brk(0x558d98bd1000) = 0x558d98bd1000
brk(0x558d98bf3000) = 0x558d98bf3000
brk(0x558d98c14000) = 0x558d98c14000
brk(0x558d98c35000) = 0x558d98c35000
brk(0x558d98c56000) = 0x558d98c56000
brk(0x558d98c78000) = 0x558d98c78000
brk(0x558d98c99000) = 0x558d98c99000
brk(0x558d98cba000) = 0x558d98cba000
brk(0x558d98cdc000) = 0x558d98cdc000
brk(0x558d98cfd000) = 0x558d98cfd000
brk(0x558d98d1e000) = 0x558d98d1e000
brk(0x558d98d3f000) = 0x558d98d3f000
brk(0x558d98d61000) = 0x558d98d61000
brk(0x558d98d82000) = 0x558d98d82000
brk(0x558d98da3000) = 0x558d98da3000
brk(0x558d98dc4000) = 0x558d98dc4000
brk(0x558d98de6000) = 0x558d98de6000

brk(0x558d98e07000) = 0x558d98e07000
brk(0x558d98e28000) = 0x558d98e28000
brk(0x558d98e49000) = 0x558d98e49000
brk(0x558d98e6b000) = 0x558d98e6b000
brk(0x558d98e8c000) = 0x558d98e8c000
brk(0x558d98ead000) = 0x558d98ead000
brk(0x558d98ecf000) = 0x558d98ecf000
brk(0x558d98ef0000) = 0x558d98ef0000
brk(0x558d98f11000) = 0x558d98f11000
brk(0x558d98f32000) = 0x558d98f32000
brk(0x558d98f54000) = 0x558d98f54000
brk(0x558d98f75000) = 0x558d98f75000
brk(0x558d98f96000) = 0x558d98f96000
brk(0x558d98fb7000) = 0x558d98fb7000
brk(0x558d98fd9000) = 0x558d98fd9000
brk(0x558d98ffa000) = 0x558d98ffa000
brk(0x558d9901b000) = 0x558d9901b000
brk(0x558d9903d000) = 0x558d9903d000
brk(0x558d9905e000) = 0x558d9905e000
brk(0x558d9907f000) = 0x558d9907f000
brk(0x558d990a0000) = 0x558d990a0000
brk(0x558d990c2000) = 0x558d990c2000
brk(0x558d990e3000) = 0x558d990e3000
brk(0x558d99104000) = 0x558d99104000
brk(0x558d99125000) = 0x558d99125000
brk(0x558d99147000) = 0x558d99147000
brk(0x558d99168000) = 0x558d99168000
brk(0x558d99189000) = 0x558d99189000
brk(0x558d991aa000) = 0x558d991aa000
brk(0x558d991cc000) = 0x558d991cc000
brk(0x558d991ed000) = 0x558d991ed000
brk(0x558d9920e000) = 0x558d9920e000
brk(0x558d99230000) = 0x558d99230000
brk(0x558d99251000) = 0x558d99251000
brk(0x558d99272000) = 0x558d99272000
brk(0x558d99293000) = 0x558d99293000
brk(0x558d992b5000) = 0x558d992b5000
brk(0x558d992d6000) = 0x558d992d6000
brk(0x558d992f7000) = 0x558d992f7000
brk(0x558d99318000) = 0x558d99318000
brk(0x558d9933a000) = 0x558d9933a000
brk(0x558d9935b000) = 0x558d9935b000
brk(0x558d9937c000) = 0x558d9937c000
brk(0x558d9939e000) = 0x558d9939e000
brk(0x558d993bf000) = 0x558d993bf000
brk(0x558d993e0000) = 0x558d993e0000

```
brk(0x558d99401000)                = 0x558d99401000
brk(0x558d99423000)                = 0x558d99423000
brk(0x558d99444000)                = 0x558d99444000
brk(0x558d99465000)                = 0x558d99465000
brk(0x558d99486000)                = 0x558d99486000
brk(0x558d994a8000)                = 0x558d994a8000
brk(0x558d994c9000)                = 0x558d994c9000
brk(0x558d994ea000)                = 0x558d994ea000
brk(0x558d9950b000)                = 0x558d9950b000
brk(0x558d9952d000)                = 0x558d9952d000
brk(0x558d9954e000)                = 0x558d9954e000
brk(0x558d9956f000)                = 0x558d9956f000
brk(0x558d99591000)                = 0x558d99591000
brk(0x558d995b2000)                = 0x558d995b2000
brk(0x558d995d3000)                = 0x558d995d3000
brk(0x558d995f4000)                = 0x558d995f4000
write(1, "\n", 1)                  = 1
write(1, "=== Starting Processing ===\n", 28) = 28
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=24871199}) = 0
brk(0x558d99616000)                = 0x558d99616000
brk(0x558d99637000)                = 0x558d99637000
brk(0x558d99658000)                = 0x558d99658000
brk(0x558d9967a000)                = 0x558d9967a000
brk(0x558d9969b000)                = 0x558d9969b000
brk(0x558d996bc000)                = 0x558d996bc000
brk(0x558d996dd000)                = 0x558d996dd000
brk(0x558d996ff000)                = 0x558d996ff000
brk(0x558d99720000)                = 0x558d99720000
brk(0x558d99741000)                = 0x558d99741000
brk(0x558d99762000)                = 0x558d99762000
brk(0x558d99784000)                = 0x558d99784000
brk(0x558d997a5000)                = 0x558d997a5000
brk(0x558d997c6000)                = 0x558d997c6000
brk(0x558d997e8000)                = 0x558d997e8000
brk(0x558d99809000)                = 0x558d99809000
brk(0x558d9982a000)                = 0x558d9982a000
brk(0x558d9984b000)                = 0x558d9984b000
brk(0x558d9986d000)                = 0x558d9986d000
brk(0x558d9988e000)                = 0x558d9988e000
brk(0x558d998af000)                = 0x558d998af000
brk(0x558d998d0000)                = 0x558d998d0000
brk(0x558d998f2000)                = 0x558d998f2000
brk(0x558d99913000)                = 0x558d99913000
brk(0x558d99934000)                = 0x558d99934000
brk(0x558d99955000)                = 0x558d99955000
brk(0x558d99977000)                = 0x558d99977000
```

```
brk(0x558d99998000)                = 0x558d99998000
brk(0x558d999b9000)                = 0x558d999b9000
brk(0x558d999db000)                = 0x558d999db000
brk(0x558d999fc000)                = 0x558d999fc000
brk(0x558d99a1d000)                = 0x558d99a1d000
brk(0x558d99a3e000)                = 0x558d99a3e000
brk(0x558d99a60000)                = 0x558d99a60000
brk(0x558d99a81000)                = 0x558d99a81000
brk(0x558d99aa2000)                = 0x558d99aa2000
brk(0x558d99ac3000)                = 0x558d99ac3000
brk(0x558d99ae5000)                = 0x558d99ae5000
brk(0x558d99b06000)                = 0x558d99b06000
brk(0x558d99b27000)                = 0x558d99b27000
brk(0x558d99b49000)                = 0x558d99b49000
brk(0x558d99b6a000)                = 0x558d99b6a000
brk(0x558d99b8b000)                = 0x558d99b8b000
brk(0x558d99bac000)                = 0x558d99bac000
brk(0x558d99bce000)                = 0x558d99bce000
brk(0x558d99bef000)                = 0x558d99bef000
brk(0x558d99c10000)                = 0x558d99c10000
brk(0x558d99c31000)                = 0x558d99c31000
brk(0x558d99c53000)                = 0x558d99c53000
brk(0x558d99c74000)                = 0x558d99c74000
brk(0x558d99c95000)                = 0x558d99c95000
brk(0x558d99cb6000)                = 0x558d99cb6000
brk(0x558d99cd8000)                = 0x558d99cd8000
brk(0x558d99cf9000)                = 0x558d99cf9000
brk(0x558d99d1a000)                = 0x558d99d1a000
brk(0x558d99d3c000)                = 0x558d99d3c000
brk(0x558d99d5d000)                = 0x558d99d5d000
brk(0x558d99d7e000)                = 0x558d99d7e000
brk(0x558d99d9f000)                = 0x558d99d9f000
rt_sigaction(SIGRT_1, {sa_handler=0x7fe5148940b0, sa_mask=[], sa_flags=SA_RESTORER|
SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fe51483e540}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7fe513fff000
madvise(0x7fe513fff000, 4096, MADV_GUARD_INSTALL) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe5147ff990, parent_tid=0x7fe5147ff990, exit_signal=0, stack=0x7fe513fff000,
stack_size=0x7fff80, tls=0x7fe5147ff6c0} => {parent_tid=[167782]}, 88) = 167782
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7fe5137fe000
madvise(0x7fe5137fe000, 4096, MADV_GUARD_INSTALL) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
```

```
child_tid=0x7fe513ffe990, parent_tid=0x7fe513ffe990, exit_signal=0, stack=0x7fe5137fe000,
stack_size=0x7fff80, tls=0x7fe513ffe6c0} => {parent_tid=[167784]}, 88) = 167784
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        mmap(NULL, 8392704, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7fe512ffd000
        madvise(0x7fe512ffd000, 4096, MADV_GUARD_INSTALL) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe5137fd990, parent_tid=0x7fe5137fd990, exit_signal=0, stack=0x7fe512ffd000,
stack_size=0x7fff80, tls=0x7fe5137fd6c0} => {parent_tid=[167785]}, 88) = 167785
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        mmap(NULL, 8392704, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7fe5127fc000
        madvise(0x7fe5127fc000, 4096, MADV_GUARD_INSTALL) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe512ffc990, parent_tid=0x7fe512ffc990, exit_signal=0, stack=0x7fe5127fc000,
stack_size=0x7fff80, tls=0x7fe512ffc6c0} => {parent_tid=[167786]}, 88) = 167786
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        write(1, "Progress erosion: 37/3000 rows p"..., 41) = 41
        clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=100000000}, NULL) = 0
        write(1, "\n", 1)                       = 1
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=236959439}) = 0
        write(1, "Execution time erosion: 0.2121 s"..., 39) = 39
        write(1, "Threads used: 4\n", 16)       = 16
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=237034551}) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe512ffc990, parent_tid=0x7fe512ffc990, exit_signal=0, stack=0x7fe5127fc000,
stack_size=0x7fff80, tls=0x7fe512ffc6c0} => {parent_tid=[167791]}, 88) = 167791
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe5137fd990, parent_tid=0x7fe5137fd990, exit_signal=0, stack=0x7fe512ffd000,
stack_size=0x7fff80, tls=0x7fe5137fd6c0} => {parent_tid=[167792]}, 88) = 167792
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe513ffe990, parent_tid=0x7fe513ffe990, exit_signal=0, stack=0x7fe5137fe000,
stack_size=0x7fff80, tls=0x7fe513ffe6c0} => {parent_tid=[167793]}, 88) = 167793
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
        clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x7fe5147ff990, parent_tid=0x7fe5147ff990, exit_signal=0, stack=0x7fe513fff000,
stack_size=0x7fff80, tls=0x7fe5147ff6c0} => {parent_tid=[167794]}, 88) = 167794
        rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
        write(1, "Progress dilation: 22/3000 rows "..., 42) = 42
        clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=100000000}, NULL) = 0
        write(1, "\n", 1)                       = 1
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=446198662}) = 0
        write(1, "Execution time dilation: 0.2092 "..., 40) = 40
        write(1, "Threads used: 4\n", 16)       = 17
```

```
write(1, "\n", 1)                    = 1
write(1, "=== Results ===\n", 16)     = 16
write(1, "Total execution time: 0.4213 sec"..., 37) = 37
write(1, "\n", 1)                    = 1
write(1, "Processing completed successfull"..., 35) = 35
exit_group(0)                       = ?
+++ exited with 0 +++
```

Анализ производительности

Исследование проводилось на матрицах размеров 1000×1000, 2000×2000 и 3000×3000 с применением морфологических фильтров эрозии и наращивания. Тестирование выполнялось для различного количества потоков (1, 2, 12, 8, 16) при фиксированном количестве итераций K=3.

Время выполнения:

- **1000×1000**: от 0.454с (1 поток) до 0.252с (2 потока)
- **2000×2000**: от 1.800с (1 поток) до 0.603с (8 потоков)
- **3000×3000**: от 4.040с (1 поток) до 1.034с (12 потоков)

Максимальное ускорение:

- 1000×1000: **1.80x** (2 потока)
- 2000×2000: **2.98x** (8 потоков)
- 3000×3000: **3.90x** (12 потоков)

Эффективность использования потоков:

- Наивысшая эффективность: **90%** (2 потока для всех размеров матриц)
- При 12 потоках (равно ядрам): 14-32% в зависимости от размера матрицы

Вывод

Многопоточность эффективна для обработки больших матриц, обеспечивая ускорение до 3.9x. Оптимальная конфигурация: количество потоков, равное числу логических ядер, для больших данных. Размер данных должен существенно превышать накладные расходы параллелизации для получения выгоды; дальнейшее увеличение потоков более числа ядер нецелесообразно из-за резкого падения эффективности