МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Курсовой проект
По курсу «Операционные системы»

Студент: Лернер Ф. Л.
Группа: М8О-208Б-23
Преподаватель: Живалев Е. А.
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2024

Тема: Аллокаторы памяти

Цель работы: Целью курсового проекта являлось приобретение практических навыков в:

- Целью данного проекта является исследование и сравнение двух различных аллокаторов памяти на языке C++ по ключевым характеристикам, таким как фактор использования, скорость выделения и освобождения блоков, а также простота использования. Проект направлен на понимание работы различных стратегий управления памятью и их влияние на производительность программ.

Вариант: 16. Необходимо сравнить два алгоритма аллокации: алгоритм Мак-Кьюзи-Кэрелса и блоки по 2 в степени n.

Подробное описание каждого из исследуемых алгоритмов:

1. McKusick Allocator
   Основан на сегментации памяти на классы блоков фиксированного размера (16, 32, 64 байт и т.д.). Подходит для сценариев с небольшими запросами памяти.

2. Pow2 Allocator.
   Делит память на блоки, размеры которых кратны степеням двойки. Подходит для сценариев с большими блоками, но может приводить к фрагментации.

Задачи проекта:

1. Разработка двух аллокаторов памяти: Реализовать два различных алгоритма аллокации памяти, например, "первый подходящий" (First-Fit) и "наилучший подходящий" (Best-Fit).

2. Создание интерфейса: Обеспечить наличие функций, аналогичных стандартным функциям malloc, free и (опционально) realloc.

3. Инициализация аллокаторов: Реализовать механизм инициализации аллокаторов с использованием стандартных средств выделения памяти в ядре.

4. Разработка стратегии тестирования: Определить и реализовать стратегию тестирования для оценки производительности аллокаторов по заданным характеристикам.

Описание решения:

решение разработано на языке C++ с использованием библиотеки pthread для работы с потоками. Основные компоненты системы:

Пример реализации:

```
#include "allocator_mckusick.hpp"

#include <cstring>

#include <sys/mman.h>

#include <cassert>

#include <cmath>



AllocatorMcKusick::AllocatorMcKusick(size_t total_size) {

    page_size = 4096; // Размер страницы

    memory_pool = mmap(nullptr, total_size, PROT_READ | PROT_WRITE, MAP_PRI-
VATE | MAP_ANONYMOUS, -1, 0);

    assert(memory_pool != MAP_FAILED && "Failed to allocate memory pool");



    block_classes = 10; // 10 классов блоков (например, 16, 32, 64 и т.д.)

    free_lists.resize(block_classes, nullptr);

}



AllocatorMcKusick::~AllocatorMcKusick() {

    munmap(memory_pool, page_size * block_classes);

}



void* AllocatorMcKusick::allocate(size_t size) {
```

```
    size_t class_idx = std::ceil(std::log2(size));

    assert(class_idx < block_classes && "Requested size exceeds maximum
class");



    if (!free_lists[class_idx]) {

        free_lists[class_idx] = static_cast<char*>(memory_pool) + class_idx *
page_size;

    }



    void* block = free_lists[class_idx];

    free_lists[class_idx] = static_cast<void*>(static_cast<char*>(block) + (1
<< class_idx));

    return block;

}



void AllocatorMcKusick::deallocate(void* ptr) {

    // Простая реализация: добавление блока обратно в список

    memset(ptr, 0, sizeof(ptr));
```

Данные из терминала:

```
$ ./benchmark_allocators

Running simplified tests...

=== test_usage_factor_simple ===

McKusick usage factor: 0.00122
```

```
Pow2 usage factor: 0.00122


=== test_alloc_dealloc_simple ===

McKusick: allocated & deallocated 10 blocks

Pow2: allocated & deallocated 10 blocks


=== test_exceed_memory ===

Error: Requested size (2097152 bytes) exceeds maximum block size (1024 bytes)

Error: Requested size (2097152 bytes) exceeds memory pool size

McKusick usage factor: 0

Pow2 usage factor: 0


=== test_invalid_deallocation ===

Error: Attempt to deallocate invalid pointer

McKusick: allocated & deallocated 10 blocks

Pow2: allocated & deallocated 10 blocks
```

Вывод: В результате выполнения проекта будет получено понимание работы различных аллокаторов памяти, их сильных и слабых сторон. Сравнительный анализ позволит сделать выводы о том, какой из алгоритмов более эффективен в зависимости от условий использования и требований к производительности. McKusick Allocator лучше для сценариев с частыми небольшими запросами памяти. Pow2 Allocator лучше для больших блоков, но приводит к большему количеству фрагментации.