浙江大學
ZHEJIANG UNIVERSITY

面向对象程序设计

allocator + memory pool

3220103422 刘韬

2024 年 6 月 27 日

1 问题描述

内存池是一种常见的内存分配方式，它可以减少内存分配的次数，从而提高程序的运行效率。在本次实验中，我们需要实现一个简单的内存池，然后将其与 `std::allocator`进行比较。

2 实现方案

2.1 a naive implementation

最简单的分配就是我们根据输入的需要来分配内存，然后返回一个指针，这个指针指向的内存块的大小就是我们需要的大小。这种实现方式的优点是简单，缺点是效率低下，因为每次分配都需要遍历一遍内存块，找到一个合适的内存块。

2.2 a better implementation

我们使用内存池来管理，在内存池中维护一个链表，链表的每个节点都是一个内存块，每个内存块的大小都是固定的。当 Allocator 需要分配内存时，我们会将链表指向的首个内存块取出，然后将其分配给 Allocator, 并让链表指向下一个空闲块。当 Allocator 需要释放内存时，我们会将其归还给内存池。

下面是我们的内存池的实现（具体的实现见源码）:

2.2.1 allocate

用于分配内存，其参数为所需内存的大小。对于给定大小，我们找到其在`free_list`中对应的桶，然后取出这个桶的第一个内存块，将其分配给 `Allocator`。如果这个桶为空，我们会调用`refill`函数，将这个桶重新填满。这里如果超过了内存池中的最大内存块大小，我们会直接调用`malloc`函数分配内存。

2.2.2 deallocate

用于释放内存，其参数为要释放的内存块的指针。我们会将这个内存块归还给内存池，然后将其插入到 `free_list` 的头部。如果超过了内存池中的最大内存块大小，我们会直接调用`free`函数释放内存。

3 测试结果

我们使用 pta 上的测试程序进行测试，分别对

- `std::allocator`

- `my_naive_allocator`
- `my_allocator`

进行测试，测试结果如下

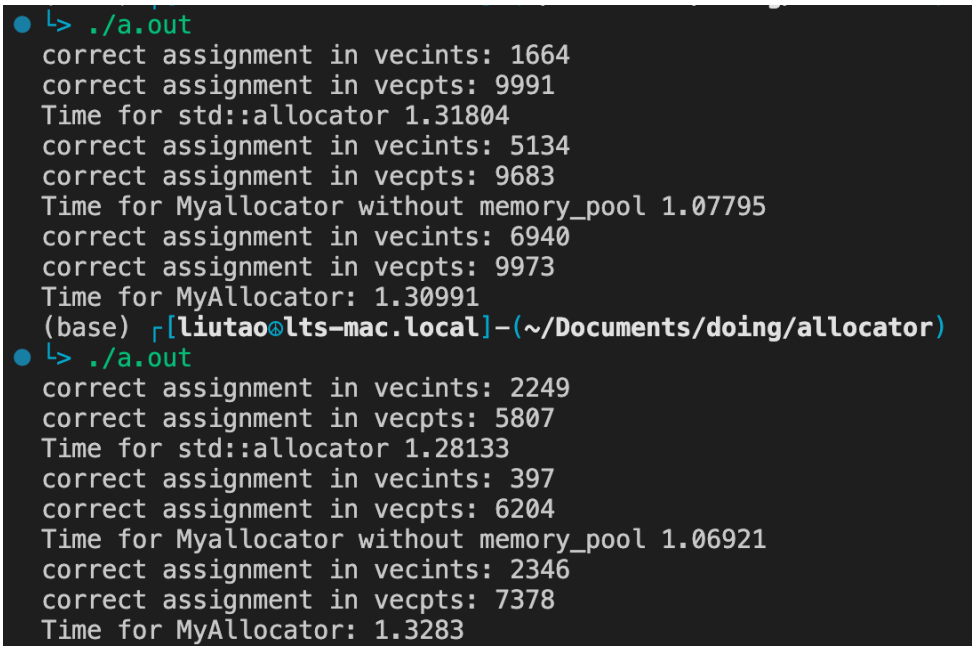

图 1: 测试结果