MPointers 2.0

Marin Navarro Jorge 2024174059
Morera Valverde Deiler 2023115868

I Semestre 2025
Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computadores
Curso: CE2103 - Algoritmos y Estructuras de Datos II

Índice

1. Introducción

MPointers 2.0 es una solución en C++ que encapsula el uso de punteros mediante una arquitectura cliente-servidor. El servidor, denominado *Memory Manager*, administra un bloque de memoria reservado en el *heap*, mientras que los clientes se comunican con él a través de una clase template llamada `MPointer<T>` utilizando gRPC.

Este enfoque permite al usuario manejar memoria de forma segura, aplicando conceptos de recolección de basura, fragmentación de memoria, y orientación a objetos.

2. Descripción del Problema

El manejo manual de memoria en C++ puede resultar en errores como fugas o corrupción de memoria. Este proyecto busca mitigar esos problemas implementando una librería que abstraiga el uso de punteros tradicionales, delegando la gestión de memoria a un servidor remoto.

3. Descripción de la Solución

3.1. Memory Manager

El *Memory Manager* es un servidor gRPC que:

- Se ejecuta mediante línea de comandos:

```
./mem-mgr --port 50051 --memsize 100 --dumpFolder /ruta/dumps
```

- Reserva un bloque único de memoria con `malloc`.
- Expone 5 tipos de peticiones:
  1. `Create(size, type)`
  2. `Set(id, value)`
  3. `Get(id)`
  4. `IncreaseRefCount(id)`
  5. `DecreaseRefCount(id)`
- Implementa un **Garbage Collector** en un hilo separado que revisa referencias periódicamente.
- Genera archivos `dump` con el estado de la memoria después de cada modificación.
- Incluye un sistema de **manejo de fragmentación** de memoria.

Problemas enfrentados

- Manejo correcto de offsets en la memoria lineal.
- Sincronización entre el garbage collector y las operaciones GRPC.
- Se enfrento problemas con las listas enlazadas.
- Se enfrentó también problemas con el manejo de versiones de Abseil, gRPC y Protobuff.

3.2. MPointer T

La clase template `MPointer<T>` simula un puntero tradicional pero con memoria remota. Sus características incluyen:

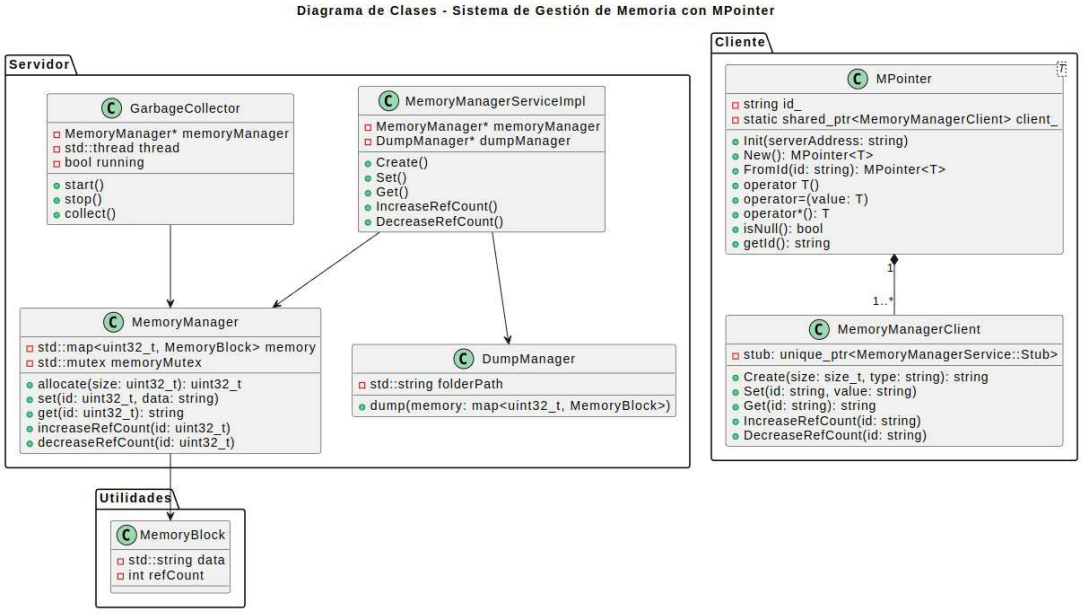
- Método `Init(port)`: establece conexión con Memory Manager.
- Método `New()`: crea un nuevo bloque remoto.
- Operador `*`: obtiene o almacena datos remotamente.
- Operador `=`: copia el ID de bloque y aumenta referencias.
- Operador `&`: retorna el ID.
- Destructor: reduce el conteo de referencias en Memory Manager.

Prueba adicional

No se pudo concretar las listas enlazadas sin embargo se dejo una pruebas que verifiquen la funcionalidad del MPointer y MemoryManager.

4. Diseño General

Diagrama de clases UML



Nota: Diagrama realizado en `https://www.planttext.com`.

5. Repositorio GitHub

- `https://github.com/zKleng/MPointer2.0/tree/MPointer-dev`

6. Conclusión

MPointers 2.0 es una solución efectiva para abstraer el manejo de memoria en C++, integrando conceptos avanzados como garbage collection y fragmentación, y aplicando principios de programación orientada a objetos. Permite experimentar con una arquitectura distribuida para control de memoria, lo cual es una experiencia enriquecedora para estudiantes de estructuras de datos.