Recolección de Basura y Gestión de Memoria

La gestión eficiente de memoria es crucial para el rendimiento del software. Este documento explora técnicas y mejores prácticas para optimizar el uso del heap y prevenir fugas de memoria.

H por Henry Mendoza

Made with Gamma

Conceptos de Recolección de Basura

1 **Definición**

Proceso automático de liberación de memoria no utilizada por el programa.

2 **Propósito**

Previene la saturación de recursos y mantiene la estabilidad del sistema.

3 **Importancia**

Fundamental para la gestión eficiente de memoria en lenguajes de alto nivel.

Made with Gamma

Mecanismos de Implementación de Garbage Collection

Marcación y Barrido

Marca objetos accesibles y elimina los no marcados. Eficiente en memoria, pero puede fragmentar el heap.

Recolección por Copia

Copia objetos vivos a una nueva región. Rápido, pero consume más memoria.

Recolección Generacional

Divide objetos por edad. Optimiza la recolección para objetos de corta vida.

Made with Gamma

Problemas de Fugas de Memoria

1

Definición

Ocurren cuando la memoria asignada no se libera correctamente después de su uso.

2

Causas

Referencias persistentes a objetos no utilizados o errores en la gestión manual de memoria.

3

Impacto

Degradación del rendimiento, aumento del consumo de recursos y posibles fallos del sistema.

Made with Gamma

Técnicas para Prevenir Fugas de Memoria

Punteros Inteligentes

En C++, gestionan automáticamente la liberación de memoria cuando ya no se necesita.

Gestión Manual

En lenguajes de bajo nivel, requiere un cuidadoso seguimiento de las asignaciones y liberaciones.

Analizadores Estáticos

Herramientas que detectan potenciales fugas de memoria durante el desarrollo del código.

Pruebas de Carga

Identifican fugas sometiendo la aplicación a condiciones de uso intensivo y prolongado.

Made with Gamma

Gestión de Memoria en el Heap

1

Asignación

El sistema reserva bloques de memoria según las necesidades del programa.

2

Uso

Los datos se almacenan y manipulan en la memoria asignada durante la ejecución.

3

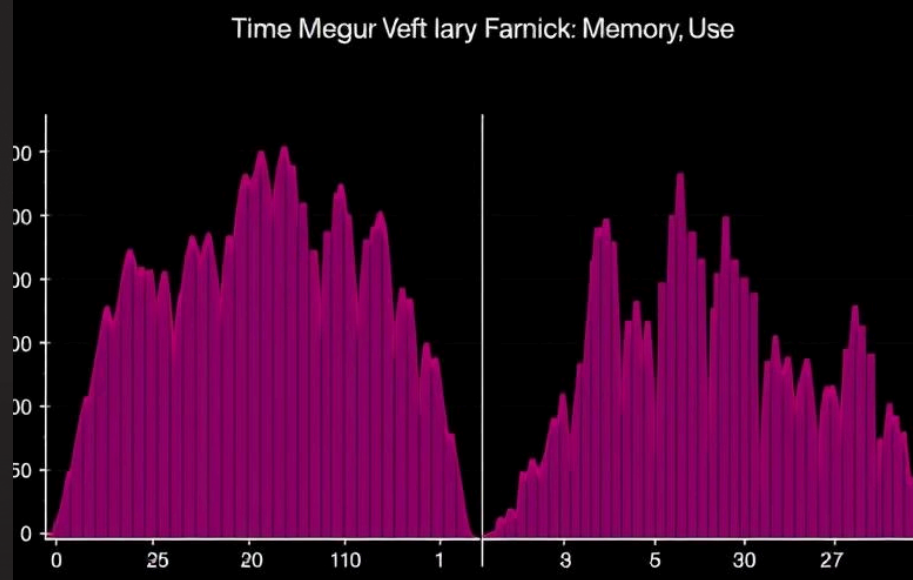
Liberación

La memoria se devuelve al sistema cuando ya no es necesaria.

Made with Gamma

Optimización del Uso del Heap con Garbage Collection

Algoritmo	Ventajas	Desventajas
Marcación y Barrido	Eficiente en memoria	Puede causar fragmentación
Recolección por Copia	Rápido y defragmenta	Mayor uso de memoria
Generacional	Optimizado para objetos efímeros	Complejo de implementar



Made with Gamma

Detección y Solución de Fugas de Memoria en RISC-V

Identificación

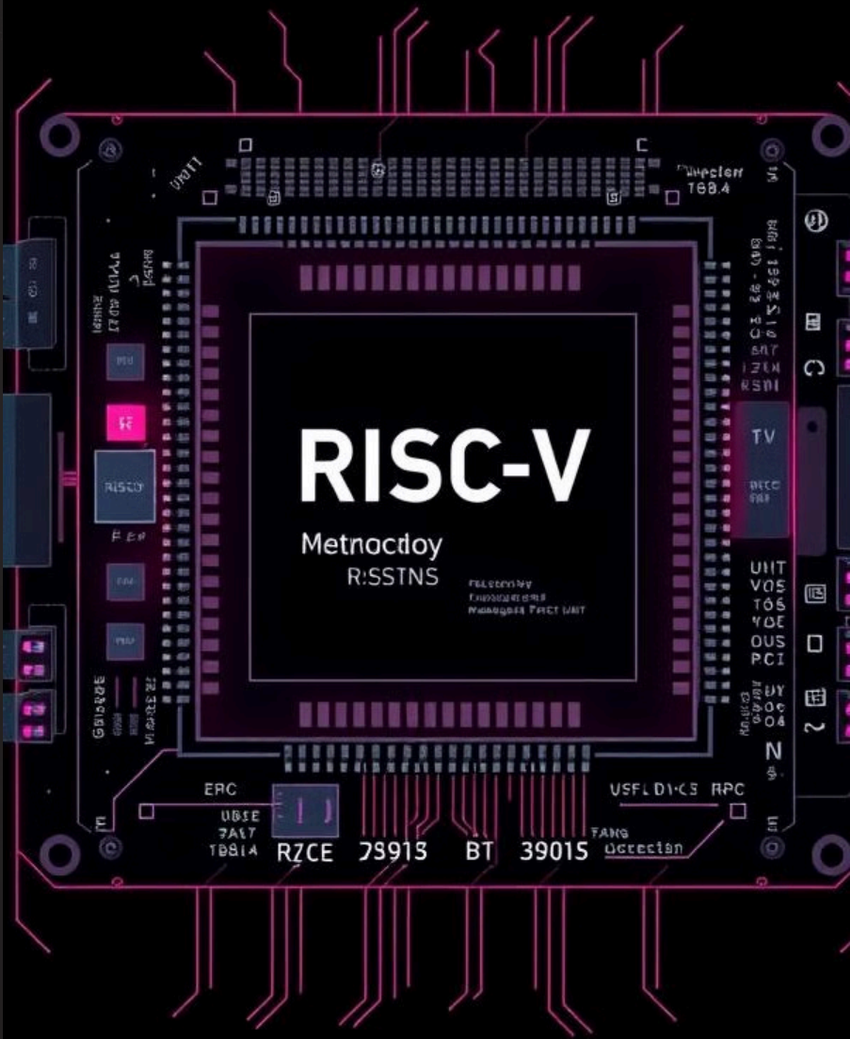
Uso de herramientas de perfilado de memoria específicas para arquitecturas RISC-V.

Depuración

Análisis del código ensamblador y seguimiento de punteros para localizar fugas.

Optimización

Implementación de técnicas de gestión de memoria eficientes para sistemas embebidos RISC-V.



Made with Gamma