UNDERSTANDING

Memory Management and GARBAGE Collection

Aditya Raj

TOPICS

1. Introduction - Memory Management
2. What is Garbage Collection(GC)
3. Pros and Cons of GC
4. Different techniques of GC
5. Memory model of Rust

MEMORY MANAGEMENT

- Memory management is the process of controlling computer's memory.
- It ensures that blocks of memory space are properly managed and is allocated for processing.
- De-allocating the acquired memory is also done under memory management.
- Memory management basically optimizes the memory usage and hence enhances the efficiency of a program.

GARBAGE COLLECTION

- Memory which is no longer in use, is called garbage memory.
- Garbage collection is a form of automatic memory management, which identifies garbage memory and frees it.
- Garbage collector attempts to reclaim garbage memory.
- Many programming languages were designed to have garbage collection as an integral part of it, example-Java, C#, etc.
- While some programming languages were designed for use with manual memory management, example- C, C++, etc. But there exists a library for garbage collection in C and C++ named- Boehm GC.
- Some languages allow both manual memory management and garbage collection to co-exist by maintaining separate heaps for manually managed and collected object. Example- Ada, Modula-3.

PROS

- Garbage collection relieves programmer from from doing manual memory management, where the programmer specifies what object to de-allocate and when to do so.
- Dangling pointers aren't left in the memory when GC is used. Dangling pointer is a pointer which points to a memory block which has already been freed but its pointer wasn't. So de-referencing such a pointer would lead to unexpected result.
- Garbage collection ensures that a same memory block isn't freed twice, which is a kind of error called double free bugs in manual memory management.

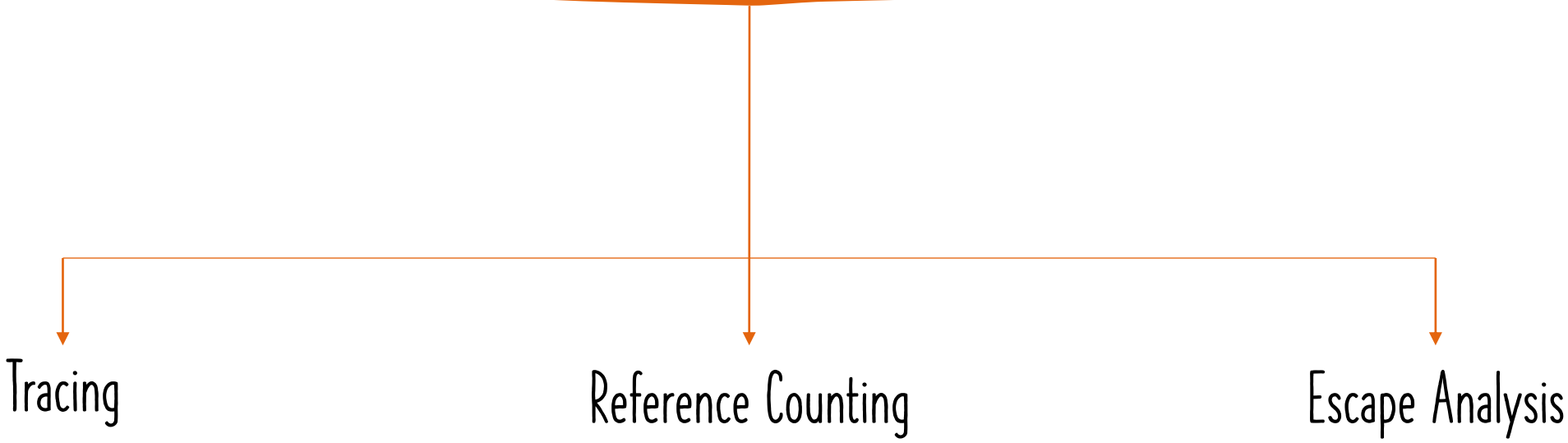
PROS

- Memory leaks are avoided. In manual de-allocation it may be a case that we free a pointer to a pointer but didn't free the pointer itself which will ultimately lead to memory exhaustion.
- Programs that manage memory manually may access memory incorrectly, which may ultimately lead to program crash but such things doesn't happen when garbage collection is used.
- Garbage collection saves programmer's time, as now they don't have to manually track and de-allocate memory.

CONS

- Not providing lifetime of variables manually creates an overhead, which can affect the program performance.
- GC uses computer resources to de-allocate memory and hence GC has its impact on performance of any program which uses GC.
- FACT - Apple has not adopted GC yet and the reason being the impact of GC on performance.
- Moment when garbage is collected can be unpredictable and hence may lead to stall.
- Such unpredictable stalls can be unacceptable in real-time environment, in transaction processing and many such scenarios.

TECHNIQUES OF GARBAGE COLLECITON



TRACING

- As the name suggests, this technique uses tracing to free long unused memory.

- This technique determines whether an object is garbage or not by tracing whether that object is reachable via a chain of references from certain root objects, if the object is reachable then the object isn't considered as garbage otherwise it is considered as garbge and the memory is freed.

REFERENCE COUNTING

- This techniques uses the count of references an object has to determine whether that object is garbage or not.
- In this method every object has a count of references to it and an object is considered garbage when its reference count reaches zero.
- An object's reference count is incremented when a reference is created to it.
- Similarly, an object's reference count is decremented when a reference is destroyed.
- When an object's reference count reaches zero, then that memory is reclaimed.

ESCAPE ANALYSIS

- Escape Analysis is a technique which can convert heap allocations to stack allocations and hence can reduce the amount of garbage to be collected.
- This method determines whether an object created within a function is accessible outside it or not.
- If a function's local allocation is accessible in some other function, then the allocation is said to escape and cannot be done on the stack.
- Otherwise the object may be directly allocated into the stack and released when the function returns, by passing the heap and hence enhancing the efficiency of the program.

```python
import gc
import time

# store starting time
begin = time.time()

# program body starts
a=5
b=a    #creating a reference of 'a'
print(a)
print(b)
b=None  #destroying that reference of 'a'
# program body ends

time.sleep(1)
# end time before calling gc by force
end1 = time.time()
gc.collect()
# end time after calling gc by force
end2 = time.time()

# total time taken before and after using gc
print(f"Total runtime of the program is {end1 - begin}")
print(f"Total runtime of the program is {end2 - begin}")
```

Result

```
5
5
Total runtime of the program is 1.050771713256836
Total runtime of the program is 1.058783531188964
```

Time taken without garbage collection

Time taken with garbage collection

MEMORY MODEL IN RUST

- Rust is a general purpose programming language which ensures that all references point to valid memory without using any of the above mentioned garbage collection technique.
- Ownership is a convention of Rust using which it ensures memory safety and that too without using Garbage Collection.
- The Rust compiler checks if a program obeys ownership rules or not at compile time. If the rules are followed then it can run otherwise the compiler will refuse to produce an executable.
- Rust verifies the rules using Borrow checker. Borrow checker verifies the ownership rules and checks whether the value is in scope or not. If a value is out of its scope then it is not available to other parts of the programs unless it is borrowed.

OWNERSHIP MODEL

- The first ownership rule is that every variable owns the value it is initialised to.
- The second ownership rule is that no two variables can point to the same memory address or we can say every value can have only one owner.
- The third ownership rule is that once a variable is out of the scope of the declared variable then the value is dropped and the memory is de-allocated.
- If we pass a value as an argument to a function then the function can access that variable although that variable wasn't declared within the function's scope. The function can access the variable in Rust because Rust moves the ownership of the value to the function at the compile time.

SUMMARY

Garbage collection is a very crucial part of any programming language(except some) as it reduces the burden of tracking each and every variable created and freeing the memory and it also offers many more benefits but then also some programming language and tech giant like Apple avoid using Garbage Collection due to its unpredictability and its negative effect on efficiency of the program.

So if any how we can develop/modify a garbage collection algorithm which takes considerably less time, space and will be more predictable than the present methods then I think Garbage Collection will have a positive impact on performance of a program and minimal negative effects on any program.


thank you