Licenciatura en Ingeniería de Software
Taller de Programación Visual III

UNIDAD I C# Avanzado	Tarea # 8
Nombre: LUIS ENRIQUE SANTOS LANDEROS	Fecha: 27/01/2019

Introducción

El garbage collector en Java, es una forma de recoger la basura dentro del sistema de Java, es en parte concurrente, paralela y para en el mundo de la programación si no es revisada correctamente, este pues se caracteriza básicamente en tener piezas, que en este caso son las partes de la memoria usada durante el transcurso de la aplicación, estos son mediante hilos, que van recolectando esos trozos, que van siendo borrados en caso de no ser usados en ningún momento de la operación antes de que se cierre el programa, algo inusual dentro de este tema es son las formas precisas vs las conservativas que desde un punto de vista técnico, la forma en que se generalizan divagan sobre la verdadera forma en cómo trabaja el recolector de basura, en la cual son mediante punteros, para no perder de vista el uso que se le está dando al uso de esa dirección en concreto, pero como ya lo dijimos puede ser concurrente o paralela, es probable que el puntero no sea realmente la clave de acceso a los trozos de memoria en concreto pero si de su uso en el sistema, Java como tal tiene un compilador JIT que siempre mejora el rendimiento de la máquina, y de la aplicación en concreto para no llenar espacios vacíos ni eliminarlos en caso de que se usen, no necesariamente deben ser usados continuamente, pero este no lo hace de forma nativa como lo podría hacer una aplicación nativa, si no que usa un código de bytes en código de maquina nativo, esto generalmente se hace desde puntos, un punto A, a un punto B, y se van moviendo los objetos en la memoria, de un punto a otro.

Desarrollo

El recolector de basura de Java, es una forma de estar moviendo los objetos usados en la aplicación, de un lado a otro en la memoria del sistema, esto permite el flujo de datos de una forma adecuada, optimizando el sistema de cualquier máquina, ya mencionado el compilador JIT que el cual trabaja de una manera eficiente con sus códigos de byte a códigos de maquina nativa, hace que estos objetos puedan moverse libremente sin obstruir, espacios en la memoria que deben estar libres para su buen funcionamiento dentro de la aplicación, dentro de esta forma de recolectar los datos de la memoria, hay un punto seguro donde se traen todos los hilos a un punto de seguridad común y que es lo que se conoce como parar el mundo (stop-the-world), esto es pausar el punto de seguridad global, para traer esos objetos muertos a una punto del cual serán remplazados o eliminados de la memoria del sistema.

Hay un recolector llamado coleccionista, que hace esto en tres pasos, paso uno encuentras toda la basura muerta, paso dos, reclamo de los objetos muertos, y paso tres, reacomodar los objetos vivos, básicamente es como la garbage collector de Java va reacomodando esos espacios muertos por decirlo así, haciendo un nuevo uso de ellos en otras direcciones de memoria que se podrían usar para el funcionamiento de los datos manejados en la aplicación, esto sucede como un queso suizo que al partirlo por la mitad encuentras espacios vacíos sin ningún uso, y quizás es el 90% de vacío en la memoria del sistema, y ya no hay lugar para poner más datos porque esos espacios ya no se reconocen por el sistema dado que siguen ahí, realmente esto no libera ningún tipo de memoria sino todo lo contrario, va acaparando mucho espacio, sin hacer uso de los espacios vacíos, por eso al copiar estos objetos, se crean otros objetos que suplen a los anteriores para reutilizar a los antiguos, no necesariamente, pero depende el caso de las condiciones en que se muevan los datos en la memoria del sistema, por eso al copiarlos no estaría generando memoria en vano sino eliminando al antigua y usando la nueva en otro objeto que es común de uso, esto compacta sin barrer la mayoría de los espacios vacíos, esto lo hace de forma lineal, para poder acceder de forma rápida en los índices de la memoria, después el Mark-Sweep compacta esta línea de memoria, es por eso que al generar objetos jóvenes, estos e reutilizan de forma consecutiva, como vestigios del pasado, haciendo uso de ellos para ganar dinero de los turistas en el lugar, esto de alguna manera hace el papel de la garbage collector que solo va cambiando de puntos estratégicos en los lugares que se ocupe esa memoria al momento que sea necesario, esto no podría suceder sin el colector de barrido de marcas simultáneo(Concurrent Mark Sweep , CMS), básicamente está diseñado para aplicaciones que prefieren pausas de recolección de basura más cortas y que pueden permitirse compartir recursos del procesador con el recolector de basura mientras la aplicación se está ejecutando, el recopilador de CMS es generacional; así se producen tanto colecciones menores como mayores.

La garbage collection de Gil Tene, muestra como es indispensable el uso de estos espacios de memoria en nuestra computadora para saberlos utilizar en otros que necesitemos, JVM (Java Virtual Machine) ocasionalmente experimentan grandes pausas de aplicaciones STW (Stop-The-World) debido a que el registro de GC de JVM está bloqueado por el tráfico de E / S de fondo, por ejemplo, la copia de caché de la página del sistema operativo, cuando el espacio de almacenamiento en Java que gestiona la JVM se recolecta como basura, la JVM podría detenerse, lo que introduce pausas STW en las aplicaciones y aunque algunas pausas de STW inducidas por GC que son bien conocidas para el escaneo / marca / compactación de objetos, descubrimos que hay algunas pausas de STW grandes causadas por el tráfico de E / S de fondo. Prácticamente juega un papel muy importante al momento de usar espacios vacíos, pero mejor aún recolectar esa basura y compactarla para hacer un buen uso de los espacios que quizás en algún futuro de nuestra memoria pueda ser usado sin ningún problema.

Conclusión

En conclusión las garbage collector de Java, tiene una manera de recolectar la basura del sistema mediante colecciones que serán usadas en futuros procesos, y esto se debe a su eficiencia al momento de manejar los objetos de un punto a otro, copiando y compactando los datos usados en la memoria del sistema, básicamente es una reutilización de esos huecos que quedan desapercibidos por el sistema al momento de ejecutar un programa en Java, y eso mejora el rendimiento de la misma por lo que el compilador JIT y el JVM juegan un papel muy importante dentro de Java, proporcionándole la velocidad de transferencia de datos y lo que comúnmente despreciamos, la basura o chache del sistema.