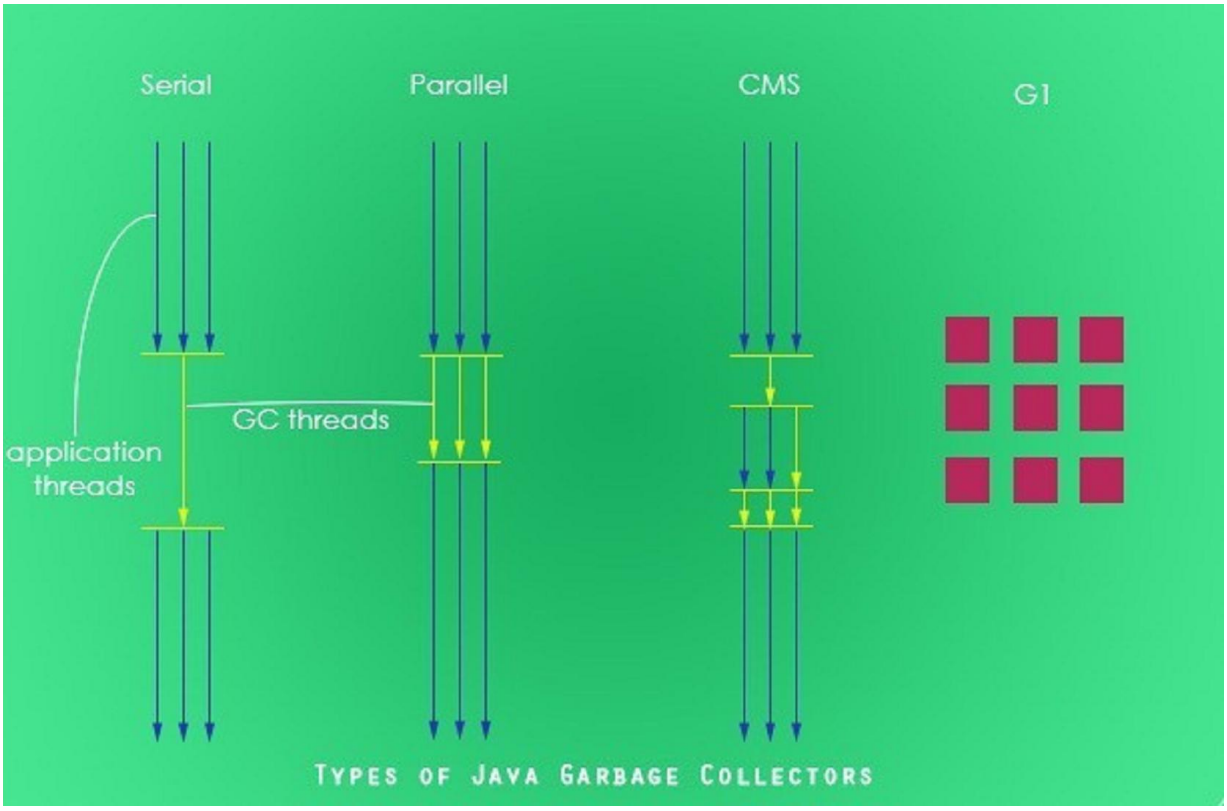
Types of Java Garbage Collectors

In this tutorial we will go through the various type of Java garbage collectors available. Garbage collection is an automatic process in Java which relieves the programmer of object memory allocation and de-allocation chores. This is the third part in the garbage collection tutorial series. In the previous part 2 we saw about how garbage collection works in Java, it is an interesting read and I recommend you to go through it. In the part 1 introduction to Java garbage collection, we saw about the JVM architecture, heap memory model and surrounding Java terminologies.

Java has **four types of garbage collectors**,

1. Serial Garbage Collector
2. Parallel Garbage Collector
3. CMS Garbage Collector
4. G1 Garbage Collector



Each of these four types has its own advantages and disadvantages. Most importantly, we the programmers can choose the type of garbage collector to be used by the JVM. We can choose

them by passing the choice as JVM argument. Each of these types differ largely and can provide completely different application performance. It is critical to understand each of these types of garbage collectors and use it rightly based on the application.

1. Serial Garbage Collector

Serial garbage collector works by holding all the application threads. It is designed for the single-threaded environments. It uses just a single thread for garbage collection. The way it works by freezing all the application threads while doing garbage collection may not be suitable for a server environment. It is best suited for simple command-line programs.

Turn on the `-XX:+UseSerialGC` JVM argument to use the serial garbage collector.

2. Parallel Garbage Collector

Parallel garbage collector is also called as throughput collector. It is the default garbage collector of the JVM. Unlike serial garbage collector, this uses multiple threads for garbage collection. Similar to serial garbage collector this also freezes all the application threads while performing garbage collection.

3. CMS Garbage Collector

Concurrent Mark Sweep (CMS) garbage collector uses multiple threads to scan the heap memory to mark instances for eviction and then sweep the marked instances. CMS garbage collector holds all the application threads in the following two scenarios only,

1. while marking the referenced objects in the tenured generation space.
2. if there is a change in heap memory in parallel while doing the garbage collection.

In comparison with parallel garbage collector, CMS collector uses more CPU to ensure better application throughput. If we can allocate more CPU for better performance then CMS garbage collector is the preferred choice over the parallel collector.

Turn on the `XX:+USeParNewGC` JVM argument to use the CMS garbage collector.

4. G1 Garbage Collector

G1 garbage collector is used for large heap memory areas. It separates the heap memory into regions and does collection within them in parallel. G1 also does compacts the free heap space on the go just after reclaiming the memory. But CMS garbage collector compacts the memory on stop the world (STW) situations. G1 collector prioritizes the region based on most garbage first.

Turn on the `-XX:+UseG1GC` JVM argument to use the G1 garbage collector.

Java 8 Improvement

Turn on the `-XX:+UseStringDeduplication` JVM argument while using G1 garbage collector. This optimizes the heap memory by removing duplicate String values to a single char[] array. This option is introduced in Java 8 u 20.

Given all the above four types of Java garbage collectors, which one to use depends on the application scenario, hardware available and the throughput requirements.

Garbage Collection JVM Options

Following are the key JVM options that are related to Java garbage collection.

Type of Garbage Collector to run

Option	Description
-XX:+UseSerialGC	Serial Garbage Collector
-XX:+UseParallelGC	Parallel Garbage Collector
-XX:+UseConcMarkSweepGC	CMS Garbage Collector

-XX:ParallelCMSThreads=	CMS Collector – number of threads to use
-XX:+UseG1GC	G1 Gargbage Collector

GC Optimization Options

Option	Description
-Xms	Initial heap memory size
-Xmx	Maximum heap memory size
-Xmn	Size of Young Generation
-XX:PermSize	Initial Permanent Generation size
-XX:MaxPermSize	Maximum Permanent Generation size

Example Usage of JVM GC Options

```
java -Xmx12m -Xms3m -Xmn1m -XX:PermSize=20m -XX:MaxPermSize=20m -XX:+UseSerialG
```

In the next part of this Java garbage collection tutorial series, we will see about how to monitor and analyze the garbage collection with an example Java application.

This Java tutorial was added on 12/10/2014.

« How Java Garbage Collection Works?

Android Get Address with Street Name, City for Location with Geocoding »

Comments on "Types of Java Garbage Collectors" Tutorial:

Java Garbage Collection Introduction says: *12/10/2014 at 11:20 pm*

[...] Types of Java Garbage Collectors [...]

Reply

How Java Garbage Collection Works? says: *12/10/2014 at 11:21 pm*

[...] is the third part of the garbage collection tutorial series and we will see about the different types of Java garbage collectors [...]

Reply

Dharmesh says: *13/10/2014 at 10:56 am*

Good One

Reply

Tom Henricksen says: *13/10/2014 at 6:02 pm*

This is a good summary of Java garbage collection, every Java developer should have a basic understanding of this. It comes up time and again and the GC Optimizations are important to know when setting up an environment.

Reply

Niraj Kumar says: *14/10/2014 at 4:59 pm*

Hi Joe,

I was looking for detailed information on Java GC starting from basic concepts to how many types of GC are there and how they work.

Can you please also add some details on Permanent Generation or PermGen. Usually, this is the hot topic in interviews :-).

Reply

Joe says: *14/10/2014 at 7:21 pm*

PermGen is the memory space where the program meta data is stored. Information like the classes and methods.

Do not worry about it much now. From Java SE 8, PermGen space is removed.

Reply

Your Comment

Name

Email

Post Comment

Java

↑ Go to top

Java Puzzles

Third Party

Tools

Java Interview Questions

Others

Java Gallery

Android

Design Patterns

Hibernate

Spring

Web Services

Servlet

JavaPapers is a tutorials site and **Joe** *runs it passionately.*

Say hi: joe@javapapers.com

Recommended Tutorials

- Java History
- Overloading Vs Overriding In Java
- Eclipse Shortcuts
- Java Serialization
- Difference Between Interface And Abstract Class
- Java Hashtable
- Difference Between Forward And SendRedirect
- Differentiate JVM JRE JDK JIT
- Java (JVM) Memory Types
- Why Multiple Inheritance Is Not Supported In Java