In Java, garbage collection (GC) algorithms manage memory by automatically reclaiming objects that are no longer in use.

1. Serial Garbage Collector

The Serial GC is the simplest and is primarily designed for single-threaded environments and small heaps. It performs all GC operations on a single thread, which can lead to long pause times as it stops all application threads during the collection process. This collector is generally not suitable for server environments but can be effective for small applications or systems with limited resources.

Use case: Single-threaded applications with small heaps. **JVM Option:** `-XX:+UseSerialGC`

2. Parallel Garbage Collector

The Parallel GC, also known as the throughput collector, uses multiple threads to perform GC tasks, making it suitable for applications that can tolerate longer pauses but need to maximize throughput. It is the default collector in many JVM configurations and is ideal for batch processing or data-intensive applications.

Use case: Applications where throughput is more important than low latency. **JVM Option:** `-XX:+UseParallelGC`

3. Concurrent Mark-Sweep (CMS) Garbage Collector

The CMS collector aims to minimize pauses by performing most of its work concurrently with the application threads. It uses multiple threads to mark and sweep unused objects. However, it can still cause significant pause times during initial and remark phases and is more CPU-intensive. The CMS collector is suitable for applications that require shorter pause times and can afford higher CPU usage.

Use case: Long-running server applications that require low pause times. **JVM Option:** `-XX:+UseConcMarkSweepGC`

4. Garbage-First (G1) Garbage Collector

The G1 GC is designed for large heaps and multi-processor machines. It divides the heap into regions and focuses on collecting regions with the most garbage first, hence the name "Garbage-First." G1 aims to provide predictable pause times while maintaining good throughput. It is the default collector for server-class machines starting from JDK 9.

Use case: Large applications requiring balanced performance between throughput and pause times. **JVM Option:** `-XX:+UseG1GC`

5. Z Garbage Collector (ZGC)

ZGC is a low-latency collector designed to handle very large heaps (up to multi-terabytes) with minimal pause times, typically not exceeding 10 milliseconds. It performs most of its work concurrently with application threads, making it suitable for latency-sensitive applications. ZGC is available as an experimental feature in JVM 11 and later versions.

Use case: Applications requiring very low latency and large heap sizes. **JVM Option:** `-XX:+UseZGC`

Summary

- **Serial GC** is suitable for small applications or single-threaded environments.
- **Parallel GC** is ideal for maximizing throughput in batch processing scenarios.
- **CMS GC** provides low pause times at the cost of higher CPU usage.
- **G1 GC** offers a balance between throughput and pause times for large heaps.
- **ZGC** is optimized for very large heaps and minimal pause times.

Each of these collectors has its strengths and is suitable for different application requirements. Understanding the specific needs of your application will help you choose the most appropriate garbage collection algorithm.