1.Serial garbage collection

This algorithm uses mark-copy for the Young Generation and mark-sweep-compact for the Old Generation. It works on a single thread. When executing, it freezes all other threads until garbage collection operations have concluded.

Due to the thread-freezing nature of serial garbage collection, it is only feasible for very small programs

2. Parallel garbage collection

Similar to serial GC, It uses mark-copy in the Young Generation and mark-sweep-compact in the Old Generation. Multiple concurrent threads are used for marking and copying / compacting phases.

Parallel Garbage Collector is suitable on multi-core machines in cases where your primary goal is to increase throughput by efficient usage of existing system resources. Using this approach, GC cycle times can be considerably reduced.

3. Concurrent mark sweep (CMS) garbage collection

CMS garbage collection is essentially an upgraded mark and sweep method. It scans heap memory using multiple threads. It was modified to take advantage of faster systems and had performance enhancements.

It attempts to minimize the pauses due to garbage collection by doing most of the garbage collection work concurrently with the application threads. It uses the parallel stop-the-world mark-copy algorithm in the Young Generation and the mostly concurrent mark-sweep algorithm in the Old Generation.

4. G1 garbage collection

The G1 (Garbage First) garbage collector was available in Java 7 and is designed to be the long term replacement for the CMS collector. The G1 collector is a parallel, concurrent, and incrementally compacting low-pause garbage collector.

This approach involves segmenting the memory heap into multiple small regions (typically 2048). Each region is marked as either young generation (further devided into eden regions or survivor regions) or old generation. This allows the GC to avoid collecting the entire heap at once, and instead approach the problem incrementally. It means that only a subset of the regions is considered at a time.

5. . The Z Garbage Collector

The Z Garbage Collector, also known as ZGC, is a scalable, low-latency garbage collector. It was first introduced in Java 11 as an experimental feature and became production-ready in Java 15.

The purpose of this feature was to minimize or eliminate long garbage collection pauses, thereby enhancing application responsiveness and accommodating the growing memory capacities of modern systems.

Comparison Summary

Garbage Collector	**Primary Goal**	**Latency**	**Throughput**	**Heap Size**	**Pause Time (STW)**	**Concurrency**
Serial GC	Simplicity	High	Low	Small	Long	No
Parallel GC	Throughput	Moderate	High	Small to Medium	Moderate	No
CMS GC	Low Latency	Low	Moderate	Medium to Large	Short	Yes
G1 GC	Predictability	Moderate	High	Medium to Large	Short	Yes
ZGC	Very Low Latency	Very Low	Moderate	Large to Very Large	Very Short	Yes