Task 4: Research and present a comparison of different garbage collection algorithms (Serial, Parallel, CMS, G1, ZGC) in Java.

Garbage Collection (GC) algorithms in Java are essential for managing memory effectively by reclaiming memory occupied by objects that are no longer in use. Different GC algorithms have been developed over the years, each with specific characteristics and optimizations suited for different application scenarios. Here's a comparison of several prominent GC algorithms used in Java:

1. Serial Garbage Collector

- **Type**: Single-threaded, stop-the-world (Young Generation) and Old Generation)
- **Suitability**: Suitable for small applications or client-side applications where low pause times are not critical.
- **Behavior**:
  - Uses a single thread for garbage collection.
  - Pauses application threads during both Young and Old Generation collections.
  - Efficient for applications with small to medium-sized data sets.

2. Parallel Garbage Collector

- **Type**: Multi-threaded, stop-the-world (Young Generation) and Old Generation)
- **Suitability**: Generally used on server-class machines with multiple CPUs, intended for throughput performance.
- **Behavior**:
  - Utilizes multiple threads to perform garbage collection.
  - Pauses application threads during both Young and Old Generation collections.
  - Optimized for applications that prioritize throughput and can tolerate longer pause times.

3. Concurrent Mark-Sweep (CMS) Garbage Collector

- **Type**: Multi-threaded (mostly concurrent for Old Generation)
- **Suitability**: Suitable for applications requiring low pause times and responsive applications.
- **Behavior**:
  - Concurrently marks live objects in the Old Generation while the application continues running.
  - Pauses application threads briefly during initial marking and remark phases.
  - Not suitable for applications with large heaps or on machines with a small number of CPUs due to potential CPU overhead.

4. Garbage-First (G1) Garbage Collector

- **Type**: Multi-threaded, region-based (Young Generation and Old Generation)
- **Suitability**: Designed for large heaps (multi-GB) with low pause time requirements.
- **Behavior**:
  - Divides the heap into regions and performs garbage collection on smaller regions (mostly Young Generation) to reduce pause times.
  - G1 dynamically adjusts the size of regions based on application behavior.
  - Can be used for both client-side and server-side applications, suitable for mixed workloads.

5. Z Garbage Collector (ZGC)

- **Type**: Multi-threaded, concurrent, region-based (Whole heap)
- **Suitability**: Designed for applications requiring very low pause times (less than 10ms), especially in large heaps (multi-terabyte).
- **Behavior**:
  - Uses colored pointers and load barriers to ensure that all references are consistently updated.
  - Concurrently compacts memory and performs GC cycles alongside the application threads with minimal pause times.
  - Suitable for applications like high-frequency trading, large-scale analytics, or any application where low latency is critical.
  - **Parallel**: Higher throughput compared to Serial, suitable for batch processing.
  - **CMS**: Lower throughput compared to Parallel and G1 due to concurrent nature.
  - **G1**: Balanced throughput with low pause times, suitable for mixed workloads.
  - **ZGC**: High throughput with very low pause times, suitable for large-scale applications.