1. **Consolidate the JDK Forest into a Single Repository**

Consolidating the JDK Forest into a single repository is a process that involves combining multiple repositories or components of the JDK (Java Development Kit) into a single repository. This consolidation can simplify management, development, and version control of the JDK

1. **Garbage Collector Interface**

Improve the source code isolation of different garbage collectors by introducing a clean garbage collector (GC) interface.

1. **Parallel Full GC for G1**

In Java 10, the G1 (Garbage-First) garbage collector introduced enhancements that included parallel Full GC. Prior to Java 10, Full GC in G1 was single-threaded, which could lead to long pause times, especially for large heaps. The introduction of parallelism for Full GC aimed to reduce these pause times by allowing multiple threads to perform the garbage collection work concurrently.

1. Parallelism in Full GC: With the introduction of JEP 307, G1's Full GC became multi-threaded. This means that multiple threads can be used during Full GC to speed up the process.

2. Configuration and Tuning:

You can configure the number of threads used for parallel Full GC using the -XX:ParallelGCThreads option. By default, the JVM chooses a suitable number based on the available processors.

For example, to set the number of threads to 4, you can use:

diff

Copy code

-XX:ParallelGCThreads=4

3. G1 GC Tuning Options:

-XX:+UseG1GC: Enables the G1 garbage collector.

-XX:MaxGCPauseMillis: A target maximum pause time goal for the G1 GC. This is a soft goal and the collector will try to achieve it with high probability.

-XX:InitiatingHeapOccupancyPercent: Percentage of the (entire) heap occupancy to start a concurrent GC cycle. Default is 45%.

-XX:G1HeapRegionSize: Sets the size of the G1 regions. The value can be between 1 MB and 32 MB and should be a power of 2.

4. Monitoring and Diagnostics:

-Xlog:gc\*: Enables detailed logging for garbage collection, which can help in diagnosing and understanding GC behavior.

For example, to enable GC logging with time stamps:

ruby

Copy code

-Xlog:gc\*:file=gc.log:time

5. Benefits of Parallel Full GC:

Reduced Full GC pause times due to parallel processing.

Improved overall application responsiveness and throughput, especially for applications with large heaps or heavy allocation rates.

1. **Application Class-Data Sharing**

Application Class-Data Sharing (AppCDS) in Java 10 extends the existing Class-Data Sharing (CDS) feature, which was originally introduced to improve startup time and reduce memory footprint by sharing common class metadata across multiple Java Virtual Machine (JVM) instances.

1. **Heap Allocation on Alternative Memory Devices**

Heap Allocation on Alternative Memory Devices is a feature in Java that allows the Java Virtual Machine (JVM) to allocate the Java heap on non-volatile memory (NVM) devices or other alternative memory hardware. This can be particularly useful in environments where such devices are available and can offer benefits like faster access times compared to traditional disk-based storage.