7 Types of Garbage Collectors in Java

- An overview

Agenda

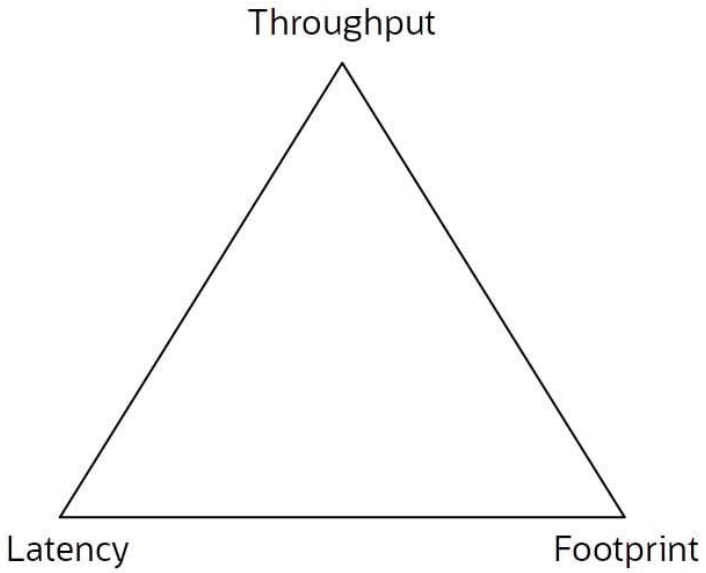
- About Garbage Collection
- Three metrics of GC
- GC Algorithms
  - Serial GC
  - Parallel GC
  - Concurrent Mark Sweep GC
  - Z GC
  - Garbage First (G1) GC
  - Shenandoah GC
  - Epilson GC

About Garbage Collection

- Garbage collection is process of JVM where unused objects are removed from the heap memory.
- Various Java versions provide different default and optional implementations of GCs
- Choice of a GC algorithm will depend on the Java version and nature of Java application

Three Metrics of a GC algorithm

- Every garbage collection algorithm occupies a part of this triangle based on where it is targeted and what it is best at.



Three Metrics of a GC algorithm

- **Latency**:- Whether the GC operation induces pauses and how long the GC pauses would be
- **Throughput**:- The amount of garbage collection work the algo. performs per unit time
- **Footprint**:- How much memory is needed by the collector to manage GC

Serial Garbage Collector

- Uses single thread to perform GC
- Both Major and Minor GCs are done serially
- Low memory footprint

Use Cases

- Suitable for client machines and embedded systems
- Suitable for multiprocessor machines when dataset is small (less than 100MB)
- Not Suitable for Large datasets

Parallel Garbage Collector

- Default for JDK 8 and earlier
- Similar to Single Thread GC, except that it uses multiple threads to speed-up the process
- High throughput

Use cases

- Intended for applications that use medium to large size datasets and are run on multiprocessor or multithreaded hardware

Concurrent Mark Sweep GC

- Deprecated in Java 9 and removed in Java 14
- Minimizes GC pause time
- Requires the application to share resources with GC during concurrent phases
- Higher throughput, higher latency

Use cases

- Should be used for applications that require low pause times and can share resources with GC
- Example - A Desktop UI application that respond to events or a webserver responding to a request

Z Garbage Collector

- Introduced in Java 15 as stable, prod-ready version
- Scalable , low-latency GC
- Performs all expensive work concurrently without stopping the work of application
- Intended for applications which require low latency

Use cases

- Applications that need large memory (big data)
- Applications that need reduced GC pause time (low latency systems)

Garbage First or G1 Garbage Collector

- Default collector since Java 9
- Uses multiple threads to perform GC
- Achieves balance between throughput and latency
- Very efficient with gigantic datasets

Use cases

- Trading platforms
- Interactive graphics programs

Shenandoah Garbage Collector

- Introduced in Java 12
- Low pause time garbage collector
- Focus on latency at cost of throughput
- GC Pause times are not proportional to heap size – 200 GB or 2 GB heap should have similar pause times
- Executes in concurrent threads to the application threads

Use cases

- Applications needing low GC pause times

Epilson Garbage Collector

- Introduced in Java 11, still an experimental feature
- Only manages memory allocation but does not clean objects – that is, it does no garbage collection. When all the heap memory of application is used up, the application will exit.

Use cases

- Short term running applications where allocated memory size is largely sufficient
- If you've ever wondered how much of your program's performance is affected by garbage collection, Epsilon is your solution to test the performance
- Other uses are discouraged

Further reading

- https://blogs.oracle.com/javamagazine/post/java-garbage-collectors-evolution
- https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html#overview
- https://blog.devgenius.io/from-java-9-to-java-15-evolutions-and-new-features-part-2-7405530ab748

Follow Me for more such content

- Follow Amol Limaye to more see such content in your feed

https://www.linkedin.com/in/amolrlimaye/