JAVA

Garbage Collector

Sunit

What is it?

Automatic memory management system

Collects unused objects

Improves performance

Prevents memory leaks

Different types available

Sunit

Types

Serial: Simple, single-threaded collector

Parallel: Multi-threaded for improved throughput

Concurrent Mark Sweep (CMS): Low-latency, concurrent collector

Garbage-First (G1): Optimized for large heaps

Z Garbage Collector (ZGC): Scalable, low-latency collector

Shenandoah: Low-pause time collector

Epsilon: No-op garbage collector

Mostly Concurrent (MCS): Concurrent collector with stop-the-world fallback

Balanced: Hybrid collector for throughput and pause time

Mark-and-Compact: Marks live objects and compacts them in memory

Generational: Separates objects by age and collects them differently

How it happens?

Tracing References
Identifies live objects by tracing references

Marking Phase
Marks live objects as reachable

Sweeping Phase
Frees memory for unreachable objects

Compacting Phase
Reorganizes memory to reduce fragmentation

Parallel Processing
Uses multiple threads for faster collection

Sunit

Young Generation

Area of memory where newly created objects are allocated

Separate Memory Space

Allocates objects in a separate space

Minor Collection

Collects young objects with minor collection

Eden Space

Frees memory for unreachable objects

Survivor Spaces

Reorganizes memory to reduce fragmentation

Old Generation

Area of memory where long-lived objects are allocated

Long-lived Objects
Contains long-lived objects

Major Collection
Collects old objects with major collection

Tenured Space
Objects surviving major collection are moved here

Fragmentation
Can lead to fragmentation

Generational Hypothesis

Most Objects Die Young

Most objects become garbage quickly

Short-lived Objects

Short-lived objects are allocated in young generation

Long-lived Objects

Long-lived objects are allocated in old generation

Different Collection Strategies

Different collection strategies for young and old generations

Improves Performance

Tuning

Heap Size
Adjust heap size for performance

Generation Sizes
Adjust sizes of generations

Collection Algorithms
Choose appropriate collection algorithms

Tuning Flags
Use tuning flags for customization

Performance Monitoring
Monitor performance with diagnost
tools

Sunit

Tools

jstat
Monitors JVM statistics

jmap
Generates heap dump

jvisualvm
Visualizes JVM performance

GCViewer
Analyzes GC logs

VisualGC
Provides visual representation

Sunit

Follow

Sunit Ghosh

to get #interesting
and latest #titbits
on #java, #AiMl, #cloud
technologies