목차

Java Garbage Collector 비교 3
- ***GC 종류와 장단점*** 3
- ***JDK 버전별 기본 GC*** 3
- *참조* 3

Java Garbage Collector 비교

GC 종류와 장단점

GC 종류	설명	장점	단점
Serial GC	단일 스레드로 동작하는 GC. 작은 힙 메모리에서 유용.	단순한 구현, 적은 메모리 사용.	큰 힙에서 지연 시간이 길어짐.
Parallel GC	병렬로 처리하는 GC. Throughput 중심.	다수의 스레드를 활용해 처리량 극대화.	지연 시간이 길어질 수 있음.
CMS (Concurrent Mark-Sweep)	Old 영역을 처리할 때 멀티스레드로 병렬 처리.	GC 지연 시간이 짧음.	메모리 단편화 문제. stop-the-world 이벤트 발생 가능.
G1 (Garbage First)	힙을 Region 단위로 나눠 GC를 실행. 지연 시간과 처리량의 균형을 중시.	지연 시간 단축 및 큰 힙에 적합.	구성 및 튜닝 복잡. 초기 오버헤드 있음.
ZGC (Z Garbage Collector)	초저지연 GC. 매우 큰 힙(테라바이트 규모)에서 유용.	지연 시간이 짧고 큰 힙에 적합.	초기 버전에서는 일부 플랫폼에서만 사용 가능.
Shenandoah GC	초저지연 GC. 힙 메모리 크기와 관계없이 낮은 지연 시간 제공.	지연 시간 짧음. G1보다 빠른 수집 가능.	CPU 사용량이 상대적으로 높음.
Epsilon GC	“No-Op” GC. 메모리를 수집하지 않음.	매우 낮은 오버헤드. 테스트 용도에 적합.	메모리 부족 발생 시 OutOfMemoryError를 유발.
Generational ZGC	ZGC의 세대 구분 버전(자바 21 도입).	ZGC의 장점과 함께 세대 구분에 의한 최적화.	새로운 기술로서 튜닝 경험이 적음.

JDK 버전별 기본 GC

자바 버전	기본 GC
자바 8	Parallel GC
자바 9~10	G1 GC (기본 GC로 변경됨).
자바 11	G1 GC
자바 12~15	G1 GC, 하지만 새로운 GC(ZGC, Shenandoah)도 옵션으로 사용 가능.
자바 16~17	G1 GC, ZGC 및 Shenandoah 성능 개선.
자바 18~20	G1 GC, Generational GC를 실험적으로 도입.
자바 21	Generational ZGC (기본 GC로 변경되지는 않았지만, 성능 최적화를 위해 도입됨).

참조

JDK 8을 포함한 대부분의 모든 버전에서 기본적으로 ‘G1 ” 방식을 권장하며21 이후 버전부터 점진적으로 ZGC로 변경될것으로 예상됨.

From:
https://atl.kr/dokuwiki/ - **AllThatLinux!**

Permanent link:
https://atl.kr/dokuwiki/doku.php/java_garbage_collector_%EB%B9%84%EA%B5%90?rev=1733373491

Last update: **2024/12/05 04:38**