G1, or Garbage-First, is a garbage collector for the Java Virtual Machine (JVM). It was introduced in Java 7 Update 4 and is the default garbage collector for Java 9 and later.

G1 is a generational garbage collector, which means that it divides the heap into two or three generations:

* The young generation is where newly allocated objects are placed.
* The old generation is where objects that have survived multiple garbage collections are placed.
* The tenured generation is a subset of the old generation that is specifically reserved for long-lived objects.

G1 is also a regionalized garbage collector, which means that the heap is divided into a number of equally sized regions. This makes it easier for G1 to track live objects and to reclaim memory.

G1 works by first marking all of the live objects in the heap. This is done in two phases:

* The initial mark phase is a short, concurrent phase where all of the live objects in the young generation are marked.
* The concurrent mark phase is a longer, concurrent phase where all of the live objects in the old generation are marked.

Once all of the live objects have been marked, G1 then collects the garbage. This is done by copying all of the live objects from the old generation to a new region. The old region is then marked as free memory.

G1 is designed to meet garbage collection pause time goals with high probability while achieving high throughput. It does this by using a number of techniques, including:

* Parallel garbage collection: G1 garbage collection is performed concurrently with the application threads. This helps to minimize the impact of garbage collection on application performance.
* Regionalized garbage collection: G1 divides the heap into a number of equally sized regions. This makes it easier for G1 to track live objects and to reclaim memory.
* Garbage first collection: G1 collects the regions with the least live data first. This helps to minimize the amount of time that is spent copying live objects during garbage collection.

G1 is a powerful garbage collector that can be used to improve the performance and scalability of Java applications. It is the default garbage collector for Java 9 and later, and it is a good choice for applications that require predictable garbage collection pause times.

Here are some of the benefits of using G1:

* Predictable garbage collection pause times: G1 can be configured to meet specific garbage collection pause time goals. This is important for applications that cannot tolerate long garbage collection pauses.
* High throughput: G1 can achieve high throughput, even with large heaps. This is because G1 garbage collection is performed concurrently with the application threads.
* Low memory fragmentation: G1 is designed to minimize memory fragmentation. This helps to improve the performance of applications that allocate and deallocate objects frequently.

Here are some of the drawbacks of using G1:

* Complex configuration: G1 has a number of configuration options that can be used to fine-tune its performance. This can make it difficult to configure G1 for optimal performance.
* Not as well-understood as other garbage collectors: G1 is a relatively new garbage collector, and there is not as much information available about its performance as there is for other garbage collectors.

Overall, G1 is a powerful garbage collector that can be used to improve the performance and scalability of Java applications. However, it is important to understand the benefits and drawbacks of G1 before deciding whether to use it.