Java™

ORACLE®

Working through Garbage Collectors

Vaibhav Choudhary (@vaibhav_c)
Java Platforms Team
https://blogs.oracle.com/vaibhav

Java
Your
Next
(Cloud)

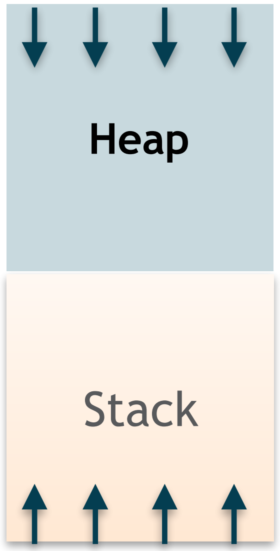
Safe Harbor Statement

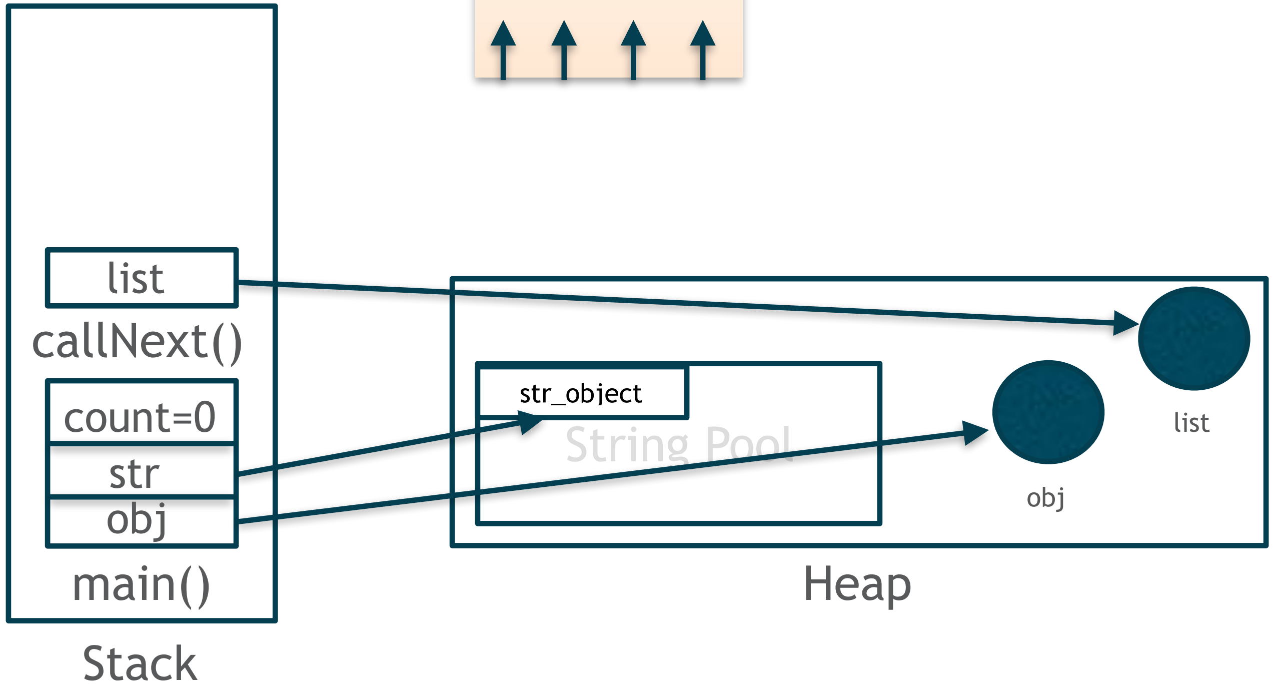
Agenda of the day …

1. Basics of Garbage Collector
2. Concurrent Mark and Sweep (CMS)
3. CMS implementation and challenges
4. Garbage First (G1)
5. G1 implementation and comparison to CMS

Stack and Heap in Java

```
public static void main(String[] args) {

     Object obj = new Object();
     String str = new String();
     int count = 0;
     // Do some work here
     callNext();

}

public void callNext() {

     List list = new ArrayList();
     // some work
}
```

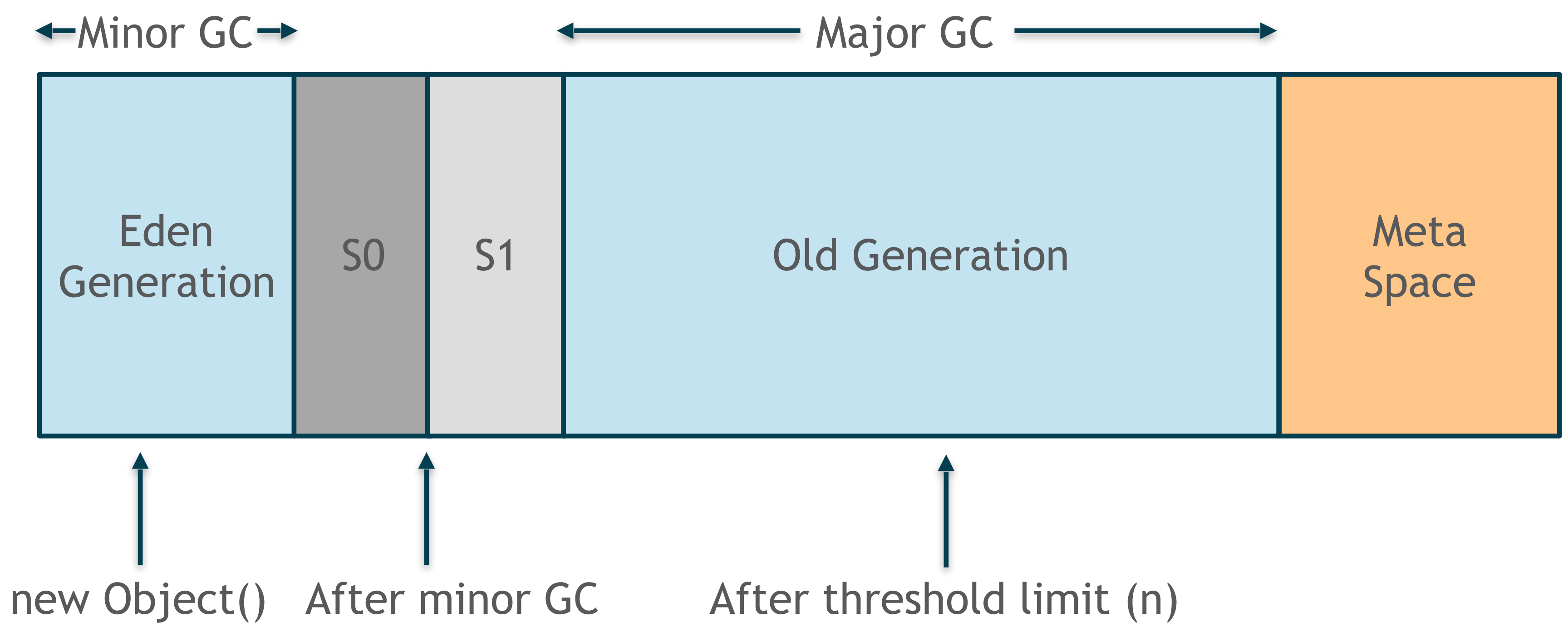




Definition

- **Mark**
  - ➤ find and mark all accessible objects
- **Sweep**
  - ➤ scans through the heap and reclaims all the unmarked objects
- **Copy**
  - ➤ copies all live objects from one part of the heap to another empty part
- **Compaction**
  - ➤ moving all the live objects into contiguous memory locations

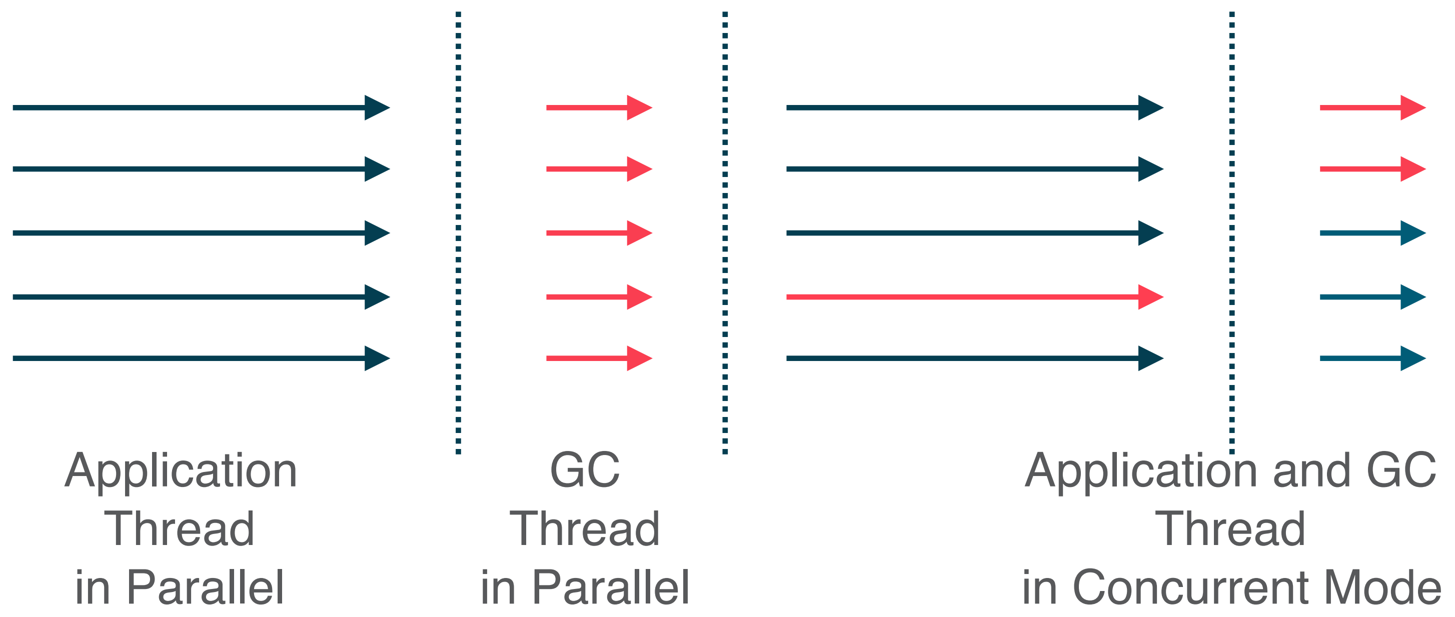
Java Heap Structure



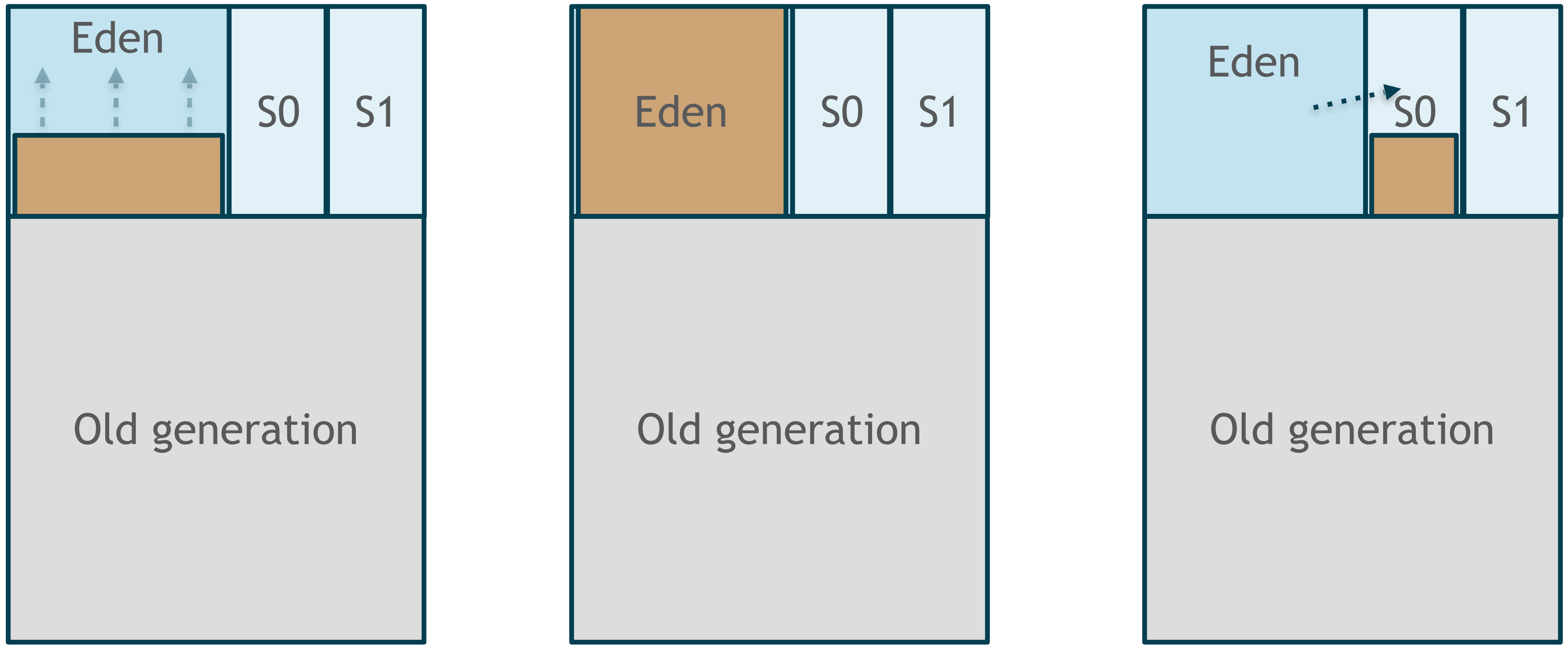
Weak Generational Hypothesis :

- Most of the object die young.
- There are very few old to young reference.

Parallel And Concurrent

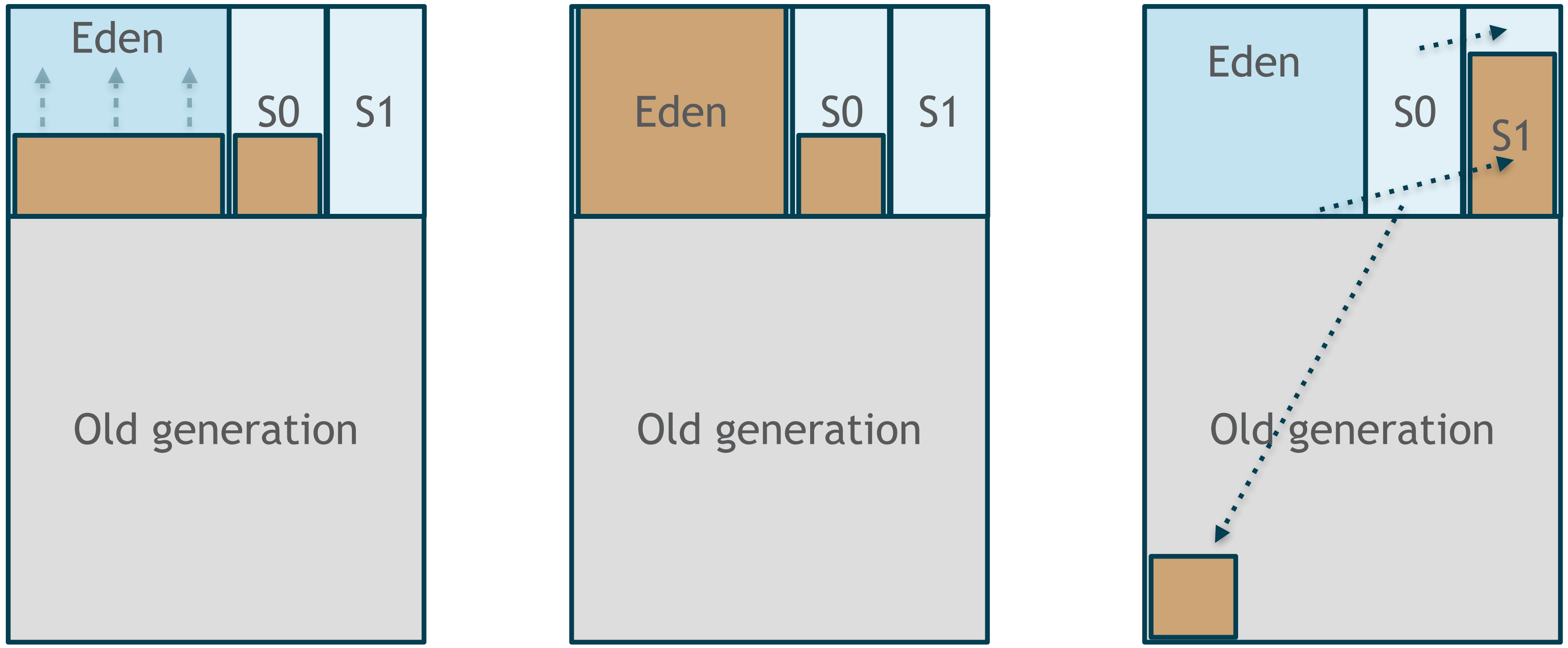


CMS Simplified (In Young generation)…



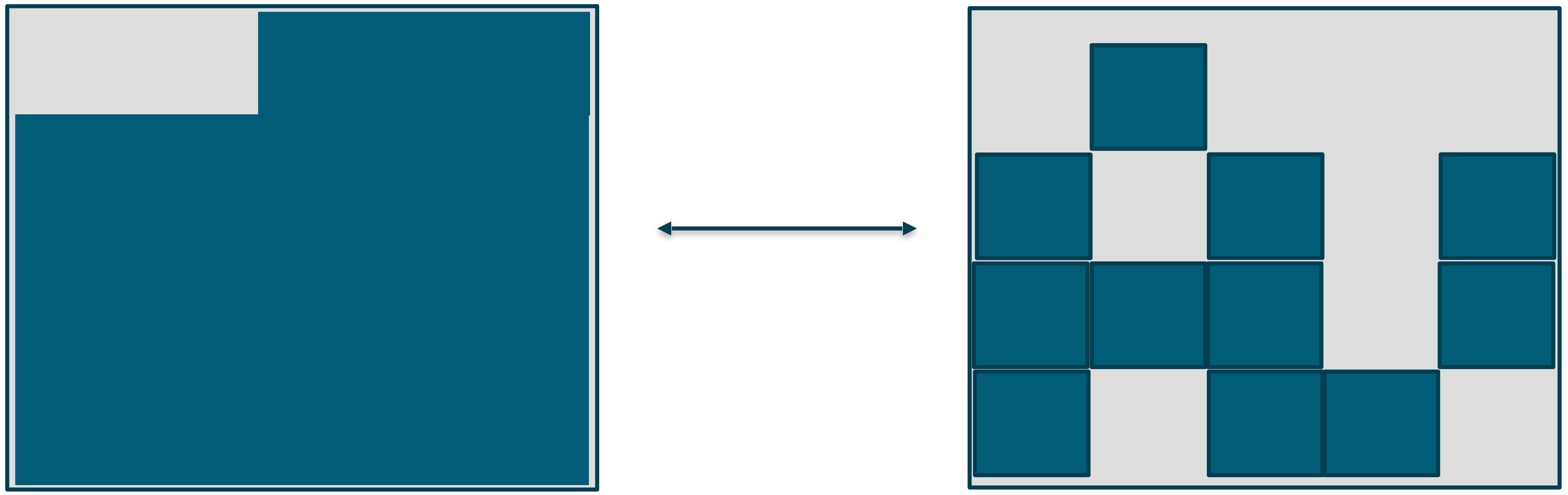
When eden space will be full. Minor GC will be triggered.
Usages of S0 and S1.

CMS Simplified (In Young generation)…



Empty Eden and one of the Survivors after Minor GC completion.
Stop the world process.
Some objects can be promoted to Old generation.

CMS Simplified (In Old generation)…



May be the first snapshot of Old generation

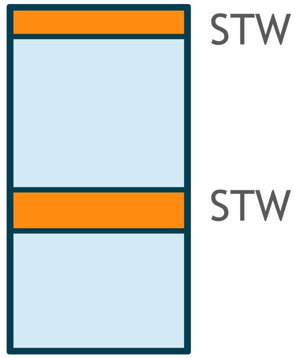
Old generation will mostly look like this

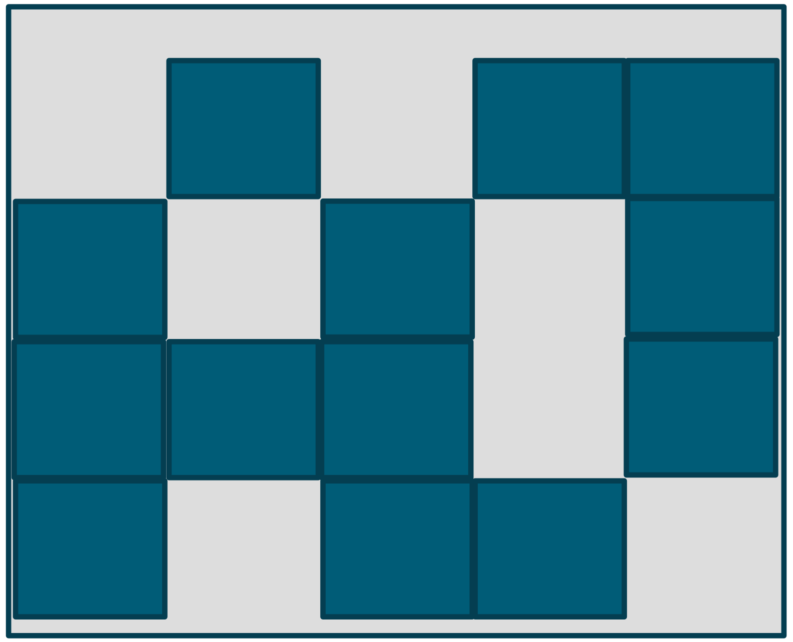
No compaction in CMS
Mostly concurrent

CMS Simplified (In Old generation)…

CMS phases :-

1. Initial marking phase.
2. Concurrent Marking phase.
3. Concurrent Pre-cleaning Phase.
4. Remarking phase.
5. Concurrent Sweep Phase.
6. Concurrent Reset Phase.





CMS Simplified (In Old generation)…

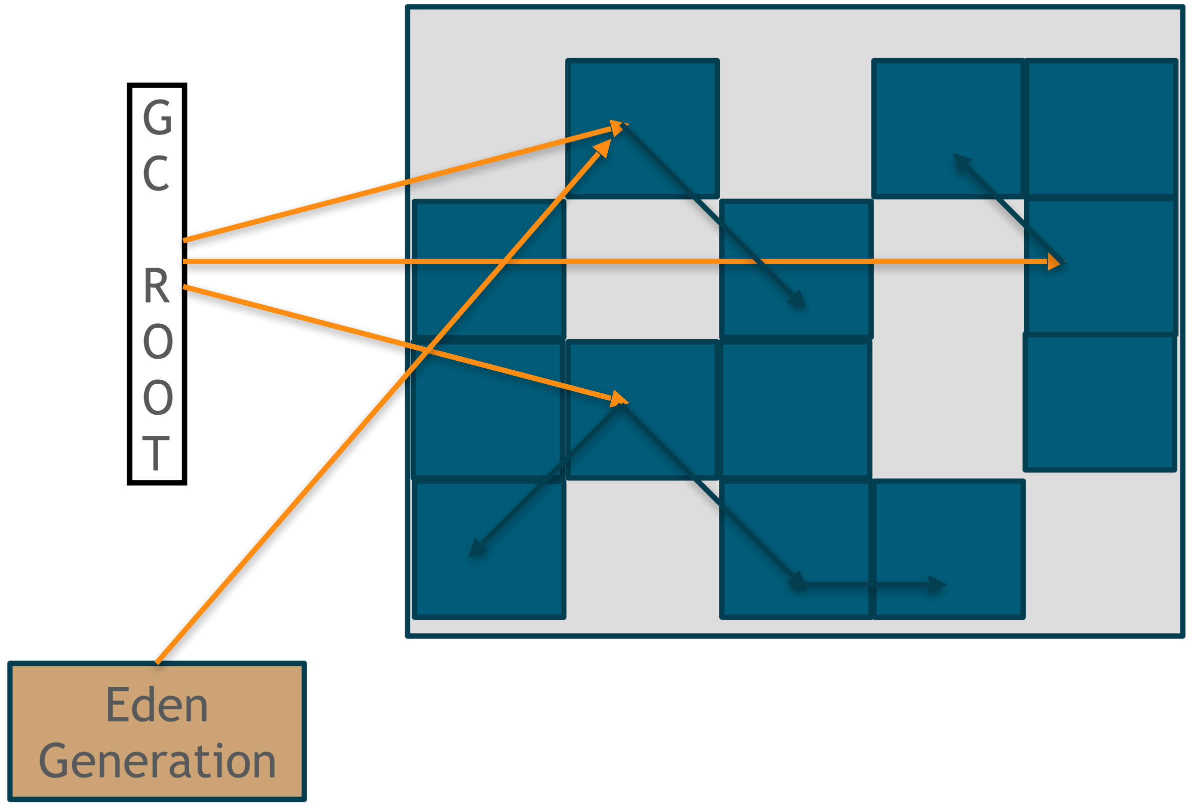
Initial Marking Phase :-

1. Scan the object directly reachable from root or from Eden Generation.

2. All the mutator thread need to be stopped (STW).

3. STW - We don't want to mess up with this information.

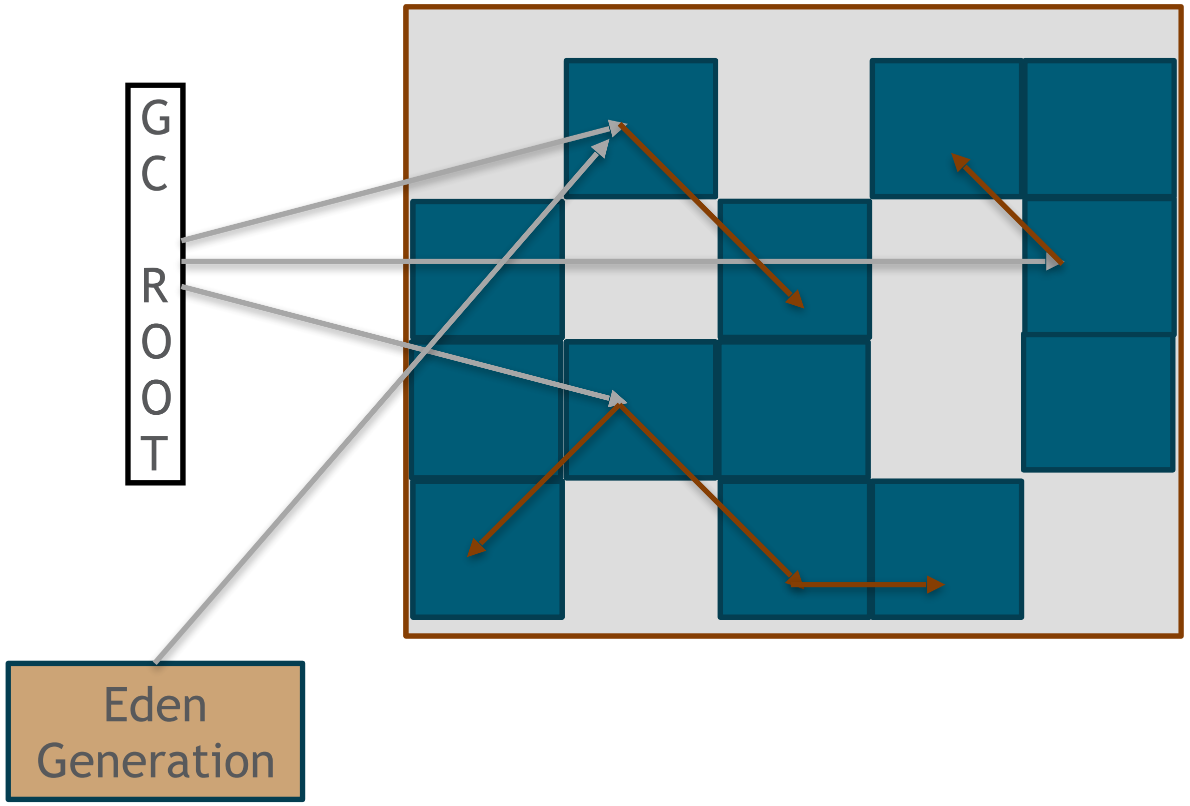
4. Generally small pause in nature.



CMS Simplified (In Old generation)…

Concurrent Marking Phase :-

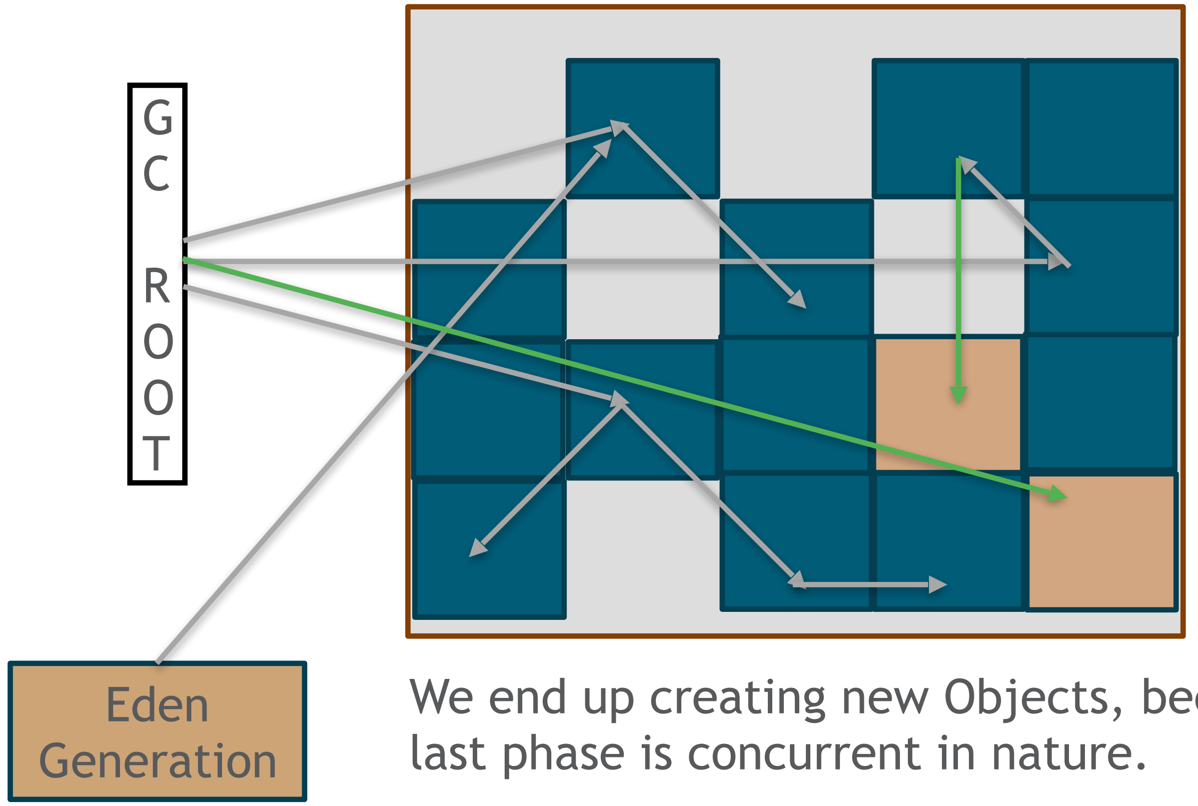
1. Scan **almost** all live objects from the object marked from phase 1.
2. This will be done in concurrent fashion.
3. And this will lead to a remarking phase.
4. It will take time but it's concurrent.



CMS Simplified (In Old generation)…

Remarking Phase :-

1. Scan the newly created objects.
2. We need to do STW.
3. Generally, small pause in nature.



We end up creating new Objects, because last phase is concurrent in nature.

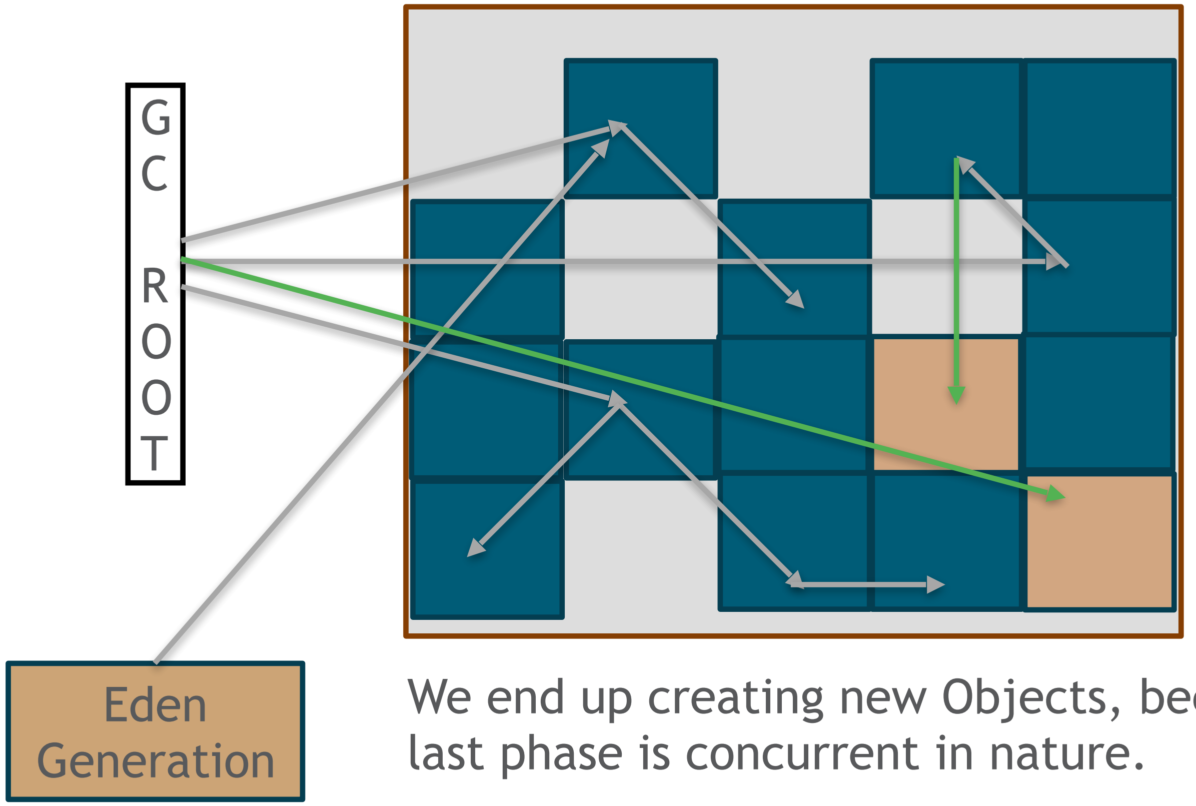
CMS Simplified (In Old generation)…

Remarking Phase :-

1. Scan the newly created objects.

2. We need to do STW.

3. Generally, small pause in nature.

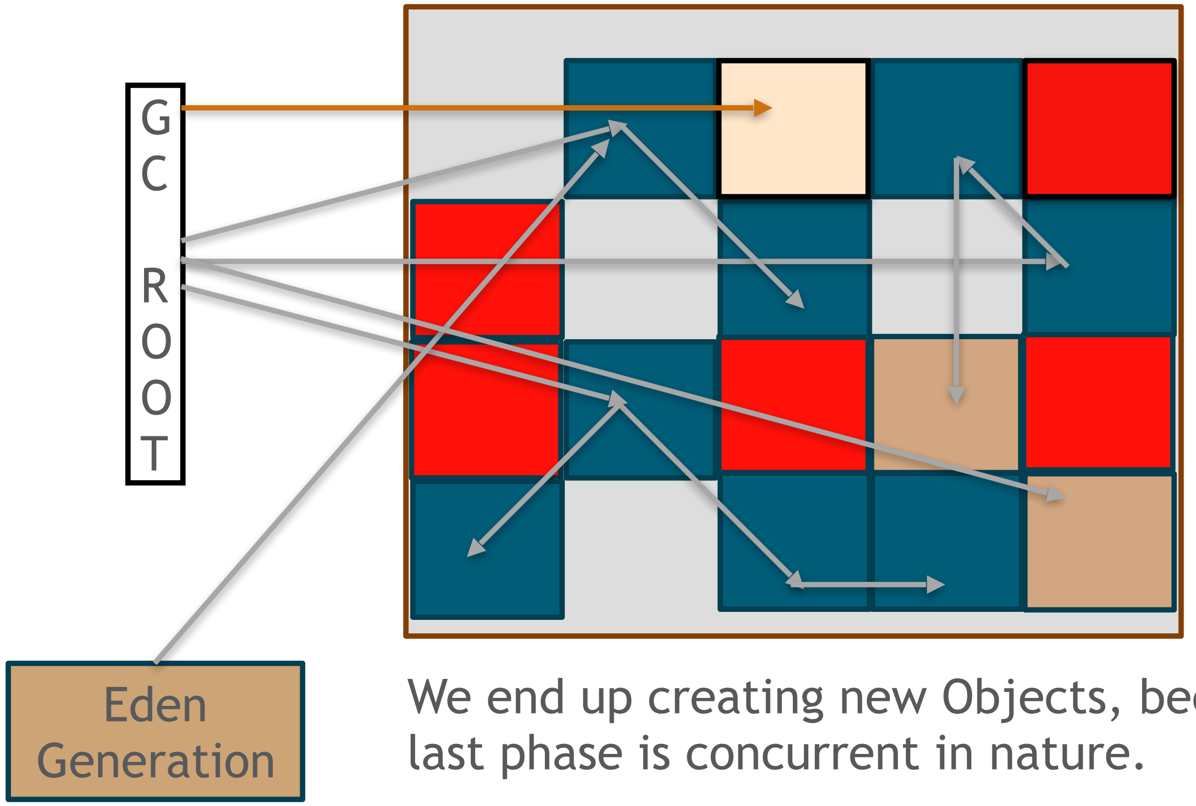


We end up creating new Objects, because last phase is concurrent in nature.

CMS Simplified (In Old generation)…

Concurrent Sweep Phase :-

1. Start sweeping all the non-marked Object.
2. Concurrent in nature.
3. Red boxes can be claimed by GC.



We end up creating new Objects, because last phase is concurrent in nature.

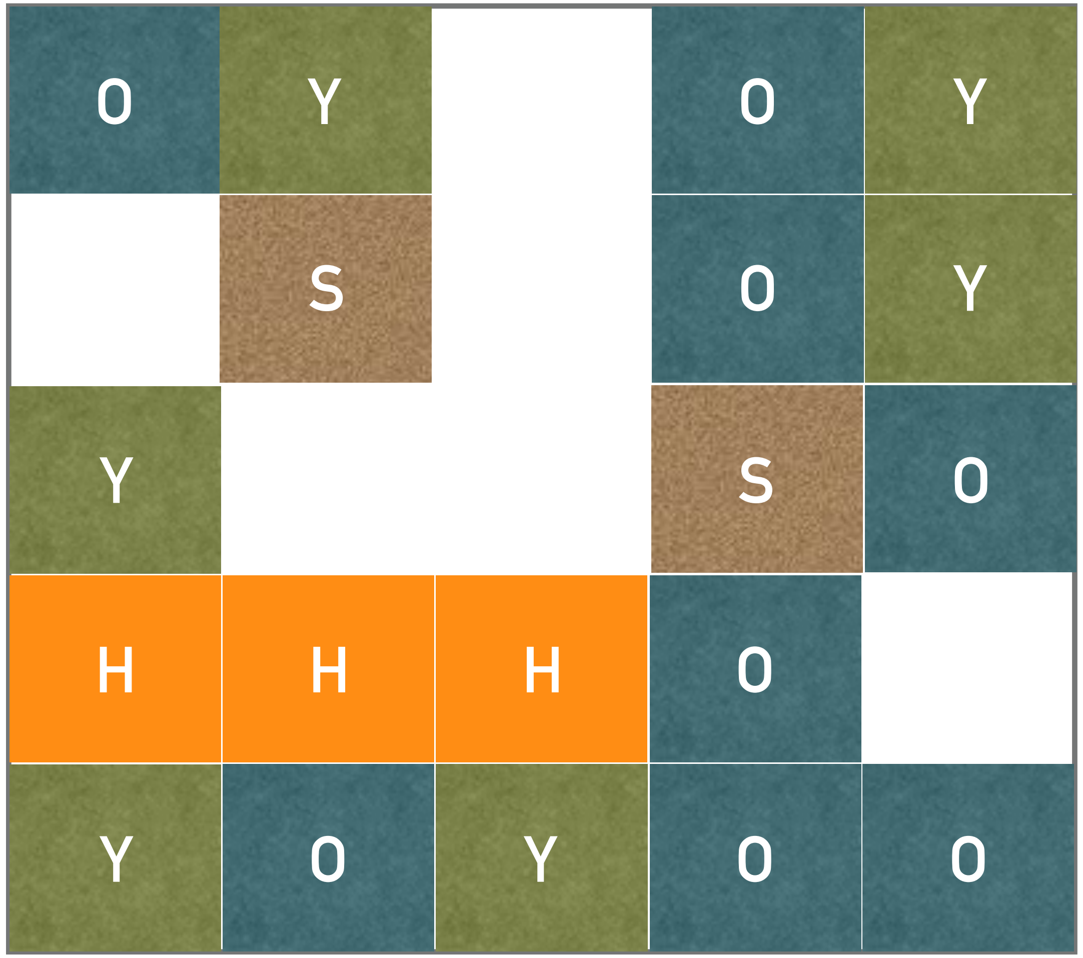
Final thoughts for CMS

- It's almost concurrent other than some small STW's.
- Memory Fragmentation - so can't fit large object.
- Less predictable.
- Can Lead to **promotion failure -** Young to old promotion is not happening because object is too big or old space is almost full.
- Can Lead to **Concurrent mode failure** - Old generation not finished the collection work and Old generation is full. STW and then run a different GC algorithm, probably Mark-Sweep-Compact.
- If application is working fine with CMS, let it work !

Welcome to Garbage First (G1)

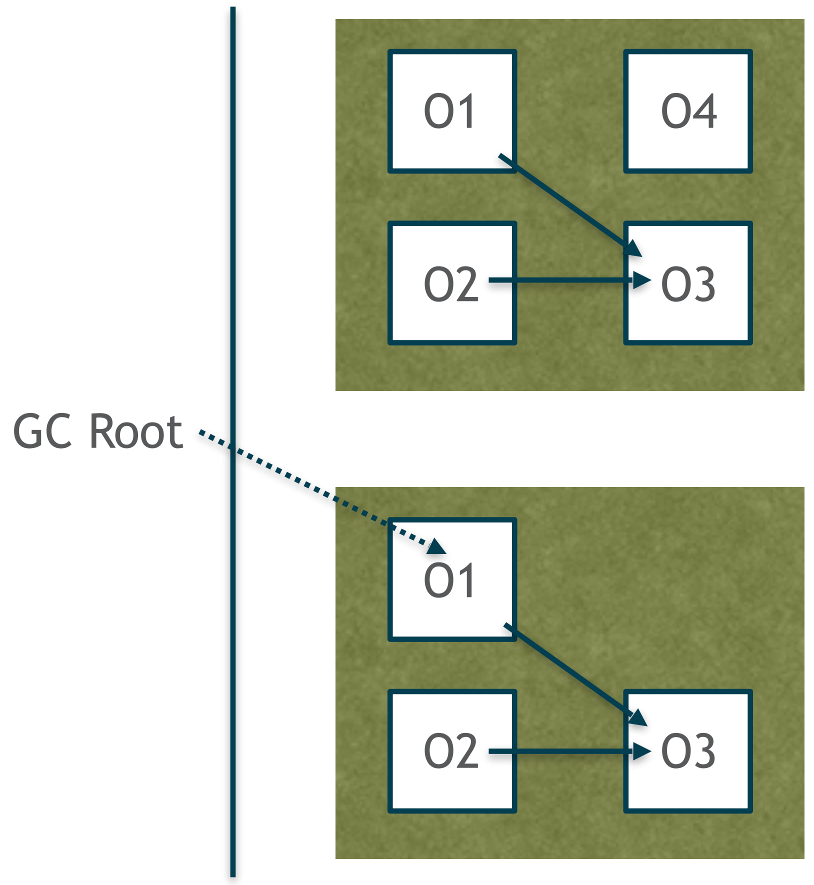
- Default GC algorithm for JDK9.
- G1 Goals
  - Low latency
  - Predictable (Can’t be 100 percent)
  - Easy to use (Less parameter settings)
- Concurrent, Parallel and better Compacting.
- If you are not on JDK9, use -XX:+UseG1GC.
- Careful with your greedy throughput desires.

Garbage First (G1) - Memory layout



- Memory is divided into small regions
- More than 2000 regions
- More flexible boundaries
- Use -XX:+G1HeapRegionSize
- Different regions :
  - Young
  - Survivor
  - Old
  - Humongous

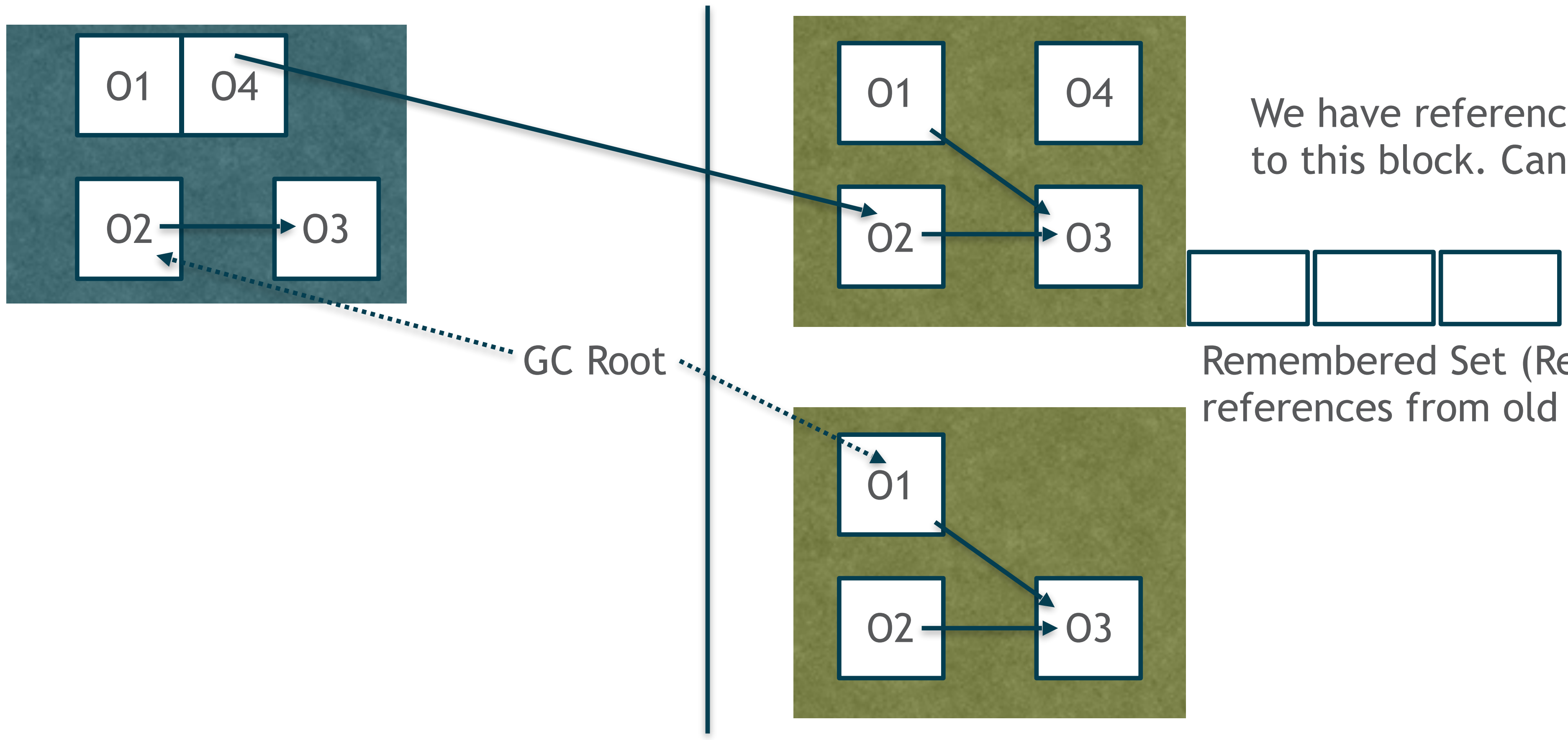
G1 - Young region View



None of the object is reachable from GC root in first block. Looks like the complete region can be collected. **(No you can't)**

GC root is pointing to some object. Can't collect this region.

G1 - Young region View

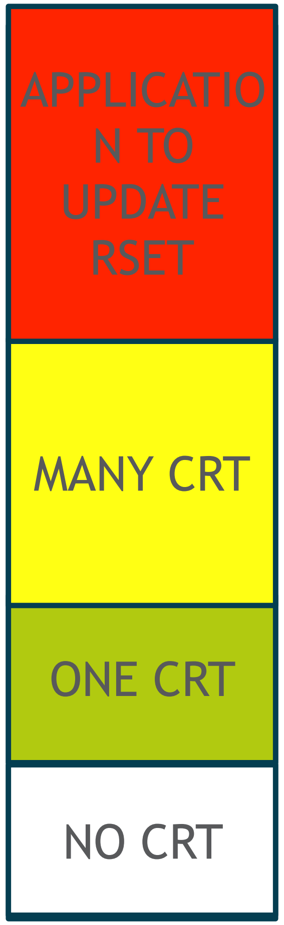


We have reference from Old region to this block. Can't be collected.

Remembered Set (Remembers the references from old region)

G1 RS and Dirty Card Table Concept

- A card table is a type of remembered set.
- Hotspot uses byte array as a Card Table.
- Each byte is referred as a Card which corresponds to range of address in Heap.
- Dirtying a card means changing the value of byte.
- Processing a card means seeing a old to young pointer.
- -XX:+ConcRefinementThread



G1 - Ease to use

- java -Xmx50m -XX:+UseG1GC -XX:+MaxGCPauseMillis=200 -jar Java2DDemo.jar
- MaxGCPauseMillis=200 - A soft goal. G1 will try to respect as much as possible, but can’t guarantee you.
- Important to know
  - -XX:+InitiatingHeapOccupancyPercentage=45

G1 - Young GC Phases

- Stop the world event.
- Builds collection Set (cSet) for the regions which are subject of collection.
- In Young GC, cSet will contain :-
  - Eden Region
  - Survivor Region

G1 - Young GC Phases

- Phase 1
  - G1 [Young] Root Scanning
  - Find out the GC Roots like old time.
- Phase 2
  - G1 [Young] Update RSet
  - Update RSet from dirty card queue.
- Phase 3
  - G1 [Young] Process RSet
  - Detects old to young generation pointers.

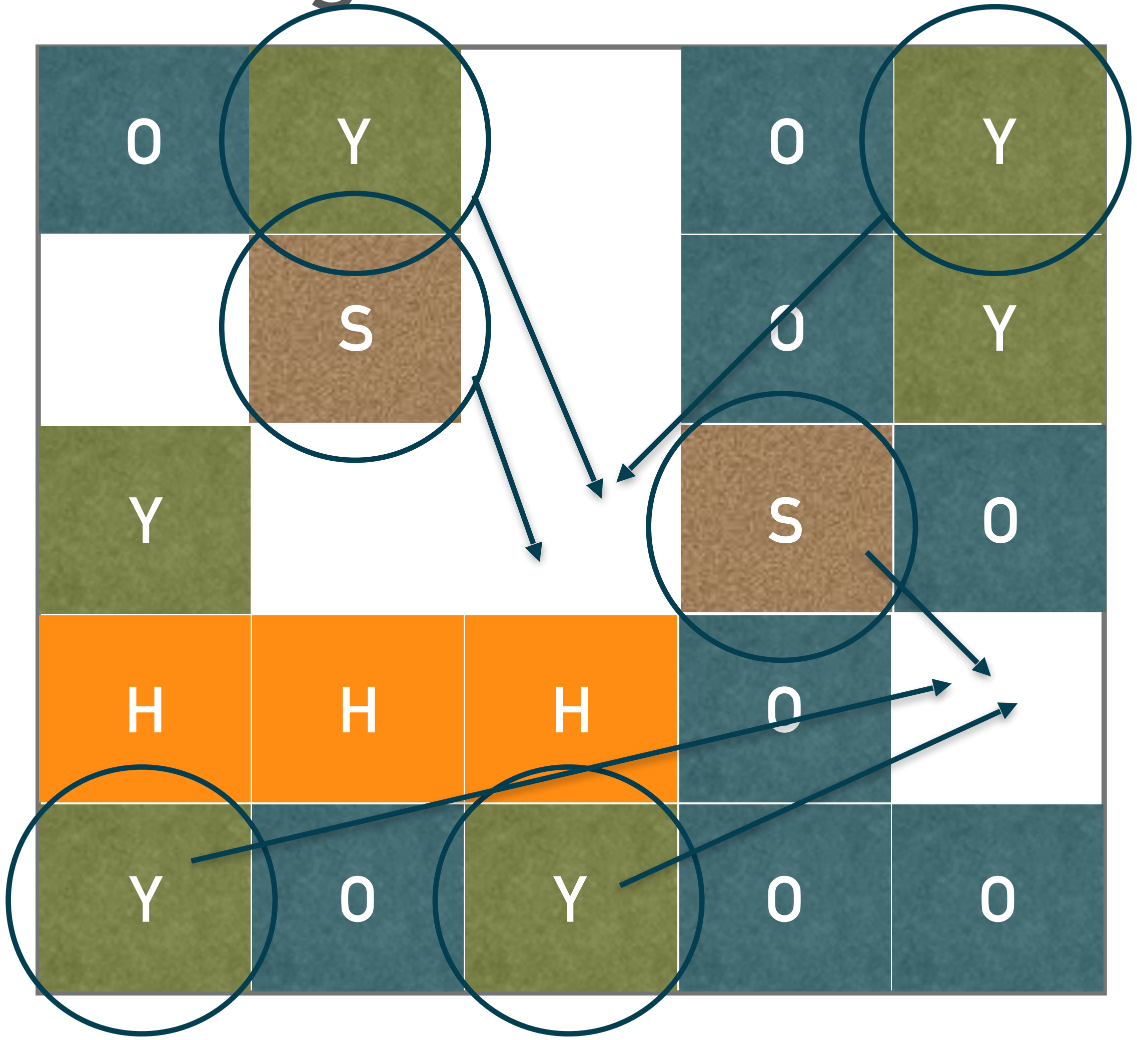
G1 - Young GC Phases

- Phase 4
  - G1 [Young] Object copying
  - Traverse object graph
  - Copy to either New Eden Region or Survivor Region
- Phase 5
  - G1 [Young] Reference Processing
  - Finding the type of reference while copying (Phase-4)

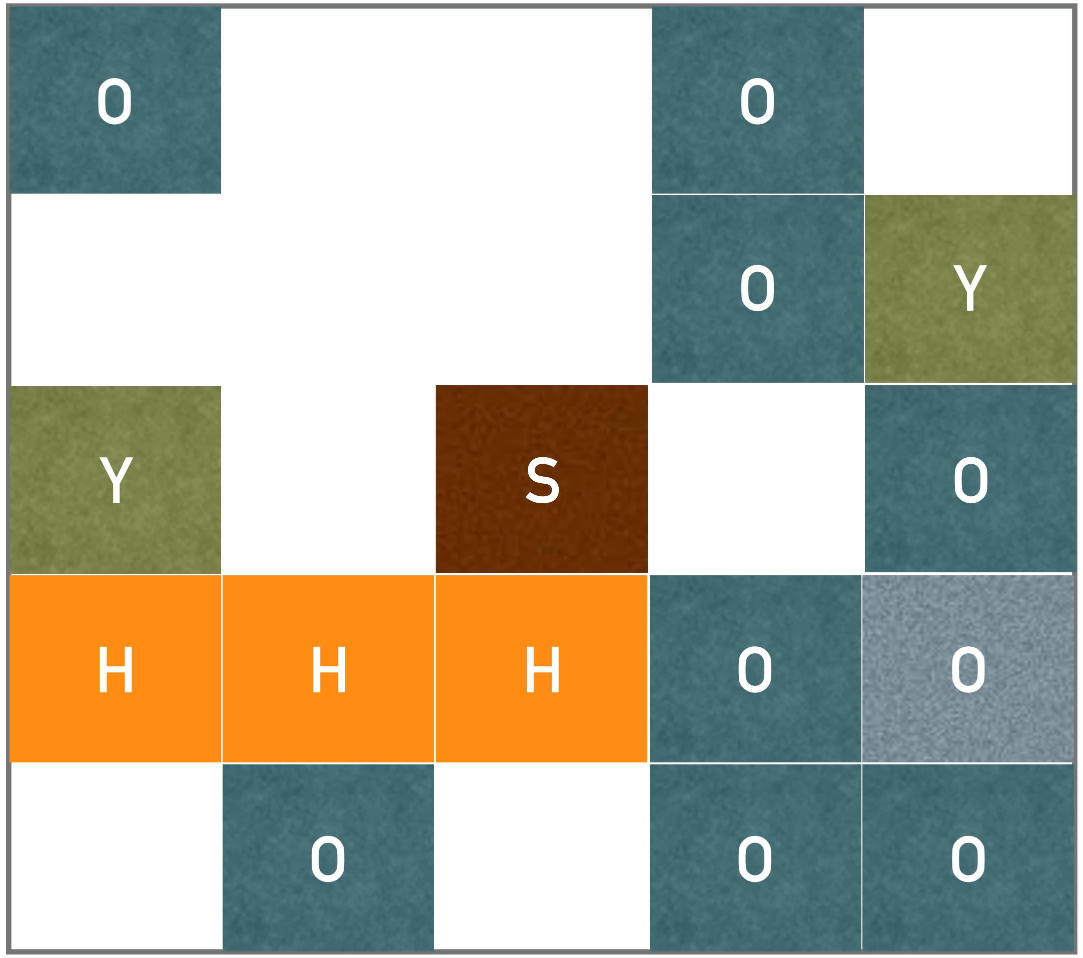
G1 - Young GC Phases

- Dynamically setting up no. of Eden/Survivor Region
- MaxPauseMillis - Can increase or decrease the no. of region to respect the pause time.
- After fulfilling the latency goals, it will try to fulfil the throughput goal.
- So, if possible, don’t set Xmn
- Shrinking/Expansion can happen from 20-80 percent.

G1 - Young GC View



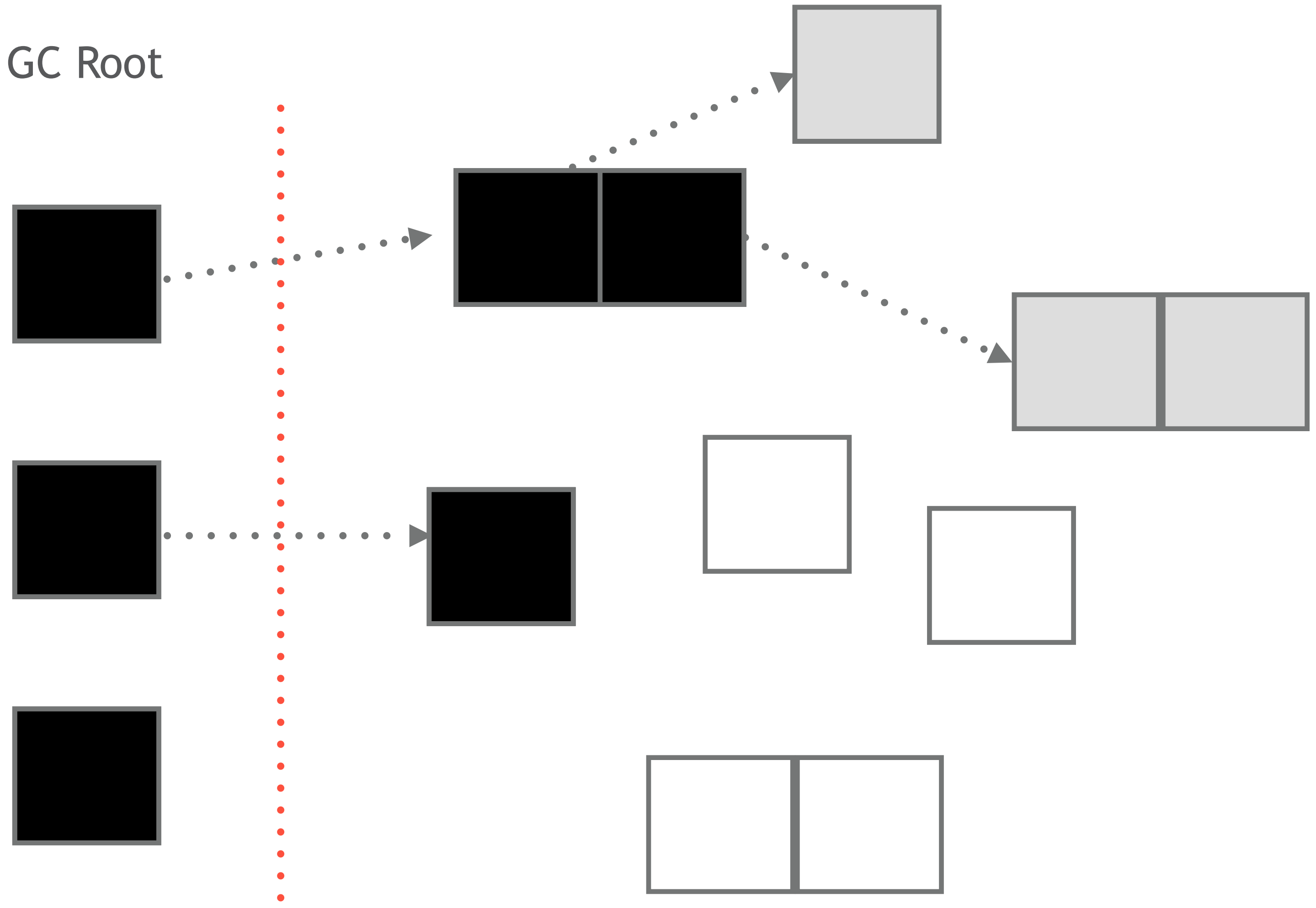
G1 - Young GC View



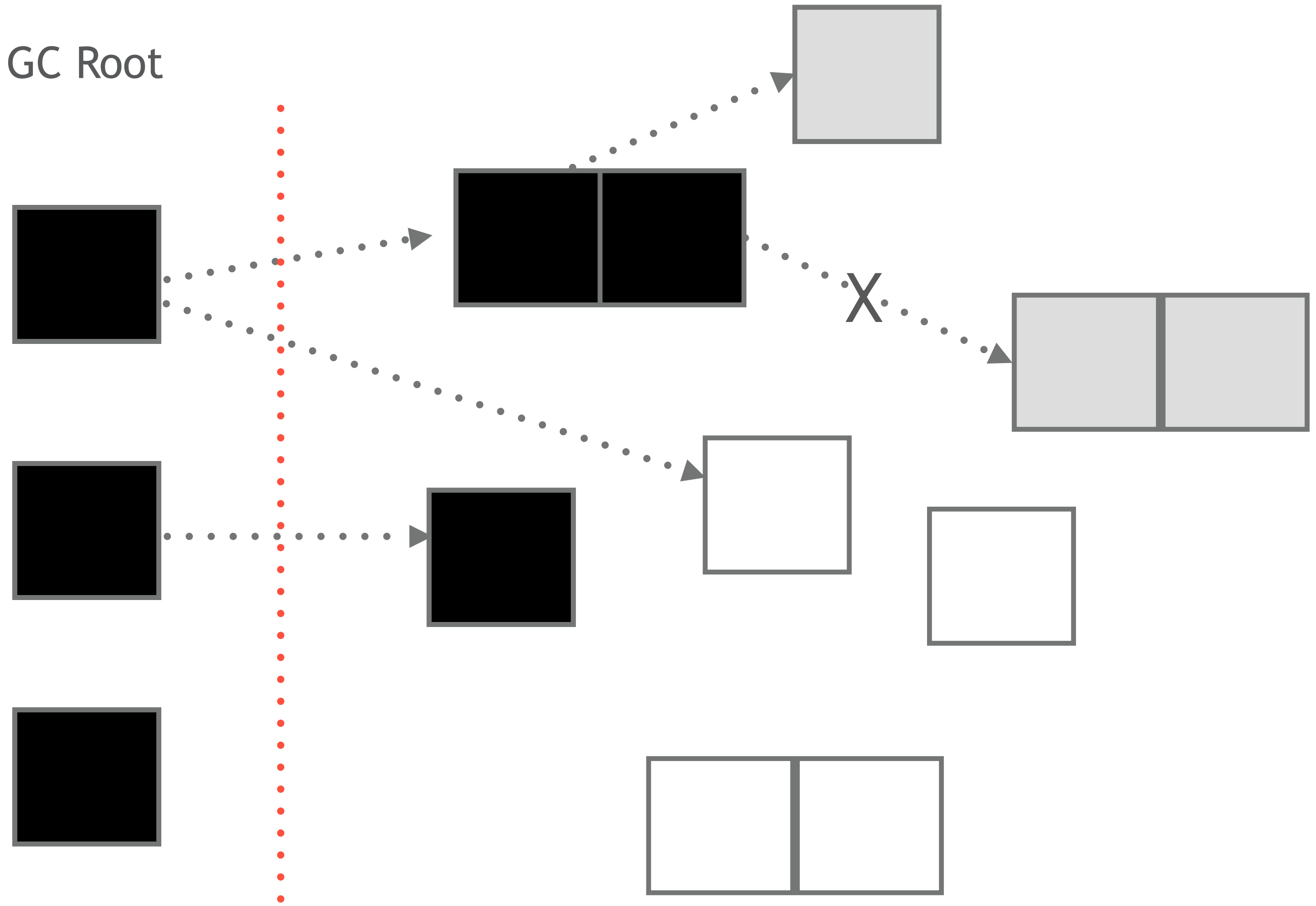
G1 - Old GC Phases

- Kicks out when the heap occupancy is 45%.
  - -XX:+InitialHeapOccupancyPercentage=<n> (Can be changed)
- G1 uses Concurrent Marking in Old region
  - Uses STAB [Snapshot at the beginning]
  - Tri-color Marking
  - Floating Garbage

Tri-color Marking



Concurrency Problem - Floating Garbage



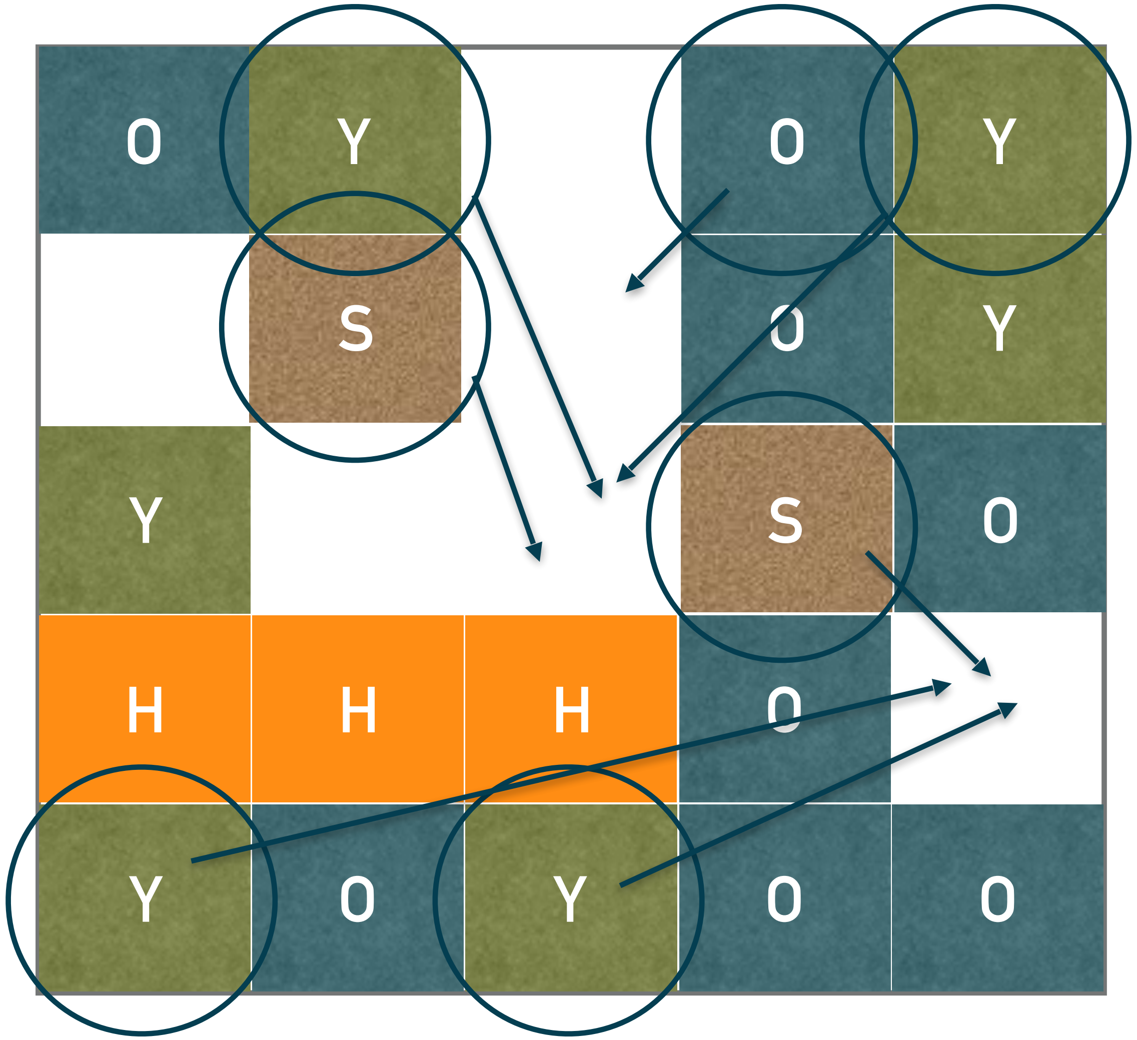
G1 - Old GC Phases

- Performs a Young GC
  - Piggybacks root detection.
- Concurrent Old region marking starts
  - Also, profile per region liveliness.
  - Which region, I can claim the best garbage.
- Remarking Phase
  - Process STAB
  - Process Reference Queue
- Cleanup Phase
  - No tree traversal in the region - So, its full of Garbage.

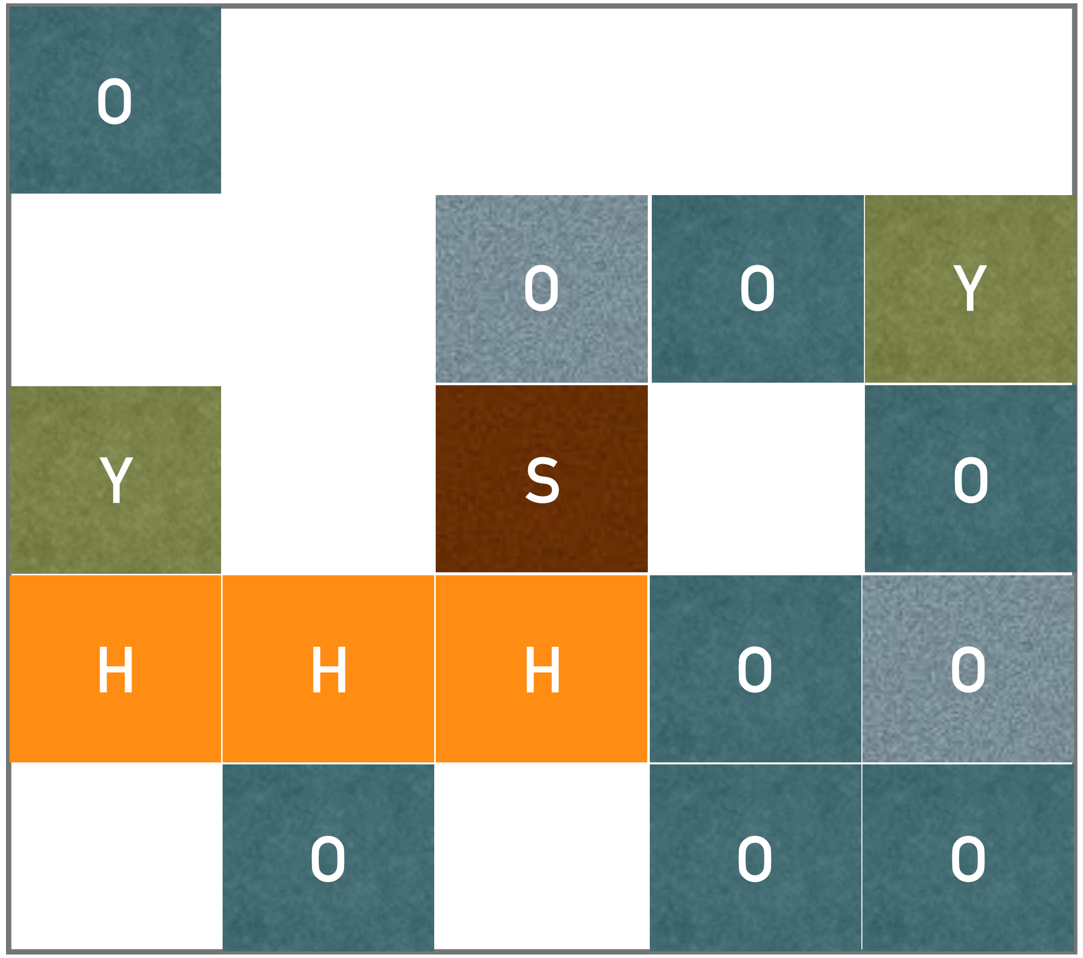
G1 - Mixed GC Phases

- Partial Regions
  - With young generation collection, G1 adds some area of Old region as well to perform GC.
  - -XX:MixedGCCountTarget
  - Which old region to give ? - One which can give max benefit
    - -XX:G1MixedGCLiveThresholdPercentage
  - Not interested in seeing the regions which are full of live objects.
    - -XX:G1HeapWastePercentage

G1 - Mixed GC View



G1 - Mixed GC View



G1 - Best Practices

- Avoid setting up too much of parameter.
  - You will end up G1 behaving abruptly.
- Avoid full GC. Look at the adaptive policy.
- Avoid allocation Failure
  - Very similar to Concurrent Mark Failure in CMS.
  - It will call STW and a full GC
  - Issue with heap or with the code
- Careful with Humongous Regions. Change region size, if it is more.

Golden References

- Charlie Hunt - G1 Evaluation
- Simone Bordet - G1 Details and Tuning
- G1 - Devoxx talks
- Oracle Documentation on G1
- Poonam Bajaj Blogs

- If you are seeing any unexceptional behaviour, feel free to contact us.