How Garbage Collection Works in the Oracle JVM

Kevin Jones

@kevinrjones

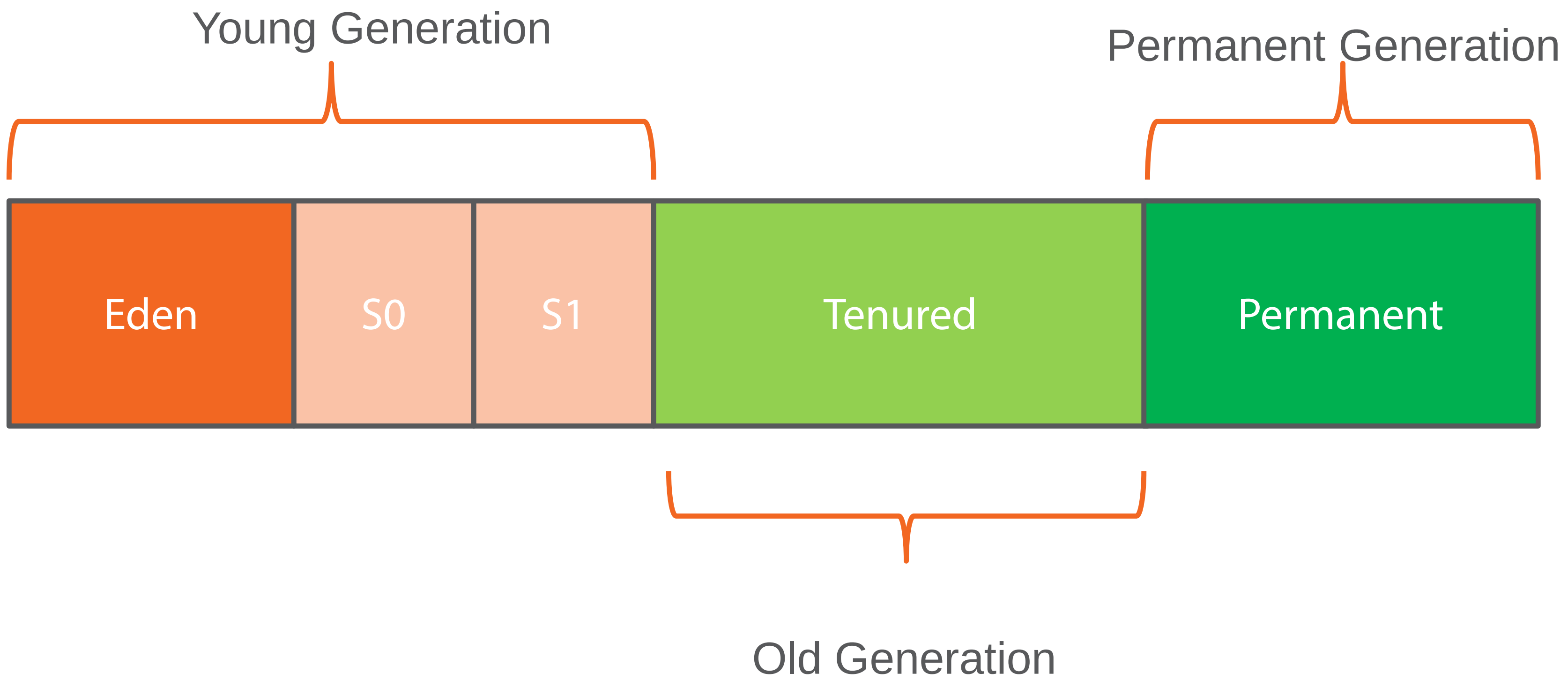
Introduction

- Things to consider
  - Stop the world events
  - Memory Fragmentation
  - Throughput
- Different GCs
  - Generational GC
  - Copying
  - Mark and Sweep
- Multi-core

Basic Ideas

- Has a ‘young generation' and an ‘old generation’
- Most initial objects allocated in ‘Eden space’
  - Part of young generation
- Young generation also has two ‘survivor’ spaces
  - Objects that survive a GC get moved to the survivor space
  - Only one survivor space in use at a time
  - Objects copied between survivor spaces
- Old generation is where long lived objects go to die

Memory – The Players



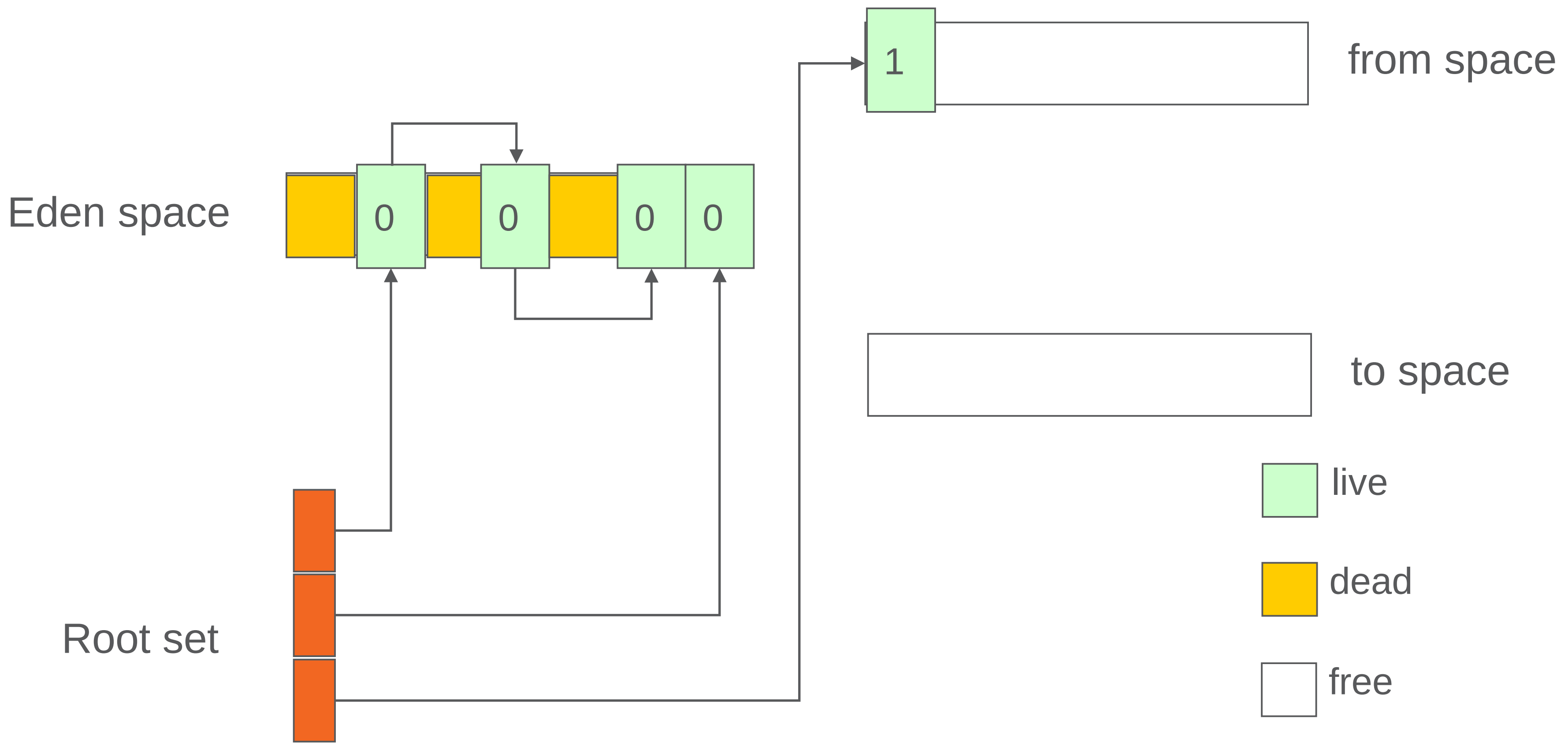
Young Generation

- Most objects live for a very short time
  - i.e. you die young or live ‘forever’

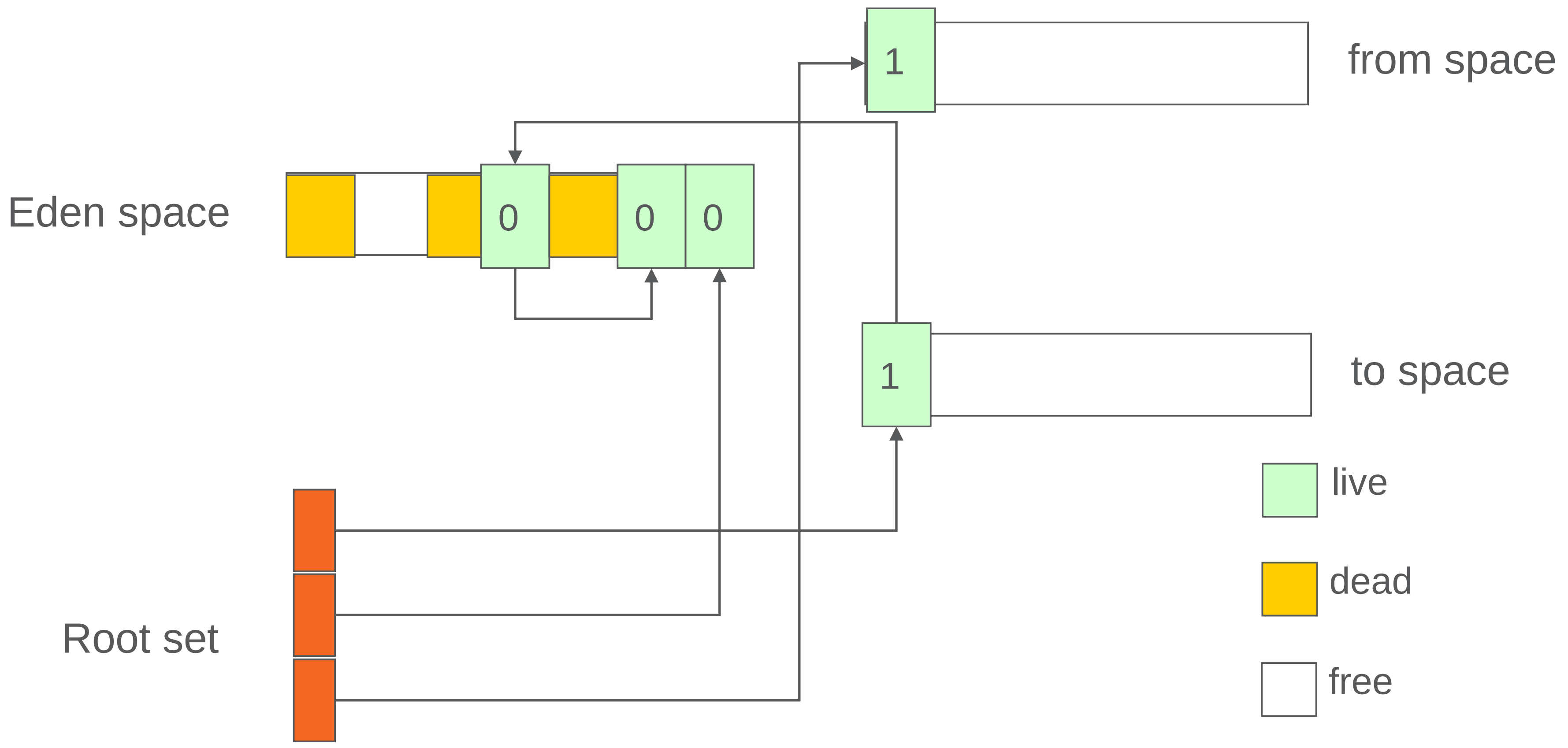
Minor Garbage Collection

- Objects allocated into Eden space
  - When GC runs objects are copied to ‘newer’ survivor space
  - Objects from ‘older’ survivor space also copied to ‘newer’ survivor space
  - Survivor spaces are swapped
- New objects allocated into Eden

Young Generation Allocation



Young Generation Allocation

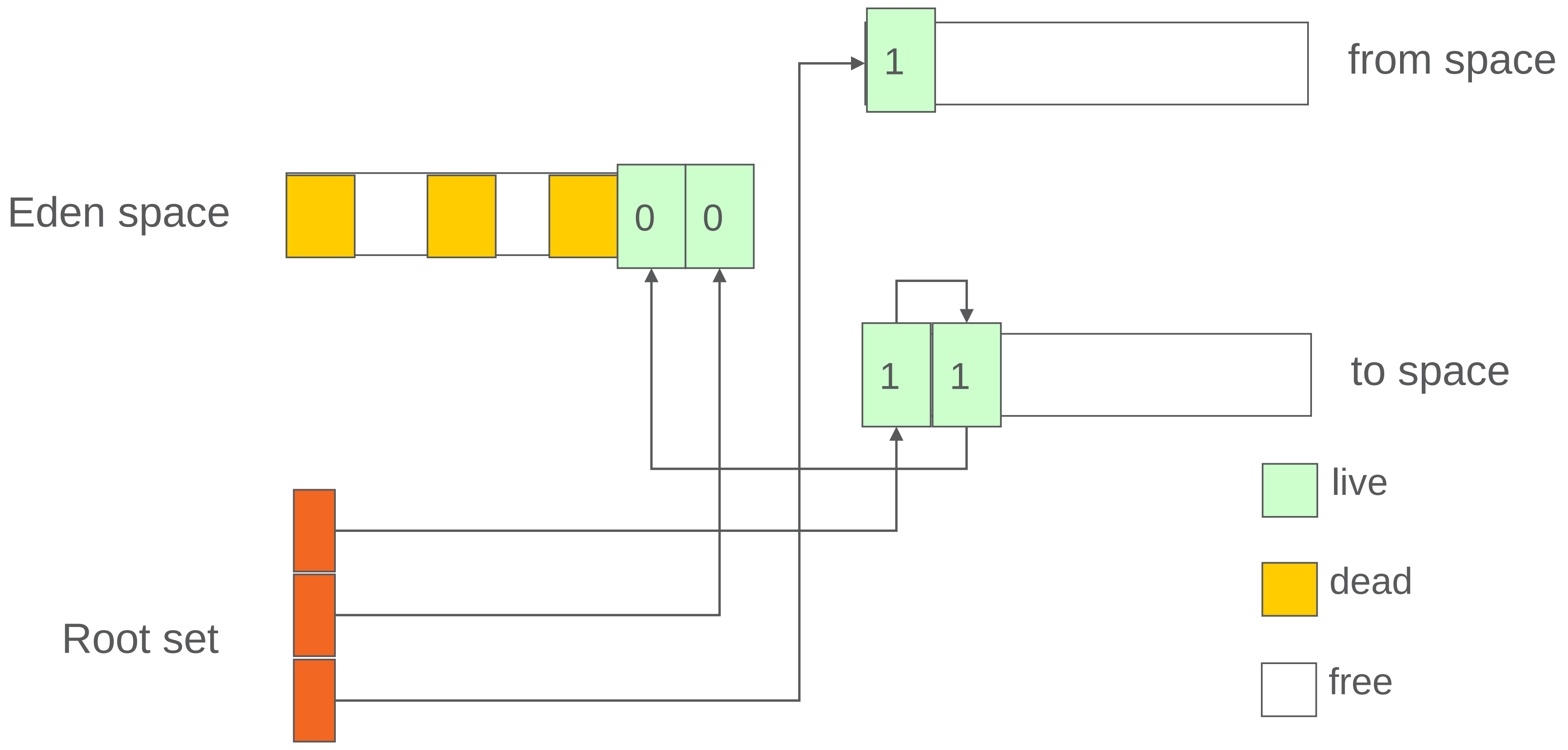


🟩 live

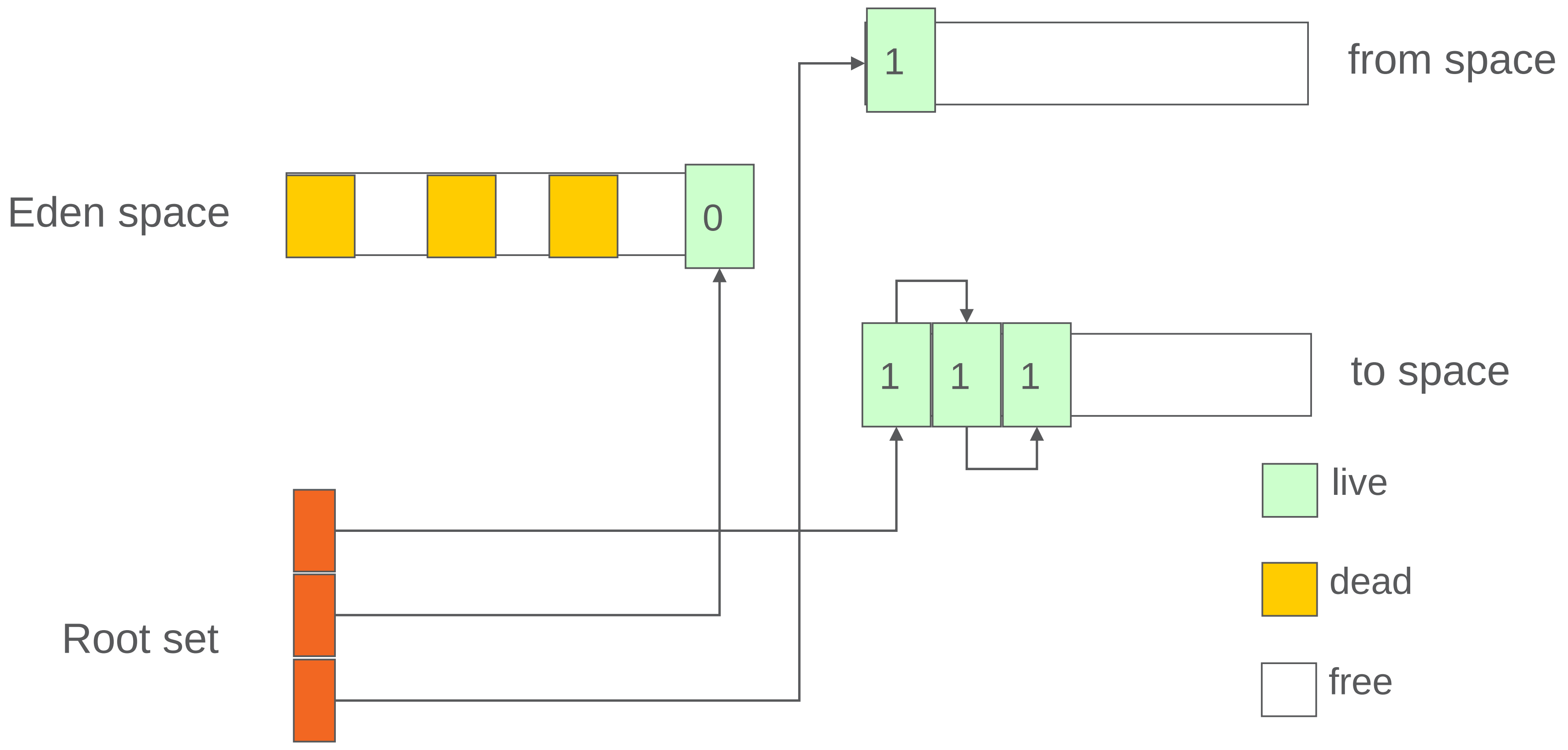
🟨 dead

⬜ free

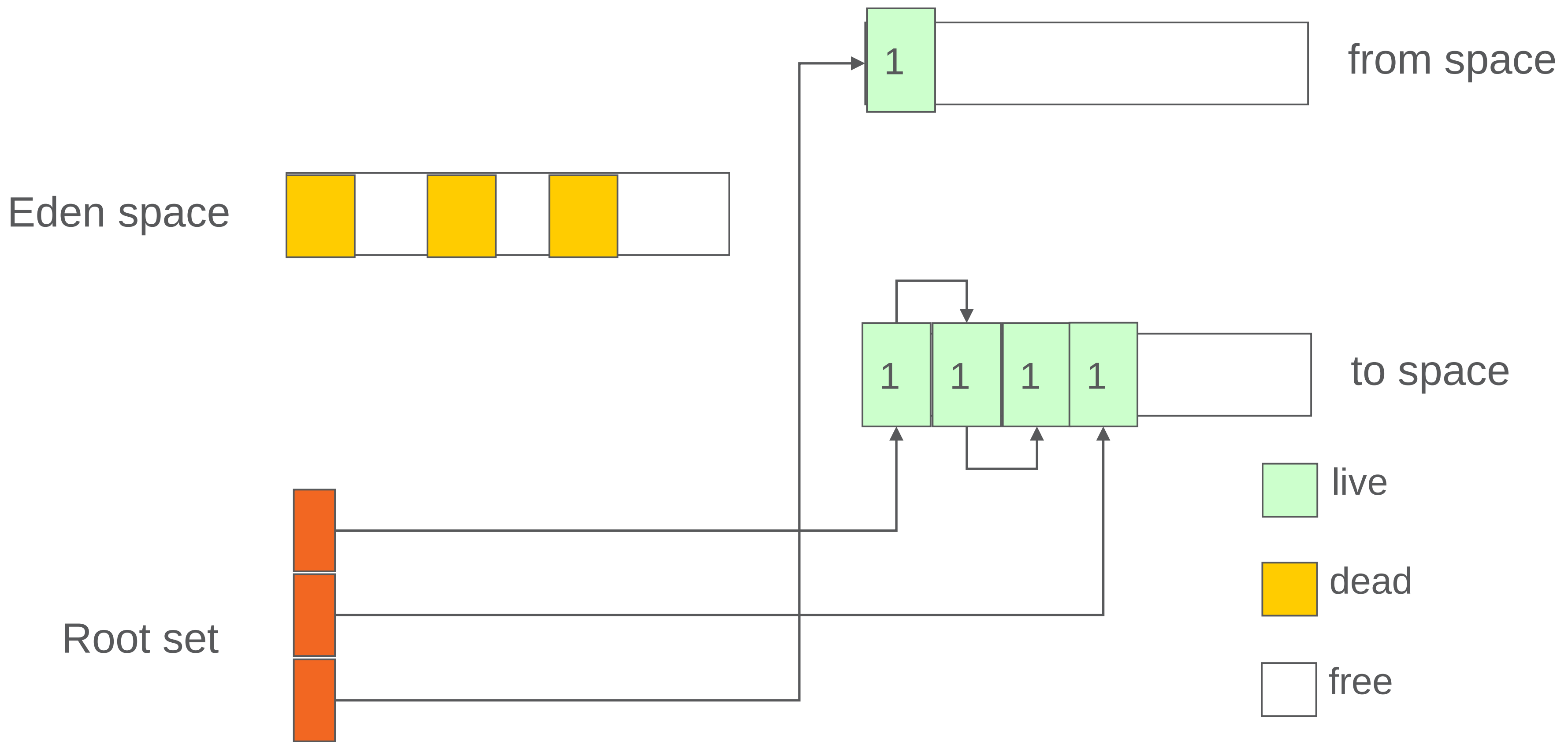
Young Generation Allocation



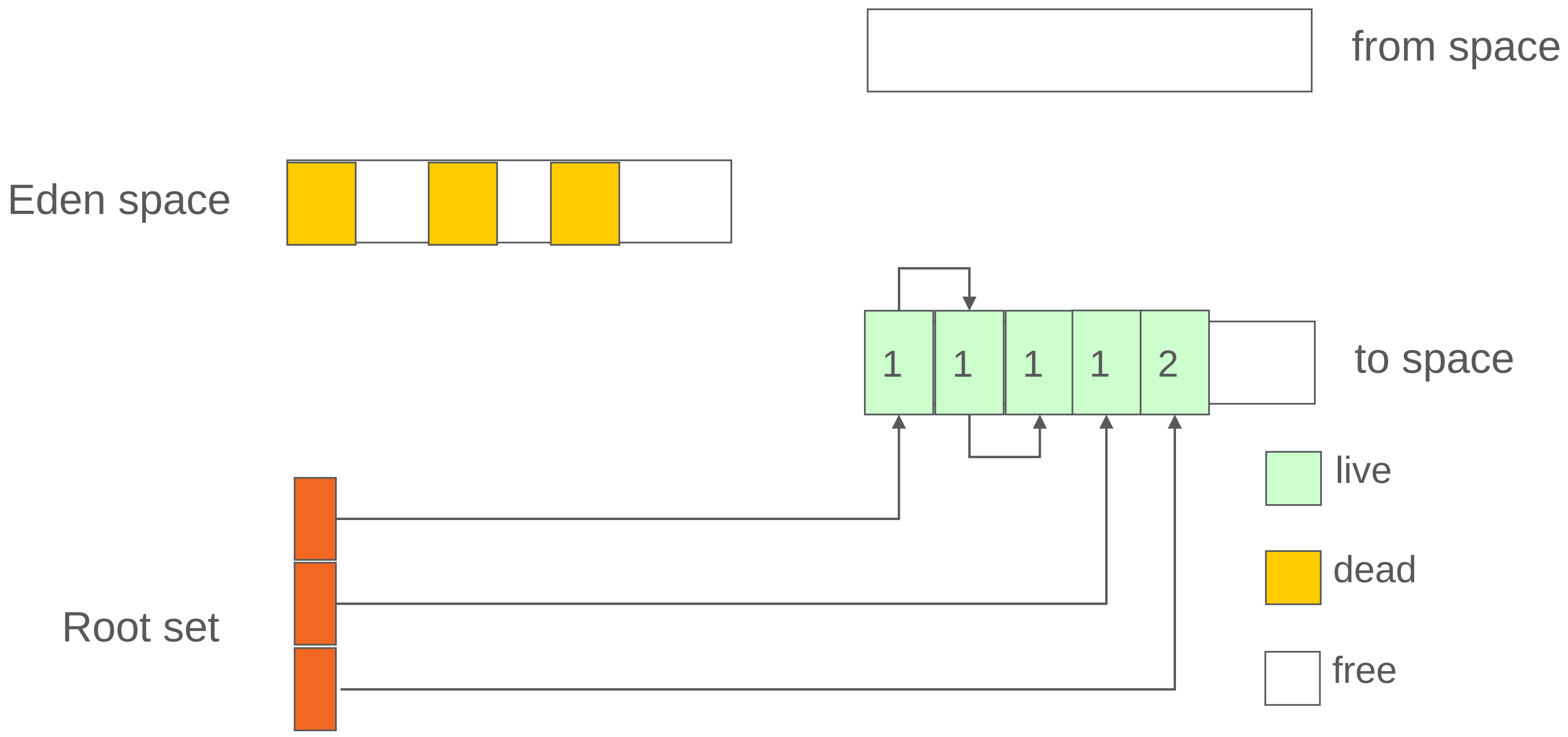
Young Generation Allocation



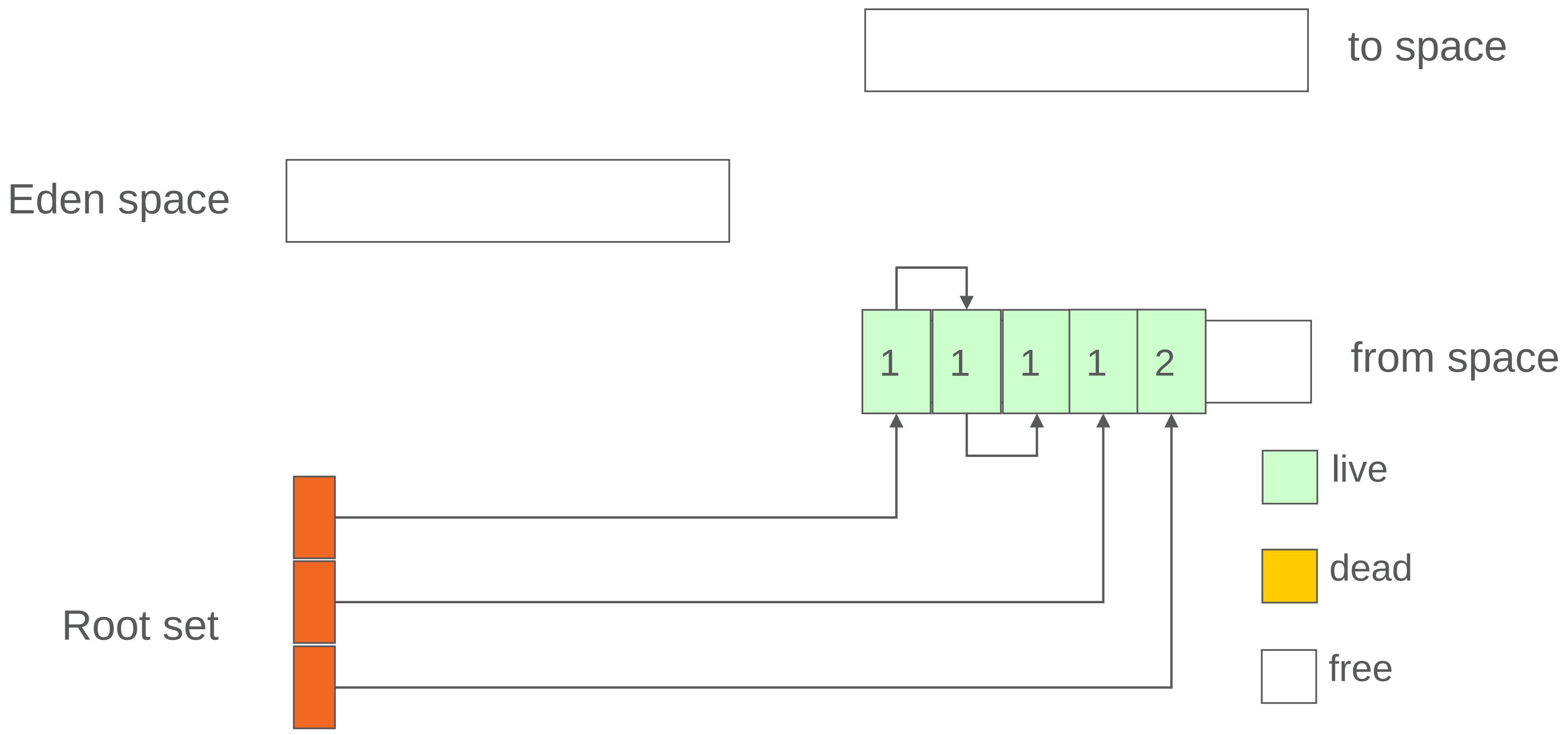
Young Generation Allocation



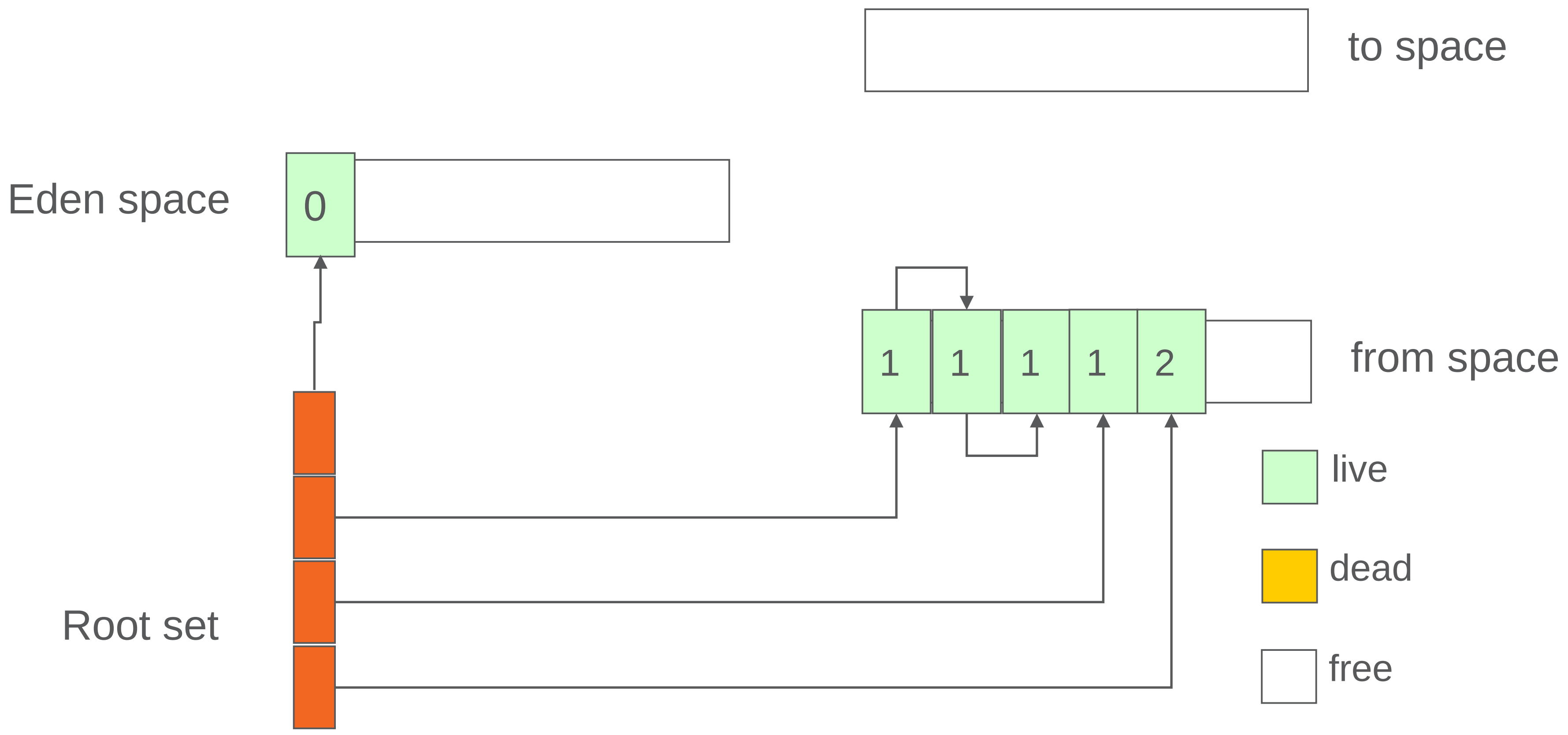
Young Generation Allocation



Young Generation Allocation



Young Generation Allocation



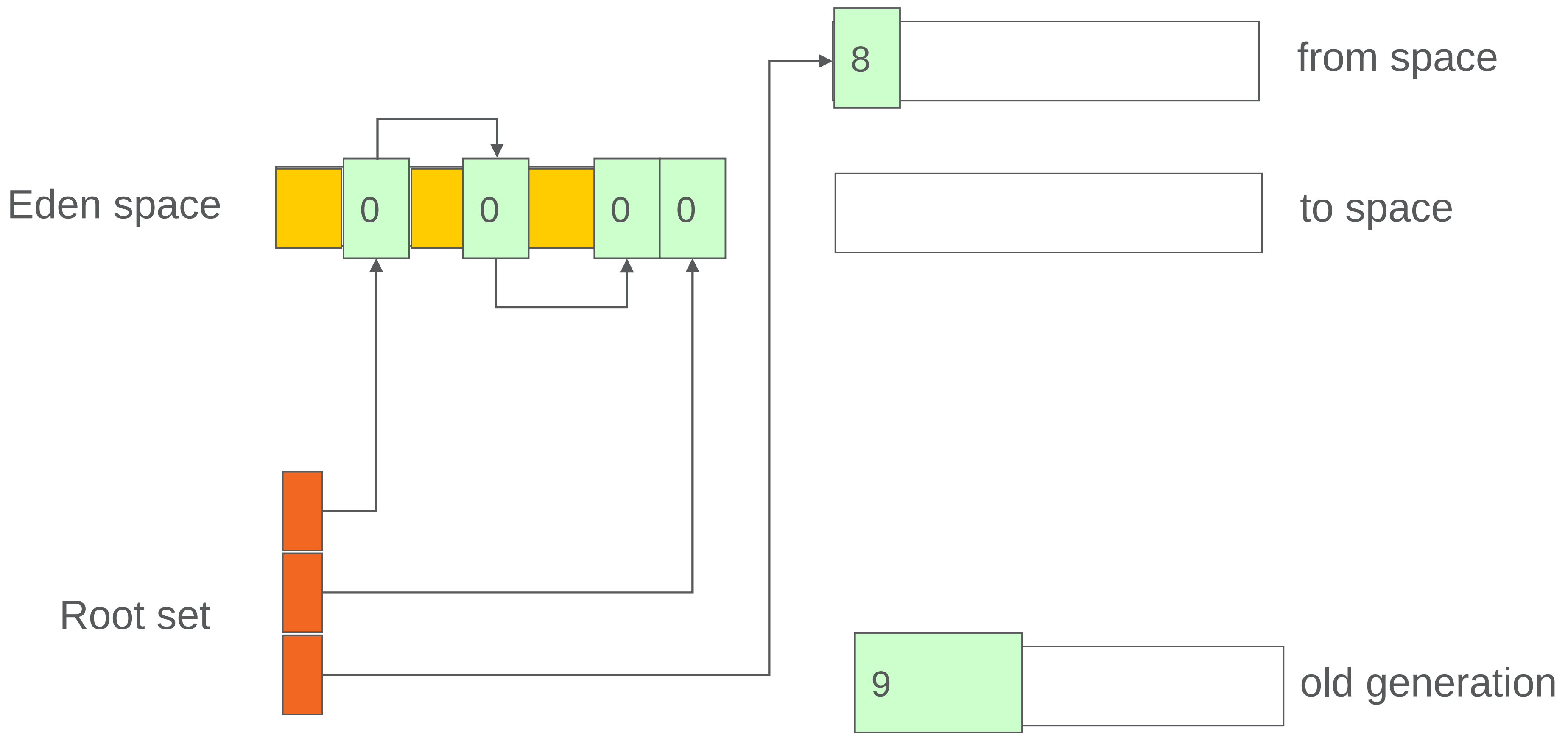
Major Garbage Collection

- Triggered when the tenured space is full
- Collects old and young generations
  - Although this is really a ‘Full GC’

Copying to Old Generation

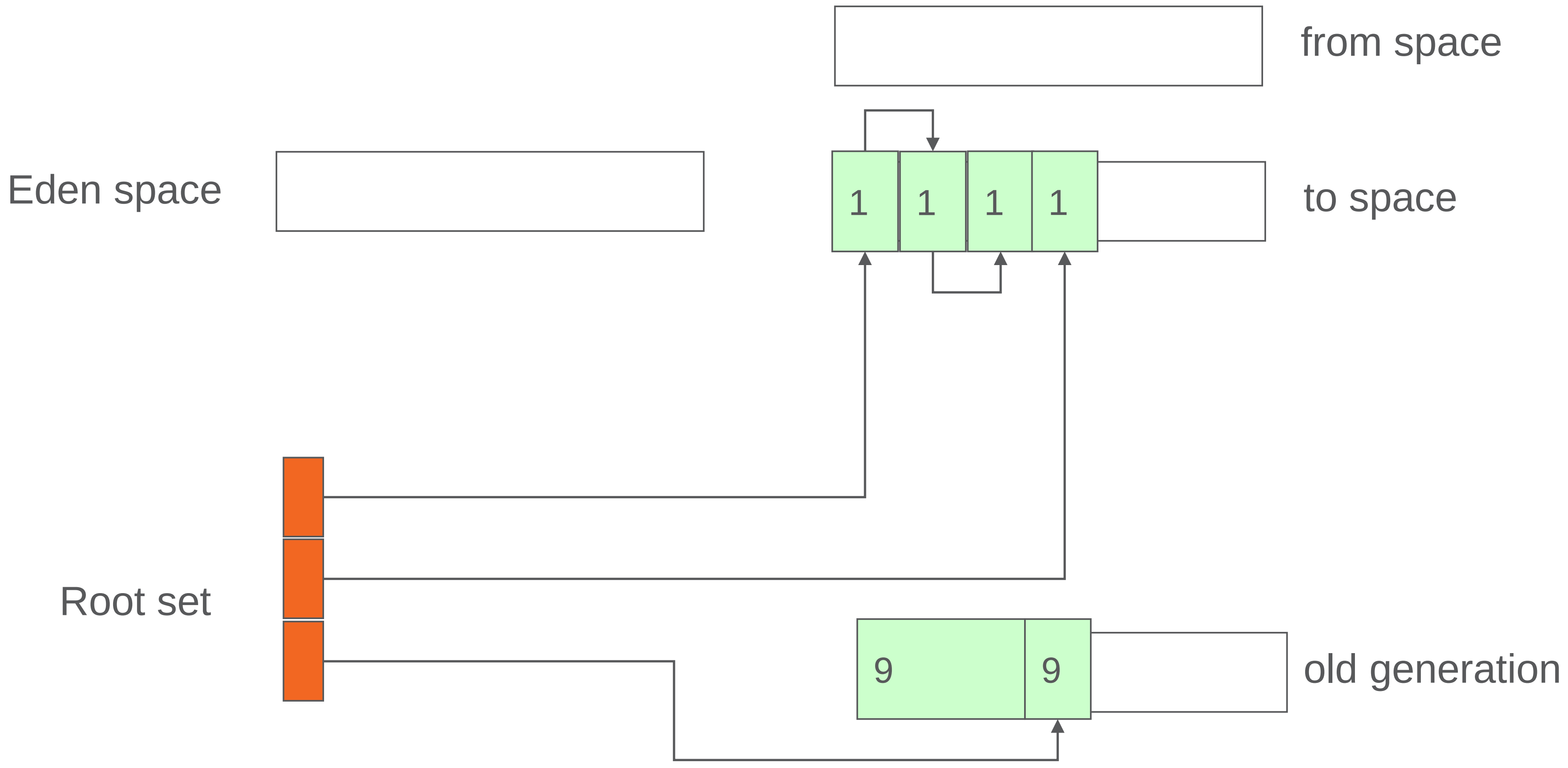
- JVM will eventually promote to old generation
  - After a certain number of garbage collects
  - If survivor space is full
  - If JVM has been told to always create objects in old space
    - -XX:+AlwaysTenure flag to JVM

Promotion



8
from space
to space
Eden space
0
0
0
0
Root set
9
old generation

Promotion

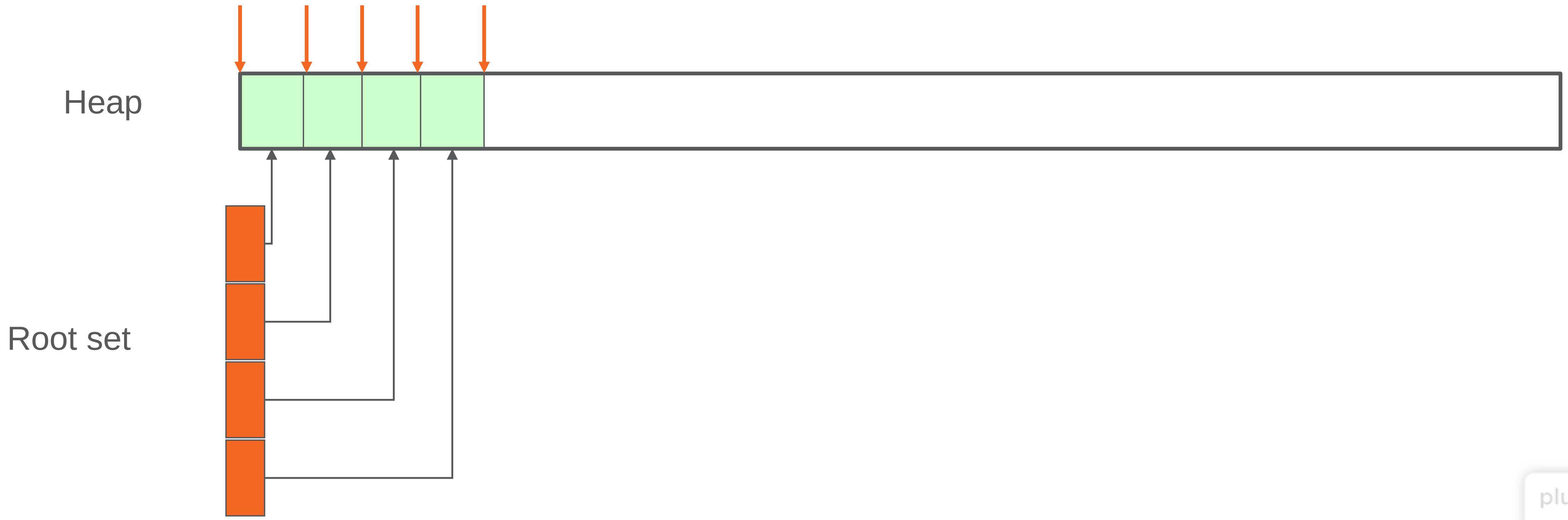

from space
Eden space
to space
1
1
1
1
Root set
9
9
old generation

Allocating Objects to Old Space

- Objects over a certain size will be allocated directly in old space
  - No JVM option to force objects to old space
- Option -XX:PretenureSizeThreshold=<n>
  - all objects larger than <n> bytes should be allocated directly in old space
  - However if object size fits TLAB, JVM will allocate it in TLAB
  - You should also limit TLAB size

Memory Allocation

- Want memory allocation to be as quick as possible
  - Can simply increment a pointer
  - Young always uses this, old may use it



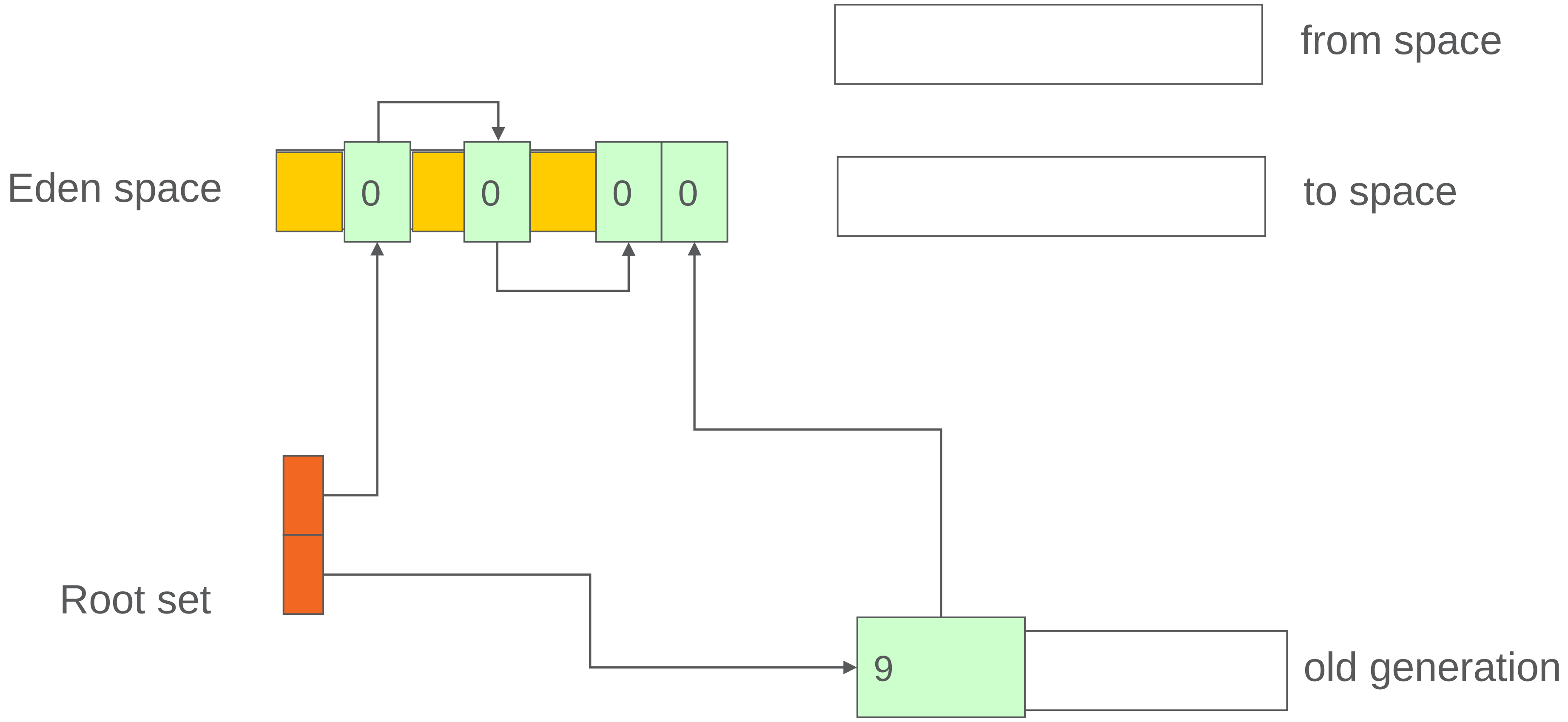
Allocation

- However, have to be aware of multi threading issues
  - Java use Thread Local Allocation Buffers (TLABs)
  - Each thread gets its own buffer in the Eden space
  - No locking required

What Does Live Mean?

- Live roots
  - From stack frames
  - Static variables
  - Others such as JNI and synchronization ‘monitors’
- References from live rooted objects are followed to other objects
- What about references from Old Generation to Young?

References From Old Generation



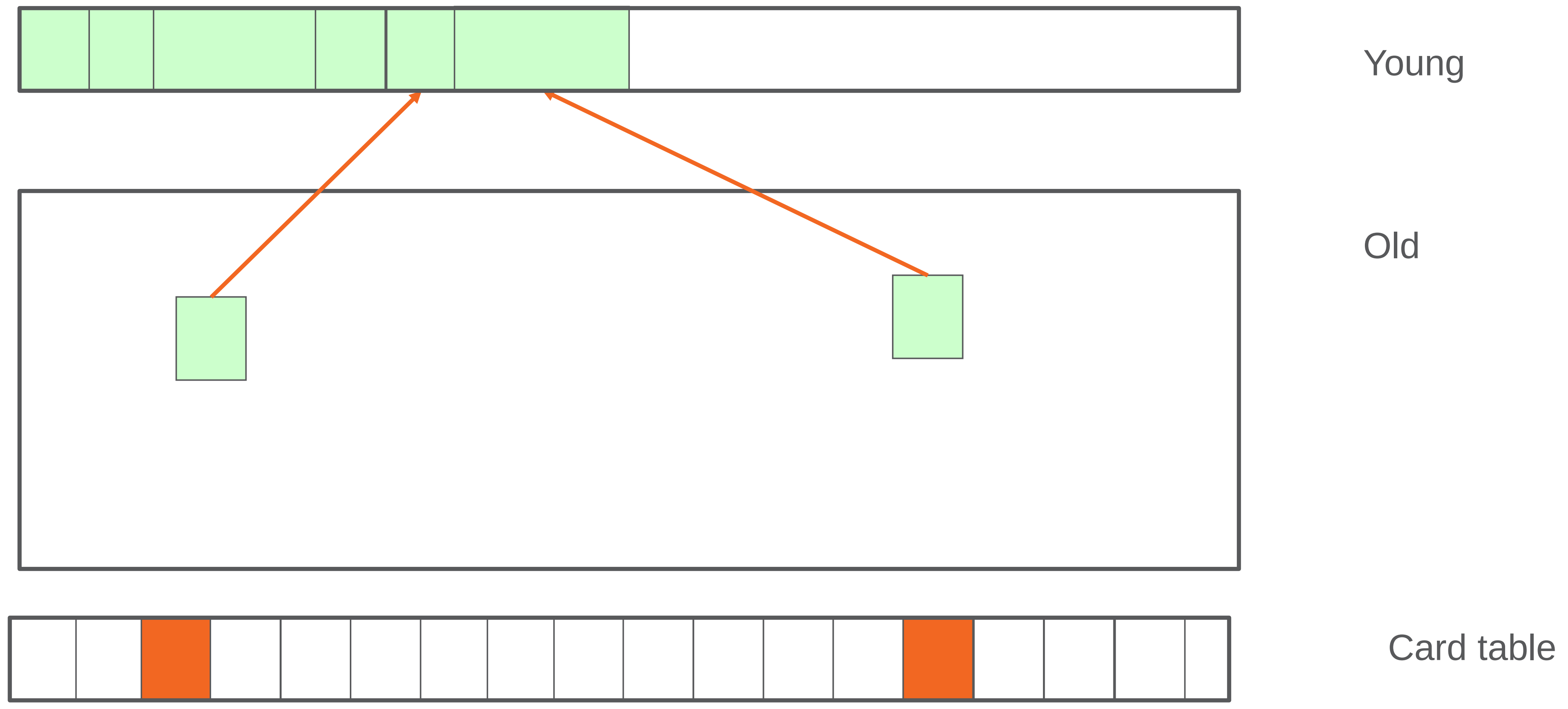
References From Old to New

- This is an issue - Young GC has to scan 'old' space
- Sort of defeats the purpose
- Enter 'card tables'

Card Table

- Each write to a reference to a young object goes through a write barrier
- This barrier updates a card table entry
- One entry per 512 bytes of memory
- Minor GC scans table looking for the areas that contain references
- Load that memory and follow the reference

Card Table



Different Garbage Collectors (i)

- Serial generational collector
  - -XX:+UseSerialGC
- Parallel for young space, serial for old space generational collector
  - -XX:+UseParallelGC
- Parallel young and old space generational collector
  - -XX:+UseParallelOldGC

Different Garbage Collectors (ii)

- Concurrent mark sweep with serial young space collector
  - -XX:+UseConcMarkSweepGC
  - -XX:-UseParNewGC
- Concurrent mark sweep with parallel young space collector
  - -XX:+UseConcMarkSweepGC
  - –XX:+UseParNewGC
- G1 garbage collector
  - -XX:+UseG1GC

Serial Collector

- Single threaded
- Mark and sweep
- OK for small applications running on the client

Parallel Collector

- Multiple threads for minor collection
- Single thread for major collection
- Same process as Serial
- Use on servers

Parallel Old Collector

- Multiple threads for minor and major collections
- Preferred over ParallelGC

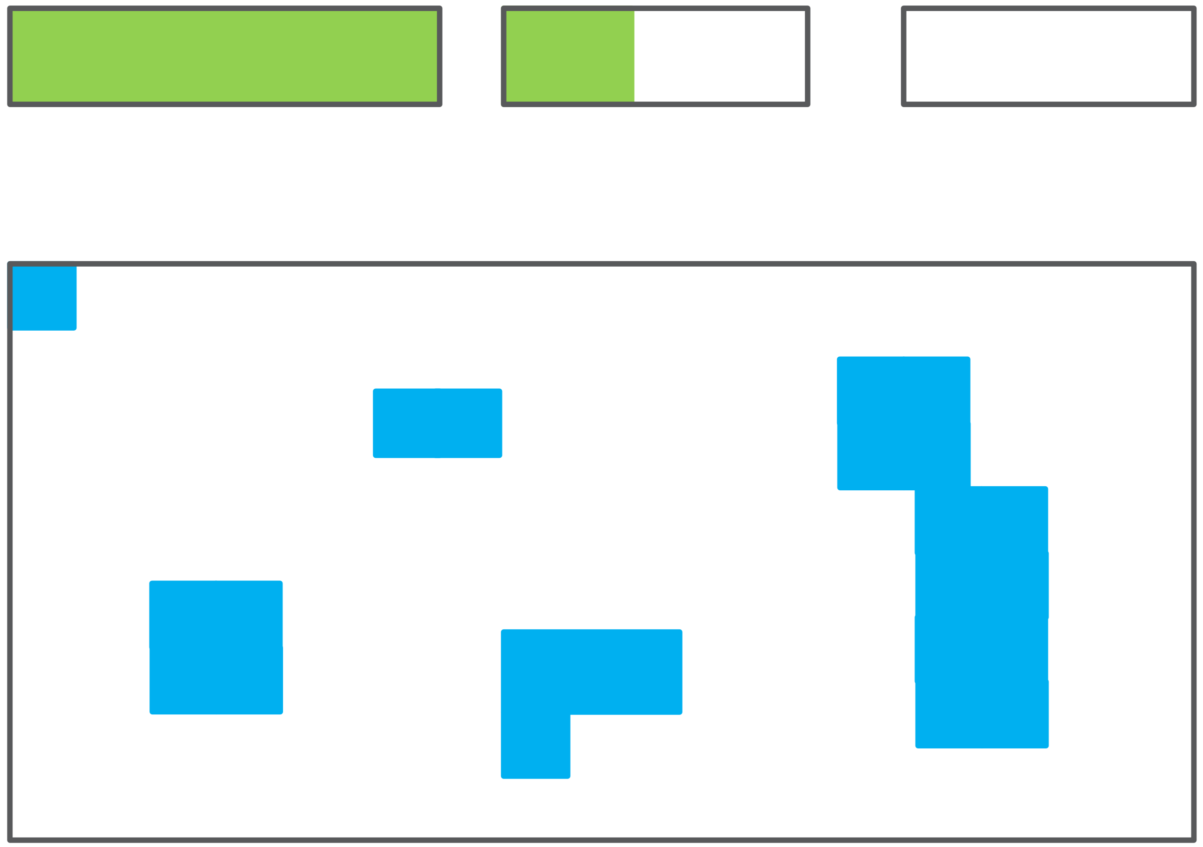
Concurrent Mark And Sweep

- Only collects old space
- No longer 'bump the pointer' allocation
- Causes heap fragmentation
- Designed to be lower latency

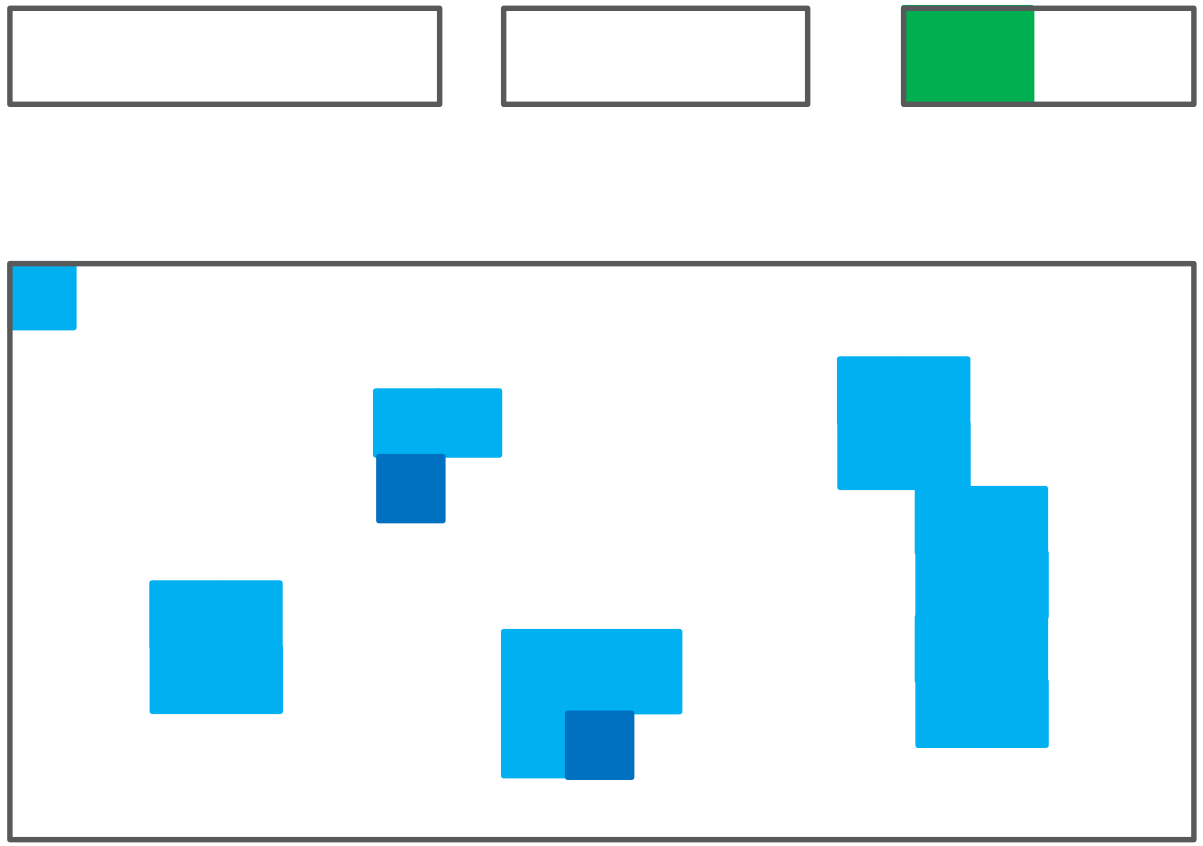
Concurrent Mark Sweep Details

Phase	Notes	Description
Initial Mark	Stop the world	Mark objects in the old generation reachable from root references
Concurrent Mark	Concurrent	Traverse object graph looking for live objects Any allocations made during this phase are automatically marked as live
Remark	Stop the world	Finds objects created after the previous phase stopped
Concurrent Sweep	Concurrent	Collects objects
Resetting	Concurrent	Get ready for the next run

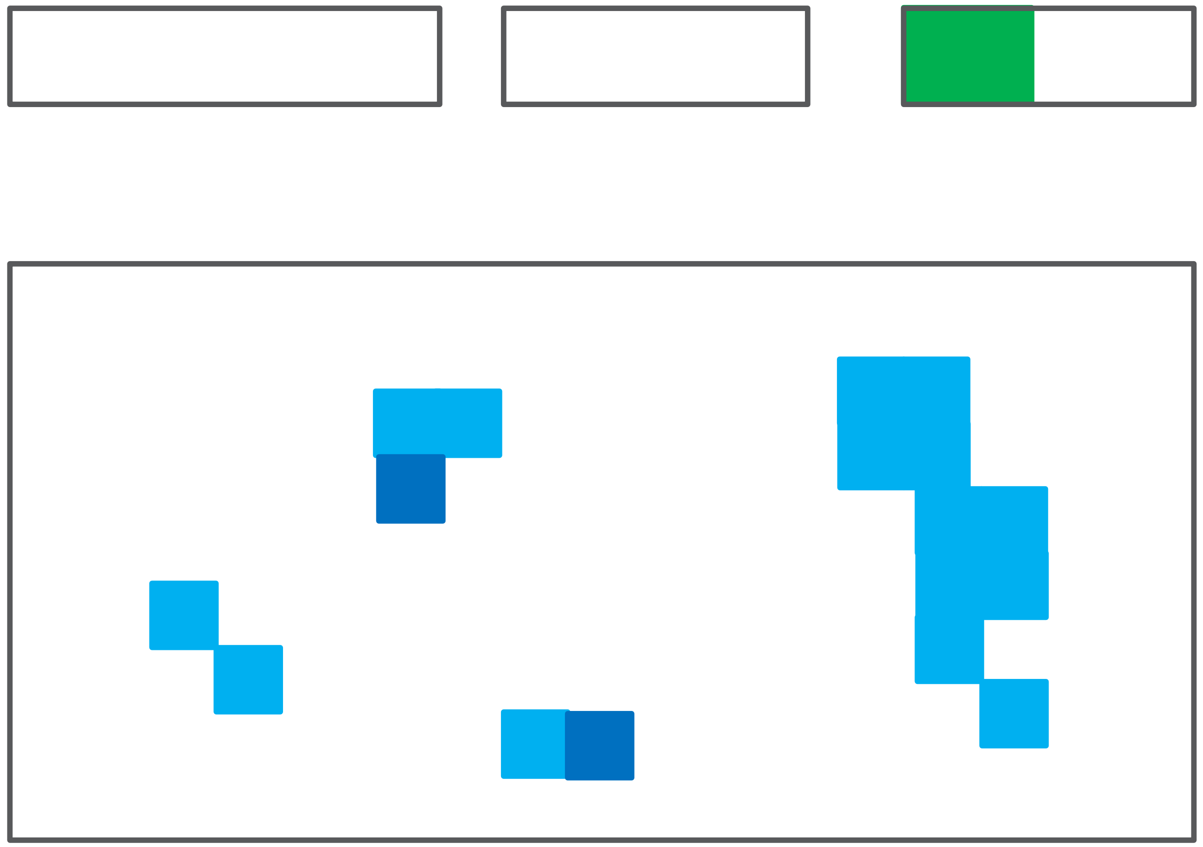
CMS GC Steps



CMS GC Steps



CMS GC Steps



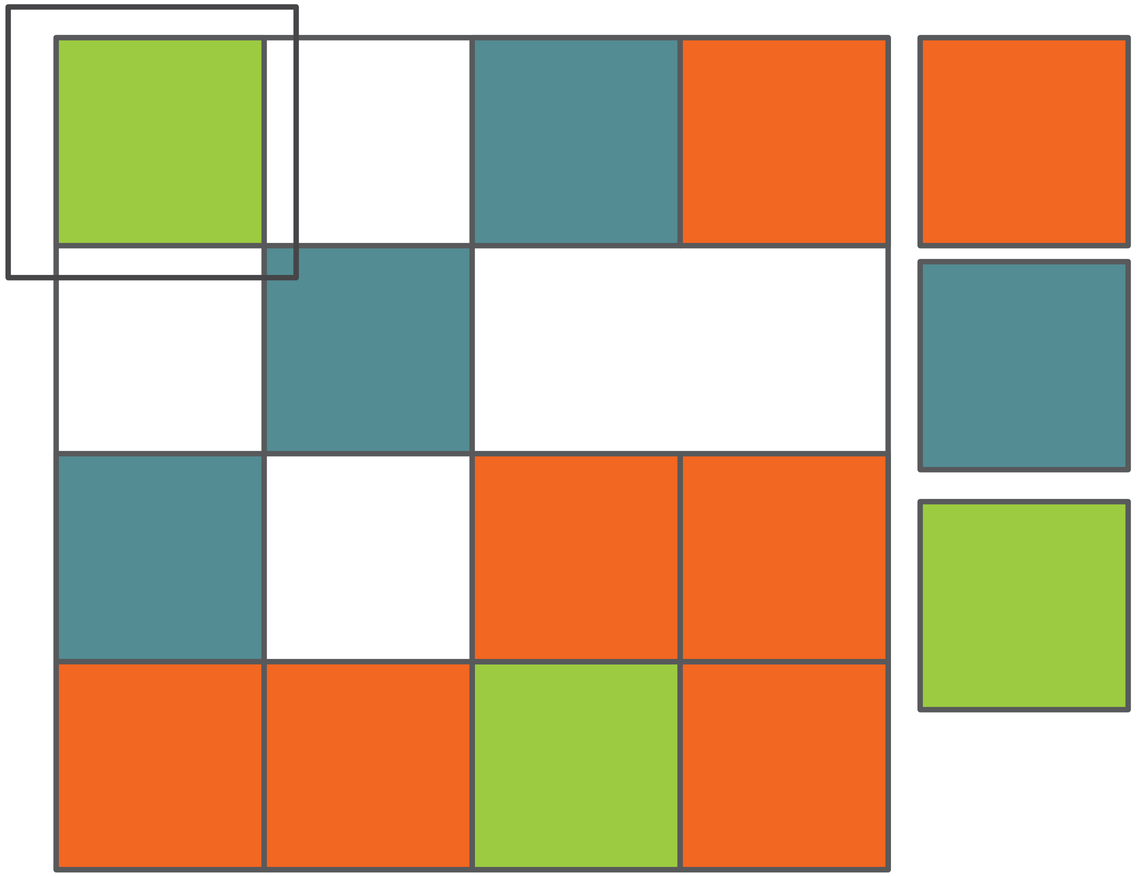
G1 Collector

- New in Java 6
  - Officially supported in Java 7
- Is a compacting collector
- Planned as a replacement for CMS

G1 Collector

- Meant for server applications
  - Running on multiprocessor machines with large memories
- Breaks heap into regions
  - Still has concept of Eden, Survivor and Tenured spaces

G1 Collector Memory Layout

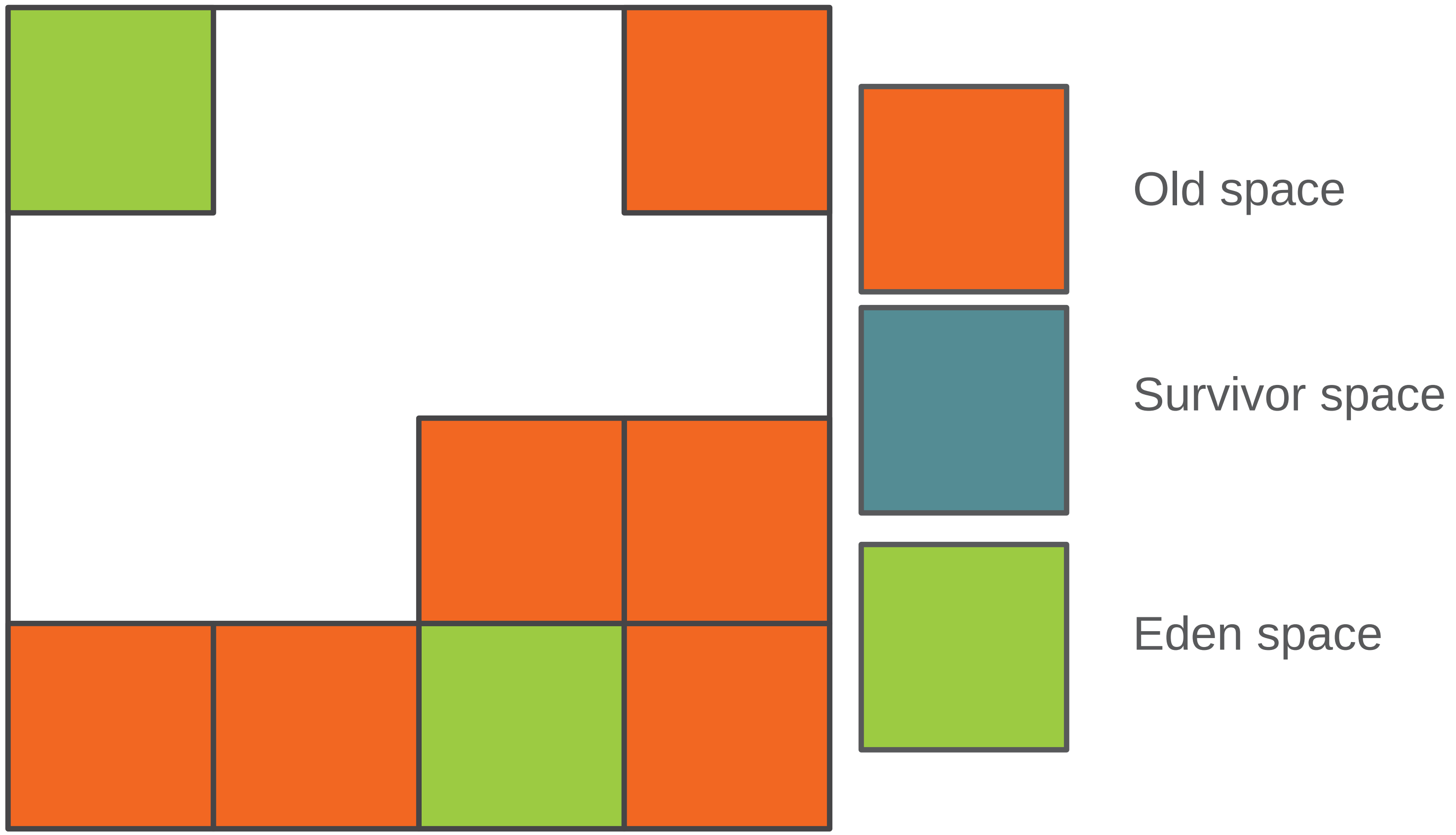


G1 Collector

- Objects are 'evacuated'
  - Moved/Copied between regions

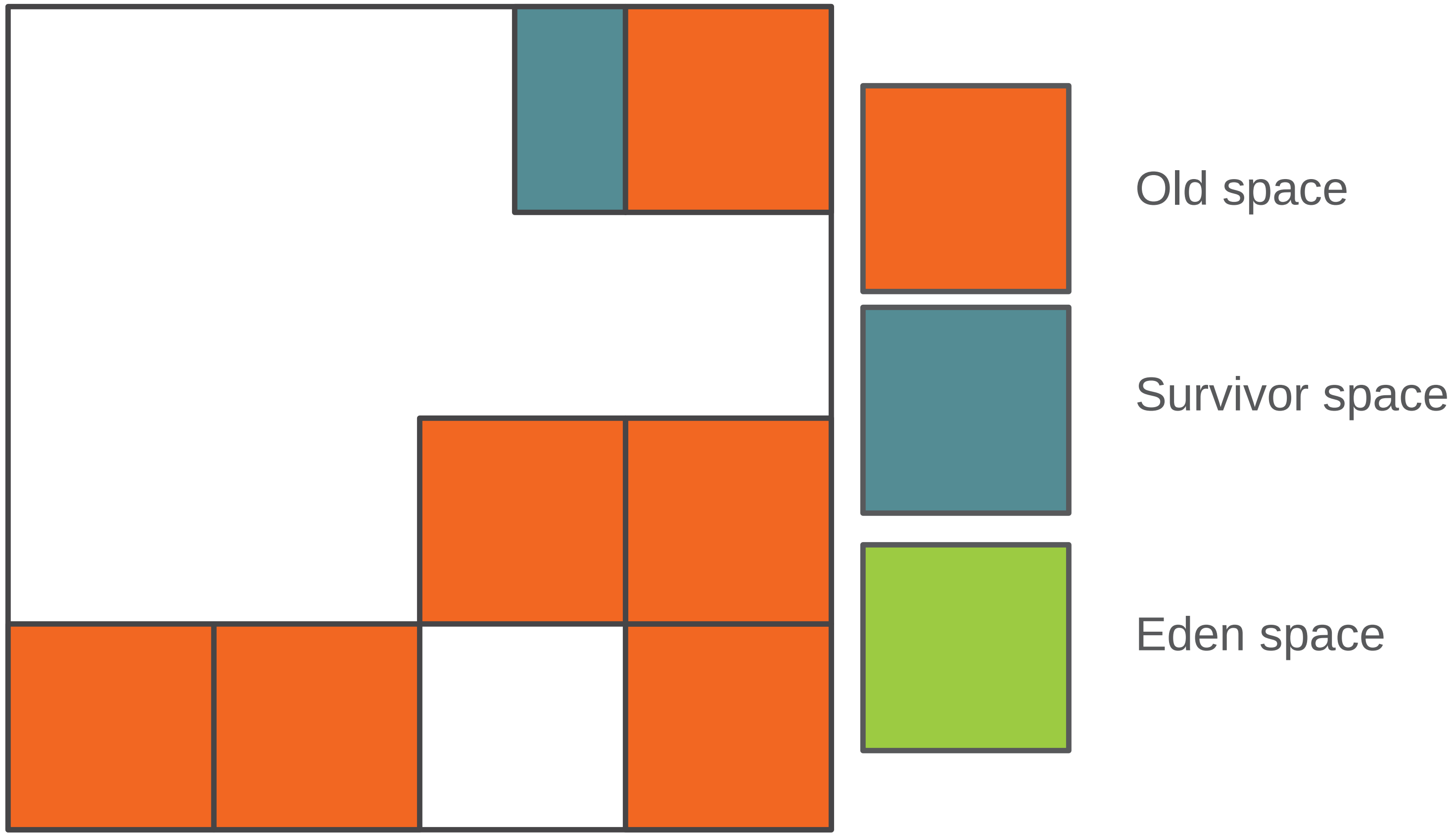
G1 Young GC

- Young objects moved to one or more survivor regions



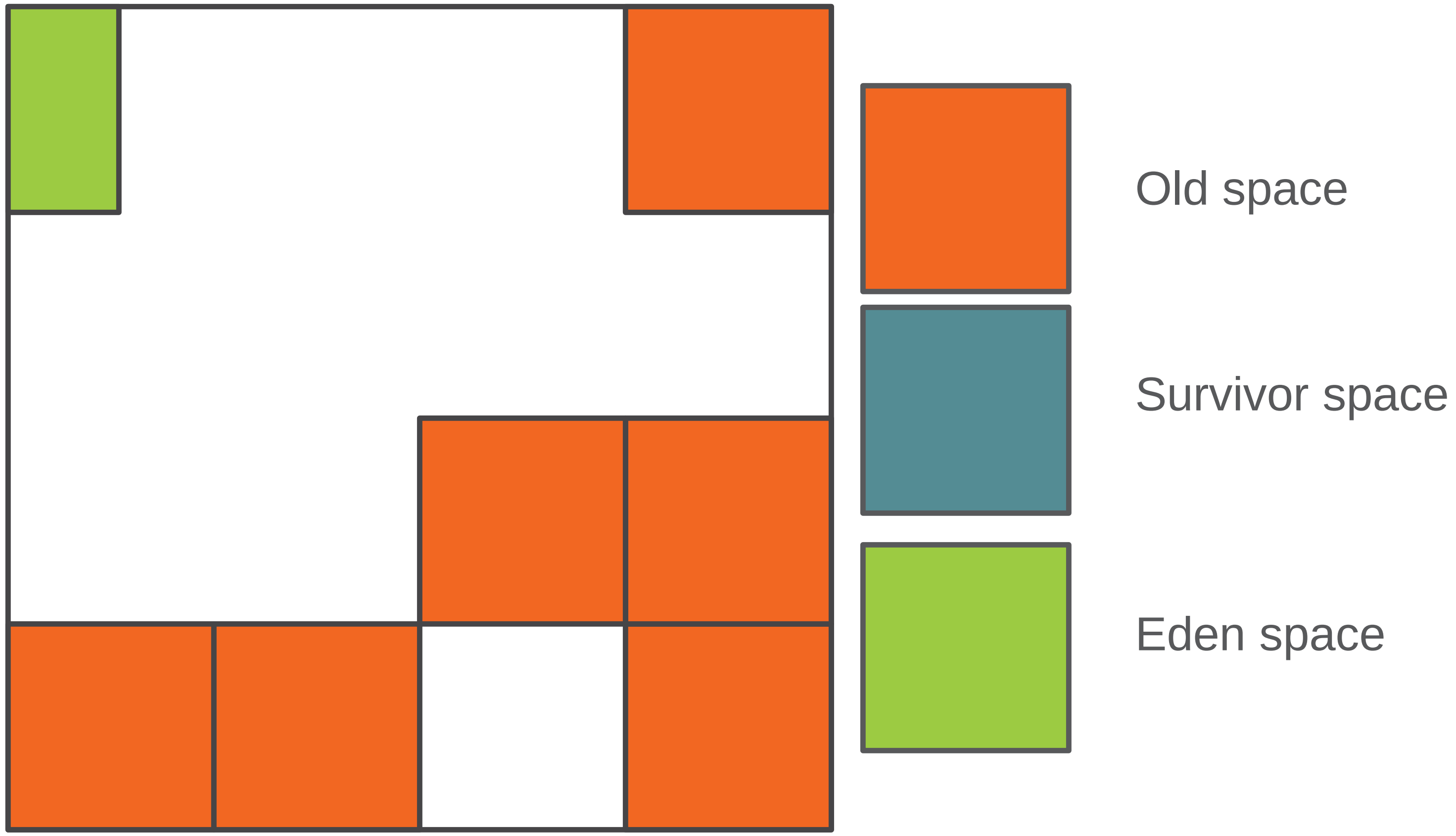
G1 Young GC

- Young objects moved to one or more survivor regions



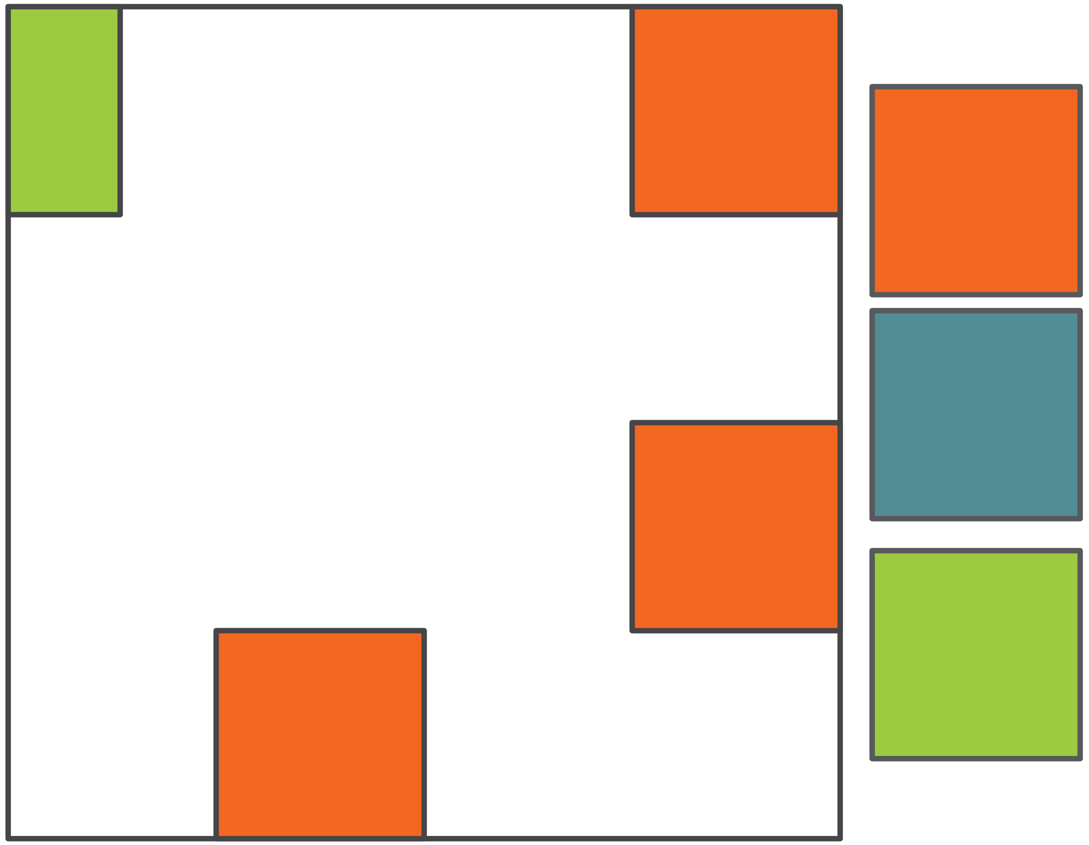
G1 Old GC

- Initial mark phase done at same time as Young GC



G1 Old GC

- After GC



Which Collector

- No easy answer to this question
- Java offers a mixture of garbage collectors
  - From serial
  - To G1
- Picking a collector is not a simple job
- Profile the application under as close to production load as possible
- Test under the different garbage collectors

References

- http://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/cms.html
- http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/G1GettingStarted/index.html