

# Jvm (gc) tuning 101

Ferid Sabanovic, IBS JavaSolutions  
[jsolutions.se](http://jsolutions.se)

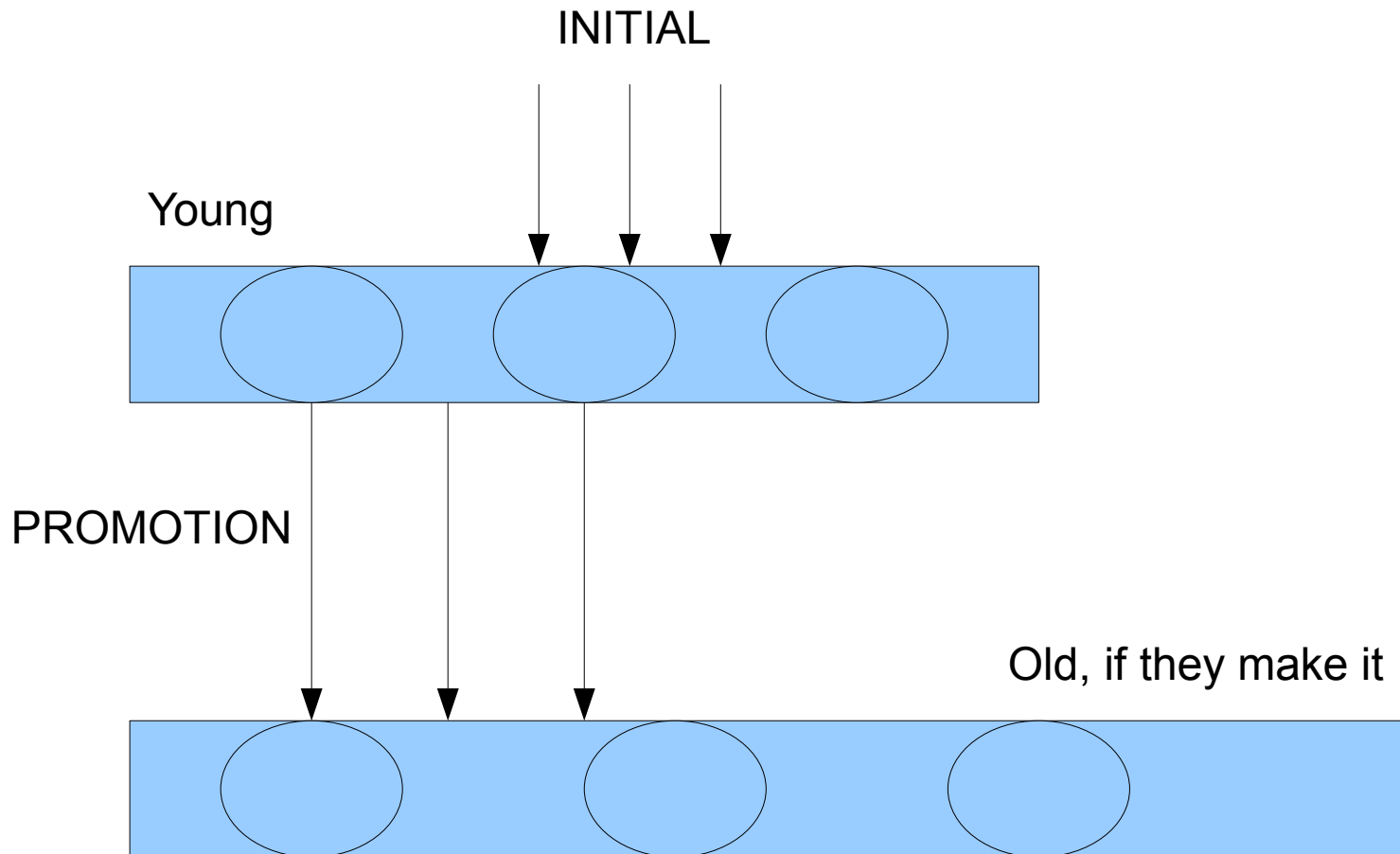
# Jvm (gc) tuning 101

- Introduction
- Test, “problems”
- Tuning

# jvm (gc) tuning 101

- The garbage collector
  - Young
  - Old
- Different GC algorithms
- People

# jvm (gc) tuning 101



# jvm (gc) tuning 101

- Most important thing
  - About 80 - 98% die very young, few million instructions

# jvm (gc) tuning 101

- Discover the problem?
  - Look at your GC behavior
  - How often does it sweep?
  - To often in young? (I bet it is)

# jvm (gc) tuning 101

- Testing
  - java is not C/C++ etc.
  - Get real tests, not fake ones, and do not “DOS attack” your JVM
  - The “java” you use is probably not “java real time”

# jvm (gc) tuning 101

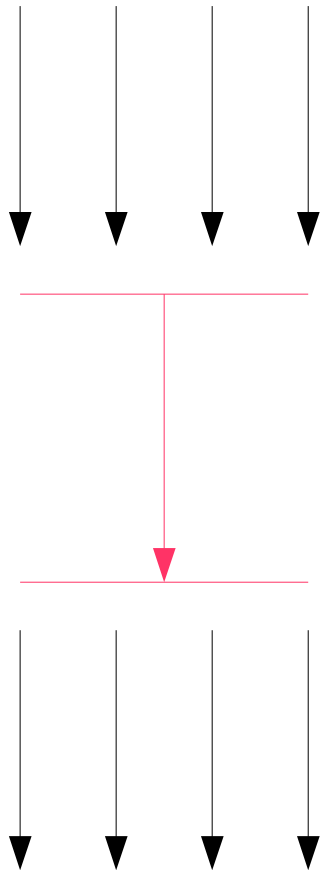
- There are about 500+ different jvm options to tell it what to do
- Most people think “default” (or only -Xmx)
- Lots of cores in the machines these days

# jvm (gc) tuning 101

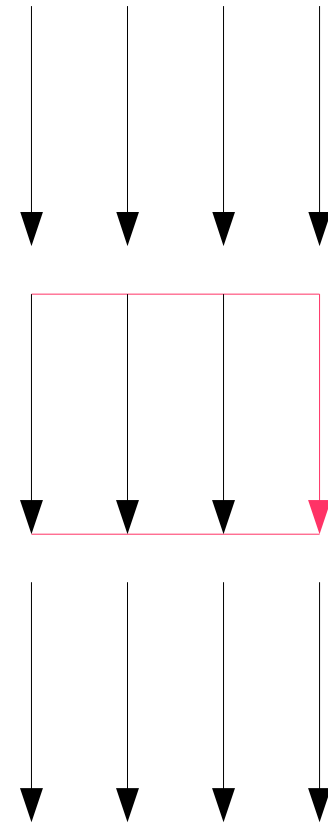
- Concurrent mark sweep
  - Some of us always want an answer

Use the CMS collector if your application needs shorter garbage collection pauses and can afford to share processor resources with the garbage collector when the application is running

# jvm (gc) tuning 101



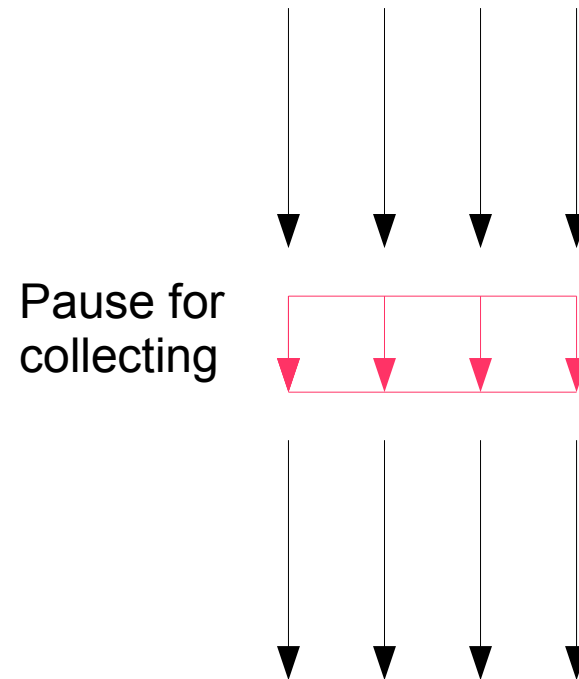
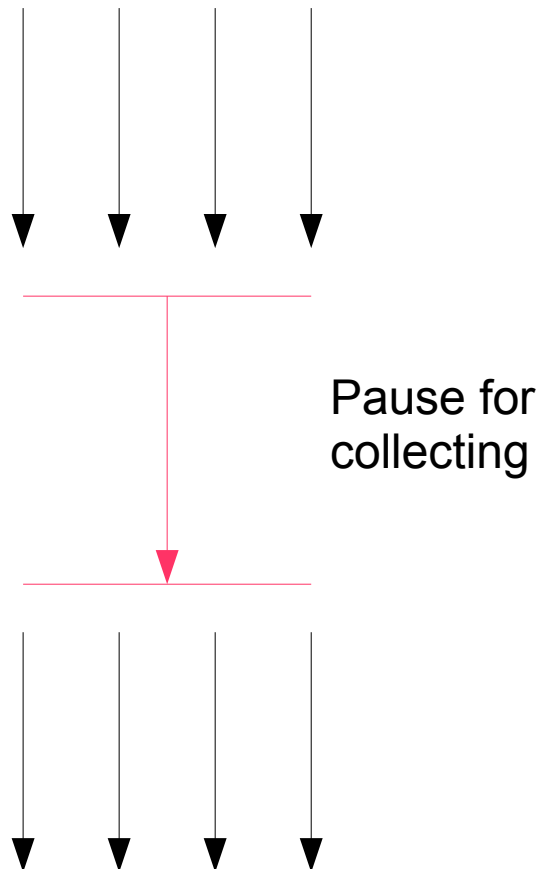
Sweeping with an answer



# jvm (gc) tuning 101

- Parallel
  - More cores to use for collecting garbage
  - Why use serial?

# jvm (gc) tuning 101



# jvm (gc) tuning 101

- So how do we do this?

# Jvm (gc) tuning 101

- The young generation
  - Most probably you need a larger one
  - `-Xmn=2048m` (fixed size)
  - `-XX:MaxNewSize=2048m`
  - `-XX:NewRatio=2`

# jvm (gc) tuning 101

- **ConcMarkSweep**
  - Dont stop the world
- `-XX:+UseConcMarkSweepGC`

# jvm (gc) tuning 101

- Parallel collecting the young ones
  - Put all cores in work for the younger generation
- `-XX:+UseParNewGC`
  - Combination with `ConcMarkSweep`
- `-XX:+UseParallelGC`

# jvm (gc) tuning 101

- For the old ones left behind
  - Old generation parallel collecting,
    - java se 5 update 6, default
    - Jdk 1.4.1 available
- `-XX:+UseParalellGC`
  - No `ConcMarkSweep`, add it yourself

# jvm (gc) tuning 101

- More on the parallel old collector
  - Set the amount of threads to be used
- `-XX:+ParallelGCThreads=8`
  - Number of CPUs

# jvm (gc) tuning 101

- Multiple threads wants the young ones at the same time

For multithreaded applications, allocation operations need to be multithread-safe. If global locks were used to ensure this, then allocation into a generation would become a bottleneck and degrade performance. Instead, the HotSpot JVM has adopted a technique called Thread-Local Allocation Buffers(TLABs)

a region of Eden that is used for allocation by a single thread

- `-XX:+UseTLAB`

# jvm (gc) tuning 101

- Other fun stuff
- `-XX:+CompileThreshold=n`
  - 5000 default
- `-XX:+DontCompileHugeMethods`

# Jvm (gc) tuning 101

- Yes, this can also tune your JVM
  - ;)
- `-XX:+DisableExplicitGC`

	Default Collectors	Low Pause Collectors		Throughput Collectors	Heap Sizes
Generation		1 CPU	2+ CPUs	2+ CPUs and 1GB+ Heapsize	
Young	Copying Collector <i>default</i>	Incremental Train Collector  -Xincgc  (causes appx. 10% slower overall GC)	Parallel Copying Collector  -XX: +UseParNewGC	Parallel Scavenge Collector  -XX: +UseParallelGC -XX: +UseAdaptiveSizePolicy or -XX: +AggressiveHeap	-XX:NewSize, -XX:NewRatio, -XX:SurvivorRatio or -Xmn (fixed size)
Old	Mark-Compact Collector <i>default</i>		Concurrent Collector  -XX: +UseConcMarkSweepGC	Mark-Compact Collector <i>default</i>	-Xms, -Xmx
Permanent	Can be turned off with -Xnoclassgc. Use with care!				-XX:PermSize, -XX:MaxPermSize

# jvm (gc) tuning 101

<http://www.folgmann.com/en/j2ee/gc.html>

[http://java.sun.com/performance/reference/whitepapers/6\\_performance.html](http://java.sun.com/performance/reference/whitepapers/6_performance.html)

<http://java.sun.com/javase/technologies/hotspot/gc/index.jsp>

[http://java.sun.com/javase/technologies/hotspot/gc/memorymanagement\\_whitepaper.pdf](http://java.sun.com/javase/technologies/hotspot/gc/memorymanagement_whitepaper.pdf)