GC-Tuning

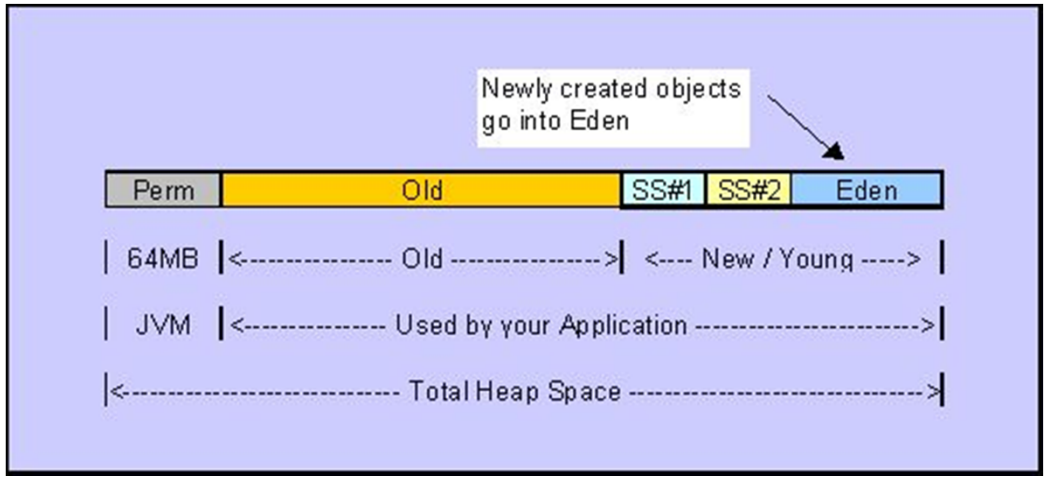
Erfahrungsbericht

Themen

- Grundlagen
  - Java Memory Management
  - GC-Algorithmen
- GC-Tuning
  - Erfahrungsbericht

Java Memory Management

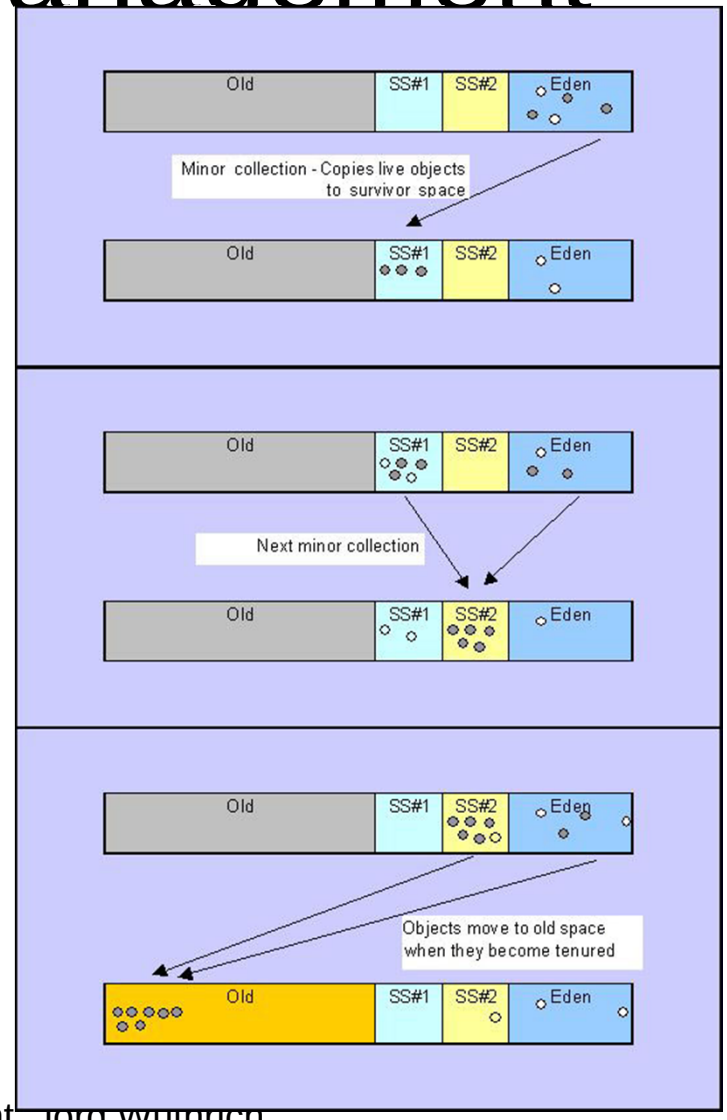
- Aufteilung des Speichers



- Begrifflichkeit abhängig von Hersteller
  - new = young = nursery | old = tenured

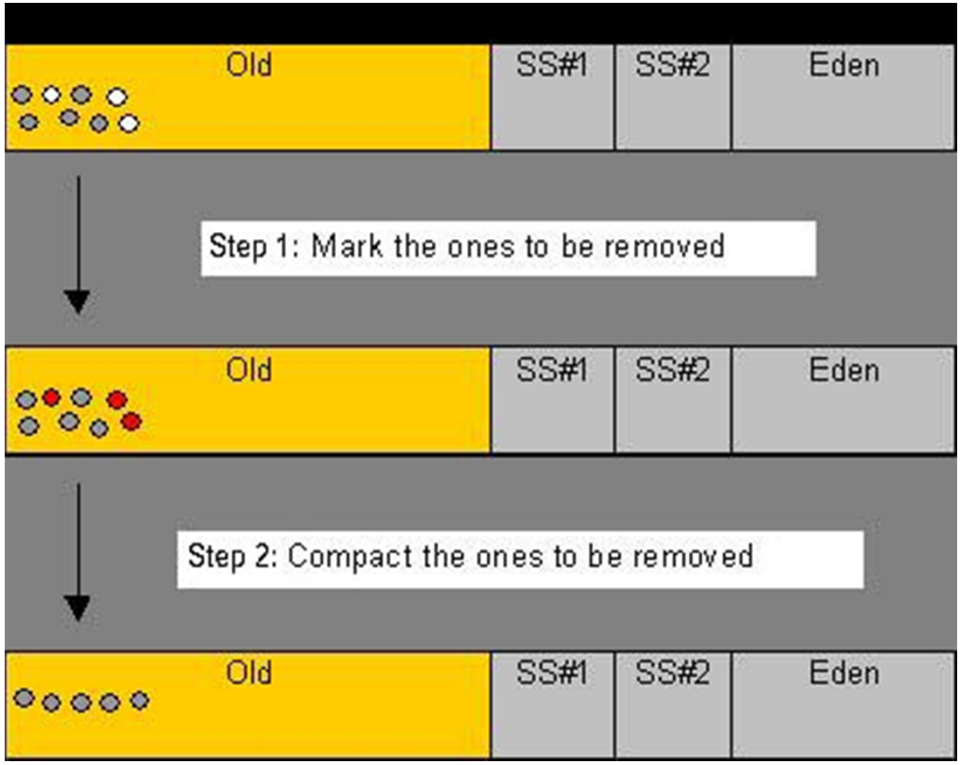
Java Memory Management

- minor collections
  - räumen im young space auf
  - Durchführung, wenn Speicher im young space knapp
  - Verschiebung in den old space



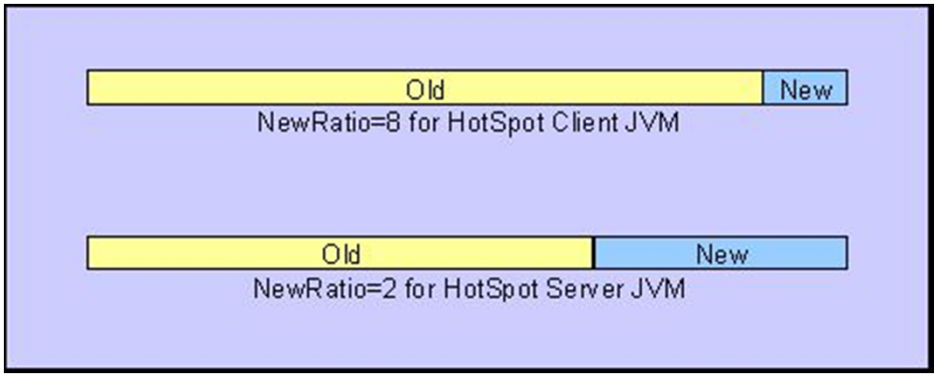
Java Memory Management

- Major collections
  - räumen im old space auf
  - werden durchgeführt, wenn old space knapp wird

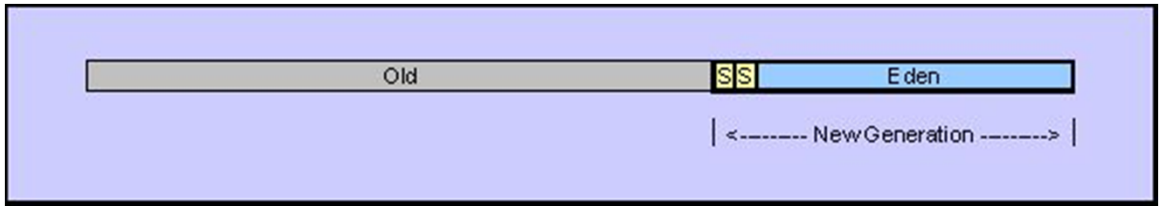


Java Memory Management

- Konfiguration (Beispiele)
  - newRatio



  - survivorRatio

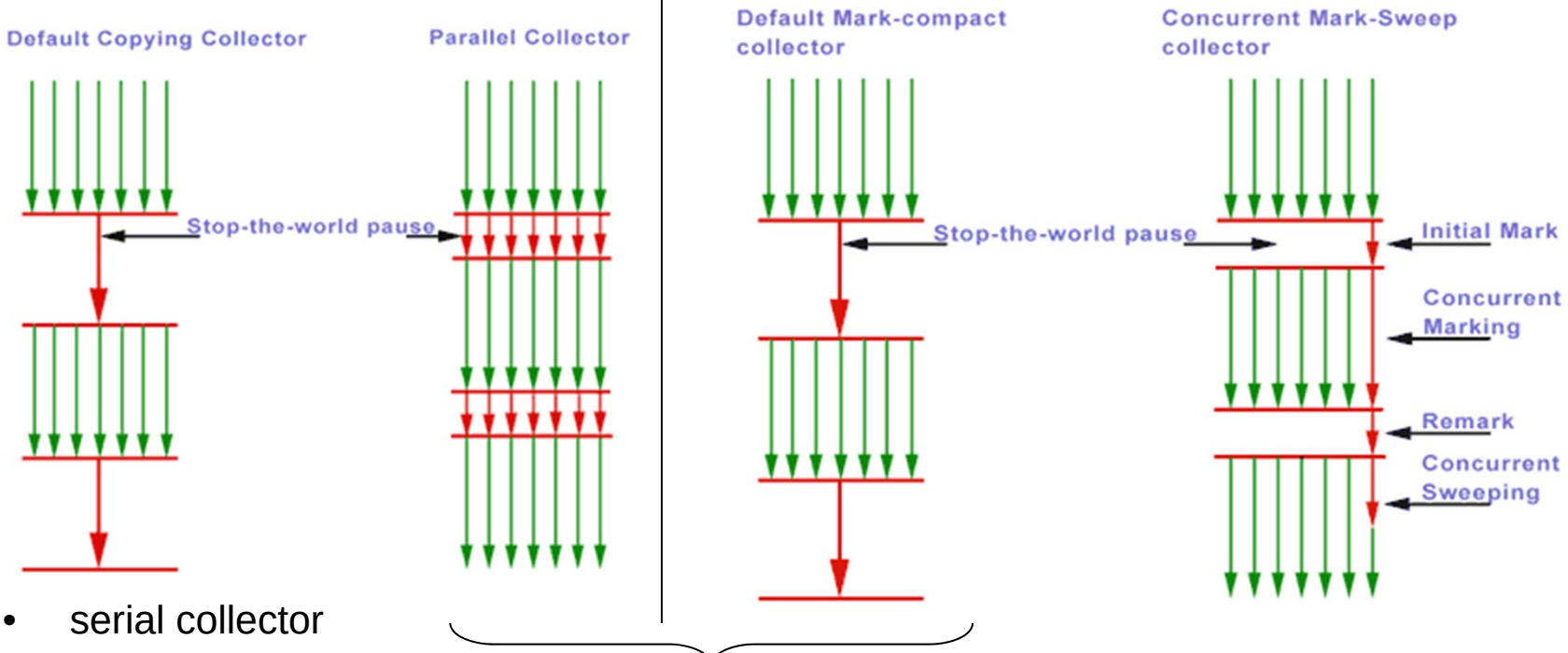


GC-Algorithmen

- Serial Collector
  - der älteste; alles seriell
- Parallel Collector
  - collection wird parallelisiert -> kürzere Collections
- Parallel Compacting Collector
  - zusätzlich Defragmentierung des Speichers
- Concurrent Mark+Sweep Collector (CMS)
  - Collection erfolgt gleichzeitig mit Applikations-Threads
- Generation First Collector (G1)
  - optimiert, heutige Multiprozessor-Umgebungen auszunützen

GC-Algorithmen

- serial <-> parallel
- mark+compact <-> concurrent

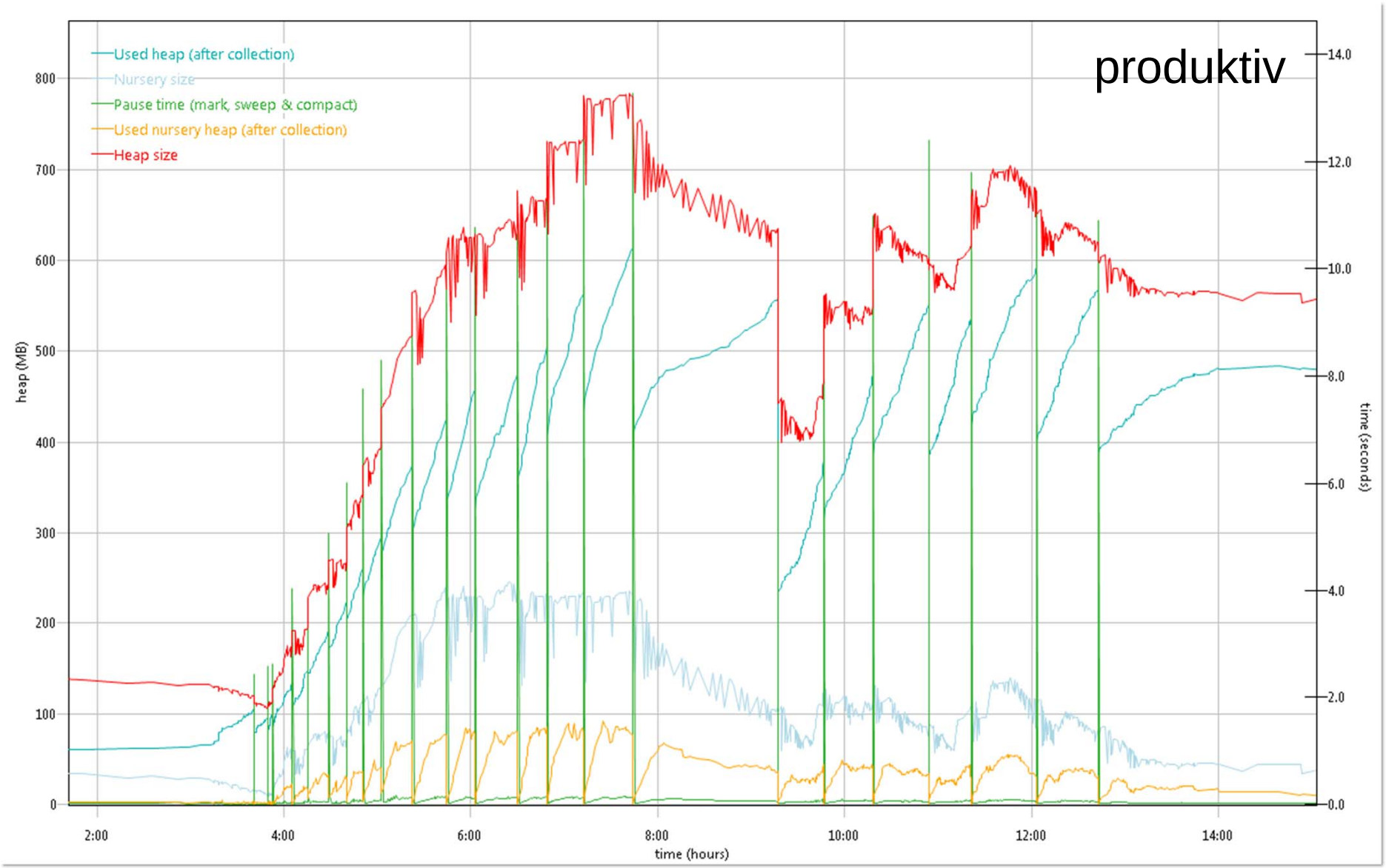


- serial collector

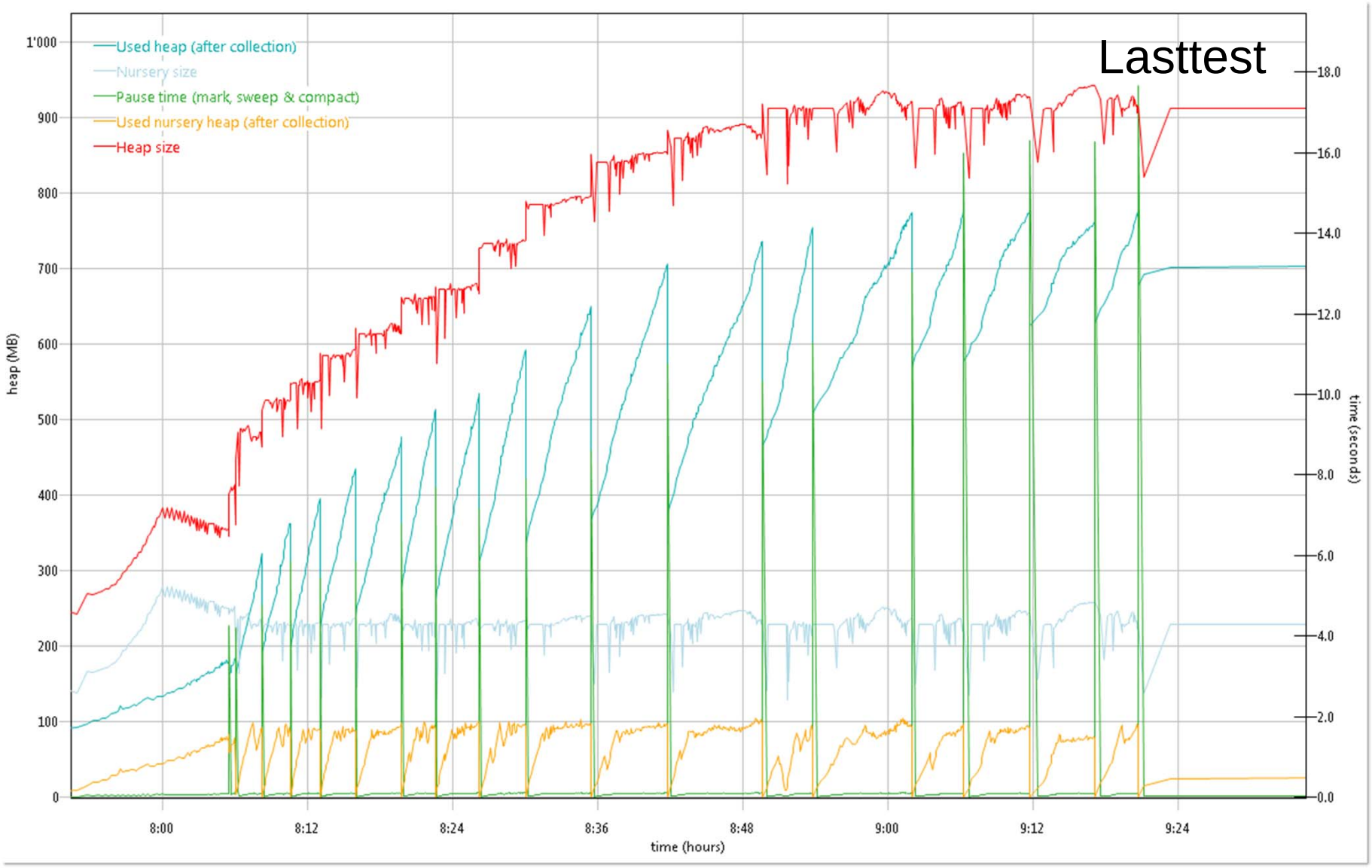
parallel collector

concurrent mark + sweep collector (CMS)

Applikation vor Tuning / Umbau

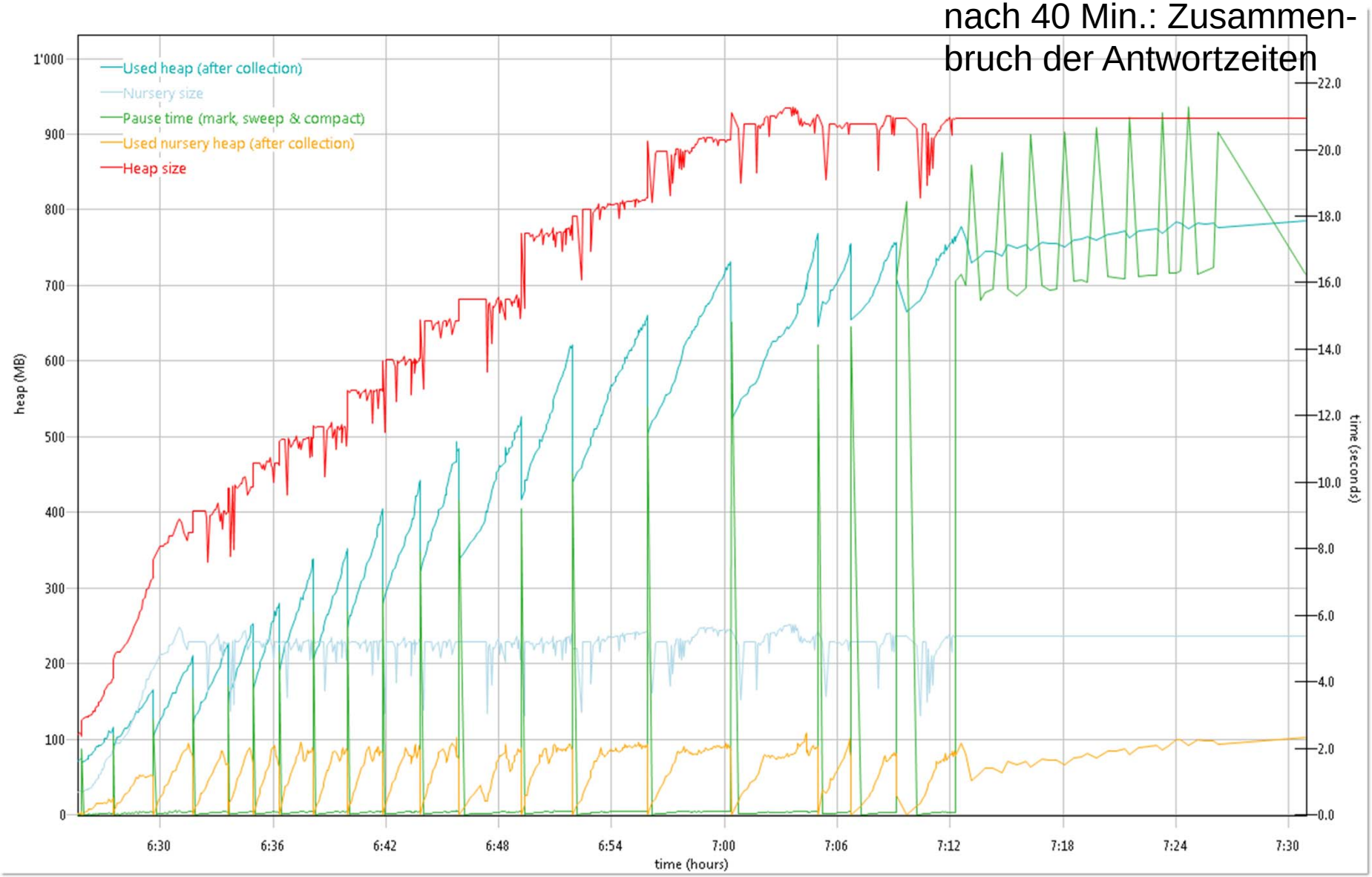


Applikation vor Tuning / Umbau



Applikation vor Tuning / nach Umbau

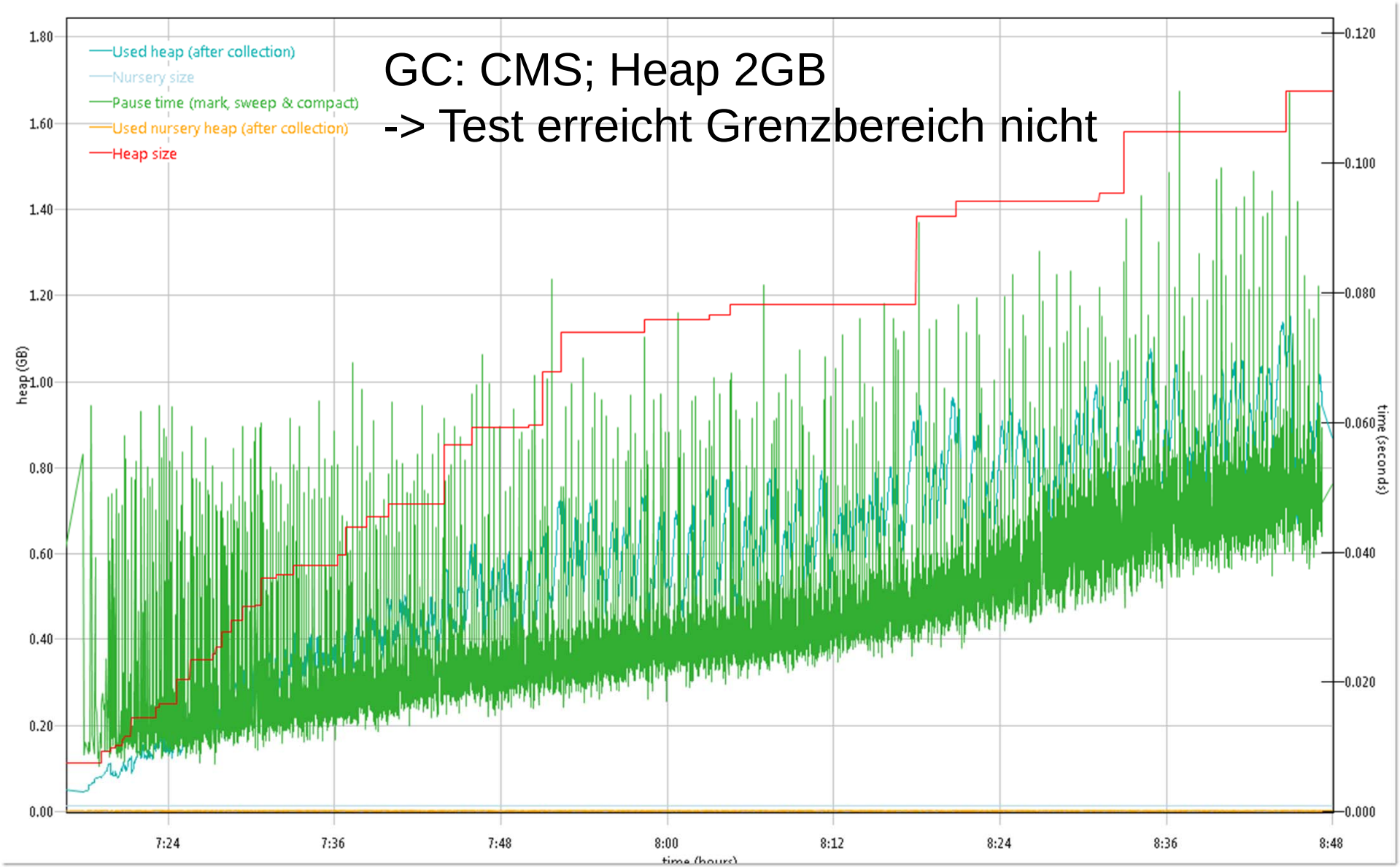
nach 40 Min.: Zusammen-
bruch der Antwortzeiten



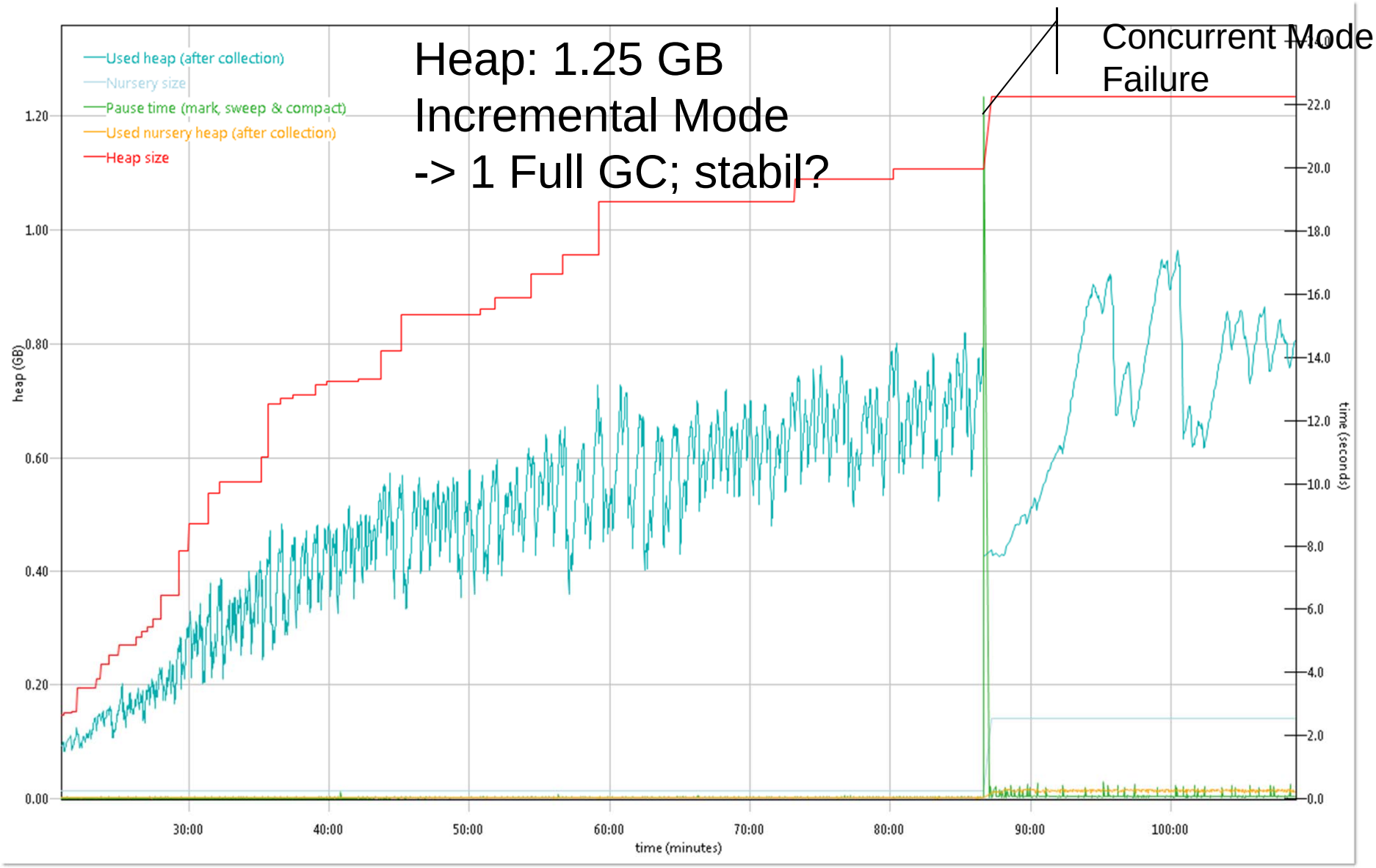
Tuning Massnahmen

- Problem
  - GC kommt nicht nach mit Aufräumen
  - viele „stop the world“ Full GCs
  - Performance sackt ab
- Lösungs-Ansätze
  - mehr Heap
  - anderer GC-Algorithmus
  - Tunen des neuen GC-Algorithmus

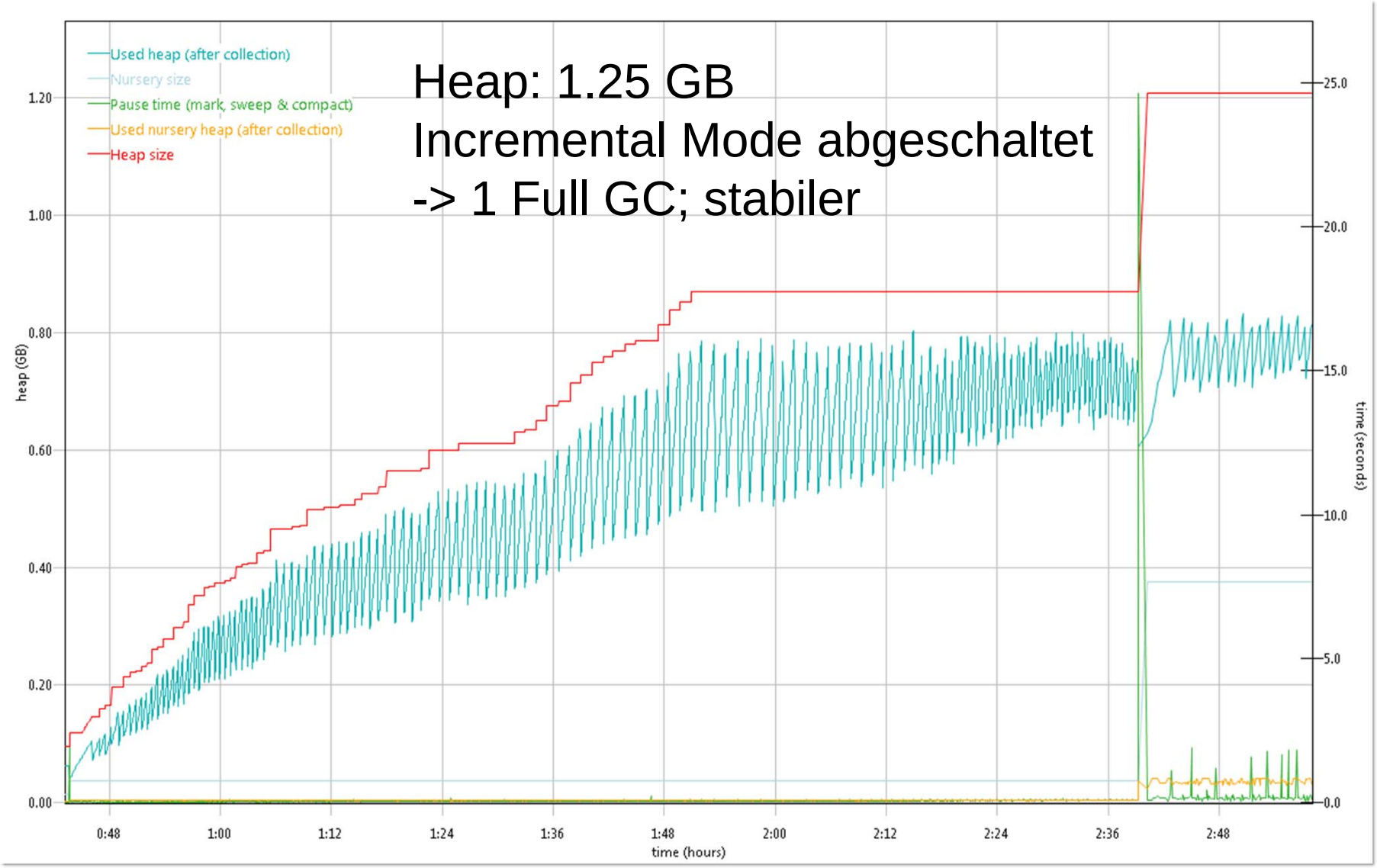
neuer GC Algorithmus / mehr Heap



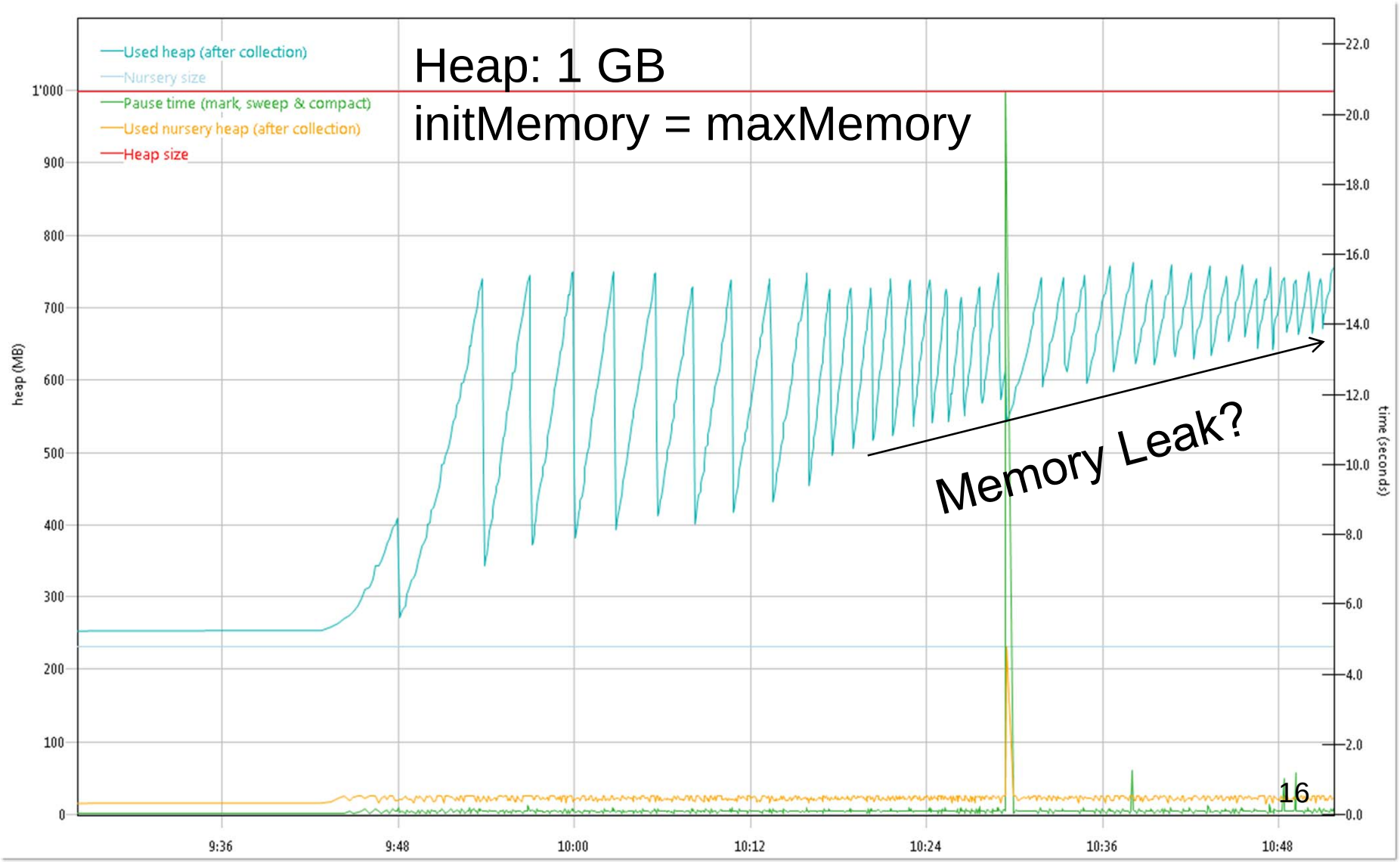
Tuning: Grenzbereich suchen



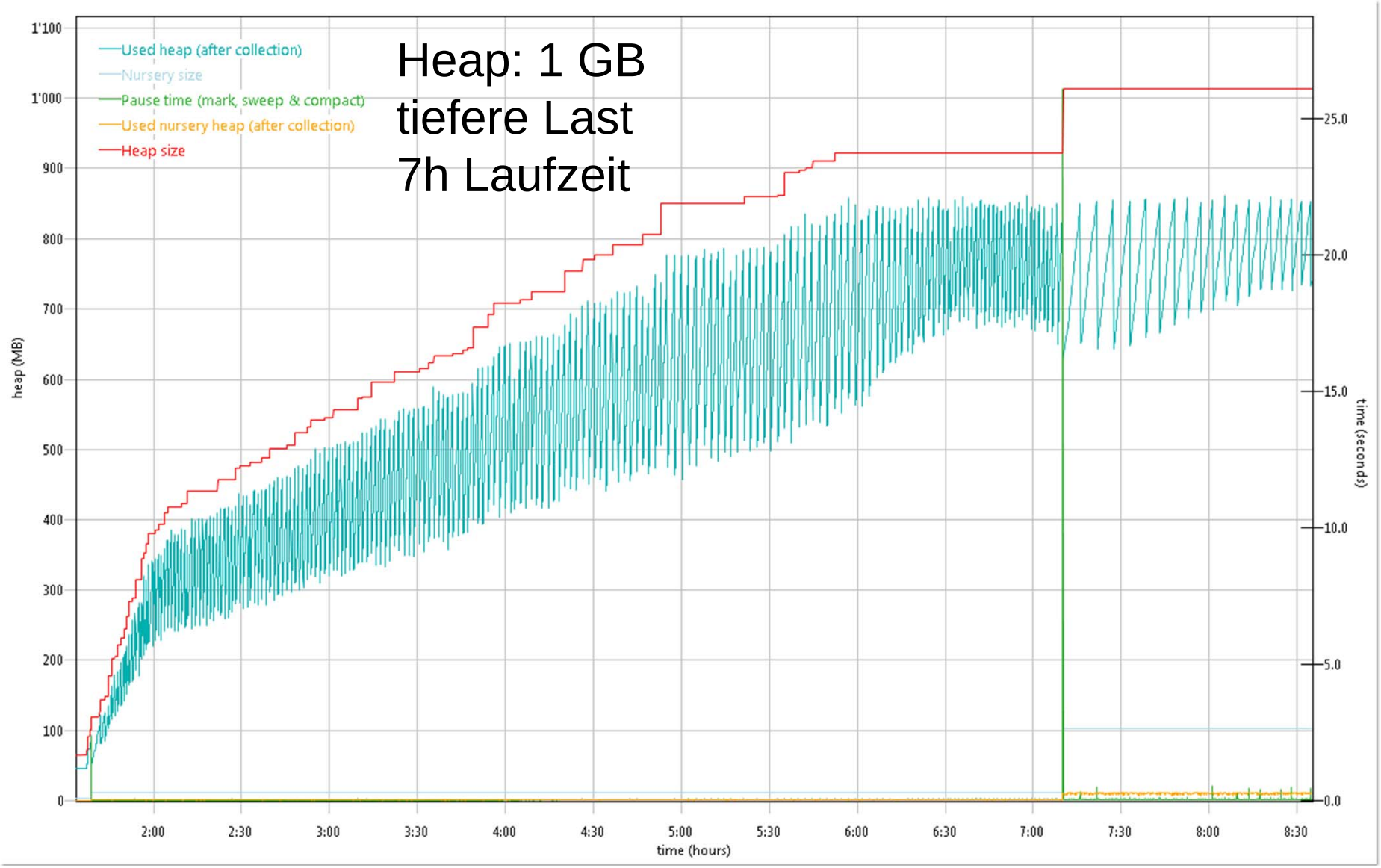
Tuning: langfristig stabil?



Tuning: langfristig stabil?



Tuning: langfristig stabil?



Entscheid: Go

- Rahmenbedingungen
  - Memory Leak kann nicht ausgeschlossen werden
  - Server werden täglich neu gestartet; eventuelles Memory Leak schlägt somit nicht zu
  - Last in Lasttests ist viel höher, als sie aktuell in Produktion ist; Situation aus Lasttest wird voraussichtlich nicht erreicht in Produktion
- Weitere Massnahmen
  - Beobachtung der produktiven Logs nach GoLive
  - Dann weitere Tuning-Massnahmen falls notwendig

Ressourcen

- http://www.javaworld.com/javaworld/jw-01-2002/jw-0111-hotspotgc.html - Erklärung zur Memory-Aufteilung und Funktionsweise der GC
- http://java.sun.com/developer/technicalArticles/Programming/turbo/ - Tuning mit parallelem und concurrent GC
- http://www.oracle.com/technetwork/java/javase/tech/memorymanagement-whitepaper-1-150020.pdf - Basiswissen für GC-Tuning
- http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140228.html - alle offiziellen Links zur GC
- http://www.ibm.com/developerworks/java/library/j-ibmtools2/index.html - IBM Garbage Collection and Memory Visualizer