HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

iSYS

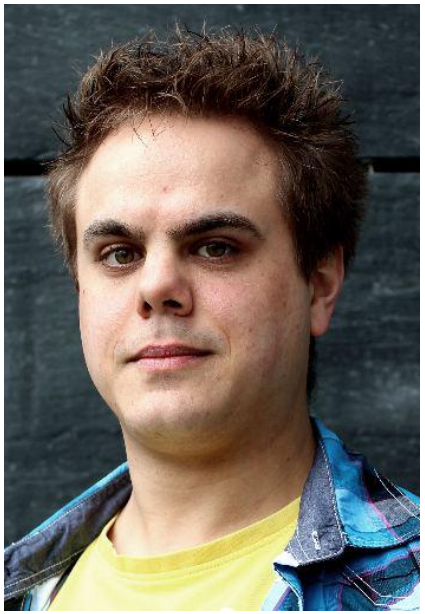
Java Garbage Collector: Funktionsweise und Optimierung

Mathias Dolag
Prof. Dr. Peter Mandl
(DOAG 2012, 20.11.2012)

Agenda

- Algorithmen
- Tuning
- Performance-Messungen
- Zusammenfassung

Der Referent



- Studium
  - 2012: Bachelor of Science (Wirtschaftsinformatik)
  - Aktuell: Master Informatik (Schwerpunkt Software-Engineering)
- Beruflich
  - Software-Entwickler bei iSYS Software GmbH
    - JavaEE
    - Rich Client Platform (RCP)

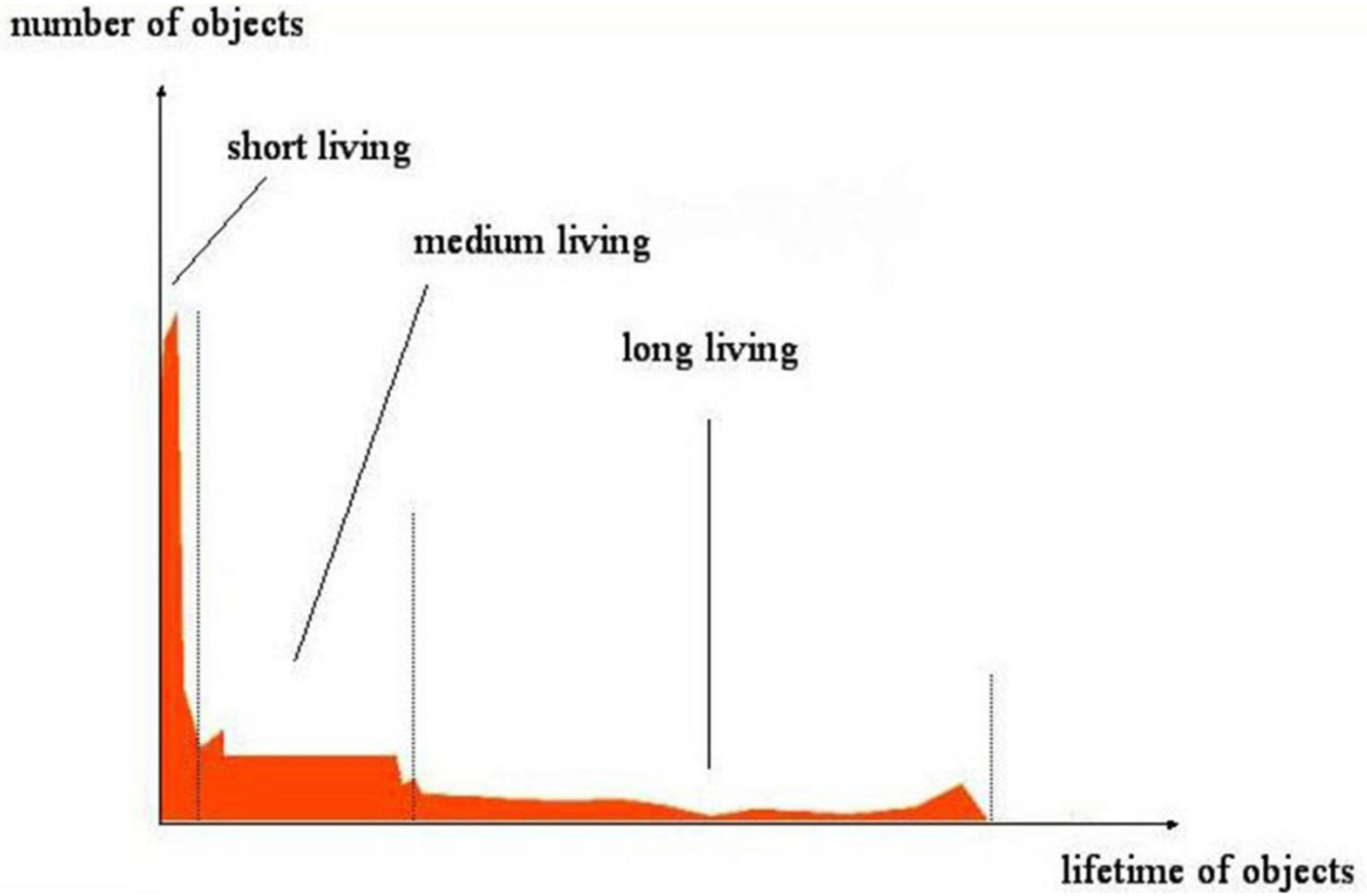
HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

iSYS

Algorithmen

Algorithmen – Generational Garbage Collecting

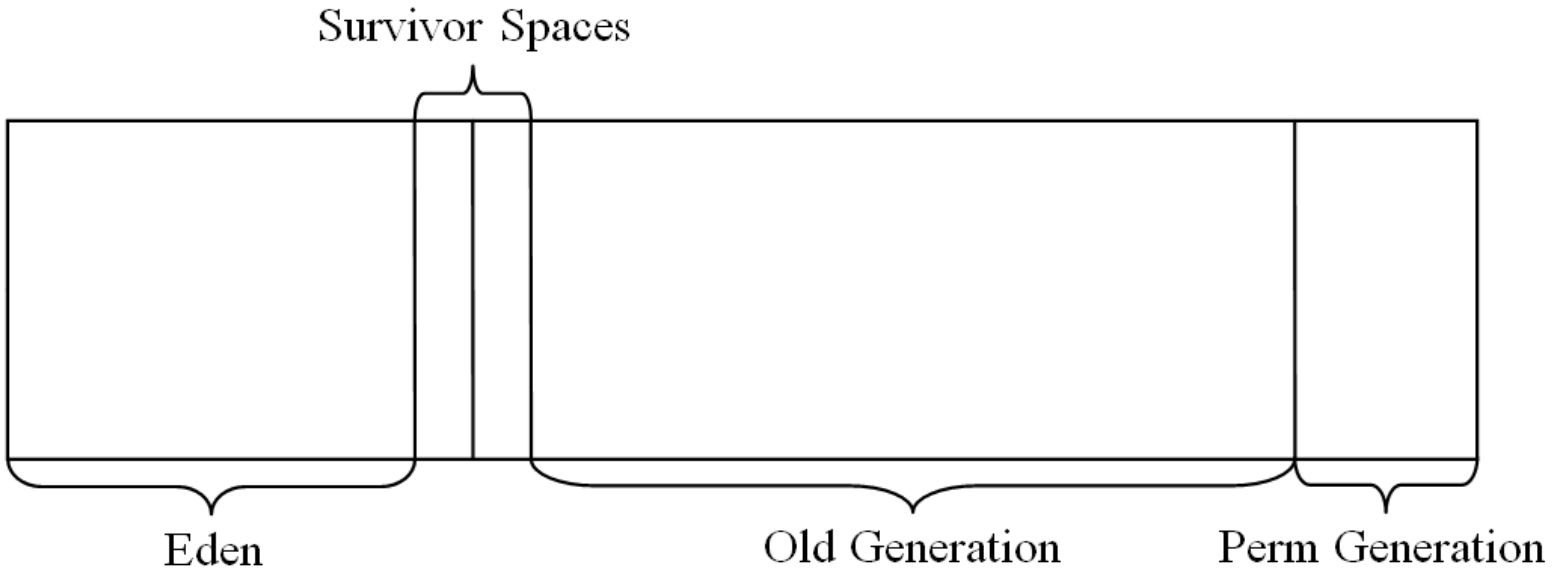
- Typische Objektpopulation einer Java-Applikation



Quelle: Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection

Algorithmen - Generational Garbage Collecting II

- Heap-Aufteilung der Sun/Oracle HotSpot JVM

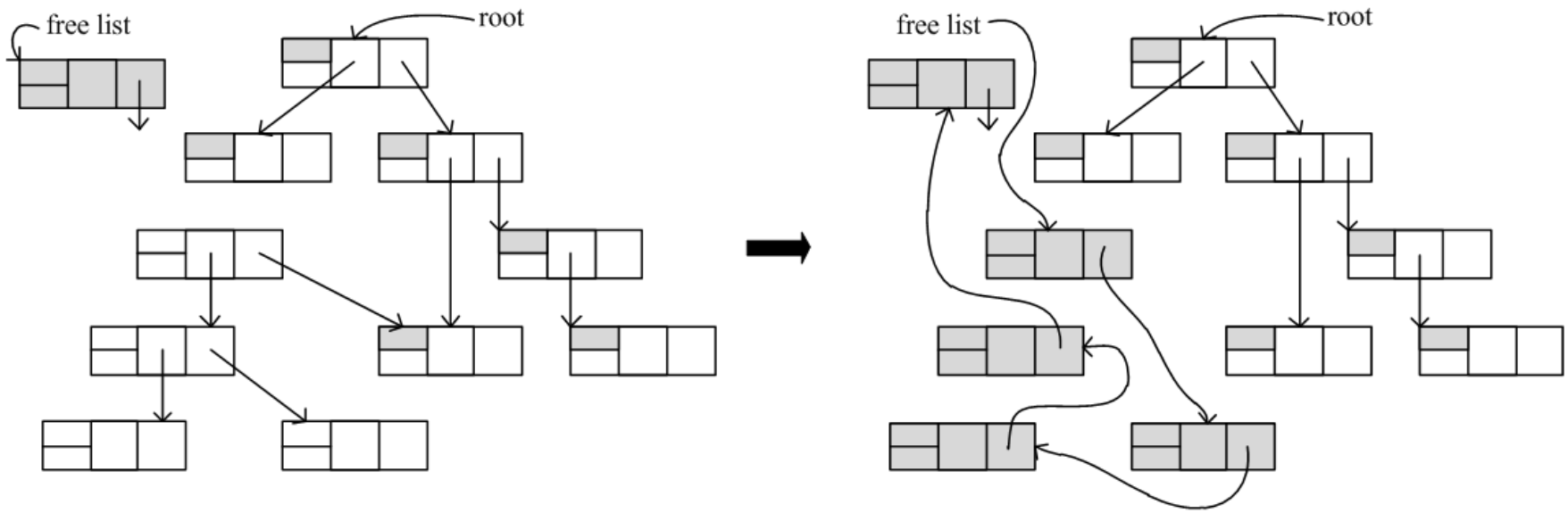


Quelle: Anlehnung an *Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection*

- Minor / Major Collection
- Tenuring Threshold
- Promotion

Algorithmen – Mark-and-Sweep

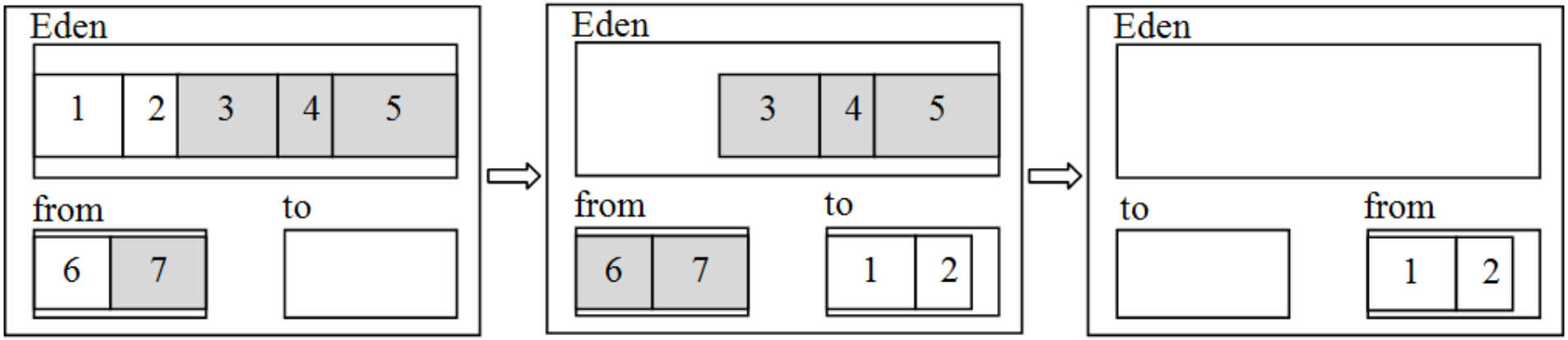
- Grundlegendster GC-Algorithmus
- Nur zwei Phasen
  - Markierung (Mark)
  - Aufräumen (Sweep)



Quelle: Anlehnung an *Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection*

Algorithmen – Mark-and-Copy

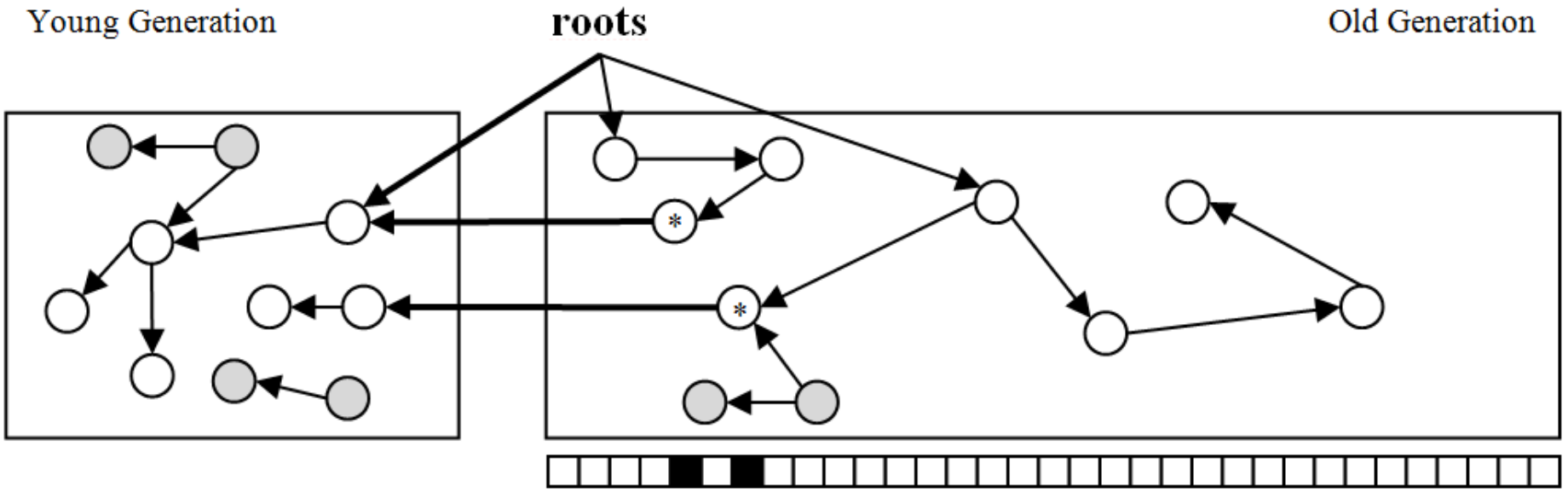
- Sun/Oracle HotSpot JVM – Variante
  - Young Generation = Eden + 2x Survivor Space
- Ziele
  - Promotion in Old Generation herauszögern
  - Vermeiden von Fragmentierung
  - Schnelle Allokation neuer Objekte



Quelle: Anlehnung an *Wie funktioniert der Java Garbage Collector; http://it-republik.de/jaxenter/artikel/2452*

Algorithmen – Mark-and-Copy II

- Intergenerational References

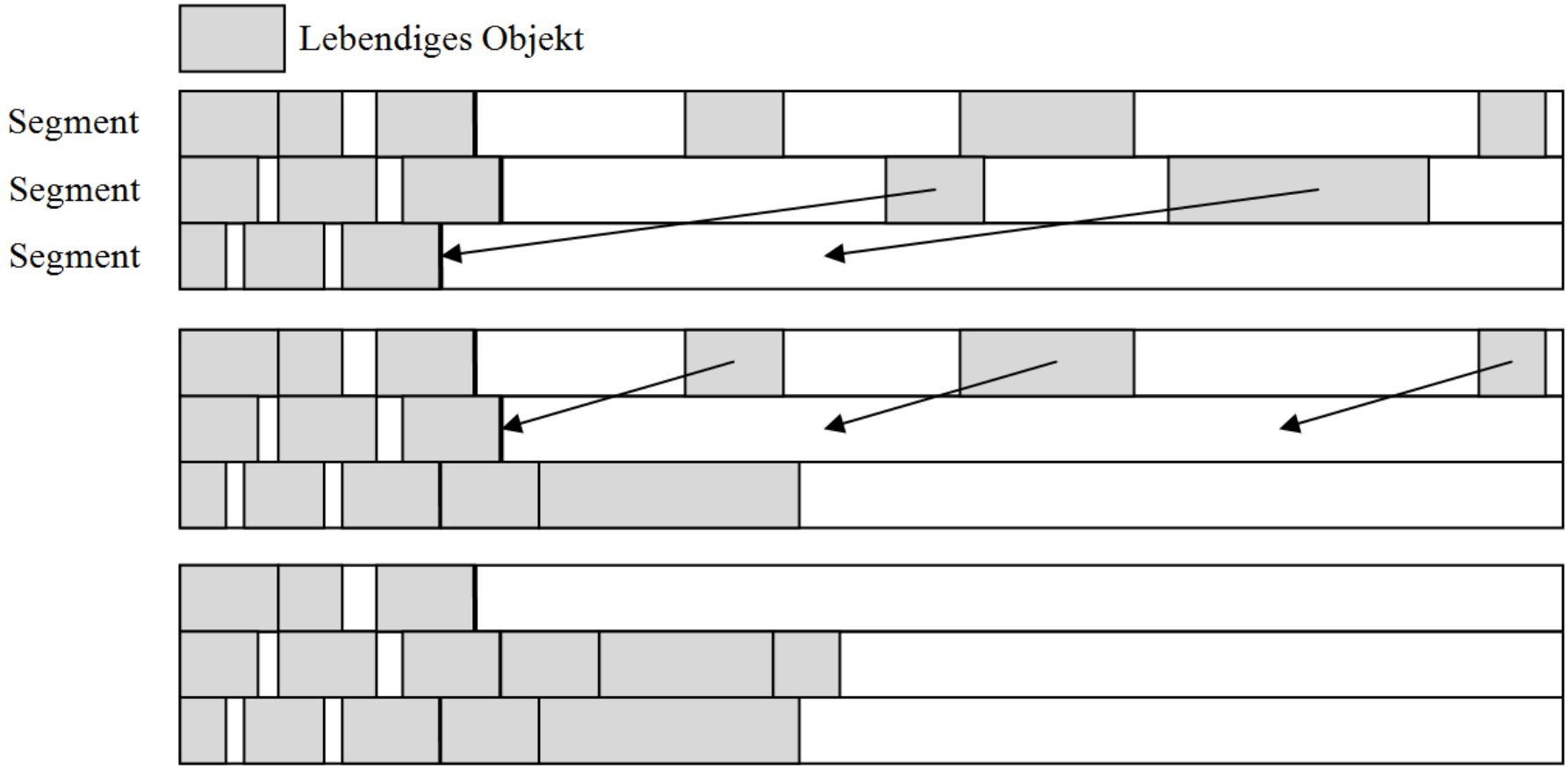


Quelle: Anlehnung an *Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection*

Algorithmen – Mark-and-Compact

- Zusammenschieben/Verdichten überlebender Objekte
- Stop-the-World – Algorithmus
- Einsatz meist auf Old Generation
- Hauptziele
  - Vermeiden von Fragmentierung
  - Schnelle Allokation
- Serielle und parallele Variante

Algorithmen – Mark-and-Compact II



Quelle: Anlehnung an *Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection*

Algorithmen – Concurrent Mark-and-Sweep

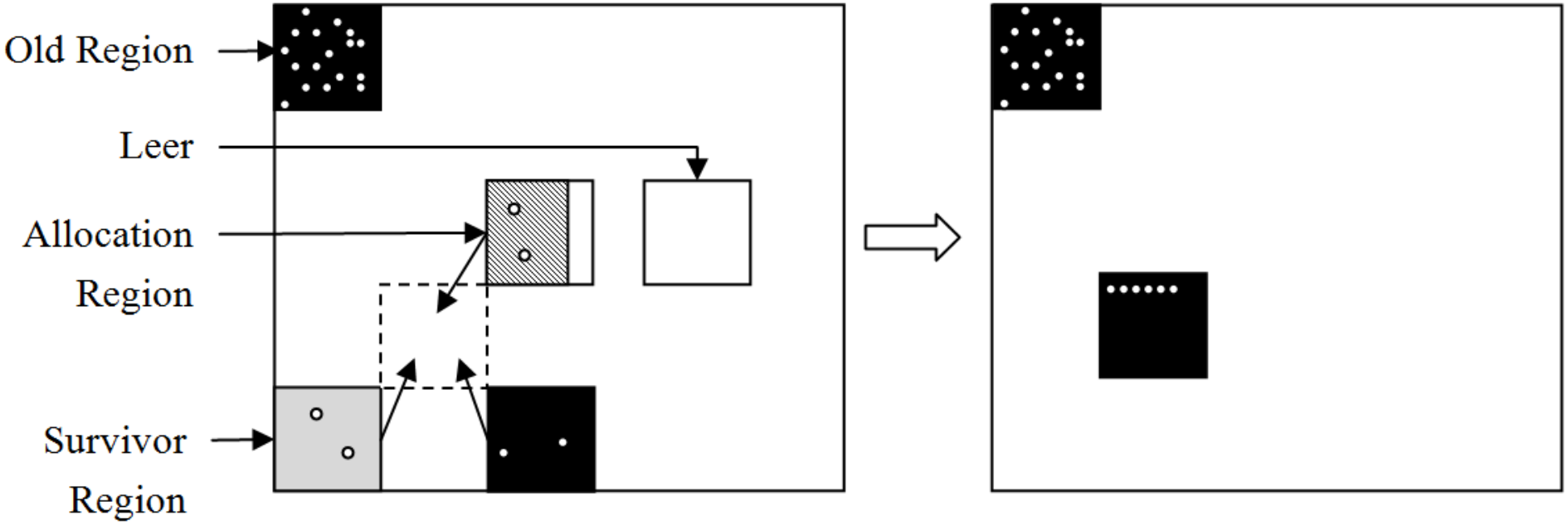
- Nebenläufig zur Applikation ausführbarer Collector
- Old Generation – Algorithmus
- Hauptziel: Verkürzung von Stop-the-World – Pausen
- Reiner *Sweep* – Algorithmus → Fragmentierung
- Allokation neuer Objekte mittels *Free Lists*
  - Annahmen über zukünftigen Speicherbedarf

Algorithmen – Garbage First (G1)

- Neuer von Sun/Oracle entwickelter Collector (seit 2004)
- Final seit Java 7, experimentell seit Java 6 U14
- Vorgebbar: Pausenzeiten / Durchsatz
- Verdichtung des Heap über Laufzeit der Applikation
- Vorrangiges Aufräumen von Regionen mit hohem „Müllanteil“

Algorithmen – Garbage First (G1) II

- Evakuierung und Verdichtung
- Bestimmung eines *Collection Set*
  - Zwei Modi: *Fully / Partially Young Mode*



Quelle: Anlehnung an *Langer, A.; Kreft, K.: Java Core Programmierung. Memory Model und Garbage Collection*

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

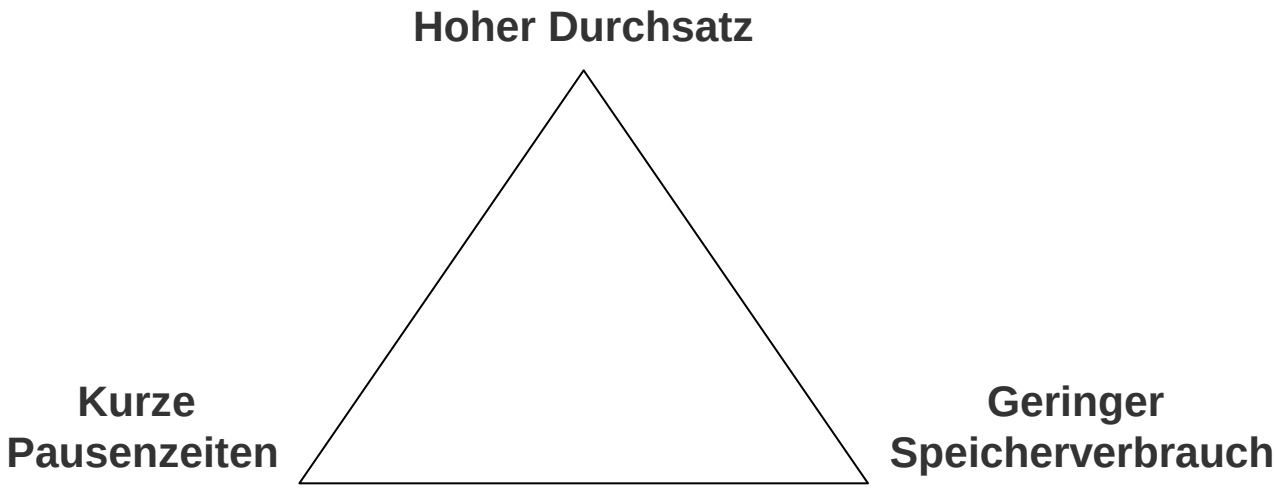
iSYS

Tuning

Tuning – Ziele & Kennzahlen

- Speicherverbrauch verringern
- Durchsatz erhöhen
- Pausenzeiten verringern

- Zielkonflikte

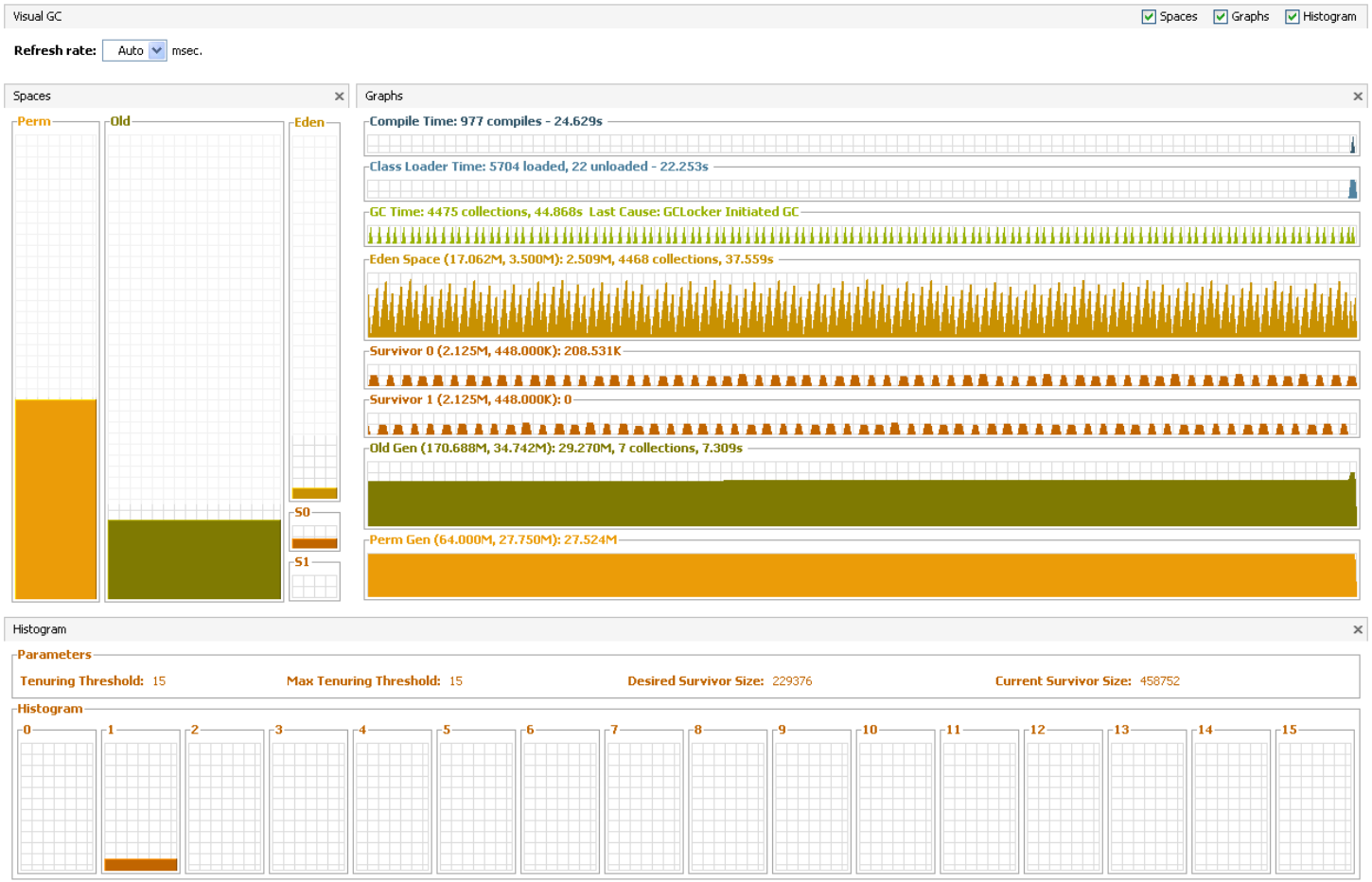


Tuning – Maßnahmen

- Durchsatz erhöhen
  - Anpassung Größenverhältnisse
    - Young Generation / Old Generation
    - Eden / Survivor Spaces
  - Promotion von Objekten verhindern
  - Parallelen Collector einsetzen (z.B. Parallel Mark-and-Compact)
- Pausenzeiten verringern
  - Geeignete Collectoren
    - CMS / G1
  - Garbage Collection hinauszögern
    - Starke Vergrößerung der Old Generation
  - Objekte in Young Generation sterben lassen
  - Explizite Garbage Collection auslösen (*System.gc();*)

Tuning – Tools

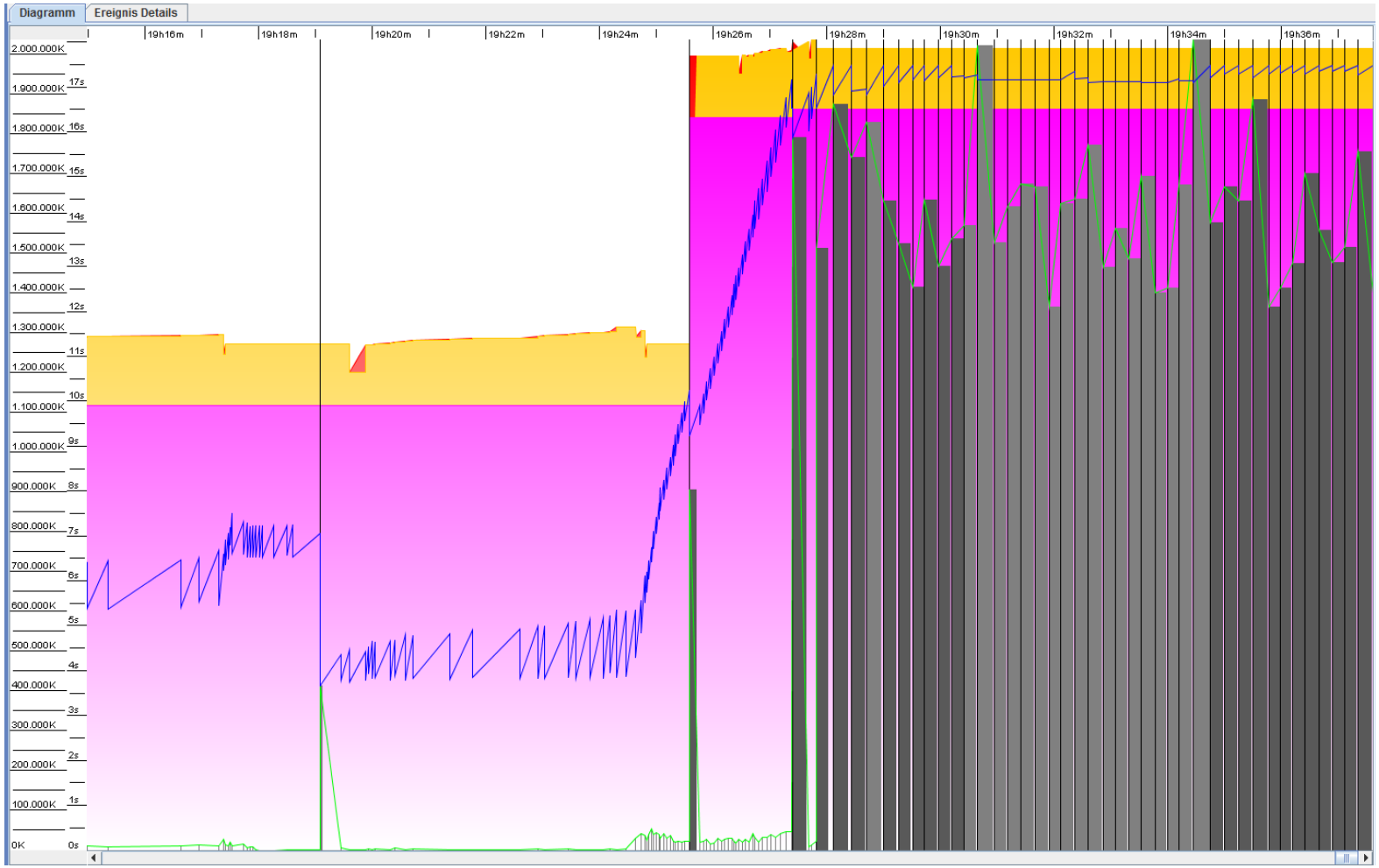
- JVisualVM (inkl. VisualGC-Plugin)



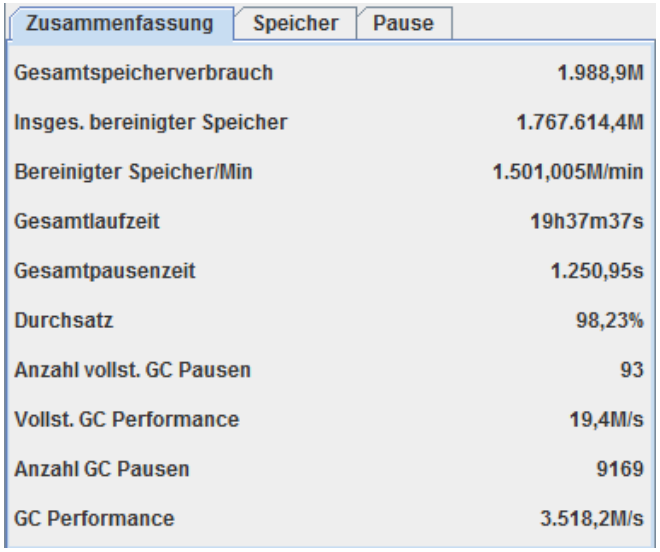
Quelle: http://cybergav.in/wp-content/uploads/2010/01/VisualVM_VisualGC.png

Tuning – Tools II

- GCViewer

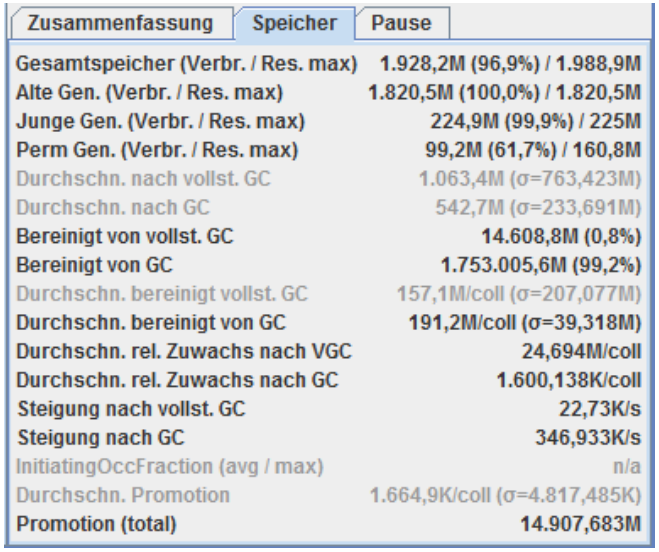


Tuning – Tools III



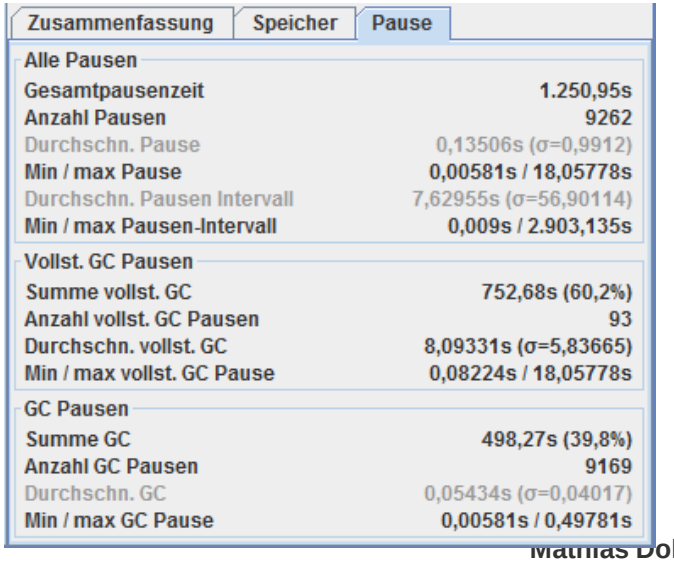
Zusammenfassung | Speicher | Pause

Gesamtspeicherverbrauch	1.988,9M
Insges. bereinigter Speicher	1.767.614,4M
Bereinigter Speicher/Min	1.501,005M/min
Gesamtlaufzeit	19h37m37s
Gesamtpausenzeit	1.250,95s
Durchsatz	98,23%
Anzahl vollst. GC Pausen	93
Vollst. GC Performance	19,4M/s
Anzahl GC Pausen	9169
GC Performance	3.518,2M/s



Zusammenfassung | Speicher | Pause

Gesamtspeicher (Verbr. / Res. max)	1.928,2M (96,9%) / 1.988,9M
Alte Gen. (Verbr. / Res. max)	1.820,5M (100,0%) / 1.820,5M
Junge Gen. (Verbr. / Res. max)	224,9M (99,9%) / 225M
Perm Gen. (Verbr. / Res. max)	99,2M (61,7%) / 160,8M
Durchschn. nach vollst. GC	1.063,4M (σ=763,423M)
Durchschn. nach GC	542,7M (σ=233,691M)
Bereinigt von vollst. GC	14.608,8M (0,8%)
Bereinigt von GC	1.753.005,6M (99,2%)
Durchschn. bereinigt vollst. GC	157,1M/coll (σ=207,077M)
Durchschn. bereinigt von GC	191,2M/coll (σ=39,318M)
Durchschn. rel. Zuwachs nach VGC	24,694M/coll
Durchschn. rel. Zuwachs nach GC	1.600,138K/coll
Steigung nach vollst. GC	22,73K/s
Steigung nach GC	346,933K/s
InitiatingOccFraction (avg / max)	n/a
Durchschn. Promotion	1.664,9K/coll (σ=4.817,485K)
Promotion (total)	14.907,683M



Zusammenfassung | Speicher | Pause

Alle Pausen

Gesamtpausenzeit	1.250,95s
Anzahl Pausen	9262
Durchschn. Pause	0,13506s (σ=0,9912)
Min / max Pause	0,00581s / 18,05778s
Durchschn. Pausen Intervall	7,62955s (σ=56,90114)
Min / max Pausen-Intervall	0,009s / 2.903,135s

Vollst. GC Pausen

Summe vollst. GC	752,68s (60,2%)
Anzahl vollst. GC Pausen	93
Durchschn. vollst. GC	8,09331s (σ=5,83665)
Min / max vollst. GC Pause	0,08224s / 18,05778s

GC Pausen

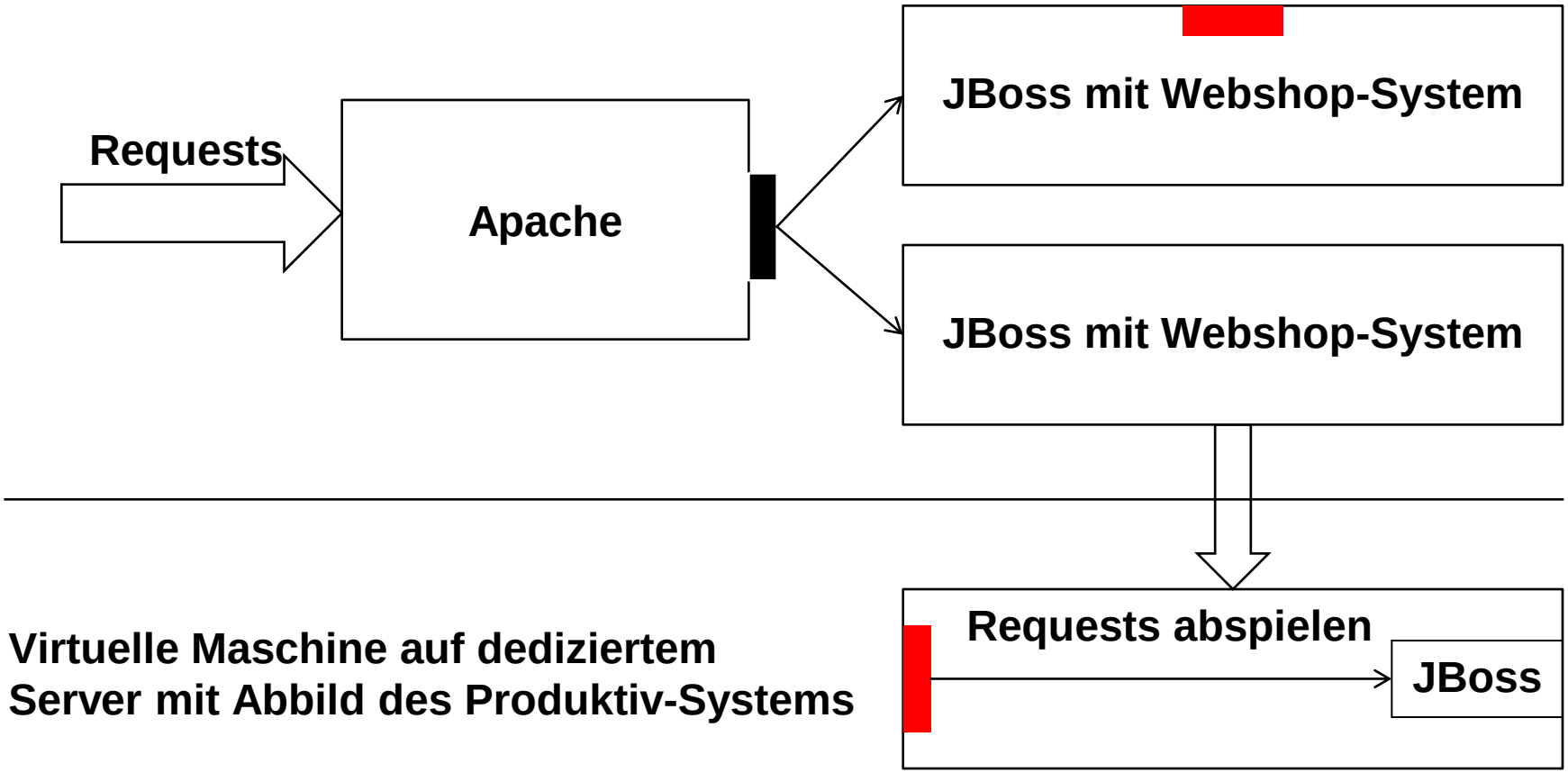
Summe GC	498,27s (39,8%)
Anzahl GC Pausen	9169
Durchschn. GC	0,05434s (σ=0,04017)
Min / max GC Pause	0,00581s / 0,49781s

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

iSYS

Performance Messungen – Ergebnisse

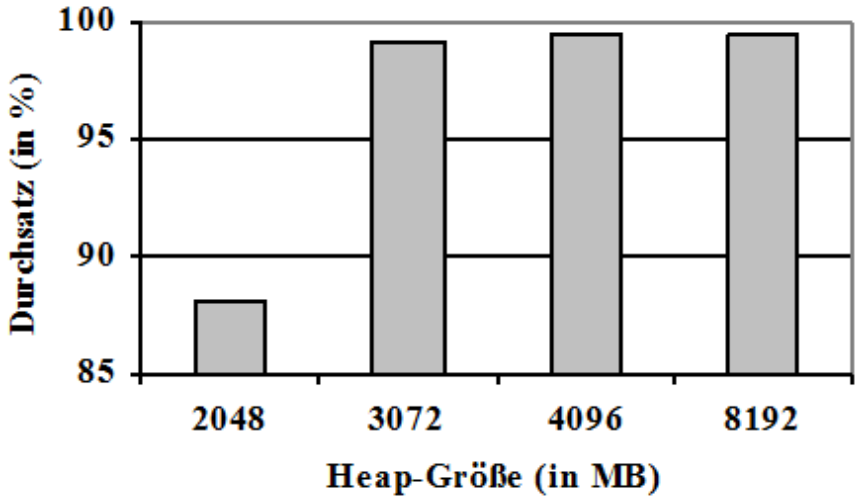
Performance Messungen – Testumgebung



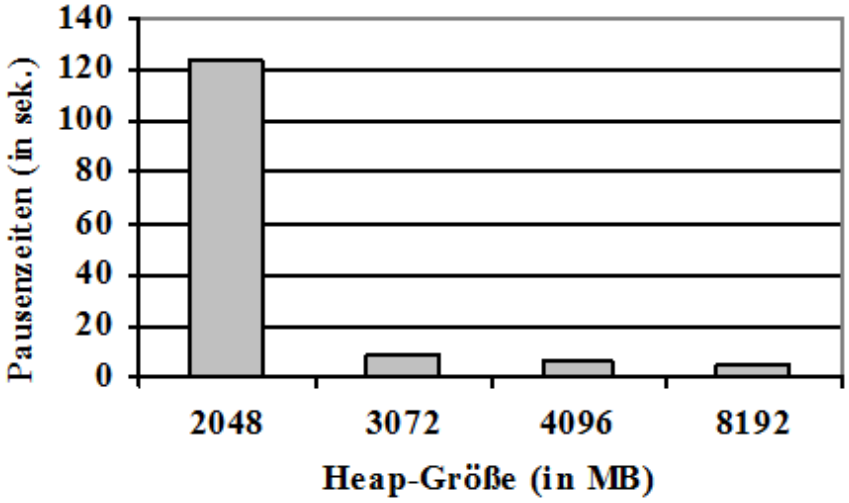
Virtuelle Maschine auf dediziertem Server mit Abbild des Produktiv-Systems

Performance Messungen – Ergebnisse

- Garbage Collection Standardeinstellungen
  - Young Generation: Parallel Mark-and-Copy
  - Old Generation: Parallel Mark-and-Compact
  - Parallele GC-Threads: 6



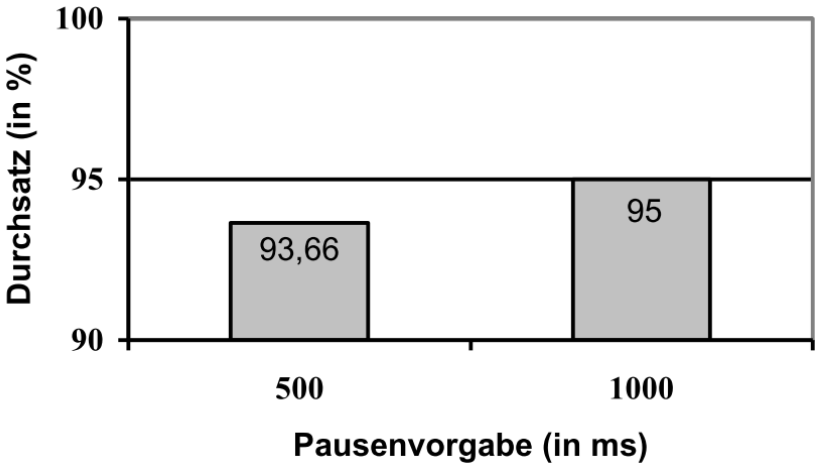
Durchsatz



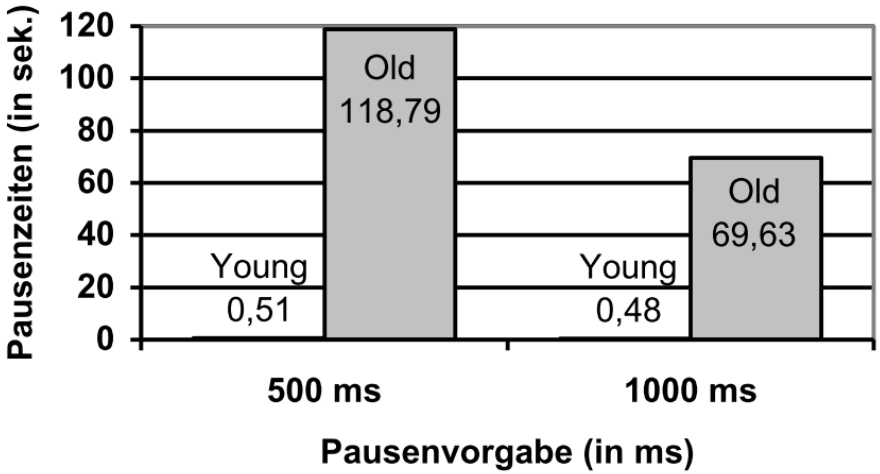
Gesamtpausenzeit (in sek.)

Performance Messungen – Ergebnisse II

- Concurrent Mark-and-Sweep
  - Heap-Größe: 16384MB
  - Young Generation: Parallel Mark-and-Copy
  - Old Generation: Concurrent Mark-and-Sweep



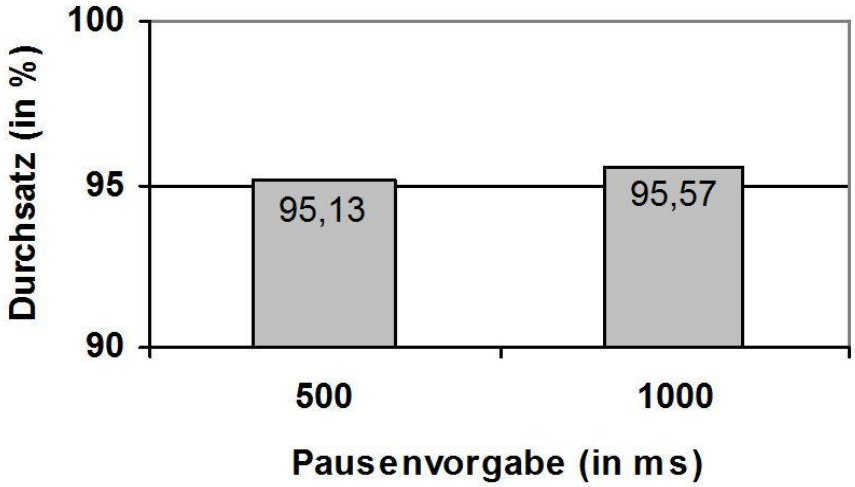
Durchsatz



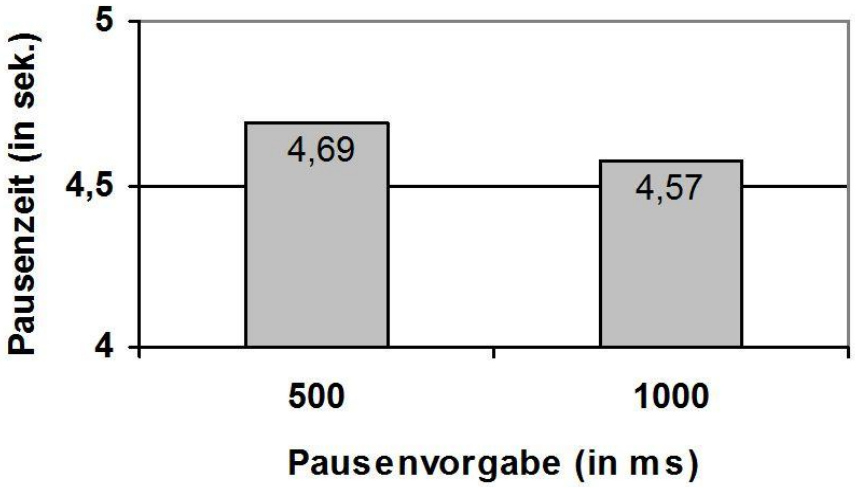
Durchschnittliche Pausenzeit

Performance Messungen – Ergebnisse III

- Garbage First (G1)
  - Heap-Größe: 16384MB
  - Parallele GC-Threads: 30



Durchsatz



Pausenvorgabe (ms)

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

iSYS

Zusammenfassung

Zusammenfassung

- Unterschiedlichste Garbage Collectoren
- Beste Kennzahlen zur Performance Messung
  - Durchsatz
  - Pausenzeit (Stop-the-World-Pause)
- Nicht möglich allgemeingültige Garbage Collection - Einstellungen zu geben
- Richtlinien
  - JVM viel (physikalischen) Arbeitsspeicher zuweisen
  - Verhältnis Young Generation – Old Generation
  - Passenden Collector-Typen auswählen

Vielen Dank für Ihre Aufmerksamkeit !

Kontaktinformationen

Mathias Dolag
iSYS Software GmbH
Grillparzer Str. 10
80665 München

E-Mail: m.dolag@isys-software.de
Telefon: +49 89 / 46 23 28 – 0
Web: http://www.isys-software.de

Prof. Dr. Peter Mandl
Hochschule München
Loth Str. 64
80335 München

E-Mail: mandl@cs.hm.edu
Telefon: +49 89 / 1265 – 3704
Web: http://www.prof-mandl.de