A Unified Theory of Garbage Collection

David F. Bacon, Perry Cheng, V. T. Rajan

presented by Andreas Haas
University of Salzburg

2009-4-29

Outline

Garbage Collection

- Garbage Collector
  - Überwache die Objekte auf dem Heap
  - Finde unerreichbare Objekte und gib deren Speicher frei
- Zwei grundlegende Lösungsansätze
  - Tracing
    - Suche alle Referenzen im Stack und in den globalen Variablen (Rootreferenzen)
    - Durchlaufe von den Rootreferenzen aus die Objektstrukturen und markiere jedes besuchte Objekt
    - Scanne den Heap und gib alle Objekte frei, die nicht markiert sind
  - Reference Counting
    - Jedes Objekt speichert die Anzahl an eingehenden Referenzen in einem Reference Counter
    - Ein Objekt wird freigegeben, wenn der Reference Counter gleich '0' wird

Gegenüberstellung: Tracing - Reference Counting

	Tracing	**Reference Counting**
Collection Style	Batch	Incremental
Kosten pro Referenzsetzen	Keine	Hoch
Durchsatz	Hoch	Gering
Wartezeiten	Lang	Kurz
Echtzeitfähig?	Nein	Ja
Findet Schleifen?	Ja	Nein

Anpassung der Algorithmen

- Tracing
  - Verwende nicht nur ein Mark-Bit, sondern einen Zähler
- Reference Counting
  - Warte mit dem Verringern des Zählers
    - Stattdessen wird das Objekt einer Liste hinzugefügt
    - Diese Liste wird abgearbeitet, wenn Garbage Collection gestartet wird

Notationen

V ... Die Menge aller Objekte

E ... Die Menge aller Referenzen

$p(v)$ ... Der Referenz Count des Objektes v

W ... Die Menge der Objekte, die sich der Garbage Collector zum Abarbeiten gemerkt hat

Die Algorithmen

Tracing	Reference Count
collect-by-tracing() **initialize-for-tracing(W)** scan-by-tracing(W) sweep-for-tracing()	collect-by-counting(W) scan-by-counting(W) sweep-for-counting()
scan-by-tracing(W) while W not empty remove w from W **p(w) = p(w) + 1** **if p(w) == 1** for each reference x from w add x to W	scan-by-counting(W) while W not empty remove w from W **p(w) = p(w) - 1** **if p(w) == 0** for each reference x from w add x to W
sweep-for-tracing() for each v in V if p(v) == 0 free (v) **p(v) = 0**	sweep-for-counting() for each v in V if p(v) == 0 free (v)
new(x) p(x) = 0	new(x) p(x) = 0
	dec(x) add x to W
	inc p(x) = p(x) + 1
	assign(a, p) l = [a] [a] = p dec(l) inc(p)

Vergleich der Algorithmen

	Tracing	**Reference Counting**
Startmenge	Roots	Anti-Roots
Traversierung	Vorwärts von den Roots	Vorwärts von den Anti-Roots
Welche Objekte	Live Objects	Dead Objects
Initialer Ref.Counter	Niedrig (0)	Hoch
Berechnung	Addition	Subtraktion
Zusätzliche Iteration	Sweep Phase	Zyklenerkennung

Fixpunkt Formulierung für Garbage Collection

- Berechne für ein Objekt die Anzahl der Referenzen
  - Von Objekten, deren Reference Count nicht null ist
  - Vom Stack und von globalen Variablen (Roots)
- Tracing konvergiert gegen den kleinsten Fixpunkt
  - Da nur bei den Root Referenzen begonnen wird, werden keine Cycles eingeschlossen
- Reference Counting konvergiert gegen den größten Fixpunkt
  - Weil Cycles in sich Fixpunkte sind
  - Weil Cycles mit Reference Counting nie gefunden werden

Analyse bestehender Garbage Collectors

1. Wie wird der Heap aufgeteilt?
   - Unified Heap: Der gesamte Heap wird als ein Speicherbereich betrachtet
   - Split Heap: Der Heap wird in zwei Bereiche aufgeteilt, die voneinander getrennt betrachtet werden
   - Multiple Heap: Der Heap wird in beliebig viele Bereiche aufgeteilt
2. Wie werden die einzelnen Bereiche im Heap verwaltet?
   - Tracing oder Reference Counting
3. Wie werden die Referenzen zwischen den Heaps verwaltet?
   - Tracing oder Reference Counting
4. Wie werden die Root Referenzen zu den einzelnen Heaps verwaltet?
   - Tracing oder Reference Counting

Beispiel: Generational Garbage Collector

1. Wie wird der Heap aufgeteilt?
   - Split Heap
     - Ein Bereich für langlebigen Speicher: Mature Space
     - Ein Bereich für kurzlebigen Speicher: Nursery
2. Wie werden die einzelnen Bereiche im Heap verwaltet?
   - Mit Tracing (Gibt auch Versionen mit Reference Counting im Mature Space und Tracing im Nursery)
3. Wie werden die Referenzen zwischen den Heaps verwaltet?
   - Nursery to Mature Space: Gar nicht
     - Der Mature Space kann erst aufgeräumt werden, wenn der Nursery leer ist
   - Mature Space to Nursery: Durch Reference Counting
4. Wie werden die Root Referenzen zu den einzelnen Heaps verwaltet?
   - Mit Tracing

Beispiel: Deferred Reference Counting

1. Wie wird der Heap aufgeteilt?
   - Unified Heap
2. Wie wird der Heap verwaltet?
   - Mit Reference Counting
   - Ein Objekt wird beim Zählerwert nicht gleich freigegeben, sondern in einer Zero Count Table gespeichert
3. Wie werden die Root Referenzen zum Heap verwaltet?
   - Mit Tracing
   - Alle Objekte, auf die eine Root Referenz zeigt, werden aus der Zero Count Table entfernt
   - Alle anderen Objekte werden freigegeben

- Das Ändern von Root Referenzen wird billiger
  - Ist dafür nicht mehr rein Incrementell

Beispiel: Partial Tracing

1. Wie wird der Heap aufgeteilt?
   - Unified Heap
2. Wie wird der Heap verwaltet?
   - Mit Tracing
3. Wie werden die Root Referenzen zum Heap verwaltet?
   - Mit Reference Counting
     - Ich halte mein Root Set immer aktuell
     - Ich erspare mir das Scannen des Stacks und der globalen Variablen bei der Collection

- Sinnvoll, wenn das Finden der Roots sehr aufwendig ist
  - Bei Memory Management ist genau das Gegenteil der Fall

Kosten für Tracing

- Speicherkosten
  - Ein Mark Bit pro Objekt
  - Ein Stack für das recursive Durchlaufen des Objektgraphen
    - Die Stackgröße ist nicht vorhersehbar
    - Meistens wird ein Stack mit fixer Größe verwendet
- Zeitkosten
- Jedes Mal wenn der Heap voll wird:
  - Die Anzahl der Roots (Mark Phase)
  - Anzahl der Live Objects (Mark Phase)
  - Anzahl der abgegangenen Referenzen (Mark Phase)
  - Anzahl aller Objekte (Sweep Phase)

Kosten für Reference Counting

- Speicherkosten für Reference Counting
  - Ein 'Reference Counting'-Word pro Objekt
- Speicherkosten für die Zyklenerkennung
  - Ein Stack wie beim Tracing
- Zeitkosten pro Collection
  - Anzahl der Dead Objects
  - Anzahl der Referenzen von diesen Objekten
- Zeitkosten pro Mutation
  - Jedes Ändern einer Referenz kostet
- Zeitkosten für die Zyklenerkennung
  - Wie beim Tracing
  - Die Zyklenerkennung wird aber nicht so oft gestartet

Zusammenfassung

- Jeder konventionelle Garbage Collector ist ein Hybrid aus Tracing und Reference Counting
  - Auf welche Bereiche wird welche Methode verwendet?
  - Bestehende Kollectoren können kategorisiert werden
- Die beiden Verfahren sind zueinander dual
- Die Kosten sind sehr unterschiedlich
- Beide Methoden haben fundamentale Nachteile
  - Tracing:
    - lange Wartezeiten
  - Reference Counting:
    - Fehlende Zyklenerkennung
    - Referenzsetzen dauert lange

Danke für eure Aufmerksamkeit