Evaluation επιδόσεων των Java Garbage Collectors

Λευτέρης Τριποδιανός 4475

Φώτης Πελαντάκης 4988

G1

- Χωρίζει την μνήμη σε ίσου μεγέθους regions
- Εντοπίζει τα regions με μεγάλα ποσοστά από garbage
- Συλλέγει όλα τα garbage και τα διαγράφει
- Στοχεύει σε ισορροπία μεταξύ καλού throughput και Stop-the-World pauses

Parallel

- Ο GC κάνει την συλλογή των garbage παράλληλα με πολλαπλά νήματα
- Όλες οι φάσεις του GC γίνονται σε Stop-the-world pauses
- Θυσιάζει latency για να πετύχει όσο το δυνατόν υψηλότερο throughput

Shenandoah GC

- Χωρίζει το heap σε regions
- Εκτελεί το collection και την μετακίνηση αντικειμένων από γεμάτα regions σε λιγότερο γεμάτα
- Στόχος είναι τα pauses να είναι πολύ μικρά, ανεξάρτητα από το μέγεθος του heap, αποφεύγοντας stop-the-world pauses

ZGC

- Τα πάντα εκτελούνται σχεδόν παράλληλα με την εφαρμογή
- Η μετακίνηση των αντικειμένων γίνεται χωρίς σημαντικές παύσεις
- Χρησιμοποιεί *barriers* για να ανακατευθύνει τις προσπελάσεις κατά τη διάρκεια της μετακίνησης.

Γιατί επιλέξαμε τα G1, Parallel, Shenandoah και ZGC;

- G1 ➔ ισορροπία ανάμεσα σε latency και throughput.
- Parallel ➔ υψηλό throughput, μεγάλα pauses.
- Shenandoah ➔ παράλληλο collection και μετακίνηση αντικειμένων με πολύ χαμηλά pauses.
- ZGC ➔ πολύ μικρά pauses, ιδανικό για αρκετά μεγάλα heaps.

- Δοκιμάζουμε τη συμπεριφορά του κάθε GC σε διαφορετικά μεγέθη heap.
  - Ορισμένοι GC δουλεύουν καλύτερα σε μεγάλα heaps (Shenandoah, ZGC)
  - Άλλοι είναι αποδοτικότεροι σε πιο μικρά heaps (Parallel)

Γιατί Avrora, Lusearch και Tomcat;

- Το **Avrora** είναι ένα εργαλείο προσομοίωσης και ανάλυσης για προγράμματα που τρέχουν σε μικροελεγκτές AVR.
  - Αξιολόγηση του GC σε περιπτώσεις που υπάρχουν πολλές γρήγορες κατανομές και απελευθερώσεις μνήμης.
- Το **Lusearch** είναι μια μηχανή αναζήτησης βασισμένη στη βιβλιοθήκη Lucene.
  - Έντονη χρήση του GC
  - κατάλληλο για αξιολόγηση του GC σε περιβάλλον με έντονο collection
- Το **Tomcat** προσομοιώνει έναν web server → αντικατοπτρίζει ένα ρεαλιστικό server workload.
  - Ιδανικό για αξιολόγηση ενός GC σε περιβάλλοντα ανάπτυξης όπως εφαρμογές web.

Avrora

GC	Heap Size (GB)	Max Pause (ms)	Avg Pause (ms)	Throughput	STW Pauses
G1	4	11.818	11.599	99.44%	3
Parallel	4	8.337	4.387	99.57%	6
Shenandoah	4	0.217	0.045	99.99%	12
ZGC	4	0.023	0.008	99.999%	9
G1	12	30.61	26.835	98.85%	3
Parallel	12	8.676	4.594	99.55%	6
Shenandoah	12	0.293	0.063	99.99%	12
ZGC	12	0.025	0.009	99.999%	9

Lusearch

GC	Heap Size (GB)	Max Pause (ms)	Avg Pause (ms)	Throughput	STW pauses
G1	4	19,6	5,43	97,82%	32
Parallel	4	16,83	1,84	98,57%	57
Shenandoah	4	0,285	0,071	99,9%	116
ZGC	4	0,073	0,012	99,98%	84
G1	12	37,96	11,31	97,94%	14
Parallel	12	20,48	3,36	98,94%	23
Shenandoah	12	0,36	0,098	99,92%	68
ZGC	12	0,11	0,019	99,992%	33

Tomcat

GC	Heap Size (GB)	Max Pause (ms)	Avg Pause (ms)	Throughput	STW Pauses
G1	4	46.36	13.676	99.90%	14
Parallel	4	41.377	9.739	99.89%	20
Shenandoah	4	0.328	0.087	99.997%	56
ZGC	4	0.128	0.012	99.999%	132
G1	12	65.808	24.191	99.90%	8
Parallel	12	0.333	0.096	99.999%	23
Shenandoah	12	0.293	0.063	99.99%	12
ZGC	12	0.024	0.010	99.9998%	36

Σχολιασμός αποτελεσμάτων για τα pauses

- ZGC έχει τα πιο μικρά pauses.
- Shenandoah παραμένει κοντά στον ZGC, αν και έχει ελαφρώς μεγαλύτερα pauses.
- G1 έχει σημαντική αύξηση παύσεων στα 12GB, κάτι που δείχνει πρόβλημα στο scaling.
- Parallel κρατά σταθερές τιμές σε κάθε heap size, αλλά έχει σχετικά μεγαλύτερες παύσεις από τους concurrent GCs.

- Πώς επηρεάζουν οι παύσεις την εκτέλεση της εφαρμογής;
  - Στους ZGC και Shenandoah τα προγράμματα έχουν τη χαμηλότερη συνολική διάρκεια εκτέλεσης, παρόλο που έχουν περισσότερες STW παύσεις ➔ οι παύσεις είναι πολύ μικρές.
  - G1 έχει τις μεγαλύτερες παύσεις, και αυτό επιβαρύνει άμεσα τον συνολικό χρόνο εκτέλεσης.

Αξιολόγηση του throughput

- ZGC και Shenandoah προσφέρουν το υψηλότερο throughput ➔ σχεδόν 100% του χρόνου αφιερώνεται στην εφαρμογή.
- G1 χάνει απόδοση σε μεγάλο heap (98.85%), πιθανόν λόγω των stop-the-world pauses.
- Parallel διατηρεί καλό throughput, αλλά δεν αγγίζει τους concurrent collectors.

Ανάλυση επίδρασης του μεγέθους του heap

Μεταβολή παύσεων:

- **G1**: η αύξηση heap από 4GB σε 12GB **αυξάνει έντονα τις παύσεις**
- **Shenandoah** και **ZGC** βελτιώνουν ή διατηρούν παύσεις χαμηλά σε μεγαλύτερο heap.
- **Parallel** διατηρεί σταθερές παύσεις.

Μεταβολή throughput:

- **G1** χειροτερεύει με μεγαλύτερο heap.
- Οι άλλοι GC δεν επηρεάζονται αρνητικά.

STW και memory overhead:

- **Shenandoah και ZGC** έχουν περισσότερες αλλά **πολύ μικρές STW παύσεις**.
- **Parallel** έχει λιγότερες αλλά πιο "κοφτές" παύσεις.

Συσχέτιση με θεωρία των Garbage Collectors

GC	Θεωρία	Αποτελέσματα
G1	Region-based, parallel, mixed GC, στόχος pause predictability	Σε μικρό heap, δουλεύει καλά. Σε μεγάλο heap, κάνει μεγάλες παύσεις και πέφτει το throughput.
Parallel	• "Throughput first“ • stop-the-world • Multithreaded	• Καλό throughput • Καθυστερεί με μεγάλες παύσεις.
Shenandoah	Concurrent GC με low pause design	Διατηρεί χαμηλές παύσεις ακόμα και σε μεγαλύτερο heap. Αποδοτική συμπεριφορά.
ZGC	• Concurrent • region-based • color-marking	οι παύσεις είναι αμελητέες, throughput μέγιστο.

THANK YOU