FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Výkonnost a spolehlivost prog. systémů

Benchmark zvolené knihovní funkce

Ondřej Drtina
A20N0077P
drtinao@students.zcu.cz

Obsah

1 Zadání

V rámci této úlohy bylo cílem vytvořit benchmark vybrané knihovní funkce. V mém případě bylo požadováno srovnat výkonnost knihovní funkce retainAll ve spojitosti s kolekcemi ArrayList a LinkedList. Na implementaci byly kladeny následující základní požadavky:

- musí být využita vhodná metoda benchmarku
- musí být omezeny vnější vlivy (garbage collector atp.)
- benchmark bude proveden s vypovídajícím množstvím dat
- parametr fce retainAll bude kolekce stejného typu (= ArrayList / LinkedList)
- bude zachována cca polovina původní kolekce

Na formát / obsah dokumentace jsou kladeny následující požadavky. Dodaná dokumentace musí obsahovat alespoň:

- popis prostředí, ve kterém test probíhal - platformu, verze OS, build OS, verze JRE atp.
- popis způsobu měření - funkce, pomocí které měření probíhalo atd.
- získané výsledky a jejich statistické vyhodnocení - střední hodnota, odchylky, histogram atp.
- analýzu problému, které byly zjištěny během měření - např. garbage collector

1.1 Celé znění úlohy

Celé zadání úlohy je možno nalézt online zde.

2 Analýza problému

Po přečtení zadání úlohy bylo jasné, že její splnění bude vyžadovat znalost základních vlastností programovacího jazyka Java - nic jiného není z programátorského hlediska třeba. Nutností je pouze schopnost práce s objekty typu ArrayList / LinkedList a znalost funkcí pro měření času, jež Java poskytuje.

Pro vyhotovení hodnocení měření je nutné znát matematické vzorce pro střední hodnotu, odchylky a umět vytvořit histogram z naměřených hodnot.

2.1 ArrayList

Obousměrně zřetězený seznam s dynamickou velikostí. Tedy každá položka obsahuje dva pointery - jeden ukazuje na předchozí prvek, druhý na následující prvek.

2.2 LinkedList

Jedná se o lineární datovou strukturu, v rámci které má každý objekt svou datovou a adresní část. Na elementy (objekty) je odkazováno pomocí pointerů / adres.

2.3 Metoda retainAll

Metoda využívána v prog. jazyce Java. Po její exekuci nad určitým objektem zůstanou v objektu pouze ty prvky, jež obsahuje i kolekce, která byla předána funkci retainAll jako parametr.

2.4 Střední hodnota, histogram

Za střední hodnotu lze v dané oblasti považovat aritmetický průměr - tedy součet všech naměřených hodnot vydělíme jejich počtem. Histogram říká, kolikrát během měření bylo dosaženo určitého výsledku.

3 Popis implementace

Předmětem kapitoly je základní popis tříd v přítomných v programu. Metody zde nejsou popsány, jelikož zdrojový kód je důkladně okomentován.

3.1 Starter.java

Vstupní bod aplikace, obsahuje metodu `main`. Při spuštění programu jsou přítomnými metodami nejprve vygenerována testovací data. Za demo data je považováno 300 000 náhodně generovaných textových řetězců - každý délky 10.

Testovací data jsou následně převedeny na objekty typu ArrayList (resp. LinkedList), jež je možno použít jako základní kolekci pro benchmark.

Následně je z vygenerovaných dat zvolena polovina prvků, jež bude určena pro smazání - tedy 150 000 prvků. Smazán je každý druhý prvek z původní kolekce - aby bylo dosaženo smazání prvků napříč celou množinou prvků a nikoliv třeba jen prvků na začátku kolekce.

3.2 Benchmarker.java

Třída zajišťující benchmark požadovaných datových struktur. Obsažené metody jsou statické a vždy přijímají dva parametry - objekt obsahující všechna vygenerovaná data a objekt, který je použit jako parametr metody retainAll.

Časová značka je získána bezprostředně před voláním metody retainAll, i po jejím volání. Celkový čas trvání získám jako rozdíl těchto hodnot.

3.3 Použitý programovací jazyk, vývojové prostředí

Program jsem vytvořil v programovacím jazyce Java a vývoj probíhal v rámci prostředí IntelliJ IDEA. Vzhledem k povaze úlohy bylo možno využít pouze uvedený programovací jazyk.

4 Výsledky benchmarku, naměřené hodnoty

4.1 Podmínky testu

Benchmark byl proveden na notebooku následující HW výbavou:

- CPU: i5-3427U
- RAM: 8 GB (DDR3L)
- disk: 128 GB SSD.

Softwarová výbava zahrnuje:

- OS: Windows 10 Pro
- Java: Oracle JDK 8.

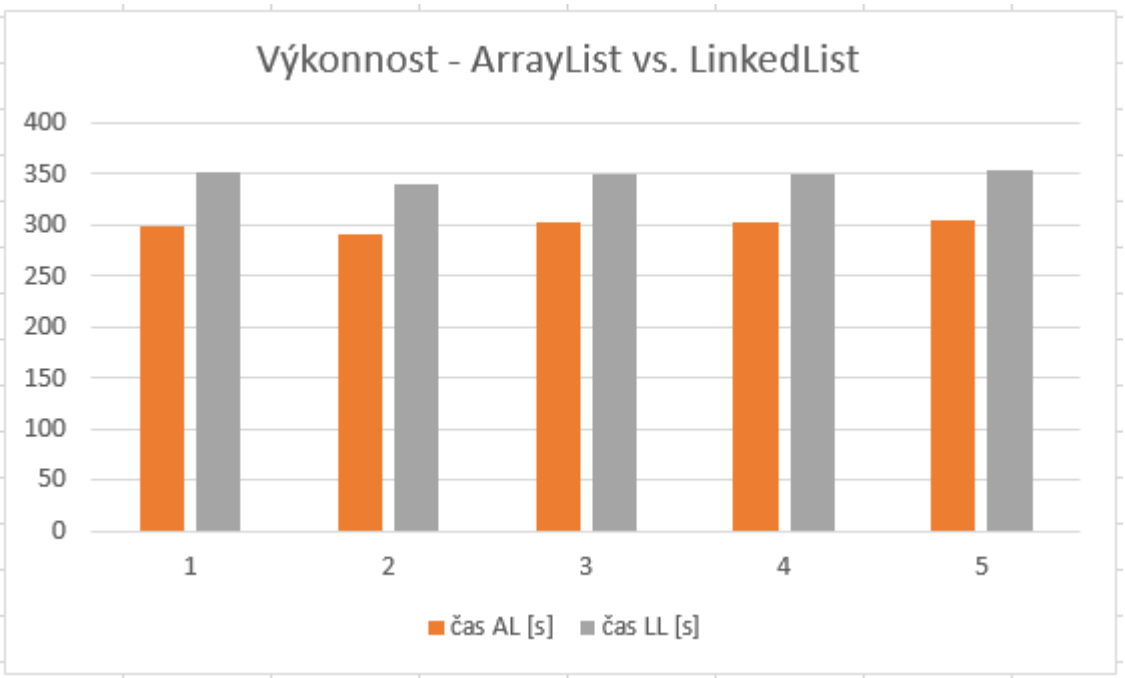
Veškeré testy byly spuštěny přímo z příkazové řádky - tím se eliminovala možnost vlivu běžícího vývojového prostředí na výsledky benchmarku.

4.2 Naměřené hodnoty benchmarku

Benchmark byl spuštěn celkem 5x, byly získány následující hodnoty:

pořadí testu	čas AL [s]	čas LL [s]
1	299	351
2	291	340
3	302	349
4	303	349
5	305	353

Pro lepší vizualizaci výsledků jsem vytvořil graf:



Je vidět, že metoda retainAll provádí operace nad ArrayListem vždy rychleji než nad objektem typu LinkedList.

4.3 Střední hodnoty, odchylky

4.3.1 Střední hodnota - ArrayList

Výpočet: sečteme časy všech benchmarků ArrayListu / počet benchmarků, tedy: $299 + 291 + 302 + 303 + 305/5 = 1500/5 = 300$.

4.3.2 Střední hodnota - LinkedList

Výpočet: sečteme časy všech benchmarků LinkedList / počet benchmarků, tedy: $351 + 340 + 349 + 349 + 353 = 1742/5 = 348,4$.

4.3.3 Odchylky od střední hodnoty

Největší odchylky jsem v případě ArrayListu dosáhl během 2. měření, kdy benchmark doběhl v čase 291 s. Uvedený čas se od střední hodnoty liší o 9 s.

V případě LinkedListu největší odchylku představuje měření č. 2, které doběhlo v čase 340 s. Od střední hodnoty se tento čas liší o 8,4 s.

4.4 Získané logy benchmarku

Logy jednotlivých spuštění benchmarku přikládám níže.

4.4.1 1. spuštění

```
***CONDITIONS - START***
total items generated (strings of length 10): 300000
items to be kept: 150000
***CONDITIONS - END***
***BENCHMARK ON ARRAYLIST - START***
start ns: 193878534825500
end ns: 194177839701400
took time: 299304875900ns => 299 seconds.
***BENCHMARK ON ARRAYLIST - END***
***BENCHMARK ON LINKEDLIST - START***
start ns: 194177840810300
end ns: 194529037329500
took time: 351196519200ns => 351 seconds.
***BENCHMARK ON LINKEDLIST - END***
```

4.4.2 2. spuštění

```
***CONDITIONS - START***
total items generated (strings of length 10): 300000
items to be kept: 150000
***CONDITIONS - END***
***BENCHMARK ON ARRAYLIST - START***
start ns: 194535653063200
end ns: 194827007353500
took time: 291354290300ns => 291 seconds.
***BENCHMARK ON ARRAYLIST - END***
***BENCHMARK ON LINKEDLIST - START***
start ns: 194827008480000
end ns: 195167911514800
took time: 340903034800ns => 340 seconds.
***BENCHMARK ON LINKEDLIST - END***
```

4.4.3 3. spuštění

```
***CONDITIONS - START***
total items generated (strings of length 10): 300000
items to be kept: 150000
***CONDITIONS - END***
***BENCHMARK ON ARRAYLIST - START***
start ns: 195198741998700
end ns: 195501228875000
took time: 302486876300ns => 302 seconds.
***BENCHMARK ON ARRAYLIST - END***
***BENCHMARK ON LINKEDLIST - START***
start ns: 195501232233800
end ns: 195850446389900
took time: 349214156100ns => 349 seconds.
***BENCHMARK ON LINKEDLIST - END***
```

4.4.4 4. spuštění

```
***CONDITIONS - START***
total items generated (strings of length 10): 300000
items to be kept: 150000
***CONDITIONS - END***
***BENCHMARK ON ARRAYLIST - START***
start ns: 195867116759400
end ns: 196170547499500
took time: 303430740100ns => 303 seconds.
***BENCHMARK ON ARRAYLIST - END***
***BENCHMARK ON LINKEDLIST - START***
start ns: 196170551151400
end ns: 196519624023400
took time: 349072872000ns => 349 seconds.
***BENCHMARK ON LINKEDLIST - END***
```

4.4.5 5. spuštění

```
***CONDITIONS - START***
total items generated (strings of length 10): 300000
items to be kept: 150000
***CONDITIONS - END***
***BENCHMARK ON ARRAYLIST - START***
start ns: 196672511445100
end ns: 196977921599800
took time: 305410154700ns => 305 seconds.
***BENCHMARK ON ARRAYLIST - END***
***BENCHMARK ON LINKEDLIST - START***
start ns: 196977924836400
end ns: 197331173266300
took time: 353248429900ns => 353 seconds.
***BENCHMARK ON LINKEDLIST - END***
```

4.5 Zhodnocení

Výsledky měření ukazují, že funkce retainAll pracuje s ArrayListem rychleji než s LinkedListem. Největší dosažená odchylka měření se pohybuje okolo 3%, nedošlo tedy k enormnímu zkreslení výsledků vnějším vlivem (spuštěný AV test atd.)

5 Uživatelská příručka

5.1 Spuštění programu

Je dodán soubor `run.bat`, který umožňuje spuštění programu na PC s Linuxem i Windows - program nepřijímá žádné parametry.

5.2 Výstup programu

Program uživatele pravidelně informuje o průběhu benchmarku. Nejprve proběhne benchmark ArrayListu, následně LinkedListu.

U každé kolekce je uživateli vypsán čas, kdy byl benchmark zahájen a čas, kdy benchmark skončil - závěrem je uživatel informován o době, kterou benchmark trval [sekundy + nanosekundy].

6 Závěr

Program byl otestován na OS Windows 10 Pro. Funguje dle očekávání a získané časy nemají příliš velký časový rozptyl.