ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

KIV/VSP

Okruh 4, zadání 5

Autor: Antonín NEUMANN, A14N0139P

E-mail: neumann@students.zcu.cz

Datum narození: 7. 9. 1989

Akademický rok: 2015/2016

Zadání

Pro vypracování se hodí informace z přednášek o benchmarkování i profilování. Pro zadaný příklad proveďte benchmarkování a výsledky popište ve zprávě. Zpráva by měla obsahovat:

- Popis prostředí ve kterém benchmarkování probíhalo - popis HW na kterém byly testy spuštěny a SW který mohl mít na výsledky testu vliv. Minimálně uveďtě základní parametry počítačové sestavy (CPU, základová deska, RAM, pevný disk), platformu (OS, verze) a případně další použité programy (verze JRE, parametry JRE, použitý překladač)
- Popis způsbu měření (použité fuknce pro měření času, jejich umístění, případně způsob kterým je vytvořena zátěž).
- Naměřené výsledky (surová data) a jejich statistické vyhodnocení (vypočtené střední hodnoty, odchylky, histogram ...).
- Analýzu případných problémů během měření (např. jak často probíhala grabage kolekce pokud se jí nepodařilo vyhnout).

Abyste testování mohli provést, musíte danou operaci provést s dostatečným počtem opakování - dávejte tedy pozor na to aby v kolekci bylo dost dat na celý běh testu a aby se neprojevily příliš různé vedlejší jevy (např. garbage kolekce, rozšiřování ArrayListu a podobně). Dostatečně velká popluace znamená něco kolem 100 000 unikátních objektů v kolekci.

Úkol 5

Porovnejte rychlost operace indexOf nad kolekcemi ArrayList a LinkedList s dostatečným množstvím dat.

Popis prostředí

Hardware

- Notebook Lenovo Thinkpad E330
- Operační paměť: 8GB DDR3 (2x4GB SODIMM, 1600 MHz)
- Procesor: Intel Core i5-3210M, 2,50 GHz, 64bit
- Pevný disk
    - Intel 530 180GB SSD bulk (SSD disk 2.5" mSATA 6Gb/s, MLC, 20nm, čtení až 540MB/s, zápis až 490MB/s, 7mm)
    - Toshiba MK5061GSY (MC102E) 500GB

Software

- Operační systém: Windows 10 Pro, 64bit
- Verze Java: 1.8.0_25
- IDE: Eclipse Java EE IDE for Web Developers.
    - Version: Luna Service Release 1a (4.4.1); Build id: 20150109-0600

Popis bechmarku

V tomto testu jsem porovnával rychlost operace indexOf() nad implementacemi rozhraní List v programovacím jazyce Java. Jednotlivé porovnávané implementace jsou ArrayList, LinkedList a Vector.

Příprava

Nejdříve v programu vygeneruji několik řetězců různé délky, které uložím do ArrayListu. Lze ovlivnit proměnnými WORDS_NUM, SHORTEST_WORD, LONGEST_WORD, implicitně nastavenými na hodnoty 100000, 3 a 12. Poté náhodně zaměním 20 předem připravených "známých" slov (lze přidat další přepínačem -c a to za předem připravená slova), které poté budu metodou indexOf hledat. Nakonec převedu ArrayList do ostatním implementací, tedy do LinkedListu a Vectoru.

Vlastní měření

Měření probýhá vždy nad všemi třemi implementacemi Listu a to ve 100 (lze ovlivnit proměnnou LOOP) opakováních a vždy se hledají všechna "známá" slova (minimálně tedy základních 20). Celý tento proces spouštím celkem dvakrát, přičemž jednou měřím čas metodou System.currentTimeMillis() a podruhé metodou System.nanoTime().

Měřím vždy rozdíl času (tedy dobu trvání) potřebnout pro nalezení všech "známých" slov v jedné implementaci listu. Tento rozdíl si ukládám do pole a z těchto hodnot nakonec vypočítám průměrnou dobu potřebnout pro vyhledání.

Spuštění

Export celého projektu byl proveden v programu Eclipse do JAR včetně všech potřebných knihoven (Commons CLI a Commons Lang3). Při spuštení programu s přepínačem --help nebo -h se vypíše nápověda, ve které jsou popsané všechny dostupné přepínače a jejich význam.

Přepínače JVM `-verbose:gc -Xms1024m` slouží pro výpis práce garbage collectoru.

První přepínač způsobí výpis na konzoli v případě, že dojde k aktivování garbage collectoru, nebo můžeme pro více detailů použít přepínač `-XX:+PrintGCDetails`. Druhý přepínačem nastavíme JVM minimální množství dostupné paměti na 1024 MB (podlě všeho by to ale od Java 8 nemělo být zapotřebí).

Naměřené výsledky

Kompletní výstup programu se zapnutým přepínačem --verbose a parametry pro JVM -verbose:gc -XX:+PrintGCDetails -Xms1024m (výstup z vývojového prostředí Eclipse).

```
================================================
Benchmark properties:
================================================
Tested: ArrayList, LinkedList, Vector
Number of words: 1000000
Shortest word: 3, longest word: 12
Number of words searches with indexOf: 20
Number of loop for time measurement: 100
================================================
Specific words for testing indexOf() [20]:
  - Tonda
  - Pavel
  - Petr
  - Růžena
  - Karel
  - Praha
  - Londýn
  - Paříž
  - New Yourk
  - Bratislava
  - Mazda
  - Renault
  - Škoda
  - Peugeot
  - Lamborgini
  - Chleba
  - Šunka
  - Okurka
  - Zmrzlina
  - Hamburger

Lenght stats of generated words:
  3: 100179
  4: 100127
  5: 99328
  6: 100076
  7: 99914
  8: 100017
  9: 100012
  10: 100272
  11: 100155
  12: 99920

================================================
Time measurement by System.currentTimeMillis():
================================================
ArrayList: [70, 60, 60, 60, 60, 63, 57, 57, 56, 60, 60, 60, 58, 55, 56, 65, 60, 58, 60,
57, 55, 65, 65, 60, 60, 60, 60, 55, 58, 59, 55, 56, 60, 56, 60, 55, 60, 67, 60, 60, 60,
60, 60, 55, 60, 60, 55, 60, 68, 61, 60, 60, 60, 55, 60, 63, 64, 61, 62, 63, 60, 57, 60,
67, 61, 55, 55, 55, 60, 62, 53, 56, 55, 55, 60, 62, 60, 55, 60, 60, 60, 58, 59, 59, 55,
55, 55, 55, 64, 58, 55, 59, 60, 60, 60, 60, 55, 59, 60, 57]
LinkedList: [75, 66, 64, 65, 71, 65, 65, 66, 70, 62, 73, 71, 66, 70, 70, 65, 70, 70,
65, 70, 72, 69, 65, 65, 65, 67, 74, 65, 70, 65, 70, 70, 65, 72, 65, 70, 65, 64, 70, 65,
65, 65, 72, 72, 65, 65, 70, 70, 70, 70, 70, 66, 65, 75, 75, 70, 71, 70, 74, 70, 65, 70,
65, 70, 64, 70, 65, 65, 65, 67, 65, 65, 65, 65, 65, 68, 70, 65, 65, 65, 72, 66, 67, 65,
65, 65, 65, 70, 70, 66, 65, 66, 65, 65, 65, 65, 70, 64, 65, 65]
```

```
Vector: [67, 60, 60, 58, 60, 60, 60, 55, 55, 65, 60, 56, 60, 55, 70, 68, 60, 55, 60,
60, 59, 55, 56, 60, 57, 60, 60, 60, 55, 60, 60, 63, 65, 55, 60, 57, 62, 60, 55, 60, 55,
60, 57, 58, 55, 60, 55, 55, 60, 54, 55, 60, 60, 60, 86, 60, 61, 62, 62, 60, 55, 55, 55,
68, 60, 55, 60, 55, 59, 57, 60, 60, 58, 55, 60, 59, 55, 55, 55, 58, 58, 58, 57, 58, 60,
66, 56, 59, 57, 57, 55, 55, 55, 55, 55, 55, 64, 55, 55, 55]
=== Average ===
ArrayList: 59 milliseconds
LinkedList: 67 milliseconds
Vector: 58 milliseconds

================================================
Time measurement by System.nanoTime():
================================================
ArrayList: [58118531, 58754440, 62876968, 59988078, 57675161, 58610755, 58305321,
58202689, 58985567, 58808630, 57928456, 58918651, 59103799, 65343013, 59018820,
58079531, 58322154, 57946520, 58120173, 57192379, 56526501, 57016262, 57270379,
59590276, 60267648, 60575544, 59436738, 61839562, 61905247, 60274217, 57752340,
56916503, 57671465, 56589722, 57100831, 58341859, 58589407, 56914040, 58054489,
56383227, 56608196, 56654996, 56372143, 56957967, 56701387, 57377117, 65508456,
58031499, 57892330, 56510901, 56921019, 57310200, 57198126, 56509259, 58446954,
58036836, 58736787, 60125195, 63363855, 59623940, 57563086, 57454706, 56544975,
58235121, 57993731, 58020005, 58251542, 56912809, 60987715, 58340217, 58212132,
57429664, 58668640, 59032367, 56206288, 57835676, 56403753, 57368495, 57720729,
60871125, 58114425, 58371416, 56502691, 56973567, 58386196, 58164510, 65714952,
85618198, 59187958, 58042172, 59864509, 59478612, 57666128, 58729808, 58936714,
73512122, 58357049, 63964048, 60450333, 59296337]
LinkedList: [67471604, 65031422, 70172471, 65980154, 66647263, 66080733, 66141080,
67581625, 66464577, 67239655, 66478535, 66819684, 66386166, 67539341, 65569214,
65571267, 66466219, 65699352, 72551074, 66511788, 65458371, 64387713, 66005196,
66252744, 67020433, 67870638, 68145692, 69516446, 67483919, 73999008, 66924780,
64211596, 64677957, 64607346, 64247722, 65344245, 65548688, 65338907, 65650499,
64616378, 67307803, 65657068, 65325360, 64373345, 65203433, 66408745, 64628283,
64920579, 64627052, 65358203, 65357792, 70415504, 65076170, 64835600, 66224007,
66006838, 66669842, 66136975, 66159554, 65010485, 66290924, 64244439, 73558101,
64988316, 66239197, 66056101, 65579888, 65039633, 64593388, 65063443, 65009664,
66256440, 65265423, 66789715, 64170133, 66066775, 64921811, 63799836, 64853253,
64719010, 64729273, 64284260, 64530167, 64132774, 71517774, 67352140, 65372982,
69500436, 66139439, 67747069, 67947817, 66686673, 66608673, 66439945, 66643157,
71081792, 70433568, 76695360, 69209371, 75983503]
Vector: [61086653, 56390617, 58244564, 57867287, 58072141, 57504380, 57730582,
57281053, 58337753, 58162868, 58162868, 57831982, 56014162, 58001120, 58489239,
57806940, 57381222, 57428022, 57182116, 57212495, 56554828, 56132394, 57437875,
57338937, 60005731, 59737656, 59986847, 59596024, 60468396, 58130437, 56314258,
57010925, 57326211, 55987888, 63154075, 56647607, 60295564, 55998151, 56900903,
57257653, 56578227, 56964125, 56076973, 56483395, 57132442, 59208484, 57526960,
56932925, 56587260, 57563497, 57146810, 56597112, 57703487, 58592692, 57391895,
57227684, 60319375, 65519130, 61396192, 57464559, 58765114, 56960841, 58180111,
57994962, 57617276, 57601675, 56645965, 64023985, 56840966, 57756855, 57026525,
57400516, 57716213, 56622565, 57718266, 56482986, 56405395, 55587623, 60986895,
55595422, 56880377, 55815055, 56523217, 57778613, 58315995, 65600825, 57746593,
59029905, 56903367, 60042678, 57088515, 56967820, 58584892, 58146037, 61195853,
58765935, 58454343, 62446734, 57327853, 71129003]
=== Average ===
ArrayList: 58995875 milliseconds
LinkedList: 66579558 milliseconds
Vector: 58175574 milliseconds
================================================
```

Závěr

Podle výsledků uvedených výše se zdá, že rozdíl mezi měřeními pomocí metod System.currentTimeMillis() a System.nanoTime() jsou, alespoň v mém případě, zanedbadelné.

U jednotlivých implementací listů už nějaké rozdíly pozorovat lze. Jako nejpomalejší vychází implementace LinkedList (tato doměnka byla potvrzena použitím více tzv. “známých” slov - celkem 60, kde už byl rozdíl snadněji rozeznatelný - ArrayList 179 ms, LinkedList 210 ms, Vector 179 ms). Ovšem u implementací ArrayList a Vector proběhnou operace téměř ve stejném čase.

Po přezkoumání implementace ArrayList a Vector, je zřejmé že výsledky musí být stejné, jelikož obě implementace používají pro uchování prvků pole a implementace metody indexOf() je u obou totožná.

Zdroje

- http://developer.classpath.org/doc/java/util/ArrayList-source.html
- http://developer.classpath.org/doc/java/util/Vector-source.html
- http://developer.classpath.org/doc/java/util/LinkedList-source.html
- http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html#DebuggingOptions