Java Profiling

Hibakeresés és kód optimalizálás futási statisztika alapján

zoltan.kiss@loxon.eu
Kiss Zoltán
chief software architect

Budapest, 2017.09.26.

Bemelegítés

Quiz – 10 kérdés

Elmélet

Működés, képességek, eszközök

Elmélet - működés

- Modes
  - attach (1.6+)
  - startup run/prepare (-agentpath:/path/to/native/lib)
  - offline
- What to profile
  - Time (execution paths, method level)
  - Memory (heap, allocation, GC)
  - Thread (synchronization issues, locks, monitors)
  - Probes (higher level analysis – I/O, net, jdbc, etc.)
- How to collect statistics
  - via JVM interface (<= 1.5 – JVMPI, >=1.5 - JVMTI)
    - JVMTI events: JVM, class, thread, object, monitor, GC + incremental heap dump
  - via instrumented classes
  - via JMX (Java Management Extension)
  - via native JDK service (FR)
- How to start
  - -agentpath:[path to jprofilerti library]
  - Attach API (1.6+)
  - Custom JVM switches (-XX)

Elmélet - képességek

- Milyen problémákra
  - Magas CPU használat
  - Magas memória használat
  - Lassú futás java ill. kapcsolódó rendszer oldalán (DB, MQ, stb.)
  - Sok GC
  - OOM / memory leak (heap, class)
  - Deadlock
  - Alacsony párhuzamosság
- Mit vizsgálhatunk
  - Memória (GC, allocation)
  - CPU (hotspot, hivási lánc)
  - Thread (párhuzamosság)
  - Monitor (lock-ok)
  - Probes (magasabb szintű elemzések)
  - Snapshot összehasonlítás (trendek)
- Milyen áron
  - Overhead (mem / cpu)
  - JVM crash

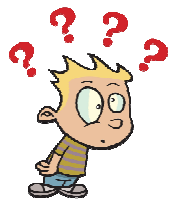
Elmélet - eszközök

- Top 3
  - Java Visual VM (JVVM, 1.6+, free, JDK built-in)
    - http://docs.oracle.com/javase/8/docs/technotes/guides/visualvm/intro.html
    - Local (full profiling – thread dump, heap dump, monitors, threads, cpu profiling, memory profiling, stats)
    - Remote (jstatd, no profiling only stats)
    - Process core dumps (linux, solaris – local machine only)
    - Process snapshots (profiling, application)
  - Jprofiler (10 days trial, commercial, v10.0.3, EJ Technologies)
    - Native JNI agent (local/remote full profiling)
    - Statup (run/prepare) mode (+ offline, attach)
    - probes
  - Java Mission Control / Java Flight Recorder (PROD non-free, BCL license, JDK built-in 1.7u40)
    - https://docs.oracle.com/javacomponents/jmc-5-5/jmc-user-guide/toc.htm
    - also for PROD, < 2% impact (thread > global buffers, optional disk snapshot or circular global buffer)
    - Non-intrusive (non-JVMTI) -> FlightRecorder native JVM feature for offline processing
    - Local discovery, JMX (agent URL – agent - client), JDP (Java Discovery Protocol – remote JDP server – JDP discovery client)
    - Jmc.exe > JVM brower, JMX console, JFR
    - -XX:+UnlockCommercialFeatures -XX:+FlightRecorder vs. jcmd
- Others
  - YourKit (fizetős), Netbeans Profiler (GNU, bundled), Jprobe (apache license, servlet filter), etc.

Gyakorlat

szünet után

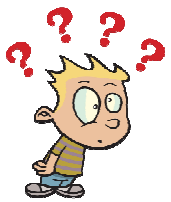
Gyakorlat – JVVM



- Probléma
  - „Tegnap végre megint gyors volt az alkalmazásom. De aztán kitettem a teszt környezetbe és teljesen belassult. Az világos, hogy több adat van a teszt környezetben, de hogy fogok rájönni, hogy melyik adat lassítja be igazán a működést? Vagy minden része lassabb kicsit és ez adódik össze?”

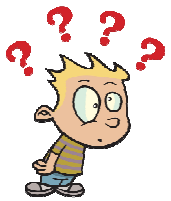
- Megoldás
  - JVVM
    - jvisualvm.exe -J-Dorg.netbeans.profiler.separateConsole=true
    - preset: JavaIdeaLab
    - sampling először (cpu, memory)
      - Memória használat vizsgálata
      - Problémás metódus azonosítása
      - Javítási ötletek, kipróbálásuk

Gyakorlat – JVVM - eredmények



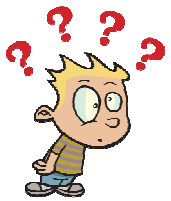
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)

Gyakorlat – JVVM - eredmények



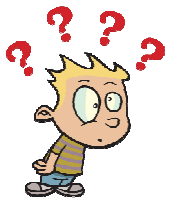
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00

Gyakorlat – JVVM - eredmények



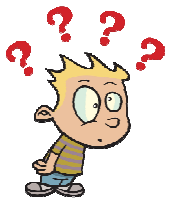
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00

Gyakorlat – JVVM - eredmények



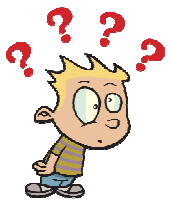
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17

Gyakorlat – JVVM - eredmények



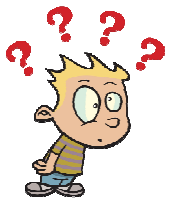
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43

Gyakorlat – JVVM - eredmények



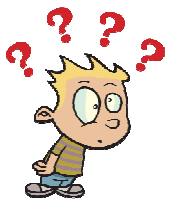
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09

Gyakorlat – JVVM - eredmények



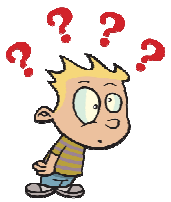
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09
Előző + kevesebb logolás	100 000	0,13	12820,51	139,56	4,23

Gyakorlat – JVVM - eredmények



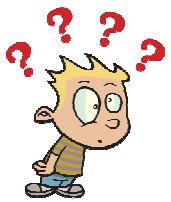
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09
Előző + kevesebb logolás	100 000	0,13	12820,51	139,56	4,23
Előző + kevesebb logolás	100 000	0,02	83333,33	907,14	6,50

Gyakorlat – JVVM - eredmények



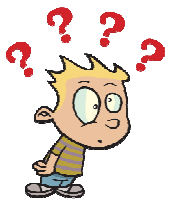
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09
Előző + kevesebb logolás	100 000	0,13	12820,51	139,56	4,23
Előző + kevesebb logolás	100 000	0,02	83333,33	907,14	6,50
Előző + több adat	140 000	0,04	66666,67	725,71	0,80

Gyakorlat – JVVM - eredmények



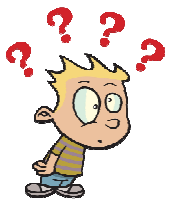
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09
Előző + kevesebb logolás	100 000	0,13	12820,51	139,56	4,23
Előző + kevesebb logolás	100 000	0,02	83333,33	907,14	6,50
Előző + több adat	140 000	0,04	66666,67	725,71	0,80
Előző + több adat	185 000	0,05	61666,67	671,29	0,93

Gyakorlat – JVVM - eredmények



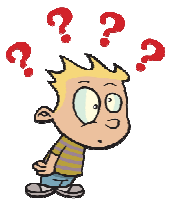
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	10	0,00	91,86	1,00	0,00
Előző + több adat	7 000	1,27	91,86	1,00	1,00
String összefűzés javítva + azonos adat	7 000	0,06	1944,44	21,17	21,17
Előző + több adat	30 000	0,18	2777,78	30,24	1,43
Előző + több adat	100 000	0,55	3030,30	32,99	1,09
Előző + kevesebb logolás	100 000	0,13	12820,51	139,56	4,23
Előző + kevesebb logolás	100 000	0,02	83333,33	907,14	6,50
Előző + több adat	140 000	0,04	66666,67	725,71	0,80
Előző + több adat	185 000	0,05	61666,67	671,29	0,93
Előző + több adat -> OOM	200 000				

Gyakorlat – JProfiler



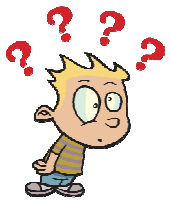
- Probléma
  - „Az alkalmazásom csak akkor gyors ha nagy heap size-ot állítok be, pedig alig tárol a memóriában valamit. Így nem tudok sok process-t indítani belőle, mert vagy swappel az operációs rendszer vagy out of memory-t dob az alkalmazás. Mi lehet az oka a lassulásnak, amikor leviszem a max heap size-ot?”
  - „Az alkalmazásom nem végez számításokat mégis magas a CPU használata. Nem értem, mi lehet a lassú? Biztos nem az én kódom okozza, hanem a 3rd party library a hibás. Na de hogyan bizonyítsam?”
- Megoldás
  - JProfiler
    - Sampling
    - Trendek vizsgálata, összevetése
    - Iteratív módon javítások

Gyakorlat – Jprofiler - eredmények



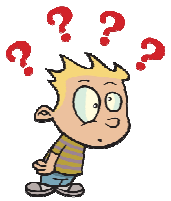
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)

Gyakorlat – Jprofiler - eredmények



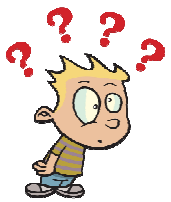
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00

Gyakorlat – Jprofiler - eredmények



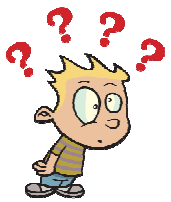
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00

Gyakorlat – Jprofiler - eredmények



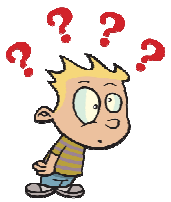
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33

Gyakorlat – Jprofiler - eredmények



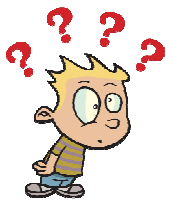
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00

Gyakorlat – Jprofiler - eredmények



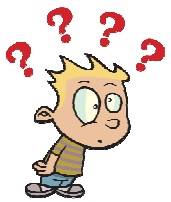
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00
Előző + nagyobb adat -> OOM	500 000	2,57	3242,54	0,55	1,67

Gyakorlat – Jprofiler - eredmények



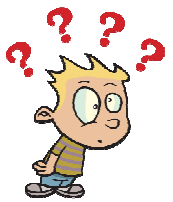
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00
Előző + nagyobb adat -> OOM	500 000	2,57	3242,54	0,55	1,67
Input 1st level cache-ben ne legyen + előző adat	500 000	4,95	1683,50	0,29	0,52

Gyakorlat – Jprofiler - eredmények



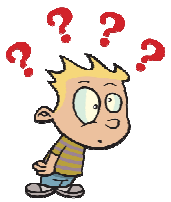
Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00
Előző + nagyobb adat -> OOM	500 000	2,57	3242,54	0,55	1,67
Input 1st level cache-ben ne legyen + előző adat	500 000	4,95	1683,50	0,29	0,52
getCityFor cache-ből + előző adat	500 000	1,17	7122,51	1,22	4,23

Gyakorlat – Jprofiler - eredmények



Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00
Előző + nagyobb adat -> OOM	500 000	2,57	3242,54	0,55	1,67
Input 1st level cache-ben ne legyen + előző adat	500 000	4,95	1683,50	0,29	0,52
getCityFor cache-ből + előző adat	500 000	1,17	7122,51	1,22	4,23
commit, batch size növelés + előző adat	500 000	0,84	9920,63	1,70	1,39

Gyakorlat – Jprofiler - eredmények



Teszteset / változtatás jellege	Input sorok (db)	Futásidő (perc)	Sebesség (sor / sec)	Gyorsulás (x eredetihez)	Gyorsulás (x előzőhöz)
Induló nem optimalizált kód, teszt adatmennyiség	50	0,00	5847,95	1,00	0,00
Előző + nagyobb adatmennyiség	200 000	0,57	5847,95	1,00	1,00
Előző + nagyobb adatmennyiség -> OOM	300 000	2,57	1945,53	0,33	0,33
Periodikus 1st level cache ürítés + előző adat	300 000	2,57	1945,53	0,33	1,00
Előző + nagyobb adat -> OOM	500 000	2,57	3242,54	0,55	1,67
Input 1st level cache-ben ne legyen + előző adat	500 000	4,95	1683,50	0,29	0,52
getCityFor cache-ből + előző adat	500 000	1,17	7122,51	1,22	4,23
commit, batch size növelés + előző adat	500 000	0,84	9920,63	1,70	1,39
sequence nextval csökkentés + előző adat	500 000	0,16	52083,33	8,91	5,25

Kérdések?

Köszönöm a figyelmet!