Java Performance Tuning

Ing. Jiří Pinkas

Twitter: @jirkapinkas

Github: https://github.com/jirkapinkas

Nástroje pro load testing

- Soap UI (pro základní scénáře stačí i open-source edice, ale pro cokoli pokročilejšího je zapotřebí placená verze):
  - https://www.soapui.org/load-testing/creating-and-running-loadtests.html
- Apache JMeter
  - https://jmeter.apache.org/
  - https://www.seleniumeasy.com/jmeter-tutorials/http-request-sampler-example
  - https://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html
- Gatling
  - https://gatling.io/

Nástroje pro profiling

- Zdarma
  - JMC (Java Mission Control) + FlightRecorder – v Oracle JDK zdarma pouze pro dev & test, v OpenJDK zdarma i na produkci (v současnosti jsou k dispozici pouze SNAPSHOT verze na https://adoptopenjdk.net/jmc.html)
  - Visual VM
    - https://visualvm.github.io/
- Placené
  - JProfiler
    - https://www.ej-technologies.com/products/jprofiler/overview.html
  - XRebel, YourKit, …

Nástroje pro monitoring

- Java Melody
  - https://github.com/javamelody/javamelody/wiki
- Glowroot
  - https://glowroot.org/
- Kibana APM
  - https://www.elastic.co/products/apm
- NewRelic (pouze v cloudu & placené)
  - https://newrelic.com/
- AppDynamics (placené)
  - https://www.appdynamics.com/
- Dynatrace (placené)
  - https://www.dynatrace.com/
- Prometheus, Zabbix, …

Spring Boot Actuator

- Spring Boot Actuator

  management.endpoints.web.exposure.include=*

  management.endpoint.health.show-details=always

- Spring Boot Admin
  - https://github.com/codecentric/spring-boot-admin

Microservice monitoring

- Zipkin:
  - https://github.com/openzipkin/zipkin
- Jaeger:
  - https://www.jaegertracing.io/
- Kibana APM:
  - https://www.elastic.co/products/apm

Garbage Collector I.

- Od Java 9 (a použitelné od Java 7u4) je výchozím GC G1 (G1 = Garbage First)
  - Od Java 8u20 také obsahuje String Deduplication funkcionalitu – G1 při svém běhu hledá duplicitní Stringy a odstraňuje je z paměti.

    -XX:+UseStringDeduplication
  - https://blog.codecentric.de/en/2014/08/string-deduplication-new-feature-java-8-update-20-2/

Garbage Collector II.

- Low-latency GC pro velké heapy:
  - ZGC (od Java 11, aktuálně experimentální):
    - https://www.opsian.com/blog/javas-new-zgc-is-very-exciting/
  - Shenandoah GC:
    - https://wiki.openjdk.java.net/display/shenandoah/Main

Analýza GC – low level

- Analýzu je možné provést pomocí hromady profilovacích nástrojů (například JMC + FlightRecorder), ale je také možné logovat volání GC tímto způsobem:

  -XX:-PrintGC -XX:-PrintGCDetails -Xloggc:gc.log

- Logy budou uložené do souboru gc.log. Pro jejich analýzu je výborný tento web:
  - http://gceasy.io/

Analýza paměti

- Eclipse Memory Analyzer (MAT):
  - https://www.eclipse.org/mat/
- JProfiler

Microbenchmarking

- Testování rychlosti nějakého kódu není tak jednoduchá záležitost jak by si člověk mohl myslet, JVM má řadu optimalizací, díky kterým je složité navrhnout správně tzv. microbenchmark. Jak na to správně? Pomocí JMH:
  - http://tutorials.jenkov.com/java-performance/jmh.html
    - Praktický příklad:
      - https://github.com/jirkapinkas/java-mapping-frameworks

Tipy a triky I.

- Příliš moc logování může být problém (to ale neznamená nelogovat vůbec) … přijděte na přednášku Petra Adámka abyste se dozvěděli jak na to ;-) Také je v současnosti nejrychlejší Log4j2:
  - https://logging.apache.org/log4j/2.x/performance.html
    - Log4j2 má no-gc (pro Java SE) a low-gc (pro web. aplikace) logování:
      - https://logging.apache.org/log4j/2.x/manual/garbagefree.html
- Ultra pomalé:
  - Vyhazování výjimek (musí se vytvořit celý stacktrace)
  - String.replace (do Java 8 včetně), String.split (výrazně výkonnější je Splitter v Guava)
  - Spojování Stringů v cyklu
  - Regulární výrazy
  - Reflexe
  - Dozer, Orika, nejlepší je MapStruct:
    - https://github.com/jirkapinkas/java-mapping-frameworks
    - https://github.com/arey/java-object-mapper-benchmark

Tipy a triky II.

- Jackson AfterBurner:
  - https://github.com/FasterXML/jackson-modules-base/tree/master/afterburner
  - https://stackoverflow.com/questions/49454849/how-can-i-register-and-use-the-jackson-afterburnermodule-in-spring-boot
- Pozor na CORS na produkci:
  - https://www.freecodecamp.org/news/the-terrible-performance-cost-of-cors-api-on-the-single-page-application-spa-6fcf71e50147/
- HikariCP je nejlepší connection pool (výchozí CP v Spring Boot 2):
  - https://github.com/brettwooldridge/HikariCP
- Jackson nebo GSON? To záleží …
  - https://blog.overops.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

Tipy a triky III.

- Hibernate:
  - Pokud vás opravdu zajímá performance, pak nepoužívejte Hibernate.
  - Nepoužívejte EAGER fetching, místo toho používejte JOIN FETCH.
  - Pozor na exotické vazby: @OneToOne, @ManyToMany
  - Když chcete jenom zavolat SELECT, pak byste neměli používat entity, ale pomocí „select new“ syntaxe byste měli rovnou získat data do DTOček (ale to je tolik práce, že to asi nikdo nedělá …):
  - Je možné rychlit selecty a další operace nastavením „batch size“ (lze nastavit globálně, na entitách, anebo na vazbách).
  - https://www.amazon.com/High-Performance-Java-Persistence-Vlad-Mihalcea/dp/973022823X

Tipy a triky IV.

- SQL:
  - Pro batch operace je nejlepší jdbc batch:
    - http://tutorials.jenkov.com/jdbc/batchupdate.html
  - Je nejlepší vybírat pouze ty data, která skutečně potřebujete a nic víc. Bohužel entity příliš zjednodušují pravý opak.
  - Use the index Luke ☺ (Jak používat správně indexy a další věci):
    - https://use-the-index-luke.com/
      - Také obsahuje hezké vysvětlení proč není dobrý nápad používat pro stránkování věci jako LIMIT & OFFSET
  - Explain plan je výborná věc:
    - https://www.youtube.com/watch?v=-IWxdI9-Z-U

Tipy a triky V. – Novinky v Java 8

- Java 8 Streams jsou pomalejší než imperativní způsob programování.
  - Pozor na autoboxing (to platí obecně) ☺
  - Pozor na parallel stream – ve výchozím nastavení parallel stream používá jeden pool vláken (per instance JVM), ve kterém je počet vláken = „počet jader Vašeho procesoru – 1“.
- Pořadí operací ve streamu je významné, filtry patří na začátek.
- Nové java.time je je rychlejší než staré (rychlejší i než Joda time)
- Lambdy jsou stejně rychlé jako anonymní třídy.

Ideal Graph Vizualizer

- Ideal Graph Vizualizer (děkuji za info Jardovi Tulachovi):
  - https://www.graalvm.org/docs/reference-manual/tools/#ideal-graph-visualizer
- Ukazuje různé fáze optimalizace překladu z bytecode do nativního kódu.

Děkuji za pozornost.

www.**JavaDays**.cz

www.**gopas**.cz