HOME CREDIT IT CENTRE

Optimalizace a ladění výkonu

Petr Adámek
Home Credit International a.s.

HOME CREDIT IT CENTRE

Osnova

- **Optimalizace**
  - První pravidlo optimalizace
  - Druhé pravidlo optimalizace
- **Princip práce JVM**
  - Vykonávání kódu
  - Správa paměti
- **Nástroje pro správu a ladění**

HOME CREDIT IT CENTRE

První pravidlo optimalizace

Nejdůležitější pravidlo optimalizace zní:

Neoptimalizovat !

Předčasná optimalizace je ZLO!

- Dopředu téměř nikdy nejsme schopni odhalit skutečná úzká místa, která způsobují výkonnostní problémy.
- Pokud optimalizujeme brzo, optimalizujeme zbytečně.
- Naším cílem by měl být vždy jednoduchý a přehledný kód (viz pravidlo KISS).
- Takový kód se snadno udržuje a snadno optimalizuje.

Optimalizujeme až teprve tehdy, kdy je to nutné a kdy máme identifikovaná úzká místa v programu.

HOME CREDIT IT CENTRE

Příklad

Mějme v aplikaci dvě metody

- Metoda a(), ve které program tráví 2 % času
- Metoda b(), ve které program tráví 30 % času
- Metodu a() je možné zrychlit 1000x
- Metodu b() je možně zrychlit o 20 procent (1,2x)

Jaký vliv bude mít optimalizace těchto metod na celý program?

- Počítejme úsporu za jednu sekundu
- Metodu a() zrychlíme z 20 ms na 20 µs, takže ušetříme 19,98 ms.
- Metodu b() zrychlíme z 300 ms na 240 ms, takže ušetříme 60 ms.

Jak ale zjistit, kde jsou úzká místa a kolik času spotřebují jednotlivé metody?

Druhé pravidlo optimalizace

Druhé pravidlo optimalizace zní:

Neoptimalizovat bez profileru!

Proč profiler?

- Profiler je nástroj, který nám umožňuje sledovat spotřebu paměti a strojového času jednotlivými částmi programu.
- Chod programu bývá ovlivněn tolika faktory, že bez použití profileru nejsme schopni identifikovat úzká místa programu.

Spoléhání se na vlastní úsudek je v tomto případě spolehlivá cesta do míst, kde se nám rozhodně nebude líbit.

HOME CREDIT IT CENTRE

Jak optimalizovat

Při optimalizaci

- Neoptimalizujeme dříve, než je to nutné
- Pro identifikaci úzkých míst používáme vhodné nástroje (např. profiler)

Jak se to dělá v praxi

- Součástí specifikace by měly být také nefunkční požadavky definující požadavky na výkon (latence, propustnost, paměťové nároky)
- Tyto parametry je nutné během vývoje sledovat a provádět zátěžové testy
- Zátěžové testy i optimalizace by měly probíhat ve stejném prostředí, jako produkční systém

HOME CREDIT IT CENTRE

Princip JVM

JVM

- Virtuální počítač pracující se zásobníkem, s vlastní instrukční sadou a správou paměti
- javac kompiluje zdrojové kódy do bytekódu, což je strojový kód JVM
- Automatická správa paměti (Garbage Collector)

Způsob zpracování bytekódu

- Interpretace
- JIT (just-in-time)
- HotSpot

Interpretace

- Pomalé
- Staré verze JVM
- Dnes možno aktivovat parametrem -Xint

HOME CREDIT IT CENTRE

Zpracování bytekódu

JIT (Just-in-time)

- Ve chvíli, kdy se bytekód provádí, je přeložen do strojového nativního kódu (bez ukládání do cache)
- Pouze jednoduché optimalizace (překlad musí být rychlý)

HotSpot

- Adaptivní optimalizace
- Často používané části kódu jsou překládány s vysokou mírou optimalizace

Překlad do nativního kódu

- Výsledkem je spustitelný soubor (.exe, elf)
- Nepoužívá se, není rychlejší než moderní HotSpot JVM

HOME CREDIT IT CENTRE

Správa paměti

Garbage collector

- Pravidelně prochází paměťový prostor a hledá nepoužívané (tj. neodkazované objekty)
- Stop-The-World × paralelně pracující
- Možno zavolat explicitně (System.gc())
  - Nepoužívat, automatické volání je efektivnější!
- Rozdělení paměti podle generací

HOME CREDIT IT CENTRE

Výhody a nevýhody Garbage Collectoru

Výhody

- Eliminace chyb v práci s ukazatelovou aritmetikou (např. neplatné odkazy)
- Redukce úniků paměti (memory leak)
- Eliminace problémů s fragmentaci paměti

Nevýhody

- Garbage collector spotřebovává strojový čas
- Vyšší spotřeba paměti

HOME CREDIT IT CENTRE

Úniky paměti

Může v Javě dojít k úniku paměti (memory leak)?

- Může, ale v menší množině případů, než např. v C/C++

Kdy může dojít k úniku paměti?

- Pokud někde stále držíme odkaz na objekt, který už nepotřebujeme.
- Typycké např. u různých vyrovnávacích pamětí
- Řešením je WeakReference (viz balík java.lang.ref)

HOME CREDIT IT CENTRE

Nástroje

Profiler

- NetBeans profiler
- Řada dalších komerčních nástrojů

Sledování běžící JVM

- Java Console (JMX)
- DTrace (pouze Solaris)

Analyzátory paměti

- HAP
- PMAT

Zátěžové testy

- JMeter

Závěr

?