Protokol pro výstup profilingu

KALKULAČKA v2.3

Obsah

1 Úvod

Cílem profilingu je určit, kde program traví nejvíce času, pomocí výpočtu výběrové směrodatné odchylky s vlastními funkcemi z matematické knihovny MathLib.java.

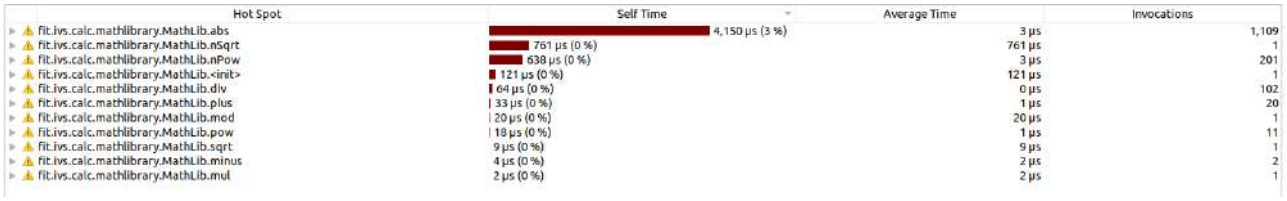
$$s = \sqrt{\frac{1}{N-1}(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2)}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2 Testování

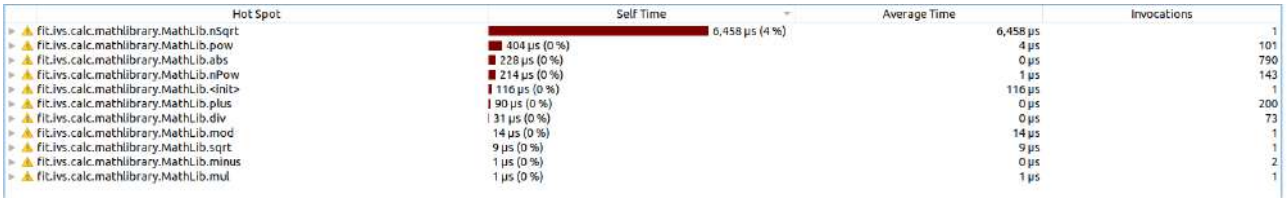
Profiling byl testován nad náhodně vygenerovanými čísly na intervalu -1 000 000 až 1 000 000 o počtu 10, 100 a 1000, jenž následně byly zapsány do textového souboru, který pak je vložený na standartní vstup. Pro profiling byl vybrán JProfiler který vygeneroval .html soubory s výsledky.

3 Výsledky



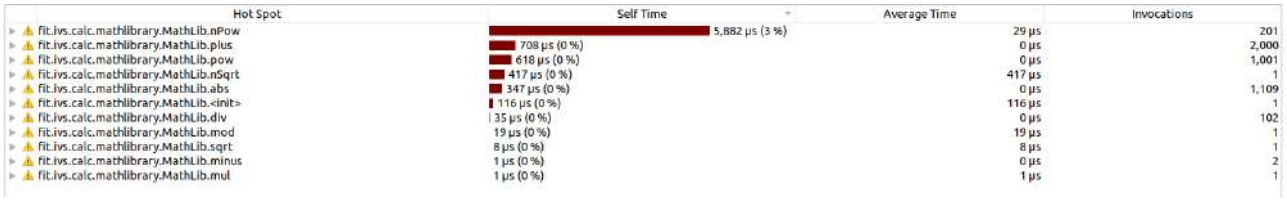
Hot Spot	Self Time	Average Time	Invocations
fit.ivs.calc.mathlibrary.MathLib.abs	4,150 µs (3 %)	3 µs	1,109
fit.ivs.calc.mathlibrary.MathLib.nSqrt	761 µs (0 %)	761 µs	1
fit.ivs.calc.mathlibrary.MathLib.nPow	638 µs (0 %)	3 µs	201
fit.ivs.calc.mathlibrary.MathLib.<init>	121 µs (0 %)	121 µs	1
fit.ivs.calc.mathlibrary.MathLib.div	64 µs (0 %)	0 µs	102
fit.ivs.calc.mathlibrary.MathLib.plus	33 µs (0 %)	1 µs	20
fit.ivs.calc.mathlibrary.MathLib.mod	20 µs (0 %)	20 µs	1
fit.ivs.calc.mathlibrary.MathLib.pow	18 µs (0 %)	1 µs	11
fit.ivs.calc.mathlibrary.MathLib.sqrt	9 µs (0 %)	9 µs	1
fit.ivs.calc.mathlibrary.MathLib.minus	4 µs (0 %)	2 µs	2
fit.ivs.calc.mathlibrary.MathLib.mul	2 µs (0 %)	2 µs	1

Obrázek 1: Profiling s 10 čísly



Hot Spot	Self Time	Average Time	Invocations
fit.ivs.calc.mathlibrary.MathLib.nSqrt	6,458 µs (4 %)	6,458 µs	1
fit.ivs.calc.mathlibrary.MathLib.pow	404 µs (0 %)	4 µs	101
fit.ivs.calc.mathlibrary.MathLib.abs	228 µs (0 %)	0 µs	790
fit.ivs.calc.mathlibrary.MathLib.nPow	214 µs (0 %)	1 µs	143
fit.ivs.calc.mathlibrary.MathLib.<init>	116 µs (0 %)	116 µs	1
fit.ivs.calc.mathlibrary.MathLib.plus	90 µs (0 %)	0 µs	200
fit.ivs.calc.mathlibrary.MathLib.div	31 µs (0 %)	0 µs	73
fit.ivs.calc.mathlibrary.MathLib.mod	14 µs (0 %)	14 µs	1
fit.ivs.calc.mathlibrary.MathLib.sqrt	9 µs (0 %)	9 µs	1
fit.ivs.calc.mathlibrary.MathLib.minus	1 µs (0 %)	0 µs	2
fit.ivs.calc.mathlibrary.MathLib.mul	1 µs (0 %)	1 µs	1

Obrázek 2: Profiling s 100 čísly



Hot Spot	Self Time	Average Time	Invocations
fit.ivs.calc.mathlibrary.MathLib.nPow	5,882 µs (3 %)	29 µs	201
fit.ivs.calc.mathlibrary.MathLib.plus	708 µs (0 %)	0 µs	2,000
fit.ivs.calc.mathlibrary.MathLib.pow	618 µs (0 %)	0 µs	1,001
fit.ivs.calc.mathlibrary.MathLib.nSqrt	417 µs (0 %)	417 µs	1
fit.ivs.calc.mathlibrary.MathLib.abs	347 µs (0 %)	0 µs	1,109
fit.ivs.calc.mathlibrary.MathLib.<init>	116 µs (0 %)	116 µs	1
fit.ivs.calc.mathlibrary.MathLib.div	35 µs (0 %)	0 µs	102
fit.ivs.calc.mathlibrary.MathLib.mod	19 µs (0 %)	19 µs	1
fit.ivs.calc.mathlibrary.MathLib.sqrt	8 µs (0 %)	8 µs	1
fit.ivs.calc.mathlibrary.MathLib.minus	1 µs (0 %)	0 µs	2
fit.ivs.calc.mathlibrary.MathLib.mul	1 µs (0 %)	1 µs	1

Obrázek 3: Profiling s 1000 čísly

4 Vyhodnocení

V prvním případě testování s deseti čísly byl celkový procesorový čas téměř optimální. **Ve druhém** případě testování se sto čísly už je lépe poznat, že funkce **nSqrt** je pomalejší než ostatní. **Ve třetím** případě s tisíci čísly lze vidět, jak se při větším výskytu čísel nabývá celková časová hodnota vykonání, převážně u funkcí **plus**, **nPow**, **pow** a **nSqrt**. Samozřejmě záleží na velikosti čísel, které prochází skrze vyvolané funkce, ale i tak je alespoň nutné optimalizovat více používané funkce.

5 Shrnutí

Z provedeného profilingu vyplívá, že většina funkcí je dobře optimalizovaná. Samozřejmě tu jsou vyjímky např. funkce **nPow** a **nSqrt**, které jsou na sobě závislé, protože funkce **nSqrt** vyvolává funkci **nPow**, tudíž z toho vyplívá, že bychom nejvíce měli optimalizovat funkce **nPow** a **nSqrt**. Nejjednodušší optimalizací funkce **nSqrt** by bylo snížení její přesnosti.