MENDELOVA UNIVERZITA V BRNĚ
PROVOZNĚ EKONOMICKÁ FAKULTA

Paralelní programování - projekt

Paralelizace procesu filtrování stopwords
nad datasetem recenzí z Amazonu

2020 Bc. Lukáš Prokop, Bc. Michal Vaško

Obsah

1 Definice problému

Jako téma našeho projektu do předmětu Paralelní programování jsme si vybrali paralelizaci procesu filtrování stopwords nad datasetem recenzí z Amazonu. Stopwords jsou slova, která se v jazyce vyskytují velmi často, avšak nenesou žádnou významovou informaci. Slouží tedy ke správnému tvoření větných konstrukcí a slovosledu. Typicky jako slopwords můžeme označit spojky, předložky, zájmena, ale i častá téměř bezvýznamová slovesa jako „být“ či „mít“.

Filtrování stopwords je proces, který se používá při počítačovém zpracování přirozeného jazyka. Jednou z oblastí, kde se filtrování využívá, jsou webové vyhledávače, které tato slova při vyhledávání ingnorují s cílem poskytnout co nejrelevantnější výsledky vyhledávání. Stopwords jsou obvykle definována formou slovníku, který je následně porovnáván na shodu s jednotlivými slovy z textu, u kterého chceme stopwords filtrovat.

Text před filtrováním stopwords	Text po filtrování stopwords
Package arrived a bit ragged but it was all there, easy to install, takes less than 5 minutes, works as described.	Package arrived bit ragged, easy install, takes 5 minutes, works.
Auto battery life can be significantly lengthened with the use of a trickle charger.	Auto battery life significantly lengthened trickle charger.
Learning to drive was one of the most liberating experiences I have had.	Learning drive liberating experiences.

Obrázek 1: Ukázka filtrování stopwords

V případě sériového zpracování se tak jedná o časově náročný proces závislý jak na podrobnosti slovníku stopwords (například v angličtině se počet těchto slov přibližně pohybuje od 300 do 1500 slov), tak na délce textu, u kterého chceme slova filtrovat. Pro zefektivnění tohoto procesu jsme se rozhodli provést paralelizaci výpočtu a zjistit, jaký vliv bude mít na rychlost s ohledem na počet vláken.

2 Spuštění skriptu

Skript obsahuje třídy `Main.java` a `Parallel.java`. První z nich obsahuje kód vykonávající se sériově a druhá obsahuje paralelní variantu stejného kódu. Celkem je možné použít 3 parametry, přičemž u sériové varianty kódu není možné použít poslední zmíněný:

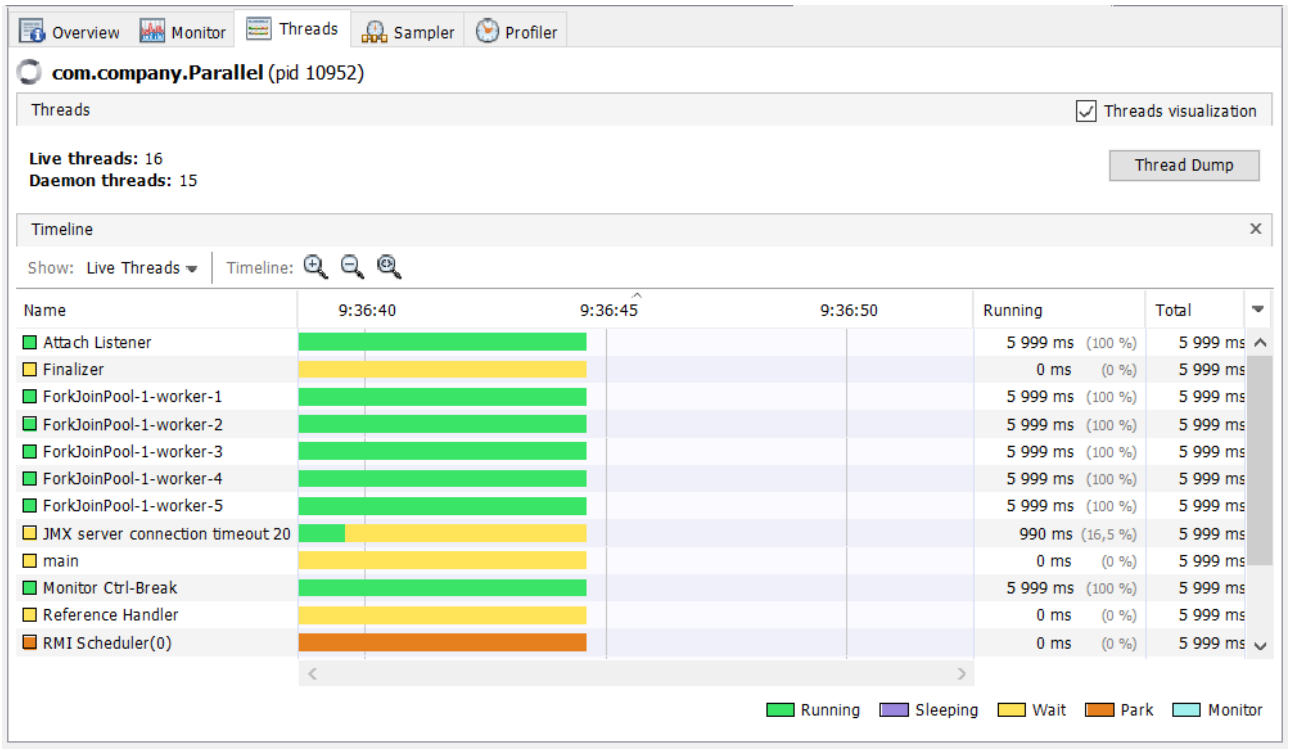
- --file / -f ... JSON soubor s recenzemi[1]
- --stopwords / -s ... textový soubor se seznamem stopwords[2]
- --threads / -t ... počet vláken pro paralelní zpracování

3 Implementace

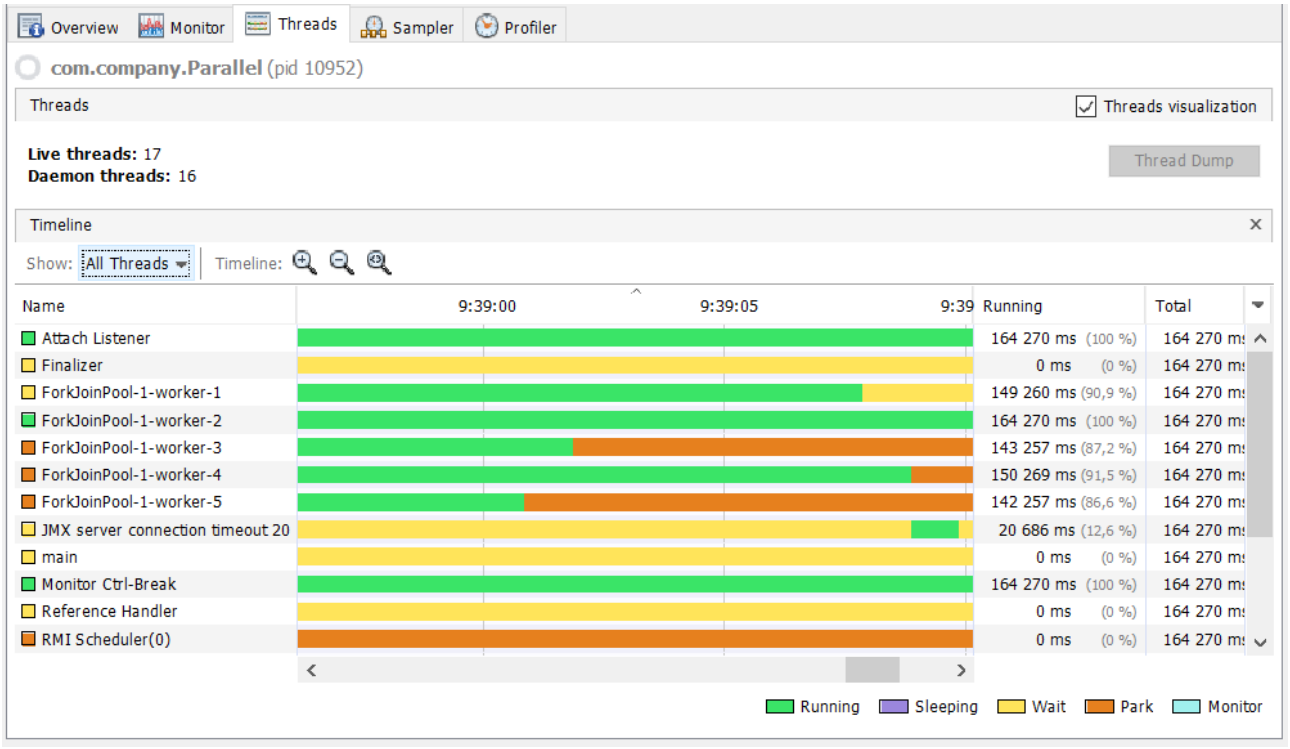
Pro implementaci tohoto skriptu jsme si zvolili programovací jazyk Java, který poskytuje nástroje pro práci s vlákny, a byl tak vhodným kandidátem pro řešení problému paralelizace.

[1] pro testování je možné použít libovolný dataset z webu `http://jmcauley.ucsd.edu/data/amazon/`
[2] každé slovo musí být na samostatném řádku

4 Vizualizace paralelního řešení



Obrázek 2: Vizualizace běhu paralelní varianty při použití 5 vláken



Obrázek 3: Vizualizace posledních vteřin běhu paralelní varianty při použití 5 vláken

5 Měření

K měření trvání jednotlivých částí skriptu jsme se rozhodli použít třídy `java.time.Instant` a `java.time.Duration`. Kód se skládá ze 3 základních částí:

- zpracování argumentů, načtení dat ze souboru a načtení stopwords
- algoritmus zajišťující filtrování stopwords
- zápis získaných výsledků do souboru

Každou z těchto částí jsme měřili zvlášť, abychom zjistili, jakou část kódu je možné paralelizovat a zda paralelizace má v případě tohoto skriptu smysl. Začátek i konec každého měření jsme stanovili pomocí volání `Instant.now()` a následný rozdíl mezi těmito údaji jsme zjistili pomocí `Duration.between(start, finish).toMillis()`.

5.1 Single thread varianta

Jak je možné vidět v tabulce, provedli jsme 5 různých pokusů nad stejným datasetem i seznamem stopwords a následně vypočítali průměrnou hodnotu měření pro jednotlivé části. Všechny údaje uvedené v tabulce jsou v milisekundách.

	Pokus 1	Pokus 2	Pokus 3	Pokus 4	Pokus 5	**Průměr**
Načtení dat ze souboru	855	697	690	695	677	**722,8**
Filtrování stopwords	44400	45459	45407	45146	45509	**45184,2**
Zápis dat do souboru	61	50	46	46	49	**50,4**

Z průměrných hodnot jednotlivých měření jsme zjistili, že samotný algoritmus zajišťující filtrování stopwords trvá **98,318 %** času a zbylých **1,682 %** času zabere zpracování argumentů, načtení dat a stopwords ze souboru a zápis výsledků do souboru. Ukázalo se tedy, že tento problém je vhodný pro paralelizaci.

5.2 Multi thread varianta

Po naimplementování a otestování multi thread varianty kódu jsme provedli měření pro jednotlivé počty vláken. Abychom dosáhli maximální přesnosti v měření, pro každé vlákno jsme provedli 3 pokusy. Hodnoty získané během jednotlivých pokusů pro různé počty vláken můžete vidět v následujících tabulkách.

Počet vláken	**1**			**2**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	891	711	810	787	718	888
Filtrování stopwords	48130	45900	48210	30139	29419	29979
Zápis dat do souboru	50	59	50	72	53	52

Počet vláken	**3**			**4**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	807	750	806	673	753	769
Filtrování stopwords	23904	24037	03719	22316	22484	22577
Zápis dat do souboru	48	52	51	59	52	47

Počet vláken	**5**			**6**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	698	738	830	770	736	760
Filtrování stopwords	21708	21710	21644	21099	21267	21377
Zápis dat do souboru	51	69	50	61	54	90

Počet vláken	**7**			**8**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	775	759	744	833	793	754
Filtrování stopwords	21390	21588	21331	22099	22000	22150
Zápis dat do souboru	57	50	53	51	49	55

Počet vláken	**16**			**32**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	860	713	706	748	743	757
Filtrování stopwords	21748	21860	21740	22629	21960	21820
Zápis dat do souboru	52	56	54	58	59	58

Počet vláken	**64**			**128**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	710	759	778	790	710	759
Filtrování stopwords	21769	21875	21804	22063	21700	21721
Zápis dat do souboru	52	46	50	55	53	50

Počet vláken	**256**			**512**		
	Pokus 1	Pokus 2	Pokus 3	Pokus 1	Pokus 2	Pokus 3
Načtení dat ze souboru	758	818	735	696	689	648
Filtrování stopwords	22023	21746	21943	22220	21294	21528
Zápis dat do souboru	48	54	57	53	55	70

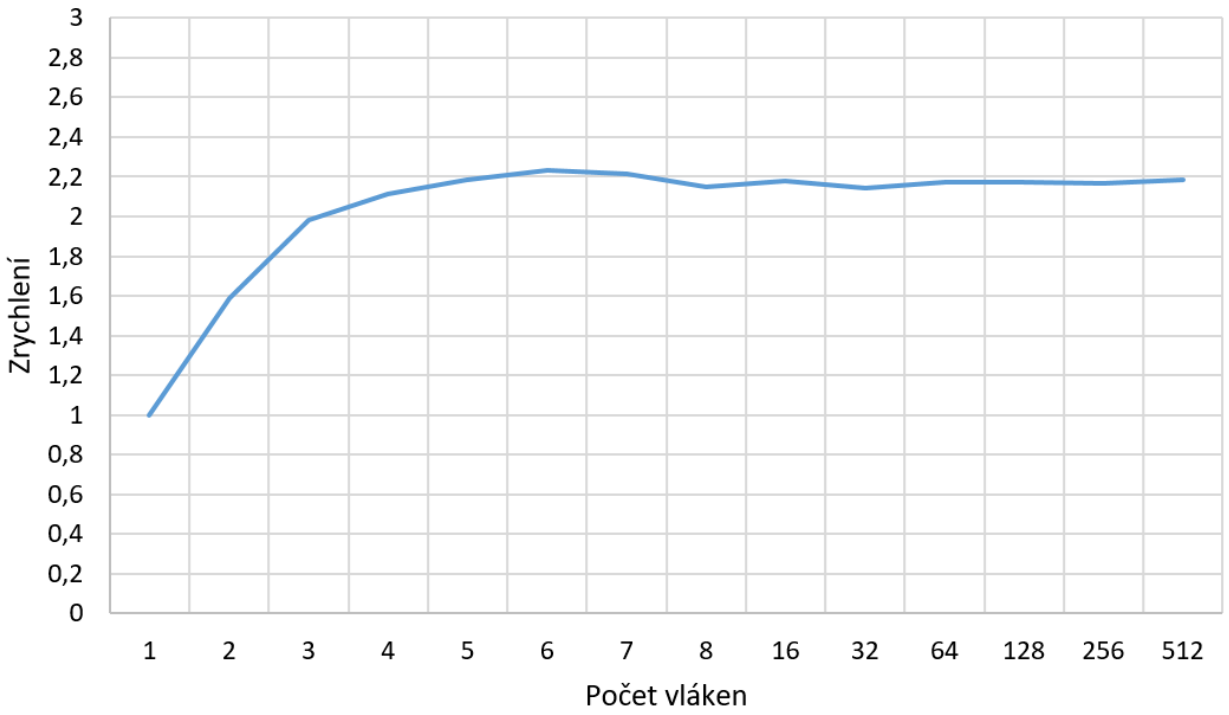
Z hodnot získaných během jednotlivých pokusů jsme pro každé vlákno vypočítali průměrnou hodnotu. Jednotlivé průměrné hodnoty pro všechna vlákna můžete vidět v následující tabulce. V tabulce je taktéž uvedeno zrychlení, které vychází z naměřeného času v řádku ”filtrování stopwords”a které udává, jaké zrychlení vůči času pro 1 vlákno nastalo.

Počet vláken	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Načtení dat ze souboru	804	797,67	787,67	731,67	755,33	755,33	759,33
Filtrování stopwords	47413,33	29845,67	23886,67	22459	21687,33	21247,67	21436,33
Zápis dat do souboru	53	59	50,33	52,67	50	68,33	53,33
Zrychlení	1	1,589	1,985	2,111	2,186	2,231	2,212

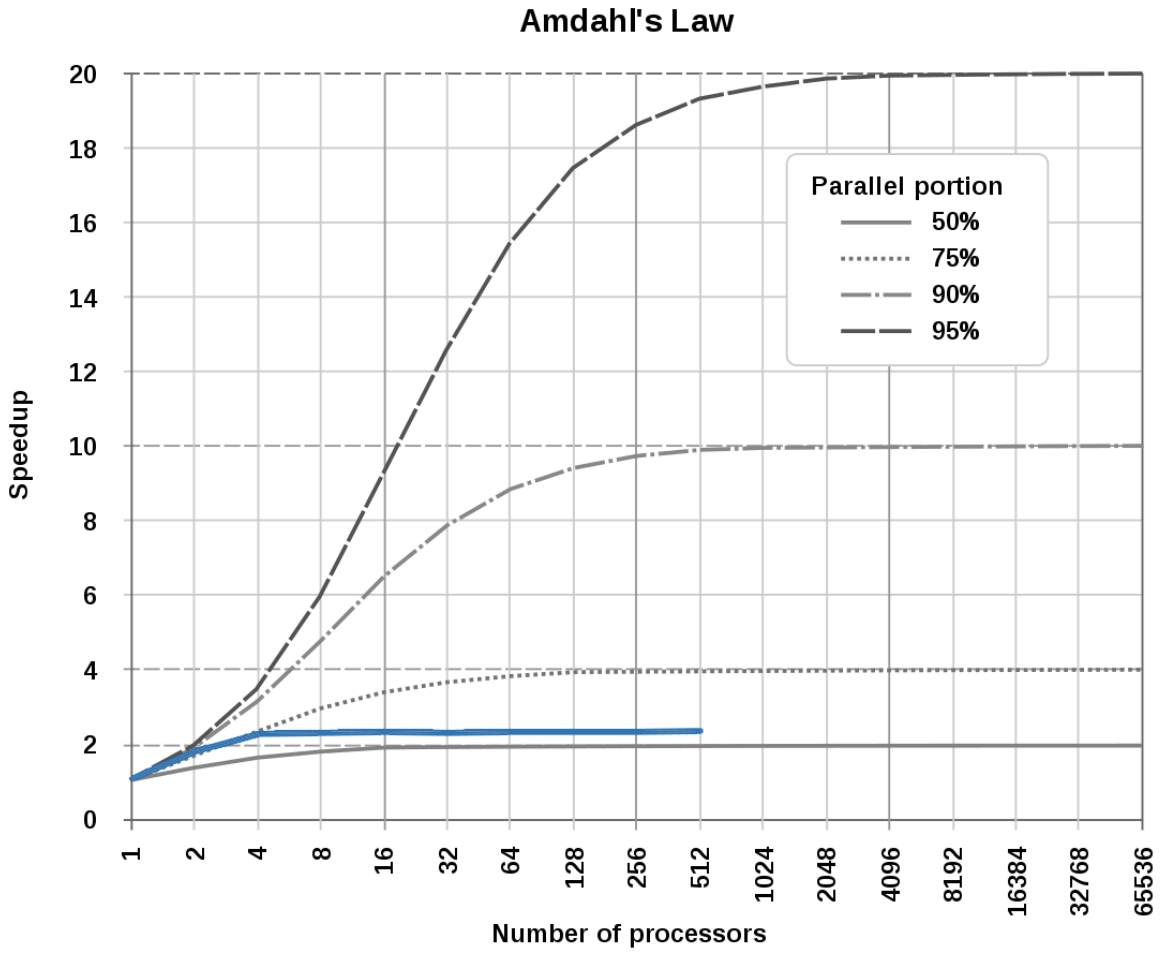
Počet vláken	**8**	**16**	**32**	**64**	**128**	**256**	**512**
Načtení dat ze souboru	793,33	759,67	749,33	749	753	770,33	677,67
Filtrování stopwords	22083	21782,67	22136,33	21816	21828	21904	21680,67
Zápis dat do souboru	51,67	54	58,33	49,33	52,67	53	59,33
Zrychlení	2,147	2,177	2,142	2,173	2,172	2,165	2,187

6 Amdahlův zákon

Amdahlův zákon se využívá k vyjádření maximálního předpokládaného zrychlení systému poté, co se vylepší některá z jeho částí. V našem případě šlo o vylepšení části kódu řešící filtrování stopwords.



Obrázek 4: Graf Amdahlova zákona získaný na základě našeho měření



Obrázek 5: Porovnání dosažených výsledků

7 Závěrečné zhodnocení

Cílem projektu bylo zparalelizovat proces filtrování stopwords nad datasetem recenzí z Amazonu. V grafu Amdahlova zákona je možné vidět, že jsme dosáhli maximálního zrychlení **2,231** při počtu **6 vláken**. S přibývajícím počtem vláken již nedocházelo ke zrychlení, naopak nastávalo mírné zpomalení procesu filtrování stopwords.