Paralelní a distribuované algoritmy
Algoritmus 4-means – implementace a analýza

David Hudák (xhudak03)

7. dubna 2023

1 Úvod

V této stručné dokumentaci se budu zabývat svým řešením druhého projektu do předmětu paralelní a distribuované algoritmy.

Zadáním bylo implementovat zástupce algoritmů typu k-means – 4-means. Ten se oproti obecné variantě liší pouze v tom, že uvádí konkrétní počet středů, které jsou algoritmem počítány.

2 Použité nástroje

Vyjma C++ v kombinaci s OpenMPI a spouštěcího skriptu Shellu po stylu prvního projektu byl pro experimenty použit ještě jednoduchý skript v PHP `measures.php` a pro grafy skript v Pythonu `plots.py` (v kombinaci s Pandasem, Matplotlibem a Seabornem).

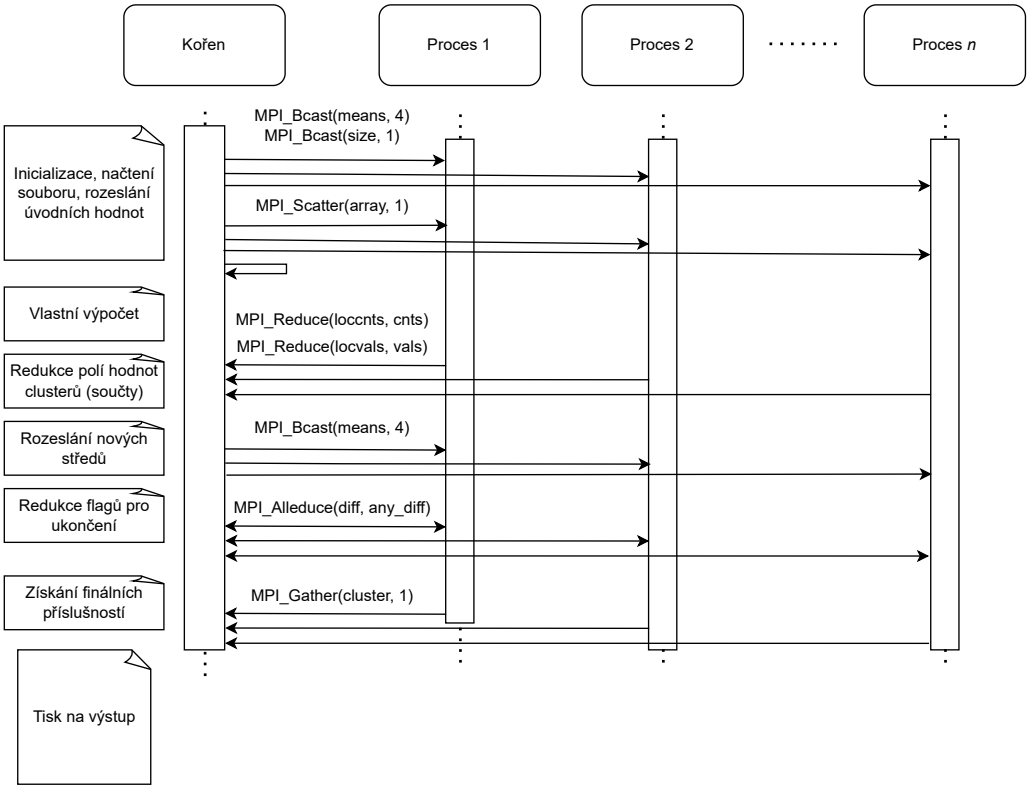
3 Návrh a implementace řešení

V řešení jsem částečně vycházel z řešení prvního projektu v rámci předmětu PRL. To spočívá v tom, že kořenový uzel (procesor s rankem 0) přečte soubor, zkontroluje, zdali odpovídá počet prvků a vytvoří úvodní čtyři středy clusterů. Kořenový uzel následně rozesílá s pomocí broadcastu všem počáteční středy a velikost pole a scatterem všem procesorům jejich hodnotu, se kterou budou po dobu života pracovat.

Následně probíhá cyklus, kdy každý spočítá příslušnost ke svému clusteru. Pokud se příslušnost některého procesoru změní oproti předchozímu, nastaví flag změny, který se pak s pomocí `Allreduce` a použitím logického **OR**u (`LOR`) rozesílá ostatním. Tím se zajistí ukončení cyklu, až všichni dosáhnou stabilního bodu. Druhou fází každé iterace je vytvoření vlastních dvou polí o 4 prvcích, kde v prvním je na pozici příslušnosti ke clusteru hodnota 1 (na zbytku 0) a analogicky ve druhém jeho hodnota na příslušné pozici (na zbytku 0). S pomocí procedury `reduce` dostane procesor s rankem 0 nové hodnoty pro výpočet středů – ty vypočítá jednoduše jako:

$$\text{Nový střed clusteru } i = \frac{\text{Součet hodnot pro daný cluster } i}{\text{Počet hodnot v daném clusteru } i}$$

Znázornění toho, jak vypadá sekvenční diagram komunikace použitého protokolu lze vidět na obrázku 1. Ten zahrnuje i informaci o způsobu tisku na výstup – kořenový procesor přebírá s pomocí procedury `Gather` jednotlivá přiřazení ke clusterům od ostatních uzlů, kterou použije jako mapování pro tisk



Obrázek 1: Sekvenční diagram komunikace výsledného algoritmu

4 Analýza složitosti algoritmu 4-means

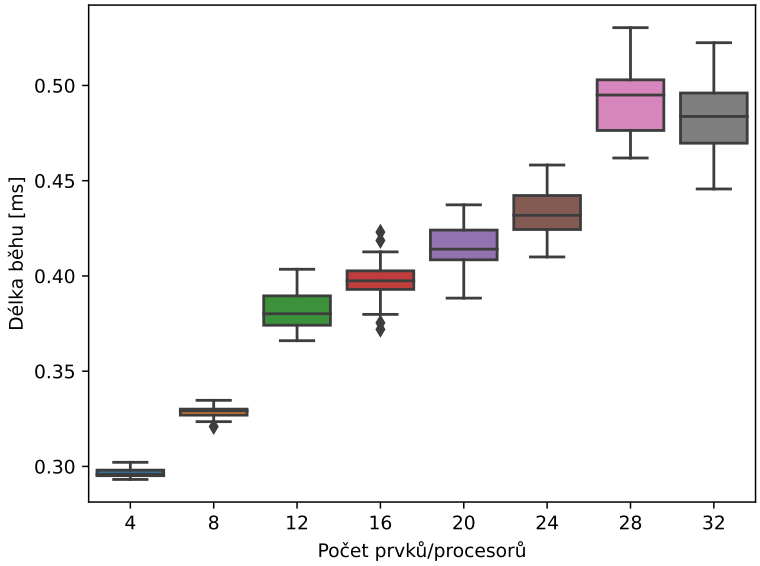
Nejintuitivnější odhad je odhad počtu procesorů. Ten je obecně $p(n) = O(n)$, jelikož každý vzor potřebuje jeden procesor. Pro odvození složitosti vycházím z článku [1], který uvádí, že celková složitost sekvenčního algoritmu k-means je $t(n) = O(TKn)$. V tomto vzorci T zastupuje počet potřebných iterací, K počet clusterů a n počet vzorů, který je potřeba v každé iteraci ohodnotit (všechny). Tento vzorec zjednodušují na $t(n) = O(n^2)$, teda na kvadratickou složitost. Odůvodňují to tak, že počet iterací potřebných k vyřešení úlohy k-means je potřeba zhruba tolik, kolik je vstupních vzorů. Počet K pak bývá nějaká konstanta (v našem případě 4).

V paralelním prostředí, kdy počet procesorů odpovídá počtu prvků, jsme schopni ohodnotit příslušnosti všech vzorů v konstantním čase $O(1)$ (pro konstantní počet clusterů), počet výpočtů nových středů clusterů se nesníží – pro každou následující iteraci potřebujeme znát ohodnocení předchozí. Z toho vyplývá, že jsme schopni v paralelním prostředí řešit úlohu v lineárním čase $t(n) = O(n) * O(1) = O(n)$. Cena algoritmu je pak kvadratická $c(n) = p(n) * t(n) = O(n^2)$, což znamená, že navržený algoritmus je optimální.

5 Měření

Zadáním projektu je pracovat s posloupnostmi 4 až 32 prvků, kdy každý procesor si bere právě jeden prvek. Dohromady to tak dává 28 možností, jak experimentovat. V rámci této dokumentace jsem zvolil výběr z experimentů, kdy proběhlo pro každou možnost 30 měření s tím, že jsem vzal pouze každé čtvrté množství prvků. Testováno bylo na lokálním zařízení (Merlinovi se nelíbilo opakované žádání o procesory) s operačním systémem Ubuntu ve verzi 20.04 a procesorem AMD Ryzen 5 5600. Na krabicovém grafu 2 pak lze nahlédnout, kolik času zhruba bylo potřeba pro jednotlivá měření. Můžeme vidět, že čas algoritmu vždy při zvednutí počtu prvků (a tedy i procesorů) stoupl,

až na výjimku 28 procesorů a 32 procesorů. Tohoto jevu se podařilo dosáhnout i při opakovaných měřeních, přičemž odůvodňuji si to tak, že nejspíše jde o stav, kdy se operačnímu systému lépe se 32 procesy než s 28 (mocnina 2).



Obrázek 2: Krabicové grafy pro měření, kdy byl vykreslen pouze každý čtvrtý počet procesorů.

6 Závěr

V rámci řešení projektu se podařilo vytvořit paralelní verzi algoritmu 4-means, která snižuje sekvenční časovou složitost z kvadratické na paralelní lineární, a to za optimální cenu n procesorů. I v grafu se potvrdilo, že časová náročnost stoupá zhruba lineárně, ale je také nutno počítat s tím, že možnosti experimentování byly omezené. Algoritmus je zaměřen na 4 až 32 prvků, což není mnoho a nedá se tak z měření odvodit nějaký větší závěr. Na druhou stranu by se dalo předpokládat, že v situaci, kdy by například jeden procesor spravoval nějakých k prvků, by tento algoritmus mohl prakticky dosahovat vyššího zrychlení, jelikož by se zmenšil dopad režie.

Reference

[1] Pakhira, M. K.: A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting. In *2014 International Conference on Computational Intelligence and Communication Networks*, 2014, s. 1047–1051, doi:10.1109/CICN.2014.220.