1 Úvod

Tato dokumentace se zabývá popisem paralelní verze algoritmu K-means. Implementován byl v jazyce C++. Dokumentace je součástí projektu do předmětu PRL na VUT FIT v letním semestru 2023 a jejím autorem je Vojtěch Fiala (xfiala61).

2 Rozbor algoritmu

Tato část se zabývá analýzou nejprve sekvenční a poté paralelní verze algoritmu K-means. Vždy bude nejprve uvedena časová a poté prostorová složitost.

2.1 Sekvenční algoritmus

Sekvenční algoritmus vychází z několika kroků, které jsou uvedeny v zadání projektu. Zde jsou uvedeny v pořadí, v jakém mají být vykonány.

Volba středů

Počet středů odpovídá počtu clusterů. Středy se v našem případě volených podle počátečních hodnot a clustery jsou vždy 4, složitost je tedy konstantní O(4), tedy O(1). Prostorová složitost je taktéž O(4), tedy O(1).

Rozřazení prvků

Prvky se řadí ke středu podle toho, ke kterému mají nejblíže – pro n prvků je potřeba provést porovnání s 4 clustery, tedy $O(4 \cdot n)$, což lze zjednodušit na O(n). Prostorová složitost je také O(n), neboť se mění pouze příslušnost prvků ke clusteru.

Výpočet nových středů

Nové středy se počítají podle průměru přiřazených hodnot. Pokud nějaký střed nemá přiřazenu žádnou hodnotu, nic se nemění. Všechny prvky je nutné při počítání zohlednit, složitost je tedy O(n). Prostorová složitost se zde nezvyšuje, neboť se akorát přepisují už existující středy.

Iterace

Pokud se hodnota nějakého středu změnila, opakuj od bodu 2, jinak skonči – nelze přesně odhadnout kolik iterací bude potřeba provést, obecně lze řici, že jich bude *i* a složitost vžpočtu jedné iterace bude O(i · n · n). Prostorová složitost iterace je O(4), tedy O(1), protože je potřeba ukládat předchozí hodnoty středů.

Složitosti

Časová složitost sekvenční verze algoritmu K-means je O(1) + O(i · n · n), což lze zjednodušit na O(n^2). Z prostorového hlediska je celková složitost O(1) + O(n) + O(1) = O(n).

2.2 Paralelní algoritmus

V paralelní verzi se ze zadání předpokládá, že jeden proces řeší jednu hodnotu a tedy na *n* hodnot je potřeba *p* procesů, tedy p(n) = n. Jako první je opět uvedena v jednotlivých krocích časová složitost a poté prostorová složitost. Následuje analýza jednotlivých kroků paralelní verze algoritmu:

Volba středů

Pro 4 clustery je složitost O(4 · log p), tedy O(log p), kde *p* je počet procesů – „kořenový“ proces nejprve určí počáteční středy, uloží je do vektoru a procedurou `Broadcast` je předá ostatním procesům. Prostorová složitost každého procesu je O(4), tedy O(1) protože každý proces musí mít uloženy středy.

Rozřazení prvků

Jeden proces pracuje s jedním číslem, tedy každý proces svůj prvek porovnává s 4 středy, tedy O(1). Kromě toho pracuje ještě s pomocným vektorem, do něhož se v cyklu trvajícím 4 iterace (podle počtu clusterů) ukládá počet prvků v clusterech, tedy celkem O(1) + O(1) = O(1). Prostorová složitost je O(1) pro každý proces, neboť musí ukládat do jakého clusteru hodnota patří a počet hodnot v clusteru.

Výpočet nových středů

Nejprve se procedurou `Reduce` spočítá celková suma všech hodnot v clusterech, což má časovou složitost O(4 · log p), tedy O(log p), neboť existují 4 clustery a p

procesů.

Následuje další volání Reduce (O(log p)), tentokrát pro pomocný vektor a určení kolik hodnot jednotlivé clustery obsahují. S těmito hodnotami pracuje pouze kořenový proces, který provede v čase O(1) výpočet nových středových bodů, které pak procedurou `Broadcast` (O(log p)) opět rozdistribuje mezi ostatní procesy. Celková složitost tohoto kroku je tedy $3 \cdot O(\log p) = O(\log p)$.

Prostorová složitost se zde zvyšuje pouze pro kořenový proces, neboť v něm dochází k alokaci nových vektorů do kterých se provádí redukce – O(8) pro 2 vektory o velikosti počtu clusterů, tedy O(1). Pro ostatní procesy se nic nepřidává.

Iterace

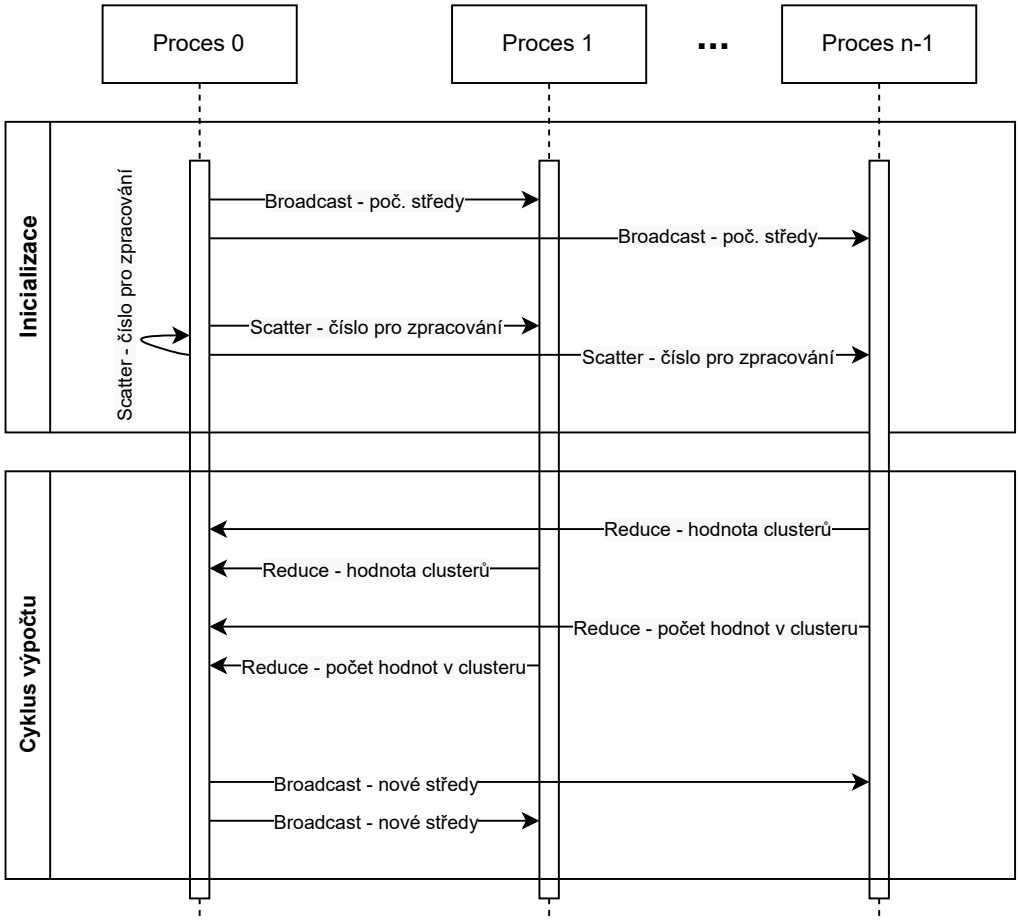
Obecně lze řici, že iterací bude *i* a složitost výpočtu iterace bude $O(i \cdot 1 \cdot \log p)$. Prostorová složitost pro každý proces je O(4), tedy O(1), z důvodu uchovávání předchozích středů.

Složitosti

Časová složitost paralelního algoritmu je $O(\log p) + O(i \cdot 1 \cdot \log p)$, což lze zjednodušit na $O(\log p)$ a jelikož počet procesů odpovídá počtu prvků, lze zapsat i jako $O(\log n)$. Celková cena algoritmu je $n \cdot O(\log n)$, tedy $O(n \cdot \log n)$. Algoritmus je v tomto případě lepší než optimální, což je nejspíše způsobeno sub-optimální verzí popsaného sekvenčního přístupu.

Z prostorového hlediska je složitost pro kořenový proces $O(n) + O(1) + O(1) + O(1) + O(1) = O(n)$. Pro ostatní procesy je to $O(1) + O(1) + O(1) = O(1)$. Z prostorového hlediska je algoritmus srovnatelný se sekvenčním s tou výjimkou, že pomocné procesy mají prostorovou náročnost výrazně nížší než kořenový.

3 Komunikační protokol



K sekvenčnímu diagramu je vhodné zmínit, že procedury Broadcast, Scatter a Reduce jsou v diagramu znázorněny pouze ilustrativně, v realitě nekomunikuje kořenový proces se všemi ostatními procesy postupně, ale skrz stromovou strukturu, kdy si data ostatní procesy distribuují i mezi sebou.

4 Závěr

Bylo ukázáno, že paralelní algoritmus je oproti popsanému sekvenčnímu z hlediska ceny lepší než optimální, v případě efektivnější sekvenční implementace[1]

[1] `https://www.researchgate.net/publication/268347680_A_Linear_Time-Complexity_k-Means_Algorithm_Using_Cluster_Shifting`

by byl neoptimální. Z prostorového hlediska jsou obě varianty srovnatelné pro kořenový proces, ostatní procesy mají nároky nižší.

Paralelní algoritmus by se dal mírně urychlit tehdy, když by se namísto procedury `Reduce` použila procedura `Allreduce` a každý proces si středové body počítal sám namísto čekání na výpočet a předání dat od kořenového procesu. To by ovšem bylo za cenu mírného navýšení prostorové složitosti, kdy by si každý proces musel ukládat mezivýsledky pro výpočet středů.