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

Volba středů (V našem případě podle počátečních hodnot) – k, kde k je počet clusterů, tedy O(k). Prostorová složitost je také O(k).

Rozřazení prvků

Rozřazení prvků do shluků podle středu, ke kterému mají nejblíže – pro n prvků je potřeba provést porovnání s k clustery, tedy $O(k \cdot n)$. Prostorová složitost je O(n), neboť samotné prvky se akorát přesouvají v rámci clusterů.

Výpočet nových středů

Výpočet nových středů jako průměru přiřazených hodnot. Pokud nějaký střed nemá přiřazenu žádnou hodnotu, nic se nemění – teoreticky v jednom clusteru může být až n prvků a složitost výpočtu průměru je tedy O(n). Prostorová složitost je zde konstantní, 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 tedy bude O(i). Prostorová složitost iterace je O(k), neboť je potřeba ukládat středy z předchozí iterace.

Celkově lze tedy určit, že časová složitost sekvenční verze algoritmu K-means je O(k) + O(i · k · n), což lze zjednodušit na O(n). Prostorová složitost je O(k) + O(n) + O(k) = O(2k + n) = O(n).

Z prostorového hlediska je celková složitost O(k) + O(n) + O(1) + O(k) = 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

Volba středů (V našem případě podle počátečních hodnot) – O(k · log p), kde *k* je počet clusterů a *p* počet procesů – „kořenový“ proces nejprve ze získaných náhodných hodnot určí *k* středů, uloží je do vektoru a procedurou `Broadcast` je předá ostatním procesům. Prostorová složitost je n · O(k), protože každý proces musí mít uloženy středy. Kořenový proces také musí data načíst ze souboru – složitost O(n).

Rozřazení prvků

Rozřazení prvků do shluků podle středu, ke kterému mají nejblíže – v tomto projektu jeden proces pracuje s jedním číslem, tedy každý proces svůj prvek porovnává s *k* clustery, tedy O(k).

Kromě toho ovšem musí pracovat ještě s pomocným vektorem, který určuje, kolik prvků tento proces uložil do jednotlivých clusterů, což je další cyklus přes všechny clustery O(k) a tento krok má tedy celkovou složitost O(k^2). Prostorová složitost jsou 2k pro každý proces, neboť musí ukládat hodnoty v clusterech a jejich počet.

Výpočet nových středů

Výpočet nových středů jako průměru přiřazených hodnot. Pokud nějaký střed nemá přiřazenu žádnou hodnotu, nic se nemění – nejprve se procedurou `Reduce` spočítá celková suma všech hodnot v clusterech, což má časovou složitost O(k · log n), kde *k* je počet clusterů a *n* počet prvků (a také procesů).

Následuje další volání Reduce, tentokrát pro pomocný vektor a určení kolik hodnot jednotlivé clustery obsahují. S těmito hodnotami pracuje pouze kořenový proces, který provede v čase O(k) výpočet nových středových bodů, které pak procedurou `Broadcast` opět rozdistribuje mezi ostatní procesy. Celková složitost tohoto kroku je tedy 3 · O(k · log n). Prostorová složitost jsou 2k pro kořenový proces, neboť v něm dochází k alokaci nových vektorů do kterých se provádí redukce. Pro ostatní procesy je prostorová složitost této části konstantní.

Iterace

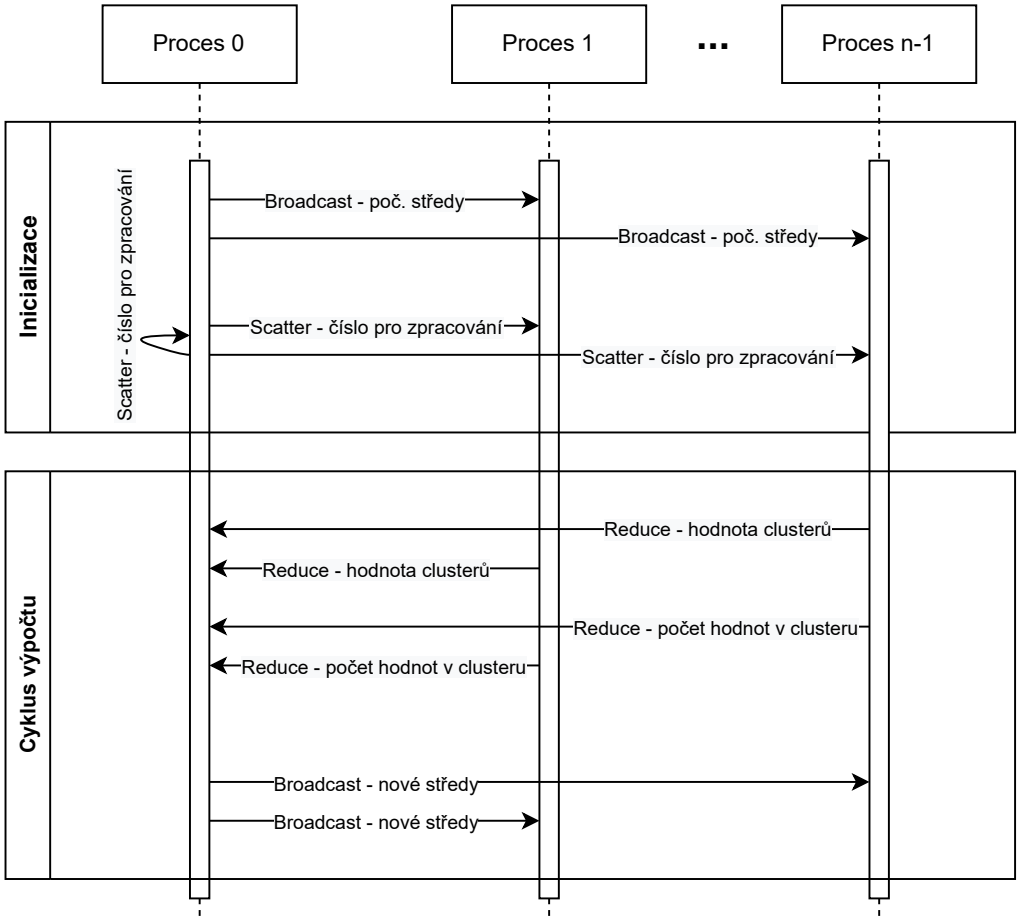
Pokud se hodnota nějakého středu změnila, opakuj od bodu 2, jinak skonči – zde platí totéž co u sekvenčního algoritmu – nelze přesně odhadnout kolik iterací bude potřeba provést, obecně lze řici, že jich bude *i* a složitost tedy bude O(i). Prostorová složitost pro každý proces je O(k) z důvodu uchovávání předchozích středů.

Na základě popsaných kroků lze usoudit, že složitost paralelního algoritmu je O(k · log p) + O(i · 3 · k · log p · k^2) = O(k · log p) + O(i · 3 · k^3 · log n).

Obecně lze usoudit, že zrychlení v závislosti na počtu prvků je veliké (log n versus n), což je ovšem zaplaceno vyšší režií a počtem potřebných procesorů. Celková cena algoritmu je O(k · log p) + O(i · 3 · k^3 · n · log n), což lze zjednodušit na O(n · log n). Algoritmus tedy není optimální.

Z prostorového hlediska je celková složitost O(n, pouze kořenový) + O(k) + O(2k) + O(k) + O(2k, pouze kořenový) – lze tedy říci, že pro kořenový proces O(n), pro ostatní konstantní a tedy zanedbatelná. Z prostorového hlediska je algoritmus srovnatelný se sekvenčním s tou výjimkou, že pomocné procesy mají prostorovou náročnost výrazně nížší než kořenový.

3 Komunikační protokol



4 Závěr

Bylo ukázáno, že paralelní algoritmus není z časového hlediska optimální, ovšem není ani příliš náročnější oproti sekvenční variantě. Z prostorového hlediska jsou obě varianty srovnatelné.

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.