PRL - Projekt 2

Samuel Repka - xrepka07

Rozbor a analýza algoritmu

Z analýzy vynechám načítanie údajov a záverečné vypísanie výsledkov.

Časová zložitosť

Ako prvé po načítaní dát program distribuuje údaje jednotlivým procesom. Myšlienka je taká, že každý procesor bude počítať zhluk pre jedno číslo. Toto číslo sa v priebehu programu nebude meniť, budú sa meniť len centroidy zhlukov.

Treba teda každému procesoru doručiť jeho číslo + počiatočné centroidy. Na toto využívam metódy *MPI_Scatter* a *MPI_Bcast*. Podľa tohoto zdroja https://stackoverflow.com/questions/10625643/mpi-communication-complexity, je komplexita *MPI_Scatter* **O(log(p) + n)** a *MPI_Bcast* **O(n log(p))**, kde **p** je počet procesorov a **n** je počet prvkov. Keďže nemám predstavu, na akej architektúre sa nachádzam, budem pracovať s týmito údajmi. Centroidy sú 4, teda *MPI_Bcast* bude mať komplexitu **O(4 log(p))**.

Po doručení potrebných dát sa môžme presunúť na samotné k-means. Každý procesor vypočíta, ku ktorému zhluku patrí jeho číslo. Každý procesor musí prejsť každým zhlukom, teda časová zložitosť tohoto kroku je v mojom prípade **O(4)**.

Následne treba prepočítať centroidy. Toto robí koreňový proces pomocou dát, ktoré do neho doručím pomocou *MPI_Reduce*. Redukujem pre každý zhluk raz, zakaždým dvojzložkové pole, kde prvý prvok je číslo, za ktoré je zodpovedný daný procesor alebo 0, v závislosti na tom, či sa práve redukuje zhluk, ktorému dané číslo momentálne patrí. Druhý prvok je indikátor, či sa v prvom prvku nachádzalo číslo, alebo nie (1 alebo 0). Výsledok v koreňovom procese je teda tiež dvojzložkové pole, kde prvý prvok je suma čísel patriacich zhluku a druhý je počet čísel, ktoré tam patria. Ak budeme predpokladať ideálnu stromovú štruktúru, jedna redukcia prebehne v **O(log(p))** čase. Prebehne ale 4 krát, teda komplexita tohoto kroku je **O(4 log(p)).**

Po prepočítaní centroidov ich treba doručiť do ostatných procesorov, na čo použijem ďalší *MPI_Bcast* so zložitosťou **O(4 log(p))**.

Tento k-means cyklus sa bude opakovať do konvergencie, počet iterácií je neznámy, označím ho teda **i**.

Po dosiahnutí konvergencie treba ešte doručiť priradenia do zhlukov koreňovému procesoru. Na to používam funkciu *MPI_Gather*, s pravdepodobnou zložitosťou **O(log(p) + n)**.

Celková zložitosť algoritmu je teda **O(log(p) + n + 4 log(p) + i*(4 log(p) + 4 log(p)) + log(p) + n)**. Zo zadania tiež vieme, že **n == p**. Výsledná zložitosť po odstránení pomaly rastúcich členov a konštánt je **O(n + i*(log(n))).** V [1] je uvedené, že **i** v mojom prípade bude nanajvýš **n**. Teda výsledná časová zložitosť je **O(n log(n))**, teda linearitmická**.**

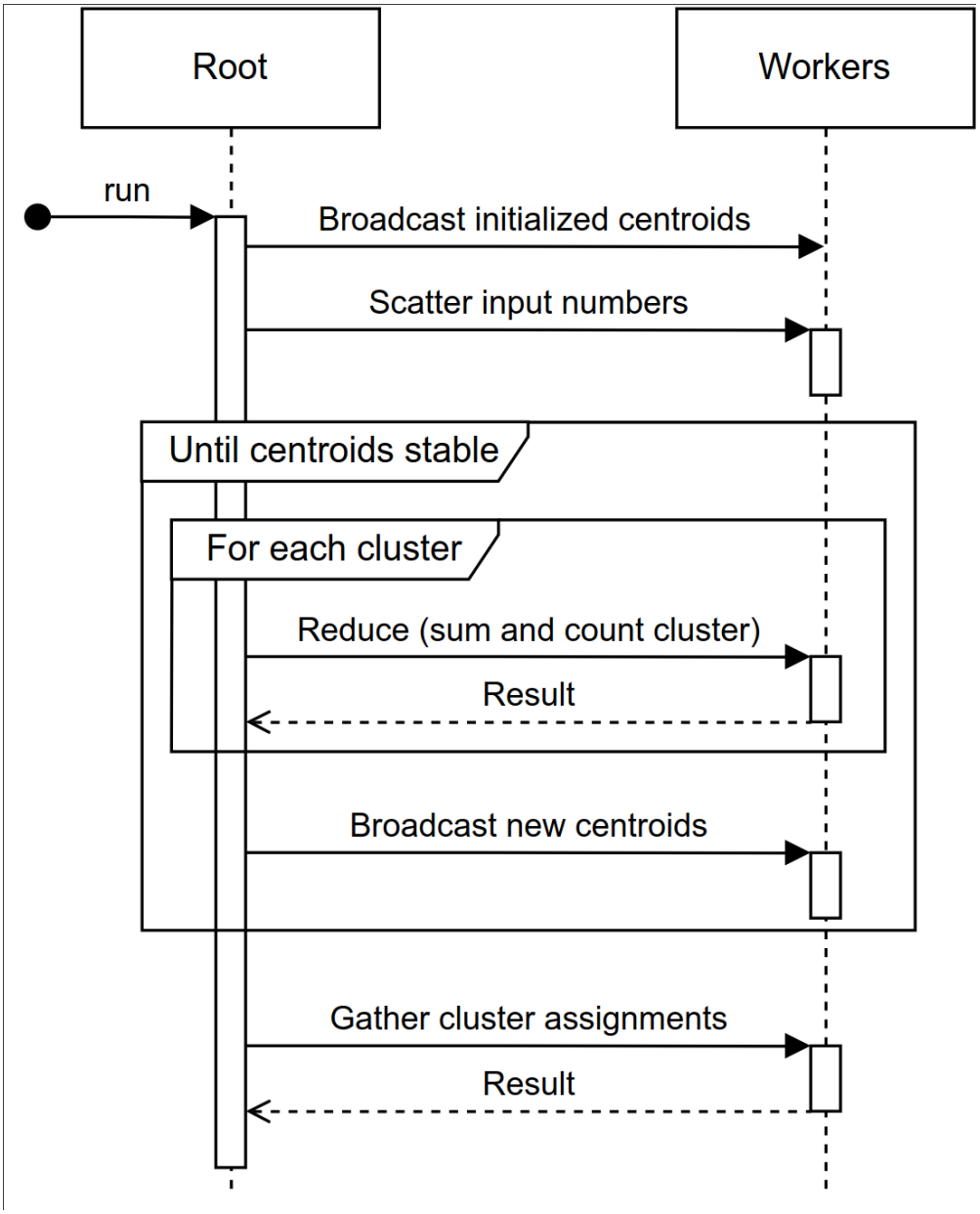
Priestorová zložitosť

Jediné dynamicky alokované polia sa nachádzajú na koreňovom procesore. Všetky ostatné procesory pracujú v konštantnom priestore. Polia na koreňovom procesore sú 2, a obe majú veľkosť **n**. Priestorová zložitosť algoritmu je teda **O(n)**.

Celková cena

Celková cena algoritmu je **O(n log(n)) * n**, teda **O(n^2 log n)**.

Komunikačný protokol



Záver

Časová zložitosť sekvenčného k-means sa vypočíta ako **O(n*i*k),** kde **k** je počet zhlukov [2]. V mojom prípade je teda optimálna cena **O(4 n^2)**. Môj algoritmus s cenou **O(n^2 log n)** teda nie je optimálny.

Zdroje

[1] DASGUPTA, S. How fast is k-means?. In: *LEARNING THEORY AND KERNEL MACHINES* [online]. BERLIN: Springer Nature, 2003, s. 735-735 [cit. 2023-04-15]. ISBN 9783540407201. ISSN 0302-9743. Dostupné z: doi:10.1007/978-3-540-45167-9_56

[2] ZHAO, Yanping a Xiaolai ZHOU. K-means Clustering Algorithm and Its Improvement Research. *Journal of Physics: Conference Series* [online]. Bristol: IOP Publishing, 2021, **1873**(1), 12074 [cit. 2023-04-15]. ISSN 1742-6588. Dostupné z: doi:10.1088/1742-6596/1873/1/012074