# Paralelní a distribuované algoritmy (PRL 22/23)

Slavomír Svorada (xsvora02)

April 2023

## 1 Popis implementácie

Program je riešený v jazyku C++, ktorý implementuje algoritmus K-means clustering s využitím Open MPI. Na začiatku sa načítaju dáta zo súboru "numbers" pomocou funkcie `readNumbers`. Následne sa inicializujú centroidy prvými 4 hodnotami zo súboru. Program potom vstúpi do while cyklu, kde sa vykonáva algoritmus K-means. Pre rozposielanie aktuálnych centroidov na všetky procesy sa použije funkcia `MPI_Bcast`. Následne použitím funkcie `MPI_Scatter` na rozdelia dáta medzi procesy. Každý proces priradí dáta k najbližšiemu centroidu pomocou funkcie `closeCenter`, ktorá vráti index najbližšieho centroidu. Priradené dáta sa potom ukladajú do vektorov *processA* a *processB*, kde *processA* ukladá súčet priradených dát a *processB* ukladá počet priradených dát.

Vďaka použitiu funkcie `MPI_Reduce` sa sčítajú dáta naprieč všetkými procesmi. Výsledok sa potom rozposiela na všetky procesy pomocou funkcie `MPI_Bcast`. Pri *rank == 0* sa vypočítajú nové centroidy ako priemer priradených dát pre každý centroid. Tento proces sa opakuje, kým sa centroidy už nezmenia. Ak sú nové centroidy rovnaké ako predchádzajúce, program ukončí while cyklus a vypíše konečné centroidy a dáta, ktoré k jednotlivých centroidom patria (zhluky).

V programe sa používajú vektory na ukladanie dát a centroidov a taktiež sa používa niekoľko pomocných funkcií na čítanie dát, zobrazovanie dát či výpočet najbližšieho centroidu. Ako dátový typ sa používa *float* pre centroidy a dátový typ *int* pre dátové body.

### 1.1 Teoretická zložitosť (časová a priestorová zložitosť, celková cena)

K-means algoritmus je algoritmus, ktorý slúži na klastrovanie dát do určeného počtu zhlukov. Časová zložitosť tohto algoritmu závisí na počte dátových bodov `(n)`, počte zhlukov `(k)` a dimenzionalite dát `(d)`. V každej iterácii sa vypočíta vzdialenosť každého dátového bodu od každého centroidu zhluku, čo vyžaduje `O(n*k*d)` času. Potom sa každý dátový bod priradí k najbližšiemu centroidu, čo vyžaduje `O(n*k)` času. Následne sa vypočítajú nové centroidy zhlukov, čo vyžaduje `O(n*d)` času. Celková časová zložitosť K-means algoritmu je zvyčajne

`O(I*n*k*d)`, kde `(I)` je počet iterácií. Priestorová zložitosť K-means algoritmu závisí od počtu dátových bodov, dimenzionality dát a počtu zhlukov, a to konkrétne na ukladaní dátových bodov a centroidov zhlukov. Celková cena K-means algoritmu závisí od času a priestoru potrebného na výpočet. Cena K-means algoritmu sa zvyšuje s narastajúcim počtom iterácií, dimenzionality dát a počtom dátových bodov.

# 2 Komunikačný protokol

Pri implementácií sa využívajú 3 funkcie z Open MPI. MPI je štandard slúžiaci pre posielanie správ a komunikáciu medzi procesmi v paralelnom výpočte. Medzi používané funkcie patria `MPI_Scatter`, `MPI_Bcast` a `MPI_Reduce`. Tieto funkcie slúžia pre komunikáciu medzi procesmi.

- `MPI_Scatter`: rozdeľuje dáta z jedného procesu na menšie časti a rozosiela ich do ostatných procesov v skupine. Každý proces obdrží len svoju časť dát.
- `MPI_Bcast`: rozosiela rovnaké dáta zo zdrojového procesu všetkým ostatným procesom v skupine. Všetky procesy dostanú rovnaké dáta. (rozosielanie centroidov).
- `MPI_Reduce`: kombinuje hodnoty z rôznych procesov a výsledok poskytuje späť zdrojovému procesu (rootu). V tomto prípade sa používa operácia súčet na vypočítanie výsledku.

Použitie týchto funkcií je znázornené na tomto sekvenčnom diagrame:

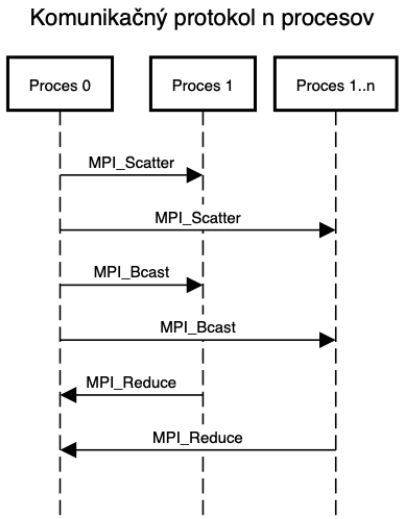


Figure 1: Komunikácia medzi procesmi.

# 3 Záver

Projekt by mal byť implementovaný tak, že spĺňa požadovanú funkčnosť. Ako vstup je postupnosť 4 až 32 náhodných čísel. Ako výstup je zobrazenie jednotlivých zhlukov nasledujúcim spôsobom:
[46] 34, 82
[232.25] 226, 248, 215, 240
[158.5] 147, 170
[12] 22, 2