Dokumentace PRL projektu 2

Lukáš Plevač < *xpleva07@vutbr.cz* >

1 Algoritmus

Algorithm 1 MPI Parallel 4 k-means clustering

numbers ← *read file on rank* 0
*centeroids*1 ← *first* 4 *numbers from numbers on rank* 0
MPI_SCATTER numbers to number ▷ one number per one process
*MPI_BCAST centeroids*1 ▷ 4 float numbers
*centeroids*2 ← *centeroids*1
do
  LClustersSums ← {0, 0, 0, 0, 0, 0, 0, 0} ▷ 4 for sums 4 for counts
  LClusterI ← *index of nearest centeroid in centeroids*1
  LClustersSums on LClusterI ← *number*
  LClustersSums on LClusterI + 4 ← 1
  *MPI_ALLREDUCE MPI_SUM LClustersSums to centeroids*2
  *centeroids*2 *range*(0..3) ← *mean of clusters in centeroids*2
  *Exchange centeroids*2 *with centeroids*1
while *centeroids*2 ≠ *centeroids*1
Print results on rank 0

1.1 Časová složitost

V této části bude algoritmus rozebírá z pohledu časové složitosti. Protože k-means clustering je algoritmem, který počítá středy shluků iterativně a počet iterací souvisí přímo se vstupem, respektive závisí na pořadí vstupních dat a nedá se jednoduše určit kolik iterací bude nutné pro dokončení výpočtu, bude uvedena složitost jedné iterace algoritmu.

- **O(N)** Načtení N čísel ze vstupního souboru
- **O(1)** Zvolení počátečních středů clusterů
- **O(log(N))** MPI_SCATTER N čísel na N procesorů
- **O(log(N))** MPI_BCAST 4 čísela na N procesorů
- **O(1)** výpočet přítomnosti v clusteru, nalezení nejbližšího středu

- **O(2 log(N))** MPI_ALLREDUCE pro sumu všech středů přes N procesů
- **O(1)** zprůměrování shluků

Pokud nebudeme počítat náročnost načtení vstupu ze souboru je asymptotická časová složitost jedná iterace algoritmu **O(log(N))** celková složitost všech iterací algoritmu je poté **O(k * log(N))**, kde k závisí na vstupu, tedy jak rychle bude konvergovat k správnému řešení.

1.2 Prostorová složitost

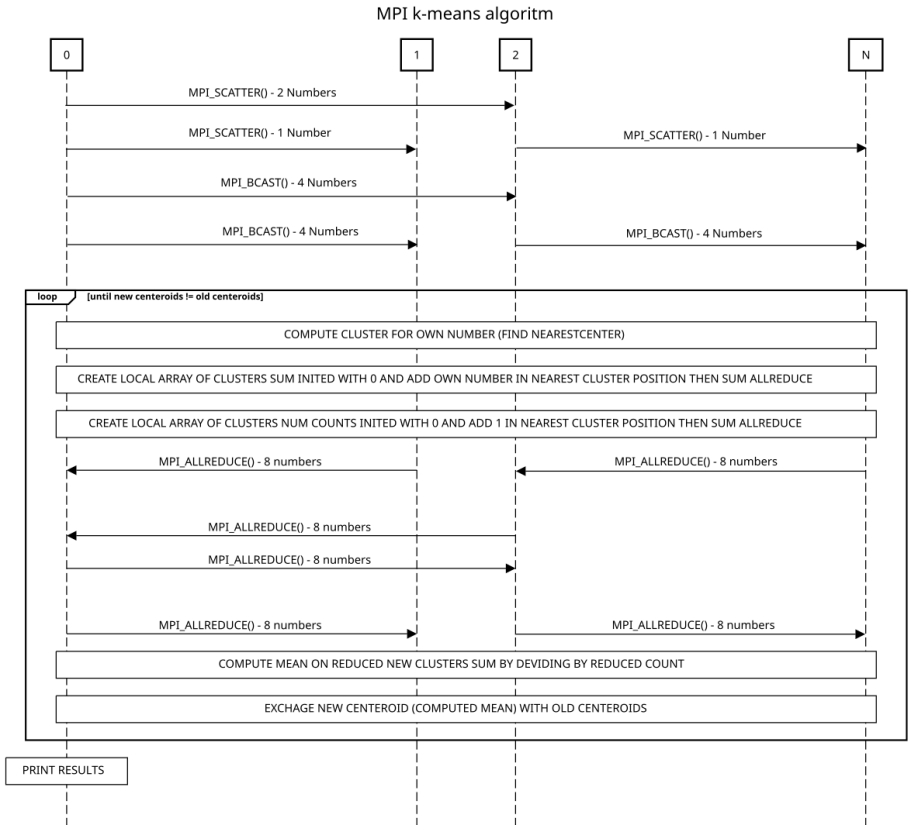
Algoritmus na každém procesu alokuje místo pro jedeno číslo, dále 3x místo pro 4 středy shluků - 2 verze pro uložení pozic středů (Stará a nová) a jeden pro lokální součet pro potřeby redukce. Všechny tyto pole je nutné ještě rozšířit 2x kvůli uchování počtu prvků v shluku. Výsledná prostorová složitost je tedy **2*4*3 + 1** na každém procesy, celkem nad všemi procesy tedy **25 * N**, kde N je počet čísel v vstupu. Bitově se jedná o jedno 1B číslo a o **2*4*3 = 24** float čísel (na standardní architektuře 4B) celkem tedy **24 * 4 + 1B = 97B** na jednom procesu na všech procesech tedy **N * 97B**.

1.3 Cena

Cena paralelního algoritmu se vypočte jako součinem součinem paralelního času a počtu procesorů. V našem případě můžeme spočítat jako součin časové složitosti a počtu procesorů, který je **O(P*k*log(N))**, kde **P** je počet procesorů **k** je rychlost konvergence algoritmu (pro jednoduchost 1) a **N** je počet čísel. V našem specifickém případě je **N = P**, tedy máme tolik procesorů jako čísel.

2 Sekvenční diagram komunikace mezi procesy

Dikagram níže ukazuje komunikaci N procesů při výpoctu stredu shluků algoritme k-means. mezi procesem 2 a N se nachezi N - 2 procesů které jsou pro potřeby vizualizace skrity.



3 Závěr

Algoritmus má sice menší teoretickou časovou složitost než jeho sekvenčí verze O(N*N), ale při reálném nasazení budeme pozorovat praví opak, protože teoretická časová složitost do sebe nemá započítán čas nutný pro komunikaci mezi procesory, proto je tento algoritmus v reálném nasazení nepoužitelný. Nicméně pokud bychom N čísel nerozdělovali mezi N procesorů, ale M procesorů kde $M < N$ byly bychom pro vhodném M schopni pozorovat navýšení výkonu oproti sekvenčí verzi, protože narozdíl od N verze, M verze bude pro vhodný počet prvků na jednom procesoru dostatečně vytěžovat procesory, narozdíl od N verze kde se bude většinu času čekat na dokončení komunikace.