

Implementace paralelního algoritmu K-means

Cílem druhého projektu do předmětu paralelní a distribuované algoritmy (PRL) bylo implementovat paralelní algoritmus K-means s využitím knihovny OpenMPI. Tato dokumentace popisuje využitý shlukovací algoritmu spolu s jeho složitosti, komunikaci mezi jednotlivými procesy a příklad použití vytvořeného programu.

1 K-means

V této sekci ukážeme sekvenční a paralelní algoritmus K-means a jejich složitosti. Shlukování bylo prováděno na posloupnosti jednorozměrných hodnot. Za účelem projektu jsou vytvářeny 4 shluky (v důkazech složitosti bude počítáno s $k = 4$). Jako první hodnoty středů shluků jsou zvoleny první 4 hodnoty shlukované posloupnosti.

1.1 Sekvenční K-means

Na vstupu sekvenčního algoritmu je posloupnost (množina) jednorozměrných bodů *points* a počet tvořených shluků. Jako středy (*means*) shluků je vybráno k prvních bodů z posloupnosti *points*. Každý shluk (*cluster*) obsahuje množinu přiřazených bodů. Pro $i \in \mathbb{N} > 1$ platí $\forall n, m \in \mathbb{N} \leq k : cluster_i(n) \cap cluster_i(m) \neq \emptyset \iff n = m$ a také $\bigcup_{n \in \mathbb{N} \leq k} cluster_i(n) = points$. V každé iteraci algoritmu jsou body přiřazeny do shluku, který má nejbližší střed. Pokud během iterace nedošlo ke změně žádného shluku, je algoritmus ukončen.

Algorithm 1: Sekvenční K-means

Input: *points*, $k \in \mathbb{N}$

Output: $cluster : k \rightarrow 2^{points}$

```

1 forall  $p_n \in \{p_1, \dots, p_k\} \subseteq points$  do
2    $mean(n) \leftarrow p_n$ 
3    $cluster_0(n) \leftarrow \emptyset$ 
4 end
5  $i \leftarrow 1$ 
6 repeat
7   forall  $p \in points$  do
8      $c \leftarrow \operatorname{argmin}_{n \in \mathbb{N}; n \leq k} (|mean(n) - p|)$ 
9      $cluster_i(c) \leftarrow cluster_{i-1}(c) \cup \{p\}$ 
10  end
11  forall  $n \in \mathbb{N} : n \leq k$  do
12     $means(n) \leftarrow \frac{1}{|cluster_i(n)|} \sum_{p \in cluster_i(n)} p$ 
13  end
14   $i \leftarrow i + 1$ 
15 until  $\forall n \in \mathbb{N} \leq k : cluster(n)_i = cluster_{i-1}(n)$ 
16 return  $cluster_i$ 

```

1.1.1 Časová složitost

Časová složitost sekvenčního algoritmu K-means, pro $K = 4$, je $O(n^2)$, kde n je počet shlukovaných bodů.

Důkaz. Časová složitost inicializace shluků (1–4) je $O(k)$, kde k je počet shluků. Složitost smyčky aktualizující shluky (6–15) je $O(n + nk) = O(n)$, kde n je počet bodů. V [2] bylo ukázáno, že konvergence algoritmu nastává po $O(n^{kd})$, kde d je počet dimenzí. Tato hranice byla v [1] upřesněna pro speciální případy, kdy $k < 5$ a $d = 1$ na $O(n)$ iterací. Z předchozího vyplývá, že celková časová složitost algoritmu 1 je $O(n^2)$. \square

1.1.2 Prostorová složitost

Prostorová složitost sekvenčního algoritmu K-means, pro $K = 4$, je $O(n)$, kde n je počet shlukovaných bodů.

Důkaz. Algoritmus využívá proměnnou *means* s prostorovou složitostí $O(k) = O(1)$. Proměnné $cluster_i$ se dají znovu využívat, je potřeba uchovávat pouze $cluster_i$ a $cluster_{i-1}$. Jejich prostorová složitost je $O(n)$. Celková prostorová složitost algoritmu je $O(n)$. \square

1.2 Paralelní K-means

Algorithm 2: Paralelní K-means

Input: *points*, $k \in \mathbb{N}$

Output: *cluster* : $k \rightarrow 2^{points}$

```

1 forall  $p_n \in \{p_1, \dots, p_k\} \subseteq points$  do
2    $mean(n) \leftarrow p_n$ 
3    $Broadcast(means(n))$ 
4 end

5 forall  $p_i \in \{p_1, \dots, p_n\} = points$  do in parallel
6    $C^{(i)} \leftarrow 0$  // Přiřaď 0 do proměnné shluku C procesu i.
7    $changed^{(i)} \leftarrow \text{True}$  // Přiřaď True do proměnné changed procesu i.
8 end

9  $changed \leftarrow \text{True}$ 
10 while  $changed$  do
11   forall  $p_i \in \{p_1, \dots, p_n\} = points$  do in parallel
12      $c \leftarrow \operatorname{argmin}_{n \in \mathbb{N} : n \leq k} (|mean(n) - p_i|)$ 
13      $changed^{(i)} \leftarrow c \neq C^{(i)}$  // Logický výraz
14   end
15    $cluster\_size \leftarrow GetSizesOfClusters()$  // Redukcí získej velikosti všech shluků.
16    $cluster\_sum \leftarrow GetSumInClusters()$  // Redukcí získej sumu v každém shluku.
17   forall  $n \in \mathbb{N} : n \leq k$  do
18      $means(n) \leftarrow cluster\_sum(n) / cluster\_size(n)$ 
19      $Broadcast(means(n))$ 
20   end
21    $changed \leftarrow ORReduce(changed^{(1)}, \dots, changed^{(n)})$  // Redukce procesových changed.
22 end

23 forall  $n \in \mathbb{N} : n \leq k$  do
24    $cluster(n) = \{p_i \in points \mid C^{(i)} = n\}$ 
25 end

26 return cluster

```

Paralelní algoritmu K-means využívá n procesů, kde n je shodné s počtem shlukovaných bodů. Na vstupu algoritmu je posloupnost (množina) jednorozměrných bodů *points* a počet tvořených shluků k . Jako středy (*means*) shluků je vybráno k prvních bodů z posloupnosti *points*. Tyto středy jsou následně pomocí operace *Broadcast* rozeslány ostatním procesům. Každý procesor obsluhuje jeden vstupní bod. Procesor i obsluhuje bod p_i . Každý proces obsahuje lokální proměnné $C^{(i)}$ a $changed^{(i)}$. $C^{(i)}$ určuje příslušnost bodu p_i do shluku. $changed^{(i)}$ určuje, zda v současné iteraci algoritmu změnil bod p_i shluk. Po ukončení rozmístění bodů do shluků jsou pomocí operace *Reduce* získány velikosti (počet bodů ve shluku) a sumy bodů v každém shluku. Pomocí

těchto hodnot proces *root* přepočítá hodnoty středů shluků a operací *Broadcast* je rozešle všem procesům pro další iteraci výpočtu. Všechny lokální hodnoty *changed*⁽ⁱ⁾ jsou operací *ORReduce* zredukovány do proměnné *changed*. Pokud je *changed* rovno *True* (nějaký bod změnil shluk) je započata nová iterace. Výpočet je ukončen pokud už žádným bod nemění shluk.

1.2.1 Časová složitost

Časová složitost paralelního algoritmu K-means, pro $K = 4$, je $O(n \cdot \log n)$, kde n je počet shlukovaných bodů.

Důkaz. Časová složitost výběru středů a jejich rozeslání procesorům (1–4) je $O(\log n)$. Složitost inicializace procesových proměnných (5–8) je $O(1)$. Smyčka pro výpočet nových středů má časovou složitost $O(k) = O(1)$. Časová složitost Reduce pro získání *cluster_size* (15), *cluster_size* (16) a *changed* (21) je $O(\log n)$. Přepočítání středů shluků a jejich rozeslání procesům (17–20) má časovou složitost $O(\log n)$. Protože z [1] víme, že ke konvergenci pro jednorozměrná data a $k < 5$ dochází po $O(n)$ iteracích hlavního cyklu (10–22), pak je celková časová složitost paralelního algoritmu $O(n \cdot \log n)$. \square

1.2.2 Cena

Cena paralelního algoritmu K-means, pro $K = 4$, je $O(n^2 \cdot \log n)$, kde n je počet shlukovaných bodů. Algoritmus není optimální.

Důkaz. Dříve jsme ukázali, že časová složitost paralelního algoritmu K-means je $O(n \cdot \log n)$, kde n je počet shlukovaných bodů. Při použití p procesů, kde $p = n$ je cena paralelního algoritmu $O(pn \cdot \log n) = O(n^2 \cdot \log n)$. Lze vidět, že paralelní algoritmus není efektivní, protože cena sekvenčního algoritmu je $O(n^2)$. \square

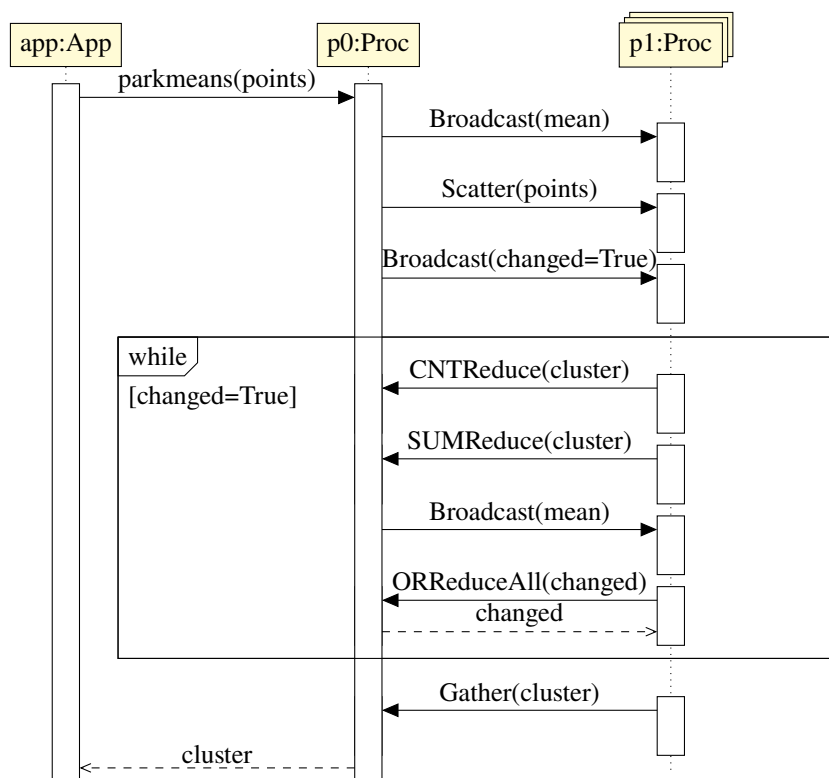
1.2.3 Prostorová složitost

Prostorová složitost paralelního algoritmu K-means, pro $K = 4$, je $O(n)$, kde n je počet shlukovaných bodů.

Důkaz. Prostorová složitost proměnné *means* je $O(k) = O(1)$. Proměnná *cluster*, a všechny lokální proměnné *c* a *changed* mají dohromady prostorovou složitost $O(3n) = O(n)$. Proměnné *cluster_size* a *cluster_sum* mají dohromady prostorovou složitost $O(1)$. Celková prostorová složitost paralelního algoritmu je tedy $O(n)$. \square

2 Komunikace

Následující sekvenční diagram popisuje komunikaci procesů paralelního algoritmu K-means implementovaného v `parkmeans.cc`. Během výpočtu je využito n procesů. Kořenový proces p_0 přijme požadavek na shlukování bodů *points*. Inicializuje středy shluků a ty pomocí *Broadcast* rozešle ostatním $n - 1$ procesům. Dále je každému procesům rozeslána hodnota *changed = True*. Každému procesu je pomocí *Scatter* přidělen jeden bod. Smyčka (*while*) je opakována, dokud nedojde ke konvergenci (*changed = False*). Ve smyčce kořenový proces p_0 získá pomocí *Reduce* počet a součet bodů v jednotlivých shlucích. K zjištění, zda některý bod změnil shluk využije kořenový proces *ORReduceAll* a výslednou hodnotu *changed* rozešle zpět všem procesům. Aktualizované hodnoty jsou pomocí *Broadcast* opět rozeslány ostatním $n - 1$ procesům. Po ukončení smyčky získá kořenový proces rozdělení bodů do shluků operací *Gather*.



Obrázek 1: Sekvenční graf popisující komunikaci procesu v paralelním programu `parkmeans.cc`.

3 Obsluha programu

Program načítá sekvenci nezáporných čísel (minimálně 4 a maximálně 32) o velikosti 1 byte ze souboru `numbers`. Tento soubor s n čísly lze vytvořit příkazem `dd if=/dev/random bs=1 count=n of=numbers`. Program je nutné spouštět se stejným počtem procesů, jako je počet čísel určených ke shlukování. Středů shluků jsou inicializovány prvními 4 unikátními hodnotami. Pokud takové hodnoty neexistují, vytvoří se duplicitní středů shluku s tím, že hodnoty bodů budou přiřazovány pouze jednomu z nich.

Příklad

Vstup: 1, 10, 50, 1

Výstup:

```

[1.0] 1
[10.0] 10
[50.0] 50
[1.0]

```

4 Závěr

Cílem tohoto projektu bylo implementovat s využitím OpenMPI paralelní shlukovací algoritmus K-means pro jednorozměrná data vyžívající 4 shluky ($K = 4$). Bylo ukázáno, že prostorová složitost sekvenčního i paralelního algoritmu je $O(n)$, kde n je počet shlukovaných bodů. Na základě [1] lze určit, že pro počet shluků menší 5 a jednorozměrná data je časová složitost sekvenčního algoritmu $O(n^2)$. Časová složitost paralelního algoritmu

je $O(n \cdot \log n)$. V případě využití n procesů je cena paralelního algoritmu $O(n^2 \cdot \log n)$. Z těchto poznatků vyplývá, že paralelní algoritmus není optimální.

Reference

- [1] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 144–153, New York, NY, USA, 2006. Association for Computing Machinery.
- [2] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based-clustering. pages 332–339, 06 1994.