Projektna naloga

Vzporedni in porazdeljeni sistemi in algoritmi

Patricio Bulić, Davor Sluga, Rok Češnovar

Jesen 2021

Splošne informacije

Projektno nalogo naredite sami. Rok za oddajo poročil in kode je **24. 1. 2022**.
Za pozitivno oceno iz projektne naloge:

- pripravite sekvenčno različico algoritma, ki bo služila za primerjavo;
- pripravite paralelno različico algoritma z uporabo tehnologije OpenMP ali pthreads in OpenCL;
- naredite primerjalne teste med omenjenimi različicami na superračunalniku NSC za različne velikosti slik, in število barv (gruč);
- napišite poročilo, ki podrobno opisuje vašo rešitev in komentarje rezultatov kot so čas izvajanja, in pohitritve;
- poročilo in izvorno kodo oddajte preko spletne učilnice.

1 Stiskanje slik s pomočjo razvrščanja z voditelji

Vseprisotnost multimedijskih vsebin na spletu močno obremenjuje internetno infrastrukturo [6]. Da bi zmanjšali količino prenesenih podatkov se poslužujemo raznovrstnih postopkov za izgubno [1] in brezizgubno [2] stiskanje takih vsebin. Tukaj si bomo ogledali postopek stiskanja s pomočjo razvrščanja z voditelji [3; 4], ki spada med metode nenadzorovanega učenja.

Razvrščanje z voditelji

Razvrščanje z voditelji je metoda, namenjena razvrščanju vzorcev v skupine. Zaradi svoje enostavnosti in robustnosti je zelo razširjena. Za osnovni algoritem je bilo razvitih tudi precej izboljšav in prilagoditev namenjenih reševanju različnih problemov [5].

Recimo, da imamo podano množico n vzorcev $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n)$, kjer je vsak vzorec D-dimenzionalni vektor. Te vzorce želimo razvrstiti v k skupin ali gruč $\mathbf{C} = \{C_1, C_2, C_3, \ldots, C_k\}$, kjer je $k \leq n$. Poleg same razvrstitve želimo izračunati še centroide $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$, ki predstavljajo povprečen vzorec znotraj posamezne gruče. Razvrščanje v gruče želimo opraviti tako, da minimiziramo raznolikost oziroma varianco znotraj gruče:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \quad . \tag{1}$$

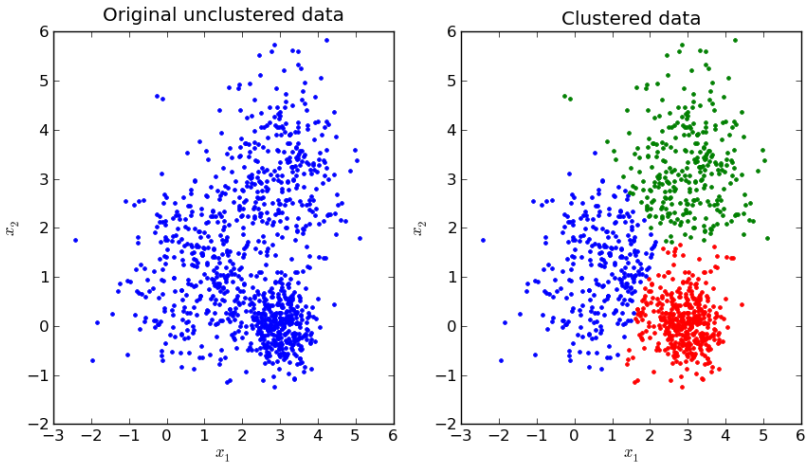
Najbolj običajen algoritem za reševanje tega problema je metoda iterativnega izboljševanja:

Inicializiraj začetne vrednosti centroidov $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$ tako, da jim priredíš naključen vzorec $\mathbf{x}_i$
while $I <$ Iteracij **do**
  for vsak vzorec $\mathbf{x}_i$ **do**
    poišči $\boldsymbol{\mu}_j$ z najmanjšo evklidsko razdaljo do $\mathbf{x_i}$
    nastavi indeks centroida $c_i = j$
  end for
  for vsak centroid $\boldsymbol{\mu}_j$ **do**
    izračunaj povprečje $\mathbf{m}$ vseh $\mathbf{x}_i$, kjer $c_i == j$
    nastavi $\boldsymbol{\mu}_j = \mathbf{m}$
  end for
end while
Polje c vsebuje indekse centroidov za vse vzorce
$\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$ so končne vrednosti centroidov

Slika 1: Algoritem razvrščanja z voditelji.

Število iteracij določimo sami. Mora bit pa dovolj veliko, da vzorci ne prehajajo več med gručami. Pri razvrščanju se lahko zgodi, da katera izmed gruč ostane prazna. Temu

se običajno želimo izogniti. To lahko storimo tako, da prazni gruči pripišemo tisti vzorec, ki je najdlje od svoje trenutne gruče. Da bo pa programska rešitev enostavnejša (sploh na GPU), je bolje, če prazni gruči priredite naključen vzorec, ki se nahaja v eni od drugih gruč.



Slika 2: Primer razvrščanja 2D vzorcev v tri gruče.

Stiskanje slike

Za zapis pikslov neke slike običajno uporabljamo predstavitev RGBA (angl. red, green, blue, alpha) pri čemer uporabimo 8 bitov za zapis intenzitete posameznega barvnega kanala (razpon vrednosti 0–255), skupaj torej 32 bitov na piksel. Število bitov na piksel lahko zmanjšamo, če zmanjšamo število različnih barv v sliki. Naiven pristop je, da enostavno zmanjšamo število razpoložljivih nivojev intenzitete za vsak barvni kanal (na primer iz 256 na 16, kar nam da 16 bitov na piksel) in ustrezno preračunamo nove vrednosti intenzitet. Ta pristop lahko precej pokvari kvaliteto slike. Da bi bolje ohranili informacijo v sliki, je potrebno upoštevati porazdelitev barv v njej.

Ena od možnosti je, da uporabimo razvrščanje z voditelji in podobne barvne odtenke združimo v gruče, katerih centroide (povprečne vrednosti vzorcev znotraj gruče) nato uporabimo kot nadomestno barvo za vse piksle znotraj gruče. V tem primeru so naši vzorci piksli - vektorji v štiri-dimenzionalnem prostoru RGBA. Ciljno število različnih barvnih odtenkov, ki jih želimo uporabiti v sliki, pa je število gruč v katere razvrščamo piksle. S takim pristopom učinkovito zmanjšamo število različnih barv v sliki in posledično njeno velikost ter bolje ohranimo kvaliteto slike. Kanal A (prosojnost) ignoriramo. Samega slikovnega zapisa ni potrebno spreminjati; slika naj bo na disk še vedno zapisana v predstavitvi RGBA in formatu PNG.

(a) Vhodna slika: 32 bitov na piksel.

(b) Izhodna slika: 64 različnih barv (6 bitov na piksel).

Slika 3: Stiskanje slike s pomočjo razvrščanja z voditelji.

Naloga

S pomočjo OpenCL in pthreads/OpenMP napišite vzporedno različico algoritma za stiskanje slik s pomočjo razvrščanja z voditelji. Program naj omogoča stiskanje slik poljubnih velikosti za poljubno število barv. Program naj slike bere in zapisuje v formatu PNG. Za branje in zapisovanje slik lahko uporabite ustrezno knjižnico, na primer STB [7] ali FreeImage [8]. Program naj omogoča izbiro števila niti, ki bodo uporabljene pri procesiranju (OpenMP/pthreads).

Nekaj idej in namigov:

- Algoritem ustrezno preoblikujte, da zmanjšate število dostopov do pomnilnika (vsoto pikslov znotraj posamezne gruče lahko računate hkrati s prirejanjem pikslov gruči). Uporabite atomične operacije kjer je to potrebno.
- Razmislite o možnosti uporabe lokalnega pomnilnika.
- Mogočih je več pristopov k paralelizaciji algoritma. Med niti lahko razdelite vzorce ali pa gruče. V večini primerov je prvi pristop boljši, saj je na voljo veliko več vzorcev kot gruč.
- Algoritem, ki bo tekel na GPE boste morali razdeliti v dva ščepca, saj niti pri računanju povprečne vrednosti pikslov znotraj gruče potrebujejo informacijo o skupnem številu pikslov v gruči.
- Optimizirajte število niti na OpenCL (skupno število niti, število niti na blok).
- Uporabite več grafičnih kartic hkrati.

Literatura

[1] Wikipedia, https://en.wikipedia.org/wiki/Lossy_compression, 2021.

[2] Wikipedia, https://en.wikipedia.org/wiki/Lossless_compression, 2021.

[3] J. MacQueen, Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297), 1967

[4] S. Lloyd, Least squares quantization in PCM. IEEE transactions on information theory Vol. 28, No. 2, pp. 129-137, 1982.

[5] Wikipedia, https://en.wikipedia.org/wiki/K-means_clustering, 2021.

[6] Arnes, https://www.arnes.si/infrastruktura/six-sticisce-omrezij/statistika-prometa/, 2021.

[7] STB library, https://github.com/nothings/stb, 2021.

[8] FreeImage, https://freeimage.sourceforge.io/, 2021