Paralelna implementacija PageRank algoritma upotrebom Intel TBB tehnologija

Predlog projekta za predmet Paralelno programiranje

Autor: Sara Stojkov SV38/2023
Mentor: dr Dušan Kenjić

1. Kratak pregled projekta

Cilj projekta je implementacija paralelne verzije PageRank algoritma koristeći *Intel Threading Building Blocks* (TBB). PageRank algoritam je poznat po tome što je osnova *Google* pretraživača, a naziv je dobio po jednom od njegovih osnivača, *Larry Page*.

PageRank rangira čvorove u grafu na osnovu povezanosti između čvorova i toga koliko su povezani čvorovi (web stranice) „bitni". S obzirom na način na koji veze između stranica funkcionišu, graf je usmeren. Sekvencijalna implementacija ovog algoritma može biti prilično spora za velike grafove, što je motivacija za korišćenje task-based paralelizma, *parallel_pipeline* i konkurentnih struktura koje obezbeđuje TBB kako bi se obezbedilo brže izvršavanje na višejezgarnim procesorima.

2. Programski zahtevi

2. 1. Funkcionalni zahtevi

a) Učitavanje i generisanje grafa
   - Učitavanje grafa iz fajla ili generisanje *random* grafa (za to postoje algoritmi poput *Erdős-Rényi* modela – svaka grana kompletnog grafa ima šansu p da bude uključena u rezultujući graf u graf, $p \in [0,1]$)
   - Predstavljanje grafa kroz listu povezanih čvorova (*adjacency list*) ili matricu povezanosti – za n čvorova to je n x n matrica (*adjacency matrix*)

b) Inicijalizacija
   - Inicijalizacija vektora PageRank vrednosti (gde će biti rezultati) – sve vrednosti su 1/N, pri čemu je N broj čvorova, a zbir rezultata bi trebao da bude 1

c) Iterativno računanje PageRank vrednosti
   - Iteracije dok se ne dostigne konvergencija ili zadati maksimalan broj iteracija
   - Računanje novih vrednosti PageRank-a u paralelnom modu

d) Pipeline model
   - *Parallel_pipline* za organizaciju svake iteracije kroz faze:
     - Generisanje čvorova za obradu (*serial_in_order*)
     - Paralelno računanje doprinosa za dati čvor (*parallel*)
     - Ažuriranje vrednosti u privremenom vektoru (*serial_out_of_order*)

e) Thread-safe operacije
   - Korišćenje TBB konkurentnih kontejnera (npr. *concurrent_vector*) za čuvanje rezultata.

f) Evaluacija rezultata – poređenje serijskog i paralelnog izvršenja

- Ispis bi obuhvatao broj iteracija, vreme izvršavanja, faktor ubrzanja (*speedup factor*) i top 10 čvorova sa najvećim PageRank score-om (kao rezultat pretrage)

2.2. Nefunkcionalni zahtevi

- Efikasno zakazivanje zadataka koristeći TBB *parallel_for*, *parallel_reduce* i *parallel_pipeline*
- Implementacija *adjacency* matrice radi jasnijeg predstavljanja i paralelizacije uz opciju *adjacency* liste za veće grafove
- Analiza ponašanja programa na različitom broju niti i jezgara
- Algoritam mora obraditi grafove sa čvorovima bez visećih čvorova tj. nema čvorova koji nemaju ni jednu izlaznu granu (eng. *dangling nodes*)
- Poređenje performansi sekvencijalnog i paralelnog algoritma uz prikaz speedup faktora

3. Kompleksnost i izazovi

- Predstavljanje grafa:
  - *Adjacency* matrica daje jasnu matematičku formulaciju (matriksko-vektorska multiplikacija)
  - *Adjacency* lista smanjuje memorijsku potrošnju kod većih grafova
- Paralelizacija:
  - Svaki čvor računa svoju novu PageRank vrednost nezavisno od drugih, dakle to su prirodno nezavisni zadaci za obradu (taskovi)
  - Postoji N zadataka po iteraciji, što omogućava dobro skaliranje sa brojem jezgara
- Objedinjavanje rezultata:
  - Po završetku iteracije svi parcijalni rezultati se objedinjuju u novi PageRank vektor vrednosti (double-buffering).
  - *parallel_reduce* prikuplja globalnu grešku radi detekcije konvergencije.
- Pipeline:
  - Organizacija iteracija u faze (generisanje → računanje → skladištenje)
- Balansiranje opterećenja:
  - Čvorovi sa mnogo ulaznih veza zahtevaju više računanja, zbog čega je potrebno ravnomerno rasporediti posao

4. Literatura:

https://acikerisim.bahcesehir.edu.tr/server/api/core/bitstreams/8b157687-1dc7-47b7-9ca9-36563429a2a2/content

https://vldb.org/pvldb/vol14/p1668-wang.pdf

https://www.ijcsit.com/docs/Volume%206/vol6issue03/ijcsit2015060306.pdf

https://web.stanford.edu/class/cs315b/assignment4.html

https://www.geeksforgeeks.org/dsa/erdos-renyl-model-generating-random-graphs/