Implementazione di algoritmi paralleli su GPU per il calcolo d'indici di centralità nell'ambito dell'Analisi delle Reti Sociali

Alma Mater Studiorum · Università di Bologna

Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

Tirocinante	Riccardo Battistini, 0000873514
Tutor didattico	Moreno Marzolla
Laboratorio	High-Performance Computing
Periodo di svolgimento del tirocinio	01/04/21 - 31/05/21

Tavola dei contenuti

1. Introduzione

#F7F7F7 In questa sezione il tirocinante introduce il contesto aziendale o di laboratorio in cui ha lavorato e spiega l'obiettivo del tirocinio.

Reperimento, selezione e sintesi di articoli scientifici e loro valutazione critica. Applicazione di tecniche di parallelizzazione nel contesto della programmazione su GPU.

2. Tecnologie

In questa sezione il tirocinante elenca e commenta brevemente le tecnologie utilizzate (linguaggi, piattaforme, sistemi operativi, ecc).

Durante il tirocinio si è sviluppata un'applicazione a linea di comando accelerata dalla GPU per il calcolo della *betweenness centrality*, una metrica comunemente impiegata nel campo dell'Analisi delle Reti Sociali.

Per realizzare l'applicativo sono stati impiegati i linguaggi C, C++ e CUDA C. In particolare il software realizzato impiega il Compute Unified Device Architecture (CUDA) Toolkit 11.2 [12] e gli standard C99 e C++11. Come sistema per l'automazione dello sviluppo è stato impiegato il software multipiattaforma CMake 3.9. Git 2.3 è stato utilizzato per il controllo di

versione.

Per la realizzazione della documentazione del progetto sono stati impiegati il linguaggio di markup Asciidoc, Gnuplot 5.2 per la creazione dei grafici e il linguaggio Bibtex per la gestione della bibliografia.

Come formati per la memorizzazione dei dati sono stati impiegati Matrix Market, standard per la rappresentazione di matrici sparse [1], e CSV per memorizzare i risultati della valutazione delle prestazioni e della scalabilità.

Infine per verificare il funzionamento dell'applicazione sono stati impiegati degli script in Bash 4.4.2, per la raccolta dei risultati, e Python 3.8, per processare i dati ottenuti.

L'applicazione è stata sviluppata e testata solo sul sistema operativo Ubuntu 18.04. In ogni caso il funzionamento dovrebbe essere garantito senza problemi anche su altre distribuzioni GNU/Linux, MacOS e Windows, a patto che siano soddisfatte le dipendenze.

3. Attività

Questa sezione, eventualmente frazionata in sottosezioni, è quella centrale e più corposa dell'elaborato e deve illustrare (dal punto di vista tecnico, non necessariamente cronologico) le attività svolte durante il tirocinio.A eccezione di eventuali esempi, non deve includere il codice sviluppato; per descrivere algoritmi si usino piuttosto pseudo-codice e/o diagrammi di vario tipo.Per lavori di tipo progettuale è utile includere una sintetica documentazione formale di progetto.

I risultati sono testati su set di dati sintetici e reali e discussi in Sezione 3.6.

Di seguito si assume che un grafo sia descritto come $G = (V, E)$, dove V è l'insieme dei vertici ed E l'insieme degli archi. Si impiegano n ed m rispettivamente per indicare il numero di vertici e di archi. Infine si considerano solo grafi connessi non diretti e non pesati.

Per maggiori informazioni si consultino [11, 10, 4].

3.1. Misure di centralità e betweenness

Le misure di centralità permettono di misurare l'importanza di un nodo in base a diversi criteri, come la sua posizione nella rete, e costituiscono uno

strumento fondamentale per l'analisi delle reti sociali.

Una delle più note e diffuse misure di centralità è la *betweenness centrality* [6, 8]. Si tratta di una misura della proporzione dei cammini minimi di una rete che attraversano uno specifico nodo.

In base alla classificazione proposta da Freeman [6, 7] si distinguono tre tipi di misure di betweenness centrality a seconda che si stia considerando la betweenness centrality di un singolo vertice normalizzata, non normalizzata o dell'intero grafo.

La definizione di betweeness centrality si basa sul concetto di *pair dependency* δ_{st} di s su t. Quest'ultima è definita come:

$$\delta_{st}(v) \overset{def}{=} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Dove $\sigma_{st}(v)$ indica il numero di cammini minimi che passano da s a t e attraversano v e σ_{st} indica il numero di cammini minimi che passano per ogni coppia $s, t \in V$.

$$BC(v) \overset{def}{=} \sum_{s \neq v \neq t \in V} \delta_{st}(v) \qquad (1)$$

$$BC(v) \overset{def}{=} \sum_{s \neq v \neq t \in V} \frac{\delta_{st}(v)}{\delta_{st}} \qquad (2)$$

La betweenness centrality si presta a diverse interpretazioni. Può esprimere la capacità di un nodo di controllare, distorcere, inibire o bloccare lo scambio d'informazioni in una rete oppure può indicare quanto un nodo possa unire due o più gruppi di nodi e favorire lo scambio d'informazioni.

La betweenness centrality è di particolare interesse anche perché è alla base di algoritmi più complessi come la *Community Detection* e il suo calcolo efficiente in parallelo non è facile da realizzare.

3.2. Algoritmo seriale di Brandes

I primi algoritmi introdotti per il calcolo della betweenness centrality sono piuttosto onerosi, [6, 8] eseguono in tempo $\theta(n^3)$ e richiedono $\theta(n^2)$ in termini di spazio.

La determinazione della betweenness centrality avviene in due passi:

1. Calcolo della lunghezza e del numero dei cammini minimi tra tutte le

coppie di vertici;

2. Somma delle dipendenze di tutte le coppie.

L'algoritmo sequenziale introdotto da Brandes [2], rappresenta un notevole miglioramento rispetto ai precedenti in quanto esegue in tempo $\theta(nm)$ in grafi non pesati e richiede una minore occupazione di memoria. Come evidenziato nell'Algoritmo 1, la betweenness centrality (3) è calcolata come somma delle *dipendenze*:

Dove $\delta_{st}(v)$ indica la dipendenza di s su v.

Algoritmo 1. Calcolo della betweenness centrality di Brandes

```
1 procedure BC(G)
2     for v in G.V
3         label v.bc as 0
4     for v in G.V
5         let Q be a queue
```

L'algoritmo di Brandes impiega una tecnica di accumulazione che si integra con la risoluzione del problema dei cammini minimi tramite algoritmi di attraversamento dei grafi. Applicando il risultato di Brandes, si ha che tutte le misure di centralità che richiedono il calcolo dei cammini minimi possono essere computate simultaneamente. Esempi di misure di questo tipo sono la *closeness centrality* [13] e la varianti della betweenness centrality [3], come la *load centrality* e la *stress centrality* (APP ?).

Nell'algoritmo di Brandes per il calcolo dei cammini minimi in un grafo non pesato si impiega una visita in ampiezza (BFS), ovvero l'Algoritmo 2. Per l'implementazione si è consultato [4 p. 497]. Il tempo richiesto dalla BFS è pari a $\theta(m)$. Di conseguenza il tempo richiesto per il calcolo di δ_{st}, $s, t \in V$ è pari a $\theta(nm)$.

Algoritmo 2. Visita in ampiezza

```
 1 procedure BFS(G, s)
 2     for v in G.V - {s} ①
 3         label v.c as White
 4         label v.d as nil
 5     label s.c as Grey
 6     label s.d as 0
 7     let Q be a queue
 8     Q.enqueue(s) ②
 9     while Q is not empty
10         u := Q.dequeue() ③
11         for v in G.adjacentVertices(u) ④
12             if v.c is not Grey ⑤
13                 label v.c as Grey
14                 label v.d as u.d + 1
15                 Q.enqueue(v)
16         label u.c as Black ⑥
```

① Inizializza ciascun vertice del grafo eccetto l'origine;

② Scopre il vertice d'origine;

③ Esamina il vertice `u`;

④ Esamina l'arco `(u,v)`;

⑤ Scopre il vertice `v`;

⑥ Rimuove il vertice `u`.

3.3. Shortest Path Vertex Betweenness

3.4. Edge and Node parallelism

[9]

3.5. Implementazione di un algoritmo parallelo su GPU

3.6. Studio di scalabilità e throughput

Come visto nell'introduzione agli algoritmi di Cormen, Leiserson e Rivest [4]...]

Per i test sono stati impiegati una CPU Intel Core i7-10700 con frequenza di funzionamento pari a 2.9 Ghz, una cache di 16 Mb e 16 Gb di DRAM.

La GPU è una Quadro P620 con quattro Streaming Multiprocessors e clock di base di 2505 Mhz. Ci sono due Gb di memoria GDDR5 a disposizione e la *compute capability* è pari a 6.1 (architettura Pascal).

I dataset per effettuare i test sono stati presi dalla Sparse Matrix Collection dell'Università della Florida [5] e dalla Standford Network Analysis Platform (SNAP).

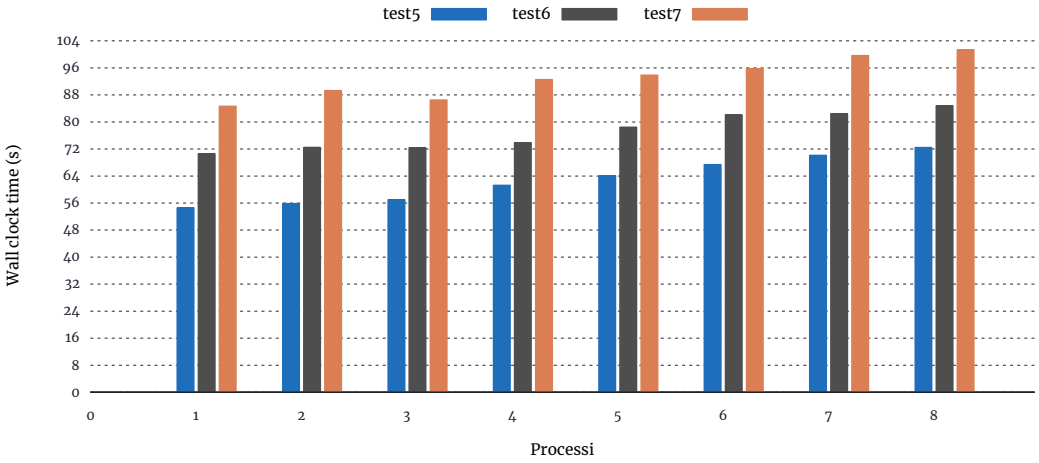

Fig. 1. Confronto delle prestazioni, definite in termini di speedup, della versione OpenMP con alcuni degli insiemi di dati forniti.

4. Conclusioni

In questa parte lo studente trae le conclusioni del lavoro svolto, valutando pregi e difetti dell'esperienza e, più specificamente, riassumendo quanto appreso.

Il software sviluppato è liberamente disponibile in un repository su GitHub.

Riferimenti bibliografici

[1] R. Boisvert, R. Pozo, and K. Remington, "The Matrix Market Exchange Formats: Initial Design," *NISTIR*, vol. 5935, Dec. 1996.

[2] U. Brandes, "A faster algorithm for betweenness centrality*," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, Jun. 2001, doi: 10/fn726f.

[3] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, May 2008, doi: 10.1016/j.socnet.2007.11.001.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[5] T. A. Davis and Y. Hu, "The university of Florida sparse matrix collection," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011, doi: 10.1145/2049662.2049663.

[6] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, p. 35, Mar. 1977, doi: 10/btcpw5.

[7] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, Jan. 1978, doi: 10/bx3m36.

[8] J.M. Anthonisse, "The rush in a directed graph," *Stichting Mathematisch Centrum*. Jan-1971.

[9] Y. Jia, V. Lu, J. Hoberock, M. Garland, and J. C. Hart, "Chapter 2 - Edge v. Node Parallelism for Graph Centrality Metrics," in *GPU Computing Gems Jade Edition*, W.-mei W. Hwu, Ed. Boston: Morgan Kaufmann, 2012, pp. 15–28.

[10] M. Lambertini, M. Magnani, M. Marzolla, D. Montesi, and C. Paolino, "Large-Scale Social Network Analysis," in *Large-Scale Data Analytics*, A. Gkoulalas-Divanis and A. Labbi, Eds. New York, NY: Springer New York, 2014, pp. 155–187.

[11] M. Magnani and M. Marzolla, "Path-Based and Whole-Network Measures," in *Encyclopedia of Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds. New York, NY: Springer New York, 2017, pp. 1–16.

[12] NVIDIA, P. Vingelmann, and F. H. P. Fitzek, "CUDA, release: 11.2.142." 2020.

[13] G. Sabidussi, “The centrality index of a graph,” *Psychometrika*, vol. 31, no. 4, pp. 581–603, Dec. 1966, doi: 10/fqhhrk.