Implementazione di Finding Triangles con Hadoop MapReduce

Sistemi di elaborazione di grandi quantità di dati 2016

Nicola Febbrari
Università degli Studi di Verona
Facoltà di Scienze
nicola.febbrari@studenti.univr.it

3 maggio 2017

1 Introduzione

Lo scopo del progetto è quello di implementare un algoritmo per calcolare il numero di triangoli presenti in un grafo non diretto, utilizzando le tecniche di MapReduce e il Framework Haddop.

2 Il problema

I social network negli ultimi anni hanno avuto una notevole diffusione, l'aumento esponenziale del numero di utenti che interagistcono con questi sistemi ha avuto come diretta conseguenza un incremento della quantità di dati che devono essere registrati, gestiti ed ovviamente elaborati. Un social network può essere rappresentato matematicamente da un grafo e una caratteristica molto interessante di questo grafo e' il numero di triangoli[1] contenuti in sesso. Il numero di questi triangoli rapportato al numero totale di triangoli che esisterebbero con una distribuzione casuale e uniforme delle relazioni può essere un indice di quanto sia social il grafo analizzato.

3 Strumenti e Framework

Per progettare, implementare e testare il programma ho utilizzato:

Infrastruttura Per semplicità e rapidità di configurazione ho deciso di utilizzare Claudera come distribuzione di Hadoop nella sua versione per Docker.

[1] Dati 3 nodi (A,B,C) in un grafo, se un nodo A si relaziona sia con B che con C, nel grafo viene a formarsi un triangolo se esiste anche la relazione che lega B con C.

Sviluppo Dovendo utilizzare il Framework Hadoop è il programma è stato scritto in Java utilizzando le API di Hadoop 2.6
L' IDE di sviluppo utilizzato è IntelliJ.

Versioning Come sistema di versioning ho utilizzato Git e GitHub come spazio di hosting dei files sorgenti.

4 Implementazioni

4.1 Algoritmo 2 Jobs

L'algoritmo a 2 Jobs prevede un prima fase in cui vengono elaborate tutte le relazioni e viene creata in output la combinazione di tutti i possibili triangoli a meno dell'ultimo arco necessario per chiudere il triangolo.

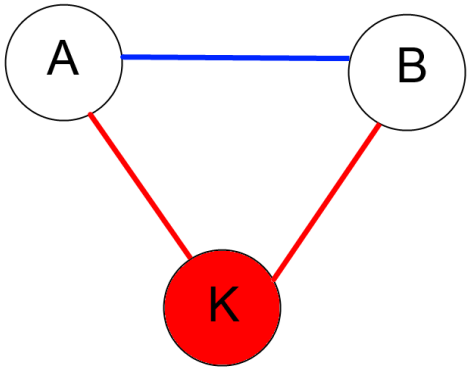

Figura 4.1: *Il Job 1 elabora le relazioni rosse in cui K è il nodo minore del triangolo. Il Job2 cerca la relazione blu A B.*

Job1 Mapper Nella fase di mapper del primo Job viene caricato il file di testo e per ogni relazione contenuta in esso viene emessa in output una copia di valori in cui la chiave è il nodo minore e come valore il nodo maggiore. L'ordinamento presente del file di testo non viene valutato per in questo caso stiamo lavorando su grafi non diretti.

Job1 GroupComparator Il GroupComparator raggruppa tutte le coppie generate dal mapper e per ogni possibile chiave crea una lista con i valori delle coppie aventi la stessa chiave

Job1 Reducer Il reducer elabora l'input raggruppato dal GroupComparator e per ogni valore in cui K è chiave costruisce una lista L in cui aggiunge tutti i valori della lista abbinati a K. Quando la costruzione della lista è terminata e tutti i valori abbinati alla chiave K sono stati inseriti esegue il processo di scrittura dell'output delle coppie chiave-valore in cui la chiavi sono date dalla combinazione di tutte le possibili coppie di valori presenti in L, mentre il valore è il nodo K.

Job2 Mapper Nella secondo Job l'output del primo Job viene unito nuovamente alla sorgente dati iniziale. Il secondo mapper scorre l'input, nel caso in cui la relazione è una relazione presente nel grafo iniziale scrive in output un elemento ((A,B,false),0) se invece è generata dal Job 1 scrive in output ((A,B,true),K) dove A,B sono nodi della relazione e K è nodo minore che rappresenta il valore della coppia chiave-valore generata dal Job1.

Job2 Reducer Il secondo reducer ridefinendo in modo analogo al Reducer1 una seconda versione del SortComparator e del GroupingComparator, scorre l'input prima cerca le relazioni AB che chiuderebbero un triangolo e poi per ogni relazione generata dal Job1 con output AB manda in output KAB. Questo algoritmo se dal punto di vista implementativo è stato estremamente semplice tuttavia testato con grafi ad elevato numero di nodi e con una grande densità di triangoli si è rivelato inefficiente.

4.2 Algoritmo 4 Jobs

L'algoritmo a 4 Jobs prevede prima una fase in cui i primi 2 Jobs costuiscono una struttura dati che poi verrà usata negli ultimi 2 Jobs, i quali andranno ad eseguire il calcolo dei triangoli.

Job 1 Il primo Job è molto semplice e per ogni arco emette un valore 1. Il redoucer conta tutti questi valori, ne fa la somma che poi scrive come output.

Job 2 Il mapper del secondo Job per ogni arco (A,B) emette 2 valori (A 1 e B 1) un per ogni nodo che compone (A,B).
Le funzioni di grouping e il partitioner mi garantiscono che tutti i valori di uno stesso nodo vengono accumulati nello stesso redoucer e nello stesso ciclo iterativo. Completato il ciclo scrive in output per ogni nodo la somma di tutti i valori emessi dal mapper. Questo valore rappresenta il grado del nodo.

Job 3 Nella seconda fase viene eseguito l'algoritmo vero e proprio.
Nel Job 3 l'algoritmo estrapola tutti quei triangoli che sono composti da nodi Heavy Hitter, ovvero con un grado maggiore alla radice di m dove m è il numero degli archi delgrafo. Il mapper utilizzando il lavoro dei Jobs precedenti costruisce un indice con i gradi di ogni nodo e successuvamente elimina dall'analisi tutti i nodi non HH.
Con questo sottografo costruisce una suddivisione delle relazioni definendo delle chiavi strutturate e abbinate ai vari task di Reduce. Questa suddivisione viene realizzata

definendo una funzione Hash sul valore del nodo stesso.
Come prima cosa viene definita una relazione $\prec$ di ordinamento dei nodi in base al grado. Ogni trinagolo è formato da 3 nodi (x,y,z) i quali sono legati dalle 3 relazioni typeA=(x,y) con x$\prec$y, typeB=(x,z) con x$\prec$z e typeC=(y,z) ocn y$\prec$z. Ogni relazione presente nel grafo può essre una delle 3 che compongono il triangolo, quindi dato R=(u,v), R può essere una relazione typeA e quindi il triangolo sarebbe (u,v,?) oppure typeB con (u,?,v) o typeC con (?,u,v). Il task di Map per sfruttare il parallelismo dei redoucer divide le relazioni in tante parti quante le possibili casistiche in cui la relazione può essere utilizzata per completare un triangolo.
Per implementare questa suddivisione, come prima cosa, viene definito un paramentro b che indica quanti sono i possibili valori in output della funzione hash. In base a questo parametro ogni relazione di input R(x,y) viene distribuita nei rispettivi 3bk Reducers utilizzando la funzione di hash h e seguendo questo schema: (h(x),h(y),1$\leq$i$\leq$b) impotizzando che R sia di tipo A , (h(x),1$\leq$i$\leq$b,h(y)) se R fosse di tipo B e (1$\leq$i$\leq$b,h(x),h(y)) se R fosse di tipo C. Ogni relazione R viene inclusa in tutti i possibili reducer in cui potrebbe essere utilizzata per il completamento di un triangolo. Il reducer, doppo una opportuna ridefinizione delle classi di Gruouping e Ordinamento, scorrere tutti gli elementi in input. L'ordinamente è definito in modo da analizzare, per ogni nodo a, prima le possili relazioni di tipo A, per ognuna di esse viene inserito il nodo destinazione b in una Lista LAa. Successivamente vengono analizzate le relazioni di a di tipo B con c come nodo di destinazione, per ognuna di esse viene preso creata una Map in cui la chiave è la copia formata da ogni elemento di Laa e c e il valore è il nodo a. L'analisi successiva della relazione di tipo C fra a e d, se nella Map esiste una chiave formata dalla coppia a,d con valore e allora nel grafo esisterà un triangolo e,a,d. L'ordinamento con cui viene eseguita questa analisi consente di ottimizzare lo spazio di memoria utilizzato e ci garantisce che ogni triangolo viene rilevato solo una volta.

Job 4 Il Job 4 è molto simile al Job3 l'unica differenze è che in questo caso si volgiono escludere tutti i triangoli di tipo HH.
Data una relazione x,y se x è HH x,y potrebbe appartenere ad un triangolo non HH se solo se x,y è di tipo C. Infatti per la relazione di ordinamento rel definita precedentemente dato il triangolo T1(x,y,?) in cui x,y è di tipo A, qualsiasi ? e y sarebbero maggiori di x e quidi T1 sarebbe HH, la stessa cosa se x,y fosse di tipo B.
Grazie a questa caratteristica il mapper esclude le relazioni di tipo A e B in cui x,y ha x HH. Il reducer è lo stesso del Job4.

5 Valutazioni

Algoritmo 2 Jobs Semplice ma ha evidenziato problemi analizzando grafi di grandi dimensioni.

Algoritmo 4 Jobs Più complesso ottimizzato dal punto di vista del tempo, tuttavia presenta sempre dei problemi nell'utilizzo della memoria. Sicuramente è possibile fare

delle ottimizzazioni utilizzando delle strutture dati di opportune.
Nella mia implementazione ho cercato di utilizzare le strutture dati che mi permettevano di avere tempi di accesso rapidi e che non utlizzassero troppa memoria. Purtroppo ho notato che eseuendo più volte il programma rimane allocata della memoria, questo mi fa presupporre che vi sono alcuni memory leak.

6 Sviluppi futuri

Oltre alle ottimizzazioni nell'utilizzo di strutture dati più performanti, mi sarebbe piaciuto generare come artefatto del processo un immagine dock pronta all'uso. Sarebbe interessante provare a fare un deploy per testare questo container in un ambiente cloud che supporta Docker.