

Analisi di dati su larga scala

Prof. Malchioni dario
5 CFU

Luca Cappelletti

Lectures Notes
Year 2017/18



Magistrale Informatica LM-18
Università statale di Milano
Italy
October 7, 2017

Contents

1	Chapter 2	2
1.1	Analisi di complessità di un job map-reduce	2
1.1.1	Es: moltiplicazione di matrici	2
1.1.2	Join multi-way	2
1.1.3	Rilassamento lagrangiano	2
1.1.4	Es: Join sui nodi di facebook	3
1.1.5	Es: Google pagerank	3

Chapter 1

Chapter 2

1.1 Analisi di complessità di un job map-reduce

1.1.1 Es: moltiplicazione di matrici

$$A_{m \times n} \times B_{n \times o}(i, j, a_{ij}) \mapsto M_A((i, j), (A, k, a_{ik})) \forall j = 1, \dots, o(k, j, b_{kj}) \mapsto M_B((i, j), (B, k, b_{ik})) \forall i = 1, \dots, m(i, j)[(A, 1, a_{i1}), \dots, (A, m, a_{im})]$$

$$R(A, B) \bowtie S(B, C) \bowtie T(C, D)(a, b) \mapsto_{M_R} (b, (R, a))(b, c) \mapsto_{M_S} (b, (S, a))$$

Quale è il costo di questo algoritmo. Indichiamo con r, s, t i rispettivi numeri di tuple delle tabelle R, S, T. Il primo processo M_R riceve tutte e sole le tuple di R, quindi ha costo r . Il secondo, similmente, ha costo s .

Il risultato del costo di complessità sarà quindi un $O(r + s)$. Ma questo è tra due relazioni. Se volessi farlo da 3 relazioni (**join in cascata**) cosa andrei ad ottenere?

$$(R \bowtie S) \bowtie T$$

Otengo il costo $O(r + s + t + rsp)$, con p rappresentante la probabilità che due valori di R e S hanno un attributo uguale.

1.1.2 Join multi-way

Date due funzioni di hash, una h per l'attributo B ed una g per l'attributo C, con b **bucket** e c **bucket**, avendo che $bc = k$.

Nel caso di una tupla $(u, v) \in R$ viene inviata ad un'unica colonna verticale, riducendo i nodi (**c reducer**).

Nel caso di una tupla $(w, z) \in T$ viene inviata ad un'unica colonna orizzontale, riducendo i nodi (**b reducer**).

Nel caso di una tupla $(v, w) \in S$ viene inviata ad un'unica cella, riducendo i nodi ad uno soltanto (**1 reducer**).

Il costo quindi risulta essere:

$$O(r + 2s + t + cr + bt)$$

1.1.3 Rilassamento lagrangiano

N.B. Il parametro lambda non può essere negativo.

$$L(b, c) = cr + br - \lambda(bc - k)$$

$$\frac{dL(b, c)}{db} = 0$$

$$\frac{dL(b, c)}{dc} = 0$$

Ottengo quindi un sistema:

$$\begin{cases} t - \lambda c = 0 \\ r - \lambda b = 0 \end{cases} \implies \begin{cases} t = \lambda c \\ r = \lambda b \end{cases}$$

$$\lambda = \sqrt{\frac{rt}{k}}$$

$$c = \sqrt{\frac{kt}{r}}$$

$$b = \sqrt{\frac{kr}{t}}$$

Il **costo ottimizzato** della **join multiway** risulta quindi essere:

$$O(r + 2s + t + 2\sqrt{kr t})$$

1.1.4 Es: Join sui nodi di facebook

Prendiamo ad esempio il grafo dei nodi facebook, dotato di 10^9 nodi.

$R(U_1, U_2), |R| = r = 3 \times 10^1$ (dati arbitrari)

$$R \bowtie R \bowtie R$$

Approccio Multi-way: $r + 2r + r + 2r\sqrt{k} = 4r + 2r\sqrt{k} = 1 \times 2 \times 10^1 2 + 6 \times 10^1 1\sqrt{k}$

Approccio cascata (nell'ipotesi che $absR \bowtie R = 30r$): $r + r + r + r^2 \times p = \dots = 2r + 60r = 1 \times 2 \times 10^1 2 + 1 \times 86 \times 10^1 3$

Ottengo quindi che: $6 \times 10^1 1\sqrt{k} \leq 1 \times 86 \times 10^3 3 \implies k \leq 961$, e risulta quindi migliore utilizzare l'approccio multi way quando si hanno meno di 961 nodi da allocare a dei reducer.

1.1.5 Es: Google pagerank

Come funziona **pagerank**:

	A	B	C	D
A	0	$\frac{1}{2}$	1	0
B	$\frac{1}{3}$	0	0	$\frac{1}{2}$
C	$\frac{1}{3}$	0	0	$\frac{1}{2}$
D	$\frac{1}{3}$	$\frac{1}{2}$	0	0

$$v_j(t+1) = P(\text{Trovarsi in } j \text{ al tempo } t+1) = \sum_i P(\text{trovarsi in } i \text{ al tempo } t) \cdot P(\text{spostarsi da } i \text{ a } j \mid \text{trovarsi in } i \text{ al punto } t)$$

$$\sum_i v_i(t) m_{ji} = \sum_i m_{ji} v_i(t) = (Mv(t))_j$$

con $M_{ij} = P(\text{spostarsi da } j \text{ a } i)$.

$$\vec{V}(t+1) = M\vec{v}(t)$$