Esercitazioni di Calcolo Parallelo: Map Reduce

Paolo Avogadro

DISCo, Università di Milano-Bicocca
U14, Id&aLab T36 -WEBEX
paolo.avogadro@unimib.it
Aule Lezione T014, edificio U14
Ore: Martedi' 13:30 - 15:30
Ore: Mercoledi: 8:30 - 10:30

Esempio di prodotto Matrice-Matrice

Qui viene descritto un prodotto **matrice-matrice** particolarmente adatto a matrici sparse. In generale vogliamo fare il seguente prodotto:

$$C = A \cdot B \tag{1}$$

Uso qui la seguente notazione,

- nome della matrice in maiuscolo senza indici (A, B, C...)
- sottoblocchi di matrice in maiuscolo con la stessa lettera della matrice madre ma con indici che specificano il blocco (in quanto sono matrici) (A_{lm}..);
- elementi di matrice (edm) in minuscolo (a_{ij}, b_{kl}, ...).

Ogni elemento di C si ottiene tramite la formula classica:

$$c_{ij} = \sum_k a_{ik} b_{kj} \tag{2}$$

Quindi dal punto di vista delle operazioni da eseguire (considerando il caso peggiore di matrici completamente prive di zeri), e' fatto in questo modo. Per ogni valore di C devo eseguire N prodotti (dove N e' il numero di colonne di A e di righe di B) e devo eseguire una somma con N termini. Se suppongo che $A_{M \times N}$ e $B_{N \times L}$ allora serviranno: $M \times L \times N$ prodotti. In pratica se $M = L = N$ (matrici quadrate) allora servono N^3 prodotti e N^3 somme
Idee prese da: `http://www.norstad.org` (occhio temo che il suo sito non esista piu' o sia stato hackerato!)

Formati ingresso

Bisogna tenere in considerazione come sono i **formati** dei dati che arrivano ai mapper e reducer in modo che possano essere emesse coppie chiave-valore tali da rendere il calcolo comodo.
Per fare il prodotto matrice-matrice con MapReduce, ha senso prima notare che normalmente le matrici sono memorizzate per righe.
Proviamo a considerare un caso in cui si hanno delle matrici sparse.
Cos'e' una matrice sparsa? Matrice in cui la maggior parte degli ingressi e' zero.
Se noi dovessimo inserire tutti gli ingressi delle matrici sparse dovremmo passare molti 0, e questo risulta inefficiente. Una soluzione alternativa potrebbe essere ottenuta con il seguente formato, in cui vengono passare delle righe con:

- se una riga e' vuota, non appare nel file
- se una riga contiene almeno un elemento non nullo, allora verra' scritto un intero, che indica il numero di riga
- nel seguito ci sarnno coppie intero reale, dove l'intero definisce la colonna, e il reale contiene il valore dell'elemento di matrice.

```
0  0  0.43243   3 1.3443  # riga 0 ha elementi colonna non nulli lo 0 e il 3
1  3  2.334     4 3.24    # riga 1 solo nella colonna 3 e il 4 ci sono elementi non nulli
                          # riga 2 ha tutti gli elementi nulli, quindi non c'e'!
3  1  734.6                # in riga 3, l'unico elemento non nullo, e' quello della colonna 1
```

Supponiamo quindi che sia A che B sono memorizzate in questo modo

Passi necessari per un prodotto MapReduce

In questa implementazione servono 3 passi (1 map e 2 reduce):

1. allineare su **colonne** le matrici A e B (vedi dopo cosa significa)
2. fare il prodotto $a_{ik}\,b_{kj}$ per ogni ingresso individuato da una coppia (i, j) (quest'ultima e' la nuova chiave da mandare nel secondo reducer).
3. fare la somma di tutti i prodotti $a_{ik}\,b_{jk}$ che condividono la stessa chiave (i, j).

Per quanto riguarda il punto 1) (allineamento), nota che se prendo l'elemento a_{00} devo moltiplicarlo per l'elemento b_{00}, b_{01}, ..,b_{0N}. Questo perche' la prima riga deve essere moltiplicata per la prima colonna, ma anche per la seconda colonna, e cosi' via... ovvero:

- l'elemento a_{00} deve essere moltiplicato per tutti i b_{0k} $(k = 0, .., N-1)$
- anche l'elemento a_{10} deve essere moltiplicato per tutti i b_{0k} $(k = 0, .., N-1)$
- anche l'elemento a_{20} deve essere moltiplicato per i b_{0k} $(k = 0, .., N-1)$
- $\vdots$

Quindi tutti gli a_{k0} vanno moltiplicati per tutti i b_{0k}. Questo significa che tutti gli elementi della prima colonna di A vanno moltiplicati per tutti gli elementi della prima riga di B!

Similmente gli tutti gli elementi della seconda colonna di A (a_{k1}) vanno moltiplicati per tutti gli elementi della seconda riga di B (b_{1k}). In totale ci sono N di queste relazioni: sono i prodotti esterni delle colonne di A e delle righe di B che contengono tutti i prodotti necessari per calcolare il prodotto matrice matrice!

prodotto esterno 1

Visualizzo qui gli N prodotti esterni tra le prime N colonne di A e N righe di B:

$$\begin{pmatrix} a_{00} & \cdot & \cdot & \cdot \\ a_{10} & \cdot & \cdot & \cdot \\ a_{20} & \cdot & \cdot & \cdot \\ a_{30} & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Questo prodotto esterno da luogo a 16 prodotti:

$$= \begin{pmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{00}b_{02} & a_{00}b_{03} \\ a_{10}b_{00} & a_{10}b_{01} & a_{10}b_{02} & a_{10}b_{03} \\ a_{20}b_{00} & a_{20}b_{01} & a_{20}b_{02} & a_{20}b_{03} \\ a_{30}b_{00} & a_{30}b_{01} & a_{30}b_{02} & a_{30}b_{03} \end{pmatrix}$$

Domanda: come sono gli indici interni delle coppie di elementi di matrice?
Risposta: sono identici per costruzione (e questo risulta utile nel momento in cui si ricorda che $c_{ij} = \sum_k a_{ik} b_{kj}$)

prodotto esterno 1bis

$$\begin{pmatrix} a_{00} & . & . & . \\ a_{10} & . & . & . \\ a_{20} & . & . & . \\ a_{30} & . & . & . \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

$$= \begin{pmatrix} a_{00}\,b_{00} & a_{00}\,b_{01} & a_{00}\,b_{02} & a_{00}\,b_{03} \\ a_{10}\,b_{00} & a_{10}\,b_{01} & a_{10}\,b_{02} & a_{10}\,b_{03} \\ a_{20}\,b_{00} & a_{20}\,b_{01} & a_{20}\,b_{02} & a_{20}\,b_{03} \\ a_{30}\,b_{00} & a_{30}\,b_{01} & a_{30}\,b_{02} & a_{30}\,b_{03} \end{pmatrix}$$

prodotto esterno 2

$$\begin{pmatrix} . & a_{01} & . & . \\ . & a_{11} & . & . \\ . & a_{21} & . & . \\ . & a_{31} & . & . \end{pmatrix} \begin{pmatrix} . & . & . & . \\ b_{10} & b_{11} & b_{12} & b_{13} \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

$$= \begin{pmatrix} a_{01}b_{10} & a_{01}b_{11} & a_{01}b_{12} & a_{01}b_{13} \\ a_{11}b_{10} & a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} \\ a_{21}b_{10} & a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} \\ a_{31}b_{10} & a_{31}b_{11} & a_{31}b_{12} & a_{31}b_{13} \end{pmatrix}$$

prodotto esterno 3

$$\begin{pmatrix} \cdot & \cdot & a_{02} & \cdot \\ \cdot & \cdot & a_{12} & \cdot \\ \cdot & \cdot & a_{22} & \cdot \\ \cdot & \cdot & a_{32} & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ b_{20} & b_{21} & b_{22} & b_{23} \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$= \begin{pmatrix} a_{02}b_{20} & a_{02}b_{21} & a_{02}b_{22} & a_{02}b_{23} \\ a_{12}b_{20} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ a_{22}b_{20} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \\ a_{32}b_{20} & a_{32}b_{21} & a_{32}b_{22} & a_{32}b_{23} \end{pmatrix}$$

prodotto esterno 4

$$\begin{pmatrix} \cdot & \cdot & \cdot & a_{03} \\ \cdot & \cdot & \cdot & a_{13} \\ \cdot & \cdot & \cdot & a_{23} \\ \cdot & \cdot & \cdot & a_{33} \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ b_{30} & b_{31} & b_{32} & b_{33} \end{pmatrix}$$

$$= \begin{pmatrix} a_{03}b_{30} & a_{03}b_{31} & a_{03}b_{32} & a_{03}b_{33} \\ a_{13}b_{30} & a_{13}b_{31} & a_{13}b_{32} & a_{13}b_{33} \\ a_{23}b_{30} & a_{23}b_{31} & a_{23}b_{32} & a_{23}b_{33} \\ a_{33}b_{30} & a_{33}b_{31} & a_{33}b_{32} & a_{33}b_{33} \end{pmatrix}$$

Osservazioni sul prodotto esterno

- Domanda: Quanti prodotti vengono eseguiti tra elementi di matrice se si fanno i prodotti esterni ?
- Risposta: Ogni volta che si esegue un prodotto esterno si fanno $N \times N$ prodotti. Poiche' il numero di prodotti esterni da eseguire e' N (uno per ognuna delle righe della matrice), il numero totale di prodotti tra singoli elementi delle matrici e' $N \times N \times N = N^3$.
- Domanda: Quanti prodotti vengono eseguiti tra elementi di matrice se si procede in modo “pedissequo”, con prodotti riga per colonna?
- Risposta: N^3 (usiamo dei trucchi, non miracoli...)

Map, Reduce 1, Reduce 2: Map

Torniamo a MapReduce, e ricordiamoci che gli elementi delle matrici A e B vengono mandati a una funzione che fa da mapper (**RICORDA** ho **una sola** funzione che deve andare bene per tutti gli elementi di matrice) Il mapper riceve in ingresso righe nel formato che abbiamo visto sopra, in aggiunta pero' deve poter distinguere se stiamo lavorando con la matrice A o con la B.
La funzione di **Map**, non fa nessun calcolo, semplicemente serve a costruire un nuovo formato con le corrette chiavi intermedie.
Nel dettaglio emette coppie chiave-valore del seguente formato:
- per gli elementi di matrice **A**:

- come chiave l'indice di **colonna**
- come valore una terna contentente: indice **riga**, valore elemento, e poi un indice (bit 0) che dica che l'elemento proviene dalla matrice A.

Per esempio per l'elemento di matrice $a_{24} = 4.23$ verra' emessa come chiave 4 e poi la terna $(2, 4.23, 0)$
In notazione Hadoop streaming: `4 TAB (2, 4.23 ,0).`
- Per gli elementi di matrice **B**:

- come chiave l'indice di **riga**
- come valore la terna: indice di **colonna**, valore dell'elemento di B e un indice che dica che e' matrice B (1 bit).

Per esempio se il mapper riceve $b_{63} = 0.34$: la chiave e' 6 il valore e' la terna $(3, 0.34, 1)$. Chiaramente gli elementi di questo esempio non vanno allo stesso reducer, mentre se l'elemento di B fosse stato: $b_{47} = 1.72$: la chiave sarebbe stata 4 il valore la terna $(7, 1.72, 1)$, o in notazione Hadoop: `4 TAB (7, 1.72, 1)` . Questo secondo elemento di B viene inviato tramite lo *shuffle* allo stesso *reducer(1)* del primo esempio di elemento di A ($a_{24} = 4.23$).

Visualizzazione Mapper

Per esempio, ad un mapper arrivano 16 elementi di matrice (con i rispettivi indici), qui per facilita' di lettura li metto in un formato facilmente leggibile dall'essere umano e non inserisco il valore.

ingresso

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
...	...	...	...
b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}

$\longrightarrow$

uscita	
k int	v int
0	$(0, a_{00}, 0)$
1	$(0, a_{01}, 0)$
2	$(0, a_{02}, 0)$
3	$(0, a_{03}, 0)$
0	$(1, a_{10}, 0)$
1	$(1, a_{11}, 0)$
2	$(1, a_{12}, 0)$
3	$(1, a_{13}, 0)$
0	$(0, b_{00}, 1)$
0	$(0, b_{01}, 1)$
0	$(0, b_{02}, 1)$
0	$(0, b_{03}, 1)$
1	$(0, b_{10}, 1)$
1	$(1, b_{11}, 1)$
1	$(2, b_{12}, 1)$
1	$(3, b_{13}, 1)$

Visualizzazione Mapper: numeri

Per non generare dubbi su eccessiva ridondanza vediamo un esmepio nel dettalgio dove non compaiono formule, ma numeri...

ingresso

$a_{00} = 3.42$	$a_{01} = 1.24$	$a_{02} = 6.42$	$a_{03} = 8.00$
$a_{10} = 0.54$	$a_{11} = 9.01$	$a_{12} = 7.75$	$a_{13} = 3.23$
...	...	...	...
$b_{00} = 6.21$	$b_{01} = 5.34$	$b_{02} = 8.11$	$b_{03} = 1.21$
$b_{10} = 3.44$	$b_{11} = 6.57$	$b_{12} = 2.98$	$b_{13} = 8.34$

$\longrightarrow$

uscita	
k int	v int
0	(0, 3.42, 0)
1	(0, 1.24, 0)
2	(0, 6.42, 0)
3	(0, 8.00, 0)
0	(1, 0.54, 0)
1	(1, 9.01, 0)
2	(1, 7.75, 0)
3	(1, 3.23, 0)
0	(0, 6.21, 1)
0	(1, 5.34, 1)
0	(2, 8.11, 1)
0	(3, 1.21, 1)
1	(0, 3.44, 1)
1	(1, 6.57, 1)
1	(2, 2.98, 1)
1	(3, 8.34, 1)

Map, Reduce 1, Reduce 2: Reduce1

Torniamo a MapReduce, e ricordiamoci ho **una sola** funzione (generatore) che deve lavorare su tutte le possibili coppie chiave-valore in ingresso. Il reducer produce effetti differenti in funzione della chiave di ingresso!
Questo primo reducer ottiene in ingresso coppie chiave-valore del seguente formato:

- il numero di colonna (se e' un e.d.m. della matrice A) o di riga (per la B)
- una terna contenente il numero di riga (o colonna per le B), il valore dell'ingresso e un indice che specifichi se l'elemento appartiene alla matrice A o B.

A questo punto **ogni** elemento di A deve essere moltiplicato per **ogni** elemento di B giunto al reducer.
In uscita verranno quindi emesse delle coppie chiave-valore in cui:

- **la chiave** e' data dall'indice di riga dell'elemento A e l'indice di colonna dell'elemento B
- **il valore** e' il risultato del prodotto dei due elementi

Ovvero:

chiave	valore
(i, j)	$(a_{ik}\, b_{kj})$

Per esempio se arrivassero le due seguenti coppie chiave valore ad un reducer:

```
4          (2, 4.23 ,0)
4          (7, 1.72, 1)
```

Questo primo reducer emetterebbe quindi la seguente coppia chiave valore

```
(2,7)      7.2756
```

Questa coppia chiave valore contiene dunque le coordinate i, j dell'elemento di matrice C e uno dei prodotti che vanno a comporre l'oggetto stesso.

Visualizzazione reducer 1

Supponiamo che nel reducer 1 arrivino le seguenti coppie chiave valore dal mapper:

chiave	valore
0	$(0, a_{00}, 0)$
0	$(1, a_{10}, 0)$
0	$(0, b_{00}, 1)$
0	$(0, b_{01}, 1)$
0	$(0, b_{02}, 1)$
0	$(0, b_{03}, 1)$

$\longrightarrow$

uscita chiave intermedia	valore intermedio
00	$a_{00} b_{00}$
01	$a_{00} b_{01}$
02	$a_{00} b_{02}$
03	$a_{00} b_{03}$
10	$a_{10} b_{00}$
11	$a_{10} b_{01}$
12	$a_{10} b_{02}$
13	$a_{10} b_{03}$

Le coppie chiave valore emesse contengono il prodotto di tutte le *a* e tutte le *b* all'interno del reducer. In pratica per ottenere i valori di emissione dei reducer 1 faccio il prodotto esterno della colonna di edm di A e la riga degli edm di B.

Map, Reduce 1 , Reduce2 : Reduce 2

Torniamo a MapReduce, a questo punto la riduzione finale e' abbastanza scontata, infatti al secondo reducer arrivano coppie chiave valore in cui

- la chiave contiene la coppia l'indice di riga e l'indice di colonna dell'elemento della matrice C
- il valore e' **uno** dei prodotti che vanno sommati per ottenere il valore corretto dell'edm di C.

Per ottenere il risultato voluto e' sufficiente **sommare tutti** i valori ottenuti dal reducer! A questo punto possiamo emettere coppie chiave valore della forma:

- chiave finale: indice di riga e colonna i, j
- valore finale: c_{ij}

Visualizzazione reducer 2

Per esempio:

00	$a_{00}b_{00}$	
00	$a_{01}b_{10}$	
00	$a_{02}b_{20}$	
00	$a_{03}b_{30}$	
00	$a_{04}b_{40}$	$\longrightarrow a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20} + a_{03}b_{30} + a_{04}b_{40} + a_{05}b_{50} + a_{07}b_{70} + \ldots = c_{00}$
00	$a_{05}b_{50}$	
00	$a_{06}b_{60}$	
00	$a_{07}b_{70}$	
…		

Il reducer(2) in questo caso deve solo **sommare** tutti i valori che ottiene in ingresso delle coppie chiave-valore. Come emissione, per esempio il reducer 2 puo' emettere gli edm della matrice C nel formato iniziale che e' stato dato in pasto al primo mapper.

Dimensioniamo il processo

Supponiamo che le matrici siano quadrate $N \times N$:

- I dati che vengono inviati al mapping sono divisi automaticamente dall' HDFS. Quindi in linea di principio posso mandare un minimo di un e.d.m. per mapper (ci sono N^2 edm per A e altrettanti per B).
- Il primo reducer puo' essere eseguito da un numero di nodi che non deve eccedere N, questo perche' ci sono solo N prodotti esterni da eseguire (occhio che ogni prodotto esterno consiste in N^2 prodotti tra edm.
- Il numero di reducer(2) deve essere al piu' N^2 perche' ci sono N^2 edm diversi da calcolare e quindi ci possono essere al piu' N^2 chiavi diverse (se ne metto di meno devo fare una ulteriore riduzione e se ne metto di piu' alcuni non lavorano).

Il primo reducer rappresenta quindi il collo di bottiglia (ammesso di avere un numero di nodi sufficientemente grande) del processo perche' il numero di reducer attivi e' al massimo N.

Altro algoritmo per il prodotto di matrici: sottoblocchi

Una strategia alternativa si basa sulla divisione in blocchi delle matrici A, B e C. La procedura che viene presentata qui include l'uso di un mapper e di due reducer: reducer(1) e reducer(2).
Domanda: quale vantaggio puo' derivare nello spezzettare la matrice in blocchi?
Risposta: mi serve che i blocchi siano sufficientemente piccoli da poter stare in memoria sui singoli nodi.

I mapper ricevono come input una terna:

1. un sottoinsieme (qualsiasi) di elementi di una matrice (A o B) (in generale questo sottoinsieme sara' determinato dalla dimensione dei chunk di Hadoop.)
2. gli indici di riga e colonna associati ad ognuno degli elementi
3. il fatto che appartengano alla matrice A o B.

In output i mapper emettono coppie chiave-valore. L'idea e' quella di suddividere le matrici A e B in sottomatrici (quadrate nell'esempio) dove:

- la **chiave** sono due indici che identificano il **blocco** a cui appartiene l'elemento di matrice ricevuto in input
- il **valore** e' una terna contentente l'elemento di matrice stesso, le sue coordinate, e un bit che indichi se appartiene ad A o a B.

Con questa procedura ai reducer(1) arrivano tutti gli elementi corrispondenti un ad un sottoblocco di matrice A e B.

Come procedere

Come si puo' ottenere questo risultato?

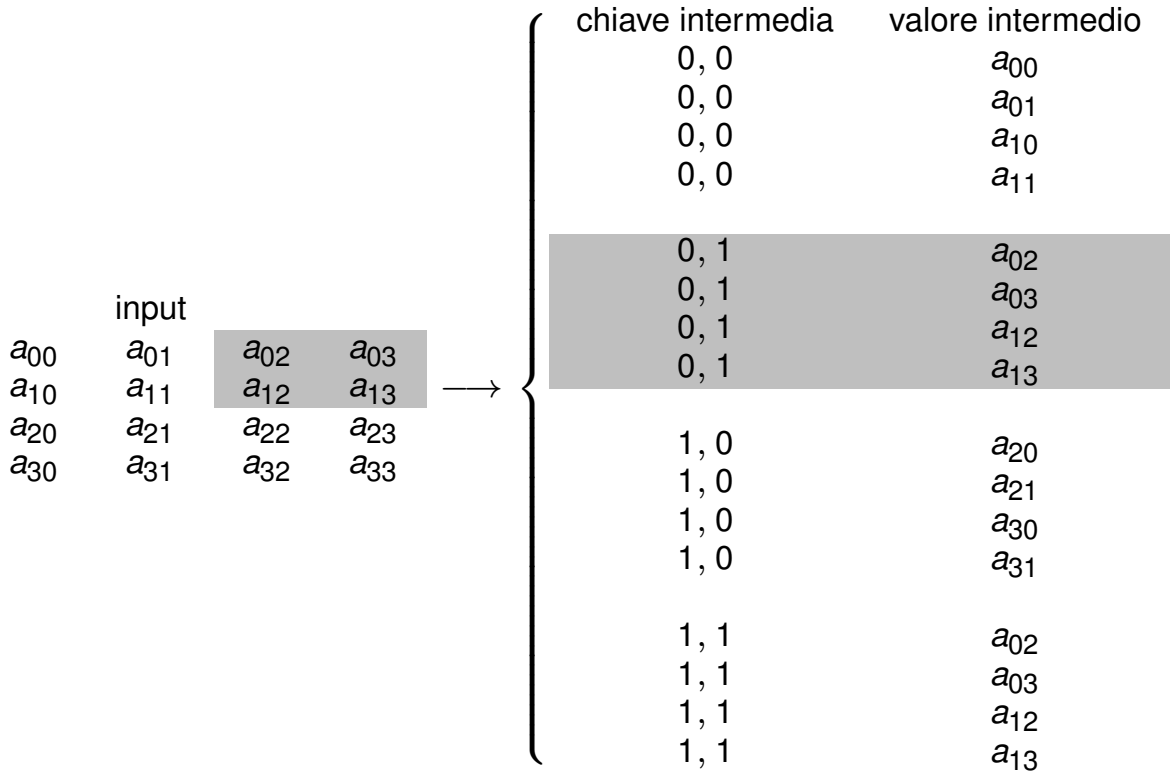
Per esempio le **chiavi** possono essere ottenute facendo una divisione degli indici di ciascun elemento di matrice per il numero di blocchi sull'asse x e y in cui e' stata divisa la matrice.

- Supponiamo di avere deciso di usare B_l blocchi sull'asse delle x e B_l blocchi sull'asse delle y (con matrici $N \times N$).
- Supponiamo che un mapper abbia ricevuto l'elemento a_{ij}.

Creiamo la seguente chiave intermedia: $(i/B_l, j/B_l)$ (attenzione qui si intende che, p.es. 3/5 = 0 o 13/5 =2).

Visualizzo Mapper

In pratica, supponiamo che il numero di blocchi, $B_l = 2$, e il primo mapper riceva tutti gli elementi di matrice di A (16 elementi).



(la stessa cosa viene fatta per gli elementi della matrice B che magari arriveranno ad un altro mapper).

Come procedere 2

- Ad ogni reducer(1) arrivano tutti gli e.d.m. corrispondenti ad un determinato blocco (e l'informazine che indica se sono e.d.m di A_{ij} o di B_{ij}).
- Posso quindi usare in ogni reducer(1) un blocco di A e uno di B (con gli stessi indici di blocco).

Supponiamo che in un reducer(1) ci siano A_{12} e B_{12}. Questi due blocchi entrano nella creazione di piu' di un blocco di C, dobbiamo capire in quali!
Per esempio:

- A_{12} deve essere inviato a tutti i reducer (2) che devono calcolare blocchi del tipo C_{1k} $(k = 0, B_l)$, questo perche' $C_{1k} = A_{11}B_{1k} + \mathbf{A_{12}}B_{2k} + ... + A_{1n}B_{nk}$
- similmente gli B_{12} vanno a tutti i reducer(2) che calcolano i blocchi della forma C_{k2} $(k = 0, B_l)$, questo perche' $C_{k2} = A_{k1}\mathbf{B_{12}} + A_{k2}B_{22} + ... + A_{kn}B_{n2}$

Reducer 1

Il reducer(1), non fa alcun calcolo, ha il solo scopo di riformattare le informazioni emettendo molte coppie chiave-valore dove:

- la chiave e' un'identificativo associato ad una sottomatrice di C: gli indici della sottomatrice
- il valore e' una coppia contentente il blocco che ha ricevuto e il bit che indica se e' un blocco di A o di B.

Domanda: come verranno costruiti i valori delle coppie chiave-valore dei reducer(1)?
Risposta: Per esempio il reducer(1) ha ottenuto (tutti gli elementi di matrice per costruire).

Emissioni

Vediamo un esempio: dal blocco A_{ij} e B_{ij}, verranno emesse delle bag di coppie chiave valore del tipo:

$$A_{ij} \longrightarrow \begin{cases} (i,0)\texttt{TAB}(A_{ij},0) \\ (i,1)\texttt{TAB}(A_{ij},0) \\ \vdots \\ (i,B_l-1)\texttt{TAB}(A_{ij},0) \\ (i,B_l)\texttt{TAB}(A_{ij},0) \end{cases} \qquad A_{02} \longrightarrow \begin{cases} (0,0)\texttt{TAB}(A_{02},0) \\ (0,1)\texttt{TAB}(A_{02},0) \\ (0,2)\texttt{TAB}(A_{02},0) \\ (0,3)\texttt{TAB}(A_{02},0) \end{cases}$$

$$B_{ij} \longrightarrow \begin{cases} (0,j)\texttt{TAB}(B_{ij},1) \\ (1,j)\texttt{TAB}(B_{ij},1) \\ \vdots \\ (B_l-1,j)\texttt{TAB}(B_{ij},1) \\ (B_l,j)\texttt{TAB}(B_{ij},1) \end{cases} \qquad B_{13} \longrightarrow \begin{cases} (0,3)\texttt{TAB}(B_{13},1) \\ (1,3)\texttt{TAB}(B_{13},1) \\ (2,3)\texttt{TAB}(B_{13},1) \\ (3,3)\texttt{TAB}(B_{13},1) \end{cases}$$

Reducer 2

I reducer(2) ottiengono tutti i blocchi necessari per costruire il blocco di C corrispondente alla chiave di input.

- Per esempio se la chiavi di input e' la coppia $(3, 5)$, qusto reducer costruira' il blocco C_{35}.
- Questo e' possibile perche' i reducer(1) gli hanno mandato tutti i blocchi di A e B necessari. Quindi dovra' fare B_l prodotti di matrici (blocchi) di dimensione $N/B_l \times N/B_l$ ed infine sommare questi prodotti.

Domanda: Quanti reducer(2) ha senso che ci siano?

Risposta: Ci sono al massimo B_l^2 di questi reducer, ovvero uno per ognuno dei sottoblocchi in cui e' stata divisa la matrice C.

Reducer 2 tris

Per calcolare il blocco C_{00} servono i seguenti blocchi di A_{00} e A_{01} e di B_{00} e B_{10}. Quindi un reducer(1) emette delle coppie, in cui:

- la **chiave** sono gli indici del blocco di C da calcolare (p.es se devo calcolare il blocco C_{13}, la chiave e' $(1, 3)$
- il **valore** e' un blocco di A o B necessario per questo calcolo.

Domanda: Come si fa a calcolare se un blocco A_{kl} e' necessario per l'elemento di matrice C_{12}?

risposta: E' facile sapere se un blocco di A e' necessario o meno: tutti i blocchi di A che contengono come indice di riga l'indice di riga del blocco C servono per calcolarlo: $C_{ij} = \sum_k A_{ik} B_{kj}$. Quindi nel caso d'esempio se devo calcolare il blocco C_{13} mi servono $A_{10}, A_{11}, A_{12}, \ldots$ e similmente $B_{03}, B_{13}, B_{23}, \ldots$

nel dettaglio...

reducer 2 quadris

Il blocco A_{01} serve per la produzione dei blocchi C_{00} e C_{01}

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \rightarrow \begin{pmatrix} C_{00} & * \\ * & * \end{pmatrix}$$

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \rightarrow \begin{pmatrix} * & C_{01} \\ * & * \end{pmatrix}$$

mentre la sottomatrice di B, con gli stessi indici: $B_{01} \rightarrow C_{00}, C_{10}$:

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \rightarrow \begin{pmatrix} * & C_{01} \\ * & * \end{pmatrix}$$

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \rightarrow \begin{pmatrix} * & * \\ * & C_{11} \end{pmatrix}$$

$$\begin{matrix} A_{01} \\ \\ B_{01} \end{matrix} \longrightarrow \left\{ \begin{array}{ll} \text{k int} & \text{v int} \\ 0, 0 & A_{01} \\ 0, 1 & A_{01} \\ & \\ 0, 1 & B_{01} \\ 1, 1 & B_{01} \end{array} \right.$$

Risultato

A questo punto, in **ognuno** dei reducer arrivano tutti i blocchi di matrice che servono per calcolare **un** blocco di matrice C. Quello che si fa e' un normale prodotto matrice matrice (con somme finali).
Appare evidente che a questo punto si potrebbe allungare il processo di mapper e reducer in modo tale, per fare i prodotti e somme dei blocchi si possa sfruttare ulteriormente il fatto che abbiamo a disposizione molti nodi, e ognuno di questi calcoli e', de facto indipendente dall'altro!

$$
\begin{array}{ll}
0,0 & A_{00} \\
0,0 & A_{01} \\
0,0 & B_{00} \\
0,0 & B_{10}
\end{array}
\longrightarrow C_{00}
$$

$$
\begin{array}{ll}
0,1 & A_{00} \\
0,1 & A_{01} \\
0,1 & B_{01} \\
0,1 & B_{11}
\end{array}
\longrightarrow C_{01}
$$

$$
\begin{array}{ll}
1,0 & A_{10} \\
1,0 & A_{11} \\
1,0 & B_{00} \\
1,0 & B_{10}
\end{array}
\longrightarrow C_{10}
$$

$$
\begin{array}{ll}
1,1 & A_{10} \\
1,1 & A_{11} \\
1,1 & B_{01} \\
1,1 & B_{11}
\end{array}
\longrightarrow C_{11}
$$

E' utile pensare all'utilizzo delle risorse con questo tipo di algoritmo. In particolare all'inizio posso usare fino a $2N^2$ mapper (uno per ogni elemento di A e B). Il numero di reducer (1) da usare e' invece N^2 / B_l^2. Tanto piu' piccoli sono i blocchi in cui e' divisa la matrice tanti piu' sono i reducer. I reducer (2) finali, sono al massimo tanti quanti i blocchi della matrice C B_l^2.

Altre strategie di moltiplicazione e i loro pregi e difetti

Presso questo sito `http://www.norstad.org/matrix-multiply/`
Si trovano varie possibili strategie da utilizzare per ottenere un prodotto matrice matrice tramite job Mapreduce. Domande

- quale procedura richiede il minor traffico possibile di rete?
- quale procedura permette di bilanciare nel miglior modo possibile i carichi di lavoro tra mapper e reducer?

Altre strategie 2

- strategia 1: ogni reducer ha una sottomatrice di A e una di B e fa il prodotto. a questo punto queste sottomatrici C (parziali dato che devo sommarle) vengono sommate in un secondo reducer.
- strategia 2: ogni reducer fa il prodotto di una sottomatrice di A e di tutte le sottomatrici di B necessarie. A questo punto vengono mandati i pezzi ad un successivo reducer che somma i pezzi corretti.
- strategia 3: speculare rispetto alla 2, ma dove un singolo reducer fa prodotto matriciale tra una sottomatrice di B e le corrispondenti di A.

Esercizio: riassumere un file

Scrivere un job MrJob che trovi:

- il numero di righe
- il numero di parole (usare la funzione di python `len()`)
- il numero di caratteri (usare la funzione di python `len()`)

quindi:

1. definire una funzione `mapper` che restituisca 3 tipi di coppie chiave valore, **una** contentente il numero di righe, **una** per il numero di parole e **una** per il numero di caratteri
2. definire una funzione `riduttore` che sommi il numero di caratteri,di righe e di linee
3. fuori dalla classe, usare il metodo `.run()` per fare partire il job.

Esercizio: la parola piu' comune

Scrivere un job MrJob che trovi la parola piu' usate in un testo (e quante volte appare):

1. importare MrJob
2. importare il modulo `re`, per dividere le linee correttamente
3. definire una funzione `mapper` che da ogni riga restituisca le parole e il conteggio 1
4. definire una funzione `combiner` che prenda le coppie parola 1 e sommi tutte le occorrenze
5. definire una prima funzione `reducer` che restituisca le parole il loro conteggio
6. definire una seconda funzione `reducer` che trovi per quale di queste parole si ha il massimo
7. definire una fuzione `steps` che indichi come vanno usate le funzioni nel job.
8. fuori dalla classe, usare il metodo `.run()` per fare partire il job.

Esercizio: Matrix-Matrix 1

Scrivere un job MrJob che faccia un prodotto tra matrici con i prodotti esterni (vedi slide 14):

1. importare MrJob
2. importare le matrici
3. creare un mapper che emetta correttamente i valori
4. fare le moltiplicazioni nei reducer
5. fare le somme finali
6. lanciare il job: `python $myscript $matA $matB 1> $myoutput`