Paula Camargo Romera
Alicia Garrido Muñiz
Núria Lladós Armengol

Pràctica 1. Multiplicació de matrius de forma distribuïda

Disseny

La funció my_functionparalel s'encarrega de generar dos matrius aleatòries (A i B) amb un nombre de files i columnes introduït per paràmetre. A continuació, subdivideix les matrius en funció del número de workers introduït. El número de workers podrà ser inferior o igual a nombre de files de la matriu A o igual a les files de A * columnes de B. En el primer cas, és dividirà el nombre de files totals pel nombre de workers de manera que obtindrem submatrius de A amb el mateix nombre de columnes que tenien i rows1/workers nombre de files. Cada submatriu es pujarà al COS per separat amb la forma 1a.txt, 2a.txt, 3a.txt, 4a.txt... Pel que fa a matrius amb un nombre imparell de files, la última submatriu s'encarregarà de fer la resta de files. La matriu B es pujarà sencera amb la forma 1b.txt ja que cada worker s'encarregarà de multiplicar una submatriu de A amb la matriu B. Per últim, en el cas que el número de workers sigui igual a rows1 * cols2 es dividirà la matriu A en files i la matriu B en columnes fent que cada worker calculi un sol número.

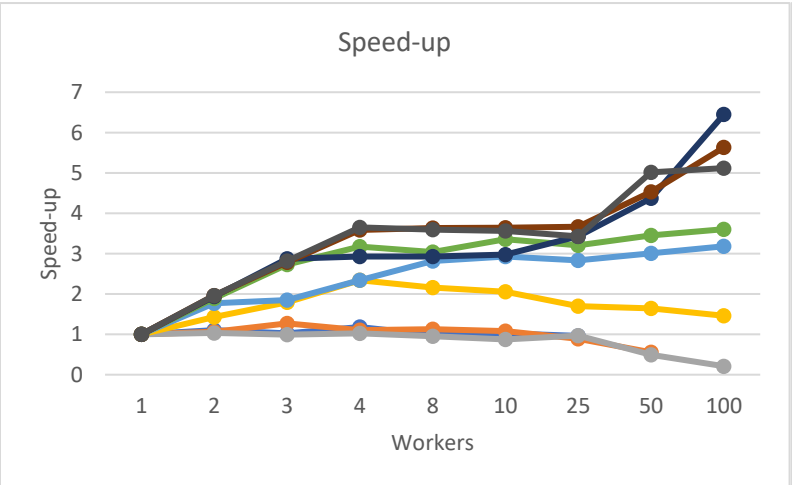
Una vegada que s'han pujat els diferents fitxers al cloud, en la funció my_map_function el que es fa és baixar aquests fitxers i anar fent la submultiplicació. Al final d'aquesta funció, el que es fa és guardar el resultat de la multiplicació en un diccionari que estarà ordenat fent servir el nom del fitxer que s'ha baixat. El que s'ha fet és aprofitar el valor de la variable key1 i key2 fent una divisió d'aquesta per quedar-se únicament amb el que esta davant del ".". D'aquesta forma, ens quedem amb el valor numèric que ens indica l'ordre i el caràcter que ens indica quina de les dues matrius era.

Posteriorment, en la funció my_reduce_function el que s'ha fet és aprofitar que els resultats venen ordenats de l'anterior funció i s'ha creat un nou diccionari el qual s'ha anat actualitzant poc a poc amb tots els resultats anteriors. D'aquesta forma s'aconsegueix tindre un únic diccionari ordenat amb tots els valors de la multiplicació. A continuació, aquest diccionari es transforma a una única llista i és aquesta llista la que es transforma en la matriu final indicant-li la mida de la matriu final i les columnes per fer la partició de la llista en la dimensió indicada.

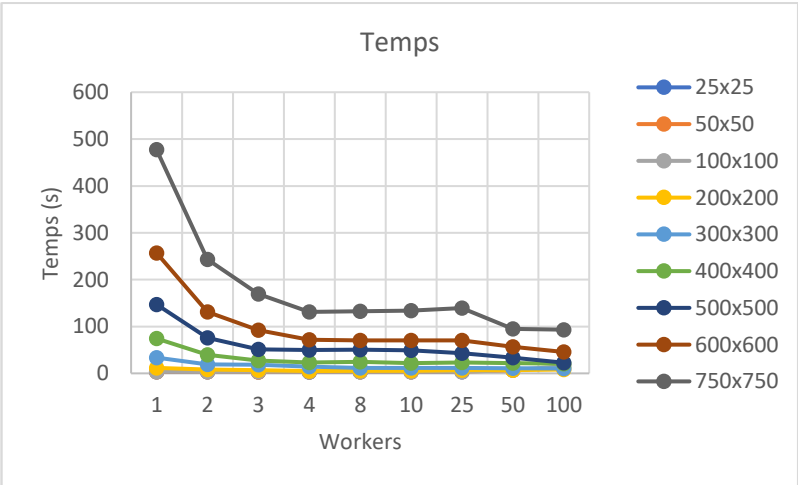
Finalment, es retorna la matriu final creada que es pujarà de forma automàtica al COS.

Anàlisi dels resultats

Els resultats s'han analitzat de dos maneres, respecte el **temps**, i respecte el **speed-up**.



Gràfic 1: Speed-up



Gràfic 2: Temps

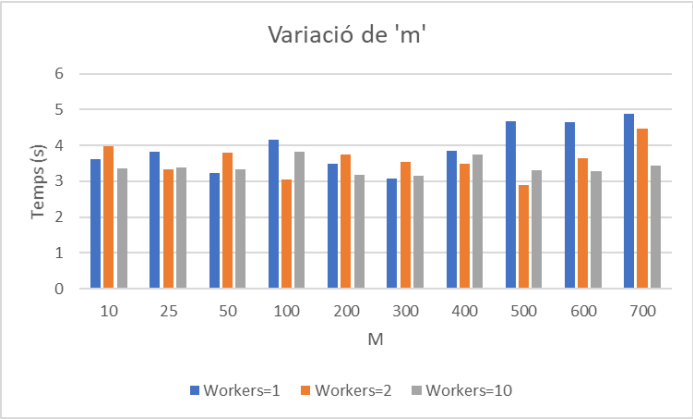
Si s'observa el *Gràfic 2*, podem veure com en matrius amb dimensions petites fer la distribució al multiplicar no és eficient ja que es tarda més en fer la distribució i pujar i baixar els fitxer que en realitzar la operació en sí. Fins i tot, es pot observar com produeix un lleu increment al executar-ho amb més workers que de forma seqüencial (1 worker).

D'altra banda, si ens fixem en matrius de dimensions superiors, es pot observar com la paral·lelització en la multiplicació de matrius si augmenta la eficiència de la multiplicació.

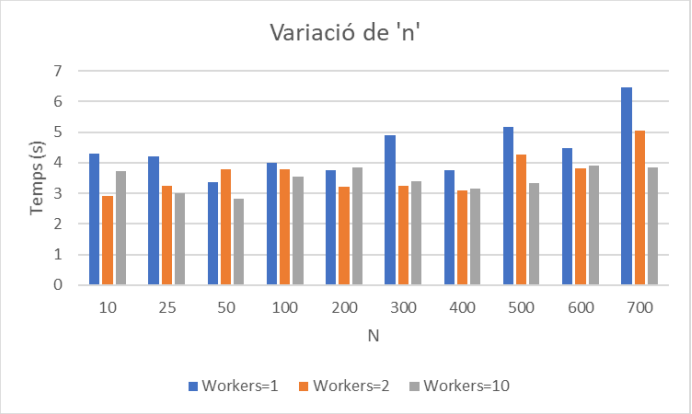
Tot això, es pot verificar al dur a terme el speed-up, on es pot observar al *Gràfic 1* que per dimensions petites el speed-up es veu reduït al dur-lo a terme amb més de 1 worker, és a dir, la forma paral·lela empitjora la eficiència de la seqüencial. Altrament, amb grans dimensions es veu una gran millora en l'speed-up. Per exemple, es pot observar com en la matriu A= 500x500 i B=500x500 al executar-ho amb workers = 100 es veu un increment de 6 cops la forma seqüencial, pel que la seva eficiència es veu molt incrementada. Tot i així, en alguns casos també s'observa com tot i ficar més workers el temps d'execució s'acaba estabilitzant, pel que arribarà un punt en la corba que no augmentarà més.

En conclusió, s'observa com el rendim és màxim en matrius de mida igual o superior a 200x200 i workers = 4.

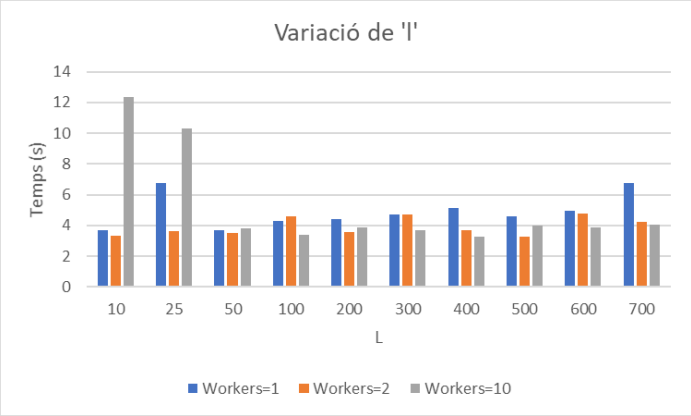
A les següents gràfiques es pot veure representat el temps d'execució variant M, N o L i deixant fixes la resta de valors. M = nombre de files de la matriu A, N = nombre de columnes de A i files de B, L= nombre de columnes de B.



Gràfic 3: Variació de 'n'



Gràfic 4: Variació de 'm'



Gràfic 5: Variació de 'l'

De forma general i, com era d'esperar, s'observa com un increment de M, N o L provoca un increment del temps d'execució sent aquest increment molt més notable quan es treballa amb

workers 1 que quan es fa amb un nombre més elevat de workers. Per tant, es fa palès que el nostre programa fa una subdivisió de matrius de manera òptima, repartint la carrega de treball entre els workers. També es pot observa que, de la mateixa manera que passava en les gràfiques anteriors, la disminució del temps d'execució al treballar amb més workers es fa més notable amb mides de M,N o L majors.

Per últim, s'observa un gran increment del temps al treballar amb workers = 10 i L= 10 i 25. Creiem que aquesta discrepància pot ser deguda a que un valor tan petit de L fa que la càrrega de treball de cada worker sigui molt petita en comparació al fet d'haver de baixar tots els fitxers del cloud.

Joc de proves

S'ha realitzat un joc de proves comprovant que:

- Les matrius que no es poden multiplicar (columnes matriu 1 i files matriu 2 no són iguals), no suposen un error pel programa, sinó que s'assigna el nº de files de la matriu 2 pel nº de columnes de la matriu 1.
- El nº de workers no pot:
    - Ser superior a 100.
    - Ser superior al nº de files de la matriu 1, a excepció de que workers sigui nº files matriu 1 * nº columnes matriu 2.
- La mida de la matriu de sortida és correcta, és a dir, el nº de files és igual al nº de files de la matriu 1 i el nº de columnes de la matriu 2.

Entrada	Entrada vàlida?	Workers	Workers vàlids?	Temps	Sortida	Sortida vàlida?
3x3 – 3x3	Sí	2	Sí	4,32 s	3x3	Sí
3x4 – 4x2	Sí	2	Sí	2,67 s	3x2	Sí
20x20 – 20x15	Sí	21	No, s'assigna Workers=1 de forma predeterminada	3,65 s	20x15	Sí
50x50 – 50x50	Sí	50	Sí	6,5 s	50x50	Sí
50x50 – 50x50	Sí	100	No, s'assigna Workers=1 de forma predeterminada	3,61 s	50x50	Sí
100x100 – 50x100	No, es canvia el valor 50 per 100 de forma predeterminada	100	Sí	15,35 s	100x100	Sí
300x300 – 300x300	Sí	50	Sí	10,57 s	300x300	Sí
750x750 – 750x750	Sí	800	No, s'assigna Workers=1 de forma predeterminada	477,35 s	750x750	Sí
750x750 – 750x750	Sí	100	Sí	95,21 s	750x750	Sí

També s'ha comprovat que la multiplicació sigui correcta. $\begin{pmatrix} 2 & 24 & 3 \\ 46 & 5 & 82 \\ 23 & 1 & 56 \end{pmatrix} x \begin{pmatrix} 2 & 54 \\ 33 & 15 \\ 43 & 8 \end{pmatrix} = \begin{pmatrix} 925 & 492 \\ 3783 & 3215 \\ 2487 & 1705 \end{pmatrix}$

A la *Figura 1* observem que la multiplicació es realitza correctament.

```
ExecutorID 232850/0 - Cleaning temporary data
[[925, 492], [3783, 3215], [2487, 1705]]
Elapsed time:5.08
```

Figura 1: Execució matriu

Referències

1. GitHub - pywren/pywren-ibm-cloud: CloudButton Toolkit implementation for IBM Cloud Functions and IBM Cloud Object Storage. https://github.com/pywren/pywren-ibm-cloud.