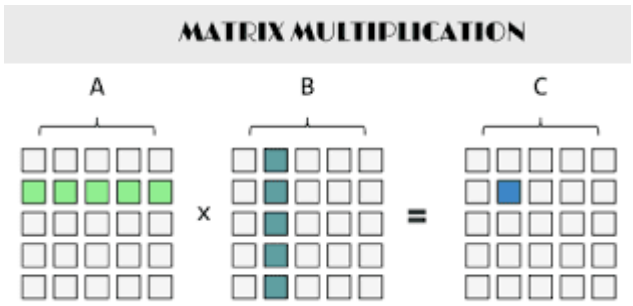
Sistemes distribuïts

Pràctica 1: Multiplicació de matrius

Objectius de la pràctica

Implementar un algorisme que ens permeti paral·lelitzar la multiplicació de matrius utilitzant el núvol. Ja que a casa tenim només 16 cores, i en el núvol podem arribar a tenir-na 100 o més. Per aquest motiu en la pràctica ens demanen implementació i l'execució d'una multiplicació de matrius en el núvol.

En aquest problema de multiplicació de matrius ens demanen resoldre-ho fent ús de IBM Cloud. Aquest servei ens dona funcionalitats necessàries per la pràctica, utilitzarem les següents utilitats: Object storage i Cloud function.



Funcionament de la multiplicació entre matrius

Estructura

Per començar utilitzarem com a llenguatge de programació python, encara que es poden utilitzar diversos llenguatges de programació com per exemple JavaScript. Sabent el llenguatge de programació proposat ens disposem a estructurar el problema. Aquí surten diverses qüestions de disseny que s'han de resoldre:

Què farà cada Worker?

Dues opcions principals, o bé, que cada worker s'encarregui de generar un únic element de la matriu resultant (filaMatriuA *columnaMatriuB) o que a cada worker s'encarregui de generar diversos elements de la matriu resultant. Cada una de les opcions té punts positius i negatius. En el primer cas la implementació és menys complexa i hi ha el número màxim de workers, però aquest aproximadament acaba sent ineficient degut al poc volum de dades que ha de tractar cada worker, ja que tarda més temps l'invocació de cada funció que el propi tractament de les dades, en canvi en el segon cas el nombre de workers seria menor, ja que cada worker s'encarregaria de calcular més d'un element de la matriu resultant, i cada un d'ells tindria un volum de treball més gran de manera que no es produiria un overhead a causa del número excessiu d'innovacions.

Qué li enviem a cada worker?

En un principi vam pensar que la millor opció seria pasar-li a cada worker un objecte amb el subconjunt de dades que li toqués tractar, però aquesta opció té un inconvenient i és que la funció map té una mida màxima del vector de dades (iterdata) de 4 MB, això causava que al passar-li una matriu gran, la matriu superés aquesta mida màxima. Degut a això vam optar per, en lloc de passar-li a cada worker un objecte amb el subconjunt de dades que li toqués tractar, pujar aquests objectes al núvol utilitzant l'object storage de IBM cloud i passar-li a cada worker una llista amb els noms dels objectes pujats al núvol, d'aquesta manera, cada worker, utilitzant aquesta llista es descarregaria els objectes que continguessin les dades que li toqués tractar i les tractaria. D'aquesta última manera es pot treballar amb matrius molt més grans

Llibreries

Ara explicarem les llibreries que hem tingut que utilitzar per a poder realitzar la pràctica.

Pywren

La llibreria que ens deixa interaccionar amb ibm cloud per a poder utilitzar cloud function de manera més senzilla. Aquesta llibreria ens donarà accés a funcions com call_async() que serveix per cridar la funció que desitgem en el cloud, map() que serveix per aplicar una funció a un conjunt de dades de manera que cada conjunt aplica una vegada la funció, i per últim la única funció no asíncrona get_result() que serveix per a esperar els resultats de les funcions cridades de manera asíncrona.

Numpy

Al treballar amb matrius també utilitzarem una llibreria per facilitar l'accés, la manipulació i l'operació de matrius.

CosBackend

Per accedir als nostres objectes en el cloud cos_backend que es va realitzar en el lab 2, on hi ha programades les funcions que ens deixen interaccionar amb l'object storage amb les següents funcions: put_object() que com diu el nom puja l'objecte al núvol , get_object() agafar l'objecte ja pujat, delete_object() que ens servirà per borrar els objectes que no utilitzarem més...

Pickle

Una de les llibreries més importants per a pujar i agafar objectes del núvol, pickle, que ens servirà per a serialitzar (dumps()) i desserialitzar (loads()) els objectes que es pugen al núvol i es baixen del núvol.

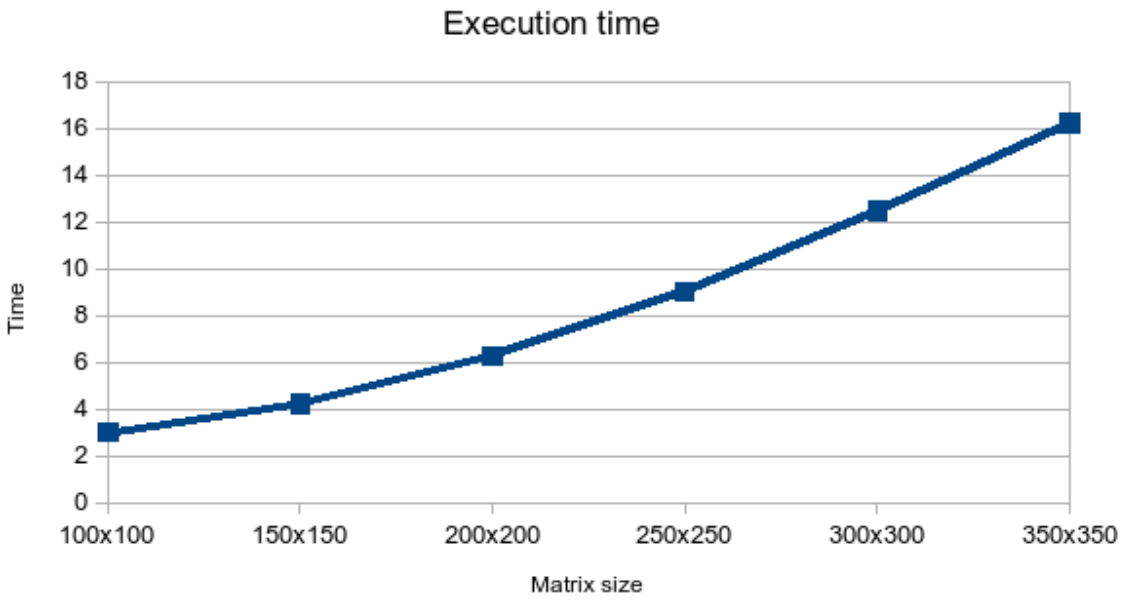
Time

Per últim tenim time per a poder contar el temps d'execució per a tots els paràmetres utilitzats i guardar-los.

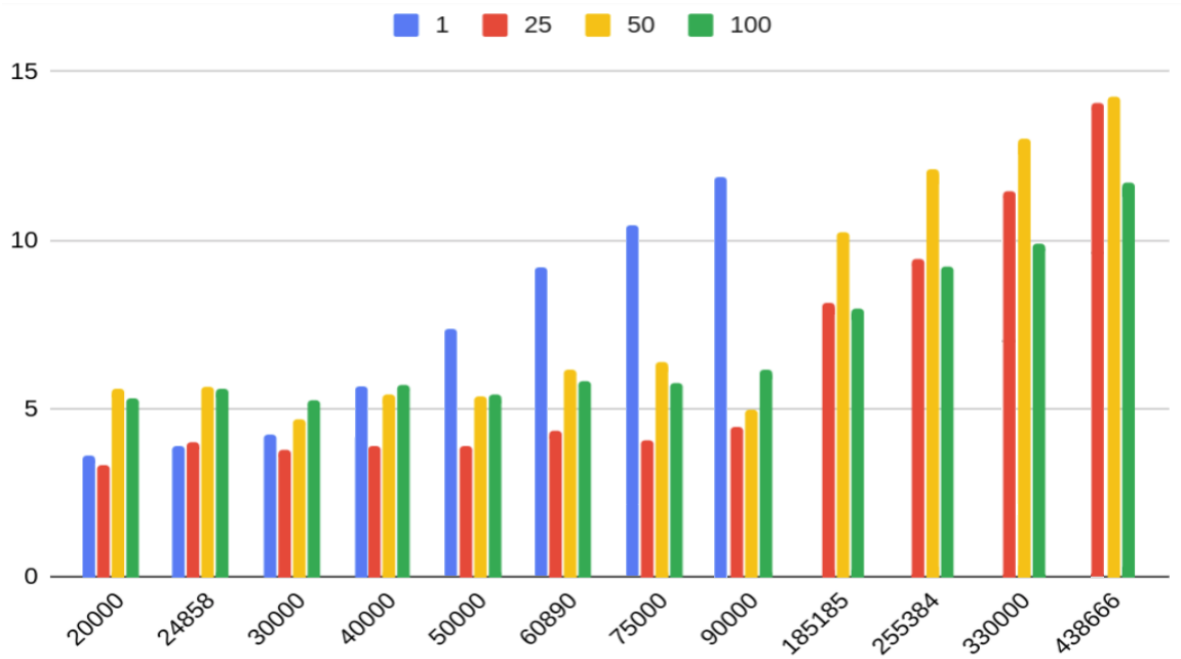
Resultats obtinguts

Temps d'execució

Els resultats que hem obtingut mostren com al augmentar la mida de les matrius, el temps d'execució augmenta de manera quasi lineal.



Aquest temps corresponen a l'execució seqüencial de la multiplicació de matrius amb 1 sol worker.



Gràfic de temps, Workers utilitzats per treball a repartir.

Com podem veure en el gràfic, cada barra representa un número exacte de workers. Sota les barres hi ha el número d'operacions que fan i fins on arriba la barra és el temps que tarda en realitzar-la en segons. Cada barra és el promig de segons que tarden tots els workers en fer les operacions amb diferents valors:

- m : És el número de files que té la MatriuA.
- n : És el número de columnes de la MatriuA i el número de files de la MatriuB (per a poder fer una multiplicació de matrius, es necessita que siguin iguals).
- l : És el número de columnes de la MatriuB.

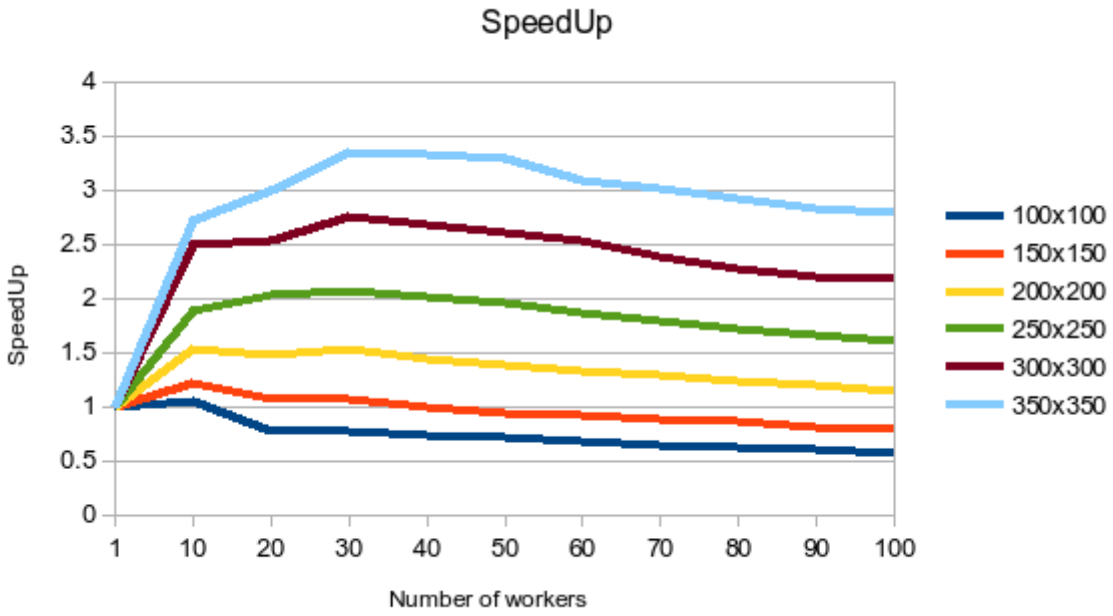
Amb això podem veure que el número de treballs que es necessiten fer amb el nostre plantejament del problema és la multiplicació: m * l

Llavors amb tots els diferents valors de n fem un promig del temps tardat.

Com podem veure en 185185 desapareix el worker, això és degut a la falta de memòria RAM de la màquina virtual que hem utilitzat per realitzar la pràctica, ja que al serialitzar les dades utilitzant la llibreria Pickle, aquesta consumeix una gran quantitat de memòria RAM i al ser un únic worker i per tant un únic fitxer, la màquina virtual no tenia pro memòria per carregar tot aquest fitxer en memòria per poder serialitzar-lo.

SpeedUp

A partir dels resultats que hem obtingut, podem observar que a mesura que augmenta la mida de les matrius, l'speedup també augmenta, això és degut a que, en les matrius més petites, el volum de dades amb el que treballen cada un dels workers, és més petit, això provoca que pràcticament es trigui més estona en descarregar i rebre les dades, que en fer les pròpies multiplicacions, degut a que establir les connexions per fer la transmissió de les dades, introdueix un retard considerable



Referències

Funcionalitat de map i funcions com get_result, etc:

https://github.com/pywren/pywren-ibm-cloud/

Llibreria numpy i funcionalitats:

https://github.com/numpy/numpy

Llibreria time:

https://docs.python.org/3/library/time.html

Participants i feina feta :

Adrià Ribas Jaumà

Nicolás Fadul Bonamusa

Ens hem repartit la feina equitativament. Si algú dels dos tenia cap problema ho resolíem entre els dos.