TASK 1 : Communication models and Middleware

Guillem Martí Mesas
Sergio Fernández Aunión
Universitat Rovira i Virgili
15/04/2020

ÍNDEX

Objectiu de la pràctica **3**

Implementació de la pràctica **3**

Resultats **4**

Repartiment de la pràctica **5**

Webgrafía **5**

Objectiu de la pràctica

El principal objectiu d'aquesta pràctica es implementar de manera distribuida la multiplicació de dos matrius A $\in \mathbb{R}^{m \times n}$ i B $\in \mathbb{R}^{n \times l}$. Per senzillesa, implementarem una versió de multiplicació no del tot óptima.Concretament, cada treballador descargarà i multiplicarà a x n del bloc fila de la matriu A i n x a del bloc columna de B, per completar la matriu C $\in \mathbb{R}^{m \times l}$ que serà el resultat, Figura 1.

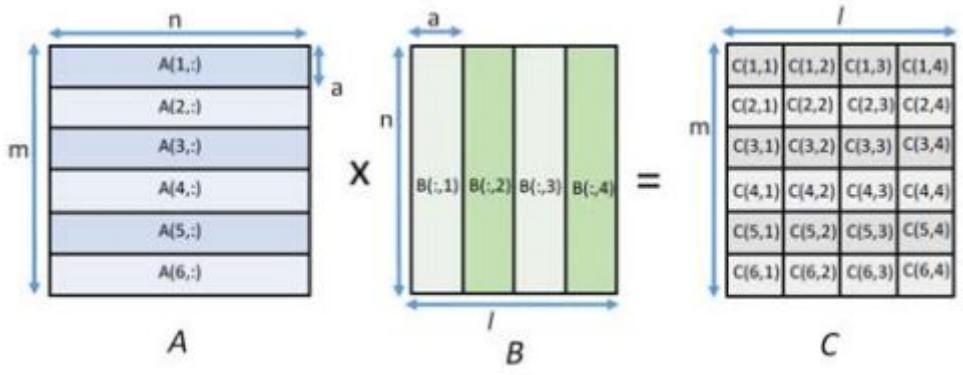


Figura 1.

Cada treballador haurà de descargar el bloc de fila i columna corresponent del servidor, i després de fer el càlcul tornar a emmagatzemar el bloc al servidor.

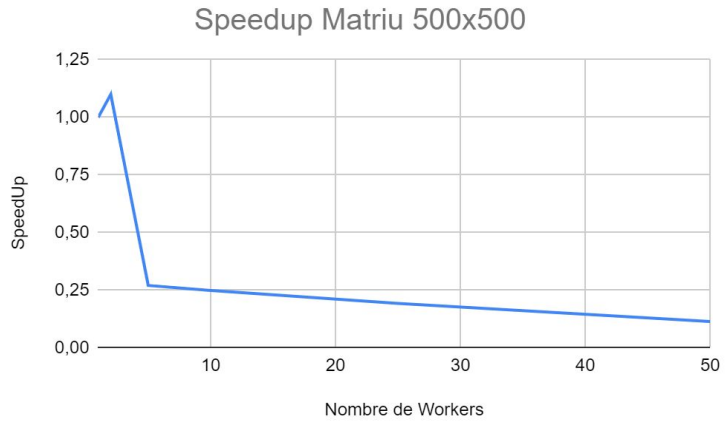
Implementació de la pràctica

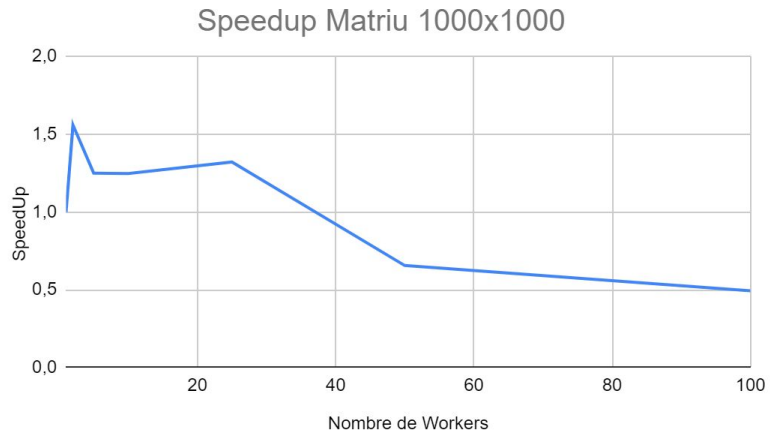
Per implementar aquesta pràctica hem realitzat les següents funcions:

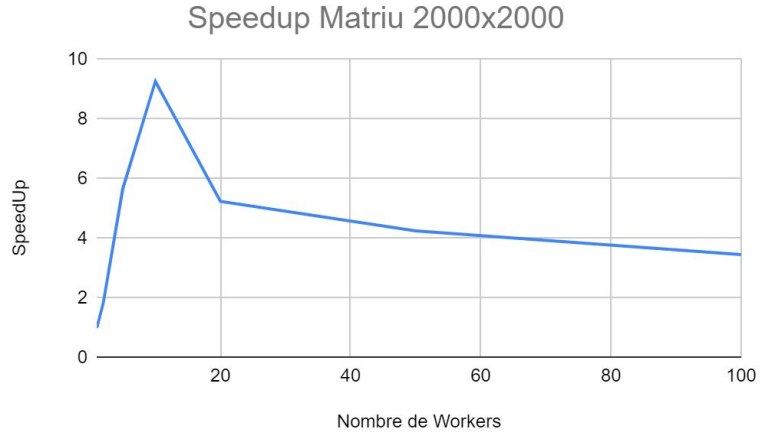
- Primer de tot hem creat una llista dels paràmetres necessaris per a poder executar el codi, aquesta llista conté [Nombre de files de matriu A , Nombre de columnes de la matriu A/Nombre de files de la matriu B, Nombre de columnes de la matriu B, rang de generació de nombres, Nombre de treballadors]
- Seguidament hem definit una funció d'inicialització de matrius la qual crea dues matrius de mida MxN i NxL i les omple de números aleatoris i posteriorment les parteix en tantes submatrius com workers hi hagi i crea l'iterdata amb les tuples pertinents que les multiplicació
- La funció de map, cada worker recorre la seva posició de l'iterdata i va descarregant les submatrius que necessita per fer les multiplicacions pertinents i les va retornant.
- Per últim la funció reduce rep els resultats de les diverses multiplicacions dels diferents workers i les va combinant en la matriu final que es la que es retorna i mostra per pantalla.
- Dins del main es fa una crida la funció time() a l'inici i al final una resta per a calcular el temps que es triga a executar tot el procés.

Resultats

Hem realitzat les següents execucions amb diferents nombres de workers, i diferents tamanys de matrius. Primer hem provat amb la multiplicació de dues matrius de 500x500, com podem observar en la primera gràfica, el speedup més alt es produeix amb un nombre baix de workers. Aquesta tendència ocorre per les matrius de 1000x1000 i 2000x2000. En el cas de les matrius de 2000x2000 veiem que amb 10 workers, arriba a un speedup de 9.







Repartiment de la pràctica

Per realitzar la pràctica, a causa del confinament de l'Estat d'Alarma d'Espanya, hem estat en videotrucada en tot moment, Guillem ha estat escrivint el codi mentre compartia la pantalla amb Sergio i comentàvem com hauria de ser l'algoritme, si sorgia cap dubte Sergio era l'encarregat de buscar-ho per Internet. Per realitzar la documentació també hem estat treballant els dos alhora, gràcies a Google Docs, que ens permet escriure en el mateix document.

Webgrafía

Funció Pickle : https://www.journaldev.com/15638/python-pickle-example

Funció Randint:
https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.randint.html#numpy.random.randint

Generar matrius amb nombres aleatoris: https://stackoverrun.com/es/q/4192415

Funció Array_split:
https://docs.scipy.org/doc/numpy/reference/generated/numpy.array_split.html

Totes les funcions de Pywren:
https://github.com/pywren/pywren-ibm-cloud
https://github.com/JosepSampe/pywren-ibm-cloud