Universitat Rovira i Virgili

Curs 2021-2022

Tasca 1: Models de Comunicació i Middleware

Alumnes: Robert Crusells Nollas i David Gutiérrez Pasqual

Assignatura: Sistemes Distribuïts

Data d'entrega: 19/06/2022

Índex

Enunciat

L'objectiu és implementar un Python Distributed DataFrame (Dask Dataframes) inspirat a la llibreria Dask Python. La idea és executar operacions en Dataframes que resideixen en nodes remots. S'ha d'implementar quatre solucions utilitzant tecnologies diferents: XMLRPC, gRPC, Redis i RabbitMQ.

Implementar una arquitectura de clúster Master/Worker on una interfície accedeixi als diferents nodes. El Master hauria de suportar una API senzilla per afegir o eliminar nodes del clúster. També s'ha de poder oferir un subconjunt de l'API Dask DataFrame per tractar les dades CSV.

Admet operacions com: read_csv, apply, columns, groupby, head, isin, items, max, min.

Es recomana la copia de codi d'Internet i de Dask si cal.

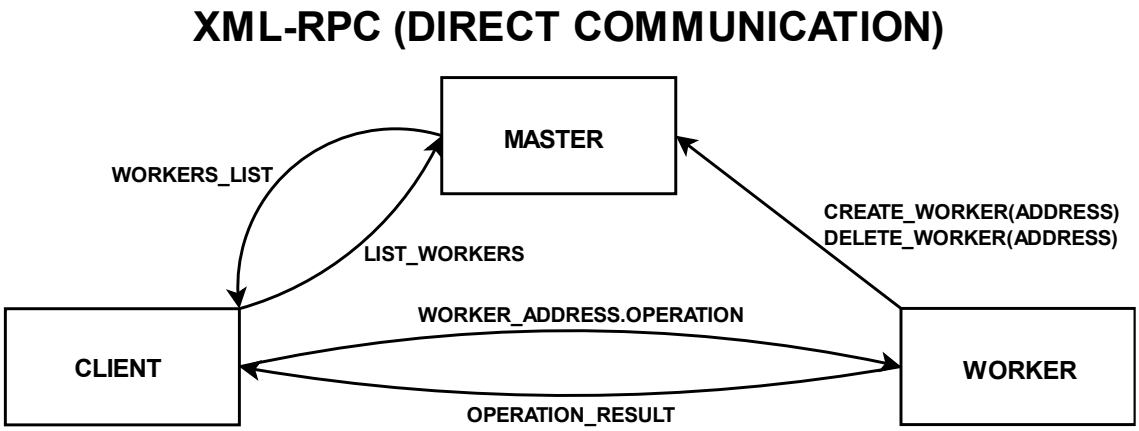
Tasques Teòriques

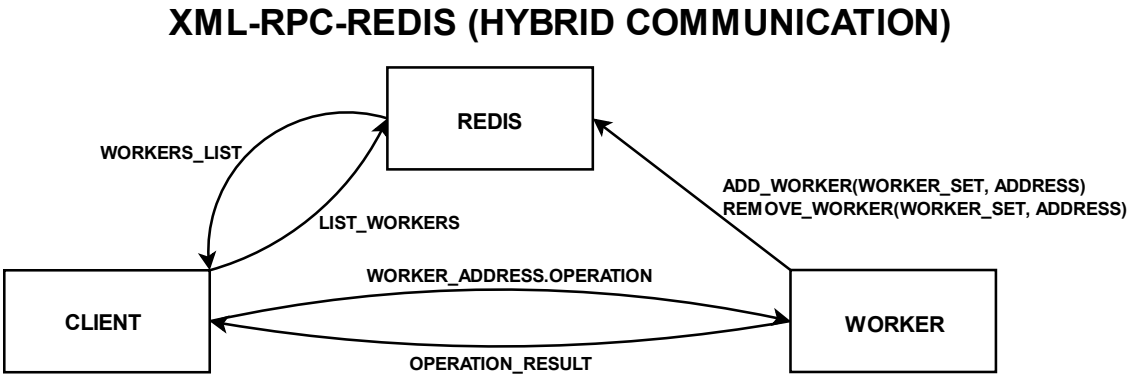
Tasca 1

Enunciat

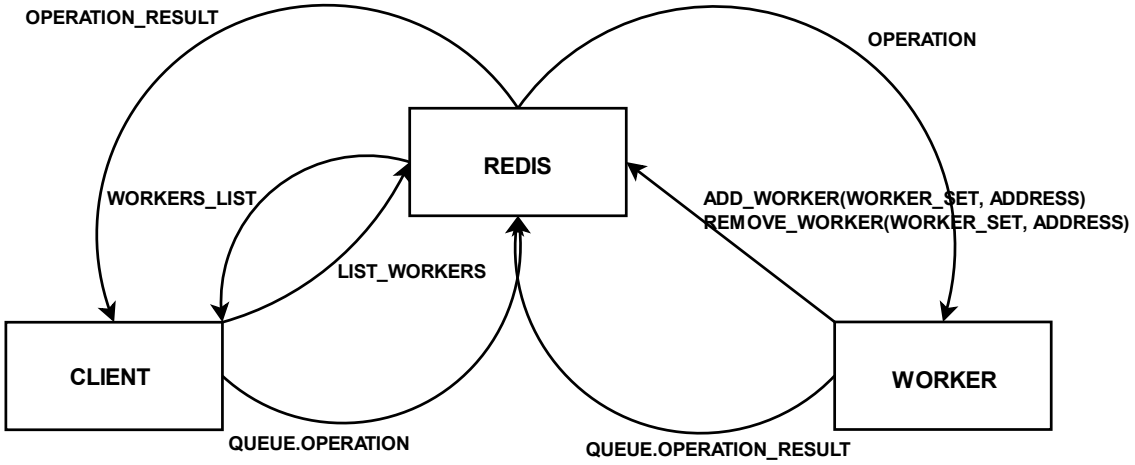
Proporcioneu esquemes que expliquin les solucions que heu implementat. Mostra arguments que validen l'execució de tasques distribuïdes en comparació amb la mateixa execució en una única màquina.

Esquemes

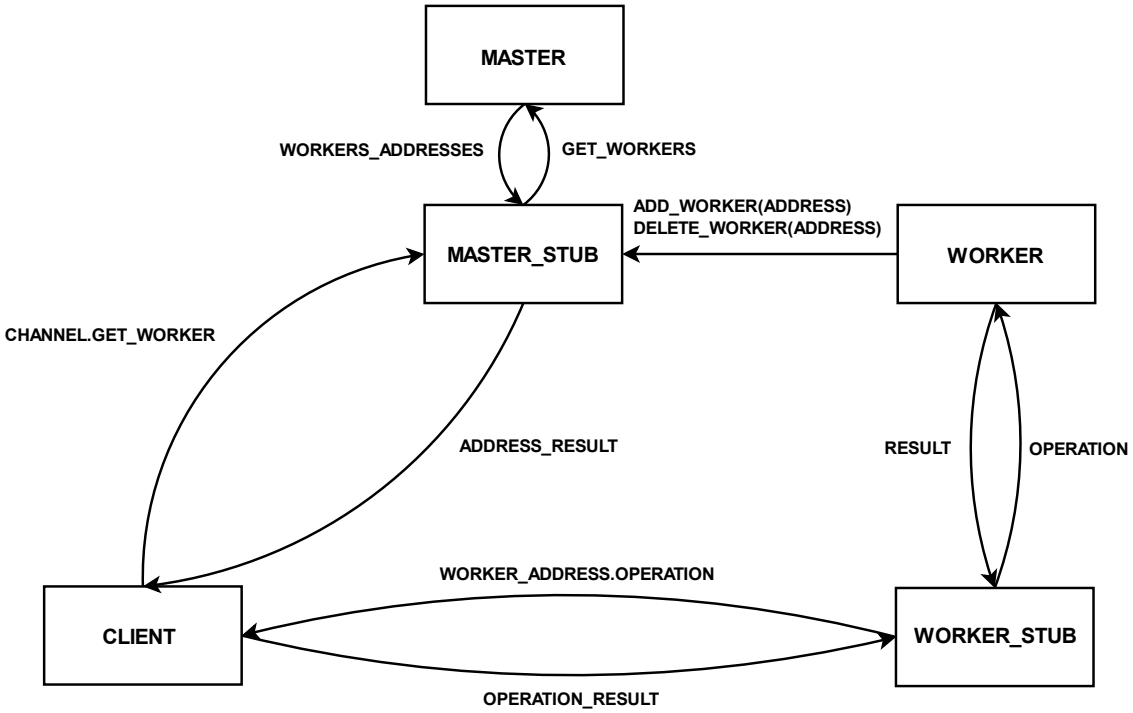




REDIS (INDIRECT COMMUNICATION)



GRPC (DIRECT COMMUNICATION)



Tasca 2

Enunciat

Compareu les arquitectures que heu dissenyat utilitzant middlewares de comunicació directa o indirecta. Fins i tot podeu proposar solucions que combinen diferents serveis de middleware (com gRPC i Redis).

XML-RPC (Directe)

El client es comunica amb el màster per obtenir una llista d'adreces que pertanyen als workers. Llavors el client mitjançant les adreces de la llista ja es pot comunicar amb els workers. Els workers només podran comunicar-se amb el client si aquest s'ha comunicat abans.

Per tant podem dir que el model de comunicació emprat és síncron, ja que per continuar necessita la resposta del worker. Si aquest falla, cau el sistema. També podem dir que és un sistema pull, ja que és el client qui sol·licita informació al servidor.

XML-RPC i REDIS (Híbrid)

Els workers s'afegeixen a la llista de Redis. El client demana al Redis la llista de workers (es tracta d'un canal on hi ha totes les adreces dels workers). Llavors el client ja es pot comunicar amb els workers.

És un model de comunicació híbrid, perquè combina la llista de d'adreces dels workers a traves de redis, el qual funciona mitjançant events, i la comunicació directe del model XML-RCP.

En aquest cas la comunicació entre client i workers és síncrona i és un sistema pull. Per tant és semblant al model directe. La comunicació entre client i redis o redis i workers és basada en esdeveniments, per tant és asíncrona i push.

REDIS (Indirecte)

Els workers es subscriuen al canal de redis anomenat "functions".

El client publicarà els missatges, els quals seran les funcions, al canal del redis "functions". D'aquests missatges els workers extrauran la funció, el nom de la cua d'on després el client extraurà els resultats i els paràmetres per la funció.

Un cop els workers tinguin el resultat, aquests faran un push a la cua amb el resultat. Seguidament el client farà el pop per extreure els resultats. S'utilitza un timer per esperar a que els workers tinguin els resultats.

Tasca 3

Enunciat

Llegiu aquest article i descriu l'arquitectura distribuïda i el middleware de comunicació utilitzat en aquesta solució.

Aquest article ens parla sobre la implementació d'una arquitectura serverless, el qual es refereix a un model d'execució de computació al cloud on el proveïdor dels serveis assigna recursos de màquines virtuals sota demanda, amb funcions stateless, les quals no tenen estat, com a abstracció unificadora per al processament de dades.

Les funcions stateless, permeten una semàntica senzilla de tolerància a errors. Quan falla una funció, es reinicia (possiblement en una ubicació diferent) i s'executa en el mateix input.

Com que les funcions i totes les dades necessiten ser persistents, per tant s'utilitza S3 per guardar-les.

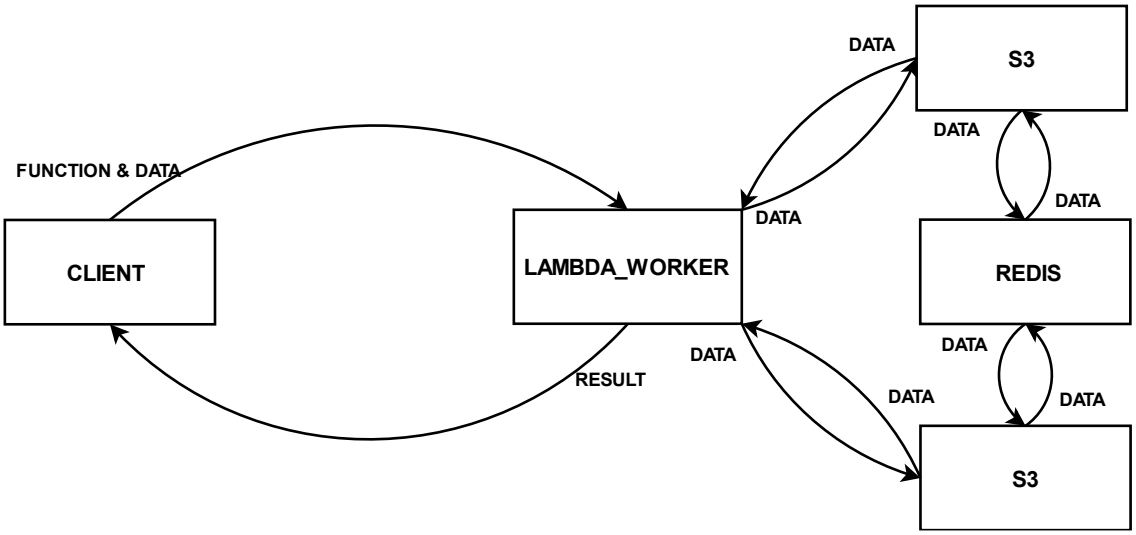
Només es requereixen escriptures atòmiques a l'emmagatzematge de dades utilitzant S3, per fer un seguiment de quines funcions han tingut èxit. Això significa, o s'emmagatzema correctament a tots els llocs on s'emmagatzemi la dada corresponent o no es fa.

El prototip que han desenvolupat, utilitza AWS Lambda per executar les funcions, i aquesta s'invoca per a cada funció diferent que s'executa. Per sobre de Lambda s'utilitza mapes de Python.

Llavors, quan un client executa una funció, el que fa primer el prototip és serialitzar la funció i les dades corresponents, les quals seguidament les emmagatzema a un contenidor de S3.

Tot seguit s'invoca instancies de Lambda necessàries per llegir les dades i la funció del contenidor. Les instancies fan un pull del la informació necessària del contenidor de S3 i utilitzant Python executa la funció. Un cop Python ha executat la funció, el client agafa el resultat.

Per combinar funcions, s'utilitza Redis entre contenidors de S3, la qual actua com una memòria cau, així és pot passar informació entre contenidor i contenidor.


S3
DATA
DATA
DATA
FUNCTION & DATA
CLIENT
LAMBDA_WORKER
DATA
REDIS
DATA
DATA
RESULT
DATA
DATA
S3