75.74 - Sistemas Distribuidos I

TP1: Escalabilidad

Grupo 08

2° Cuatrimestre 2023

Nombre y Apellido	Mail	Padrón
Prada, Joaquín	jprada@fi.uba.ar	105978
Sotelo Guerreño, Lucas Nahuel	lsotelo@fi.uba.ar	102730

1. Introducción

Se creó un sistema distribuído para analizar 6 meses de registros de precios de vuelos para proponer mejoras en la oferta a clientes.

2. Arquitectura

Objetivo de la arquitectura

El objetivo de la arquitectura es poder paralelizar lo máximo posible el filtrado y procesado de cada entrada de vuelos. Es importante que solo se lea cada entrada una única vez, ya que al ser muchos vuelos sería muy ineficiente tener que volver a leerlas. La arquitectura apunta a ser escalable, en donde se necesite más cómputo se puede escalar y acelerar su procesado.

La cantidad de datos a procesar provienen de un archivo de aproximadamente 31GB.

Puntos claves

Nuestra arquitectura tiene como corazón el módulo `communication`, encargado de la unión entre todos los componentes/containers que usen rabbit.

Luego están los `processors`, encargados de manejar toda la lógica de negocio.

Por último tenemos los `utils` que sirven como intermediarios entre processors para sacarle lógica innecesaria a los processors, como pueden ser los `filters` o `taggers`.

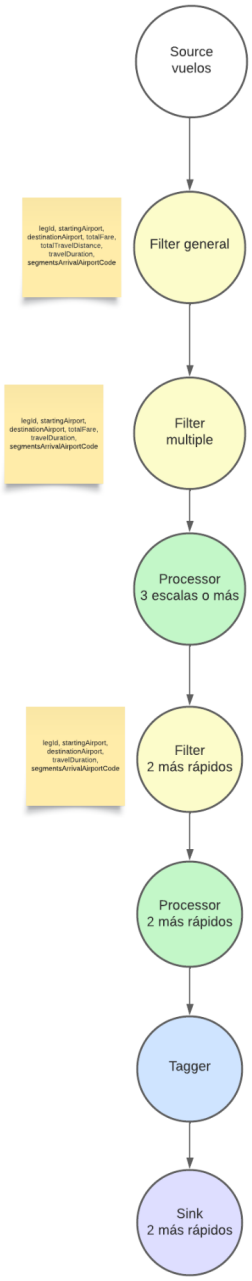
3. Escenarios

Queremos cubrir las siguientes consultas:

1. ID, trayecto, precio y escalas de vuelos de 3 escalas o más.
2. ID, trayecto y distancia total de vuelos cuya distancia total sea mayor a cuatro veces la distancia directa entre puntos origen-destino.
3. ID, trayecto, escalas y duración de los 2 vuelos más rápidos para todo trayecto con algún vuelo de 3 escalas o más.
4. El precio avg y max por trayecto de los vuelos con precio mayor a la media general de precios.

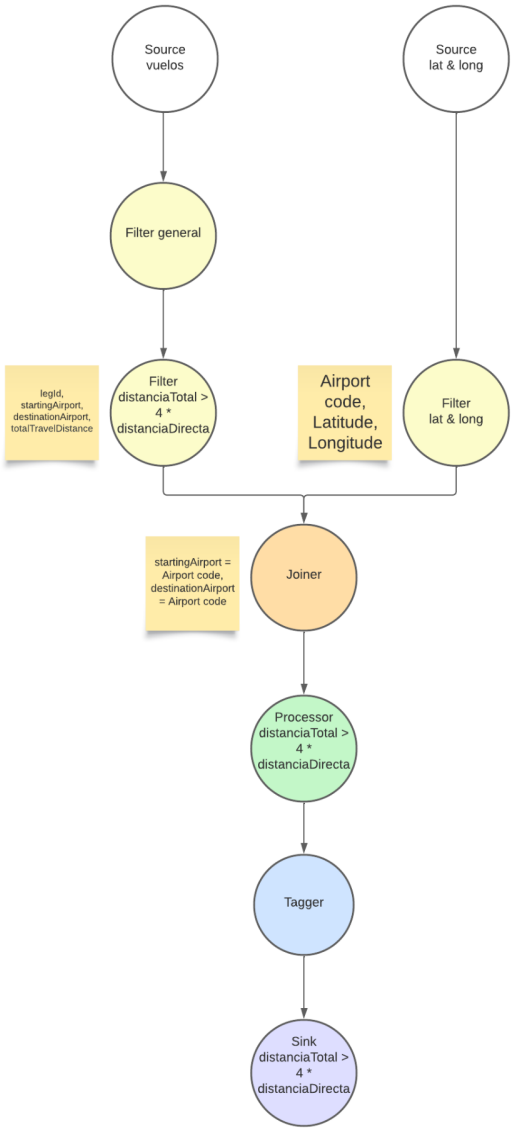
4. Vista Lógica

Query 1



Primero pasa por unos filtros para que solo le lleguen los datos necesarios al processor. Luego el processor chequea si cada vuelo recibido tiene tres escalas o más, en caso afirmativo lo envía al siguiente filter, en caso contrario lo descarta. El siguiente filter elimina las columnas innecesarias para el resultado final. Por último, el tagger agrega un tag al vuelo para indicar que tipo de resultado es y lo envía al sink donde se acumulan todos los vuelos con 3 escalas o más.

Query 2



Para realizar esta query necesitamos la latitud y longitud de cada aeropuerto, que no se encuentran en el dataset de vuelos. Por esto realizamos una junta de cada vuelo con dichos datos que provienen de otro dataset.

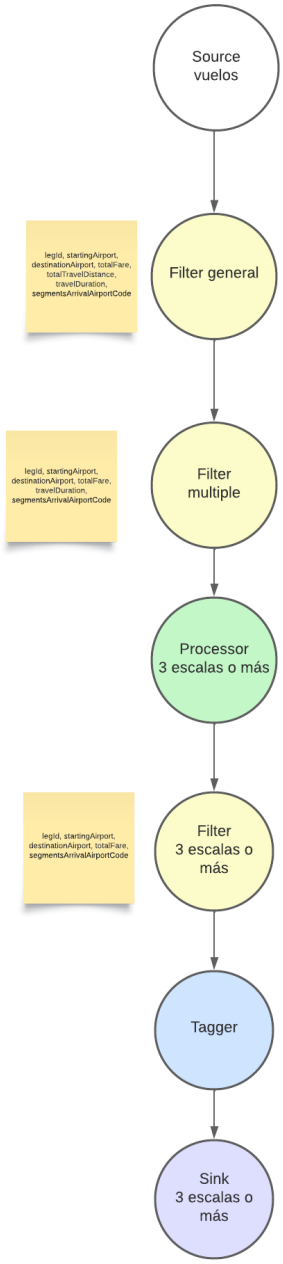
El joiner es el encargado de hacer la junta, por lo tanto, primero espera recibir todos los datos del dataset de latitud y longitud, al ser pequeño (~875kB) lo mantiene guardado en memoria.

Una vez recibido todo el dataset de latitud y longitud, comienza a escuchar pedidos del dataset de vuelos y por cada vuelo recibido le agrega la latitud y longitud del aeropuerto de salida y el aeropuerto de llegada.

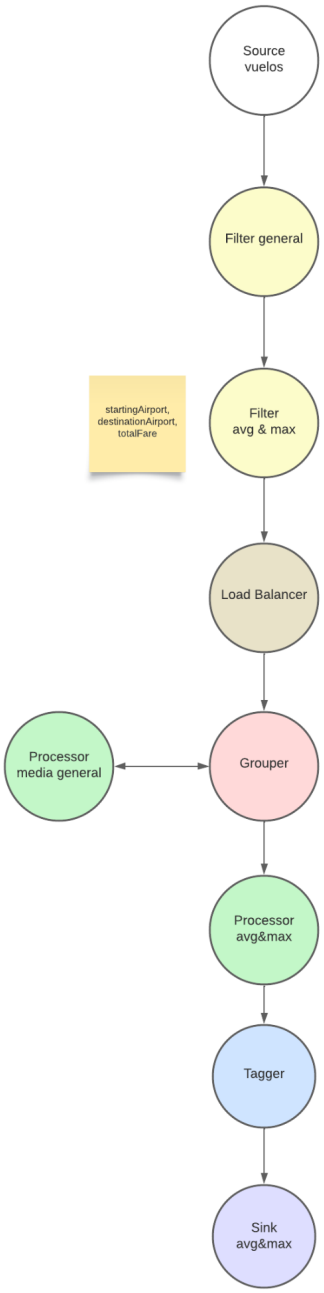
El vuelo junto a las latitudes y longitudes se envían al processor que compara la distancia total con la distancia directa y si cumple la condición es enviado al tagger.

El tagger finalmente agrega un tag al vuelo para indicar que es un resultado de este tipo de query y lo envía al sink correspondiente.

Query 3

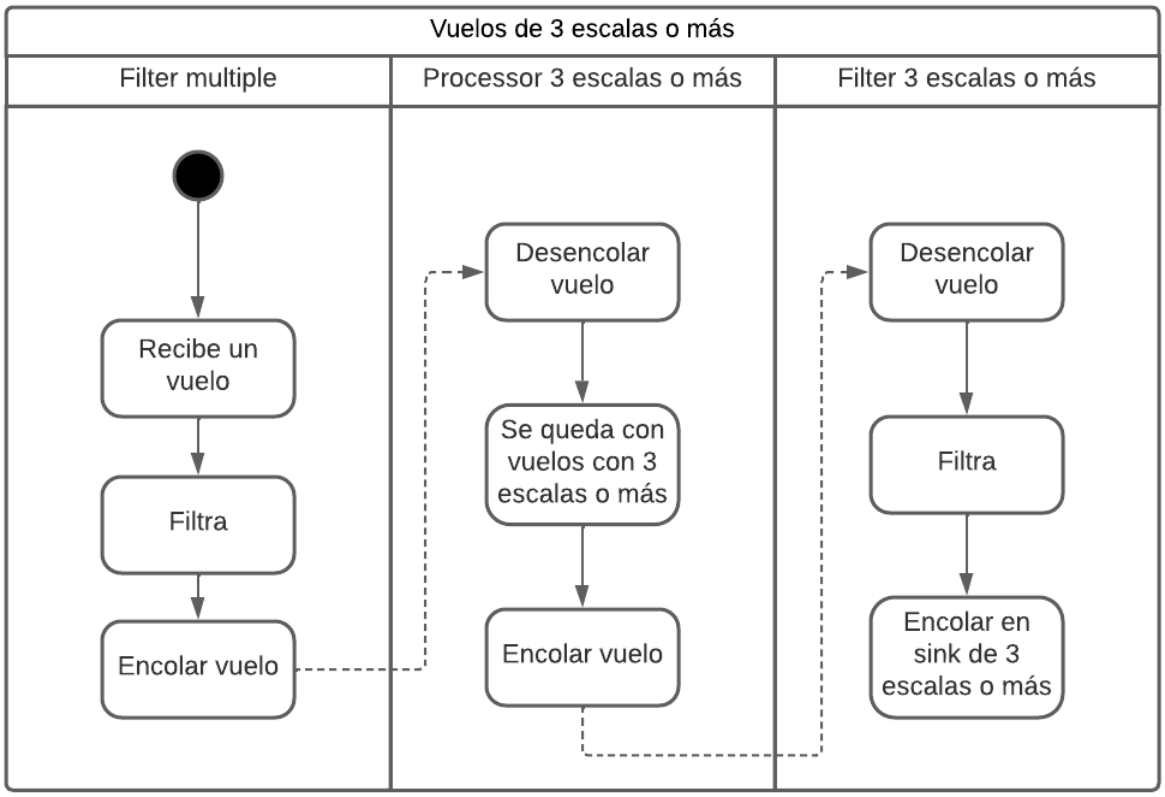

Source
vuelos
legId, startingAirport, destinationAirport, totalFare, totalTravelDistance, travelDuration, segmentsArrivalAirportCode
Filter general
legId, startingAirport, destinationAirport, totalFare, travelDuration, segmentsArrivalAirportCode
Filter
multiple
Processor
3 escalas o más
legId, startingAirport, destinationAirport, totalFare, segmentsArrivalAirportCode
Filter
3 escalas o
más
Tagger
Sink
3 escalas o
más

Query 4

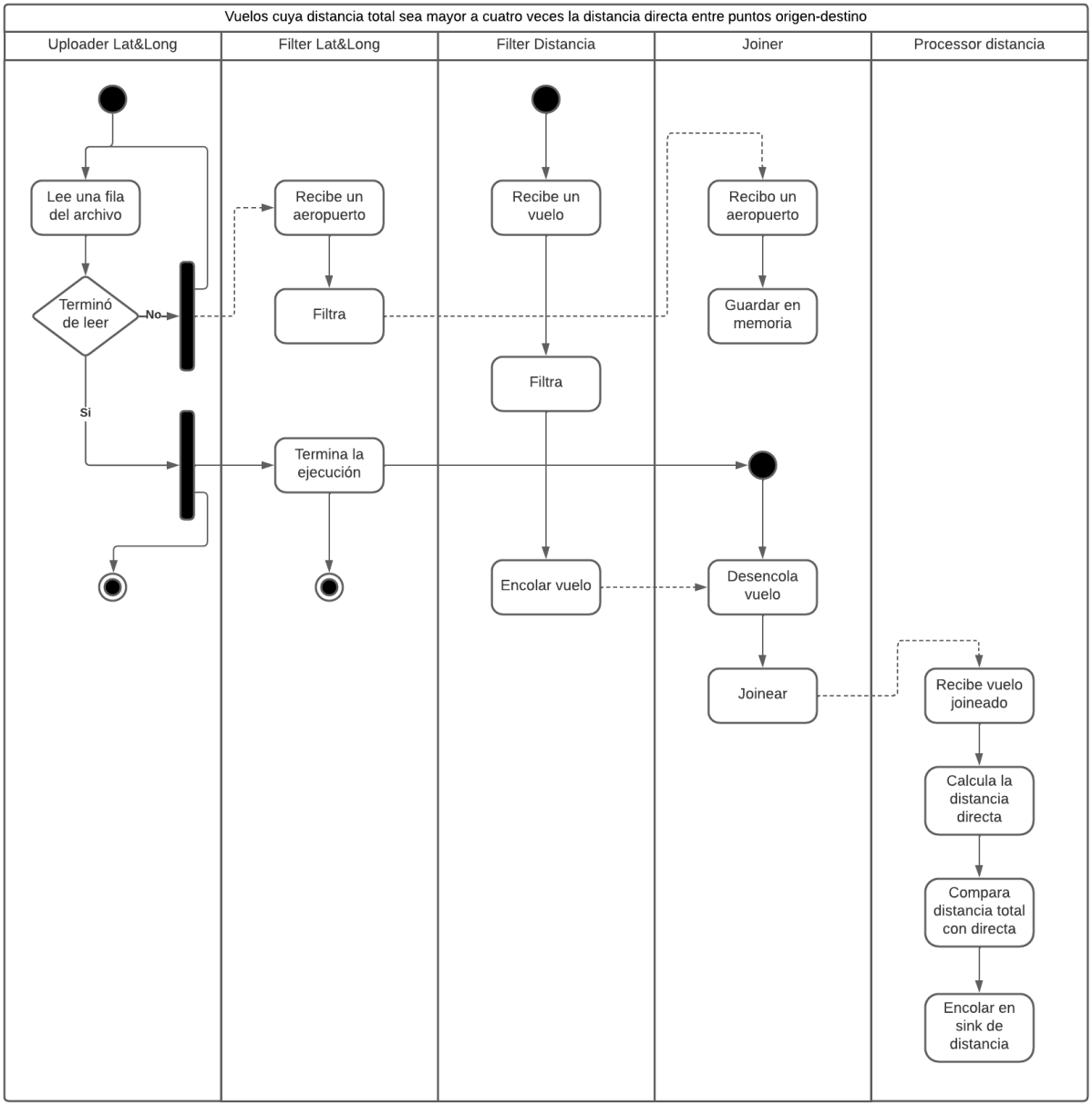

Source
vuelos
Filter general
startingAirport,
destinationAirport,
totalFare
Filter
avg & max
Load Balancer
Processor
media general
Grouper
Processor
avg&max
Tagger
Sink
avg&max

5. Vista de Proceso

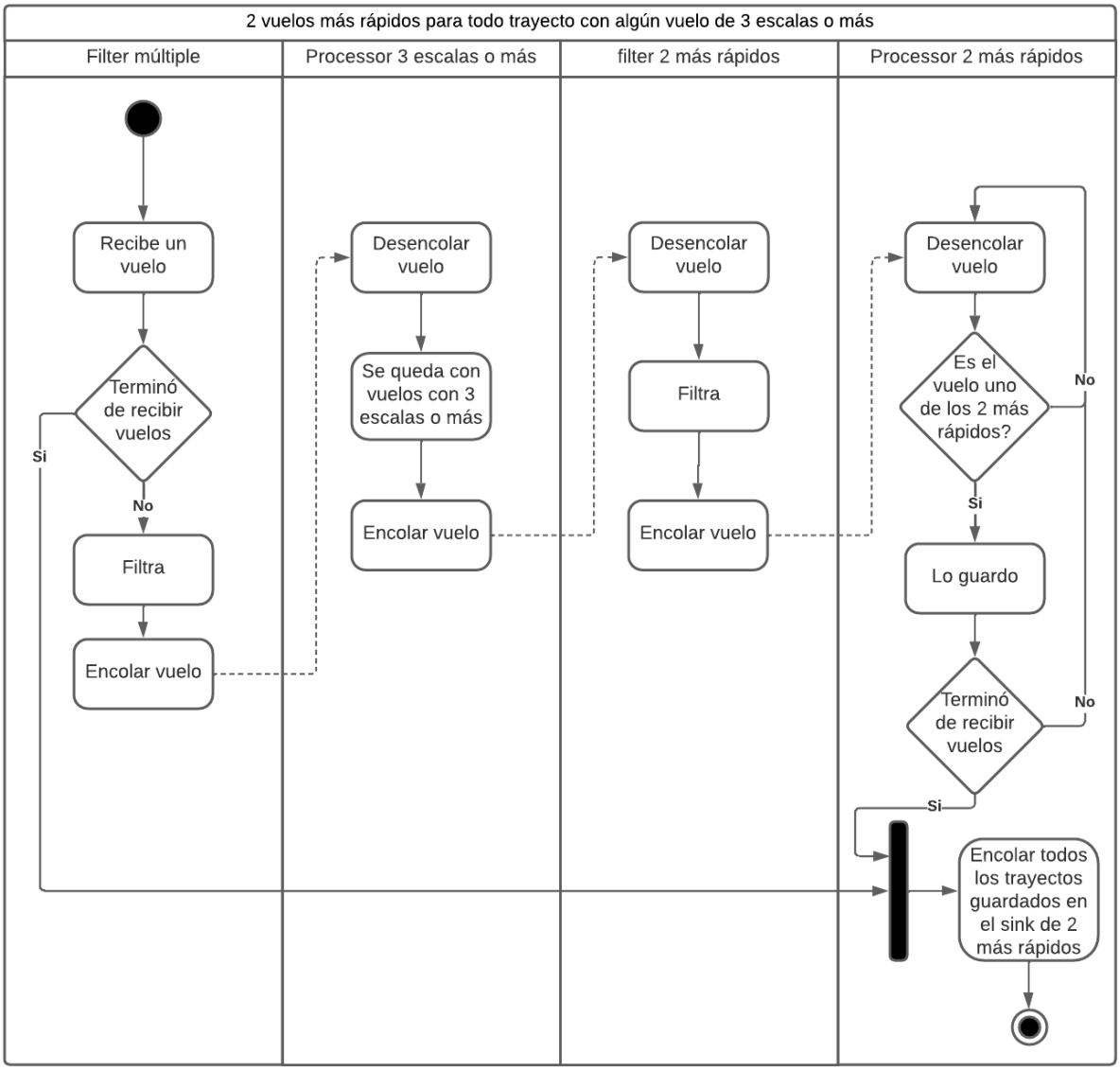
Query 1



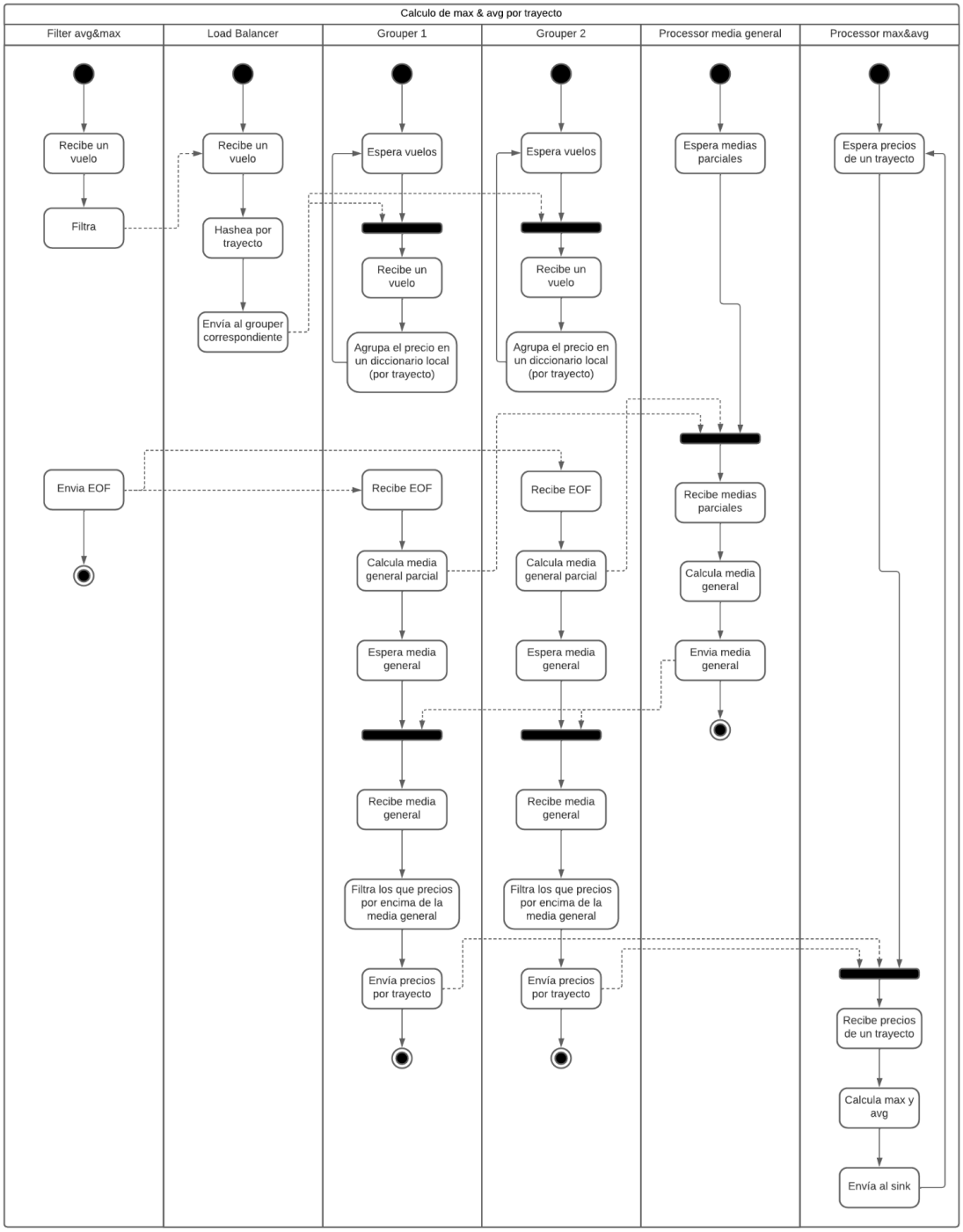
Query 2


Vuelos cuya distancia total sea mayor a cuatro veces la distancia directa entre puntos origen-destino
Uploader Lat&Long
Filter Lat&Long
Filter Distancia
Joiner
Processor distancia
Lee una fila del archivo
Terminó de leer
No
Si
Recibe un aeropuerto
Filtra
Termina la ejecución
Recibe un vuelo
Filtra
Encolar vuelo
Recibo un aeropuerto
Guardar en memoria
Desencola vuelo
Joinear
Recibe vuelo joineado
Calcula la distancia directa
Compara distancia total con directa
Encolar en sink de distancia

Query 3


2 vuelos más rápidos para todo trayecto con algún vuelo de 3 escalas o más
Filter múltiple
Processor 3 escalas o más
filter 2 más rápidos
Processor 2 más rápidos
Recibe un vuelo
Terminó de recibir vuelos
Si
No
Filtra
Encolar vuelo
Desencolar vuelo
Se queda con vuelos con 3 escalas o más
Encolar vuelo
Desencolar vuelo
Filtra
Encolar vuelo
Desencolar vuelo
Es el vuelo uno de los 2 más rápidos?
No
Si
Lo guardo
Terminó de recibir vuelos
No
Si
Encolar todos los trayectos guardados en el sink de 2 más rápidos

Query 4


Calculo de max & avg por trayecto
Filter avg&max
Load Balancer
Grouper 1
Grouper 2
Processor media general
Processor max&avg
Recibe un vuelo
Filtra
Envia EOF
Recibe un vuelo
Hashea por trayecto
Envía al grouper correspondiente
Espera vuelos
Recibe un vuelo
Agrupa el precio en un diccionario local (por trayecto)
Recibe EOF
Calcula media general parcial
Espera media general
Recibe media general
Filtra los que precios por encima de la media general
Envía precios por trayecto
Espera vuelos
Recibe un vuelo
Agrupa el precio en un diccionario local (por trayecto)
Recibe EOF
Calcula media general parcial
Espera media general
Recibe media general
Filtra los que precios por encima de la media general
Envía precios por trayecto
Espera medias parciales
Recibe medias parciales
Calcula media general
Envia media general
Espera precios de un trayecto
Recibe precios de un trayecto
Calcula max y avg
Envía al sink

6. Vista Física

Diagrama de despliegue

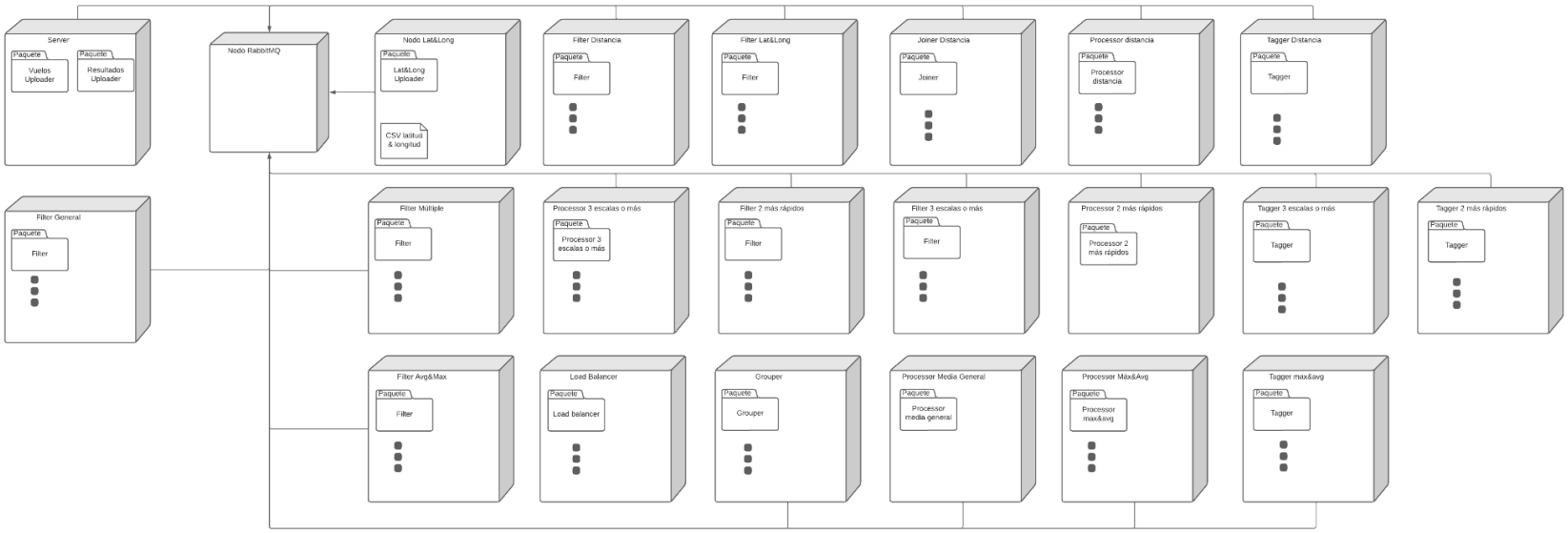


Diagrama de robustez