



75.74 Sistemas Distribuidos I

16 de Noviembre de 2021

Nombre y apellido	Padrón	Correo electrónico
Matias Ezequiel Iglesias	99635	meiglesias@fi.uba.ar

Segundo cuatrimestre 2021

Índice

1. Alcance	3
2. Arquitectura del software	3
3. Metas y restricciones arquitectónicas	3
4. Vista lógica	4
5. Vista de proceso	4
6. Vista de desarrollo	7
7. Vista física	7
8. Escenarios	9
9. Tamaño y rendimiento	9

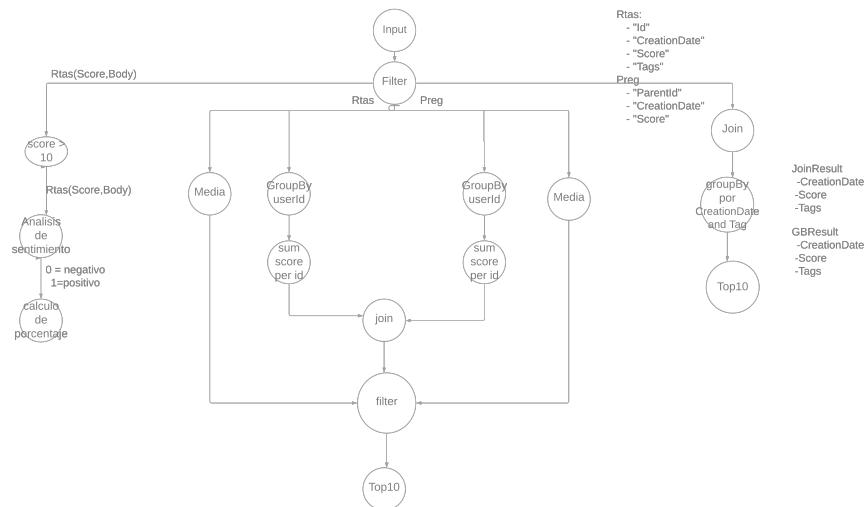
1. Alcance

Con el objetivo de poner en práctica los conceptos aprendidos en clase se desarrolló un sistema distribuidos que procese un subconjunto de preguntas y respuestas de Stack Overflow haciendo uso del lenguaje Go. El mismo debe brindar la siguiente información:

- Porcentaje de respuestas con score mayor a 10 que posea un sentiment analysis negativo
- Top 10 de usuarios según score total (preguntas + respuestas), que tengan un puntaje promedio de preguntas mayor a la media general y un puntaje promedio de respuestas mayor a la media general
- Top 10 de tags con mayor score (incluye preguntas y respuestas) de cada año

2. Arquitectura del software

Para lograr procesar la información subministrada al sistema se diseñaron distintas etapas por las que pasaría la información para obtener los resultados de los análisis requeridos. Dichas etapas y el orden de las mismas se pueden ver reflejados en el siguiente diagrama.



Es importante destacar que las etapas fueron pensadas para ser independientes entre si. Como se puede ver en el diagrama cada nodo tiene una entrada y una salida y se unen formando un camino definido por el que la información debe pasar para obtener los distintos resultados. Esto permite que cada etapa sea llevada a cabo por procesos distintos los cuales, por medio de un medio de comunicación, pueden estar físicamente distribuidos en múltiples nodos. Para el desarrollo del trabajo práctico se usó como medio de comunicación para los procesos corriendo en distintos nodos colas de (Rabbit).

3. Metas y restricciones arquitectónicas

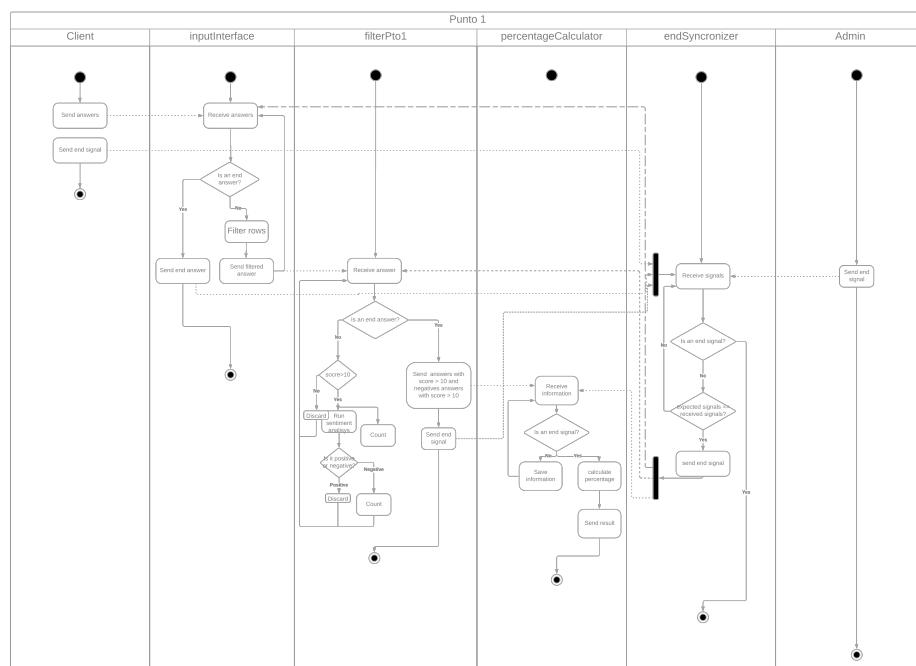
- El sistema debe estar optimizado para entornos multicoreadoras
- El sistema debe ser invocado desde un nodo que transmite los datos a ser procesados
- Se debe soportar el escalamiento de los elementos de cómputo
- De ser necesaria una comunicación basada en grupos, se requiere la definición de un middleware

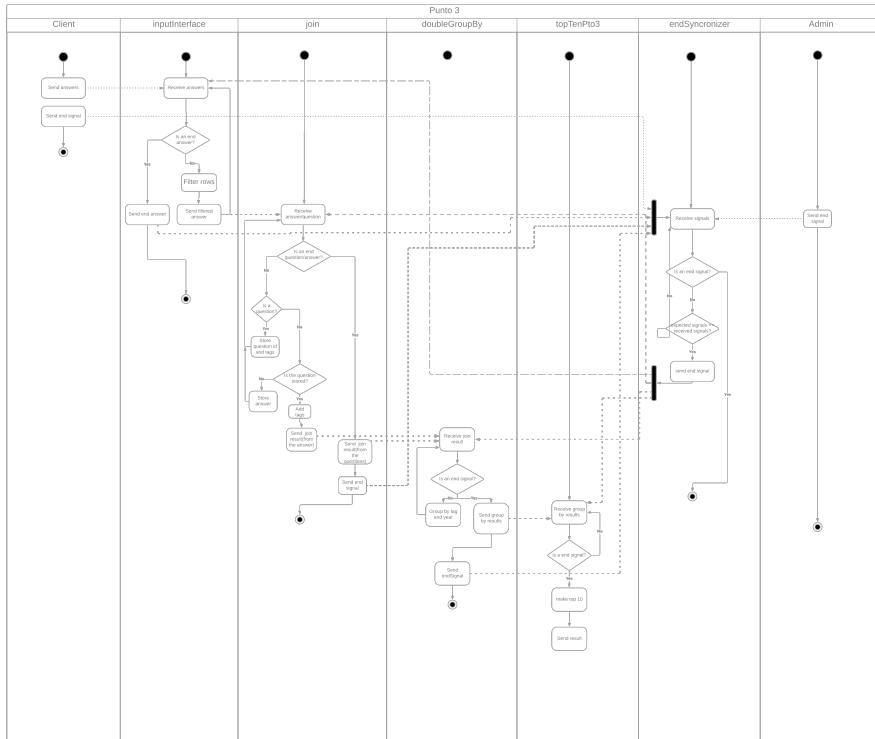
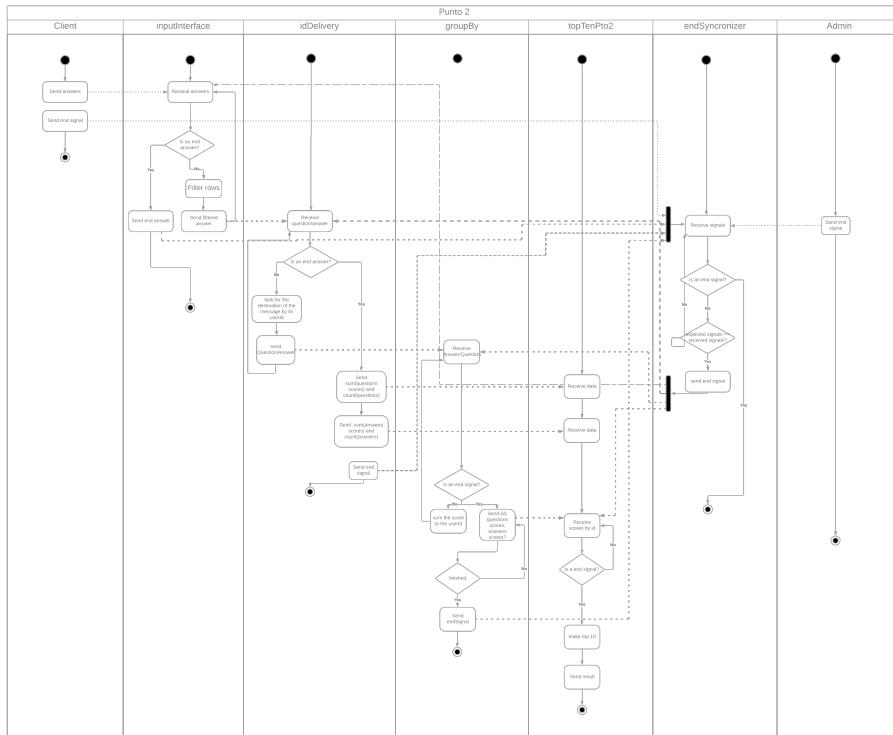
- El diseño debe permitir una fácil adaptación a otros datasets de stack overflow similares
 - Debido a restricciones en el tiempo de implementación, se permite la construcción de un sistema acoplado al modelo de negocio. No es un requerimiento la creación de una plataforma de procesamiento de datos

4. Vista lógica

5. Vista de proceso

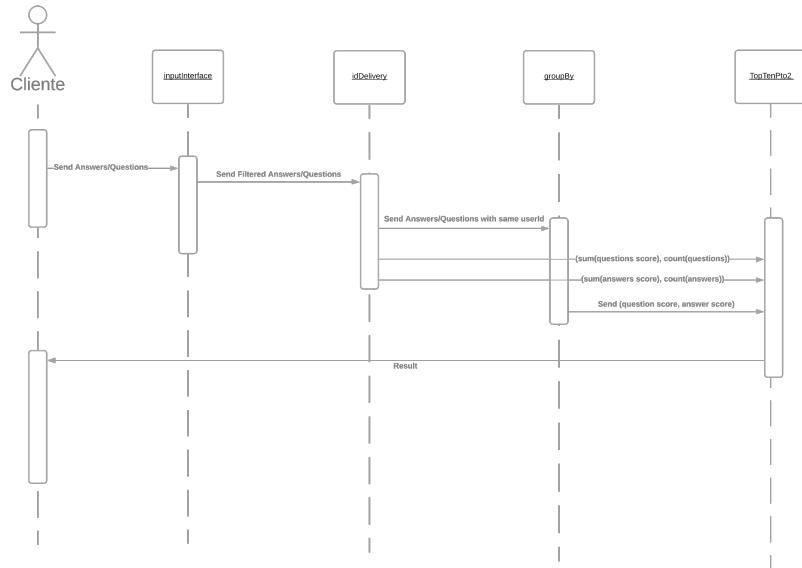
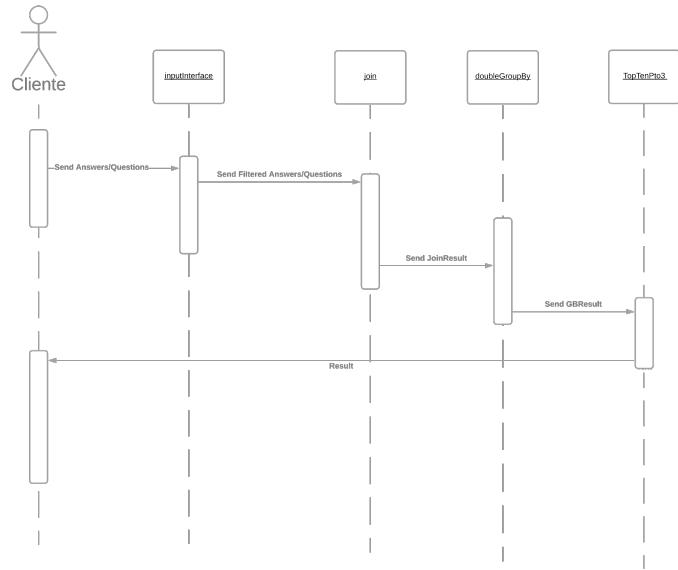
Para comenzar es importante comprender que cuales son las responsabilidades y actividades de cada nodo o proceso y como estas terminan impactando en los demás nodos logrando, en su conjunto, procesar correctamente la información ingresada por el usuario. A continuación se muestran los diagramas de actividades correspondiente a los tres puntos solicitados. Todos tienen en común como se inicia el proceso y eso se debe a que luego de que el cliente empieza a enviar los datos, los mismos pasen por un filtro donde los campos que no se van a utilizar se descartan y luego los datos se procesan en paralelo, de forma independiente para cada punto. También es importante destacar que en los diagramas se muestra como es gestionada el fin de cada nodo dentro del sistema una vez terminado el procesamiento



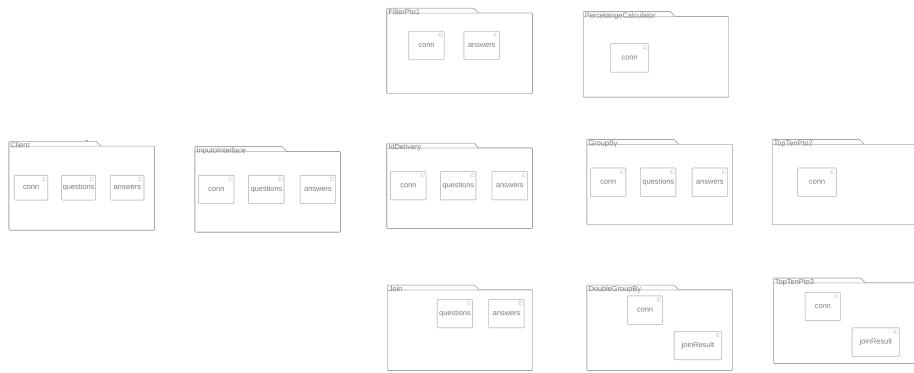


También, con el fin entender el flujo de información y como el mismo pasa de nodo a nodo, se realizaron los siguientes

diagramas de secuencia donde se pueden ver tanto el punto 2 como el punto 3:

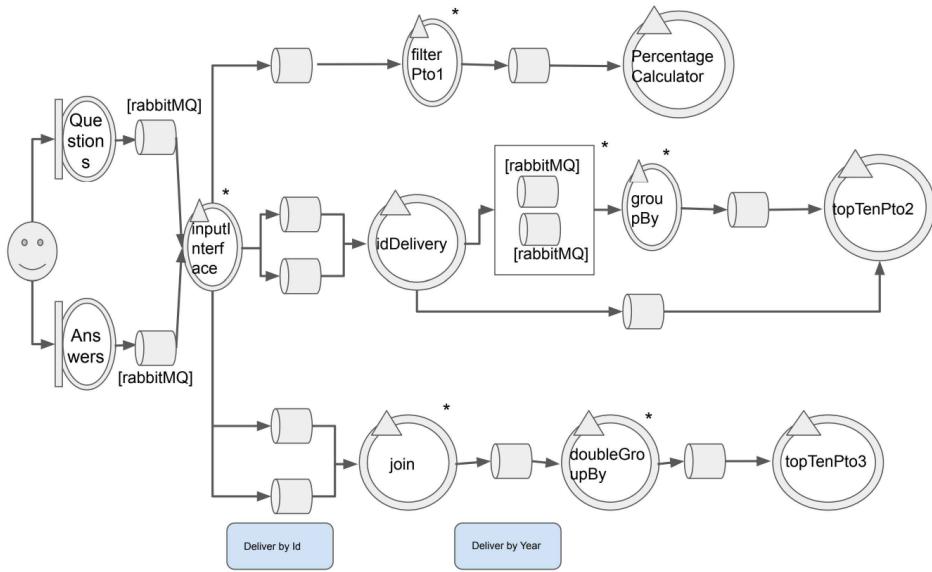


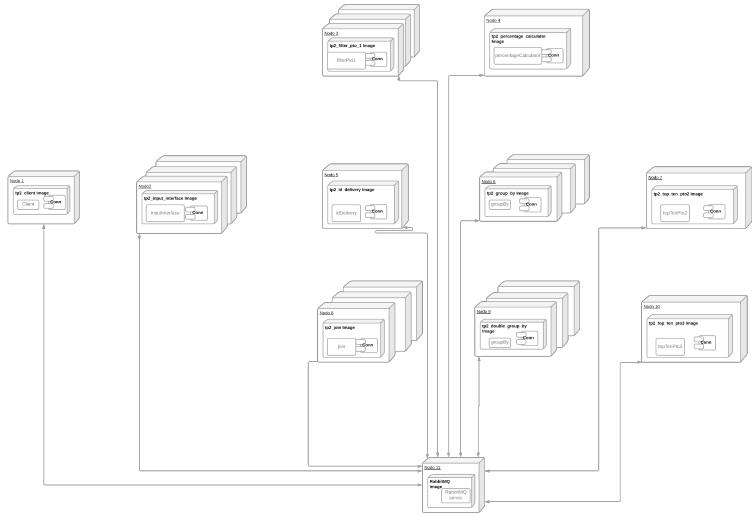
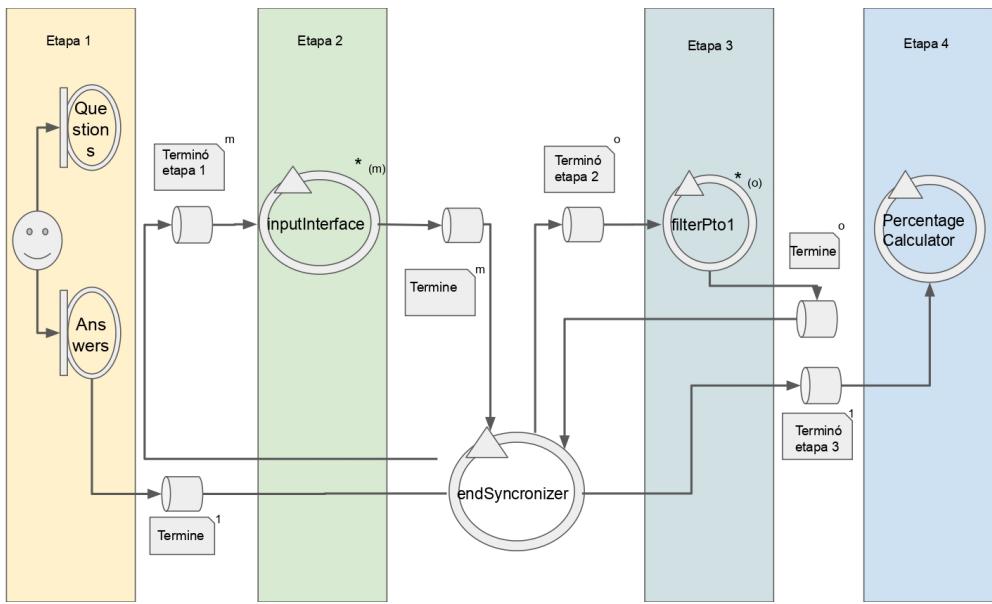
6. Vista de desarrollo



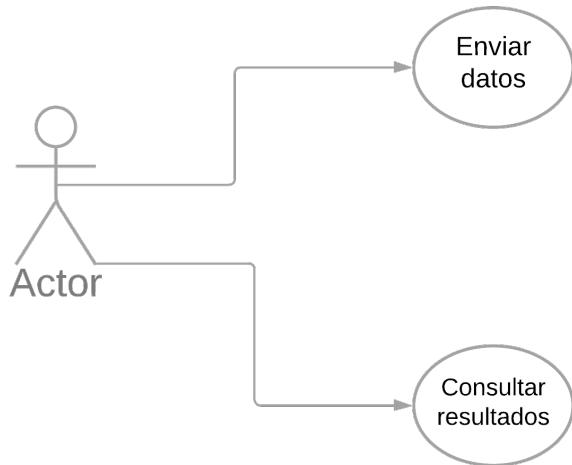
7. Vista física

Una vez comprendido el funcionamiento del sistema, es necesario entender el diseño de la arquitectura del mismo y como este puede distribuirse formando así una sistema distribuido multicompacting. Para eso primero es necesario comprender cuales son los nodos que lo componen y como interactúan entre si. A continuación se mostraran el diagrama de robustez del sistema, el cual con fin de mejorar su legibilidad fue separado en dos. En el primero podemos observar el sistema en si y sus nodos principales, mientras que en el otro, se ve reflejado en mecanismo utilizado para gestionar el fin de cada etapa. En el segundo diagrama se muestra el diseño de este mecanismo para el primer punto, el mismo se puede extender a los demás ya que la idea es la misma, lo único que cambia es que como en los otros puntos se procesan tanto preguntas como respuestas, en vez de haber un solo canal, hay dos canales (colas) uno para las preguntas y otro para las respuestas.



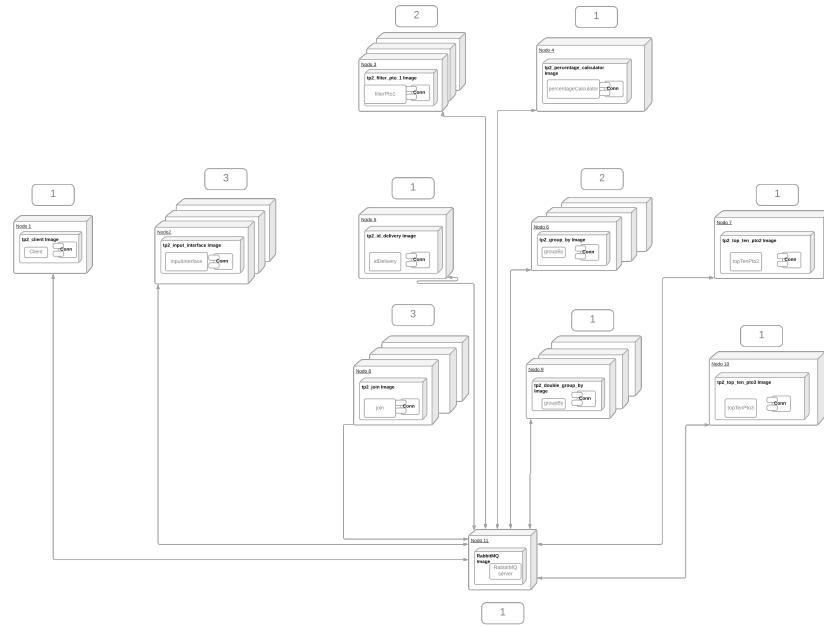


8. Escenarios



9. Tamaño y rendimiento

En cuanto al rendimiento obtenido hay que considerar que por limitaciones de recursos todos el sistema se corrió en un solo nodo, en el cual haciendo uso de contenedores de Docker, simuló los distintos nodos en los cuales corren los distintos procesos. Puntualmente el nodo en el que se testeo el sistema contaba con un procesador i5 con 4 núcleos y 16 GB de memoria RAM. Para probar el sistema se escalaron algunos de los nodos para así lograr una mayor rendimiento del sistema en general. La configuración que mejor resultó se puede ver en el siguiente diagrama. De esta forma se logro procesar los datos provisto por la cátedra (archivos reducidos a la mitad de su tamaño original) consumiendo un total de 4GB y tardando aproximadamente 10 minutos. En el siguiente link ([demo](#)) se puede ver una demo del sistema. Cabe destacar que los resultados obtenidos en la demo difieren de los expuestos en este apartado ya que el programa utilizado para grabar la pantalla consumió muchos recursos del nodo bajando en rendimiento del sistema.



Después de realizadas varias pruebas en las cuales se sobrecargó al sistema de distintas maneras se llegó a la conclusión de que el mismo se ve limitado particularmente en el flujo correspondiente al punto dos. Viendo el diagrama de robustez se puede ver que luego del filtrado, la información pasa por el nodo idDelivery el cual es único y no se puede escalar horizontalmente. También, se notó una gran diferencia entre el tiempo de encolado y el tiempo de desencolado de una cola de rabbit (siendo este último mucho mayor en comparación al primero). Esta diferencia entre los tiempos hace que el sistema se vea obligado a escalar en la etapa inputInterface para así poder lidar con la carga representada por el cliente. El hecho de que no se pueda escalar la etapa idDelivery sumado a la necesidad de escalar en la etapa inputInterface lleva a que en las colas de entrada del nodo idDelivery se enculen gran cantidad de datos, los cuales sobrepasan la capacidad de este nodo ralentizando todo el sistema ya que ese procesamiento va a tardar considerablemente más que los otros dos. También es importante mencionar que dejando de lado este problema, el sistema se vería beneficiado al seguir escalando la etapa inputInterface pero debido al problema encontrado, en caso de seguir escalando en esa etapa, peor sería el cuello de botella en el punto dos, es por eso que se decidió no modificar la configuración actual. Para lidar con este problema y mejorar considerablemente el rendimiento general del sistema se debe eliminar el nodo idDelivery. Para eso se necesita reemplazar sus dos responsabilidades. La primera de ellas es de repartir coherentemente las preguntas y respuestas, tarea que puede ser asignada al inputInterface de forma similar a como lo hace con la información pasada a la etapa join del punto tres. La segunda es la de pasar los datos de las sumas de los puntajes y las cantidades totales de las preguntas y respuestas. Esta tarea puede ser llevada a cabo por la etapa groupBy pero esto generaría la necesidad de un nodo que junte toda la información subministrada por los distintos nodos de la etapa groupBy. Esta responsabilidad podría recaer sobre el nodo topTenPto2 o sobre un nuevo nodo el cual luego le pasaría la información a este último. Estos cambios no se realizaron ya que el problema fue detectado al final del desarrollo en las etapas de prueba y no se contaba con el tiempo suficiente para plantear este reestructuración.