Sistemas Distribuidos: Proyecto My WEB SEARCH ENGINE (MyWSE)

Profesor: Nicolás Hidalgo
Ayudantes: Giovanni Benussi - Luis Celedón

Objetivo:
El objetivo del proyecto consiste diseñar un prototipo de Buscador Web a.k.a **My WEB SEARCH ENGINE** (**MyWSE**) utilizando el lenguaje de programación Java, modelando el buscador con servicios REST [1].

Enunciado:
Hoy en día los motores de búsqueda en la web (WSE por sus siglas en inglés) son muy importantes, ya que permiten que un usuario encuentre rápidamente la información que está buscando. Estos motores son utilizados por millones de personas diariamente, por lo cual deben ser capaces de soportar una gran cantidad de peticiones simultáneas. Esto es posible gracias al uso de infraestructuras distribuidas altamente escalables en conjunto con técnicas especialmente diseñadas para lidiar con los requerimientos de los usuarios del buscador.

Un buscador típico se organiza bajo una arquitectura orientada a servicios (o SOA por sus siglas en inglés). Tradicionalmente existen 3 servicios básicos: Front-Service, Caching-Service e Index-Service. Cada uno de estos servicios realiza tareas autónomas e interactúa con el resto de servicios por medio del paso de mensajes para la resolución de las consultas recibidas.

En el presente proyecto de laboratorio se propone la creación de un buscador Web llamado **MyWSE** el cual sea capaz de resolver consultas de búsqueda proveyendo como resultado una página que contiene una lista con los documentos mejor rankeados para la consulta realizada.

Modelo:
El modelo de **MyWSE** consta de 3 servicios básicos: *front-service, caching-service, e index-service*. La comunicación de estos 3 servicios deberá ser implementada bajo la arquitectura REST. Todos los servicios poseen réplicas que les permiten balancear la carga y tolerar fallas ante caídas. Adicionalmente, los servicios de cache e índice poseen particiones para distribuir los datos. Las funcionalidades de estos servicios se detallan a continuación:

- **Front-Service (FS):** recibe consultas de usuarios filtrándolas con el objetivo de eliminar las palabras no relevantes para la búsqueda (“stop-words”). Posteriormente re-dirige la consulta al servicio de cache en busca de resultados pre-computados. Si dicha consulta no está en el cache (cache miss), se re-dirige la consulta al servicio de índice para su procesamiento. Al momento de redirigir las consultas el front-service debe utilizar un algoritmo que permita balancear la carga entre las diferentes réplicas del servicio con que se comunica.

- **Caching-Service (CS):** mantiene consultas con sus resultados pre-computados con el objetivo de reducir la carga del backend (servicio de índice). El servicio de cache recibe peticiones desde el

front-service y responde con los resultados en caso de encontrarlos disponibles pre-computados (*cache hit*), o con un mensaje de *cache miss*, en caso que los resultados no estén disponibles.

- **Index-Service (IS):** es el encargado del procesamiento de las consultas, para ello debe poseer un mecanismo de almacenamiento para los documentos y un índice invertido para poder realizar las búsquedas de manera eficiente. Adicionalmente debe ser capaz de seleccionar los top documentos que cumplen con la consulta, para ello debe implementar un mecanismo de ranking basado en el número de ocurrencias del término buscado en el texto del documento. Finalmente genera una página con los resultados seleccionados en el ranking y responde con estos al front-service para entregar la consulta al usuario. Paralelamente el índice envía los resultados computados y la consulta a el cache para su inclusión.

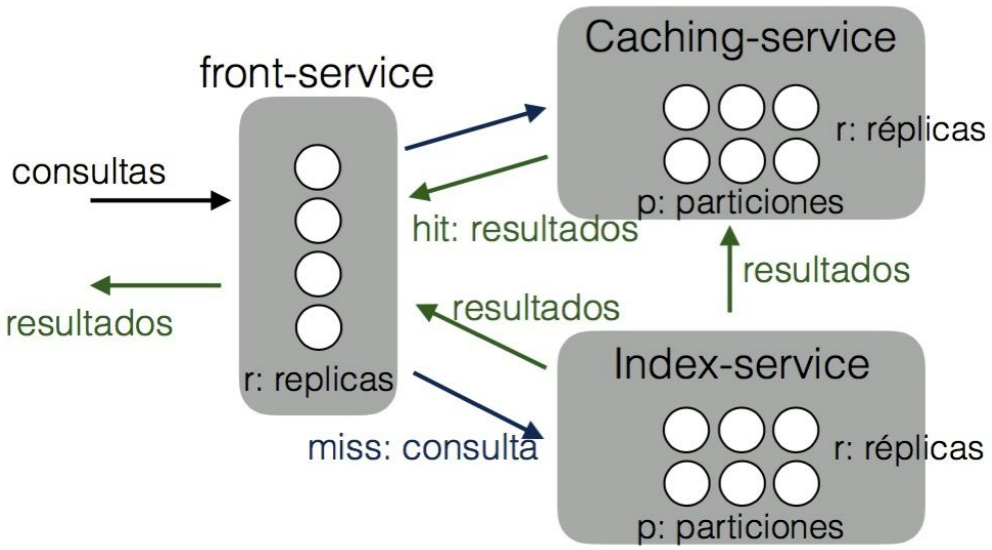

Figura 1: Arquitectura de MyWSE

La Figura 1 presenta la presenta el diagrama de buscador con las interacciones entre los diferentes servicios para resolver la consulta.

Formato consultas:
El modelo de consultas a recibir consiste en un String de largo variable, tal como se realiza con un buscador Web tradicional.

e.g: "*Servicios REST en Java*".

Requerimientos:
A continuación se detallan los requerimientos a ser evaluados en cada uno de los servicios.

Generales:

- Arquitectura REST de comunicación entre los servicios: CS, IS, FS
- Implementación Multi-thread
- Cada servicio debe poder correr en una máquina diferente en la entrega final

Caching-Service (CS):

- Esquema de Cache dinámico-estático
- Política de remoción de entradas LRU
- Soporte para réplicas (balance de carga y tolerancia a fallas)
- Soporte de particiones (balance de carga)
- Mecanismo de distribución de carga entre réplicas (balance de carga)
- Manejo de réplicas de cache (consistencia)
- API de comunicación REST:
    - recepción consulta
    - respuesta a consulta

Index-Service (IS):

- Generación índice invertido
- Inserción nueva información y re-cálculo del índice
- Manejo de particiones
- Manejo de réplicas
- Ranking de resultados
- API de comunicación REST:
    - recepción consulta
    - respuesta a consulta

Front-Service (FS):

- Manejo de réplicas
- Recepción de consultas
- Filtrado Stopwords
- Distribución de consultas (Balance carga)
- Respuesta con resultados al usuario (result page)
- API de comunicación REST:
    - recepción consulta
    - respuesta a consulta

Entregas:

Se realizarán 3 entregas correspondientes a cada uno de los módulos del buscador. Las fechas establecidas para las entregas se estipulan a continuación y serán **inamovibles**:

Entrega 1 - Caching Service: 3 de Diciembre 2015.
Entrega 2 - Index Service: 30 de Diciembre 2015.
Entrega MyWSE: 22 de Enero 2016.

La **entrega 1** consiste en la implementación del *Caching-Service*. La **entrega 2**, consiste en la implementación del *Index-Service,* finalmente la **última entrega** consiste en la implementación del buscador con sus 3 servicios funcionando e interactuando bajo el modelo planteado utilizando la arquitectura REST. En esta última entrega se pueden plantear mejoras a los servicios previamente desarrollados de manera opcional. Note que para las 2 primeras entregas los servicios pueden ser implementados de manera local (vale decir, utilizando un sólo equipo), sin embargo, la entrega final debe ser presentada utilizando una máquinas para cada servicio implementado.

Las entregas deberán ser hechas vía **UsachVirtual** adjuntando el link al repositorio **Github** de su proyecto. Para la entrega se debe realizar un **release** del proyecto para cada una de las etapas, es decir, una entrega formal de la implementación realizada, utilizando un método de versionamiento estándar. Esto es de carácter obligatorio y para cada revisión se considerará solamente el release correspondiente. Todos los códigos entregados serán sometidos a evaluación *anticopia*. Las copias detectadas serán calificadas con la nota mínima en el proyecto, significando la **reprobación inmediata** del laboratorio.

Consideraciones:

- Se trabajará en grupos de **2 personas** y la implementación deberá ser desarrollada en el lenguaje de programación **Java**.
- La ponderación de las entregas será **30/30/40**, las primeras dos entregas valen un 30% y la entrega final 40%.
- Entregas atrasadas serán aceptadas hasta un periodo máximo de **3 días** con un descuento de 1 punto de la nota por día de atraso. i.e: una entrega con 3 días de atraso parte con una nota máxima de 4.0.
- Podrán ser bonificadas aquellas soluciones innovadoras que implementen optimizaciones sobre los requerimientos básicos propuestos en el enunciado.
- El uso de librerías externas está permitido referenciando adecuadamente la fuente. Estas deben ser usadas para facilitar el desarrollo de tareas que no estén relacionadas o realicen directamente las funcionalidades requeridas. Ante dudas respecto a este punto, consultar con los ayudantes.
- Se proveerá para cada entrega una pauta de evaluación con la cual serán evaluadas las entregas.

Referencias:

[1] http://www.ibm.com/developerworks/library/ws-restful/
[2] http://link.springer.com/chapter/10.1007%2F978-3-540-75530-2_7
[3] http://infolab.stanford.edu/pub/papers/google.pdf
[4] http://cms.searchenginewatch.com/digital_assets/3859/how-search-engines-work-mike-grehan.pdf
[5] http://www0.unsl.edu.ar/~gvcosta/documents/ispa2012_submission_160.pdf
[6] http://users.dcc.uchile.cl/~mmarin/papers/hpdc10.pdf