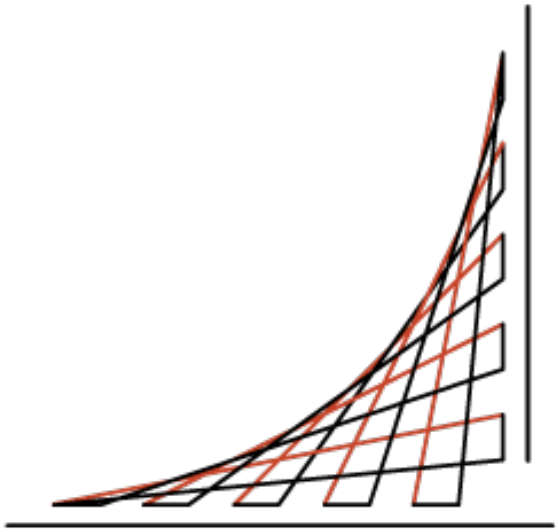

INGENIERÍA DE SISTEMAS - AREP 2020-2

Arquitecturas Empresariales

LABORATORIO 2 : EJERCICIO HEROKU-SPARK

Luis Daniel Benavides Navarro

Authors:
Juan Camilo Rojas Ortiz

Índice

Resumen

Este artículo presenta la descripción de la teoría y la arquitectura utilizada para la creación de un sistema capaz de leer un conjunto de datos de una página web y realizar el cálculo de la media y la desviación estándar de estos desde una aplicación web construida usando el Mircro-Framework Spark, utilizando una implementación propia de lista enlazada.La arquitectura se describe en terminos de las abstracciones básicas de memoria, compiladores y vinculos de enlace.Al final se presentan los datos de prueba (entrada y salida) propuestos para comprobar el correcto funcionamiento del programa y los resultados obtenidos al realizar el calculo desde la página web con estos datos, con lo que se concluye que la implementación calcula correctamente la media y la desviación estándar de los datos ingresados,además se concluye que Spark es un framework que facilita la creación de aplicaciones web y reduce en gran medida el esfuerzo del desarrollador.

1. Introducción

Actualmente la infraestructura en la nube ha tomado gran fuerza dado que permite acceder a recursos computacionales muy potentes a precios accesibles y permitiendo que el usuario se evite los esfuerzos de instalación, mantenimiento y compra de estos equipos.Consecuencia de esto, muchas compañías pueden diseñar sus aplicaciones web y llevarlas a un entorno de producción en mucho menos tiempo, sin embargo muchos de los frameworks usados actualmente tienden a ser pesados y complejos de utilizar por lo que requieren de mucho tiempo y esfuerzo de desarrollo, como solución a esto se están desarrollando micro-frameworks que permitan acelerar y facilitar la creación de estas aplicaciones, un ejemplo de esto es el micro-framwork Spark.
El objetivo de este laboratorio crear una aplicación web para calcular la media y la desviación estándar de n números reales que se leen de la página web y se envian a una API-REST desplegada en Heroku que permite hacer los calculos deseados utilizando una implementación propia de lista enlazada(*Linked List*)
Con el objetivo de explicar la arquitectura del sistema, esta se dividirá en 3 secciones, donde se explica la implementación y la teoría detrás de la lista enlazada creada y el módulo de estadísticas propuesto, además en la última sección se explica el funcionamiento del controlador construido con Spark y la arquitectura general del sistema.

2. Lista enlazada

Una lista enlazada es una estructura lineal de datos en la que cada elemento es un objeto separado.Los elementos de la lista enlazada no se almacenan en un lugar contiguo; los elementos se enlazan mediante punteros.Cada nodo de una lista se compone de dos elementos: los datos y una referencia al siguiente nodo. El último nodo tiene una referencia a nulo.[1].

2.1. Implementación

Para este problema se implementó una variación de la lista enlazada, la lista doblemente enlazada, en la que cada elemento además de tener una referencia al siguiente nodo también cuenta con una referencia al nodo anterior y el primer nodo cuenta con una referencia previa nula.La decisión de implementar la lista doblemente enlazada se debe a que esta permite buscar un elemento realizando $\frac{n}{2}$ operaciones con

n siendo la longitud de la lista, por lo que si en el futuro se necesitan funcionalidades que requieran de esto, se podrán implementar de forma eficiente.En la figura 1 se puede ver la estructura de una lista doblemente encadenada.

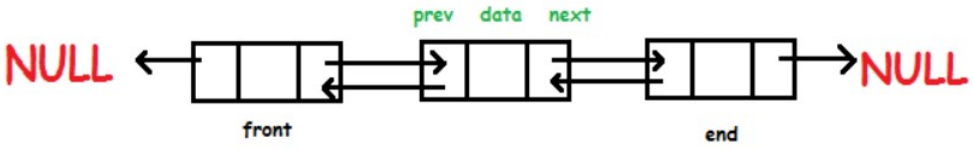

Figura 1: Lista doblemente encadenada [2]

Adicionalmente se realizó una implementación generalizada de la lista, por lo que permite almacenar y hacer operaciones con objetos de cualquier tipo,también se creó la clase implementando la interfaz *Iterable* por lo que la lógica correspondiente a la iteración de la estructura de datos se delega a la clase *LinkedListIterator* y del lado del consumidor se facilita por medio del método *foreach()*, A continuación se presenta el diagrama de clases correspondiente a la implementación realizada

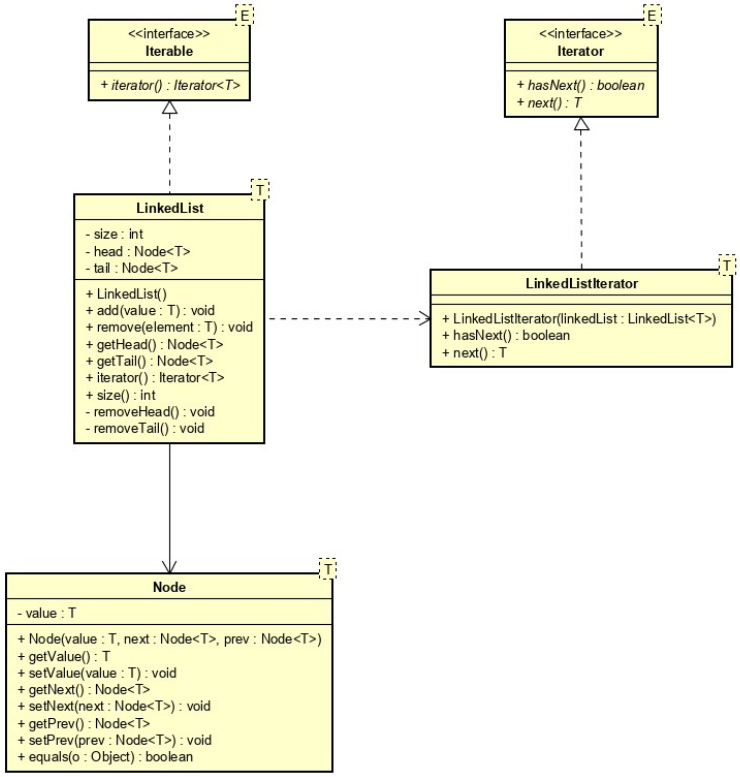

Figura 2: Diseño de la LinkedList implementada

Las operaciones principales que ofrece la estructura de datos implementada son :

- *add* : Permite añadir un nuevo elemento al final de la lista
- *remove*: Permite remover un elemento especificado si este si encuentra en la lista
- *size*: Retorna el tamaño de la lista

3. Estadísticas

3.1. Media

La media es el promedio de un conjunto de datos. La media es la medida más común de la ubicación de un conjunto de números. El promedio localiza el centro de los datos[3].Se calcula de la siguiente manera:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

3.2. Desviación estándar

La desviación estándar es una medida de la separación o dispersión de un conjunto de datos. Cuanto más ampliamente se separan los valores, mayor es la desviación estándar[3]. Su fórmula es :

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \tag{2}$$

3.3. Implementación

Para el cálculo de la media y la desviación estándar se creó la clase *StatisticsModule* que proporciona los siguientes métodos estáticos:

- *mean*: Calcula la media de un conjunto de datos almacenado en una LinkedList que recibe como parámetro, retorna la respueta redondeada a 2 decimales
- *standardDeviation*: Calcula la desviación estándar de un conjunto de datos almacenado en una LinkedList que recibe como parámetro, retorna la respuesta redondeada a 2 decimales

Se decidió que los métodos sean estáticos para que estos puedan ser utilizados sin necesidad de instanciar la clase, lo que facilita su uso.

4. Controlador Spark y Arquitectura general

La arquitectura general del sistema se presenta en la figura 3, en la que podemos observar las abstracciones de compiladores(cajas de color verde) y las abstracciones de enlaces de comunicación(líneas de color rojo). Además se muestra una arquitectura cliente-servidor en la que el cliente envia peticiones al servidor por medio de peticiones http, el servidor cuenta con un controlador contruido usando el micro-framework Spark y que ofrece 2 endpoints descritos a continuación :

- *Get ../*: Este endpoint se encarga de retornar al usuario el archivo estático index.html, que funciona como interfaz gráfica entre el usuario y la aplicación
- *Post ../calcular*: Se encarga de recibir los datos enviados por el usuario y realizar el cálculo de la media y de la desviación estándar apoyandose en una linked list.

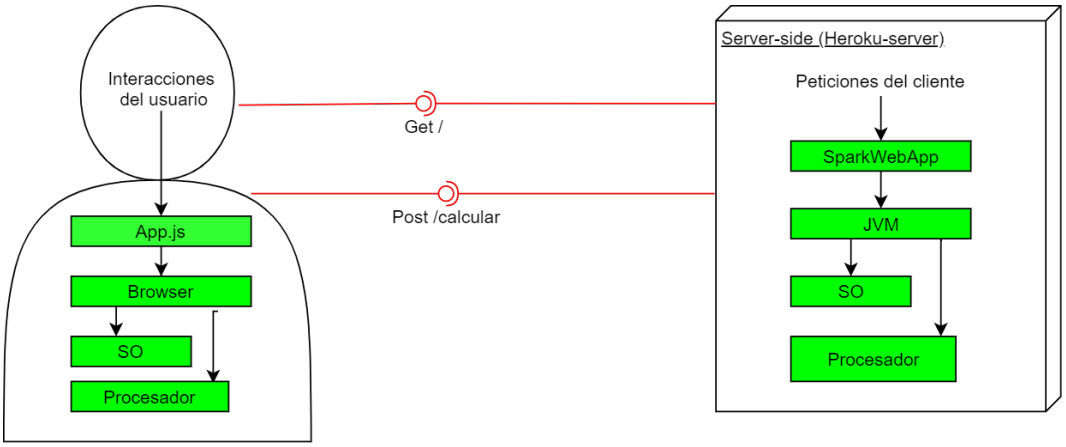

Figura 3: Arquitectura general con compiladores

En terminos de compiladores, el usuario activa la ejecución de algunas funciones al interactuar con la interfaz, estas ejecutan instrucciones dadas por el browser que utiliza modulos de sistema operativo e instrucciones del procesador,en el lado del servidor, el controlador web se activa al recibir peticiones, una vez activado ejecuta instrucciones sobre la Java Virtual Machine(JVM) quién utiliza modulos del sistema operativo y del procesador.
Los mensajes HTTP utilizados para conectar las dos partes de la aplicación son la representación de la abstracción de vinculos de enlace, es importante resaltar que esta comunicación no solo se da en la dirección cliente-servidor sino que también el servidor es capaz de responderle al cliente por medio de estos mensajes.
En cuanto a abstracciones de memoria se tiene la linked list implementada, que aplica la escritura de datos con su método *add* y la lectura de datos por medio de la implementación de una estructura iterable, el iterador implementado permite recorrer secuencialmente toda la estructura.

5. Pruebas

Para verificar el correcto funcionamiento del programa se utilizaron los datos mostrados en la figura 3, para esto, los valores de la columna 1 y la columna 2 se ingresaron por medio de la página web y se realizó el calculo.Los valores esperados tras ejecutar el programa con estos valores se pueden observar en la figura 4 .

Column 1	Column 2
Estimate Proxy Size	Development Hours
160	15.0
591	69.9
114	6.5
229	22.4
230	28.4
270	65.9
128	19.4
1657	198.7
624	38.8
1503	138.2

Figura 4: Valores de entrada

Test	**Expected Value**		**Actual Value**	
	Mean	*Std. Dev*	*Mean*	*Std. Dev*
Table 1: Column 1	550.6	572.03		
Table 1: Column 2	60.32	62.26		

Figura 5: Valores esperados para cada columna de entrada

6. Resultado

Tras usar la aplicación web con los casos de prueba propuestos anteriormente se consiguieron los resultados mostrados en la figura 6 y en la figura 7, que cómo se puede observar corresponden con los valores esperados mencionados en la tabla de la figura 4.

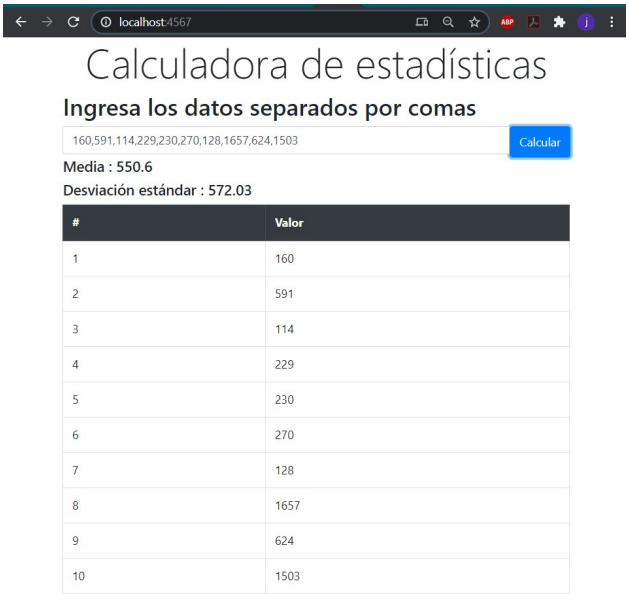

Figura 6: Resultados Columna 1

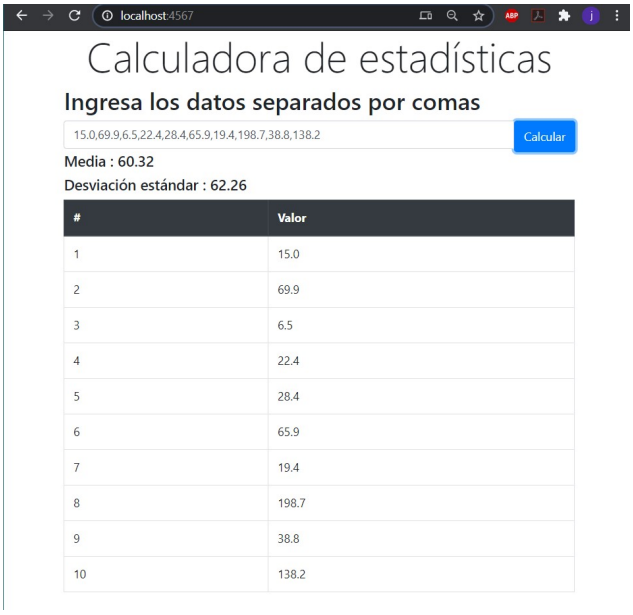

Figura 7: Resultados Columna 2

7. Referencias

[1] InterviewBit, *Linked-List*, https://www.interviewbit.com/courses/programming/topics/linked-lists/#:~:text=A%20linked%20list%20is%20a,has%20a%20reference%20to%20null., Accessed on 2020-08-07.

[2] A. Ahlawat-Studytonight, *Doubly Linked List*, https://www.studytonight.com/data-structures/doubly-linked-list#:~:text=Doubly%20linked%20list%20is%20a,next%20node%20in%20the%20list., Accessed on 2020-08-07.

[3] http://campusvirtual.escuelaing.edu.co/moodle/pluginfile.php/181210/mod_resource/content/0/EnunciadoTallerEjercicioMVNGit.pdf, Accessed on 2020-08-07.