Universidad de las Américas Puebla

Proyecto final – Laboratorio de manejadores de bases de datos
Carlos Andrés Reyes Evangelista
157068

Big Data Management and Processing

Introducción

La intención del presente reporte es presentar una sinopsis de los conocimientos adquiridos durante la realización de las prácticas del Laboratorio de Manejadores de Bases de Datos. Estos conocimientos refieren y están relacionados con diversos paradigmas que atañen la administración y almacenamiento de datos, mas no es ésa la única característica que comparten: todos están relacionados con el renombrado concepto de Big Data.

La noción de Big Data no abarca únicamente, como especifican Adiba, Castrejón, Vargas, Zechinelli y Espinosa (2016), "un diluvio de datos para procesar y administrar grandes volúmenes de bytes. Sino que además de esta visión superficial existe un consenso sobre las tres V que caracterizan a Big Data: Volumen, Variedad y Velocidad", lo que implica que la percepción sobre Big Data es más profunda que lo que un terrenal pensamiento normalmente englobaría. El doctor Zechinelli (2019) mencionó en clase que Big Data "es una colección de datos tan masiva que es difícil o imposible (impráctica) de manejar con una base de datos tradicional", pero que además es caracterizada por las tres V referenciadas previamente, éstas son definidas como sigue: Volumen se refiere a la cantidad de datos, pero "no sólo a los datos estáticos, sino a los datos en flujo" (Zechinelli, 2019); Variedad trata sobre los "diferentes tipos de representaciones que los datos pueden adoptar" (Adiba, Castrejón, Vargas, Zechinelli & Espinosa, 2016); y finalmente, Velocidad hace mención a la rapidez con que los datos son producidos y consumidos.

En principio, las tres tecnologías que fueron estudiadas y puestas en práctica están relacionadas con Big Data en alguna forma, sin embargo, cada una de ellas aborda un enfoque distinto e independiente entre ellas. A continuación, es presentada una breve descripción de cada una de ellas y posteriormente se detallará el desarrollo de cada una de las prácticas realizadas para cada una de estas herramientas respectivamente.

MongoDB es un sistema de gestión de bases de datos NoSQL orientado a documentos. De acuerdo a Rick Cattell (2010), NoSQL significa «No sólo SQL» o «No Relacional», además un sistema NoSQL se caracteriza por ciertos puntos clave entre los cuales se incluye: la habilidad de escalar horizontalmente, capacidades de replicación y distribución, modelos de concurrencia más flexibles que ACID y la habilidad de dinámicamente añadir nuevos atributos a los registros de datos. MongoDB destaca en los primeros dos de ellos: integra un concepto conocido como Balanceo de Carga mediante el cual efectúa escalamiento horizontal al aplicar una técnica conocida como fragmentación (*sharding*). "MongoDB soporta fragmentación automática, distribuyendo documentos a través de los servidores" afirma Rick Cattell (2010), a su vez Vargas-Solar realza la importancia del *sharding* al aseverar que "sharding permite a los sistemas de bases de datos paralelos aprovecha

la banda ancha de entrada/salida de múltiples discos al leer y escribir simultáneamente sobre ellos", de esta manera, MongoDB balancea la carga y replica los datos para mantenerse resiliente.

Apache Pig es una herramienta de programación originalmente utilizada para implementar programas de tipo Map-Reduce a ser ejecutados por Hadoop para solucionar problemas en paralelo sobre grandes colecciones de datos que puedan resolverse mediante la aplicación de las funciones provenientes del paradigma funcional Map y Reduce, pero adaptadas para funcionar en ambientes paralelos o distribuidos. Vargas-Solar (2017) habla de los notables beneficios que Map-Reduce ofrece: "Ambientes en tiempo de ejecución basados en Map-Reduce proveen un buen rendimiento en arquitecturas sobre todo en pruebas analíticas que operan sobre vastas colecciones de datos". Además de la ya mencionada, entre algunas de las características que hacen destacar a Pig se encuentra su lenguaje integrado conocido como *Pig Latin* que permite al usuario concentrarse en la semántica en lugar de la eficiencia debido a que, según la página oficial de Pig ("Welcome to Apache Pig", 2018): "es trivial conseguir simples pruebas de análisis de datos en paralelo. Las pruebas complejas, compuestas de múltiples transformaciones de datos interrelacionadas, son explícitamente codificadas como secuencias de flujo de datos, haciéndolas fácil de escribir, entender y mantener". *Pig Latin* también ofrece al usuario la capacidad de crear sus propias funciones para procesamiento específico.

GraphX es una herramienta de alto nivel empleada por la plataforma Apache Spark, especializada en computación en clúster sobre el que se pueden realizar tanto algoritmos iterativos como análisis de datos interactivos o exploratorios. GraphX es lo que se conoce como un método de acceso de tipo Grafo. Hellerstein, Stonebraker y Hamilton (2007) definen un método de acceso como "una colección de algoritmos y estructuras para organizar y acceder a los datos", mientras que un método de acceso de tipo grafo, aunque no es propiamente definido, es descrito por Athanassoulis e Idreos (2016) como un tipo de método de acceso que se mimetiza con el sistema propio debido a dos razones: "la primera, la representación en forma de grafo juega un rol importante en el rendimiento y cada representación puede requerir una muy diferente manera de acceder a los datos; en segundo lugar, ha habido un mucho menor esfuerzo en la estandarización comparado contra los sistemas relacionales, por lo tanto se permiten soluciones más nativas que no se adhieren a un estándar global". GraphX hace uso también de los llamados RDD o Conjunto de datos Resiliente Distribuido (*Resilient Distributed Dataset*) que forman parte de la arquitectura de Apache Spark; este tipo especial de conjuntos "representan una colección de objetos de sólo-lectura particionados a través de un conjunto de máquinas y que pueden ser reconstruidos si una partición es perdida" (Zaharia, Chowdhury, Franklin, Shenker & Stoica, 2010), de ahí que su nombre incluya el adjetivo *resiliente*, que hace referencia a la capacidad humana de recuperarse ante situaciones adversas.

Práctica 1 – MongoDB

Objetivo

Esta práctica mantenía dos objetivos principales, derivado de la división misma del trabajo. En primer lugar, fue pretendido ilustrar el funcionamiento básico de MongoDB como SGBD al hacer uso de sus comandos sobre las terminales cliente y servidor que Mongo ofrece. La segunda parte del trabajo fue orientada a la ilustración del concepto de *sharding* sobre una base de datos No Relacional mediante diferentes técnicas.

Pasos principales

La primera parte del trabajo, referente a la utilización de comandos básicos de MongoDB para manipular la base de datos no representó un gran problema pues, fuera de la sintaxis, el proceso fue bastante directo: crear y utilizar una base de datos utilizando `use database_name`, crear una colección con `db.createCollection(name)`, añadir elementos a la colección utilizando `db.collection.insert([]),` filtrar o encontrar elementos haciendo uso de `db.collection.find()` y efectuar actualizaciones con `db.collection.update()`.

La segunda parte de la práctica requirió herramientas más específicas para la obtención del resultado, concretamente fue necesario programar un servidor de configuración (*config server*), un enrutador de consultas (*query router*) y un par de fragmentos (*shards*). Para empezar, fue necesario crear y llenar una base de datos sobre la que fuese posible trabajar, posteriormente, fue creado el servidor de configuración ejecutando una instancia del comando `mongod` con la ruta de la base de datos y un puerto en específico; una vez hecho esto, fue iniciado un enrutador de consultas mediante el uso del comando `mongos` indicando el puerto de la instancia del servidor de configuración y asignándole un nuevo propio puerto independiente.

Finalmente, los fragmentos requeridos fueron registrados de acuerdo a las especificaciones de la práctica. En la primera mitad fue creado un solo fragmento (*shard*) para ilustrar el funcionamiento básico. En la segunda mitad, fueron utilizadas distintas metodologías que distribuyen la carga a partir de cierto criterio: por medio de rangos, como su nombre lo indica, la fragmentación se realiza asignando un fragmento específico con base en su pertenencia a un rango, en este caso, alfabético; por medio de valores hash, donde cada registro es asignado a un fragmento dependiendo de su valor resultante después de aplicársele una función hash administrada por el enrutador de consultas; finalmente, por medio de etiquetas, donde a cada fragmento se le asigna una etiqueta que indica que los registros que satisfagan ésta deben pertenecer ahí, así se asegura mantener datos relacionados aislados y agrupados en una localización deseada.

Conclusiones

Después del tiempo dedicado a comprender este concepto, resultan evidentes las ventajas que fragmentar una colección masiva de datos puede ofrecer. Balancear la carga permite no sólo ser más tolerante a fallas, sino que otorga la posibilidad de paralelizar y sacar provecho de la distribución inherente de los datos. La presente práctica otorgó la experiencia necesaria para afianzar los conocimientos adquiridos a lo largo del curso respecto a las nociones de *sharding* y sistemas NoSQL, por lo tanto, se concluye que la misma ha cumplido con las expectativas planteadas.

Práctica 2 – Apache Pig

Objetivo

La meta por conseguir en este proyecto era entrar en contacto con *Pig Latin* (véase Introducción) para analizar colecciones masivas de datos a través de consultas escritas en ese lenguaje. Para este efecto, fue brindada una colección de datos previamente construida que almacena pruebas de conexión de red realizadas por el *Proyecto Neubot*.

Pasos principales

Por supuesto, el primer paso involucró cargar la colección de datos desde el almacenamiento de Pig, acto seguido, fueron realizadas una serie de consultas que involucraban el filtrado de información de acuerdo a ciertos criterios para lo cual fueron utilizados los siguientes operadores:

- `filter-by`, esta función juega el papel que jugaría el operador selección (`where`) en un modelo relacional, pues dada una relación retorna únicamente sus elementos cuyo valor de verdad al aplicar la condición establecida después de la palabra clave `by` sea verdadero;
- `foreach-generate`, es una función semánticamente equivalente a la proyección (`select`) en SGBDR, su función literal es: para cada tupla en la relación indicada generar la columna, columnas o funciones de agregación pertinentes;
- `distinct`, que hace exactamente lo que su nombre indica: de una colección remueve todos los elementos repetidos para mantener exactamente un elemento de cada valor original;
- `group-by`, que funciona bajo el mismo precepto que su homónimo en modelos relacionales y, como su nombre sugiere, dada una relación y un atributo de ésta, se es generada una nueva relación que contiene diversos conjuntos donde cada uno contiene a las tuplas de valor afín en términos del atributo precisado;
- y `order-by`, que ordena los valores alfanuméricos de una colección de acuerdo a un atributo especificado.

También fueron utilizadas ciertas funciones programadas previamente por el Proyecto Neubot: `IpToNumber()` y `NumberToIp()` que, como su nombre sugiere, transforma valores numéricos en valores interpretables como direcciones IP y viceversa.

Conclusiones

A pesar de que la utilización de los operadores varía levemente en comparación con sus equivalentes en un ambiente SQL, una vez que su utilización es dominada, resulta relativamente sencillo y evidente formular consultas basadas únicamente en la semántica requerida, tal y como se sostiene en la descripción de Pig Latin, ergo, el objetivo de esta práctica también ha sido alcanzado satisfactoriamente.

Práctica 3 – GraphX

Objetivo

La finalidad de esta última práctica era entrar en contacto con el cómputo en paralelo y conocer los retos con que se debe de lidiar cuando se definen y se procesan los datos en forma de grafo mediante la utilización del API de GraphX sobre el cuadro de trabajo de Spark.

Pasos principales

El primer ejemplo sobre el que se laboró fue un grafo creado para representar una ínfima red social en donde cada vértice (nodo) representaba a un usuario del que almacenaba información personal y cada borde (arista) una conexión entre ellos que almacenaba la cantidad de *likes* que habían otorgado al otro. Para su creación primero se definen en arreglos la información que cada vértice y cada borde almacenará, respectivamente, luego, cada uno de estos arreglos es almacenado en un RDD (*véase Introducción*), finalmente, a partir de esos dos RDD es definido un grafo y sobre él se puede procesar de manera iterativa sobre vértices/bordes u obteniendo los llamados tripletes o triadas (triplets) que representan las relaciones vértice-borde-vértice existentes en el grafo y sobre ellos se pueden ejecutar operadores como count, map, filter, o reduceByKey.

Posteriormente, fue creado un grafo adicional a partir de arreglos previamente definidos que almacenaban información sobre artículos de Wikipedia que contuvieran la palabra “Berkeley”, el proceso de creación varió ligeramente con respecto del método aplicado el párrafo anterior: para crear los RDD de vértices fue necesario mapearlos en pares que contuviesen un identificador de vértice y un título de artículo y colocar a los bordes un marcador de posición con el valor 0. Al construir el grafo fueron utilizados los RDD como en el párrafo anterior, pero además fue necesario incluir un valor por defecto para vértices inexistentes, en este caso particular, para cubrir los casos donde existen enlaces rotos.

Conclusiones

La representación utilizada para acceder y procesar los datos que tuvo que ser usada en esta práctica rompe los esquemas tradicionales a los cuales habíamos estado acostumbrados ya que ofrece una representación de los datos y un punto y forma de acceso distinto. La realización de este tipo de prácticas abre los ojos a posibilidades poco ortodoxas que, al modificar la representación de los datos, abren la viabilidad a la resolución de específicos, en este caso, por ejemplo, cuando incluso la interconexión de los elementos amplia la semántica de la información. Por consiguiente, es considerado que el objetivo ha sido cumplido con éxito.

Discusión general sobre lo aprendido en las prácticas

La civilización evoluciona progresiva y súbitamente y con ella lo hace su necesidad de información, así que los sistemas que ofrecen servicios de esta índole se ven obligados a evolucionar a la par, como resultado los arquetipos que los sistemas de gestión de datos pueden seguir han visto incrementado su espectro en amplia variedad en los últimos años; este incremento en los paradigmas ha generado nuevos tipos de sistemas con características diversas que cubren y proponen soluciones a problemáticas diversas. A lo largo de la impartición del curso hemos sido testigos y hemos formado parte del hecho de que entre más conoces del mundo, más amplio se vuelve, en concreto: lo que otrora se limitaría a Sistemas de Gestión de Bases de Datos Relacionales ahora ha sido extendido a Sistemas NoSQL, Distribuidos, Paralelizados, Depósitos de Datos, sistemas con distintos Métodos de acceso y la lista continúa.

Específicamente, como alumno, he aprendido que los Sistemas de Gestión de Bases de Datos Relacionales, aunque parten de una base muy sólida y bien estructurada, no son panacea y cómo el no cerrarse a las posibilidades o extensiones posibles puede generar paradigmas distintos que, aunque ofrezcan ciertas limitaciones y debilidades frente a los SGBDR, pueden ofrecer alternativas que solucionen mejor cierta clase de problemas, en referencia, por supuesto, a los Sistemas **NoSQL** de los que ahora entiendo que, pese a no seguir una serie de lineamientos tan estricta como Relacional en el sentido de que no cumplen con “formas normales” o propiedades ACID, sus ventajas se hacen claras al cambiar el lente de perspectiva, pues al sacrificar un poco de coherencia y modelo de concurrencia, es posible obtener beneficios formidables como la replicación, distribución, escalamiento, flexibilidad y eficiente uso de índices.

Además, he comprendido el rol que juega la arquitectura **Map-Reduce** en pruebas de procesamiento intensivo sobre extensas colecciones de datos para paralelizar la ejecución. Tenía ya conocimiento previo sobre las funciones *map* y *reduce* del paradigma funcional, mas ahora soy capaz de comprender las adaptaciones que se realizaron sobre estas funciones para mantener su esencia, pero adecuarlas a un ambiente de trabajo distribuido y paralelo.

También adquirí conocimientos sobre los diversos **métodos de acceso** a datos existentes y las ventajas y limitaciones que cada uno de estos ofrecen; gracias a la práctica 3, los métodos de acceso a través de grafos son los únicos con los que he tenido real contacto hasta ahora, sin embargo, el concepto me queda claro y conozco la bibliografía existente para consultar cuando sea preciso.

No obstante, el conocimiento que considero más valioso y que obtuve a lo largo de este curso es la asimilación del concepto de **Big Data**, pues encuentro en esta noción la columna vertebral de lo que el futuro apremia para esta rama de la informática de la mano por supuesto de la nueva denominada **ciencia de datos**. Entendí que a estos niveles de representación de información la velocidad es más importante que la veracidad, que cuando se trata de fuentes masivas de datos ya no se busca por listas de respuestas correctas sino por conjuntos de resultados lo suficientemente buenos para permitir su exploración. Concluí también que la ciencia de datos es una técnica no-determinística que involucra estadística, matemáticas, técnicas de *hackeo* y

habilidades de programación debido a la ausencia de estándares para almacenar, analizar u organizar cualquier tipo de datos.

En conclusión, existe una amplia gama de técnicas para almacenar, procesar y analizar la información y nadie puede saberlo todo al respecto, pero considero que lo realmente importante para un ingeniero es conocer todas ellas al menos lo suficiente como para, después de un análisis del problema específico a resolver, poseer los conocimientos necesarios para discernir y elegir con fundamentos sólidos el arquetipo más adecuado para esa situación concreta.

Bibliografía

[1] Adiba, M., Castrejón, J., Vargas, G., Zechinelli, J., & Espinosa, J. (2016). *Big Data Management: Challenges, Approaches, Tools and their limitations* [Ebook].

[2] Catell, R. (2010). *Scalable SQL and NoSQL Data Stores* [Ebook].

[3] Vargas-Solar, G. (2017). *Efficient data management for putting forward data centric sciences* [Ebook].

[4] Kerstern, M., Idreos, S., Manegold, S., & Liarou, E. (2011). *The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds* [Ebook].

[5] Athanassoulis, M., & Idreos, S. (2016). *Design Tradeoffs of Data Access Methods* [Ebook].

[6] Idreos, S., Athanassoulis, M., Guo, D., Kester, M., Maas, L., & Zoumpatianos, K. (2016). *Past and Future Steps for Adaptive Storage Data Systems: From Shallow to Deep Adaptivity* [Ebook].

[7] J. M. Hellerstein, M. Stonebraker, and J. R. Hamilton. Architecture of a Database System. Found. Trends Databases, 1(2):141–259, 2007.

[8] Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., & Stoica, I. (2010). *Spark: Cluster Computing with Working Sets* [Ebook]. Retrieved from https://amplab.cs.berkeley.edu/wp-content/uploads/2011/06/Spark-Cluster-Computing-with-Working-Sets.pdf

[9] Welcome to Apache Pig!. (2018). Retrieved from https://pig.apache.org