

Procesamiento paralelo de datos medioambientales con Apache Spark

Diego Ferrón, Sebastián Villarroya, José R.R. Viqueira, and Tomás F. Pena

Centro de Investigación en Tecnoloxías da Información (CITIUS)
Universidade de Santiago de Compostela (USC)
Santiago de Compostela, Spain
diego.ferron@rai.usc.es, {sebastian.villarroya, jrr.viqueira,
tf.pena}@usc.es

Resumen En este artículo se proporciona un descripción breve de los primeros pasos hacia la implementación de un sistema de procesamiento analítico en línea paralelo que integre datos de entidades espaciales y grandes arrays resultado de procesos de muestreo temporal, espacial y espacio-temporal. Estos tipos de dato son fundamentales en aplicaciones de apoyo a la toma de decisiones en entornos medioambientales.

Keywords: Big Data, Environmental Data, OLAP, Spark

1. Introducción

La importancia que tiene la incorporación de datos medioambientales, de naturaleza geo-espacial, en cada vez más ámbitos de toma de decisión es un hecho contrastado. La cantidad y precisión de los datos generados es cada vez mayor, debido a los grandes avances en la fabricación de dispositivos de sensorización. Así, por ejemplo, el análisis de los factores que influyen en la aparición de incendios forestales se beneficia de la disponibilidad de los siguientes conjuntos de datos (entre otros): i) Datos meteorológicos generados por redes de estaciones públicas y privadas. ii) Datos de elevación generados por sensores Lidar. iii) Datos de cobertura vegetal del terreno (modelos de combustible). Para hacerse una idea de la cantidad de datos generados, MeteoGalicia dispone de una red de unas 80 estaciones meteorológicas que generan datos con una frecuencia temporal de 10 minutos, mientras que la resolución espacial máxima de los datos de elevación proporcionados por el Instituto Geográfico Nacional es de 5 metros, con lo que se necesitan unos 2.500 millones de valores de elevación para cubrir el territorio de Galicia.

El reto es por lo tanto el procesamiento analítico en línea (OLAP) de grandes cantidades de datos que incluyen entidades espaciales (como las estaciones meteorológicas) y grandes arrays de valores numéricos resultantes de procesos de muestreo temporales (como los generados por las estaciones) y posiblemente también espaciales (como los generados por el Lidar). Así por ejemplo, el análisis de riesgo de aparición de incendios necesitaría: i) Calcular la pendiente del

terreno a partir de su elevación (operación de ventana espacial sobre los datos de elevación), ii) Interpolación espacialmente los datos meteorológicos de las estaciones para obtener una cobertura espacial completa (join espacial entre el muestreo espacial de elevación y las observaciones de las estaciones), iii) Combinación de la pendiente, datos meteorológicos y modelos de combustible para generar el índice de riesgo (join espacial).

El procesamiento de datos medioambientales en entornos científicos se realiza en general mediante implementaciones ad-hoc sobre formatos de archivo estandarizados, a veces utilizando librerías de geo-procesamiento disponibles en el área de los Sistemas de Información Geográfica. Si nos restringimos al procesamiento de datos de entidades espaciales, existen extensiones espaciales de los Sistemas Gestores de Bases de Datos más utilizados desde hace ya más de una década. En el rango del Big Data, existen también extensiones espaciales y geográficas para entornos de procesamiento bien conocidos, como Apache Hadoop [3] y Apache Spark [5]. Finalmente, existen también soluciones para el procesamiento de grandes arrays multidimensionales [1,2,6]. Sin embargo, no existe ninguna implementación eficiente de OLAP para datos medioambientales de todo tipo, diseñada para integrar entidades espaciales y grandes arrays resultantes de muestreos espacio-temporales.

En este artículo se ilustran los retos que surgen al abordar la implementación del lenguaje MAPAL (Mapping Analysis Language) [4] sobre una arquitectura de procesamiento paralelo de altas prestaciones basada en Apache Spark. Se discuten varias alternativas para el almacenamiento de datos y su influencia en la implementación de operaciones básicas de procesamiento.

2. Solución propuesta

2.1. MAPAL: Lenguaje de análisis de funciones de datos

El sistema de tipos de dato de MAPAL [4] incluye tipos de dato convencionales, incluyendo numéricos de precisión fija y variable, tipos de dato temporales y espaciales (1D y 2D) de precisión fija y tipos de dato geométricos. Así, por ejemplo, el tipo de dato *TimeInstant*(R) representa instantes de tiempo con una resolución temporal de R segundos (R es un número real). El tipo *Point2D*(P,R) representa puntos de un espacio discreto de dimensión 2, donde cada punto se interpreta como un cuadrado (pixel) de tamaño real $R \times R$. Los tipos geométricos para representar líneas, polígonos y geometrías complejas se definen sobre la base del espacio definido por el tipo *Point2D*(P,R).

Los conjuntos de entidades espaciales y arrays se representan de forma uniforme mediante *Dimensiones* y *Cubos de Datos*. Una *Dimensión* es un subconjunto finito de los elementos de un tipo de dato (no geométrico). Casos especiales de *Dimensiones* son los *Muestreos 1D* (temporales y espaciales) y *2D* (espaciales). Un muestreo contiene una secuencia ordenada de elementos de un tipo de dato (temporal o espacial), desde un elemento inicial a un elemento final. Si las estaciones meteorológicas generan una serie de datos agregados de temperatura con

una frecuencia diaria, la *Dimensión* temporal de dicha serie se definiría como un *Muestreo 1D Temporal* sobre el tipo de datos *TimeInstant*(86400), desde una fecha inicial D_{inicio} a una fecha final D_{fin} . De forma similar, la *Dimensión* espacial para los datos de elevación se podría definir como un *Muestreo 2D* sobre el tipo de datos *Point2D*(7,5), desde un punto inicial (esquina inferior izquierda) P_{inicio} a un punto final (esquina superior derecha) P_{fin} .

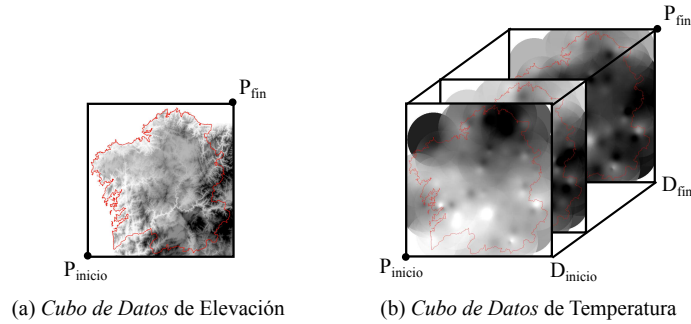


Figura 1. Ilustración de *Cubos de Datos* espaciales y espacio-temporales.

Los *Cubos de Datos* (*MappingSets* en [4]) son estructuras n-dimensionales que permiten la representación de conjuntos de funciones, cuyos dominios están definidos por Productos Cartesianos de *Dimensiones* y cuyos rangos están definidos por subconjuntos de los tipos de dato del sistema. Por ejemplo, la elevación en cada punto se representa mediante un *Cubo de Datos* con una función *elevación* definida sobre el *Muestreo 2D* descrito arriba (ver Figura 1(a)). De forma similar, el resultado de la interpolación espacial de los datos generados por las estaciones se representa mediante un *Cubo de Datos* con tres funciones, *temperatura*, *humedad* y *viento*, de tipo numérico de precisión fija, definidas sobre el Producto Cartesiano de los muestreos temporal y espacial descritos arriba (ver Figura 1(b)).

2.2. Implementación sobre Apache Spark

En general tanto las *Dimensiones* como los *Cubos de Datos* se almacenan mediante *DataFrames*, persistidos mediante el formato columnar Apache Parquet. Tres alternativas distintas se están evaluando en la implementación en marcha.

Alternativa 1 Tanto las *Dimensiones* como las funciones se almacenan en columnas de un *DataFrame*.

Alternativa 2 Sólo las funciones se persisten en los *DataFrames*, ordenadas de forma ascendente por los valores de las *Dimensiones*.

Alternativa 3 Además de las funciones, se almacena un identificador autoincremental siguiendo el orden de las *Dimensiones*.

En cuanto al procesamiento, las operaciones entre *Dimensiones* y *Cubos de Datos* se implementan mediante operaciones de Apache Spark, que incluyen operadores relacionales sobre *DataFrames*. Una operación importante es el *equijoin* entre dos *Cubos de Datos* a través de *Dimensiones* comunes. Por ejemplo, el *equijoin* entre los *Cubos de Datos* de las Figuras 1(a) y (b) a través de su *Dimensión* espacial común. Dependiendo de la alternativa de almacenamiento utilizada para los *Cubos*, el algoritmo de este *equijoin* tendrá que leer más bloques de datos y realizar menos procesamiento (Alternativa 1) o leer menos bloques de datos y realizar más procesamiento (Alternativa 2). La Alternativa 3 es un compromiso entre las dos anteriores que facilita la generación de identificadores de *Dimensión* en un entorno de procesamiento paralelo.

Otro tipo importante de operación es el join espacial entre *Cubos de Datos*. En este caso, será fundamental el uso de una estrategia de particionamiento espacial de los datos para minimizar el tráfico entre los nodos.

3. Trabajo futuro

El trabajo futuro a más corto plazo consistirá en la finalización de la implementación en curso y su evaluación, incluyendo la evaluación del impacto en el rendimiento del uso de las tres alternativas de almacenamiento descritas y su combinación con distintas técnicas de codificación y compresión de datos. Posteriormente, se explorará el uso de distintas técnicas de particionamiento espacial que puedan ser utilizadas por nuevos algoritmos para los operadores de selección y join.

Referencias

1. Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., Widmann, N.: The multidimensional database system rasdaman. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. pp. 575–577. SIGMOD '98, ACM, New York, NY, USA (1998)
2. Brown, P.G.: Overview of scidb: Large scale array storage, processing and analysis. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. pp. 963–968. SIGMOD '10, ACM, New York, NY, USA (2010)
3. Eldawy, A., Mokbel, M.F.: Spatialhadoop: A mapreduce framework for spatial data. In: 2015 IEEE 31st International Conference on Data Engineering. pp. 1352–1363 (April 2015)
4. Villarroya, S., Viqueira, J.R.R., Regueiro, M.A., Taboada, J.A., Cotos, J.M.: Soda: A framework for spatial observation data analysis. Distributed and Parallel Databases 34(1), 65–99 (2014)
5. Yu, J., Wu, J., Sarwat, M.: Geospark: A cluster computing framework for processing large-scale spatial data. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 70:1–70:4. GIS '15, ACM, New York, NY, USA (2015)
6. Zhang, Y., Kersten, M., Manegold, S.: Sciql: Array data processing inside an rdbms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. pp. 1049–1052. SIGMOD '13, ACM, New York, NY, USA (2013)