Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Tratamiento y visualización eficiente de datos multidimensionales.

Anteproyecto de título presentado por Nicolás Vásquez Eyzaguirre y Nicolás Adasme Arroyo
de la Carrera Ingeniería Civil Informática
Dirigida por Clemente Rubio Manzano y Pablo Gonzalez

2022

Resumen

El propósito de realizar este trabajo investigativo es cumplir con los requerimientos curriculares de la carrera Ingeniería Civil Informática. Nuestro trabajo investigativo tiene por nombre "Tratamiento y visualización eficiente de datos multidimensionales.". En este proyecto investigativo buscaremos conocer las bases y fundamentos de los archivos NetCDF, cómo estos funcionan y cómo ha sido su avance a través del tiempo. También nos embarcaremos en la tarea de analizar el rendimiento de estos ficheros en los principales lenguajes de programación y motores de bases de datos, además de abarcar otras áreas no tan conocidas de los archivos NetCDF, todo lo anterior con el objetivo de poder apreciar las diferentes alternativas de tratamiento y visualización de archivos NetCDF para determinar de forma empírica, cuál o cuáles son las alternativas más destacadas para ello.

Índice general

Índice de Figuras

Índice de Tablas

Estructura y composición del informe.

El presente trabajo se encuentra dividido en cuatro capítulos. A continuación se describe brevemente el contenido de cada uno de ellos.

(a) **Introducción:** En este capítulo daremos contexto a que son los archivos NetCDF, una breve descripción sobre los datos espacio temporales y su importancia en la comunidad científica de diversas áreas del conocimiento. Introduciremos acerca de la importancia de estos datos y de la relevancia en conocer y elegir correctamente las tecnologías adecuadas para manipular estos datos. Definiremos los objetivos específicos y general, así como también la metodología de trabajo.

(b) **Marco Teórico:** El presente capítulo tiene como objetivo describir algunos de los conceptos más importantes relacionados a las tecnologías que utilizaremos y sobre los aspectos teóricos relacionados con los archivos NetCDF, consiguiendo así una mejor compresión del tema en posteriores capítulos. Para una mejor comprensión sobre los archivos NetCDF lo dividimos en diversas subsecciones abarcando cada aspecto relevante de dichos archivos los cuales son Sistemas de información geográfica, Datos geográficos, Esquemas preconceptuales, Caracterización de formatos de almacenamiento, CDF, HDF y NetCDF, Análisis de la caracterización realizada, Ficheros NetCDF, Propósito de los archivos NetCDF, NetCDF (Network Common Data Form), Caracterización de los NetCDF, Interfaz de archivos NetCDF, Rendimiento de NetCDF, Estructura de formatos CDF-1, 2 y 5, Estructura de formatos NetCDF-4 HDF5 y Modelo de datos.

(c) **Estado del Arte:** Este capítulo tiene como finalidad abarcar en profundidad el medio y de qué forma se desenvuelven los archivos NetCDF en diversas áreas de la tecnología, para lograr aquello, consideramos cuatro categorías. Las distintas librerías de lenguajes de programación, bases de datos, aplicaciones comerciales e investigación.

(d) **Caso de estudio:** En el presente capitulo abarcaremos la problemática hídrica que afecta a Chile y luego ahondaremos a la crisis hídrica concretamente a la región de Ñuble como objeto de pruebas para el uso de los archivos NetCDF en el que realizaremos una carga de archivos NetCDF en un determinado lenguaje el cual previamente será evaluado ante otras tecnologías con el fin de obtener una comparativa entre que tecnología es la más adecuada para manipular este tipo de archivo.

(e) **Conclusiones:** Realizaremos una comparativa sobre la propuesta que tuvimos al inicio de la investigación, sobre lo que realmente se llevó a cabo y lo que queda por finalizar para eventuales trabajos futuros.

Capítulo 1

Introducción

Los datos-espacio temporales son una parte fundamental en el estudio de muchas disciplinas y principalmente en la investigación orientada a la meteorología, ciencia atmosférica interdisciplinaria que hoy por hoy se encuentra con una problemática importante, como es el cambio climático y sus consecuencias asociadas. Tecnologías o herramientas que nos permitan un tratamiento en estructuras multidimensionales se hacen sumamente necesarias en la actualidad ya que lograr trabajar con un alto grado de tolerancia al intercambio y manejar datos de una gran envergadura no es una tarea sencilla. Es por lo anterior que entran en juego los NetCDF (formulario de datos comunes en red) que es un formato de archivo que tiene como objetivo almacenar datos científicos multidimensionales, cómo la temperatura, humedad, presión, etc. Muchas organizaciones y grupos científicos han adoptado NetCDF como un estándar, ya que al ser los NetCDF un formato abierto de intercambio de datos científicos multidimensionales hace sumamente sencillo y claro el trabajo con este tipo de información. Los NetCDF pertenecen a UCAR (University Corporation for Atmospheric Research), que por medio de su programa comunitario con diversos grupos de instituciones educativas y de investigación (Unidata) y que es financiado por la Fundación Nacional de Ciencias (NSF), dieron forma a esta tecnología. Dada la importancia que tienen los NetCDF en la actualidad se hace sumamente importante disponer de la información necesaria para poder entender su funcionamiento, además de comprender como poder realizar un tratamiento adecuado de estos ficheros, ya que si bien la tecnología facilita muchas tareas como almacenar los datos en forma de array de una manera clara y sencilla, además de que su tratamiento sea eficaz, este no está exento de problemáticas de desarrollo, es por ello que nuestro enfoque estará dirigido a analizar el rendimiento y entender como los NetCDF funcionan en los principales lenguajes de programación y motores de bases de datos ya que esto nos permitirá lograr un tratamiento y visualización eficiente de los mismos.

1.1. Objetivos

1.1.1. Objetivo General

Análisis y estudio de datos multidimensionales para la comparacion de rendimiento de las diferentes opciones de tratamiento y visualizacion eficiente de archivos NetCDF con la finalidad de determinar las mejores alternativas para ello.

1.1.2. Objetivos Específicos

- Revisión bibliográfica de los conceptos asociados a los archivos multidimensionales: Caracterización de los archivos multidimensaionales, su procesamiento y su visualización.
- Comparar los distintos ficheros de almacenamiento de datos cientificos multidimensionales y sus características.
- Analizar las diversas tecnologías que permiten el funcionamiento de los archivos NetCDF
- Analizar el manejo de los NetCDF en paralelo, escalable, bases de datos y aplicaciones comerciales.
- Comparar las distintas librerias NetCDF que soportan el tratamiento de archivos multidimensionales.

1.2. Metodología de trabajo

- **Fase 1: Análisis conceptual de datos multidimensionales.** Se realizará un estudio focalizando el procesamiento y características que poseen los archivos multidimensionales.
- **Fase 2: Análisis de archivos NetCDF.** Se realizará un estudio de las tecnologías asociadas a NetCDF (Network Common Data Form), su funcionamiento, capacidad de procesamiento, escalabilidad y visualización.
- **Fase 3: Análisis de las librerías NetCDF asociadas a diversos lenguajes de programación.** Se realizará un estudio de las librerías NetCDF de diversos lenguajes de programación comúnmente utilizados en la comunidad científica.
- **Fase 4: Análisis de en profundidad de NetCDF.** Se realizará un estudio de los archivos NetCDF y su participación en las bases de datos, aplicaciones comerciales, NetCDF escalable y NetCDF en paralelo.
- **Fase 5: Comparar diversas tecnologías y su rendimiento en la aplicación de archivos de datos multidimensionales.** Se realizará una comparativa de diversos lenguajes de programación y su rendimiento en el procesamiento y lectura de archivos NetCDF.

Además, al final del informe se adjuntan las referencias con los artículos utilizados en el proceso de investigación.

Capítulo 2

Marco Teórico

El presente capítulo tiene como objetivo describir algunos de los conceptos más importantes relacionados a las tecnologías que utilizaremos y sobre los aspectos teoricos relacionados con los archivos NetCDF, consiguiendo así una mejor compresión del tema en posteriores capítulos.

2.1. NetCDF

2.1.1. Sistemas De Información Geográfica.

Las características de la información geográfica es que esta posee, una serie de componentes, entre estas podemos encontrar la espacial y en repetidas ocasiones una componente temporal. Durante los últimos años el uso de esta información ha incrementado enormemente debido a la expansión de las tecnologías que la captura. Específicamente, las ciencias relacionadas a la tierra, el océano y la atmosfera requieren de un análisis de la información que varía en función del tiempo y del espacio, para la toma de decisiones. Estos datos espacio temporales son gestionados por los denominados SIG, que son los Sistemas de información Geográfica, que definen su formato para su almacenamiento, transporte y visualización. Una de las principales organizaciones que definen los estándares para estos formatos es la Open Geospacial Consortium, sin embargo, existen otros también, que son ampliamente usados. Es de suma importancia conocer los distintos formatos disponibles, los elementos que lo componen, los ambientes y situaciones para la cual fueron creados, para así decidir cual formato conviene utilizar en favor de la eficiencia y optimización de su uso en los SIG. A continuación, realizaremos una caracterización de los formatos más relevantes para el almacenamiento, transporte y visualización de datos geográficos: vector, raster y series de datos, mediante esquemas preconceptuales que permiten identificar las relaciones estructurales y dinámicas de cualquier dominio del conocimiento.

Algunos ejemplos de mecanismos para la obtención de datos geográficos son los sistemas digitales de adquisición de datos, los sistemas de posicionamiento global, los sensores remotos y la simulación computacional son algunos ejemplos de ello. La definición de datos geográficos corresponde a información que varia en el espacio terrestre, esto quiere decir, que tienen asociada una coordenada geográfica que define el lugar que el dato representa en términos de elevación,

temperatura, velocidad del viento, etc. Comúnmente esta información se registra en el tiempo con cierta frecuencia. A veces se realiza con una resolución temporal y otras veces se realiza eventualmente.

En las ciencias de la tierra, se establece una herramienta para la planificación de la explotación racional de los recursos naturales, comprender las causas de los fenómenos naturales y como las personas influyen en la naturaleza con sus acciones. Por otra parte, son el medio para comprender ciertos procesos naturales que ponen en peligro la vida del ser humano y su estudio está ligado con la prevención de riesgos sísmicos, meteorológicos y volcánicos en los ámbitos continental, oceánico y atmosférico. Las disciplinas de las ciencias de la tierra requieren de un análisis de datos espaciotemporales para llevar a cabo una correcta toma de decisiones.

Un sistema de información geográfico puede reconocer y analizar las relaciones espaciales que existen en la información geográfica almacenada. Aquellas relaciones topológicas permiten crear modelos y análisis espaciales complejos. Además, un sistema de información geográfico tiene como objetivo, el análisis de rutas, geoestadística, álgebra de mapas, entre otros. Con el uso de los datos espaciotemporales, la lectura, escritura, transporte y visualización de dichos datos necesitan de formatos que faciliten este proceso. Gran parte de los sistemas de información geográficos permiten una variada gama de formatos, algunos de estos son ESRI Shaperfile, ESRI Raster y NetCDF. El Open Geospacial Consortium (OGC) es una de las principales organizaciones que definen estos estándares para estos formatos. Geographic Markup Languaje (GML), Keyhole Markup Languaje (KML) que se usan para el despliegue de datos en dos y tres dimensiones, respectivamente.

Debido a la gran variedad de formatos que existen para la gestión de datos espaciotemporales, debemos elegir uno. Para aquello, es fundamental conocerlos para optimizar su uso en tiempo de ejecución, espacio de almacenamiento y capacidad de expresión, etc.

En base a lo anterior mencionado, es necesario analizar los diferentes formatos, su composición estructural y su dinámica; la fortaleza de cada uno de ellos en términos de transporte de datos, almacenamiento o visualización; su principal aplicación, si esta es para la web, SIG, o si es ideal para gestionar formatos vectoriales, raster o series de datos. Para poder elegir un formato, presentaremos una caracterización de los formatos utilizados en la gestión de datos espaciotemporales en los SIG, concretamente, en formatos de almacenamiento de modelos raster, vector y series de datos. Esta caracterización muestra los elementos de cada formato, las situaciones y ambientes que los potencia, mediante esquemas preconceptuales (EP). Los EP permiten identificar claramente las relaciones estructurales y dinámicas de cualquier ámbito o dominio del conocimiento.

2.1.2. Datos geográficos.

En cuanto a la información geográfica (IG) esta posee una localización implícita (población de un lugar censado, una referencia catastral, etc.) o explicita (coordenadas obtenidas a partir de datos capturados mediante GPS, etc.) respecto la tierra. Dicha información geográfica es gestionada mediante los SIG con modelos de datos raster, vector y eventualmente, con la asociación de estos modelos a series de datos.

El modelo de datos raster está diseñado para la gestión de la información continua en el

espacio como elevación y precipitación. Dicho modelo se representa mediante una matriz de datos en la que cada celda corresponde a una región geográfica. El tamaño de las celdas se denomina resolución espacial y determina la precisión de la información almacenada en una capa de datos raster. Por consecuencia, a medida que es mayor la resolución espacial, la precisión disminuye, dado que el valor de cada celda debe ser representativo de un área mayor.

El modelo de datos vector gestiona información discreta en el espacio, como las calles, casas, ríos, etc. Este se basa en tres primitivas geométricas; el punto, la línea y el polígono. Una capa de datos vector se denomina shapefile y cada uno de sus elementos se denominan shapes (por ejemplo, el conjunto de casas del centro de Concepción es un shapefile y cada casa es un shape). La información que es asociada a cada shape se almacena en la fila de una tabla de atributos. Estos campos de la tabla de atributos son definidos por el interesado (por ejemplo, propietario, número de teléfono, etc.)

Los SIG permiten incorporar series de datos asociados con los objetos vector y raster. Usualmente, estas series representan a series de tiempo y a perfiles verticales de medición. Para las series de tiempo, estas pueden almacenar información con una resolución temporal definida o eventual. Para los perfiles verticales de medición son muestreos obtenidos en la columna de aire o agua.

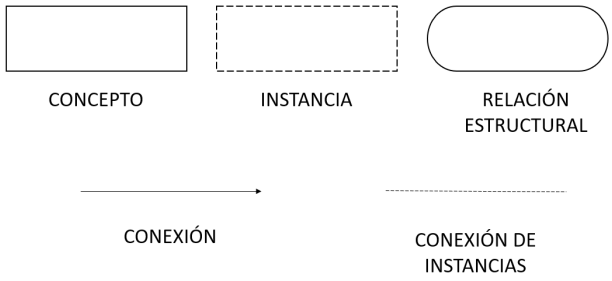

Figura 2.1: Elementos esquema preconceptual

2.1.3. Esquemas preconceptuales

Son representaciones intermedias entre las especificaciones textuales en lenguaje natural de cualquier dominio del conocimiento y los diferentes esquemas conceptuales que permiten el modelado de una pieza de software. Dichos esquemas representan las relaciones estructurales y dinámicas del dominio. En la anterior figura se presentan los elementos de los esquemas preconceptuales, cuyo significado es el siguiente:

(a) El concepto representa los sustantivos del dominio
(b) Las instancias son conjuntos de valores que pueden tomar un concepto y que sirven para aclararlo. Los valores que se presentan en las instancias no son necesariamente todos los posibles valores que puede tomar el concepto, algunos se excluyen.
(c) Las relaciones estructurales definen relaciones permanentes entre dos conceptos con el uso de los verbos “ser” y “tener”.
(d) Las conexiones son vínculos entre conceptos y relaciones estructurales o viceversa.
(e) Las conexiones de instancias permiten ligar un conjunto de instancias con el concepto del que son valores.

2.1.4. Caracterización de formatos de almacenamiento.

En la caracterización se incluirán los formatos de almacenamiento, transporte y visualización más relevantes en el dominio de los SIG, concretamente, sobre los modelos de dato vector, raster y sobre la gestión de series de datos. La caracterización se llevó a cabo a través de la definición y descripción básica de cada formato y la síntesis de los elementos principales de cada uno mediante un esquema preconceptual para facilitar la comprensión.

2.1.5. CDF, HDF, NetCDF.

Los formatos CDF (Common Data Format), HDF (Hierarchical Data Format) y NetCDF (Network Common Data Form) son formatos científicos para el almacenamiento, transporte y procesamiento de datos multidimensionales independientes del dominio. Estos son desarrollados y mantenidos por la NASA, la National Center for Supercomputing Applications (NCSA) de la University of Illinois y la National Center for Atomospheric Research (NCAR), respectivamente.

Al tratarse de formatos binarios, su gestión está dada a través de las librerías que son creadas para cada uno de estos. En el caso de los formatos CDF, cuentan con implementaciones en lenguaje de programación C, Fortran, Java, Perl y C#. En el caso de HDF cuenta con implementaciones en Java, Matlab, IDL y Python. Para el caso de los formatos NetCDF en Java, Python y Matlab. Estos formatos son autodescriptivos debido a que tienen la capacidad para almacenar metadatos y atributos de los datos. HDF cuenta con versiones HDF, HDF4 y HDF5. Por otro lado, NetCDF está basado en los modelos de datos CDF y HDF. En las siguientes figuras se presenta un breve resumen de los tres modelos de datos, respectivamente.

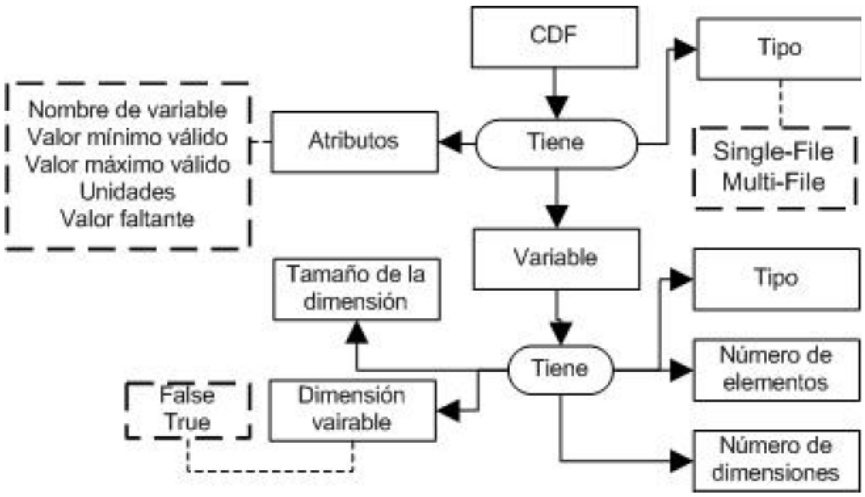

Figura 2.2: Esquema preconceptual CDF. (Alberto, 2014)

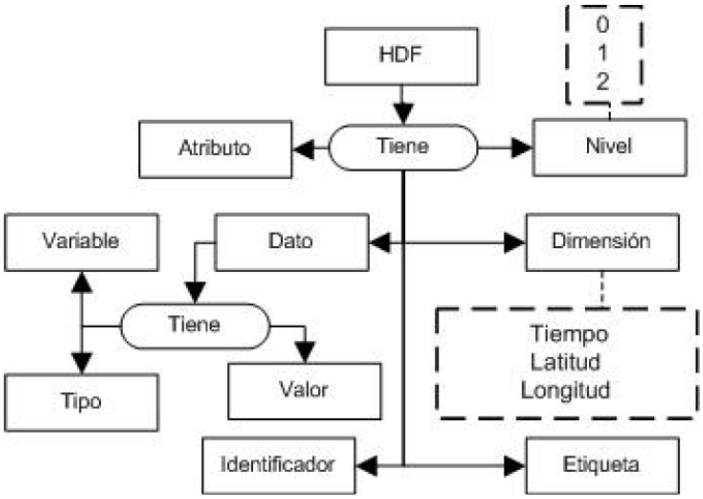

Figura 2.3: Esquema preconceptual HDF. (Alberto, 2014)

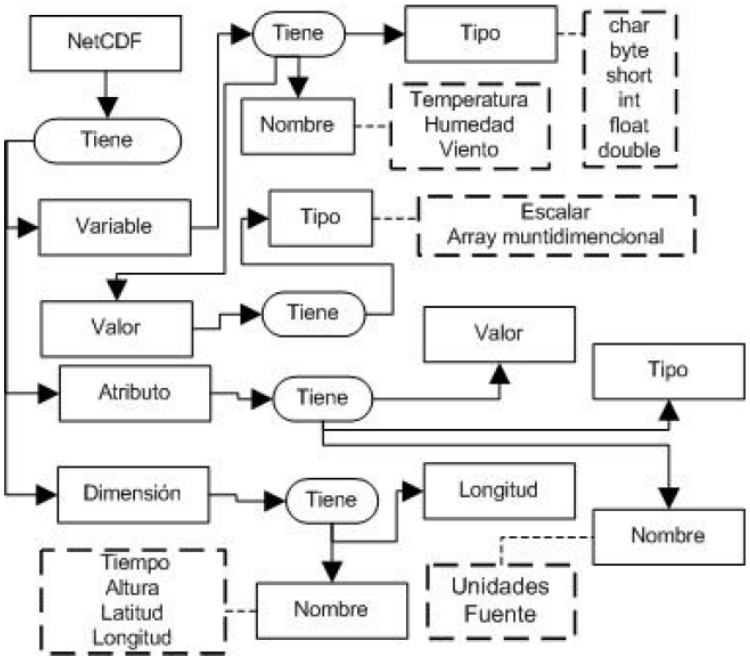

Figura 2.4: Esquema preconceptual NETCDF. (Alberto, 2014)

Formato	**Autor**	**Uso Recomendado**
KML	Google Ink.	Modelos vector web.
ESRI Shapefile	ESRI.	Modelos vector en SIG de escritorio.
ESRI Grid	ESRI.	Modelos raster en SIG de escritorio
GML	OGC.	Modelos raster y vectores en la web.
NetCDF	UCAR	Series de datos y modelos raster tanto en la web como SIG de escritorio.
CDF	NASA.	Series de datos y modelos raster tanto en la web como SIG de escritorio.
HDF	NCSA.	Series de datos y modelos raster tanto en la web como SIG de escritorio.

Tabla 2.1: Formatos de datos científicos

Formato	**Algunos programas que lo soportan**	**Tipo de formato**
KML	Google Earth, Google Maps, Microsoft Virtual Eath.	Transporte y visualización.
ESRI Shapefile	ArcInfo, MapWindow, GvSIG, GeoDA.	Almanenamiento, transporte y visualización.
ESRI Grid	ArcInfo, MapWindow, GvSIG.	Almanenamiento, transporte y visualización.
GML	ArcInfo, servicios Web WFS.	Almanenamiento y transporte.
NetCDF	ArcInfo, Servicios Web OpenNDAP, Java, Python y Matlab	Almanenamiento y transporte.
CDF	Servicios Web OPeNDAP, C, Fortran, Java, Perl y C#.	Almanenamiento y transporte.
HDF	Servicios Web OPeNDAP, Java, Matlab, IDL y Python.	Almanenamiento y transporte.

Tabla 2.2: Formatos de datos científicos, programas y tipo

2.1.6. Análisis de la caracterización realizada.

Podemos concluir que para los formatos de transporte, almacenamiento y visualización de datos geográficos revisados anteriormente se pueden reunir en tres grupos:

(a) Para la gestión de modelos vector.
(b) Para la gestión de modelos raster.
(c) Para la gestión de modelos de series de datos.

KML y ESRI shapefile son formatos recomendados para trabajar exclusivamente en el grupo A, aunque KML es mejor para el trabajo de red y ESRI shapefile en SIG de escritorio. GML se presenta como una opción para los modelos vector como para los raster en la web, y específicamente, a través de servicios web OGC como es el caso de WFS (Web Feature Service).

Por otro lado, los formatos CDF, HDF y NetCDF están pensados para almacenar y transportar paralelepípedos de datos que contienen múltiples dimensiones, incluidos el tiempo y las coordenadas espaciales. De esta manera un corte transversal sobre los datos permite obtener un modelo raster, mientras que un corte longitudinal logra series de datos. La mayor fortaleza de estos tres formatos es la posibilidad de describir la información que contienen en términos de unidades de medida, fuente de obtención y todos los metadatos o atributos que el usuario requiera almacenar. Esto hace que los datos almacenados en CDF, HDF y NetCDF sean fáciles de interpretar. Además, estos formatos actualmente son utilizados a través de servicios OPeNDAP. Sin embargo, también cuentan con un amplio conjunto de librerías que facilitan su gestión. Como pudimos apreciar en las anteriores dos tablas, se presenta un resumen con algunos de los elementos adicionales de los formatos vistos.

Entre las características comunes de estos formatos podemos encontrar la visualización o renderizado, ya que este procedimiento se hace desde la aplicación del cliente que soporta el formato del dato. Para visualizar de mejor manera a continuación veremos algunas de las salidas de visualización de los formatos explorados.

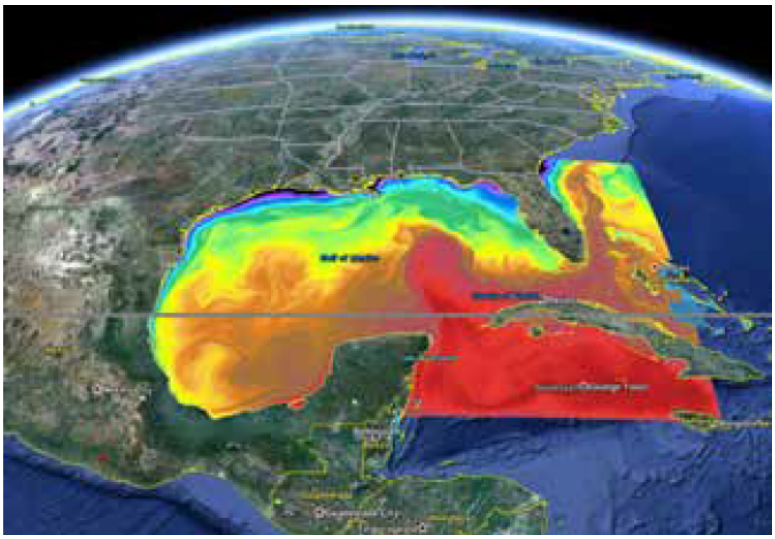

Figura 2.5: Despliegue y visualización a partir de un formato NetCDF. (Zavala, 2013).

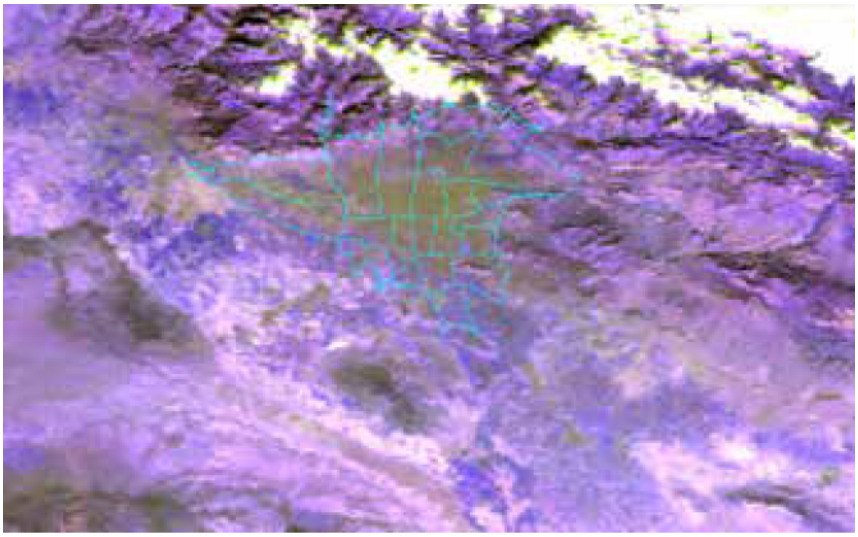

Figura 2.6: Despliegue y visualización a partir de un formato NetCDF 2. (Sohrabinia, 2012)

Convención	**Herramienta**
1	ArcGIS.
2	MapWindow GIS.
3	Google Earth.
4	Google Maps.
5	GRASS.
6	gvSIG.
7	Quantum GIS.
8	SAGA GIS.

Tabla 2.3: Convenciones de herramientas que soportan los formatos explorados

Formato	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
KML	X	X	X	X	X	X	X	X
ESRI Shapefile	X	X	X	X	X	X	X	X
ESRI Grid	X	X			X		X	
GML	X		X	X	X	X	X	
NetCDF	X				X	X	X	X
CDF	X				X		X	
HDF	X				X		X	

Tabla 2.4: Convenciones de herramientas que soportan los formatos explorados 2

Para los distintos formatos de almacenamiento, transporte y visualización presentados se encuentran soportados por varios SIG y globos virtuales, lo que hace posible su uso e interoperabilidad. En anterior tabla se presenta un resumen de diversas herramientas que permiten la visualización de los formatos explorados con base en las convenciones de la tabla anterior a esta.

2.1.7. Ficheros NetCDF

Los archivos con formato netCDF se han vuelto un estándar en el mundo actual, especialmente en lo referido a temas científicos o interpretación de variables complejas, como por ejemplo datos climáticos e información espacio-temporal. Los archivos con formato netCDF son ficheros con capas en formato ráster o vectorial que albergan una amalgama compleja de variables que otros archivos no son capaces de admitir. Cabe destacar que los archivos netCDF tienen como extensión .nc .Gracias a la superposición de capas que componen los archivos de tipo .nc logramos acceder a variables de forma estratégica ya que estos pueden ser representados tanto por su temática y también por su momento temporal. Es por estas cualidades que los archivos netCDF sean sumamente útiles cuando hablamos de trabajar con variables e índices temporales, como es por ejemplo en el caso de estudio de este proyecto con valores de latitud, longitud, temperatura entre otros.

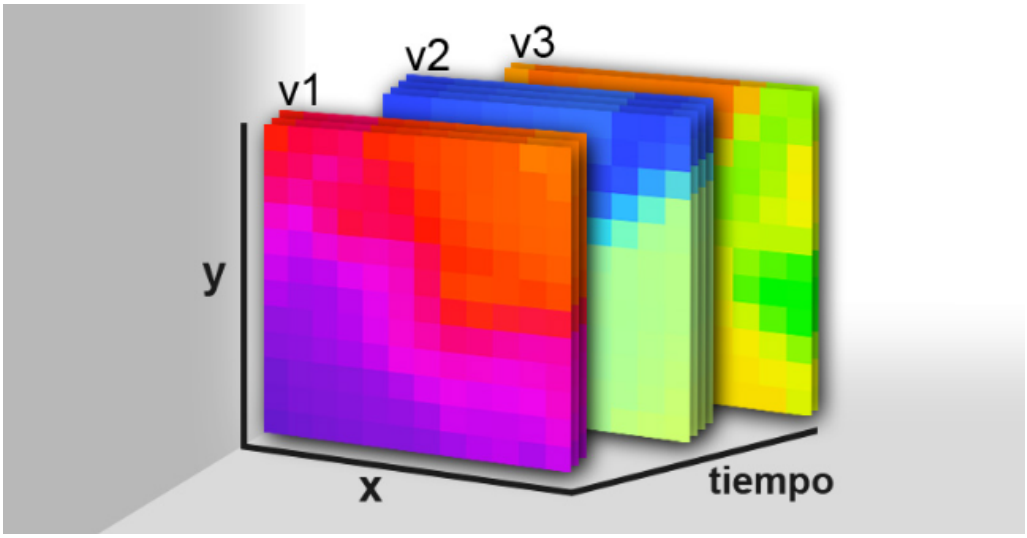

Figura 2.7: Representacion archivo NetCDF

Por otra parte, en términos de la representación temática de los archivos .nc podremos representar los datos y en el caso de la dimensión de los archivos podremos acceder a propiedades de la variable. En términos simples seria como filtrar información precisa dejando de lado los datos que no nos interesen, así también visualizando momentos temporales o características de estos. Es por ello que podemos acceder a una secuencia temporal de valores de temperatura territorial como es el caso de la región de ñuble. (Alberto, 2014)

2.1.8. Propósito de los archivos NetCDF

El principal propósito en el cual se fundamentan los Network Common Data Form (netCDF) es permitir la creación, acceso y finalmente lograr compartir datos en forma de array en un formulario portátil y autodescriptivo esto significa que los conjuntos de datos incluyen información que permite definir el contenido de estos. NetCDF fue creado por Unidata, sistema que pertenece a UCAR (University Corporation for Atmospheric Research), para el tratamiento eficaz de datos de tipo científico. Si bien su desarrollo estaba basado en el lenguaje de programación C, también existen interfaces que nos facilitan su utilización en otros lenguajes de programación, tales como Matlab, java, Ruby, Python y R.

2.1.9. NetCDF (Network Common Data Form)

Los NetCDF son particularmente un conjunto de bibliotecas de software o librerías y formato de datos que tienen como característica principal ser autodescriptivos, multiplataforma y tienen la ventaja de permitir guardar los datos en formato array de una forma eficaz y clara. Los Network Common Data Form también son un estándar comunitario con el fin de compartir datos científicos.

2.1.10. Características de los NetCDF

Los ficheros NetCDF tienen la gran particularidad de ser autodescriptivos ya que estos ficheros incluyen información que describe los datos que están alojados en ellos. Además de lo anterior son de un formato portable ya que se puede acceder a ellos desde sistemas con distintas formas de almacenamiento de caracteres, números de tipo floats y enteros. Los Network Common Data Form son escalables ya que se puede acceder a subconjuntos de grandes volúmenes de datos en distintos formatos por medio de las distintas interfacez de NetCDF, además de poder acceder a ellos desde servidores remotos. También este tipo de ficheros son anexables dado que se les pueden agregar datos sin redefinir su estructura o copiar el conjunto de información. Los archivos NetCDF tienen la facultad de poder ser accedidos por un escritor y varios lectores de manera simultánea.

2.1.11. Interfaz de archivos NetCDF

Como bien mencionamos los archivos NetCDF (Network Common Data Form) son una interfaz con una multitud de funciones de acceso a datos que permiten almacenar y recuperar información en forma matricial. Una matriz en términos simples es una estructura de datos que nos permite almacenar un conjunto de información del mismo tipo de datos (carácter de 8 bits, entero de 32 bits, etc).En otras palabras los Network Common Data Form son una abstracción que permite almacenar una colección de objetos autodescriptivos con el fin de poder acceder a ellos desde una interfaz simple e intuitiva.

Si nos adentramos un poco en el funcionamiento de los NetCDF podemos darnos cuenta que se puede acceder directamente a los valores alojados en la matriz sin la necesidad de conocer a fondo el cómo se almacenan los datos. Por otra parte, los grupos de datos NetCDF al ser acezados de forma sencilla permiten de esta manera mostrar, analizar, transformar y combinar campos específicos de datos.

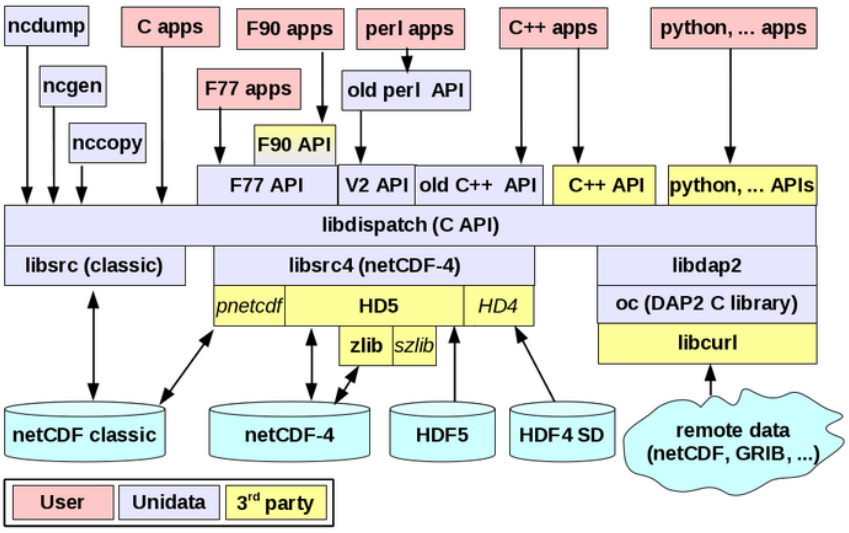

Figura 2.8: Arquitectura de la Biblioteca NetCDF
(Program, 1991)

Las bibliotecas de arquitectura netCDF basadas en C dependen de una biblioteca central y algunas bibliotecas desarrolladas externamente.

2.1.12. Rendimiento de NetCDF

Uno de los principales objetivos de los Network Common Data Form es permitir un acceso eficiente a pequeñas partes de los grandes volúmenes de datos que se trabajan en ellos. Para esta tarea, los archivos NetCDF utilizan el acceso directo en lugar de acceso secuencial, de esta manera se permite un desempeño mucho más eficaz al leer datos en un orden distinto al que se escribieron o simplemente se desean leer los datos en un orden distinto para fines diversos. La sobrecarga al trabajar con este tipo de ficheros depende de muchos factores, uno de ellos es el tipo de dato, el rendimiento del pc en el cual estemos accediendo a este tipo de ficheros, la granularidad de acceso a los datos (acceder a aquellas partes de la información que nos interesan y descartar aquellas que no son adecuadas en el contexto donde nos encontramos) y qué tan bien implementado este el sistema con el cual estemos tratando estos ficheros. Sin embargo, la sobrecarga de la capa de representación externa suele ser un problema menor cuando hablamos de acceder a la información o datos.

Versión	**Descripción**
NetCDF Clásico CDF-1	Es el formato de archivo NetCDF predeterminado (CDF-1) el cual posee la máxima portabilidad, el encabezado de este tipo de archivo se identifica mediante 4 bytes.
NetCDF 64 Bits CDF-2	Las limitaciones de formato de 2 GiB del archivo NetCDF clásico (CDF-1) se convirtieron en un dolor de cabeza para algunos usuarios que demandaban un mayor volumen de datos, por lo cual el formato que compenso este problema fue el de 64 Bits introducido en la versión 3.6.0, conocido como CDF-2.
NetCDF 64 Bits CDF-5	Nuevamente la necesidad de trabajar con un mayor volumen de datos se hizo presente, este formato introducido en la versión 4.4.0, permite trabajar con variables de mas de cuatro mil millones de elementos de una matriz.
NetCDF-4	En esta versión se incorporan nuevas funcionalidades tales como tipos compuestos de datos, matrices de longitud variable, acceso de E/S paralelo, entre otras. En esta versión 4.0 se incluye el formato subyacente HDF5.

Tabla 2.5: Rendimiento de NetCDF

2.1.13. Estructura de formatos CDF-1, 2 y 5

En estos formatos de archivo NetCDF los datos se almacenan como un solo archivo el cual se compone de dos partes. El encabezado que contiene todos los datos sobre las dimensiones, variables y atributos, sin incluir los datos variables. La parte de datos, la cual abarca datos de tamaño fijo, en esta se alojan los datos de las variables que no tienen una dimensión ilimitada, además aloja datos de tamaño variable, que contienen los datos de las variables que tienen una dimensión ilimitada.

2.1.14. Estructura de formatos NetCDF-4 HDF5

Los ficheros NetCDF-4 se pueden crear con la biblioteca HDF5, son archivos HDF5 en cualquier sentido y su lectura esta permitida sin la interfaz netCDF-4. A su vez los grupos en un fichero netCDF-4 se corresponden con los grupos HDF5.Por otro lado las variables y atributos en netCDF se atribuyen con conjuntos de datos con nombres similares en HDF5. Puesto que existen muchos más metadatos en un fichero netCDF que en un fichero HDF5, se hacen uso de conjuntos de datos especiales para poder almacenar los metadatos en netCDF.

2.1.15. Modelo de datos

En los archivos NetCDF los grupos de datos contienen dimensiones, variables y atributos los cuales se identifican por un nombre y un número. Todos estos elementos se pueden utilizar en conjunto para poder capturar el significado de los datos y las relaciones entre los distintos campos de información.

- Grupos: Son estructuras que alojan en su interior variables, atributos, dimensiones y otros grupos.
- Dimensiones: Si bien pueden existir dimensiones ilimitadas que no presentan una envergadura limitada y puede alojar una infinidad de datos en las variables que se definirán lo común es que las dimensiones presenten un tamaño de las variables que se quieran crear, las cuales tengan un nombre y que el tamaño represente la capacidad que tendrá el array, este mismo es representado por la variable definida en la dimensión.
- Variables: Estas desempeñan la labor de alojar los datos en el archivo. Como se menciono anteriormente su tamaño es definido previamente por la dimensión y este tamaño representa la capacidad misma del array. También pueden no tener una dimensión definida. Las variables pueden tener el formato de cadenas de caracteres, número entero o flotante, con signo y sin signo.
- Atributos: en los ficheros NetCDF los atributos cumplen la función de contener información sobre metadatos o datos auxiliares.

Capítulo 3

Estado del Arte

En este capítulo hemos dividido el estado del arte en cuatro categorías, las cuales son:

(a) Lenguajes de programación.
(b) Bases de datos.
(c) Aplicaciones comerciales.
(d) Investigación

3.1. Lenguajes de programación

A través de la información obtenida acerca de los lenguajes de programación más apropiados para el análisis de datos científicos, debido a que algunos archivos NetCDF son bastante grandes lo cual representa una gran carga para el procesamiento de la información por cual es fundamental elegir un lenguaje de programación con la capacidad para procesar esta información, ya que si se diera el caso de elegir alguna tecnología que no es apta para el procesamiento de grandes volúmenes de datos, esta quedará estancada sin poder procesar la información necesaria. Nos encontramos una variada gama de tecnologías las cuales algunos corresponden a lenguajes de programación tradicionales (C, C++, Java, etc.) y otros no tan conocidos pero que están emergiendo con bastante fuerza en el campo del análisis de datos.

En teoría, la mayoría de los lenguajes de programación pueden hacer análisis de datos, tan simple como abrir un fichero, leerlo y extraer la información contenida en él. Pero la tarea se dificulta cuando los volúmenes de información que contienen los ficheros son tan grandes que requiere una capacidad de procesamiento mucho mayor junto con mayor poder de cómputo.

Entre los lenguajes de programación más destacados para el análisis de datos científicos se encuentran, los siguientes:

(a) R.
(b) Python.
(c) MATLAB.
(d) Julia.

Según una encuesta realizada por la Universidad de Alcalá, España, el lenguaje de programa-

ción más usado para el análisis de datos es **R**, seguido de **Python**. A continuación, analizaremos algunas características de ambos lenguajes.

Python se le considera el primer lugar en la lista de varias áreas de TI, tanto para la seguridad informática, inteligencia artificial, etc. Debido a su potencia y su simplicidad en el código. La sintaxis que pertenece a Python es muy simple y se puede aprender fácilmente. Por lo tanto, muchos algoritmos de IA se pueden implementar fácilmente en él. Python tarda menos tiempo de desarrollo en comparación a otros lenguajes como Java, C++ o Ruby.

Además, Python admite estilos de programación multiparadigmas, soporta programación orientada a objetos, funcionales y orientada a procedimientos. Hay muchas bibliotecas en Python, lo que facilita nuestra tarea. Un ejemplo de esto es la biblioteca Numpy, que se utiliza para resolver una amplia gama de cálculos científicos.

R es uno de los lenguajes y entornos más efectivos para analizar y manipular los datos con fines estadísticos. Usando R, podemos producir fácilmente un *publication-quality plot* bien diseñado, incluyendo símbolos matemáticos y formulas donde sea necesario.

Además de ser un lenguaje de propósito general, R tiene numerosos paquetes como RODBC, Gmodels, Classs y Tm, que se utilizan en el campo del aprendizaje automático. Estos paquetes facilitan la implementación de algoritmos de aprendizaje automático para descifrar los problemas asociados de negocio. (Universidad de Alacalá, 2022)

3.1.1. R

En base a la documentación proporcionada por la UCAR la librería NetCDF debe ser la 1.5 que es compatible con R en su versión 2.2 o superior. Se han realizado intentos con netcdf 1.5 y R 2.0.1 pero sin éxito aún. Existen actualmente dos versiones de NetCDF para utilizar en R, NetCDF3 que es utilizada mayoritariamente por la comunidad científica, pero tiene algunas limitaciones en cuanto al tamaño y rendimiento de estos, a diferencia de NetCDF4, que permite conjuntos de datos mucho más grandes e incluye capacidades extras como lo es la compresión de archivos.

R tiene la cualidad de poder leer y escribir, por ende, puede analizar los archivos NetCDF, utilizando los paquetes o librerías NCDF y NCDF4 proporcionados por David Pierce, y a través de otras librerías como Raster, Metr1, RNETCDF. Los paquetes NCDF4.Helpers y EasyNCDF proporcionan algunas herramientas adicionales para el análisis de dichos archivos.

El paquete NCDF4 está disponible tanto en Windows como en Mac OS X y entornos Linux/Unix y admite el formato NetCDF3 anterior como NetCDF4.

La instalación para el sistema operativo Windows consta de básicamente 5 pasos, los cuales mencionaremos a continuación.

(a) Asegúrese de tener instalada la versión 2.2 o superior del lenguaje R.
(b) Desde el entorno R, seleccionar la alternativa “packages”.
(c) Elegir la opción descargar paquetes.
(d) Seleccionar la alternativa “netcdf”
(e) Descargar

La instalación para sistemas Linux, Unix, OS X, es similar. Consta de un proceso de dos

pasos, el primer paso es tener instalado la librería netCDF. Entonces puede instalar el paquete netcdf. El paquete netcdf necesita conocer la ubicación local de la librería para poder acceder a sus propiedades. Para la instalación del paquete ncdp (Es necesario para poder acceder de forma correcta al funcionamiento de la librería y su paquete NetCDF) debe de conocer donde está instalado la librería netCDF en su sistema operativo para poder instalar el paquete ncdf. En ocasiones esta la reconoce por defecto. Para entornos Linux el comando para encontrarlos es "locate netcdf.h" y "locate libnetcdf". Finalmente se debe ejecutar el siguiente comando desde la cosnola. *"R CMD INSTALL –configure-args=with-netcdf_incdir=/usr/local/netcdf/include -with-netcdf_libdir=/usr/local/netcdf/lib"ncdf_xx.yy.tar.gz".* (Corporación Universitaria para la investigación Atmosférica, 2010)

Ventajas.

(a) Excelente gama de paquetes de código abierto y de alta calidad. R tiene un paquete para casi todas las aplicaciones cuantitativas y estadísticas imaginables. Esto incluye redes neuronales, regresión no lineal, filogenia, cartografía, mapas y muchos otros.

(b) La instalación básica viene con funciones y métodos estadísticos integrales muy completos. R también maneja el algebra de matriz particularmente bien.

(c) La visualización de datos es una fortaleza clave en el uso de bilbiotecas como ggplot2.

Contras.

(a) Rendimineto. R no es un lenguaje rápido. Esto no es un accidente. R fue diseñado a propósito para facilitarle el análisis de datos y las estadísticas. No fue diseñado para hacer un rápido procesamiento. Mientras que R es lento en comparación con otros lenguajes de programación, para la mayoría de los propósitos, es lo suficientemente rápido.

(b) Especificidad de dominio. R es bueno para fines estadísticos y científicos de datos. Pero no es tan bueno para programadores de propósito general.

(c) Es complejo. Tiene algunas características poco frecuentes que pueden atrapar a los programadores con experiencia en otros idiomas. Por ejemplo, indexación desde 1, utilizando operadores de asignación múltiple, estructuras de datos no convencionales.

3.1.2. Python

Actualmente la librería de NetCDF para Python se encuentra en la versión 1.6.0 según el sitio web oficial de UNIDATA (Servicios de datos y herramientas para la geociencia). La librería "netCDF4" correspondiente a Python (netcdf4-python) es una interfaz de Python para la biblioteca NetCDF C.

La versión 4 de NetCDF tiene muchas características que no se encontraron en versiones anteriores de la librería y se implementan en la parte superior de HDF5. En este módulo puede leer y escribir archivos tanto en el formato anterior NetCDF3 como en el nuevo formato NetCDF4 y puede crear archivos legibles por los clientes HDF5. La API modelada después de Scientific.io.netcdf y debe ser familiar para los usuarios de este módulo.

Se implementan la mayoría de las nuevas características de NetCDF4, tales como múltiples dimensiones ilimitadas, grupos y comprensión de datos. Se implementan todos los nuevos tipos de datos numéricos (como 64 bits y tipos de enteros sin firmar). Los tipos de datos compuestos

(struct), longitud variable (VLEN) y enumerados (enum) son compatibles, pero no el tipo de datos opacos. No son compatibles con las mezclas de tipos de datos compuestos, VLEN y enum (como tipo de compuestos que contienen enums o VLEN que contienen tipos de compuestos). (Services y Tools for Geoscience, 2020)

Ventajas.

(a) Lenguaje de programación de propósito general extremadamente popular y ampliamente utilizado por la comunidad de data science.
(b) Python es un lenguaje de programación de uso general muy popular. Cuenta con una amplia gama de módulos específicos y soporte comunitario. Los principales GIS de escritorio como ArcGIS (con la ArpPy), QGIS (con PyQGIS) o gvSIG la introducción de Python.
(c) Python es un lenguaje fácil de aprender. La baja barrera de entrada lo convierte en un lenguaje ideal para aquellos que son nuevos en programación.
(d) Paquetes como pandas, scikit-learn y Tensorflow hacen de Python una opción solida para aplicaciones de aprendizaje automático.

Contras.

(a) Seguridad de tipos. Python es un lenguaje de tipado dinámico, lo que significa que debemos ser cuidadosos al momento de utilizar las variables. Los errores de tipo (como pasar un string como un argumento a un método que espera recibir un número entero) deben esperarse de vez en cuando.
(b) Para los fines específicos de análisis estadístico y de datos, la amplia gama de paquetes de R le da una ligera ventaja sobre Python. Para los lenguajes de propósito general, hay alternativas más rápidas y seguras que Python.

3.1.3. Java

La librería de NetCDF para Java implementa el modelo de datos comunes conocido como CDM para interconectar archivos NetCDF con una variedad de formatos de datos, por ejemplo: netCDF, HDF, GRIB, entre otros. En capas por sobre el acceso a datos básicos, el CDM manipula los metadatos contenidos en los conjuntos de datos para generar una interfaz de nivel superior a las características específicas de geociencia de los conjuntos de datos, en particular, proporcionando geolocalización y subconjuntos de datos en el espacio de coordenadas.

El TDS usa CDM/netCDF-Java para leer conjuntos de datos en varios formatos. El CDM también proporciona la base para todos los servicios disponibles a través del TDS.

Un framework conectable permite que otros desarrolladores agreguen lectores para sus propios formatos especializados. El CDM también proporciona una API estándar para sistemas de coordenadas de georreferenciación y consultas especializadas para tipos de características científicas como conjuntos de datos de cuadricula, punto y radial.

La biblioteca netCDF-Java es un marco de trabajo 100 % Java para leer netCDF y otros formatos de archivo netCDF-3. Escribir en el formato de archivo netCDF-4 requiere instalar la biblioteca netCDF C. La biblioteca netCDF-Java también implementa NcML, que le permite agregar metadatos a conjuntos de datos CDM, asi como crear conjuntos de datos virtuales a

través de la agregación. El servidor de datos THREDDS (TDS) está construido sobre la biblioteca netCDF-Java.

NetCDF-Java es un software gratuito y de código abierto y está alojado en Github. La mayoría de los proyectos usan "netcdfAll.jar", "toolsUI.jar" o incluyen los artefactos deseados en sus compilaciones Maven o grandle. A partir de la versión 5.0 netCDF-Java se publica bajo la licencia BSD-3. (Services y Tools for Geoscience, 2022)

Ventajas.

(a) Ubicuidad. Muchos sistemas y aplicaciones modernas se basan en un back-end de Java. La capacidad de integrar métodos de ciencia de datos directamente en la base de código existente es poderosa.
(b) De tipado fuerte. Java es un buen lenguaje cuando se trata de garantizar la seguridad de los tipos de datos. Para aplicaciones de big data de misión crítica, esto es muy importante.
(c) Java es un lenguaje compilado de propósito general y de alto rendimiento. Lo que lo hace adecuado para escribir eficientes códigos de producción ETL y algoritmos de machine learning muy intensivos computacionalmente.

Contras.

(a) Para análisis ad-hoc y aplicaciones estadísticas más dedicadas, la verbosidad de Java hace que sea una primera opción poco probable. Los lenguajes de scripting de tipado dinámico como R y Python se prestan a una productividad mucho mayor.
(b) En comparación con los lenguajes específicos de dominio como R, no dispone de muchas librerías disponibles para métodos estadísticos y de análisis de datos científicos avanzados.

3.1.4. Scala

En el caso de Scala, no existe una librería como tal para manejar archivos NetCDF a diferencia de los otros lenguajes vistos anteriormente. En los casos anteriores la tarea de manejar los archivos NetCDF se resume en básicamente en descargar la librería directamente, el código fuente o desde algún gestor de paquetes y luego utilizarla en código. Esto ocurre de forma diferente en Scala. Si bien cualquier código de Scala puede ejecutar como código de Java haciendo uso de la Java Virtual Machine (JVM) para poder manipular archivos multidimensionales ocurre un proceso diferente. Existen un par de tecnologías que nos facilita el manejo de datos espaciales a gran escala. Es ahí donde entra en juego Apache Sedona (en incubación) que es un sistema informático de clúster para el procesamiento de datos espaciales a gran escala. Apache Sedona amplía los sistemas informáticos de clúster ya existentes, como Apache Spark y Apache Flink, con un conjunto de conjunto de datos espaciales distribuidos listos para usar y Spatial SQL que cargan, procesan y analizan de manera eficiente datos espaciales a gran escala en todas las maquinas.

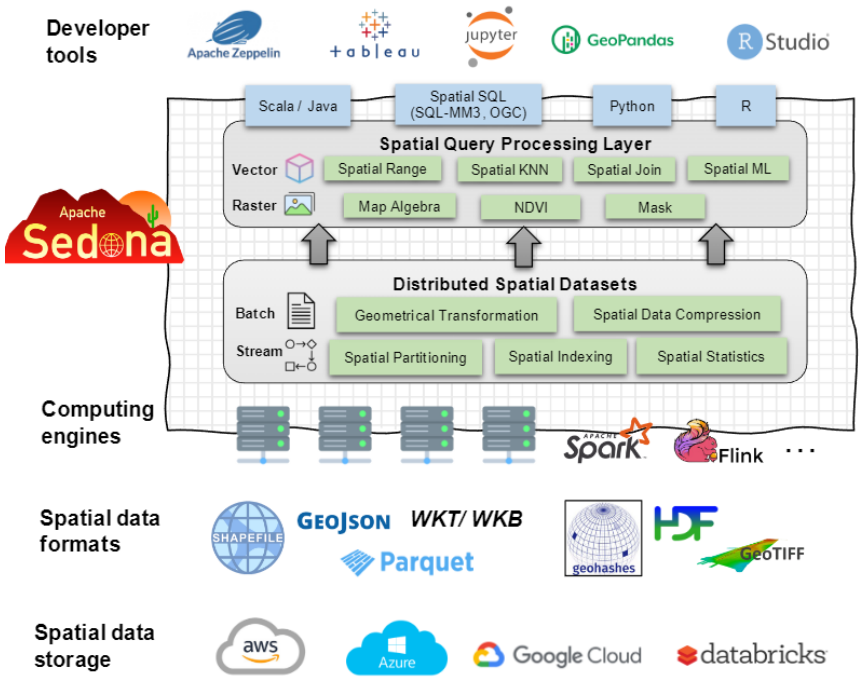

Figura 3.1: Apache Sedona. (Foundation, 2021)

Apache Spark es un motor de análisis unificado para el procesamiento de datos a gran escala. Proporciona una API de alto nivel en Java, Scala, Python y R. Además de un motor optimizado que admite gráficos de ejecución general. También es compatible con un amplio conjunto de herramientas de alto nivel, como Spark SQL para SQL y procesamiento de datos estructurados, pandas API en Spark para cargas de trabajo de pandas, MLlib para aprendizaje automático, GraphX para procesamiento de gráficos y transmisión estructurada para computación incremental y procesamiento de secuencias. (Foundation, 2022)

h5spark le permite acceder a HDF5 en Scala y el autor de h5spark demostró que Scala funcionó mucho más rápido que Python. Dado que no hay una API de Scala para leer archivos NetCDF ni HDF, el h5spark utiliza HDF-Java basado en JNI (Java Native Interface) para acceder a los datos. Dado que también hay una API HDF-Java para archivos HDF4, puede usar el mismo programa de Scala para reemplazar el lector HDF5 de h5spark. Si se desea acceder a HDF utilizando Java puro sin depender de JNI, el lector basado en Java netCDF de Unidata se implementa en SciSpark y el lector de Java puro se refina para leer conjuntos de datos individuales. Al igual que h5spark, SciSpark también usa RDD (Resilient Distributed Dataset).

SciSpark es una plataforma de procesamiento científico escalable que hace posible la computación y la exploración interactivas. Este proyecto prototipo describe la arquitectura del RDD científico (SRDD) , una biblioteca de álgebra lineal y operaciones geoespaciales. Su enfoque inicial es expresar el algoritmo Grab em'Tag em'Graph em'como un algoritmo de reducción de mapa.

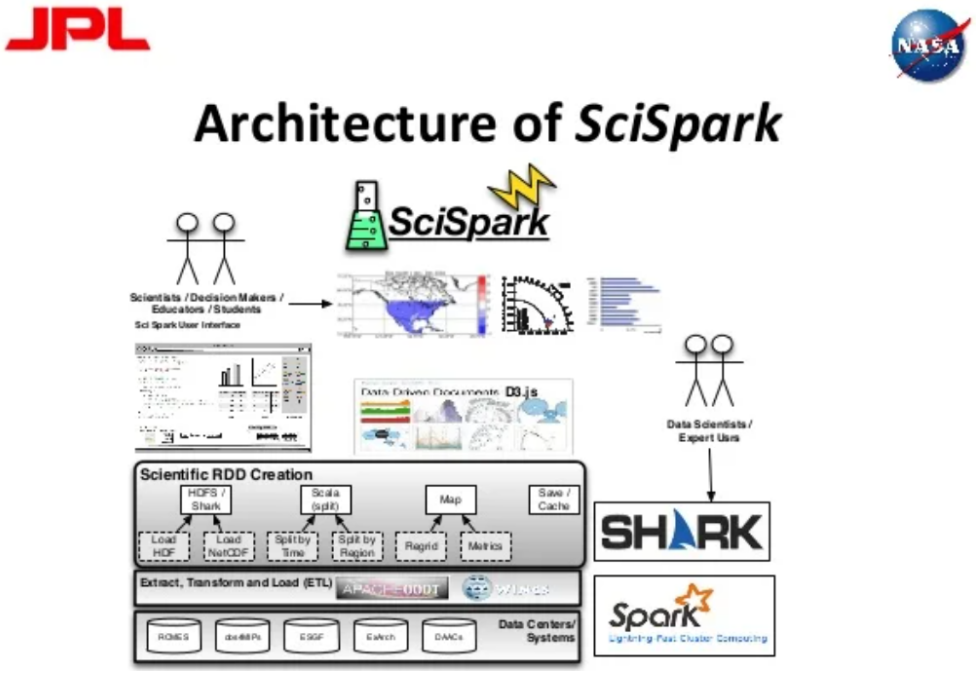

Figura 3.2: SciSpark. (Scientific Spark – a NASA AIST14 project, 2018)

Finalmente, como vimos anteriormente estas tecnologías nos permiten manipular datos espaciales a gran escala y otros formatos de archivos multidimensional por medio de clústers pero no concretamente archivos NetCDF. **La única forma de poder acceder a manipular los archivos NetCDF a través de Scala es utilizando la librería de Java para NetCDF de Unidata.**

Ventajas.

(a) Scala + Spark = computación en cluster de alto rendimiento. Scala es un lenguaje ideal para quienes trabajan con conjuntos de datos de gran volumen.
(b) Multi-paradigmatico. Los programadores de Scala pueden tener lo mejor de ambos mundos. Tanto la programación orientada a objetos como funcional.
(c) Scala se compila en el bytecode de Java y se ejecuta en una JVM (Java Virtual Machine). Esto permite la interoperabilidad con el lenguaje Java en si, haciendo de Scala un lenguaje de propósito general muy poderoso, además de ser adecuado para ciencias de datos.

Contras

(a) Scala no es un lenguaje sencillo para comenzar a utilizar si está empezando. Lo mejor es descargar sbt y configurar un IDE como Eclipse o IntelliJ con un complemento específico de Scala.
(b) La sintaxis y el sistema de tipos se describen con frecuencia como complejos. Esto hace que la curva de aprendizaje sea pronunciada para aquellos que vienen de lenguajes dinámicos

como puede ser el caso de Python.

3.1.5. Julia

Para el manejo de archivos NetCDF en el lenguaje de programación Julia, existen algunas alternativas disponibles. Una de ellas es utilizar el gestor de paquetes "Conda" escrito en Python pero que es multiparadigma y multilenguaje, por lo cual, puede gestionar proyectos que contengan código de otros lenguajes de programación, entre ellos, Julia.

Otra alternativa es utilizar las librerías *"NCDatasets.jl"* y *"NetCDF.jl"*.

Los *NCDataset.jl* permite leer y crear archivos NetCDF. El conjunto de datos y lista de atributos de NetCDF se comportan como los diccionarios de Julia y las variables como las matrices de Julia. El módulo NCDatasets proporciona soporte para las siguientes convenciones de netCDF CF:

(a) *_FillValue* se devolverá como faltante.
(b) *Scale_factor* y *add_offset* se aplican si están presentes las variables de tiempo (reconocidas por el atributo de unidades) se devuelven como objetos *DateTime*.
(c) Compatibilidad con los calendarios CF (estándar, gregoriano, gregoriano proléptico, juliano, todo bisiesto, sin bisiesto, 360 días).
(d) También se puede acceder a los datos sin procesar (sin las transformaciones anteriores)
(e) Representación de matriz irregular contigua.

Otras características incluyen:

(a) Soporte para compresión NetCDF 4 y arreglo de longitud variable (es decir, arreglo de vectores donde cada vector puede tener potencialmente una longitud diferente).
(b) El módulo también incluye una función de utilidad ncgen que genera el código de Julia que produciría un archivo NetCDF con los mismos metadatos que un archivo netCDF de plantilla.

(Barth, 2022)

Ventajas.

(a) Julia es un lenguaje compilado JIT (*just in time*) que le permite ofrecer un buen rendimiento. También ofrece las capacidades de simplicidad, tipado dinámico y scripting de un lenguaje interpretado como Python.
(b) Julia fue diseñada específicamente para el análisis numérico. Pero también ofrece programación de propósitos generales.
(c) Legibilidad. Muchos usuarios del lenguaje mencionan esto como una ventaja clave.

Contras.

(a) Madurez. Como nuevo lenguaje, algunos usuarios de Julia han experimentado inestabilidad al usar paquetes complementarios.
(b) Los paquetes limitados son otra consecuencia de la juventud del lenguaje y de la pequeña comunidad de desarrollo. A diferencia de Python o R, Julia no tiene la posibilidad de disponer de paquetes o librerías aún.

3.1.6. MATLAB

Puede leer o escribir archivos NetCDF utilizando funciones de alto nivel de MATLAB o el paquete de la librería NetCDF de funciones de bajo nivel. Las funciones de alto nivel simplifican el proceso de lectura de datos de un archivo NetCDF o la escritura de una variable del espacio de trabajo de MATLAB en un archivo NetCDF.

Para tener más control sobre el proceso de lectura y escritura, puede utilizar el paquete de biblioteca NetCDF que contiene funciones de bajo nivel. Las funciones de bajo nivel permiten un mayor control sobre el proceso de lectura y escritura al proporcionar acceso a las rutinas en la biblioteca NetCDF C.

Ventajas.

(a) Diseñado para la computación numérica. MATLAB es adecuado para aplicaciones cuantitativas con requisitos matemáticos sofisticados, como procesamiento de señales, transformaciones Fourier, algebra matricial y procesamiento de imágenes.
(b) Visualización de datos. MATLAB tiene incorporadas grandes capacidades de ploteado.
(c) MATLAB se enseña con frecuencia como parte de cursos de pregrado en asignaturas cuantitativas como física, ingeniería, matemáticas aplicadas. Como consecuencia, es ampliamente utilizado en estos campos.

Contras.

(a) Licencia propietaria. Dependiendo del uso (uso académico, personal o empresarial) es posible que tengamos que desembolsar una gran cantidad de dinero.
(b) MATLAB no es una opción obvia para programación de propósito general.
(Gleeson, 2017)

3.2. Bases de datos

Para poder almacenar archivos NetCDF en una base de datos relacional, lo que se realiza no es almacenar literalmente el fichero en una base de datos, para lograr aquello se necesitaría de un servidor de archivos para almacenar grandes cantidades de archivos, lo cual no es nuestro caso. Lo que realmente ocurre es crear una API en algún lenguaje de backend que se comunique con la base de datos y por medio de alguna librería se almacenan los rasters.

3.2.1. API

A continuación, especificaremos la API por medio del protocolo HTTP implementada por el componente EDE (*Environmental Discovery Engine*) de la imagen a continuación. En la próxima sección nos centraremos en la parte de PostgreSQL que será el motor de base de datos relacional que utilizaremos. La API se especifica como un archivo API Blueprint (Plano de API) en ede-rest-api y el lector puede mostrarla fácilmente en el documento HTML a través de un servicio como *apiary.io*. El componente EDE es simplemente código de Python (lenguaje de programación para la creación de la API) que acepta una solicitud HTTP, extrae parámetros en ella, forma una cadena de consulta SQL, envía esta consulta al motor de base de datos PostgreSQL, este recibe la respuesta de la consulta, crea la respuesta en formato JSON (Javascript Object Notation)

y finalmente devuelve ese JSON envuelto en un respuesta HTTP al componente que emitió la solicitud HTTP, el cual puede ser un cliente directamente (en la configuración de prueba) o uWSGI (en la configuración de producción) que luego la envía de vuelta al cliente. El cliente puede emitir solicitudes HTTP a través de JavaScript en el navegador, que es lo que hace la página web de *ATLAS*, o directamente a través de la línea de comando. En este punto debemos enfatizar que tanto la API de EDE como los esquemas de Postgres presentados en el siguiente capítulo fueron diseñados bajo el supuesto de que las variables de NetCDF con las que trabajamos son de la forma *var(time; lat; lon)* donde cada dimensión *time, lat , lon* tiene una variable de coordenadas correspondiente *time(time), lat(lat), lon(lon)*. Esta restricción se eliminará cuando discutamos una API HTTP más general junto con un esquema de base de datos más general. Además de nuestros datos principales, que son datos ráster provenientes de archivos NetCDF, también almacenamos algunas regiones geográficas, que son datos vectoriales provenientes de archivos. La razón por la que también almacenamos estas regiones es que los usuarios a menudo tienden a agregar datos ráster NetCDF sobre estas, principalmente porque representan límites administrativos como países, estados, condados, etc.

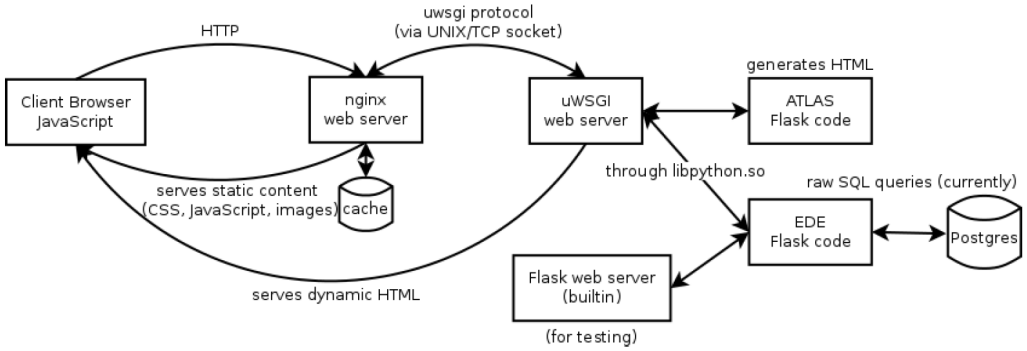

Figura 3.3: Ilustración general del prototipo. (Thaler, 2016)

3.2.2. Almacenar raster en Postgres

PostgreSQL con su extensión PostGIS es una opción común para almacenar datos ráster ya que PostGIS proporciona un tipo de ráster. Sin embargo, hay libertad para elegir el esquema de la base de datos. Podemos considerar los intervalos de tiempo en los datos ráster como bandas en el tipo de ráster PostGIS. No obstante, estas bandas tienden a usarse no para intervalos de tiempo sino para diferentes çomponentes espectrales"(por ejemplo, RGB), además, el número máximo de bandas está codificado como 65536 = sizeof(uint16_t) que admite los conjuntos de datos con los que hemos estado trabajando hasta ahora (el que tenía la mayor cantidad de pasos de tiempo fue el conjunto de datos de AgMERRA con 11323 pasos de tiempo), aunque es posible que ya no lo haga en el futuro a medida que se agregan conjuntos de datos que tienen resoluciones (digamos una medida cada hora durante los últimos 20 años, lo que significaría: 20x365x24 = 175200 que ya superaría el número máximo de bandas). Es por esto, de que probablemente tenga más sentido para usar un .[en]foque no apilado.[en] el que solo almacenamos un paso de tiempo en un solo campo

de Postgres de tipo ráster. Cuando usamos el tipo de ráster PostGIS, independientemente de si usamos el enfoque ápiladoó ño apilado', hay otra elección que podemos hacer, el tamaño de un mosaico. Podemos optar por colocar toda la extensión espacial de un ráster en un solo campo de ráster de PostGIS o en varios, cada uno de los cuales cubre un subraster/mosaico de cierto ancho y alto que podemos elegir libremente (el impacto y la optimización de este tamaño de mosaico tendrían que ser estudiado). No hay obligación a usar el tipo de ráster PostGIS, en su lugar, puede usar tipos de Postgres más fundamentales, como matrices para almacenar la serie de tiempo en un píxel fijo en un ráster. A continuación, exploramos estos diferentes diseños de datos ráster dentro de Postgres.

3.2.3. Extensión de PostGIS

En esta sección describiremos brevemente la extensión geoespacial Post-GIS de PostgreSQL. Proporciona tipos de geometría como PUNTO, LINESTRING, POLYGON y también un tipo de ráster, junto con funciones para realizar operaciones espaciales en estos tipos. La Figura a continuación muestra un diagrama UML simplificado de las estructuras ráster y de banda de PostGIS que aparecen en el código fuente, omitimos algunos miembros de la estructura como skewX, skewY. La figura a continuación es una ilustración de lo que representan estas estructuras.

El tipo de ráster de PostGIS almacena una cuadrícula regular con un número de celdas de ancho x alto de longitud escalaX x escalaY donde la esquina inferior izquierda de la cuadrícula tiene coordenadas (ipX,ipY). Asociado a cada punto medio de celda (también llamado píxel) hay un valor de datos (de tipo pixtype donde pixtype es un booleano, entero con signo/sin signo de varios tamaños, flotante de varios tamaños). Estos valores se almacenan en una matriz lineal. El identificador del sistema de referencia espacial (SRID) indica el mapeo de posiciones de píxeles a posiciones en el globo. Una trama puede tener cero o más bandas y todas las bandas tienen el mismo ancho, alto, pero pueden tener un tipo de píxel diferente.

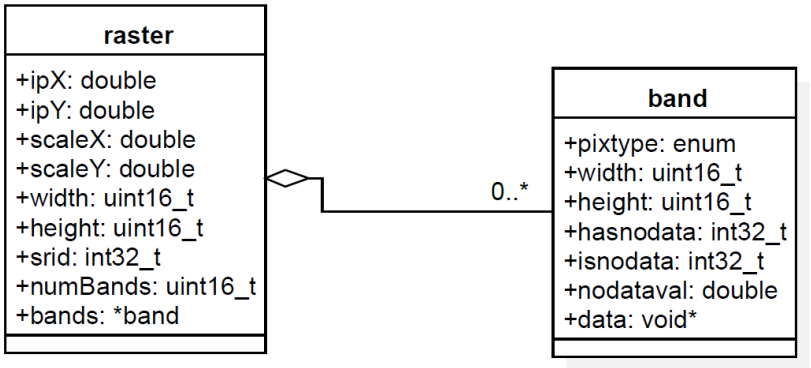

Figura 3.4: Diagrama de tipo de raster PostGIS. (Thaler, 2016)

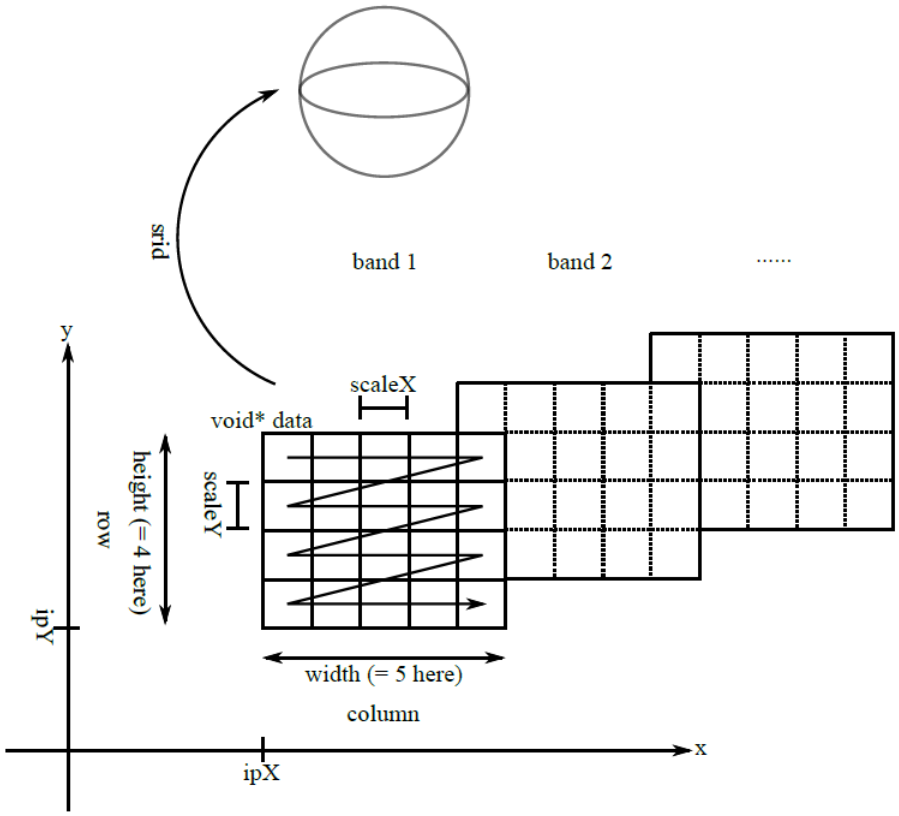

Figura 3.5: Ilustracion del tipo de raster PostGIS. (Thaler, 2016)

3.2.4. Sin utilizar el tipo de ráster PostGIS.

Como se mencionó anteriormente, no estamos obligados a usar el tipo de ráster PostGIS, en su lugar, podemos almacenar los píxeles de un ráster individualmente. Aunque este esquema ocupará más espacio, estará más cerca del formato de retorno JSON, más precisamente, requiere menos procesamiento para obtener ese formato JSON. Por lo tanto, estamos intercambiando más espacio por menos tiempo de respuesta. Sin embargo, esto solo funcionará si los índices pueden mantenerse actualizados a medida que se consumen más conjuntos de datos, lo que a su vez requerirá más recursos, verticalmente y/o horizontalmente. Las dos entidades más importantes son *raster_data_single* y *raster_data_series*. Elegimos estas dos entidades para tener una posiblemente más eficiente para consultas que solicitan datos para un solo período de tiempo, *raster_data_single*, y la otra potencialmente más eficiente para consultas que solicitan datos para múltiples períodos de tiempo a la vez, *raster_data_series*, aunque esa afirmación de eficiencia debe verificarse. Lo fundamental es que una fila en *raster_data_single* almacena el valor en un solo píxel y un solo período de tiempo, mientras que una fila en *raster_data_series* almacena los valores en un solo píxel, para todos los períodos de tiempo. Sin embargo, esta argumentación no es del todo convincente, ya que si consultamos *raster_data_single* para un solo paso de

tiempo, dependemos en gran medida de los índices, ya que esa entidad tiene muchas más filas, especialmente si confiáramos en un índice en la columna *time_id*, que no está agrupada, por ende, generaría una sobrecarga de lectura de disco.

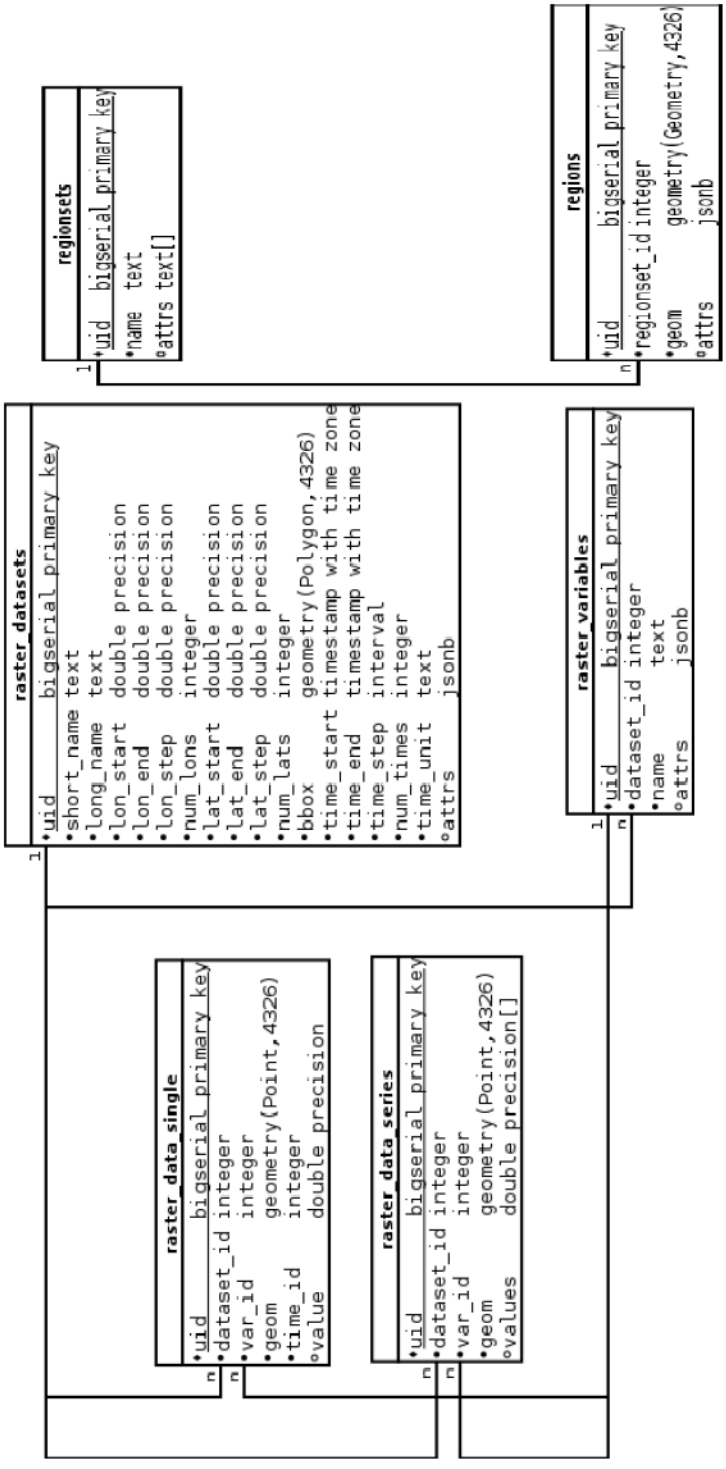

Figura 3.6: Esquema sin PostGIS. (Thaler, 2016)

3.2.5. El enfoque desapilado

Un enfoque para almacenar datos ráster en Postgres es usar el tipo de ráster proporcionado por la extensión PostGIS. La ventaja de este enfoque es que es una representación muy compacta que requiere mucho menos espacio que el enfoque anterior. Además, las operaciones de inserción espacial y temporal se pueden realizar de manera más eficiente en este formato. Sin embargo, esto tiene un precio. Ahora que ya no almacenamos las posiciones de píxeles de un ráster individualmente, sino solo los valores en estas posiciones en el tipo de ráster compacto de PostGIS necesitamos un procesamiento adicional para reconstruir estas posiciones de píxeles para la respuesta JSON. Estamos intercambiando menos espacio por más tiempo de procesamiento. Para el tipo de ráster de PostGIS se compone de varias bandas. Por otro lado, en este esquema almacenamos solo una banda en un campo ráster, que contiene los datos durante un tiempo específico. Sin embargo, lo que hacemos es colocar en mosaico la extensión completa de un ráster de NetCDF en varios campos de ráster de PostGIS. Colocar en mosaico un ráster solo significa dividir toda la extensión del ráster, que es solo un rectángulo, en subrectángulos (estos mosaicos no se superponen y además tienen la misma altura y anchura). Por lo tanto, cada campo 'rast'de PostGIS en este esquema contiene, para un conjunto de datos fijo (*dataset_id*), para una variable fija dentro de ese conjunto de datos (*var_id*), para un tiempo fijo (*time_id*) de esa variable (dependiente del tiempo), todos los valores, pero solo dentro de un mosaico de toda la cobertura espacial de esa variable. El tamaño de las teselas (es decir, el ancho y la altura expresados en número de píxeles) se decide en el momento de la ingesta, aunque se pueden volver a teselar dentro de la base de datos más adelante para obtener un tamaño de tesela óptimo.

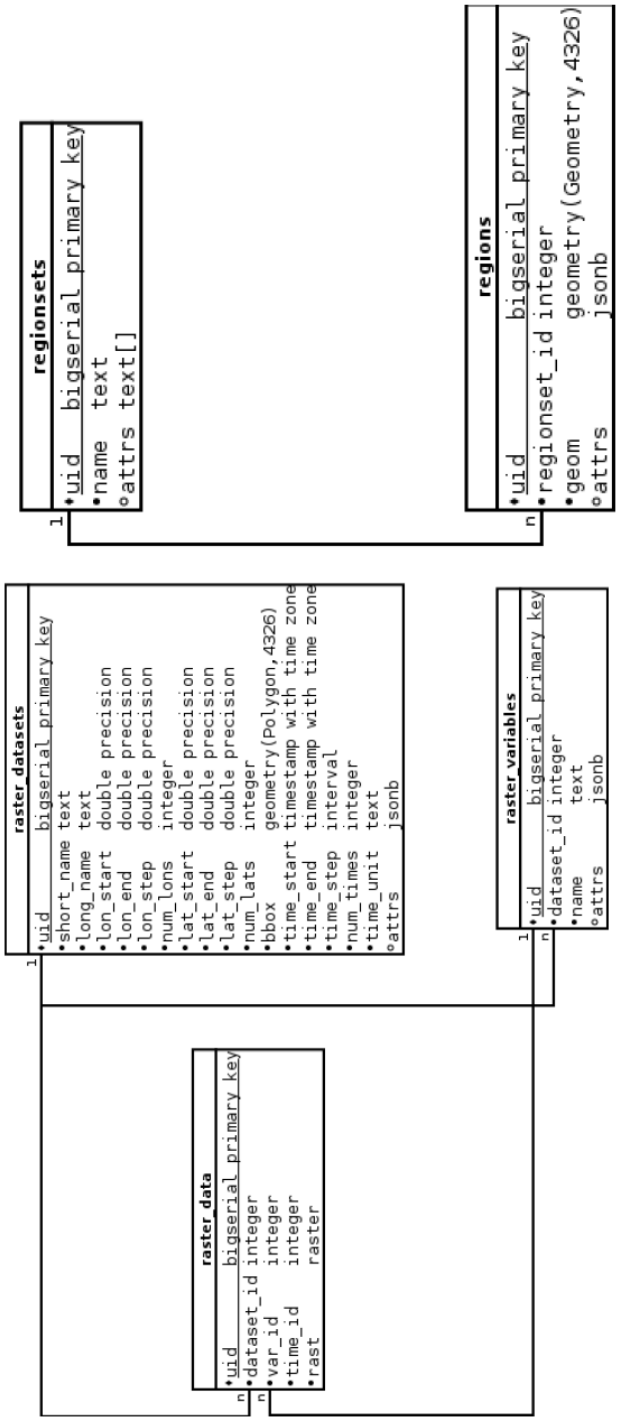

Figura 3.7: Esquema Desapilado. (Thaler, 2016)

3.2.6. Una API HTTP más general

Además de un esquema de base de datos más general, también necesitamos una API HTTP más general implementada en el componente de código EDE Flask (Python). Esta API se especifica en *ede_rest_api* esta vez no como archivos API Blueprint sino solo como archivos JSON, señalando los formatos de solicitud y respuesta. A continuación, veremos brevemente los puntos principales de estos formatos.

Obtener los metadatos de un conjunto de datos

Formato de respuesta: response_dataset.json

GET /rastermeta/datasets/dataset_id

Retorna los metadatos de un conjunto de datos específico. Si no se especifica ningún conjunto de datos, se devuelven los metadatos de todos los conjuntos de datos actualmente presentes en el backend. Los metadatos devueltos de un conjunto de datos contienen su uid, atributos globales, dimensiones y variables.

Obtener los metadatos de una variable

Formato de respuesta: response_variable.json

GET /rastermeta/variables/variable_id

Retorna los metadatos de una variable específica. Si no se especifica ninguna variable, se devuelven los metadatos de todas las variables presentes actualmente en el backend. Los metadatos devueltos de una variable en particular contienen su uid, el conjunto de datos al que pertenece, su tipo y, si es una variable de datos, las variables de eje de las que depende.

Obtener de los datos de una variable

Formato de solicitud: request_select.json

Formato de respuesta: response_select.json

POST /rasterdata/vars/variable_id

En la solicitud JSON, un usuario puede especificar filtros en las variables del eje de las que depende una variable de datos, es decir, si una variable de datos depende de T;Z; Y;X entonces el usuario puede especificar un filtro en T, uno en Z y uno en Y;X combinados.

Agregar los datos de una variable

Formato de solicitud: request_aggregate.json

Formato de respuesta: response_aggregate.json

POST /agregado/vars/variable_id

Los JSON de solicitud y respuesta para esta ruta son muy similares a los JSON de solicitud y respuesta para la ruta de obtener datos, la principal diferencia es que para esta ruta de agregación también se especifica el eje sobre el cual agregar y el tipo de operación de agregación que se debe realizar.

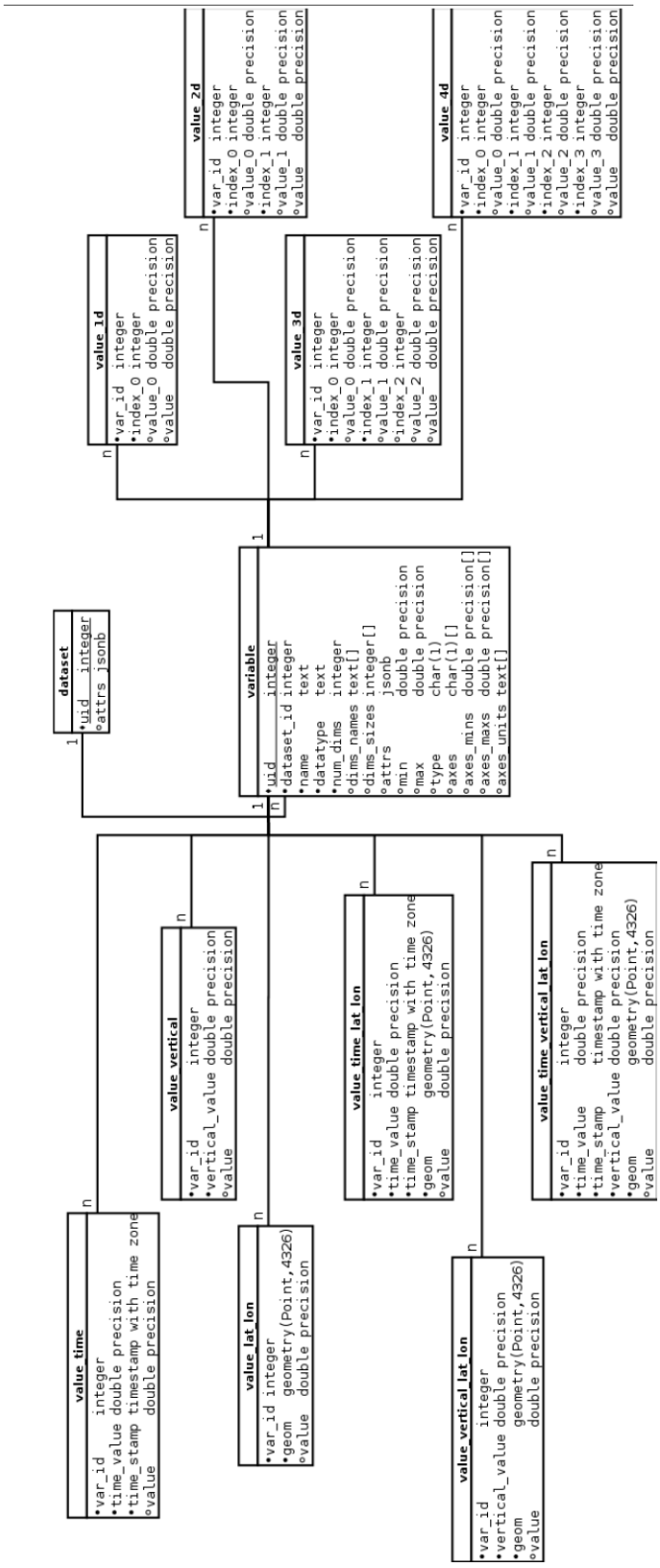

Figura 3.8: Esquema más General. (Thaler, 2016)

Capítulo 4

Caso de estudio

4.1. Problemática hidrica actual

4.1.1. Crisis hidrica Chile 2022

La gran sequia que afecta a nuestro país que ya lleva 13 años afectando a distintas regiones de nuestro territorio nacional no da tregua. Ciudades tan importantes como Santiago con un déficit de caudales cercano al 90 %, Punta Arenas cerca del 40 % y Concepción sobre el 50 % de déficit, dejan entre ver una cruda realidad que no hace más que empeorar. Dado lo anterior y sumado a otros factores es la causa que ya se esté hablando de un racionamiento por parte de las autoridades nacionales al menos por el momento para la Región Metropolitana. Sumado a lo dicho anteriormente este año 2022 estamos siendo afectados por el comúnmente conocido "fenómeno de la niña "que indica precipitaciones bajo los niveles normales para la época. Según el pronostico nacional para mam 2022(marzo, abril, mayo), se prevé un déficit en los niveles normales para la época desde el centro de la región de coquimbo hasta las cercanías de la región de los lagos.

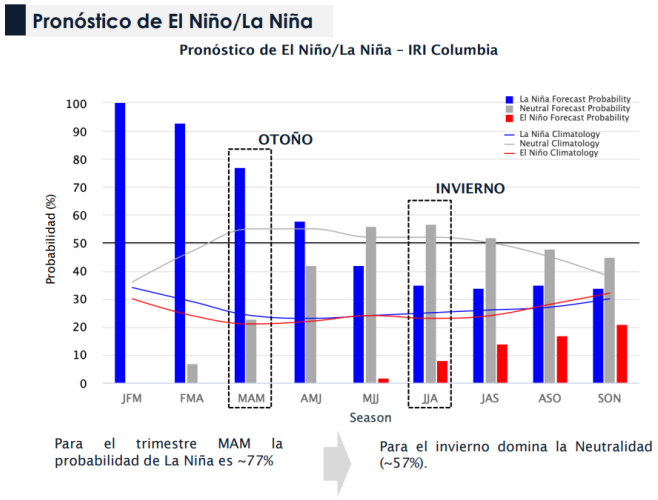

Figura 4.1: Pronóstico de El Niño/La Niña MAM 2022

En base a lo anterior se hace imperativo contar con las herramientas necesarias a fin de entender por medio de indicadores claros y prácticos como el índice de riesgo hídrico afectara a las distintas zonas de la región de ñuble.(de Servicios Climáticos Sección Climatología Dirección Meteorológica de Chile, 2022)

4.1.2. Crisis hidrica en la región de Ñuble

Según las Naciones unidas, Chile es uno de los 10 países más vulnerables al cambio climático. Además, según indicó el Centro de Ciencia del clima y la resiliencia (CR2), el cambio climático eventualmente modificará el régimen de precipitaciones en la zona central de Chile, aumentando la sequía lo que podría generar impacto en la economía del país.

La región de Ñuble no está exenta de esta problemática. Su territorio abarca 13.178 kilómetros cuadrados y una población cercana a los 500.000 habitantes, los cuales se estima que aproximadamente 30

Nuestro objetivo para dar solución a dicha problemática planteada anteriormente es realizar una aplicación de propósito general que permita extraer la información de los achivos NetCDF e interpretarlos para lograr una visualización completa de las zonas con mayor riesgo hídrico. Algunos ejemplos de aplicaciones científicas que permiten la carga y visualización de archivos NetCDF es “Panoply” desarrollada por la NASA, que es un software completo con varias funcionalidades. También existen sistemas de información geográficos (GIS), como ArcGIS, QGIS o GRIDFOUR, que son software más avanzados y que requieren un conocimiento previo en lenguajes de scripting para poder realizar ciertas funcionalidades. Esto se aleja de nuestro objetivo ya que nuestra idea es desarrollar una aplicación simple, que cualquier persona pueda utilizarla y de propósito específico. (Alfonso, 2021)

Capítulo 5

Conclusiones

En el desarrollo investigativo de esta tesis, nos enfocamos en la misión de lograr recolectar la información necesaria para poder cimentar las bases de conocimiento sobre las distintas aristas que componen el fascinante mundo de los archivos multidimensionales. Una vez logramos este primer enfoque nos encaminamos en cómo funcionan los ficheros NetCDF y la importancia de estos en algunas ramas como las ciencias de la climatología entre otras varias. La búsqueda de información sobre ficheros NetCDF nos ayudó a profundizar en áreas como el tratamiento de estos archivos y la importancia de la eficiencia cuando se manipulan ficheros de datos de gran volumen y envergadura. La profundización en el tratamiento de archivos NetCDF al día de hoy supone un desafío complejo de abordar dada la gran cantidad de herramientas y librerías en las cuales algunas tecnologías de desarrollo se sustentan para el trabajo con este tipo de ficheros, es por ello que nos embarcamos en la tarea de buscar información sobre las alternativas que mejor se ajustaran en el tratamiento eficiente de los archivos NetCDF.

Dado lo anterior y en base a la información recolectada sobre el funcionamiento de los archivos NetCDF y la búsqueda de las distintas alternativas de librerías para el trabajo con este tipo de ficheros es que nos quedara la futura labor de realizar una comparativa de las librerías recolectadas para poder entender el comportamiento de los ficheros NetCDF en distintos entornos de trabajo y con ello lograr concluir cual es la mejor alternativa de rendimiento en el tratamiento y visualización eficiente de estos archivos.

5.0.1. Trabajo Futuro

Dentro de las principales tareas pendientes se encuentran:

- Realizar una comparativa de rendimiento de las librerias recolectadas para poder concluir cual es la mejor alternativa de rendimiento en el tratamiento y visualizacion eficiente de ficheros NetCDF.
- Desarrollar un prototipo para la lectura de un fichero NetCDF.

Referencias

Soto Duran Marin Morales Maria Isabel Vargas Agudelo Fabio Alberto. Caracterizacion de formatos de almacenamiento, transporte y visualización de datos geograficos. 2014.

Pablo Gonzales Clemente Rubio Jose Antonio Olivares Elias Alfonso. Hacia la definicion de un indice de riesgo hidrico y su calculo automático a partir de archivos de datos multidimensionales netcdf. 2021.

Alexander Barth. Librería netcdf para julia. 2022.

UCAR Corporación Universitaria para la investigación Atmosférica. Librería netcdf para r. 2010.

Oficina de Servicios Climáticos Sección Climatología Dirección Meteorológica de Chile. Monitoreo de el niño/la niña y pronóstico estacional. 2022.

Apache Software Foundation. System architecture apache sedona. 2021.

Apache Software Foundation. Spark overview. 2022.

Peter Gleeson. Which languages should you learn for data science? *FreeCodeCamp*, 2017.

Unidata Program. Netcdf user's guide an interface for data access. 1991.

NASA Scientific Spark – a NASA AIST14 project. Architecture of scispark. 2018.

Data Services y UNIDATA Tools for Geoscience. Librería netcdf para python. 2020.

Data Services y UNIDATA Tools for Geoscience. Librería netcdf para java. 2022.

A.M. Sohrabinia, M. Khorshiddoust. Application of satellite data and gis in studying air pollutants in tehran. habitat international. 2012.

Severin Thaler. Almacenar archivos netcdf en una base de datos relacional. 2016.

España Universidad de Alacalá. Lenguajes para análisis de datos. 2022.

Ahmed A. Chassignet E.P. Zavala J. Fernández A. Meyer Zavala, O. An open source java web application to build self-contained web gis sites. environmental modelling software. 2013.