Facultad de Ciencias

Evaluación de las tecnologías object storage para el almacenamiento y análisis de datos climáticos

(Evaluation of object storage technologies for climate data storage and analysis)

Trabajo de Fin de Máster

para acceder al

MÁSTER EN DATA SCIENCE/CIENCIA DE DATOS

Autor: Ezequiel Cimadevilla Álvarez

Director: Dr. Antonio S. Cofiño González

Codirector: Aida Palacio Hoz

Julio - 2019

Agradecimientos

La realización de este TFM supone el fin de un pequeño objetivo autoimpuesto, hace más de dos años, de cursar un máster como el Máster en Data Science. Aquel objetivo trajo consigo un nuevo trabajo y una nueva vida en Santander, ambos plenamente satisfactorios. Todo ello no sería posible sin que Sixto me hubiera invitado a echar un CV en aquella época, por lo que es la primera persona a la que me gustaría mostrar mi agradecimiento en estas líneas.

Quiero agradecer a Antonio el esfuerzo y los recursos que pone a mi disposición con la esperanza de que haga un buen uso de ellos, ya que me permiten conocer gente nueva y ampliar mis horizontes. Sirva de ejemplo este mismo trabajo, cuya motivación surge en el F2F del ESGF de 2018. Agradecerle también el tiempo dedicado a este TFM y todos los comentarios y discusiones aportadas.

Quiero agradecer a Aida la supervisión realizada tanto en el periodo de prácticas como en el desarrollo del TFM. Sus comentarios fueron muy útiles durante la redacción del TFM y sus preguntas me sirvieron de guía durante el desarrollo del mismo.

Quiero agradecer a los profesores del máster el trabajo que realizan preparando las clases y corrigiendo las tareas. Hago una mención especial a Lara, a la que me consta que todos los alumnos recurrimos cuando tenemos la menor duda.

También me gustaría agradecer a la Universidad de Cantabria la apertura de aulas de estudio durante la época de exámenes. No conozco ningún otro lugar en el que pueda mantener la concentración durante los fines de semana.

Finalmente pero no menos importante, agradecer a mi familia el apoyo mostrado desde que me fui de Asturias, que ha sido enorme.

Abstract

Data analytics in earth science have been dominated by the download-analyze model, in which data analysts first download the desired dataset from a remote server to it's local workstation or HPC infrastructure, in order to perform the desired analysis. Over time, the size and variety of datasets have increased exponentially and new data science methodologies have appeared, along with new requirements in how datasets are stored and analyzed. In the climate community, climate data is usually stored as netCDF, which has incorporated, new functionalities such as HDF5 storage and *chunking*, that allows netCDF files to be accessed in parallel by parallel file systems. Data access has also been improved by protocols like the DAP, which allows to access only the required subset from a remote dataset. In recent years, cloud computing and more specifically *object storage*, have appeared as an alternative to store climate data and to perform data analysis. This fact has encouraged the development of new storage specifications and libraries, such as Zarr. *Object storage* works by assigning a string (hash id) to an arbitrary block of bytes (blob), combined with REST APIs. The purpose of this work is to compare these new technologies with the traditional stack both for data analysis and data storage.

Resumen

El análisis de datos en ciencias de la tierra ha estado dominado por el modelo descargar-analizar, por el cual un científico primero descarga el dataset, desde un servidor remoto, a su estación de trabajo o infraestructura HPC de su institución y después procede a su análisis. Con el paso del tiempo, el tamaño y variedad de los datasets ha aumentado de forma exponencial y, a su vez,se han introducido nuevas técnicas de análisis de datos. Estos cambios han introducido nuevos requisitos en los sistemas que almacenan los datasets y en las herramientas de análisis. En la comunidad científica del clima, el formato dominante para los datasets es netCDF, que con el paso del tiempo ha incorporado nuevas funcionalidades para permitir un almacenamiento y acceso a los datos de forma más eficiente, como el uso del formato HDF5 y su técnica de *chunking*, que permite el uso de sistemas de ficheros en paralelo. El acceso a datos también se ha visto beneficiado de protocolos que permiten el acceso a un subconjunto de los datasets, como por ejemplo DAP. En los últimos años, el *cloud computing* y en concreto el *object storage*, se han presentado como una alternativa tanto para el almacenamiento como para el análisis de datos, por lo que están propiciando la aparición de nuevas especificaciones de almacenamiento y de acceso a los datasets, como por ejemplo Zarr. El *object storage* permite asignar un identificador alfanumérico (*hash id*) a un bloque arbitrario de bytes (*blob*) combinado con APIs de tipo REST. El objetivo del trabajo consiste en la evaluación de los beneficios y la eficiencia de estas nuevas tecnologías y especificaciones respecto a las ya existentes, tanto para el almacenamiento como el acceso de datos para su análisis.

Contenido

1 Introducción....1
1.1 Motivación....1
1.2 Objetivo....2
1.3 Contexto....2
1.4 Estructura de la memoria....3
2 Estado del arte....5
2.1 Network Common Data Format....5
2.2 Sistemas de ficheros en paralelo....7
2.3 Tecnologías y protocolos de acceso remoto....9
2.4 Repositorios de datos....12
2.5 Herramientas para el análisis de datos climáticos....13
2.6 Nuevas tecnologías para el análisis de datos....15
2.7 Almacenamiento cloud....18
2.8 Resumen....19
3 Almacenamiento basado en object storage....21
3.1 Introducción....21
3.2 Ceph....22
3.3 Acceso a datos climáticos en object storage....23
3.4 Formatos nativos para object storage....24
4 Evaluación de acceso a datos climáticos en cloud y HPC....27
4.1 Instalación del entorno cloud....27
4.2 Instalación del entorno HPC....28
4.3 Test de acceso local en OpenStack....28
4.4 Test de acceso remoto en OpenStack....29
4.5 Test de acceso local en Altamira....30
5 Conclusiones y trabajo futuro....34
6 Referencias....37

Figuras

Figura 1: Modelo de datos implementado por la librería netCDF. Fuente: [12].....5

Figura 2: Arquitectura de la librería netCDF-C. Fuente: [12].....6

Figura 3: Arquitectura de un sistema de almacenamiento Lustre. Fuente: [58].....8

Figura 4: Ejemplo de petición DAP remota usando constraint expressions.....10

Figura 5: Ejemplo de catálogo TDS mostrado en un navegador web.....11

Figura 6: Arquitectura de la ESGF. Fuente: [23].....12

Figura 7: Arquitectura de la librería Climate4R. Fuente: [32].....15

Figura 8: Ejemplo de un dataset multidimensional. Fuente: [59].....15

Figura 9: Componentes de JupyterHub. Fuente: [35].....17

Figura 10: Componentes básicos de Dask. Fuente: [37].....18

Figura 11: Interfaces y componentes de Ceph. Fuente: [46].....23

Figura 12: Capas Virtual Object Layer y Virtual File Layer de HDF5. Fuente: [57, p. 5].....24

Figura 13: En C se muestra el acceso al servicio HSDS. Fuente: [60].....25

Figura 14: Arquitectura del test de acceso local. El test consiste en acceder al sistema de ficheros usando h5netcdf, netCDF4-Python y Zarr.....29

Figura 15: Arquitectura del test de acceso remoto. Las librerías acceden a distintos tipos de almacenamiento con el objetivo de mostras sus capacidades.....30

Figura 16: Arquitectura del test HPC. El trabajo será llevado a cabo de forma paralela por los nodos MPI.....31

Tablas

Tabla 1: Tiempos de acceso al sistema de ficheros ext4 de forma local.....29

Tabla 2: Tiempos de acceso a los distintos tipos de almacenamiento en función de la librería.……30

Tabla 3: Tiempos de acceso de las librerías en función del número de tareas MPI.....31

Glosario

CDS	Climate Data Store
CMIP	Coupled Model Intercomparision Project
DAP	Data Access Protocol
ESGF	Earth System Grid Federation
ESM	Earth System Model
GPFS	General Parallel File System
HDF5	Hierarchical Data Format Version 5
HPC	High Perfomance Computing
MPI	Message Passing Interface
netCDF	Network Common Data Format
POSIX	Portable Operating System Interface
REST	Representational State Transfer
SMG	SantanderMetGroup
TDS	THREDDS Data Server

1 Introducción

1.1 Motivación

Las ciencias de la Tierra, dedicadas al estudio de los sistemas oceánicos, climáticos y geoquímicos [1], están experimentando un cambio sin precedentes debido al efecto invernadero de carácter antropogénico, ya que perturba el equilibrio energético a nivel mundial. Tanto el entendimiento y la predicción de estos fenómenos, como su impacto para el bienestar humano, es a la vez un urgente problema social y un desafío científico.

Los modelos del sistema terrestre (ESM, Earth System Model) o sistemas climáticos, son modelos numéricos que modelan la interacción entre el océano, la atmósfera, la tierra, la criosfera y la biosfera. Los ESM son la herramienta indispensable sobre la que se realiza el estudio en las ciencias de la Tierra, ya que los datos generados por dichos modelos actúa como un laboratorio virtual sobre el que realizar los experimentos [2].

Los ESM consisten en aplicaciones que se ejecutan en infraestructuras de computación de altas prestaciones (HPC [3]). Con el paso del tiempo, estas infraestructuras han ido incorporando todas las mejoras tecnológicas que se producían en HPC, como la paralelización masiva de su arquitectura. Debido a estas mejoras en los ESM, estos han ganado mayor capacidad de modelización, incrementando su resolución tanto espacial como temporal, hasta tal punto que en la actualidad, la salida de estos modelos, consiste en datasets que ocupan almacenamiento en el orden del petabyte.

El flujo de trabajo tradicional para realizar análisis de datos a partir de la salida de los ESM, consiste en realizar una descarga o copia local del dataset a utilizar en el experimento, para posteriormente procesarlo con herramientas como MATLAB o software similar. Este modo de trabajo ya no es deseable en el nuevo paradigma, en el que el tamaño de los datasets se ha incrementado de manera exponencial. El almacenamiento de estos nuevos datasets también se ha convertido en un problema de gran importancia para las instituciones que hasta ahora almacenaban la salida de los ESM, por lo que nuevos paradigmas de almacenamiento, como el *cloud storage*, han pasado a formar parte como una nueva opción de almacenamiento para estas instituciones [4].

A modo de ilustración, la iniciativa CMIP [4], en su tercera etapa, conocida como CMIP3, generó 36 Terabytes de datos, mientras que su continuación, CMIP5, generó 3,3 Petabytes. La sexta fase de esta iniciativa, CMIP6, ya está en marcha y aunque la cantidad de datos generada aún no ha sido estimada, se calcula que suponga otro orden de magnitud respecto a la fase anterior. Otras iniciativas, como ERA5 [5, p. 5], producen datasets del orden de los 10 Petabytes y van a generar en tiempo real nuevos datasets del orden de 11 Terabytes cada mes [6, p. 5].

Este gran incremento de la información abre una nueva crisis respecto a las tecnologías y herramientas necesarias para llevar a cabo los estudios en las ciencias de la Tierra. Esta crisis se conoce de forma popular como el fenómeno *Big Data*, siendo este quizá un término más usado en el mundo empresarial que en el mundo científico.

La manera de afrontar la problemática planteada consiste en mover las herramientas y metodologías de análisis de datos a entornos compartidos, HPC o *cloud*, que sean capaces

de realizar tareas en paralelo, almacenar las ingentes cantidades de datos y proveer de herramientas de alto rendimiento para el análisis.

1.2 Objetivo

El principal objetivo de este TFM es proporcionar una contribución al estado del arte sobre cómo los nuevos avances tecnológicos y metodológicos están irrumpiendo en las ciencias del clima. Uno de los lugares donde estos avances están teniendo lugar es en los sistemas de almacenamiento, donde el almacenamiento de tipo *object storage* [7] aparece como una opción más escalable que los sistemas de ficheros tradicionales [8].

Al tratarse de una transformación en el sistema de almacenamiento, es decir, una de las capas más bajas en la pila tecnológica, este cambio afecta a la mayoría de las aplicaciones que hasta ahora estaban construidas asumiendo que el sistema de almacenamiento es un sistema de ficheros.

A lo largo del TFM se presentará una visión de cuáles han sido las metodologías y herramientas tradicionales, además de cómo se han visto afectadas por el fenómeno *Big Data* y cuáles son las posibilidades de adaptación a este nuevo entorno. Además, se realizarán experimentos que evalúan el rendimiento del almacenamiento en *object storage* frente al sistema de ficheros.

Los análisis, evaluaciones y conclusiones de este TFM se han llevado a cabo tomando como perspectiva el análisis de datos, es decir, asumiendo que los experimentos son llevados a cabo a partir de la salida de los ESM. Otras perspectivas, como las necesidades de almacenamiento de los ESM, solo son valoradas en la medida en que influyen en el posterior análisis de datos.

Queda fuera del alcance de este TFM realizar un análisis de costes monetarios entre los distintos sistemas de almacenamiento y acceso a datos. Las instituciones están preocupadas por el coste que puede suponer un modelo basado en *cloud* frente al tradicional almacenamiento en HPC. Las evaluaciones realizadas en este TFM se han llevado a cabo en la infraestructura del Instituto de Física de Cantabria (IFCA) y del Grupo de Meteorología de Santander (SMG, SantanderMetGroup).

1.3 Contexto

Este TFM se desarrolla como complemento del trabajo que el Autor realiza en la Universidad de Cantabria como parte del SMG. El grupo está compuesto por miembros de la Universidad de Cantabria (UC), del Instituto de Física de Cantabria (IFCA) y del Consejo Superior de Investigaciones Científicas (CSIC) que trabajan en áreas relacionadas con la predicción meteorológica, investigación climática, minería de datos y computación de altas prestaciones. El interés en la infraestructura cloud del IFCA por parte del SMG es previo al desarrollo de este TFM y es posible que este trabajo sirva como nuevo caso de uso en las relaciones entre ambos.

Además, el desarrollo de este TFM está ampliamente inspirado en el trabajo llevado a cabo por el proyecto Pangeo [9]. Pangeo está financiado por la *National Science Foundation* [10] de los EEUU, dentro del marco *EarthCube* [11]. En un período de tres años, su objetivo es proveer de infraestructura escalable para las ciencias de la Tierra en base a cuatro casos de

uso identificados por el proyecto [2].

Pangeo es una iniciativa con el objetivo de fomentar la colaboración entre la comunidad científica de las ciencias de la Tierra, con el fin de mejorar la escalabilidad de las herramientas y metodologías de trabajo para adaptarlas al nuevo reto del *Big Data*. Pangeo se basa en un ecosistema de software Python y trata de integrar los paquetes de dicho ecosistema para ofrecer un entorno escalable en el que la comunidad científica de las ciencias de la Tierra pueda desarrollar su trabajo.

1.4 Estructura de la memoria

A lo largo del Capítulo 2 se expondrá en qué consisten los datos climáticos y cuales son las metodologías y flujos de trabajo tradicionales para realizar análisis de datos sobre ellos, además de introducir las novedades tecnológicas que dan propósito a este TFM. En el Capítulo 3 se explicará el *object storage* y cuales son los retos que presenta a la hora de elegir este tipo de almacenamiento como persistencia de datos climáticos. En el Capítulo 4 se detallan las evaluaciones de análisis de datos que se han llevado a cabo utilizando las diferentes formas de acceso a datos climáticos. Por último, en el Capítulo 5 se detallan las conclusiones del trabajo así como el trabajo futuro.

2 Estado del arte

Las ciencias del clima han estado fuertemente ligadas al entorno HPC, debido a la complejidad computacional de los ESM y su necesidad de alta capacidad de computación. Por ello, encontramos tecnologías como MPI y sistemas de ficheros en paralelo formando parte de la pila tecnológica de los ESM. El análisis de datos no se realiza, habitualmente, en este tipo de entornos. El análisis de datos llevado a cabo por científicos de las propias instituciones que poseen los recursos HPC sí puede hacer uso de los mismos, ya que normalmente disponen de gestores de colas compartidos a los que se pueden lanzar *jobs* para realizar los experimentos. Sin embargo, los analistas de datos sin acceso a estos recursos recurren a flujos de trabajo en local que no son para nada escalables.

2.1 Network Common Data Format

La librería que se ha establecido como estándar para almacenar la información de los ESM y su posterior análisis es netCDF, desarrollado por Unidata y orientado a arrays de datos multidimensionales, autodescriptivos y portables [12]. NetCDF es a la vez un formato, un modelo de datos y la librería que se usa para trabajar con él. En la Figura 1 se muestra el modelo de datos, compuesto por variables multidimensionales con atributos que pueden ser clasificadas en distintos grupos. NetCDF consiste principalmente en una librería en C [13] pero permite acceder a datasets usando interfaces C++, Fortran y Python. La arquitectura de la librería netCDF-C se muestra en la Figura 2. Además existe una librería implementada completamente en Java para el acceso a netCDF, netCDF-Java [14].

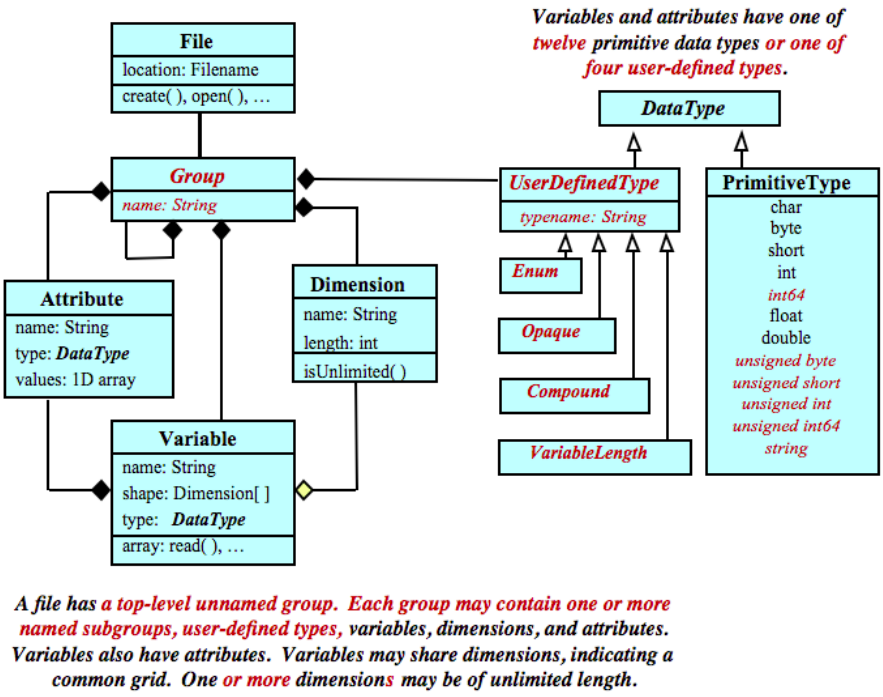

Figura 1: Modelo de datos implementado por la librería netCDF. Fuente: [12]

Es importante distinguir entre dos versiones, aunque existen más, de netCDF. Estas versiones son netCDF classic y netCDF4. En las versiones que pertenecen a netCDF classic, el formato en disco está definido por la propia librería netCDF, mientras que en netCDF4 el formato en disco es HDF5. El cambio hacia HDF5 supone un cambio importante en la librería ya que funcionalidades disponibles en HDF5, como los grupos, arrays de tamaño variable, *chunking* y el acceso en paralelo pasan a estar disponibles en la librería netCDF. Hay que tener en cuenta que la librería netCDF crea ficheros HDF5 completamente válidos [15], si bien es necesario que los ficheros sean creados usando la librería netCDF y no otras, debido a la información adicional que la librería netCDF guarda en el fichero.

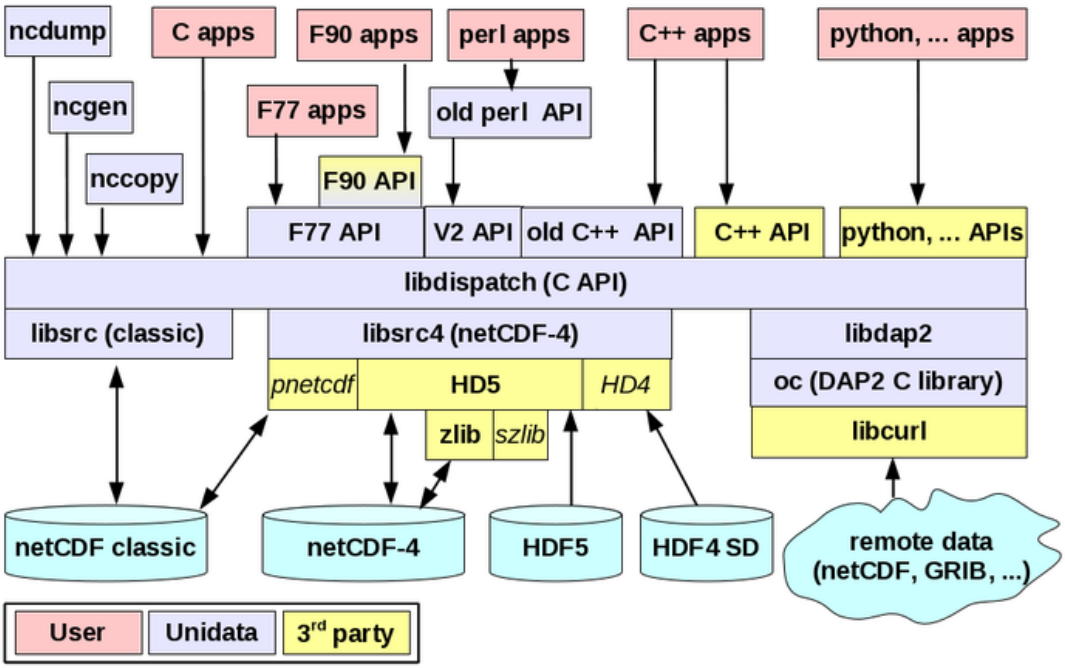

Figura 2: Arquitectura de la librería netCDF-C. Fuente: [12]

Un aspecto realmente importante a la hora de mejorar el rendimiento del acceso a datos netCDF es el *chunking*. El *chunking* consiste en almacenar los datos en pequeños contenedores denominados *chunks*. Cada *chunk* contiene información de las distintas dimensiones que conforman un array multidimensional, de forma que ciertos patrones de acceso ven mejorado el tiempo de respuesta de la librería a la hora de acceder a los datos. Una última característica de la librería netCDF es que ofrece, como cliente, acceso remoto mediante el protocolo Data Access Protocol (DAP).

Hierarchical Data Format Version 5

Hierarchical Data Format Version 5 (HDF5) es un formato de datos autodescriptivo de código abierto que combina metadatos con los propios datos. Entre sus características se encuentran: portabilidad y compatibilidad con varios lenguajes de programación, soporte a largo plazo y múltiples herramientas cliente para la visualización, manipulación y análisis de datos.

Debido a las optimizaciones que provee, se puede acceder de forma altamente eficiente a grandes y complejas cantidades de datos. Sin embargo, debido al crecimiento en la cantidad de información, el cuello de botella que limita el rendimiento computacional no son

los datos en sí mismos, sino el rendimiento de las tecnologías de almacenamiento [16]. El modelo de datos de HDF5 está compuesto por los siguientes objetos [17]:

- File - Son los bytes almacenados en el sistema de almacenamiento. Representan uno o más objetos del modelos de datos.
- Group - Colección de objetos.
- Dataset - Array multidimensional de elementos con metadatos y atributos.
- Datatype - Especificación de un elemento concreto en la que se incluye su almacenamiento en disco.
- Dataspace - Descripción de las dimensiones de un array multidimensional.
- Attribute - Un valor con identificado por nombre que está asociado a un group, dataset o datatype.
- Property List - Colección de parámetros que controlan las opciones de la librería. Algunas son persistentes junto con el objeto al que pertenecen y otras son solo transitorias durante el acceso a los datos.

Convenciones CF

Las convenciones Climate and Forecast (CF Conventions) están diseñadas para promover el procesamiento y la accesibilidad de ficheros creados por la librería netCDF [18]. Están ampliamente aceptadas por la comunidad climática y su función es la de definir metadatos que permitan identificar de forma unívoca los datos que representan las variables de los ficheros netCDF. Esto permite a usuarios de distinto origen identificar qué valores son comparables y facilita el desarrollo de aplicaciones con altas capacidades de extracción y visualización.

2.2 Sistemas de ficheros en paralelo

Los sistemas de almacenamiento HPC consisten en sistemas paralelos de ficheros, como pueden ser Lustre o General Parallel File System (GPFS), compartidos con los nodos de computación que conforman la infraestructura HPC. Estos sistemas están altamente paralelizados ya que incluyen características como el *file striping*, que permite particionar y almacenar un fichero en distintos servidores, incrementando el ancho de banda a la hora de acceder al mismo, además de realizar réplicas de cada partición. Sin embargo también se caracterizan por ser sistemas POSIX que proveen fuertes semánticas de consistencia, limitando su escalabilidad [8].

Los datasets almacenados en estos sistemas de ficheros están disponibles para su análisis a los miembros con acceso a la infraestructura HPC, que pueden aprovechar el rendimiento de los sistemas paralelos de ficheros para realizar el análisis de datos. Sin embargo, usuarios externos a la infraestructura HPC deben no pueden trabajar en estos entornos y deben recurrir a otras formas de trabajo para realizar el análisis de datos.

Lustre

Lustre es un sistema de ficheros POSIX de código abierto y orientado a clusters. El

despliegue de Lustre incluye un management server (MGS) y uno o más sistemas interconectados con red Lustre (LNet).

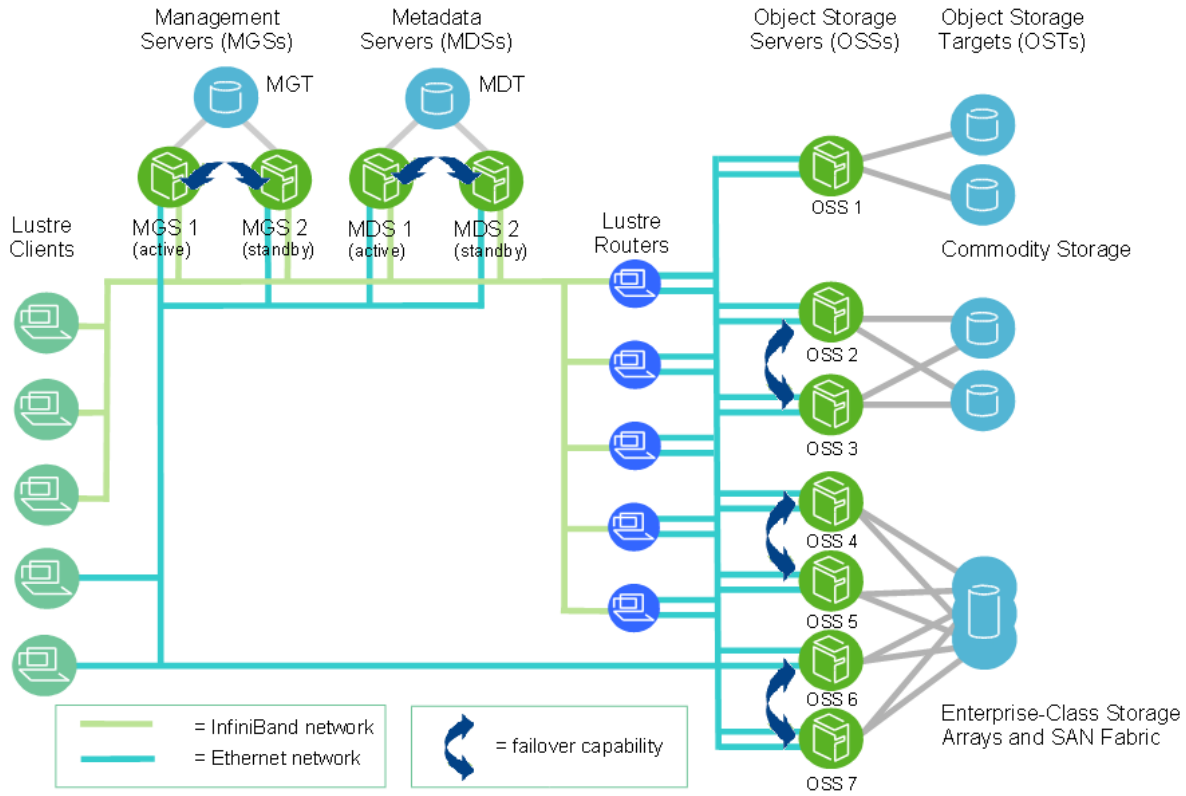

Figura 3: Arquitectura de un sistema de almacenamiento Lustre. Fuente: [58]

El MGS almacena la configuración de todos los sistemas de ficheros en el cluster y provee información al resto de componentes Lustre. Los componentes de Lustre, mostrados en la Figura 3, son:

- Metadata Servers (MDS) - El MDS almacena metadatos en uno o más Metadata Targets (MDT) disponibles para los clientes Lustre. Cada MDS administra los nombres y los directorios en los sistemas de ficheros Lustre y administra las peticiones de red de uno o varios MDT locales.
- Metadata Targets (MDT) - El MDT almacena metadatos (como nombres de ficheros, directorios, permisos y la distribución de ficheros) en almacenamiento agregado al MDS. Cada sistema de ficheros tiene un MDT asociado. Un MDT en almacenamiento compartido puede estar disponible para varios MDS, aunque solo uno puede accederlo a la vez. Si un MDS activo falla, un MDS secundario puede servir al MDT y hacerlo disponible para los clientes.
- Object Storage Servers (OSS) - El OSS se encarga de realizar el servicio I/O y de las peticiones de red a uno o más OST locales. Típicamente, un OSS sirve entre dos y ocho OST, hasta 16TiB cada uno. Una configuración típica es un MDT en un nodo dedicado y dos o más OST en cada nodo OSS.
- Object Storage Target (OST) - Los datos de los usuarios son almacenados en uno o más objetos y cada objeto es almacenado en un OST distinto en un sistema de ficheros Lustre. El número de objetos por fichero es configurable por el usuario y puede ser optimizado para flujos de trabajo determinados.

- Clientes Lustre - Los clientes Lustre son nodos computacionales, de visualización o de escritorio que corren software Lustre, lo que les permite montar el sistema de ficheros Lustre.

General Parallel File System

General Parallel File System (GPFS) es un sistema de ficheros orientado a clusters. Provee acceso concurrente a un único sistema de ficheros o a un conjunto de sistemas de ficheros desde múltiples nodos. Incluye replicación de datos, políticas de administración de datos, alta disponibilidad mediante réplicas y alta capacidad de recuperación de datos.

Cada nodo de un cluster GPFS consta de tres componentes básicos: comandos de administración, extensiones del kernel y un demonio multihilo. Los comandos de administración pueden ser ejecutados desde cualquiera de los nodos y son debidamente redirigidos por GPFS, mientras que las extensiones del kernel permiten que las aplicaciones hagan peticiones al sistema operativo como si estuvieran tratando con un sistema de ficheros genérico. El demonio multihilo se encarga de todas las operaciones de entrada/salida, las cuales son protegidas por GPFS de forma que se asegure la consistencia entre todos los nodos del cluster.

2.3 Tecnologías y protocolos de acceso remoto

Protocolos de transferencia de ficheros

El acceso por FTP es el tipo de acceso remoto más sencillo. La institución que desea publicar los datos generados, en formato netCDF, para que otros científicos puedan realizar análisis de datos sobre ellos, publica el dataset en un servidor FTP o HTTP que la misma institución gestiona. Científicos interesados en analizar los datos navegan a través del servidor y seleccionan de forma manual los datasets que desean, para realizar una descarga de los mismos a su sistema local.

Este sistema funciona cuando los datasets son pequeños tanto en cantidad como en tamaño. Sin embargo, cuando los datasets comienzan a aumentar tanto en número de ficheros como en tamaño, este sistema se vuelve altamente ineficiente. Por una parte es difícil localizar los datasets de interés, debido a la falta de herramientas de catalogación y por otra parte la descarga de datos se complica cuando el tamaño de los datasets excede la capacidad de almacenamiento local.

Data Access Protocol

El Protocolo de Acceso a Datos (DAP) es un protocolo para el acceso a datos organizados en tuplas *name-datatype-value* que permite a clientes acceder de forma remota a datos almacenados en los servidores remotos. El DAP nace en el entorno de la oceanografía y ha sido adoptado por la comunidad de las ciencias del clima, si bien es un protocolo que está diseñado para ser independiente de la disciplina científica [19]. Utiliza HTTP como capa de transporte y es un protocolo sin estado.

El protocolo define una serie de mensajes que las implementaciones, tanto de servidor como de cliente, están obligados a implementar. Las implementaciones pueden definir sus propios mensajes si así lo desean. Los mensajes definidos por el protocolo son:

Request	Response
DDS (Dataset Descriptor Structure)	DDS o error
DAS (Dataset Attribute Structure)	DAS o error
DataDDS	DataDDS o error
Server version	Información sobre la versión en texto plano
Help	Ayuda mostrando todos los pares petición-respuesta

El protocolo define una fuente de datos como un *data source*, que consiste en una colección de variables las cuales pueden tener atributos. Las variables tienen un tipo (*datatype*) asociado, que puede ser un atómico o estructurado. Los tipos atómicos son números enteros, flotantes y cadenas de texto. Los tipos estructurados pueden ser *arrays*, *structures*, *grids* o *sequences*.

El protocolo DAP, su modelo de datos y los tipos de datos están bien definidos en su especificación [19], por lo que no serán descritos en profundidad en este documento. Sin embargo, es importante destacar que características del protocolo son relevantes para el desarrollo del TFM.

La característica más importante del protocolo DAP es que permite el acceso a subconjuntos de datos remotos mediante el uso de *constraint expressions*. Las *constraint expressions* consisten en un lenguaje que permiten acceder a proyecciones y selecciones del data source. Las proyecciones permiten seleccionar subconjuntos del dataset en función del tipo de dato, mientras que las selecciones permiten seleccionar subconjuntos del dataset en función de los valores de las variables. En la Figura 4 se muestra un ejemplo de petición DAP.

```
[zequi@hera tfm] $ ncdump -h "http://193.146.75.233:8080/thredds/dodsC/chunked/tas_AERhr_CNRM-ESM2-1_
historical_r1i1p1f2_gr_185001010030-185412312330.nc?lat[0:1:127],lon[0:1:255]"
netcdf tas_AERhr_CNRM-ESM2-1_historical_r1i1p1f2_gr_185001010030-185412312330 {
dimensions:
        lat = 128 ;
        lon = 256 ;
variables:
        double lat(lat) ;
                lat:axis = "Y" ;
                lat:standard_name = "latitude" ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
        double lon(lon) ;
                lon:axis = "X" ;
                lon:standard_name = "longitude" ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;

// global attributes:
                :Conventions = "CF-1.7 CMIP-6.2" ;
                :creation_date = "2018-09-15T06:24:21Z" ;
                :description = "CMIP6 historical" ;
                :title = "CNRM-ESM2-1 model output prepared for CMIP6 / CMIP historical" ;
```

Figura 4: Ejemplo de petición DAP remota usando constraint expressions

Esta característica es muy importante para el análisis de datos, ya que los clientes que soporten el protocolo DAP ya no necesitan descargar el dataset completo, sino que pueden hacer uso de las *constraint expressions* para solicitar al servidor únicamente el subconjunto

del dataset deseado, simplificando enormemente el trabajo del analista de datos.

Servidores de acceso remoto a datos

El servidor de datos Hyrax [20] es la implementación de referencia del protocolo DAP. Está dividido en dos grandes partes que forman el *frontend* y el *backend* del servidor. El frontend consiste en un servidor basado en Java servlet denominado OPeNDAP Lightweight Front-end Servlet (OLFS). El OLFS se encarga de atender las peticiones públicas al servidor Hyrax y delega las peticiones al Back End Server (BES), un servidor implementado en C++, encargado de procesar las peticiones DAP y generar las respuestas.

Por otro lado, el proyecto THREDDS, acrónimo de Thematic Real-time Environmental Distributed Data Services, es un proyecto de Unidata con el objetivo de proveer el software necesario que simplifique el acceso a datasets científicos. El proyecto está compuesto por una librería, netCDF-java y un servidor web, THREDDS Data Server (TDS) [21] que permiten el acceso remoto a datos científicos.

Las características más importantes del proyecto son las herramientas de catalogación y el acceso remoto. Las herramientas de catalogación permiten estructurar y presentar los datasets científicos en ficheros denominados catálogos, documentos XML que son interpretados por el TDS para presentar al usuario la colección de datasets disponibles en el servidor. La Figura 5 muestra un ejemplo de catálogo en el cliente.

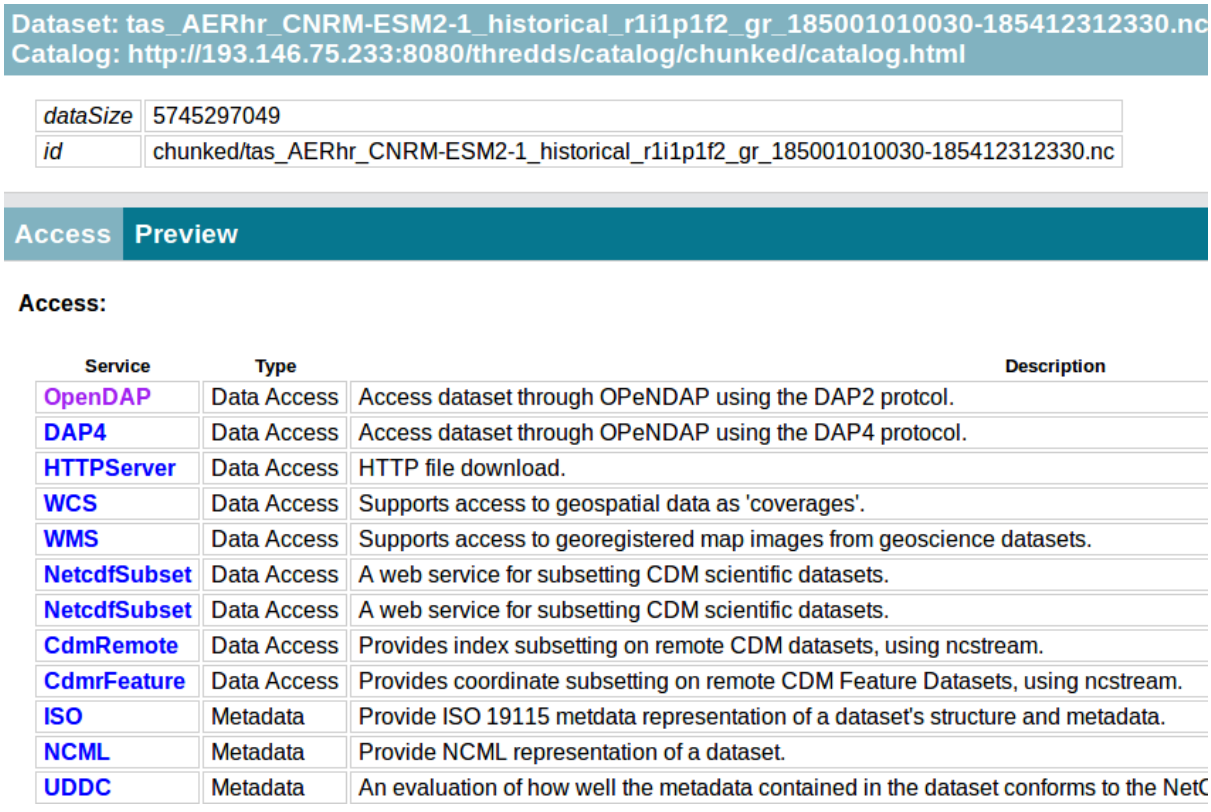

Figura 5: Ejemplo de catálogo TDS mostrado en un navegador web

El acceso a los datasets en el TDS se realiza a través de servicios. Estos servicios están bien definidos por el servidor y el administrador del servidor se encarga, haciendo uso de los catálogos, de definir mediante qué servicios son accedidos los distintos datasets. Entre estos servicios podemos encontrar los servicios DAP y HTTP. Un listado completo de los

servicios que dispone el TDS se puede encontrar en [22].

2.4 Repositorios de datos

Las instituciones que publican datos climáticos han puesto a disposición del público dichos datos mediante el uso de distintos protocolos y tecnologías. Los usuarios que quieran analizar los datos publicados, deben localizar en primer lugar el servidor alojado por la institución, normalmente identificado por un nombre de dominio en el sistema DNS. Una vez localizado el servidor, el usuario puede realizar una copia local de los datos o acceder mediante el protocolo DAP usando las herramientas cliente que comentaremos más adelante. A continuación comentamos dos ejemplos de los repositorios más relevantes a nivel mundial.

Earth System Grid Federation

La Earth System Grid Federation (ESGF) [23] es una federación internacional de colaboradores que provee software e infraestructura para la administración, diseminación y análisis de datos generados por ESM o datos de observaciones. Es una iniciativa internacional liderada por el Department of Energy (DOE) de Estados Unidos y cofinanciada por National Aeronautics and Space Administration (NASA), National Oceanic and Atmospheric Administration (NOAA), National Science Foundation (NSF), y laboratorios internacionales como Max Planck Institute for Meteorology (MPI-M) German Climate Computing Centre (DKRZ), Australian National University (ANU) National Computational Infrastructure (NCI), Institut Pierre-Simon Laplace (IPSL) y Centre for Environmental Data Analysis (CEDA).

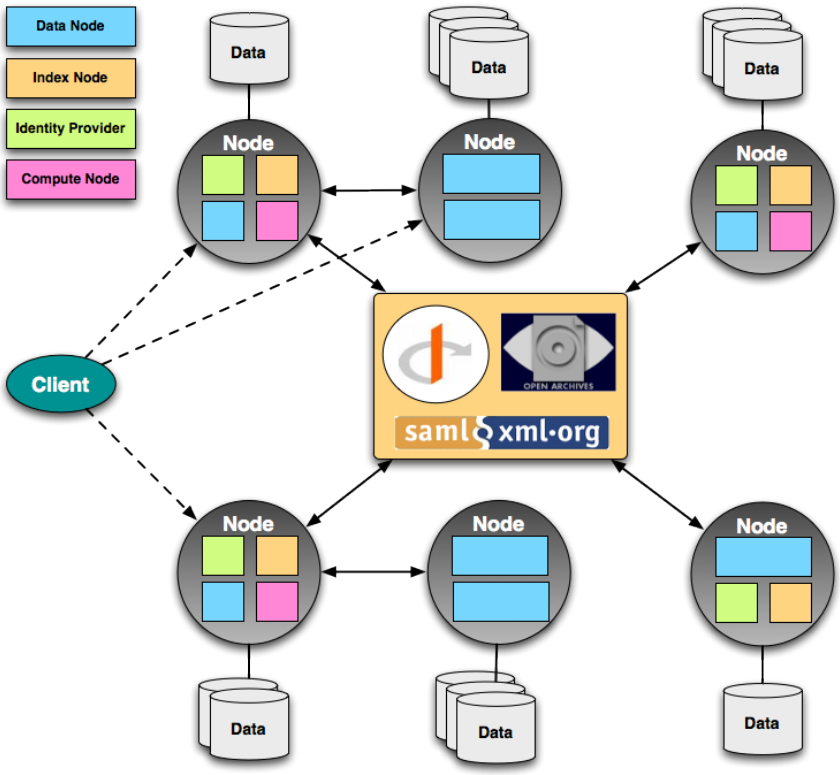

Figura 6: Arquitectura de la ESGF. Fuente: [23]

ESGF provee distintos portales, mantenidos por las instituciones mencionadas anteriormente, que permiten el acceso y la búsqueda a datos de forma distribuida como se muestra en la Figura 6. Es decir, desde el portal de, por ejemplo, el DKRZ, un usuario puede realizar búsquedas cuyos resultados mostrarán los datasets de todas las instituciones que forman parte de la federación. Las instituciones principales forman parte de ESGF como nodos primarios, mientras que instituciones de menor tamaño pueden formar parte de la federación como nodos secundarios, ofreciendo solo servicio de acceso a datos. La Universidad de Cantabria, como socio europeo de la infraestructura ENES [24], forma parte de la federación y mantiene un nodo secundario a cargo del SMG y su servicio de datos.

ESGF forma un complejo sistema distribuido a nivel internacional que permite el acceso a los datos mediante las tecnologías de acceso que se describirán más adelante. Un usuario de ESGF puede realizar una copia local de los datos mediante la ejecución de un script wget o puede acceder usando el protocolo DAP disponible en los servidores TDS que forman parte de la federación.

Copernicus Climate Data Store

Copernicus Climate Data Store (CDS) es un repositorio de acceso a datos climáticos que está adquiriendo gran importancia y relevancia a nivel mundial. Está desarrollado por el ECMWF, bajo el mandato de la Comisión Europea y la Agencia Europea del Espacio [25], cuyo objetivo es proveer un punto de acceso único para los usuarios que requieran descubrir y procesar datos y productos climáticos disponibles en repositorios distribuidos. El CDS también provee una extensa librería de software, la CDS Toolbox, que permite a los usuarios desarrollar sus propias aplicaciones. Estas aplicaciones harán uso del contenido disponible en el CDS para analizar, monitorizar y predecir la evolución del clima. Con este objetivo, el CDS incluye un conjunto de índices climáticos adecuados para los sectores energéticos, turismo y otros. Estos índices forman parte del Sectorial Information System (SIS), parte del Copernicus Climate Change Service (C3S) [26].

2.5 Herramientas para el análisis de datos climáticos

En este punto ya hemos comentado qué son los datos climáticos, cómo y dónde se almacenan y cómo son accedidos por clientes remotos. En este apartado comentaremos las distintas herramientas que permiten, una vez disponemos de los datos, realizar análisis de datos sobre ellos.

Estas herramientas consisten en librerías que acceden a los datos en formato netCDF, ya sea de forma local o remota mediante DAP y nos permiten modificar, realizar cálculos y visualizar los resultados. Las librerías de análisis de datos ofrecen distintas interfaces, desde la línea de comandos hasta entornos gráficos de análisis. Algunos ejemplos son:

NCO

Los netCDF Climate Operators (NCO) consisten en herramientas de línea de comandos que permiten realizar operaciones como derivar nuevas variables, calcular estadísticas, visualizar, seleccionar, proyectar e interpolar [27]. Aceptan como entrada ficheros netCDF y HDF, además de servidores DAP y permiten almacenar la salida en estos mismos formatos. Los NCO se pueden usar para el análisis de datos en grid o de datos no estructurados. Al

consistir en herramientas de línea de comandos, permiten trabajar tanto de forma interactiva como de forma automatizada mediante el uso de scripts.

CDO

Los Climate Data Operators (CDO) son herramientas de línea de comandos que permiten el procesamiento de datos climáticos [28]. Los operadores incluyen funciones aritméticas y estadísticas, funciones de selección y funciones de interpolación espacial. El objetivo de esta librería es ofrecer funciones comunes para el análisis de datos sobre dos formatos distintos, netCDF y GRIB. Sin embargo, el análisis de netCDF está limitado a la versión del modelo netCDF classic.

ToolsUI

ToolsUI es una librería basada en la librería netCDF-java de Unidata que permite visualizar y modificar datasets que implementan el Common Data Model (CDM) [29], además de acceder a catálogos THREDDS. Su interfaz es una aplicación gráfica desarrollada en Java que permite analizar tanto datos locales como remotos mediante mediante distintos protocolos como DAP, HTTP o ncstream [30].

netCDF4-python

netCDF4 es la interfaz, implementada por Unidata, para el acceso a datos netCDF usando el lenguaje de programación Python. La librería está implementada sobre la librería netCDF-C y permite trabajar con datos en formato netCDF-classic y netCDF4. También permite el acceso a datos remotos mediante DAP y soporta programación paralela mediante el uso de la librería mpi4py [31, p. 4].

Climate4R

Climate4R es un conjunto de paquetes, escritos en el lenguaje de programación R, para el acceso transparente a datos, post procesamiento (incluyendo corrección del bias y downscaling) y visualización. Sus principales estructuras de datos son el grid y los datos procedentes de observaciones, que permiten trabajar con datos organizados en rejilla, observaciones, reanálisis, predicciones estacionales, predicciones climáticas y ensembles [32].

Como se muestra en la Figura 7, Climate4R está integrado con el servicio de datos proporcionado por el Grupo de Meteorología de Santander (SantanderMetGroup). Este servicio de datos está basado en el servidor TDS, lo que permite al framework Climate4R acceder a datos climáticos mediante el protocolo DAP, haciendo un uso eficiente del acceso a datos.

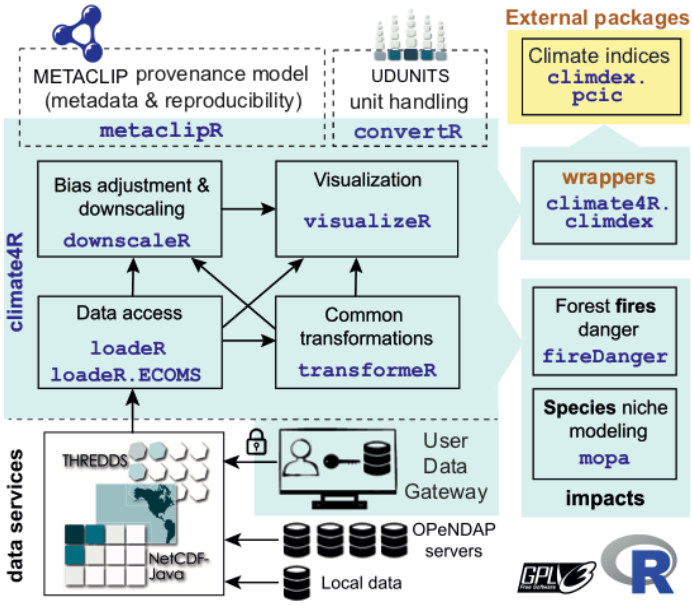

Figura 7: Arquitectura de la librería Climate4R. Fuente: [32]

Xarray

Xarray es una librería en Python para trabajar con arrays multidimensionales. Xarray usa etiquetas (labels) en forma de dimensiones, coordenadas y atributos sobre arrays de Numpy, como se muestra en la Figura 8, haciendo que el análisis de arrays multidimensionales sea más sencillo, conciso y menos propenso a errores. Su modelos de datos está ampliamente inspirado en netCDF y se integra con Dask para ofrecer análisis paralelo y escalable. También ofrece acceso a datos climáticos mediante DAP y netCDF4-Python.

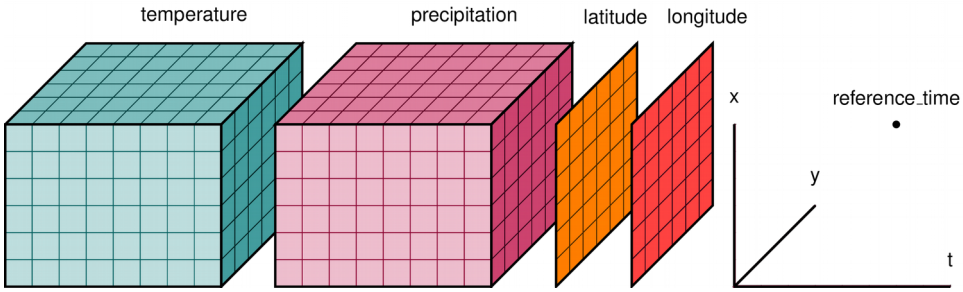

Figura 8: Ejemplo de un dataset multidimensional. Fuente: [59]

2.6 Nuevas tecnologías para el análisis de datos

Los ESM corren en entornos HPC de forma paralela, con el objetivo de incrementar su rendimiento. Este tipo de paralelización, basada en el estándar MPI (Message Passing Interface) [33], es el estándar dominante en el entorno HPC. El estándar MPI es un tipo de paralelismo distribuido basado en procesos. Estos procesos son distribuidos entre los servidores de la infraestructura HPC y son coordinados por la misma aplicación. Por lo tanto, para aprovechar este paralelismo, se requiere de un entorno HPC y que la aplicación haga uso de las librerías MPI en su implementación.

Sin embargo, las herramientas de análisis que hemos visto no están implementadas sobre MPI, ya que están destinadas a analistas de datos que pueden no disponer de un entorno HPC para ejecutarlas. Algunas de las herramientas de análisis que hemos visto intentan aprovechar el paralelismo a nivel de hilo, con el objetivo de sacar el máximo rendimiento a las CPU multicore que vienen instaladas hoy en día en todos los sistemas. Este flujo de trabajo presenta los siguientes inconvenientes:

- El científico analista de datos debe realizar la instalación de las herramientas de análisis. Esta puede no ser una tarea trivial, ya que algunas de las herramientas pueden requerir compilar el código fuente para disponer de toda su funcionalidad. Si bien existen paquetes precompilados que facilitan las distribución de estas herramientas, sería ideal disponer de entornos ya preparados y listos para ejecutar las tareas de análisis.
- Las herramientas de análisis tienen poco margen de escalabilidad. En el mejor de los casos, estas herramientas implementan paralelismo a nivel de hilo con el objetivo de aprovechar las CPU *multicore*. Otras herramientas ofrecen paralelismo basado en MPI pero los usuarios no disponen de entornos HPC para aprovechar este paralelismo o simplemente no tienen los conocimientos necesarios para aprovecharlas.

La situación ideal sería disponer de herramientas que ofrezcan las funciones necesarias para realizar el análisis de datos y, de forma automática, se encarguen de paralelizar de la ejecución de dichas funciones de manera escalable.

A continuación comentaremos las alternativas más relevantes desde el punto de vista del análisis de datos en las ciencias del clima. Estas alternativas ya han sido exploradas y puestas en práctica por el proyecto Pangeo, si bien se considerarán tecnologías adicionales y se presentará un análisis de cómo su aparición altera el entorno del análisis de datos en las ciencias del clima.

Jupyter

Jupyter es un proyecto Python cuyo objetivo es desarrollar software y estándares abiertos que permitan realizar análisis de datos de forma interactiva, desde cualquier lenguaje de programación. Jupyter está formado por dos proyectos principalmente, Jupyter Notebook y JupyterHub.

Jupyter Notebook es una aplicación web que permite crear y compartir documentos, denominados *notebooks*, que contienen código ejecutable, ecuaciones, gráficos y texto descriptivo. Los usos de los *notebooks* son: curación y transformación de datos, simulación numérica, modelización estadística, visualización de datos y machine learning [34].

Un *notebook* consiste en un fichero en formato JSON que es interpretado por el navegador para ofrecer un entorno de análisis de datos al usuario. El código ejecutable de un *notebook* puede ser ejecutado desde el navegador, que enviará al kernel del *notebook* dicho código para que lo ejecute. Jupyter Notebook es una aplicación destinada a ser usada por solo un usuario. Para disponer de un entorno multiusuario que permita la creación y edición de *notebooks* a múltiples usuarios debemos usar JupyterHub.

JupyterHub consiste en un despliegue centralizado que permite acceder de forma remota a los *notebooks*. Provee múltiples herramientas de autenticación que permiten la

autenticación de los usuarios, como módulos PAM y OAuth [35].

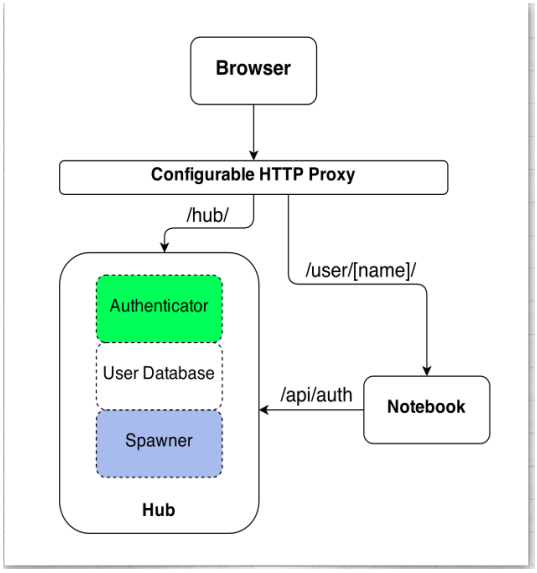

Figura 9: Componentes de JupyterHub. Fuente: [35]

JupyterHub está formado por un conjunto de procesos que, de forma conjunta, ofrecen un servidor Jupyter Notebook individual para cada usuario. Como se puede ver en la Figura 9, los componentes principales son:

- Hub (Python/Tornado): administra las cuentas de usuario, la autenticación y se encarga de la coordinación de los servidores individuales mediante un *Spawner*.
- Proxy: es la interfaz pública del JupyterHub. Usa un proxy HTTP dinámico para redirigir las peticiones de los usuarios a sus servidores individuales. Este *proxy* está basado node-http-proxy [36].
- *Notebook* de usuario (Python/Tornado): es un servidor Jupyter Notebook, dedicado e individual, que es creado para cada usuario del sistema en el momento de login. El objeto encargado de arrancar el servidor individual se denomina *Spawner*.

Dask

Dask es una librería para realizar computación paralela en Python [37]. Está compuesta fundamentalmente por dos partes, mostradas en la Figura 10:

- Planificador dinámico de tareas - El planificador se encarga de optimizar el grafo de tareas generado por el programa del usuario.
- Colecciones - Estas colecciones están basadas en librerías como numpy y pandas y son utilizadas para distribuir y escalar la computación del grafo de tareas.

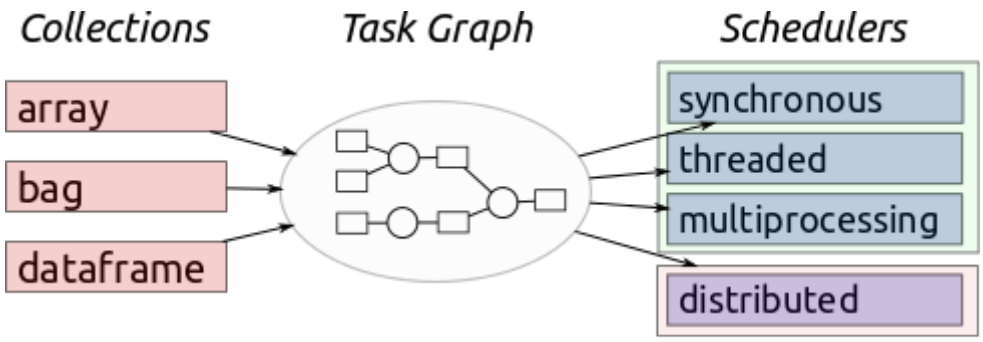

Figura 10: Componentes básicos de Dask. Fuente: [37]

Dask es una librería muy fácil de integrar con los flujos de trabajo de análisis de datos dentro del ecosistema Python, ya que presenta una interfaz muy similar a la de las principales librerías de análisis de datos, como Numpy y Pandas. Por lo tanto, no requiere que el analista de datos se acomode a un nuevo entorno de programación, sino que puede reutilizar el conocimiento adquirido por el uso de otras librerías de Python. Además, Dask permite escalar el análisis de datos desde ordenadores portátiles hasta clusters de cientos de nodos, gracias a la implementación de distintos planificadores (*schedulers*).

Servicios de análisis de datos

Los servicios de análisis tratan de aprovechar las posibilidades que ponen a nuestro alcance los proyectos como Jupyter y Dask. El proyecto JupyterHub permite a una institución ofrecer a usuarios externos un entorno donde realizar análisis de datos, sin ningún tipo de requerimiento para el analista, salvo disponer de autorización para utilizar el servicio. De esta forma, los analistas de datos no tienen que preocuparse por la instalación de las herramientas de análisis, ya que la instalación de las mismas se ha realizado, por parte de la institución, en el servidor que aloja el servicio de análisis.

Por otra parte, los usuarios que requieran hacer análisis de datos sobre datasets que sobrepasen el tamaño de la memoria o el tamaño del almacenamiento del servidor, pueden no quedar satisfechos con la capacidad del servicio de análisis. Por ello es necesario disponer de herramientas que permitan llevar a cabo análisis de datos de forma escalable, como por ejemplo Dask.

2.7 Almacenamiento cloud

Según la definición del National Institute of Standards and Technology (NIST), el cloud computing es un modelo de servicio que permite acceder por red y bajo demanda a un conjunto de recursos de cómputo previamente configurados (red, servidores, almacenamiento, aplicaciones y servicios), que pueden ser rápidamente provisionados y lanzados con mínimo esfuerzo o interacción por parte del proveedor de servicios [38].

OpenStack

OpenStack [39] es un software open source para la creación de clouds públicos y privados, que se encarga del control de un conjunto de recursos de cómputo, almacenamiento y red de un centro de datos. El acceso a recursos se realiza mediante un panel de control, vía interfaz web o línea de comandos [40], que permite a los administradores controlar los

recursos y dar acceso a los usuarios a gran parte de los mismos para que puedan crear sus propias aplicaciones. Los componentes más importantes de OpenStack son:

- Nova - Provee control y acceso, bajo demanda, a recursos computacionales para la creación de máquinas virtuales o contenedores.
- Glance - Permite registrar y administrar imágenes para la creación de máquinas virtuales. Estas imágenes pueden ser guardadas en almacenamiento mediante Cinder.
- Swift - Almacenamiento de tipo *object storage*.
- Cinder - Almacenamiento de tipo block storage.
- Neutron - Servicio de red para las máquinas virtuales.
- Keystone - Servicio de autenticación de OpenStack. Soporta LDAP, OAuth, OpenID Connect, etc.
- Horizon - *Dashboard* web desde el que administrar los recursos cloud.
- Heat - Orquestación de recursos para dar soporte a una determinada aplicación.

El servicio de almacenamiento en bloques de OpenStack se denomina Cinder y es el encargado de proveer de volúmenes de datos a las máquinas virtuales de los servicios Nova o Ironic. En el caso del IFCA, el servicio Cinder utiliza el driver rbd [41] para proveer el servicio de almacenamiento en bloques mediante el sistema de almacenamiento Ceph [42].

El servicio de *object storage* de OpenStack puede ser proporcionado mediante Swift y Ceph. Swift es una tecnología de *object storage* con las siguientes características: es redundante, ofrece alta disponibilidad, escalable hasta múltiples petabytes, distribuido, eventualmente consistente y con una interfaz HTTP REST. El IFCA utiliza Ceph como proveedor del servicio *object storage*, por lo que queda fuera del TFM comentar Swift en profundidad. La documentación sobre Swift puede ser consultada en [43].

2.8 Resumen

A lo largo de este capítulo hemos visto qué son y cómo se producen los datos climáticos, hemos descrito las infraestructuras HPC en las que son alojados y hemos visto cómo son accedidos y analizados tanto de forma local como remota, asumiendo siempre que los datos están almacenados en un sistema de ficheros.

El enorme tamaño de los datasets climáticos más recientes ha incentivado, a la instituciones que los generan, la búsqueda de alternativas de persistencia frente al almacenamiento en sus propias infraestructuras HPC [4]. Los servicios de almacenamiento cloud ofrecen dicho servicio en forma de *object storage* accedido por HTTP, que tiene características distintas frente a los sistemas de ficheros y requieren distintas formas de acceso para aprovechar toda su capacidad. En el siguiente apartado introducimos el *object storage* como una forma alternativa de almacenamiento frente a los sistemas de ficheros, además de plantear cuales son las distintas formas de acceso a datos climáticos que residen en este tipo de almacenamiento.

3 Almacenamiento basado en object storage

3.1 Introducción

Los sistemas de ficheros basados en POSIX proveen semánticas de fuerte consistencia que muchas aplicaciones no necesitan o directamente no quieren. Las tecnologías *object storage* evitan estas semánticas y están diseñadas para ser extremadamente escalables, para su uso en entornos cloud o entornos comerciales similares.

La mayoría de las aplicaciones HPC han dependido de los sistemas de ficheros para almacenar la entrada y salida de dichas aplicaciones. Los sistemas de ficheros soportan la interfaz POSIX, incluyendo las funciones *open()*, *close()*, *read()* y *lseek()*, e implementan restricciones en el comportamiento de estas funciones. Por ejemplo, el estándar POSIX estipula que los datos que son almacenados con éxito usando la función *write()*, deben estar inmediatamente disponibles para ser accedidos mediante la función *read()*.

Esta semántica entre las funciones *read()* y *write()* parece trivial a simple vista, sin embargo introduce una gran complejidad en los sistemas de ficheros paralelos en red, donde los distintos clientes no cuentan con la información de otros clientes sobre qué información están modificando. Como resultado, sistemas de ficheros paralelos como Lustre y GPFS, implementan esquemas distribuidos de bloqueo que aseguran la consistencia POSIX, beneficiando la consistencia frente al rendimiento. Esta serialización es la antítesis de escalabilidad, por lo que algunos sistemas de ficheros han elegido relajar las semánticas de consistencia en favor de una mayor escalabilidad.

Las características que permiten a los *object stores* evitar estas limitaciones son:

- Espacio de nombres plano sin metadatos - Los objetos son almacenados bajo un identificador global unívoco y corresponde a cada implementación definir los metadatos del objeto. Esto contrasta con las jerarquías y directorios de los sistemas de ficheros y los metadatos que almacenan, como propietario y grupos, permisos en la jerarquía y metadatos fijos como nombre del fichero y fechas de creación, modificación y último acceso.
- Acceso mediante operaciones atómicas sin estado - A diferencia de los sistemas de ficheros, no existe un descriptor de fichero que mantiene el estado de un fichero ni requisitos como que un fichero deba ser abierto antes de que sus datos puedan ser accedidos. Los usuarios simplemente indican el identificador del objeto al que desean acceder de forma completa.
- Inmutabilidad de los objetos - Una vez que un objeto es almacenado, por ejemplo, mediante una operación PUT, nunca podrá ser modificado. Esto, combinado con la atomicidad de la operación PUT, evita la necesidad de sistemas de bloqueo distribuidos. Si el dato puede ser accedido por una operación GET, está garantizado que es consistente e inmutable.

Estas tres características permiten a los object stores escalar por encima de las capacidades de los sistemas de ficheros, ya que no intentan proveer las semánticas definidas por el estándar POSIX. El corolario, sin embargo, es que los object stores pueden

ser mucho más difíciles de gestionar para los usuarios y las aplicaciones. El acceso a datos mediante un identificador global, en un espacio de nombres plano, es el equivalente a referenciar los inodes de los sistemas de ficheros para acceder a los archivos. La inmutabilidad de los objetos también hace que estos sistemas sean inadecuados para datos altamente dinámicos, es decir, que vayan a ser modificados en el tiempo. Como resultado, los object stores han estado limitados a industrias con tremendos volúmenes de datos y una considerable semántica WORM (Write Once Read Many).

3.2 Ceph

Ceph [44] es un sistema de almacenamiento que ofrece el servicio en forma de sistema de ficheros, almacenamiento en bloque y *object storage*, implementado sobre la librería RADOS (Reliable Autonomic Distributed Object Store). El sistema de almacenamiento Ceph consiste en un cluster formado por los monitores, los administradores, el demonio de almacenamiento de objetos (OSD) y el servidor de metadatos [45], descritos a continuación:

- Los monitores mantienen el estado interno del cluster de forma que los distintos demonios puedan coordinarse entre ellos. Normalmente se ejecutan tres monitores para proveer redundancia y alta disponibilidad.
- Los administradores se encargan de administrar las métricas del clúster, incluyendo el porcentaje de almacenamiento usado y carga del sistema. Estas métricas son accesibles, mediante el uso de módulos de Python, a través de una interfaz web y de una API REST.
- El OSD almacena los objetos, administra su replicación, se encarga de la recuperación en caso de fallo, del balanceo y ofrece información a los monitores y administradores. Ceph ejecuta tres OSD para proveer redundancia y alta disponibilidad.
- El servidor de metadatos se encarga de almacenar los metadatos POSIX necesarios para el acceso mediante sistema de ficheros en Ceph.

Ceph almacena los objetos en almacenes lógicos de datos (storage pools) mediante el algoritmo CRUSH (Controlled Replication Under Scalable Hashing), el cual permite a Ceph escalar, balancear y recuperar de forma dinámica.

Es posible acceder a Ceph mediante una API HTTP REST, haciendo uso de Ceph Object Gateway [46]. Ceph Object Gateway consiste en una interfaz implementada sobre la librería *librados* para proveer dicha funcionalidad. Como se muestra en la Figura 11, esta interfaz es compatible tanto con la API S3 de Amazon como la API Swift de OpenStack y está implementada como un servidor HTTP que se encarga de la comunicación con el cluster Ceph, con gestión propia de usuarios.

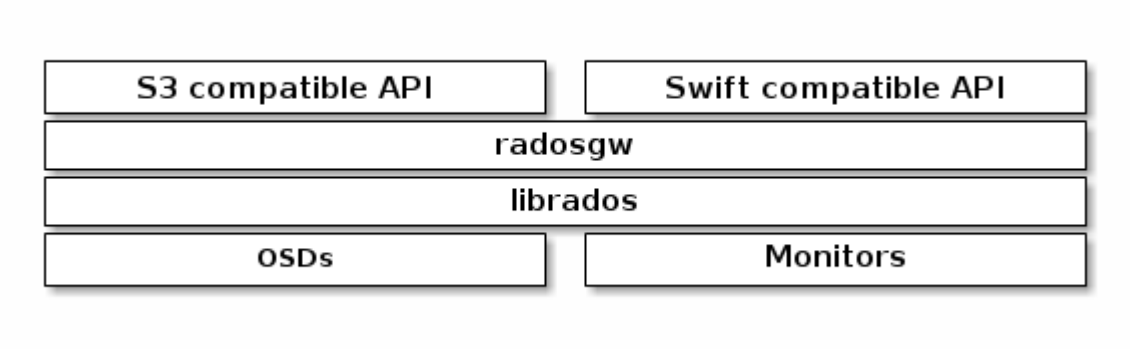

Figura 11: Interfaces y componentes de Ceph. Fuente: [46]

3.3 Acceso a datos climáticos en object storage

Este nuevo sistema de almacenamiento conlleva un replanteamiento en los flujos de trabajo que involucran la generación de los datos climáticos y el acceso a los mismos. Si no se tienen en cuenta las características de este nuevo entorno de almacenamiento, el acceso a los datos climáticos podría llegar a ser tan costoso como ineficiente. Es necesario tener en cuenta que las librerías tradicionales, como HDF5, desarrolladas para trabajar sobre sistemas de ficheros, no pueden acceder a información almacenada en *object storage* sin el *middleware* necesario.

Existen diversas formas de conseguir que los ficheros netCDF existentes puedan ser persistidos en *object storage* y accedidos por aplicaciones existentes. La primera forma de mover datos al cloud consiste en copiar todo un fichero netCDF bajo un único objeto en *object storage*, identificado por un único identificador. Esto hace que el fichero netCDF se vuelva completamente inaccesible, ya que la librería netCDF o la librería HDF5 requieren un objeto *FILE* de C.

Una forma de conseguir que el fichero netCDF, almacenado en cloud, sea accesible, es usar un sistema de ficheros FUSE (Filesystem in Userspace) [47] que simule un sistema de ficheros sobre el almacenamiento cloud. Aunque el fichero sea ahora accesible, lo es de forma muy ineficiente, ya que la librería HDF5 realizará múltiples lecturas de pequeños bloques (4kB) para leer los metadatos del fichero. Este compartimiento es aceptable cuando el fichero está almacenado de forma local pero, en este nuevo entorno, cada lectura supone una petición HTTP.

Otra forma de acceder a un fichero netCDF es usar HDF5 Virtual File Layer, mostrado en la Figura 12, seleccionando un driver de HDF5 que permita almacenar el fichero en un object store. Existen prototipos que ofrecen está funcionalidad para S3, aunque al igual que con FUSE el rendimiento sería ineficiente.

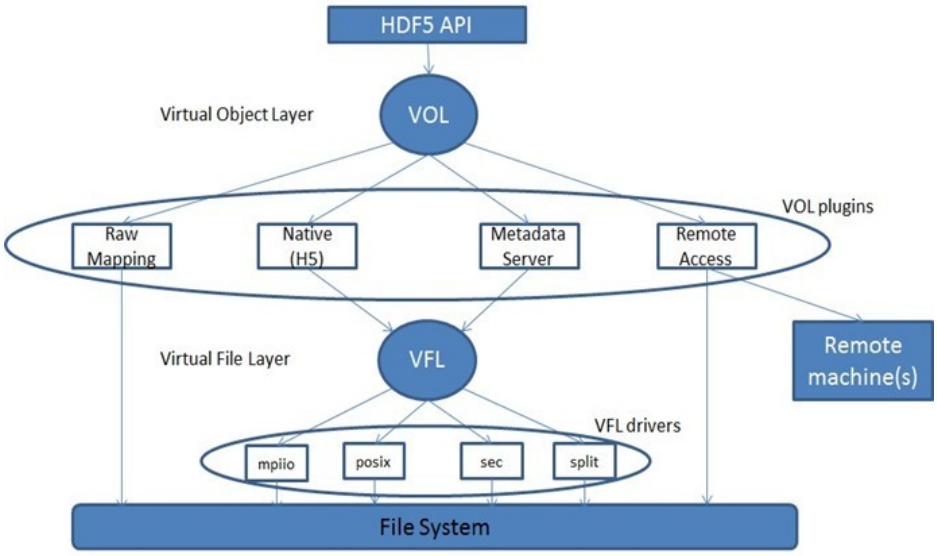

Figura 12: Capas Virtual Object Layer y Virtual File Layer de HDF5. Fuente: [57, p. 5]

3.4 Formatos nativos para object storage

La solución para almacenar datos climáticos en *object storage* de forma eficiente, requiere definir nuevos esquemas de almacenamiento que permitan aprovechar las características de este nuevo entorno. La definición de nuevos esquemas de almacenamiento no conlleva, necesariamente, la creación de un nuevo modelo de datos. Como veremos en este apartado, el modelo de datos de HDF5 puede ser implementado sobre un esquema de almacenamiento distinto, apropiado para su uso en *object storage*. Es necesario distinguir entre el modelo de datos (*data model*) y la forma concreta de persistir ese modelo dado un sistema de almacenamiento, es decir, el esquema de almacenamiento (*storage layout*).

Cloud HDF5

El formato HDF5 ha consistido, de forma tradicional, en el modelo de datos, el esquema de almacenamiento que persiste el modelo de datos en un sistema de ficheros y en la librería C que implementa el modelo y el esquema de almacenamiento. Este modelo limita las capacidades de trabajo con el formato HDF5, ya que no permite acceso remoto y limita el esquema de almacenamiento a sistemas de ficheros. Si en las ciencias del clima se ha podido trabajar con ficheros HDF5 de forma remota, ha sido gracias a la librería netCDF y el protocolo DAP, que complementan al formato HDF5 con dicha funcionalidad.

Con el paso del tiempo, el grupo HDF ha desarrollado una especificación en la que el modelo de datos de HDF5 puede ser accedido remotamente mediante una API REST HTTP, denominada RESTful HDF5 [48]. Esta especificación no implica un cambio en el esquema de almacenamiento, que sigue siendo un fichero HDF5 almacenado en un sistema de ficheros, sino que permite exponer un fichero HDF5 de forma remota utilizando un servidor HTTP que implemente la especificación, como lo es la implementación desarrollada por el grupo HDF, h5serv [49]. El cliente, desarrollado por el grupo HDF5, que da acceso a datos mediante el servidor h5serv, se denomina h5pyd [50].

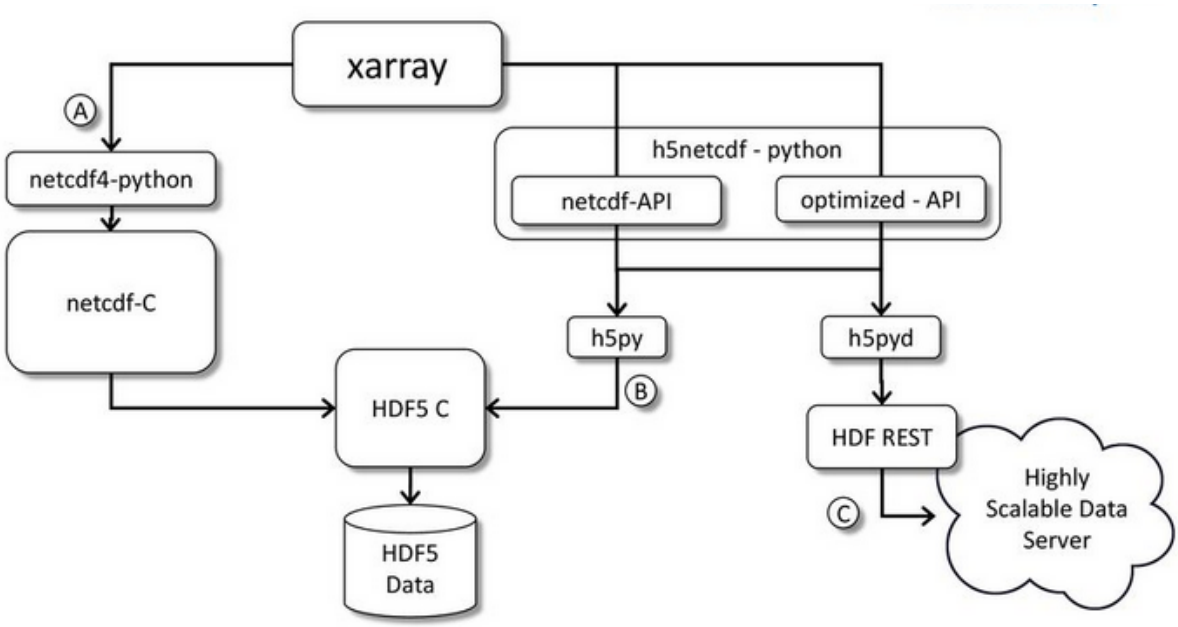

Figura 13: En C se muestra el acceso al servicio HSDS. Fuente: [60]

Una vez disponemos de una API que permite el acceso remoto a datos en formato HDF5, el siguiente paso consiste en adaptar el esquema de almacenamiento de modo que se adapte al *object storage*. El grupo HDF ha implementado un servicio, denominado HSDS (Highly Scalable Data Service), que permite la persistencia del modelo HDF5 en *object storage* y que puede ser accedido mediante el cliente h5pyd.

Zarr

Zarr es un paquete de Python que provee una implementación de arrays multidimensionales, con opciones de compresión y de *chunking*. Zarr es a la vez un modelo de datos y una librería de Python que implementa dicho modelo. El esquema de almacenamiento no está predefinido por el modelo de datos y se implementa en forma de *plugin*. El único requisito del esquema de almacenamiento es que el sistema ofrezca una interfaz clave/valor. Zarr permite la agrupación de distintos arrays multidimensionales mediante el uso de grupos.

El diseño de Zarr hace que sea un formato adecuado para su almacenamiento tanto en sistemas de ficheros, donde la clave corresponde con la ruta completa del fichero y el valor es el contenido del fichero, como en *object storage*, donde la clave es el identificador del objeto y el valor es el propio objeto. Es por esta razón que el proyecto Pangeo ha elegido el formato Zarr como forma de persistir los datos climáticos.

4 Evaluación de acceso a datos climáticos en cloud y HPC

La evaluación consiste en visualizar cuales son las distintas formas de acceder a datos climáticos, además de evaluar el throughput de los distintos entornos. Para hacer esta evaluación se creará un sencillo caso de uso que acceda de forma completa a un dataset, que consiste en realizar la media sobre la dimensión temporal de una variable climática, la temperatura en superficie (tas, Surface Air Temperature), perteneciente a un dataset generado por el ESM CNRM-ESM2-1 [51]. Los scripts y *notebooks* utilizados están disponibles en Github [52].

Este dataset está disponible en ESGF, como parte del proyecto CMIP6, en formato netCDF4. Tiene un *chunking* que provee acceso espacial óptimo, es decir, el acceso a valores de la variable temperatura en todo el grid espacial, en un instante de tiempo, es muy rápido, mientras que el acceso a múltiples valores en la coordenada temporal es mucho más lento.

El dataset será descargado y procesado tanto en la infraestructura OpenStack como en la infraestructura HPC del IFCA, donde se crearán los entornos apropiados para cada evaluación. Para realizar una comparativa de la forma más equilibrada posible, se utilizará un *chunking* común en todos los modos de acceso y persistencia del dataset. El *chunking* elegido es 2739x8x32 en las dimensiones ‘time’, ‘lat’ y ‘lon’ respectivamente. El dataset será almacenado en los formatos netCDF4 y Zarr, completamente descomprimido y con el *chunking* especificado, ocupando en ambos casos 5,4Gb de almacenamiento.

4.1 Instalación del entorno cloud

Las máquinas virtuales usarán distintos flavors de OpenStack. Cada flavor proporciona distintas capacidades hardware a las máquinas virtuales. Los dos flavors usados para las máquinas virtuales son:

- cm4.2xlarge - 8 CPUs virtuales, 14,6GB de RAM y 30GB de almacenamiento
- cm4.4xlarge - 16 CPus virtuales, 29,3GB de RAM y 30GB de almacenamiento

Las distintas máquinas virtuales creadas en este entorno están conectadas mediante una red local de 1Gb. El ancho de banda ha sido probado con iperf, obteniendo un ancho de banda efectivo de 940Mb/sec. La instalación del servidor JupyterHub ha sido automatizada usando la herramienta Ansible, lo que nos permite expresar el estado de nuestra infraestructura en forma de código (IaC, Infrastructure as Code).

La librería utilizada para llevar a cabo el análisis es xarray, que será utilizada junto con Dask para llevar a cabo la evaluación tanto usando paralelismo como acceso en serie. Los motores de xarray que dan acceso a los datos son:

- netcdf4 - Se corresponde con la librería netCDF4-python y se usará tanto para acceso local como para acceso remoto mediante DAP.
- h5netcdf - Acceso local y acceso remoto mediante HSDS.

- Zarr - Acceso local y acceso remoto a *object storage*.

4.2 Instalación del entorno HPC

La infraestructura HPC del IFCA está compuesta por el supercomputador Altamira, formado por 158 nodos de cómputo, un servidor de acceso [53] y un servidor de control. Los nodos de computación tienen instalada la distribución Scientific Linux 7.4. La red interna de Altamira está compuesta por:

- Red Infiniband (FDR) 40GB - Red de altas prestaciones usada por aplicaciones paralelas y para transferencia de datos.
- Red Gigabit - Red Ethernet usada para la administración de los servicios.

Todos los nodos están conectados al almacenamiento global GPFS que provee alrededor de 1PB de almacenamiento. El supercomputador Altamira está conectado a Internet con conexión directa a RedIris [54] a 1Gb/s [55]. La ejecución de tareas en el Altamira se realiza mediante la ejecución de *jobs* en el sistema Slurm [56].

La instalación de la librería netCDF4-Python, junto con la funcionalidad de ejecución en paralelo, requiere la instalación, desde código fuente, de las librerías HDF5 y netCDF con soporte para MPI. La instalación se ha llevado a cabo utilizando la instalación de MPI existente en Altamira, enlazando las librerías necesarias. La instalación de Python se ha llevado a cabo mediante un entorno Conda y se ha configurado la librería netCDF4-Python para que compile utilizando dicho entorno.

La instalación de Zarr se ha llevado a cabo en la misma máquina pero en un entorno Conda distinto. La instalación de Zarr no requiere ningún tipo de compilación y se puede instalar haciendo uso de los paquetes existentes en el repositorio Conda.

4.3 Test de acceso local en OpenStack

El objetivo de este test es identificar diferencias de rendimiento entre los diferentes motores de acceso. Para ello se ha creado un entorno común para los tres motores, en los que se probará tanto el acceso en serie como en paralelo.

El test de acceso local se muestra en la Figura 14 y está formado por una máquina virtual de OpenStack que actuará como cliente y servidor de datos. El cliente será un Jupyter Notebook que, mediante las distintas formas de acceso, medirá el tiempo de carga del dataset, usando tanto paralelismo como acceso en serie.

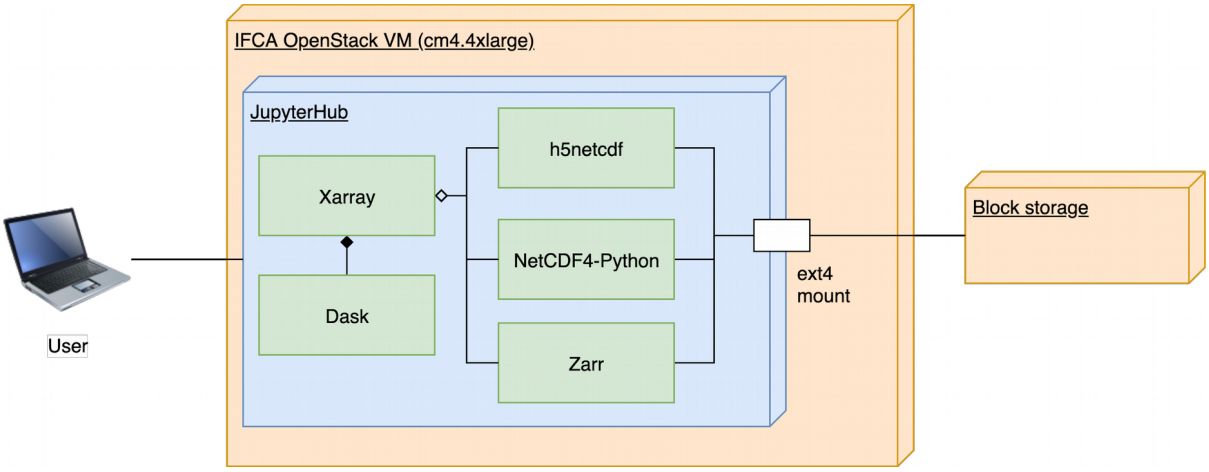

Figura 14: Arquitectura del test de acceso local. El test consiste en acceder al sistema de ficheros usando h5netcdf, netCDF4-Python y Zarr.

Los tiempos de acceso obtenidos para el test de acceso local son los siguientes:

Tipo acceso \ Librería	**netCDF4**	**Zarr**	**h5netcdf**
Serie	146,2s	148,8s	128,7s
Paralelo	90,5s	49,1s	109,1s
Aceleración	1,61	3	1,17

Tabla 1: Tiempos de acceso al sistema de ficheros ext4 de forma local

Los tres motores de acceso presentan tiempos similares respecto al acceso en serie. Sin embargo, respecto al acceso en paralelo, se observan diferencias significativas entre los métodos de acceso. Zarr es una librería optimizada para el acceso en paralelo por lo que obtiene los mejores resultados. netCDF4 y h5netcdf se benefician del acceso en paralelo pero en menor medida.

4.4 Test de acceso remoto en OpenStack

El objetivo de este test es mostrar las distintas formas de acceso remoto a datos climáticos en función del sistema de almacenamiento. Los tiempos medidos en esta evaluación no deben ser comparados entre sí, ya que las diferencias entre sistemas son demasiado grandes. El objetivo es mostrar cuales son los métodos de acceso disponibles para cada sistema de almacenamiento.

La infraestructura, mostrada en la Figura 15, está formada por dos instancias de OpenStack. Una de ellas actuará como cliente, en la que estará desplegado un servidor JupyterHub con las herramientas de cliente necesarias para el acceso a datos, mientras que en la máquina servidor estarán instalados los servidores TDS y HSDS para acceso a sistemas de ficheros y *object storage* respectivamente.

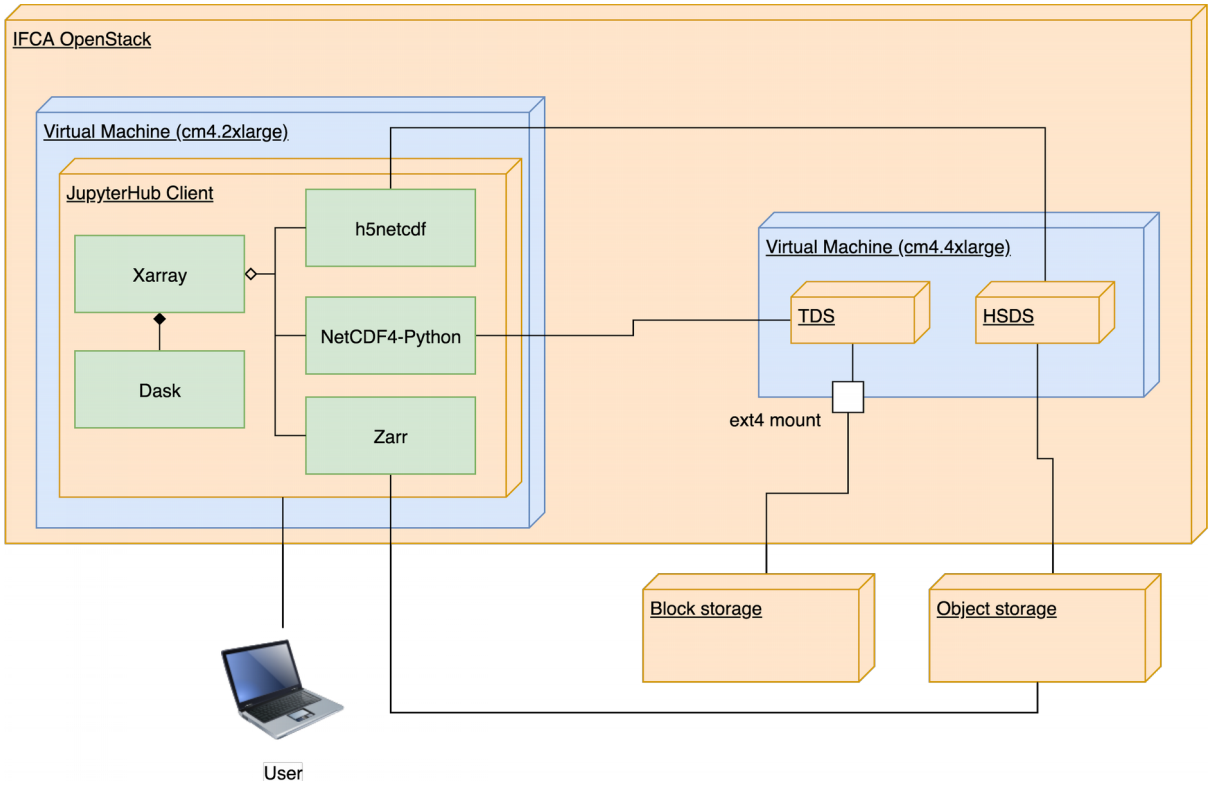

Figura 15: Arquitectura del test de acceso remoto. Las librerías acceden a distintos tipos de almacenamiento con el objetivo de mostras sus capacidades.

Los tiempos de acceso obtenidos para el test de acceso remoto son lo siguientes:

Tipo acceso \ Tecnología	netCDF4 - TDS	Zarr	h5netcdf - HSDS
Serie	302,9s	1273,6s	-
Paralelo	279,7s	97s	-
Aceleración	1,08	13,12	-

Tabla 2: Tiempos de acceso a los distintos tipos de almacenamiento en función de la librería

El caso más relevante en este apartado es HSDS, ya que no permite cargar el dataset debido a su tamaño, lo que parece ir en contra de los principios de diseño del servidor, ya que está orientado al acceso a datos de forma escalable. Zarr hace muy buen uso del acceso en paralelo pero el acceso en serie es mucho más lento de lo esperado. netCDF4 hace un uso bastante equilibrado de ambos tipos de acceso aunque parece no aprovechar de forma adecuada el paralelismo del cliente, posiblemente debido a los tiempos de respuesta del servidor.

4.5 Test de acceso local en Altamira

El test consiste en lanzar un *job*, mediante el uso de Slurm, al supercomputador Altamira, de

forma que el acceso a datos se realice de forma paralela mediante MPI. Cada tarea del *job* se encargará de computar sobre un subconjunto del dataset y el última instancia los datos serán recogidos por una de las tareas para calcular el resultado final. La infraestructura de la evaluación se muestra en la Figura 16:

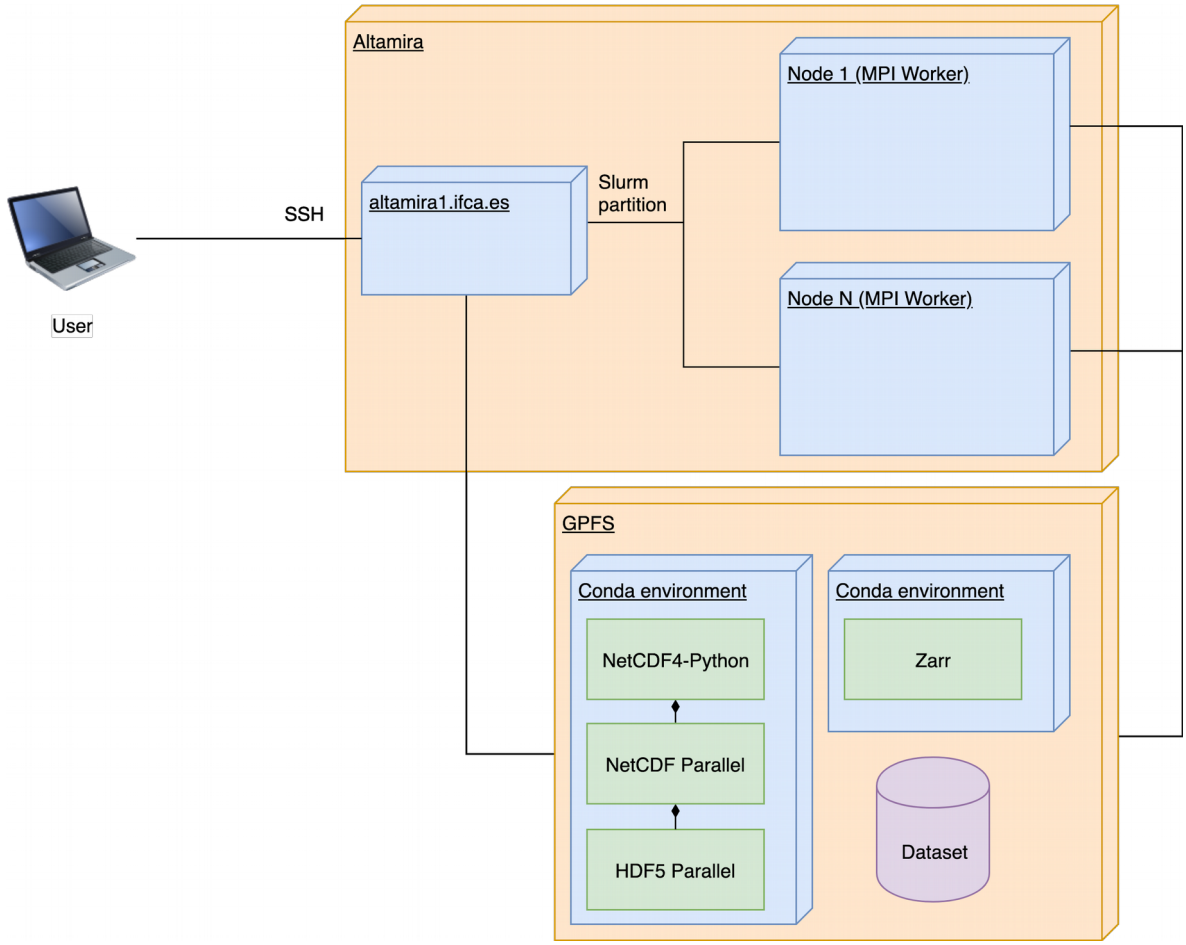

Figura 16: Arquitectura del test HPC. El trabajo será llevado a cabo de forma paralela por los nodos MPI.

Los test se han realizado utilizando distinto número de tareas para cada librería, de forma que se pueda observar el beneficio de añadir más recursos para realizar el cómputo. Esta parametrización se ha realizado mediante los ficheros de configuración de *jobs* de Slurm. Los tiempos de acceso obtenidos son:

Tareas\Librería	**netCDF4-Python Independent I/O**	**netCDF4-Python Collective I/O**	**Zarr**
2	145,04s	-	86,82s
4	80,22s	29,52s	37,35s
8	39,73s	17,75s	14,43s

Tabla 3: Tiempos de acceso de las librerías en función del número de tareas MPI

Para la librería netCDF4-Python se ha ejecutado el test haciendo uso tanto de la I/O independiente como colectiva de MPI y se puede observar la mejora en el tiempo de acceso de la segunda frente a la primera. En el caso de la librería Zarr se obtienen tiempos bastante bajos debido a que el dataset, almacenado en Zarr, no consiste en un único fichero cuyo acceso puede ser bloqueado por distintos procesos, sino que está almacenado en múltiples ficheros que no producen bloqueo.

5 Conclusiones y trabajo futuro

El desarrollo del trabajo se ha realizado en tres partes principalmente. La primera consiste en un análisis sobre las herramientas y metodologías de trabajo que, de forma tradicional, se usan a la hora de realizar análisis de datos sobre datasets de las ciencias del clima. Además, se han presentado sus limitaciones a la hora de lidiar con las grandes cantidades de datos que son generadas cada vez con más frecuencia en la actualidad, lo que supone investigar nuevas formas de realizar análisis de datos. Estas nuevas metodologías se caracterizan por pasar de ser entornos locales a entornos remotos y compartidos, cuyo objetivo es acercar el análisis de datos a los propios datos.

La segunda parte consiste en la presentación del *object storage* como alternativa de almacenamiento frente a los tradicionales sistemas de ficheros POSIX. Las grandes cantidades de datos generadas por las instituciones climáticas se está viendo altamente incrementadas, por lo que dichas instituciones están optando por mover estos datos a entornos cloud. Las compañías e instituciones que ofrecen almacenamiento cloud, ofrecen el servicio en forma de *object storage*, rompiendo la compatibilidad existente entre las aplicaciones de análisis de datos y el repositorio de los mismos. Ante este hecho, la comunidad está optando por distintas opciones, como la implementación de nuevas librerías (Zarr) o la adaptación de formatos existentes a sistemas de *object storage* (HSDS). Las alternativas para hacer frente al *object storage* se encuentran en una fase de pronto desarrollo y es de máxima importancia analizar las causas y consecuencias de cada uno de los planteamientos existentes, de forma que las soluciones sean robustas y sostenibles en el futuro.

En la tercera y última parte del trabajo se ha llevado a cabo una evaluación de las distintas formas de acceder a datos climáticos, teniendo en cuenta tanto rendimiento como funcionalidad, de algunas de las soluciones ofrecidas para el almacenamiento de datos climáticos en *object storage*. Estos tests se han realizado en entornos remotos multiusuario con la intención de simular entornos de análisis de datos eficientes y escalables.

En primer lugar, cabe destacar las ideas que la librería Zarr incorpora en cuanto al acceso de datos climáticos almacenados en cloud en *object storage*. Por un lado, desde el lado del cliente es posible apuntar a un repositorio remoto de *object storage*, lugar desde el cual los datos serán accedidos mediante HTTP como si se tratara de un sistema de ficheros remoto, sin intervención de ningún software servidor. Esto da lugar a un acceso realmente eficiente frente al limitado acceso del servidor HSDS. Además, la librería Zarr desecha el concepto de fichero como medio de almacenamiento y comienza a usar el concepto de *store* o almacén. De esta forma, un dataset climático almacenado en un sistema de ficheros, pasa a estar dividido en múltiples ficheros, separando los distintos *chunks* y los metadatos en distintos ficheros. Este concepto hace que la librería sea capaz de almacenar los mismos datos en *object storage* de forma funcional y eficiente para realizar análisis de datos.

Por otra parte, el uso de la librería Zarr en las ciencias del clima presenta inconvenientes de compatibilidad, ya que todas las aplicaciones de análisis o manipulación de datos climáticos están construidas, de una manera u otra, sobre netCDF, produciéndose así una incompatibilidad entre herramientas, cuando lo ideal sería abstraer el sistema de almacenamiento para lograr la compatibilidad con las aplicaciones. El análisis de datos

climáticos usando Zarr está limitado a la librería xarray e incluso en esta librería la implementación está en fase de desarrollo.

El trabajo futuro involucra estudiar en qué medida es posible dotar a la librería netCDF con las capacidades para almacenar datasets climáticos en *object storage*, de la misma forma en la que lo hace Zarr, considerando cada *chunk* un objeto independiente. Esto podría hacerse a nivel de HDF5, como se ha hecho con HSDS, sin tener que requerir un componente que intervenga entre la librería cliente y el repositorio de *object storage*. La abstracción necesaria para implementar esta funcionalidad ya existe en la librería HDF5 y se conoce como VOL (Virtual Object Layer) [57]. Si bien esta idea permitiría el almacenamiento en *object storage* de los datasets netCDF4, no funcionará con las versiones netCDF classic, ya que este formato no consiste en un fichero HDF5. La inclusión de este tipo de funcionalidad debe de tener en cuenta el amplio ámbito de uso de la librería netCDF y como nuevas funcionalidades pueden interferir con el comportamiento existente, como lo es el uso de la librería en entornos HPC tanto para lectura como escritura de datos, mediante el uso de MPI.

En cuanto a las evaluaciones realizadas en el trabajo, estas son de pequeña escala y funcionan a modo de prueba de concepto. Las pruebas en paralelo se han realizado únicamente a nivel de hilo y sobre un dataset de unos pocos Gb de tamaño. El paquete Dask es capaz de escalar desde un portátil hasta clusters de cientos de nodos, por lo que entre las tareas de trabajo futuro se encuentra la de desplegar un entorno cloud que permita paralelizar entre distintas máquinas, mediante el uso de un cluster Kubernetes o similar. También queda pendiente el uso de un dataset que sea considerado *Big Data*, de forma que se pueda validar la escalabilidad de las tecnologías presentadas, además de presentar un caso de uso más complejo que haga uso de la librería de machine learning de Dask.

Como conclusión final, decir que los requisitos de almacenamiento en *object storage* son una realidad que forman parte del presente y que no disponen aún de una solución consensuada, por lo que es de vital importancia incorporar soluciones que permitan aprovechar el valor de todos los datos que están siendo ya almacenados en los repositorios cloud.

6 Referencias

[1] «Earth system science», *Wikipedia*. 30-jun-2019.

[2] R. Abernathey *et al.*, «Pangeo NSF Earthcube Proposal». 31-ago-2017.

[3] «Supercomputer», *Wikipedia*. 25-jun-2019.

[4] «EOSDIS Data and Services Begin Migration to the Cloud | Earthdata». [En línea]. Disponible en: https://earthdata.nasa.gov/cmr-and-esdc-in-cloud. [Accedido: 26-jun-2019].

[5] A. Guillory, «ERA5», *ECMWF*, 03-nov-2017. [En línea]. Disponible en: https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5. [Accedido: 26-jun-2019].

[6] «What is the data volume of all ERA5 data - Copernicus Knowledge Base - ECMWF Confluence Wiki». [En línea]. Disponible en: https://confluence.ecmwf.int/display/CKB/What+is+the+data+volume+of+all+ERA5+data. [Accedido: 06-may-2019].

[7] «Object storage», *Wikipedia*. 26-feb-2019.

[8] J. Liu *et al.*, «Evaluation of HPC Application I/O on Object Storage Systems», en *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage Data Intensive Scalable Computing Systems (PDSW-DISCS)*, 2018, pp. 24-34.

[9] «Pangeo — Pangeo documentation». [En línea]. Disponible en: http://pangeo.io/. [Accedido: 26-jun-2019].

[10] «NSF - National Science Foundation». [En línea]. Disponible en: https://www.nsf.gov/. [Accedido: 26-jun-2019].

[11] «Homepage | EarthCube». [En línea]. Disponible en: https://www.earthcube.org/. [Accedido: 26-jun-2019].

[12] «NetCDF: The NetCDF User's Guide». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/netcdf/docs/user_guide.html. [Accedido: 26-jun-2019].

[13] «Unidata, 2019: *Network Common Data Format* [software]». Boulder, CO: UCAR/Unidata Program Center. https://doi.org/10.5065/D6H70CW6

[14] «Unidata | NetCDF Java». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/thredds/current/netcdf-java/. [Accedido: 26-jun-2019].

[15] «NetCDF: Interoperability with HDF5». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/netcdf/docs/interoperability_hdf5.html. [Accedido: 26-jun-2019].

[16] R. Jerger, «Modeling and Performance Prediction of HDF5 data on Objectstorage», Master's Thesis, Universität Hamburg, 2018.

[17] «HDF5 User's Guide: Data Model». [En línea]. Disponible en: https://support.hdfgroup.org/HDF5/doc1.6/UG/03_Model.html. [Accedido: 26-jun-2019].

[18] «CF Conventions Home Page». [En línea]. Disponible en: http://cfconventions.org/. [Accedido: 26-jun-2019].

[19] J. Gallagher, N. Potter, T. Sgouros, S. Hankin, y G. Flierl, «The Data Access Protocol — DAP 2.0», p. 47.

[20] «Hyrax Data Server | OPeNDAP™». [En línea]. Disponible en: https://www.opendap.org/software/hyrax-data-server. [Accedido: 26-jun-2019].

[21] «Unidata, *THREDDS Data Server version 5 [software]*». Boulder, CO: UCAR/Unidata Program Center, 2019. https://doi.org/10.5065/D6N014KG

[22] «TDS Services». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/thredds/current/tds/reference/Services.html. [Accedido: 26-jun-2019].

[23] «ESGF Home Page». [En línea]. Disponible en: https://esgf.llnl.gov/. [Accedido: 26-jun-2019].

[24] «Welcome to the ENES Portal — vERC». [En línea]. Disponible en: https://portal.enes.org/. [Accedido: 01-jul-2019].
[25] «CDS: What is the Climate Data Store (CDS)? - Copernicus Knowledge Base - ECMWF Confluence Wiki». [En línea]. Disponible en: https://confluence.ecmwf.int/pages/viewpage.action?pageId=113450743. [Accedido: 30-jun-2019].
[26] «CDS: What is the Climate Data Store (CDS) Toolbox? - Copernicus Knowledge Base - ECMWF Confluence Wiki». [En línea]. Disponible en: https://confluence.ecmwf.int/pages/viewpage.action?pageId=133260782. [Accedido: 26-jun-2019].
[27] C. S. Zender, «Bit Grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+)», *Geosci. Model Dev.*, vol. 9, n.º 9, pp. 3199-3211, sep. 2016.
[28] «CDO - Project Management Service». [En línea]. Disponible en: https://code.mpimet.mpg.de/projects/cdo/embedded/index.html. [Accedido: 26-jun-2019].
[29] «Common Data Model». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/thredds/current/netcdf-java/CDM/. [Accedido: 26-jun-2019].
[30] «ncstream». [En línea]. Disponible en: https://www.unidata.ucar.edu/software/thredds/v4.3/netcdf-java/stream/NcStream.html. [Accedido: 01-jul-2019].
[31] «MPI for Python — MPI for Python 3.0.2 documentation». [En línea]. Disponible en: https://mpi4py.readthedocs.io/en/stable/. [Accedido: 01-jul-2019].
[32] M. Iturbide *et al.*, «The R-based climate4R open framework for reproducible climate data access and post-processing», *Environ. Model. Softw.*, vol. 111, pp. 42-54, ene. 2019.
[33] «Message Passing Interface (MPI)». [En línea]. Disponible en: https://computing.llnl.gov/tutorials/mpi/. [Accedido: 23-abr-2019].
[34] «Project Jupyter». [En línea]. Disponible en: https://www.jupyter.org. [Accedido: 26-jun-2019].
[35] «Technical Overview — JupyterHub 1.0.0 documentation». [En línea]. Disponible en: https://jupyterhub.readthedocs.io/en/stable/reference/technical-overview.html. [Accedido: 23-may-2019].
[36] «jupyterhub/configurable-http-proxy: node-http-proxy plus a REST API». [En línea]. Disponible en: https://github.com/jupyterhub/configurable-http-proxy. [Accedido: 23-may-2019].
[37] «Dask — Dask 2.0.0 documentation». [En línea]. Disponible en: https://docs.dask.org/en/latest/. [Accedido: 26-jun-2019].
[38] G. Swenson, «Final Version of NIST Cloud Computing Definition Published», *NIST*, 25-oct-2011. [En línea]. Disponible en: https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published. [Accedido: 30-jun-2019].
[39] «Build the future of Open Infrastructure.» [En línea]. Disponible en: https://www.openstack.org/. [Accedido: 25-may-2019].
[40] «OpenStack Docs: OpenStackClient». [En línea]. Disponible en: https://docs.openstack.org/python-openstackclient/pike/. [Accedido: 25-may-2019].
[41] «OpenStack Docs: Available Drivers». [En línea]. Disponible en: https://docs.openstack.org/cinder/latest/drivers.html. [Accedido: 25-may-2019].
[42] «Block Devices and OpenStack — Ceph Documentation». [En línea]. Disponible en: http://docs.ceph.com/docs/master/rbd/rbd-openstack/. [Accedido: 25-may-2019].
[43] «OpenStack Docs: Welcome to Swift's documentation!» [En línea]. Disponible en: https://docs.openstack.org/swift/latest/index.html. [Accedido: 25-may-2019].
[44] «Homepage», *Ceph*. [En línea]. Disponible en: https://ceph.com/. [Accedido: 25-may-2019].

[45] «Intro to Ceph — Ceph Documentation». [En línea]. Disponible en: http://docs.ceph.com/docs/master/start/intro/. [Accedido: 25-may-2019].
[46] «Ceph Object Gateway — Ceph Documentation». [En línea]. Disponible en: http://docs.ceph.com/docs/master/radosgw/. [Accedido: 25-may-2019].
[47] «Filesystem in Userspace», *Wikipedia*. 17-jun-2019.
[48] G. Heber, «RESTful HDF5 - Interface Specification - Version 0.1», p. 43.
[49] *Reference service implementation of the HDF5 REST API: HDFGroup/h5serv*. The HDF Group, 2019.
[50] *h5py distributed: Python client library for HDF Rest API - HDFGroup/h5pyd*. The HDF Group, 2019.
[51] «CNRM-CERFACS CNRM-ESM2-1 model output prepared for CMIP6 CMIP historical», *http://cera-www.dkrz.de/WDCC/meta/CMIP6/CMIP6.CMIP.CNRM-CERFACS.CNRM-ESM2-1.historical*. 2018.
[52] E. C. Alvarez, *Evaluación de las tecnologías object storage para el almacenamiento y análisis de datos climáticos*. https://github.com/zequihg50/tfm. 2019.
[53] «Altamira Users Guide - IFCA Computing - Confluence». [En línea]. Disponible en: https://confluence.ifca.es/display/IC/Altamira+Users+Guide. [Accedido: 26-jun-2019].
[54] «RedIRIS - About RedIRIS», 20-jun-2018. [En línea]. Disponible en: https://www.rediris.es/rediris/. [Accedido: 26-jun-2019].
[55] «Supercomputing/Userguide - eScienceWiki». [En línea]. Disponible en: https://grid.ifca.es/wiki/Supercomputing/Userguide. [Accedido: 26-jun-2019].
[56] «Slurm Workload Manager - Quick Start User Guide». [En línea]. Disponible en: https://slurm.schedmd.com/quickstart.html. [Accedido: 26-jun-2019].
[57] «The HDF Group Virtual Object Layer in HDF5 Exploring new HDF5 concepts May 30-31, 2012HDF5 Workshop at PSI ppt download». [En línea]. Disponible en: https://slideplayer.com/slide/7607091/. [Accedido: 26-jun-2019].
[58] «Lustre* Software Release 2.x». [En línea]. Disponible en: http://doc.lustre.org/lustre_manual.xhtml. [Accedido: 25-jun-2019].
[59] «xarray: N-D labeled arrays and datasets in Python — xarray 0.12.2 documentation». [En línea]. Disponible en: http://xarray.pydata.org/en/stable/. [Accedido: 30-jun-2019].
[60] The HDF-EOS Tools and Information Center, «HDF for the Cloud», 01:53:31 UTC.
[61] «HDF in the Cloud». [En línea]. Disponible en: http://matthewrocklin.com/blog/work/2018/02/06/hdf-in-the-cloud. [Accedido: 26-jun-2019].