Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Propuesta de refactorización del programa de interpolación para estudio de la calidad del aire.

Que Presenta:

Pedro Damián Cruz Santiago

Que para obtener el grado de:

Especialista en Cómputo de Alto Rendimiento

Tutor:

Dr. Oscar Alejandro Esquivel Flores

miembros del comité

Dr. José Jesús Carlos Quintanar Sierra

M. en I. Juan Luciano Díaz González

Ciudad Universitaria, CD. MX., 2022

Índice general

Índice de figuras

Índice de tablas

Propuesta de refactorización del programa de interpolación para estudio de calidad del aire.

Pedro Damián Cruz Santiago
Especialización en Cómputo de Alto Rendimiento
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Resumen

La calidad del aire afecta la salud de la población, el poder reducir la exposición a ésta también reduce sus efectos. La gestión de calidad del aire ayuda a alcanzar este objetivo, dentro de la gestión se tiene que una herramienta importante son los estudios de la calidad del aire, en donde se emplea información meteorológica, de cobertura de suelo, de topografía y emisiones. Para ciertos casos de estudio de calidad del aire la resolución espacial en las que se encuentran las emisiones no es la adecuada y se requiere realizar una interpolación de los datos para su uso. En el Centro de Ciencias de la Atmósfera y Cambio Climático `ICAyCC` se desarrollo un sistema de interpolación en lenguaje Fortran90 para el tratamiento de los datos de las emisiones, este sistema requiere de mantenimiento y mejoras los cuales son desarrollados en este trabajo. En el presente trabajo mediante la técnica de `refactorización` se realizaron cambios al código y se identificaron bloques donde se agregaron directivas `OpenMP` teniendo como resultado una aceleración de $17x$ en comparación con el programa original.

Capítulo 1

Introducción

Una buena calidad del aire se puede definir como el conjunto de concentraciones de sustancias presentes en la atmósfera en un periodo de tiempo dado que no causan daño a la salud, al bienestar de la población, al equilibrio ecológico o a los materiales de valor económico. Por otro lado, la contaminación atmosférica se refiere a la situación donde la concentración de dichas sustancias son suficientemente altas sobre los niveles normales en el ambiente, produciendo efectos nocivos en seres humanos, animales, vegetación o materiales [11]. La estimación de las características del aire mediante el uso de modelos de calidad del aire (MCA) es un área de investigación en el Instituto de Ciencias de la Atmósfera y Cambio Climático (ICAyCC)[1], los MCA son un conjunto complejo de modelos matemáticos para simular numéricamente las concentraciones de contaminantes, su transporte meteorológico y sus transformaciones fisicoquímicas.

WRF (*Weather Research and Forecasting*) [2] es el MCA utilizado en el ICAyCC para la publicación del pronóstico meteorológico[3] para todo el país y de calidad el aire para la Zona Metropolitana del Valle de México (ZMVM) [4], para este último se debe proporcionar un base de datos de emisiones de contaminantes que servirán como condiciones iniciales y de frontera en una configuración especial de *WRF* denominada `WRF-Chem` [5].

La base de datos de emisiones se debe encontrar en el formato `netCDF` [9] y para México es obtenido a partir del Inventario Nacional de Emisiones de México 2016 (INEM-2016) publicado por la Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT) [5], en caso de que el estudio de calidad del aire se realizará en un área menor, por ejemplo la ZMVM, entonces es necesario realizar un proceso de interpolación con conservación de masa [8] [2] de la base de datos de emisiones del dominio global , toda la república, al área o dominio de estudio . La interpolación debe realizarse para cada dominio de estudio y debido a sus las características, la cantidad de operaciones realizadas pueden llegar a ser del orden de 10 Billones.

En este trabajo se analizará la implementación computacional del algoritmo que lleva a cabo la interpolación con la finalidad de diseñar conjunto de estrategias computacionales de alto rendimiento que permitan acelerarlo y disminuir el tiempo de ejecución.

[1] https://www.atmosfera.unam.mx/
[2] https://www.mmm.ucar.edu/weather-research-and-forecasting-model
[3] http://grupo-ioa.atmosfera.unam.mx/pronosticos/index.php/meteorologia
[4] https://www.atmosfera.unam.mx/calidad-del-aire-en-mexico/
[5] https://www.gob.mx/semarnat/documentos/documentos-del-inventario-nacional-de-emisiones

1.1. Motivación

Dentro del ICAyCC se realizan muchos casos de estudio de calidad el aire: desarrollo de tesis, artículos científicos, pronostico de calidad del aire, entre otros. La finalidad de este trabajo es reducir el tiempo de ejecución de la etapa de interpolación beneficiando con esto los diferentes estudios de calidad del aire realizados en el instituto. Los modelos de calidad del aire MCA son una herramienta para estudiar el transporte y la dinámica de los contaminantes atmosféricos, ya que simulan los procesos físicos y químicos que influyen en su comportamiento y distribución[10]. Esto permite evaluar el estado de la calidad del aire en zonas de interés y adoptar las medidas de control necesarias para salvaguardar a la población y reducir su exposición a altas concentraciones de contaminantes. Para utilizar un MCA se debe proporcionar información sobre el área de estudio, esta información incluye las coordenadas geográficas que la delimitan, el periodo de tiempo a modelar y los principales contaminantes criterio generados en el área de estudio. La información sobre los contaminantes se obtiene a partir de la interpolación de la base de datos de emisiones para todo el país generada por el método DiETE [4].

1.2. Objetivos

Proponer una estrategia para acelerar el algoritmo de interpolación.

1.3. Objetivos particulares

Para lograr el objetivo general, se consideran los siguientes objetivos particulares:

- Analizar e identificar con la ayuda de un programa de perfilado las partes del programa que consumen la mayor parte del tiempo de ejecución.
- Proponer e implementar estrategias de aceleración de código sustituyendo rutinas secuenciales por rutinas paralelas en las partes del programa que consumen más tiempo durante la ejecución.
- Diseñar e implementar un marco experimental para presentar el análisis de rendimiento de los cambios propuestos.

1.4. Metodología

Para implementar una estrategia que sirva para acelerar el algoritmo de interpolación el primer paso es investigar el proceso de interpolación y su implementación en lenguaje Fortran[1], como siguiente paso una realizar revisión técnica de los formatos de archivo de entrada y de salida, como tercer paso llevar a cabo la ejecución del programa interpola bajo el control de un software de perfilado para identificar puntualmente las subrutinas propias o del sistema operativo donde toma más tiempo la ejecución. Como cuarto paso realizar modificaciones al código original para evaluar si hay reducción en el tiempo de ejecución sin que se vea afectada el funcionamiento general.

1.5. Metas

Se espera tener una nueva versión del programa de interpolación después de aplicar las diferentes propuestas de refactorización que presenten un aumento en su eficiencia y reducción en el tiempo de ejecución. El código elaborado será de carácter público, distribuido por medio del repositorio del programa `interpola` [6] para ser descargado y utilizado en diferentes estudios de calidad del aire llevadas a cabo en el ICAyCC.

1.6. Contribución y limitaciones

El uso del programa de interpolación esta limitado solo para la república Mexicana ya que el archivo del dominio de emisiones que sirve de entrada se obtiene a partir de aplicar el DiETE al INEM publicado por la SEMARNAT. La última publicación del INEM fue en el 2016 por lo que para una nueva publicación será necesario aplicar el modelo DiETE para obtener un nuevo archivo que sirva como nuevo dominio de general de emisiones y pueda ser utilizado por el programa de interpolación para nuevos dominios de estudio.

1.7. Organización

Este trabajo se organizado de la siguiente forma: en el capítulo 2 se describen el proceso de interpolación y el código mediante el que se implementa, las características del formato de archivo `netCDF` utilizado tanto para los archivos de entrada como de salida; así como un primer análisis de rendimiento o perfilado de cada subrutina. En el capítulo 3 se describen las intervenciones al código original y la evaluación de la ejecución del proceso de interpolación bajo esta refactorización. Se finaliza con el capítulo 4 el cual contiene las conclusiones, recomendaciones y trabajo por realizar.

[6] https://github.com/JoseAgustin/interpola

Capítulo 2

Estudio y evaluación del proceso de interpolación

En el presente capítulo se describe el proceso de interpolación, el código que lo implementa y las características principales de los datos y archivos de entrada. Principalmente se presenta un análisis de rendimiento por medio del cálculo de los tiempos de ejecución del algoritmo, así como de las subrutinas principales que lo componen, por último se describe la herramienta de perfilado utilizada para obtener un diagnóstico de la ejecución del programa de interpolación.

2.1. Emisiones globales y proceso de interpolación

Los trabajos de investigación que se llevan acabo en el ICAyCC respecto a los escenarios de la calidad del aire se apoyan en el uso del software `WRF-Chem` (Grell et al. 2005, Fast et al. 2006)[5] [1] el cual requiere como datos de entrada información de **emisiones globales** que servirá para determinar condiciones iniciales y de frontera. Esta base de datos de emisiones se obtiene a partir del Inventario Nacional de Emisiones de México (INEM) [2] publicado por la Secretaría de Medio Ambiente y Recursos Naturales [3] (SEMARNAT) en formato de valores separados por coma `csv` con una resolución temporal anual y una resolución espacial a nivel municipal.

Para hacer uso de la información contenida en el INEM, el ICAyCC desarrolló e software `DiETE`(Gracía-Reynoso et al. 2012)[4] para obtener una resolución temporal de emisiones por hora y una resolución espacial en una malla de emisiones globales que cubre toda la república mexicana compuesta por celdas de tamaño 9×9 kms llamado dominio de emisiones globales. Sin embargo, en algunos casos de estudio, la celda del dominio de modelación tiene dimensión diferente al de las celdas en el dominio de emisiones globales. Para esolver esta situación es necesario llevar acabo un proceso de interpolación de la malla de emisiones globales a la malla del dominio de estudio. El proceso de interpolación es aquel mediante el cual se asignan los datos de los contaminantes de las celdas `E` que componen la malla del dominio de emisiones

[1] https://ruc.noaa.gov/wrf/wrf-chem/

[2] El INEM comprende información de las emisiones liberadas a la atmósfera de los contaminantes considerados como criterio: monóxido de carbono (CO), óxidos de nitrógeno (NOx), óxidos de azufre (SOx) y partículas con diámetro aerodinámico menor a 10 y 2.5 micrómetros (PM10 y PM2.5), compuestos orgánicos volátiles (COV) y amonio (NH3).

[3] https://www.gob.mx/semarnat/acciones-y-programas/inventario-nacional-de-emisiones-de-contaminantes-criterio-inem

[4] https://github.com/JoseAgustin/emis_2016

globales a las celdas `D` que componen la malla del dominio de estudio o de interpolación. La malla del dominio de estudio puede ser mayor o menor a la malla del dominio global como se pude observar en la figura 2.1, en ambos casos y como resultado de sobreponer las malla del dominio de estudio a la malla de emisiones globales se tendrán celdas `E` que tengan área en común con celdas `D`, posteriormente se debe calcular el área o la fracción de área común y asignar el flujo del contaminante de la celda `E` a la celda `D` de manera proporcional al área en común. El cálculo de áreas se obtiene a partir de las coordenadas geográficas de cada celda.

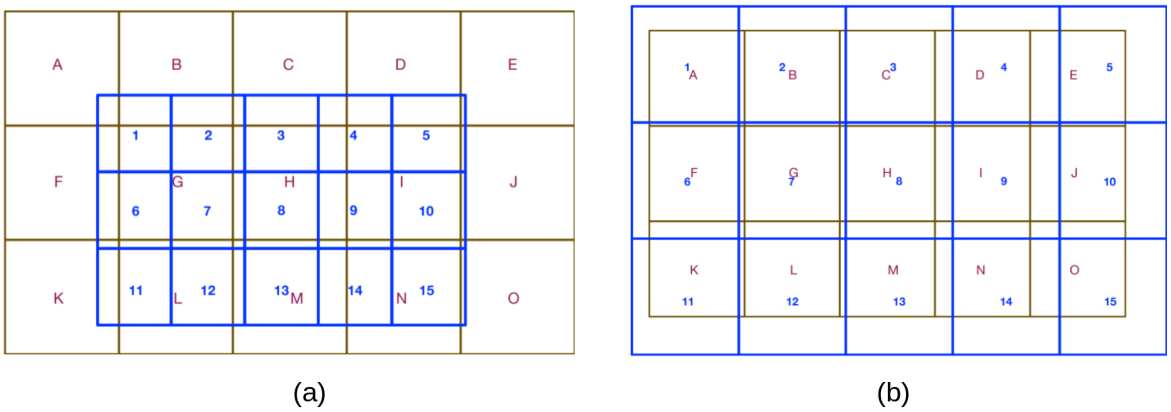

Figura 2.1: En (a) se muestra una malla de estudio en negro mayor al dominio de emisiones en azul. En (b) la malla del dominio de emisiones en negro es menor a la malla de emisiones globales.

En la figura 2.2 se puede observar el proceso de interpolación de una celda del dominio de emisiones `E` hacia 4 celdas del dominio de estudio `D`, los valores de las coordenadas para las celdas de ambos dominios y los datos correspondientes a los contaminantes del dominio global son leídos por el programa de interpolación de los dos archivos de entrada.

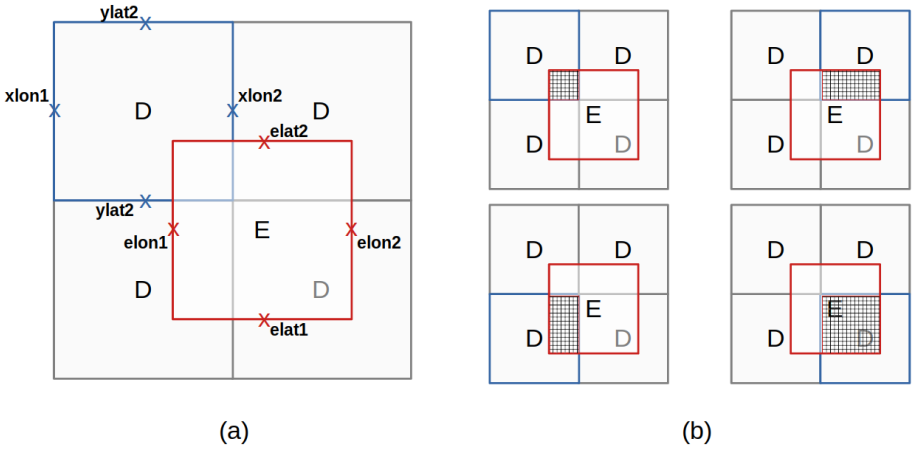

Figura 2.2: El contenido de una celda de emisiones E debe distribuirse proporcionalmente al área común con las celdas del dominio de interpolación D.

El ICAyCC sistematizó el procedimiento previamente descrito y desarrolló el software

`Interpola`.

2.2. El programa de interpolación `Interpola`

`Interpola` es el nombre que recibe el software que lleva acabo el proceso de interpolació[5] el cual se compone de siete archivos de código fuente escritos en lenguaje de programación `Fortran90` y comprende las siguientes etapas:

1. Lee malla de emisiones: archivo `wrfchemin.nc`
2. Lee malla de estudio: archivo `wrfinput`
3. Realiza la interpolación
4. Guarda malla de estudio con emisiones: archivo `wrfdom0.nc`.

Los dos archivos de entrada y el archivo de salida se encuentra en formato de almacenamiento de datos científicos: *network Common Data Format* `netCDF` (Rew y Davis 1990)[9].

2.2.1. Formato de archivo netCDF

`NetCDF` es un tipo de dato abstracto que funciona como formato de archivo y que junto a una interfaz de programación de aplicación (API) y una biblioteca que implementa la API son desarrollados y mantenidos por la UNIDATA [6]. El tipo de almacenamiento de la información en es en forma matricial, un archivo en formato `netCDF` se divide en 2 partes: la cabecera y los datos. La cabecera tiene la información sobre los datos almacenados, esta *"metadata"* consta de dimensiones, variables y atributo. Las dimensiones sirven para especificar la forma de las variables por ejemplo tiempo, latitud, longitud, altitud, temperatura, etc. Se tratan de valores enteros y solo se permite una dimensión sin límite que sirve como registro de los datos. Las variables representan arreglos de valores del mismo tipo que tienen un nombre, un tipo de dato y una lista de dimensiones, ejemplos de variables en `netCDF` son: temperatura-superficie, humedad-relativa, hora, velocidad del viento. Los atributos sirven para representar información sobre una variable o sobre todo el archivo tienen un nombre, un tipo de dato, una longitud y uno o más valores, ejemplos de atributos de una variable son: velocidad:unidades $= m/seg$, temperatura:unidades: "Celsius", longitud: valid-range: -100,70 mientras que los datos son los arreglos con la información documentada en la cabecera. La representación a nivel byte de los datos almacenados son independientes del sistema donde fueron escritos, para lograr esto `netCDF` utiliza el formato estándar no propietario *eXternal Data Representation* desarrollado por Sun Microsystems `XDR`(RFC 1014) [7], en la tabla 2.1 se muestran las variantes actuales y sus características.

El formato de datos utilizado por `netCDF` para el almacenamiento interno de la información se divide en el modelo clásico cuya principal característica es el uso del formato `XDR` y es utilizado por las variantes `CDF-1`, `CDF-2`, `CDF-4`, `CDF-5` [8]. El formato mejorado utiliza como

[5] https://github.com/JoseAgustin/interpola

[6] UNIDATA forma parte de la *University Corporation for Atmospheric Research* (UCAR), mantiene y da soporte a las interfaces para los lenguajes `C, Fortran, C++` y `Java` mediante la bibliotecas `netcdf-c, netcdf-fortran, netcdf-cxx4` y `netcdf-java` respectivamente. Más información sobre el formato `netCDF` en la documentación oficial: https://www.unidata.ucar.edu/software/netcdf/docs/index.html

[7] https://datatracker.ietf.org/doc/html/rfc1014

[8] El formato `CDF-3` oficialmente no existe pero es utilizado en la literatura para hacer referencia al modelo de datos clásico `CDF-1, CDF-2, CDF-5`.

Variante	Año	Características principales
Classic Format (`CDF-1`)	1989	Formato "default"para la creación de nuevos archivos. Máximo 2GB para arreglos asociados a una variable.
64-bit Offset Format (`CDF-2`)	2004	Máximo 4GB para arreglos asociados a un variable, funcionamiento en arquitecturas de 32 y 64 bits.
`netCDF-4` format	2008	Uso de un subconjunto de características del formato `HDF5` para el almacenamiento.
`netCDF-4` classic model format	2008	Permitir que aplicaciones que escriben o leen formatos `CDF-1, CDF-2` puedan usar el formato `CDF-4`
64-bit Data Format (`CDF-5`)	2016	Uso de enteros de 64 bits para los datos y los índices de los arreglos

Tabla 2.1: Una de las características principales del formato `netCDF` es garantizar la lectura de los archivos de formatos previos con las nuevas versiones de la biblioteca.

almacenamiento interno `HDF5`[9] que a diferencia de `XDR` en donde los tipos de datos como enteros y flotantes son almacenados utilizando agrupamientos de 4 bytes en un formato propio, `HDF5` hace uso del almacenamiento `little` y `big-endian` dependiendo la arquitectura donde son generados los datos.

La biblioteca `netcdf-c` proporciona tres herramientas que sirven para trabajar con archivos `netCDF` desde la línea de comandos `ncdump, ncgen, nccopy`. La herramienta `ncdump` lee un archivo `netCDF` y muestra su contenido en ASCII en un formato llamado `CDL` [10], `ncgen` lee un archivo de texto ASCII en formato `CDL` y genera el archivo `netCDF` correspondiente, `nccopy` toma un archivo `netCDF` y lo copia a otro archivo `netCDF` cambiando la variante, compresión y otras características de almacenamiento. Adicionalmente existen herramientas de terceros para la manipulación y visualización de archivos `netCDF`, para una lista extensa mantenida por UNIDATA [11].

Las bibliotecas utilizadas en el presente trabajo facilitarán la manipulación del `netCDV` *Operator* o `NCO`[12] y para la visualización con `ncview`[13].

2.2.2. Archivos de entrada para interpolación

Archivo de emisiones globales. El archivo base de datos de emisiones `wrfchemin.nc` se encuentra en formato `netCDF` y contiene la distribución espacial y temporal del INEM que publica la SEMARNAT en formato `csv`. A continuación se muestra la estructura del archivo `wrfchemin.nc`, resultado de hacer la consulta por medio de la herramienta `ncdump`:

[9] Desarrollada desde 1998, *Hierarchical Data Format 5* consiste de una especificación así como su implementación en lenguaje C, C++, Fortran90 y Java, para más información remitirse a la página del proyecto https://www.hdfgroup.org/solutions/hdf5/

[10] `CDL` (*Common Data Language*) es una notación legible por el humano sobre la estructura y los datos en un archivo `netCDF`, forma parte de la especificación `netCDF` (https://www.unidata.ucar.edu/software/netcdf/workshops/most-recent/nc3model/Cdl.html)

[11] https://www.unidata.ucar.edu/software/netcdf/software.html

[12] http://nco.sourceforge.net/

[13] http://meteora.ucsd.edu/~pierce/ncview_home_page.html

```
dimensions:
        Time = UNLIMITED ; // (12 currently)
        DateStrLen = 19 ;
        west_east = 1059 ;
        south_north = 677 ;
        bottom_top = 1 ;
        emissions_zdim_stag = 8 ;
variables:
        char Times(Time, DateStrLen) ;
        float XLONG(Time, south_north, west_east) ;
                XLONG:FieldType = 104 ;
                XLONG:MemoryOrder = "XYZ" ;
                XLONG:description = "LONGITUDE, WEST IS NEGATIVE" ;
                XLONG:units = "degree_east" ;
                XLONG:axis = "X" ;
        float XLAT(Time, south_north, west_east) ;
                XLAT:FieldType = 104 ;
                XLAT:MemoryOrder = "XYZ" ;
                XLAT:description = "LATITUDE, SOUTH IS NEGATIVE" ;
                XLAT:units = "degree_north" ;
                XLAT:axis = "Y" ;
        float E_CO(Time, emissions_zdim_stag, south_north, west_east) ;
                E_CO:FieldType = 104 ;
                E_CO:MemoryOrder = "XYZ" ;
                E_CO:description = "Emissions rate of Carbon Monoxide" ;
                E_CO:units = "mol km^-2 hr^-1" ;
                E_CO:stagger = "Z" ;
                E_CO:coordinates = "XLONG XLAT" ;
```

El archivo tiene definidas 6 dimensiones `Time, DateStrLen, west:east, south_north, bottom_top` y `emissions_zdim_stag`. En el listado solo muestran las variables `Times`, `XLONG`, `XLAT`, `E_CO` y sus atributos. La dimensión `Time` no tiene límite pero se reportan 12 registros almacenados en el archivo; en total el archivo tiene definidas 46 variables, 39 de ellas corresponden a los diferentes conjunto de contaminantes criterio obtenidos del INEM, un listado completo del archivo `wrfchemin.nc` puede consultarse en el apéndice A.

De la información anterior se puede deducir que se trata de un dominio compuesto por $1059x677x8$ celdas cada una de ellas con datos asociados a 39 contaminantes para 12 registros de tiempo.

Visualización. El procedimiento para visualizar el contenido del archivo `wrfchemin.nc` es abrir el visor y seleccionar de forma interactiva alguna de las variables contenidas en el archivo, `ncview` mostrará todos los datos asociados a esa variable para todas las celdas, en la figura 2.3 se puede observar la interfaz del visualizador para una variable contenida en el archivo.

Consultas especializadas. Otra herramienta utilizada en el presente trabajo para el manejo de archivos `netCDF` es `ncks` (`netcdf` *kitchen sink*) que forma parte del software `netCDF` *Operator* `NCO`, permite en este caso consultar desde línea de comandos el valor de una variable para las dimensiones dadas. Por ejemplo, para consultar el valor de la variable `E_CO` (monóxido de carbono) en

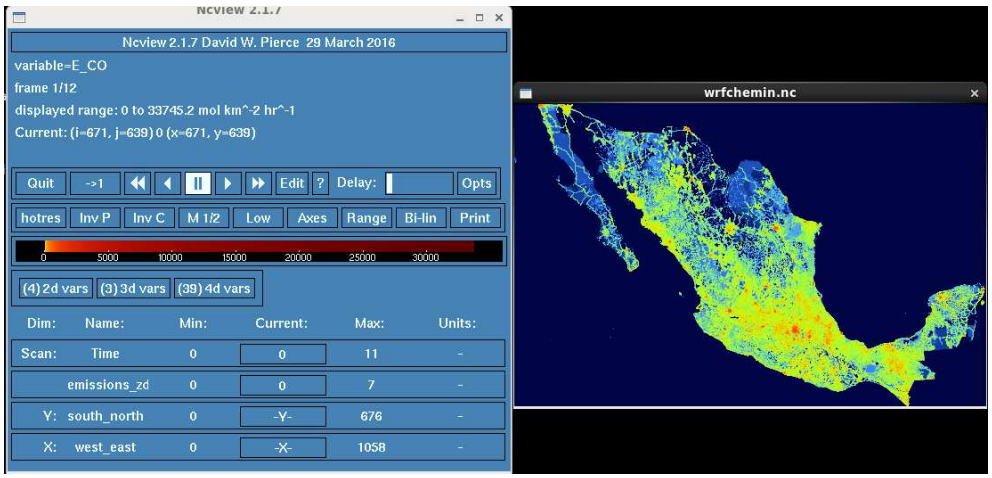

Figura 2.3: Ejemplo de uso del visualizador `ncview`, se muestran los valores del registro 0 (variable Time=0)

la celda `( emissions_zdim_stag, south_north, west_east ) = 0, 166, 628)` para todos los registros `Time = 0,..,11` el comando es el siguiente: `ncks -d south_north,166 -d west_east,628 -d emissions_zdim_stag,0 -v E_CO wrfchemin.nc`.

```
  dimensions:
    Time = UNLIMITED ; // (12 currently)
    emissions_zdim_stag = 1 ;
    south_north = 1 ;
    west_east = 1 ;
...
 data:
    E_CO =
    666.8818,
    643.1808,
    605.099,
    554.3694,
    439.5999,
    408.4514,
    363.0704,
    336.6933,
    322.3314,
    308.5619,
    330.6571,
    453.2018 ;
...
}
```

El resultado de la consulta con la herramienta `ncks` se encuentra en formato `CDL` (*Common Data Language*) lo que permite construir otro archivo `netCDF` con la herramienta `ncgen`, en este caso se muestran los datos contenidos en una celda para 12 registros que corresponden según los atributos del archivo a un valor por hora en el rango 0 a 11 hrs. El visualizador

`ncview` nos permite realizar la misma consulta de forma interactiva seleccionando un punto en el mapa de visualización, en la imagen 2.4 se muestra la consulta para la misma celda `( emissions_zdim_stag, south_north, west_east ) = 0, 166, 628)`

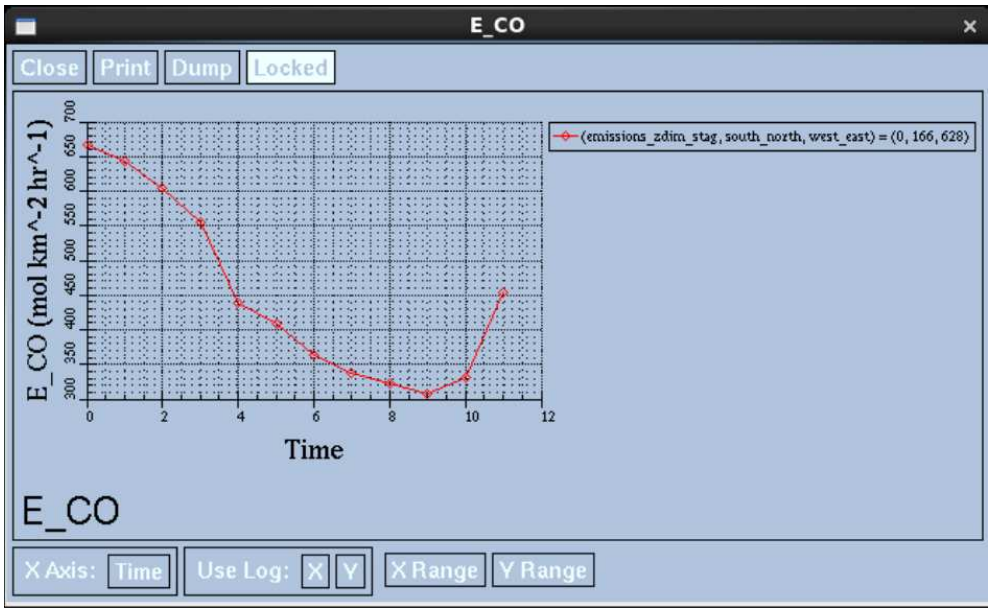

Figura 2.4: Se muestran el resultado de seleccionar un punto en el ventana de visualización, en esta caso los 12 registros (variable Time=0..11) para la variable `E_CO`

Como último punto respecto al archivo de emisiones globales con el uso de la herramienta `ncdump` consultamos la variante del formato `netCDF` y no reporta que se trata de un archivo `CDF-1` lo que significa el uso de formato de datos clásico.

2.2.3. Archivo de dominio de interpolación

El archivo de interpolación `wrfinput` contiene el dominio de estudio hacia el cual se debe realizar la interpolación a partir del archivo `wrfchemin.nc`, como resultado de la interpolación se genera el archivo `wrfchemi_00z_d01` que junto con el archivo `wrfinput` es utilizado por el modelo `WRF-Chem` para llevar a cabo una simulación. Se muestran a continuación las dimensiones contenidas en el archivo `wrfinput`, este archivo tiene definidas 313 variables:

```
dimensions:
Time = UNLIMITED ; // (1 currently)
DateStrLen = 19 ;
west_east = 150 ;
south_north = 100 ;
bottom_top = 33 ;
dust_erosion_dimension = 3 ;
bottom_top_stag = 34 ;
soil_layers_stag = 4 ;
west_east_stag = 151 ;
south_north_stag = 101 ;
DIM0010 = 5 ;
```

```
land_cat_stag = 24 ;
soil_cat_stag = 16 ;
num_ext_model_couple_dom_stag = 1 ;
```

El archivo `wrfinput` para este dominio de interpolación contiene 100 × 150 celdas correspondiente a las dimensiones `south_north`, `west_east`. Para este caso el dominio de interpolación comprende las coordenadas `(lat=16.92754, lon=-102.5687)` `(lat=21.29327, lon=-95.39038)` que corresponde a los valores almacenados para la primera y última celda respectivamente.

2.3. Diagnóstico

2.3.1. Estructura del programa

El programa de interpolación `Interpola` se compone de un archivo principal `Interpola.F90` y seis archivos con la implementación de las subrutinas llamadas desde el archivo principal. Su instalación requiere de la biblioteca `netcdf-fortran`[14] configurada para el soporte del formato mejorado `CDF-4 (HDF5)`[15].

La estructura del archivo principal, mostrada en el listado 2.2, refleja los pasos de la interpolación descrita previamente: leer archivo con datos del dominio de emisiones globales, leer archivo con datos del dominio de estudio, realizar los cálculos de interpolación, escribir los resultados.

```
program Interpola

    Call reads_emision
    Call reads_grid
    Call conversion
    Call file_out

end program Interpola
```

Lista 2.2: Contenido del archivo Interpola.F90

Los seis archivos restantes que forman parte del programa son:

1. `lee_emis.F90`: Implementa la subrutina `reads_emision` encargada de leer el archivo `wrfchemin.nc` que contiene las emisiones para un dominio en específico.

2. `lee_malla.F90`: Implementa la subrutina `reads_grid` encargada de leer el archivo `wrfinput` que contiene el dominio donde las emisiones debe interpolarse.

3. `calculos.F90`: Implementa la subrutina `conversion` encargada de realizar los cálculos de interpolación de las emisiones.

4. `salidas.F90`: Implementa la subrutina `File_out` encargada de generar el archivo en formato `netcdf` con las emisiones interpoladas al dominio de estudio

[14] https://github.com/Unidata/netcdf-fortran
[15] https://github.com/HDFGroup/hdf5

5. `s_check.F90`: Implementa la subrutina `check` utilizada en cada llamada a funciones de lectura o escritura de los archivos `netcdf` y en caso de error terminar la ejecución del programa interpola.

6. `mod_vars_dat.F90`: Implementa el módulo `vars_dat` que contiene variables comunes a todas las subrutinas del programa interpola.

Para fines del presente trabajo se llevo a cabo la instalación del programa `Interpola` siguiendo las instrucciones del repositorio, en un equipo de cómputo con 16 cores Intel(R) Xeon(R) CPU E5-2670, con el sistema operativo GNU/Linux distribución CentOS 6.10 utilizando los compiladores GNU Compiler Collections `GCC` versión 9.0.[16]. En en apéndice **??** se puede consultar los comandos realizados para la instalación del programa de interpolación.

2.3.2. Tiempos de ejecución

Ejecución Global. Con la ayuda del programa `time` [17] obtenemos el tiempo de reloj de la ejecución del programa Interpola: 2.2

Command being timed:	"./interpola.exe"
User time (seconds):	124.71
System time (seconds):	9.66
Elapsed (wall clock) time (h:mm:ss or m:ss):	2:14.37

Tabla 2.2: Resultado de ejecutar el programa de interpolación con el herramienta time, se muestran solo las líneas relacionadas con el tiempo de ejecución que fue de 134.37 segundos.

Ejecución por subrutina. Para identificar las partes del programa en las cuales la ejecución toma más tiempo se procede a la modificación del programa principal para obtener una aproximación del tiempo que toma a cada subrutina ejecutarse. Se definieron las variables `T1 y T2` de tipo `real` para ser utilizadas con el procedimiento interno `CPU_TIME()` [18]. Para obtener el tiempo que toma la ejecución se agrego una linea con la llamada al procedimiento `CPU_TIME()` con argumento la variable `T1` antes de la llamada a la subrutina y como argumento la variable `T2` inmediatamente después llamada a la subrutinas, el resultado `T2 - T1` es el tiempo que tomo la ejecución de cada subrutina.

El archivo principal `Interpola.F90` después de insertar las líneas para medir el tiempo por subrutina se muestra continuación:

```
program Interpola
real :: T1,T2,emiss,grid,conv,fout

    Call cpu_time(T1)
    Call reads_emision
    Call cpu_time(T2)
```

[16] https://gcc.gnu.org/

[17] La herramienta `time` es utilizada para ejecutar otro programa y al termino de éste reportan información sobre los recursos utilizados, entre otros el tiempo de reloj o `wall clock time`. Más información en la página del manual https://man7.org/linux/man-pages/man1/time.1.html

[18] https://gcc.gnu.org/onlinedocs/gfortran/CPU_005fTIME.html

```
    emiss = T2-T1

    Call cpu_time(T1)
    Call reads_grid
    Call cpu_time(T2)
    grid = T2-T1

    Call cpu_time(T1)
    Call conversion
    Call cpu_time(T2)
    conv = T2-T1

    Call cpu_time(T1)
    Call File_out
    Call cpu_time(T2)
    fout = T2-T1

    print *,'READS_EMISION:: ',emiss
    print *,'READS_GRIDS:: ',grid
    print *,'CONVERSION:: ',conv
    print *,'FILE_OUT:: ',fout
    print *,'TOTAL:: ', emiss + grid + conv + fout

end program Interpola
```

Lista 2.3: Modificaciones realizadas al archivo `Interpola.F90` para reportar el tiempo por subrutina.

El tiempo de ejecución del programa `interpola` con las modificaciones anteriores se reportan en la tabla 2.3.

Subrutina	tiempo segundos	%
READS_EMISION::	34.2657	25.75
READS_GRIDS::	0.0989	0.08
CONVERSION::	97.5011	73.28
FILE_OUT::	1.1808	0.89
TOTAL::	133.0467	100

Tabla 2.3: Tiempos de ejecución del programa de interpolación por subrutina después de incluir directivas llamadas al procedimiento `CPU_TIME()`.

2.3.3. Herramienta de perfilado `Perf`

`Perf` es una herramienta de perfilado disponible para versiones del kernel `Linux 2.6` o posteriores[19] que no requiere la modificación o recompilación del código fuente del programa a perfilar. Los eventos que mide `Perf` son de software (kernel o aplicaciones de usuario) y de hardware (generados en el procesador), para ambos `Perf` permite el reporte total de la ocurrencia de los eventos `"Counting"` y el reporte de muestreo de eventos `"Sampling"`. [20].

Para obtener un reporte por muestreo, el primer paso es la ejecución por medio de `Perf` con la opción `record` seguido del programa a perfilar, en este paso la herramienta `Perf` registra por default $1,000$ muestras por segundo en el archivo `perf.data` donde cada una de estas muestras corresponde a la información del puntero de instrucción *instruction pointer* (IP) que indica en donde se encuentra la ejecución del programa en el momento de la muestra[21].

El segundo paso es ejecutar de nuevo la herramienta `Perf` para leer el contenido del archivo `perf.data` y generar el reporte solicitado. El archivo `perf.data` tiene registros del puntero de instrucción en rutinas del programa, bibliotecas compartidas utilizadas por el programa así como llamadas al sistema *system calls*[22]. En la figura 2.5 se puede observar el reporte generado por la herramienta `Perf`, la columna `Samples` contiene la cantidad de muestras que fueron tomadas durante la ejecución de la subrutina de la columna `Symbol` que se encuentra dentro del ejecutable de la columna `Shared Object`. La columna `Overhead` contiene el porcentaje del total de muestras registradas durante la ejecución del programa para cada subrutina listada en el reporte. En en apéndice C se puede consultar los comandos utilizados para la ejecución de la herramienta +Perf+.

Samples	Overhead	Symbol	Shared Object
410146	75.41%	[.] conversion_	interpola.exe
69509	12.72%	[.] swapn4b	libnetcdf.so.15.1.0
59484	10.90%	[.] reads_emision_	interpola.exe
1067	0.19%	[.] file_out_	interpola.exe
104	0.02%	[.] reads_grid_	interpola.exe
77	0.01%	[.] fill_NC_var	libnetcdf.so.15.1.0
20	0.00%	[.] H5SL_search	libhdf5.so.103.1.0
10	0.00%	[.] H5SL_insert_common	libhdf5.so.103.1.0
7	0.00%	[.] memcpy@plt	libnetcdf.so.15.1.0
4	0.00%	[.] px_get	libnetcdf.so.15.1.0
3	0.00%	[.] H5O_sdspace_shared_decode	libhdf5.so.103.1.0
3	0.00%	[.] H5FL_blk_find_list	libhdf5.so.103.1.0
3	0.00%	[.] H5FL_reg_free	libhdf5.so.103.1.0
3	0.00%	[.] H5_hash_string	libhdf5.so.103.1.0
3	0.00%	[.] nc_unsafe_get_property	libnetcdf.so.15.1.0
2	0.00%	[.] H5O_dtype_decode_helper	libhdf5.so.103.1.0

Figura 2.5: En el reporte resultado del perfilado del programa `interpola.exe` se observa que el porcentaje mayor de muestras fueron tomadas durante la ejecución de las subrutinas `conversion`, `swap4b` y `reads_emision`.

[19] A partir de la versión 2.6 del kernel se desarrolló un nuevo subsistema para agrupar todo lo referente al análisis del rendimiento llamado *Performance Counters for Linux* (PCL) la cual proporciona una abstracción para exponer un amplio conjunto de características de medición: contadores de eventos de hardware y software, muestreo y seguimiento de eventos. https://lwn.net/Articles/337493/

[20] https://perf.wiki.kernel.org/index.php/Tutorial

[21] La frecuencia de toma de las muestras así como el evento a registrar puede ser personalizado mediante opciones del programa. Para más información con remitirse a la página del manual https://www.man7.org/linux/man-pages/man1/perf-record.1.html

[22] https://www.man7.org/linux/man-pages/man1/perf-report.1.html

2.4. Evaluación de rendimiento

Agregar líneas de código para medir el tiempo por subrutina reportó que el 73.28 % de la ejecución ocurre en `conversion`, el 25.75 % en `reads_emision` y el 1 % restante en `read_grids` y `file_out`, por otro lado el porcentaje de muestras tomadas por la herramienta de perfilado es proporcional al tiempo de ejecución de la subrutina `conversion` 75.41 % pero en el caso de la subrutina `read_emision` solo el 10.90 % de las muestras fueron tomadas durante su ejecución y aparece una subrutina `swapn4b` que participa de manera significativa con un 12.72 % de las muestras. Una pista sobre el origen la subrutina `swapn4b` se obtuvo del mismo reporte del perfilado debido a que en la columna `Shared Object` de la imagen 2.5 se muestra la biblioteca a la que forma parte: `libnetcdf.so.15.1.0`.

La subrutina `read_emision` es la encargada de leer los datos de emisiones globales contenidas en el archivo `wrfchemin.nc`, que en el caso del archivo utilizados para este trabajo corresponden a 12 registros de 39 contaminantes para 1059 latitudes, 677 longitudes y 8 niveles en la vertical, mientras que la subrutina `conversion` realiza los cálculos de interpolación del dominio global al dominio de interpolación. Considerando que las coordenadas geográficas de las celdas que conforman la malla del dominio de interpolación son leídas en la subrutina `reads_grids` a partir del archivo `wrfinput`, 100×150 celdas en el archivo utilizado en este trabajo, se realizan $12 \times 39 \times 1059 \times 677 \times 8 = 2,684,234,592$ operaciones de comparación para determinar las áreas comunes entre mallas y $12 \times 39 \times 100 \times 150 \times 8 = 56,160,000$ operaciones de asignación de contaminantes.

El alto porcentaje de muestras reportada en la subrutina `conversion` por la herramienta de perfilado puede ser ocasionada por el uso de 4 ciclos `for` anidados para recorrer las celdas de los dominios y posteriormente por el uso de otros 4 ciclos anidados para llevar a cabo la asignación una celda a otra, esta asignación puede llevarse de forma independiente entre las 2 celdas con área común y esa parte del código es un candidato a ser refactorizada de tal forma que la asignación se ejecute de forma paralela.

En el caso de los ciclos utilizados para recorrer las celdas de ambos dominios son candidatos por ejemplo a recorrer de forma paralela un conjunto de latitudes o longitudes del dominio de emisiones para encontrar aquellas celdas del dominio de interpolación con las que tengan un área en común. Otra de las subrutinas principales del programa de interpolación reportada con un gran número de muestras durante el perfilado es `read_emision`, encargada de leer la base de datos de las emisiones globales, y la subrutina `swapn4b` que al ser parte de la biblioteca `netcdf` es posible que haya sido llamada por `read_emision` durante el proceso de lectura del archivo.

Las características del formato `netCDF` permiten el acceso directo a un registro en específico sin llevar a cabo una lectura secuencial desde el inicio del archivo, pero en este caso se realiza la lectura de todos los registros contenidos en el archivo. Estudiaremos el código de esta rutina para determinar el motivo del gran número de llamadas a la subrutina `swapn4b` y si es posible realizar cambios para disminuirlas.

2.5. Resumen

Es este capítulo se revisaron las características principales del formato `netCDF`, formato en el que se encuentran los archivos de entrada utilizados por el programa de interpolación. Se reportan los tiempos que toman cada una de las 4 subrutinas del programa de interpolación identificando que el 25.75 % es utilizado en la lectura del archivo con la base de datos de las emisiones y el 73.28 en las operaciones de la interpolación. También se reporta el perfilado de

la ejecución del programa de interpolación donde el 75.41 % de la muestras fueron tomadas durante la ejecución de la subrutina `conversion` y 23.62 % en la subrutina `read_emision` de las cuales más del 50 % de las muestras ocurrieron durante la ejecución de la subrutina `swapn4b` contenida en la biblioteca `libnetcdf`.

Capítulo 3

Refactorización y propuesta de paralelización

3.1. Introducción

En este capítulo se utilizará la técnica de refactorización [3] el código fuente. Esta técnica consiste en la reescritura de código aplicando cambios específicos sin afectar la naturaleza funcional de los procedimientos, aquí se presentan las modificaciones a las subrutinas `read_emision` y `conversion` que fueron identificadas en el perfilado con mayor consumo de tiempo. Posteriormente se presenta una propuesta de paralelización para memoria compartida utilizando directivas `OpenMP` [1] para la subrutina `conversion.`

Finalmente se utiliza como métrica de desempeño el tiempo de ejecución [7] obtenido después de la refactorización y el uso de las directivas `OpenMP`.

3.2. Reescritura de subrutinas

Con base en los resultados del perfilado del programa `Interpola` se llevará a cabo una revisión del código de la subrutina `read_emision` para identificar los ciclos que realizan la lectura del archivo de emisiones globales haciendo cambios al código y volver a ejecutar el programa para medir los efectos. Posteriormente se realizará el mismo análisis en la subrutina `conversion`, que consume la mayor parte del tiempo durante la ejecución, identificando posibles segmentos de código principalmente ciclos anidados para insertar directivas `OpenMP` para su ejecución en paralelo.

3.2.1. Fusión de ciclos subrutina `read_emision`

La subrutina `read_emision` es la encargada de leer los datos correspondientes a conjuntos de emisiones contenidos en el archivo `wrfchemin.nc`, al examinar el código fuente se identificaron las líneas que llevan a cabo la lectura mediante dos ciclos:

[1] El conjunto de directivas para el compilador, las bibliotecas y las variables de entorno definen la especificación `OpenMP API` para el paralelismo en programas `C, C++ y Fortran`. Las directivas extienden los lenguajes `C, C++ y Fortran` para la construcción de programas `SPMD (single program miltiple data)`. Las variables de entorno permiten el control en tiempo de ejecución el comportamiento del programa. Para más información sobre la especificación `OpenMP` remitirse a la página http://www.openmp.org

1. **Ciclo de identificación de variables correspondientes a emisiones.** En este ciclo se identifican las variables definidas en el archivo `wrfchemin.nc` cuyo nombre comienza con `E_` o `e_` que corresponden a variables asociadas a un conjunto de emisiones.

```
1
2  do ikk=1,nvars
3     call check(nf90_inquire_variable(ncid,ikk,name))
4     ename(ikk)=trim(name)
5     if(name(1:2).eq."E_".or.name(1:2).eq."e_") then
6        if(trim(name).eq."E_CO".or.trim(name).eq."e_co")  L_CO=ikk
7        call check( nf90_get_att(ncid, ikk, "description", name))
8        cname(ikk)=trim(name)
9        call check( nf90_get_att(ncid, ikk, "units", name))
10       cunits(ikk)=trim(name)
11       tvar(ikk)=.true.
12    end if
13 end do
```

Listado 3.1: Código original para obtener las variables en el archivo asociadas a un contaminante

En el listado 3.1 línea 1 la variable `nvars` tiene un valor que corresponde al número de variables definidas en el archivo `netcdf`. En la línea 2 se llama a la función `nf90_inquire_variable` para obtener el nombre de cada variable, si la variable corresponde a un elemento del inventario emisiones, línea 5, se llama la función `nf90_get_att` en la línea 7 para obtener la descripción de la variable y las unidades de la misma, línea 9. Finalmente, en la línea 11 se almacena en el arreglo `tvar` el valor `.true.` indicando que el número de variable `ikk` corresponde a una variable de emisión, este valor es utilizado en el segundo ciclo para obtener los datos de cada variable.

2. **Ciclo de lectura de los datos asociados a la variables de emisión**. En este ciclo se lleva a cabo la lectura de los datos correspondientes a cada una de las variables identificadas en el primer ciclo cuyo nombre comienza con `E_CO` o `e_co`:

```
1  do ikk=1,nvars
2     if(tvar(ikk)) then
3        call check(nf90_get_var(ncid, id_var(ikk),ea))
4        do i=1, west_east
5           do j=1, south_north
6              do l=1,emissions_zdim_stag
7                 do it=1,dim(1) !times in file
8                    ei(i,j,l,it,ikk)=ea(i,j,l,it)
9                 end do
10             end do
11          end do
12       end do
13    end if
14 end do
```

Listado 3.2: Código original para obtener las datos de las variables en el archivo asociadas a un contaminante

En el listado 3.2 línea 2 si la variable `ikk` tiene el valor verdadero `.true.`, se trata de una variable asociada a emisiones, entonces se llama a la subrutina `nf90_get_var`[2], línea 3, para almacenar en el arreglo `ea` los datos la variable identificada internamente dentro del archivo `netcdf` como `id_var(ikk)`[3]. Posteriormente, en la línea 8 se asignan los valores del arreglo `ea` al arreglo `ei` elemento a elemento por medio de 4 ciclos `DO`. Las líneas 4 a 12 correspondientes a la asignación del arreglo `ea` al arreglo `ei` para cada variable `ikk`, pueden ser sustituidas por una asignación directa utilizando notación de Fortran90: `ei(:,:,:,:,ikk)=ea` [4] con lo que se tiene:

```
1 do ikk=1,nvars
2    if(tvar(ikk)) then
3       call check(nf90_get_var(ncid, id_var(ikk),ea))
4       ei(:,:,:,:,ikk)=ea(:,:,:,:)
5    end if
6 end do
```

Listado 3.3: Código resultante después de utilizar la instrucción de asignación directa para matrices en Fortran

Después de estos cambios se identifica la posibilidad de fusionar el primer y segundo ciclo ya que en éste último se comprueba si la variable `tvar(ikk)` tiene valor verdadero y este valor es asignado en el primer ciclo si el nombre de la variable comienza con `E_` o `e_`, listado 3.4

```
1  do ikk=1,nvars
2    call check(nf90_inquire_variable(ncid,ikk,name))
3    ename(ikk)=trim(name)
4    if(name(1:2).eq."E_".or.name(1:2).eq."e_") then
5      if(trim(name).eq."E_CO".or.trim(name).eq."e_co")  L_CO=ikk
6      call check( nf90_get_att(ncid, ikk, "description", name))
7      cname(ikk)=trim(name)
8      call check( nf90_get_att(ncid, ikk, "units", name))
9      cunits(ikk)=trim(name)
10     call check(nf90_get_var(ncid, id_var(ikk),ea))
11     ei(:,:,:,:,ikk)=ea
12     tvar(ikk)=.true.
13   end if
14 end do
```

Listado 3.4: Código resultante después fusionar las instrucciones de los 2 ciclos utilizados para la lectura de los contaminantes a partir del archivo de entrada.

Las líneas 10 y 11 (listado 3.4) fueron tomadas del segundo ciclo y agregadas dentro del bloque de la condición `if`, línea 4, condición que es verdadera para el nombre de la variable que

[2]En este caso los parámetros con los que es llamada `nf90_get_var` son: `ncid`=descriptor del archivo `netcdf` abierto previamente, `id_var(ikk)=identificador` de la variable `ikk` en el archivo, `ea=arreglo` de 4 dimensiones para guardar los datos a leer

[3]El API de acceso a los datos de un archivo `netcdf` impone dos operaciones de lectura para obtener la información: a) Obtención del identificador de la variable en el archivo, b) Obtención de los datos en un arreglo *n-dimensional*.

[4]https://www.fortran90.org/src/best-practices.html#multidimensional-arrays

Samples	Overhead	Symbol	Shared Object
391226	83.68%	[.] conversion_	interpola.exe
65496	13.94%	[.] swapn4b	libnetcdf.so.15.1.0
6130	1.30%	[.] reads_emision_	interpola.exe
1019	0.21%	[.] file_out_	interpola.exe
101	0.02%	[.] reads_grid_	interpola.exe
66	0.01%	[.] fill_NC_var	libnetcdf.so.15.1.0
11	0.00%	[.] H5SL_search	libhdf5.so.103.1.0
8	0.00%	[.] memcpy@plt	libnetcdf.so.15.1.0
7	0.00%	[.] H5SL_insert_common	libhdf5.so.103.1.0
4	0.00%	[.] nc_utf8proc_decompose_custom	libnetcdf.so.15.1.0
4	0.00%	[.] H5SL_remove	libhdf5.so.103.1.0
3	0.00%	[.] H5AC_protect	libhdf5.so.103.1.0
3	0.00%	[.] NC3_inq_var	libnetcdf.so.15.1.0
3	0.00%	[.] H5F_addr_decode_len	libhdf5.so.103.1.0
3	0.00%	[.] H5FL_reg_malloc	libhdf5.so.103.1.0
2	0.00%	[.] nc_utf8proc_normalize_utf32	libnetcdf.so.15.1.0
2	0.00%	[.] H5C_protect	libhdf5.so.103.1.0
2	0.00%	[.] ncx_getn_float_float	libnetcdf.so.15.1.0
2	0.00%	[.] H5G__traverse_real	libhdf5.so.103.1.0
2	0.00%	[.] H5FL_malloc	libhdf5.so.103.1.0
2	0.00%	[.] H5AC_tag	libhdf5.so.103.1.0

Figura 3.1: Perfilado después de la combinación de los dos ciclos de lectura de la subrutina `read_emision`

.

comienza con `E_` o `e_`. En la tabla 3.1 se observa una reducción de 13 segundos en comparación con el tiempo obtenido antes de estos cambios para la subrutina `read_emision` lo que significa una aceleración de $1.57x$. En la figura 3.1 se puede observar que las muestras tomadas durante el perfilado redujeron del 10.9 % a 1.3 % después de estos cambios.

Subrutina	tiempo original	tiempo después de cambios
READS_EMISION::	34.2657	21.8276
READS_GRIDS::	0.0989	0.0969
CONVERSION::	97.5011	98.1160
FILE_OUT::	1.1808	1.1378
TOTAL::	133.0467	121.1785

Tabla 3.1: Tiempos de ejecución del programa de interpolación. Después combinar los dos ciclos de lectura en la subrutina read_emision se redujo el tiempo de 34 a 21 segundos.

3.2.2. Funcionamiento de la Subrutina `swapn4b`

El archivo de entrada utilizado por el programa de interpolación `wrfchemin.nc` se encuentra en formato `netCDF` clásico (`CDF-1`) esto se pudo corroborar al utilizar la herramienta de línea de comandos `ncdump` con la opción `-k` [5]. `CDF-1` se caracteriza por utilizar como formato interno `XDR` para almacenar una secuencia de bits (bytes de 8 bits) [6] [7] de forma independiente de la arquitectura; cabe hacer notar que en la documentación de este proceso se menciona un costo computacional en la conversión del formato `XDR` al formato correspondiente a la

[5] https://www.unidata.ucar.edu/software/netcdf/documentation/NUG/netcdf_utilities_guide.html

[6] https://www.unidata.ucar.edu/software/netcdf/documentation/NUG/netcdf_introduction.html#netcdf_format

[7] https://www.unidata.ucar.edu/support/help/MailArchives/netcdf/msg06389.html

arquitectura del equipo donde se utilizarán dichos datos[8].

Considerando que la arquitectura donde se desarrollo este trabajo, y en general dentro del ICyACC, tiene como base los procesadores INTEL en la que la información multi-byte (enteros y punto flotantes) es procesada en memoria bajo el esquema `littl-endian` [9] la subrutina `swapn4b` debe ser la encargada de cambiar el formato en el que se encuentran los datos multi-byte en el archivo `netcdf` a la correspondiente con la arquitectura que en esta caso es `litte-endian`. Para comprobar lo anterior se realizó una copia del archivo `wrfchemin.nc` con la herramienta `nccopy` solicitando la conversión del formato `CDF-1` al formato `CDF-4` [10] para que los datos sean almacenados utilizando el formato `HDF5` cuya característica es almacenar la información en el formato multy-byte `(little-endian, big-endian)` de la arquitectura donde se escriben la información y en el caso de la arquitectura con procesadores intel este almacenamiento es `little-endian`. Después del cambio de formato se midieron los tiempos de ejecución y se realizo el perfilado del programa, los resultado puede verse en la tabla 3.2 y figura 3.2

Samples	Overhead	Symbol	Shared Object
389991	96.42%	[.] conversion_	interpola.exe
6356	1.50%	[.] reads_emision_	interpola.exe
1397	0.32%	[.] swapn4b	libnetcdf.so.15.1.0
1062	0.25%	[.] file_out_	interpola.exe
116	0.03%	[.] H5S__hyper_get_seq_list	libhdf5.so.103.1.0
113	0.03%	[.] reads_grid_	interpola.exe
77	0.02%	[.] fill_NC_var	libnetcdf.so.15.1.0
66	0.02%	[.] H5VM_memcpyvv	libhdf5.so.103.1.0
34	0.01%	[.] H5SL_search	libhdf5.so.103.1.0
14	0.00%	[.] H5S__hyper_free_span_info	libhdf5.so.103.1.0
12	0.00%	[.] H5FL_reg_malloc	libhdf5.so.103.1.0
11	0.00%	[.] H5SL_insert_common	libhdf5.so.103.1.0
10	0.00%	[.] H5S__hyper_iter_init	libhdf5.so.103.1.0
8	0.00%	[.] H5C_protect	libhdf5.so.103.1.0
7	0.00%	[.] H5FL_reg_free	libhdf5.so.103.1.0
7	0.00%	[.] H5FL_blk_find_list	libhdf5.so.103.1.0
6	0.00%	[.] memcpy@plt	libnetcdf.so.15.1.0
6	0.00%	[.] H5FL_reg_calloc	libhdf5.so.103.1.0
6	0.00%	[.] H5SL_remove	libhdf5.so.103.1.0
5	0.00%	[.] H5D__btree_cmp3	libhdf5.so.103.1.0
5	0.00%	[.] H5T_copy	libhdf5.so.103.1.0

Figura 3.2: Después del cambio de formato a `netcdf-4` se observa que las muestras tomadas durante la llamada a `swapn4b` se reduce del 13.9 al 1.5 porciento.

En comparación con el tiempo obtenido en los primeros cambios en la subrutina `read_emision` 21.8276 segundos ahora el tiempo obtenido con el cambio de formato fue de 5.8211 lo que representa una aceleración de $3.74x$ y en comparación con el tiempo total de ejecución del programa una aceleración de $1.5x$.

[8] https://www.unidata.ucar.edu/software/netcdf/documentation/NUG/file_structure_and_performance.html#xdr_layer

[9] El término inglés `endianness` designa el formato en el que se almacenan los datos de más de un byte y en la actualidad hay 2 variantes: `big-endian` y `little-endian`. El fortmato `big-endian` adoptado por Motorola entre otros, consiste en representar los bytes en el orden "natural": así el valor hexadecimal 0x4A3B2C1D se codificaría en memoria en la secuencia `4A, 3B, 2C, 1D`. El otro formato `little-endian` adoptado por `Intel`, entre otros, para este formato el valor 0x4A3B2C1D se codificaría como `1D, 2C, 3B, 4A`.

[10] El comando utilizado fue: nccopy -k netCDF-4 wrfchemin.nc wrfchemin.nc4. https://www.unidata.ucar.edu/software/netcdf/documentation/NUG/netcdf_utilities_guide.html#guide_nccopy

Subrutina	tiempo antes de cambios	tiempo después de cambios
READS_EMISION::	21.8276	5.8211
READS_GRIDS::	0.0969	0.0919
CONVERSION::	98.1160	98.3040
FILE_OUT::	1.1378	1.2418
TOTAL::	121.1785	105.4589

Tabla 3.2: Tiempos de ejecución del programa de interpolación después de cambiar el formato del archivo de entrada `wrfchemin.nc` de `CDF-1` a `CDF-4`.

3.2.3. Funcionamiento de la subrutina `conversion`

El algoritmo de interpolación compara todas las celdas formadas por las latitudes y longitudes del dominio de emisiones globales con las celdas del dominio de interpolación para obtener la fracción del área común entre ellas y utilizar dicha fracción como factor para asignar las emisiones de contaminantes correspondiente. Para recorrer todas las longitudes y latitudes de ambos dominios se utilizan ciclos `DO` anidados, posteriormente si existe área común entre las celdas se lleva a cabo la asignación de emisiones por medio de 3 ciclos anidados `DO`. En el algoritmo 1 se puede apreciar la cantidad de ciclos involucrados para la asignación de las emisiones.

Algorithm 1 Interpolación al dominio de estudio

```
for all latitudes dominio interpolacion do
  for all longitudes dominio interpolacion do
    for all latitudes dominio emisiones do
      for all latitudes dominio emisiones do
        calcular-area-comun
        if area-comun >0 then
          for all niveles en vertical do
            for all horas do
              for all conjunto emisiones do
                if es variable emision then
                  interpolacion ← emision x area-comun
                end if
              end for
            end for
          end for
        end if
      end for
    end for
  end for
end for
```

Para llevar a cabo el análisis de rendimiento se consideró dividir el código de la asignación de emisiones en tres secciones principales 1. Ciclos para recorrer las celdas, 2. Asignación de las emisiones, 3. Cálculo del área común entre celdas.

En el listado de código listado 3.5 correspondiente a los ciclos para recorrer las celdas se puede observar el uso de 4 ciclos `DO` anidados, 2 ciclos para las celdas del dominio de interpolación, lineas 1 y 2, y 2 ciclos para recorrer las celdas del dominio de emisiones, lineas 7 y 8. Esta estructura de 2 ciclos para recorrer cada uno de los dominios puede ser considerado como un bloque donde se agreguen directivas `OpenMP` para recorrer las latitudes y las longitudes de forma paralela[6].

```
1  do j=1,djx
2   do i=1,dix
3     ylat1=dlat(i  ,j  )
4     ylat2=dlat(i,j+1)
5     xlon1=dlon(i   ,j)
6     xlon2=dlon(i+1,j)
7     do ii=1,eix
8       do jj=1,ejx
9         elat1= elat(ii,jj)
10        elat2= elat(ii,jj+1)
11        elon1= elon(ii,jj)
12        elon2= elon(ii+1,jj)
13 !
14 !CODIGO CORRESPONDIENTE AL CALCULO AREA COMUN
15 !CODIGO CORRESPONDIENTE A LA ASIGNACION DE EMISIONES
16 !
17      end do
18    end do
19   end do
20 end do
```

Listado 3.5: Ciclos para recorrer todas las celdas de ambos dominios

El primer cambio consistió en agregar la directiva `$OMP PARALLEL DO` para que los ciclos utilizados para obtener las longitudes y las latitudes de las celdas del dominio de emisiones, el cálculo de la fracción común y la asignación de las emisiones se realicen de forma paralela. En el listado listado 3.6 se muestran parte del código después de agregar la directiva `OpenMP` y en la tabla 3.3 se pueden observar los tiempos obtenidos.

```
1  !OMP PARALLEL DO PRIVATE (elat1, elat2, elon1, elon2)
2  do ii=1,eix
3      do jj=1,ejx
4        elat1= elat(ii,jj)
5        elat2= elat(ii,jj+1) !staged lat
6        elon1= elon(ii,jj)
7        elon2= elon(ii+1,jj)!staged long
8  !
9  !CODIGO CORRESPONDIENTE AL CALCULO AREA COMUN
10 !CODIGO CORRESPONDIENTE A LA ASIGNACION DE EMISIONES
11 !
12     end do
```

```
13 end do
14 !$OMP END PARALLEL DO
```

Listado 3.6: Uso de directiva OpenMP para recorrer de forma paralela las celdas del dominio de emisiones

número de hilos	tiempo segundos	Aceleración
2	103.15	1.02
4	99.65	1.06
8	84.63	1.25
16	42.70	2.46

Tabla 3.3: Tiempos de ejecución y aceleración observada (con base en el tiempo 105.45) después de agregar la directiva `OpenMP: OMP PARALLEL DO` para recorrer de forma paralela las celdas del dominio de emisiones

número de hilos	tiempo segundos	Aceleración
2	52.89	1.99
4	30.22	3.84
8	18.71	5.63
16	14.83	7.11

Tabla 3.4: Tiempos de ejecución y aceleración observada (con base en el tiempo 105.45) después de agregar la directiva `OpenMP: OMP PARALLEL DO` para recorrer de forma paralela las celdas del dominio de interpolación

```
1  !OMP PARALLEL DO PRIVATE (ylat1,ylat2,xlon1,xlon2,
2  elat1,elat2,elon1,elon2)
3  do j=1,djx
4   do i=1,dix
5     ylat1=dlat(i  ,j  )
6     ylat2=dlat(i,j+1)
7     xlon1=dlon(i  ,j)
8     xlon2=dlon(i+1,j)
9     do ii=1,eix
10     do jj=1,ejx
11       elat1= elat(ii,jj)
12       elat2= elat(ii,jj+1)
13       elon1= elon(ii,jj)
14       elon2= elon(ii+1,jj)
15 !CODIGO CORRESPONDIENTE AL CALCULO AREA COMUN
16 !CODIGO CORRESPONDIENTE A LA ASIGNACION DE EMISIONES
17     end do
18    end do
```

```
19   end do
20  end do
21  !$OMP END PARALLEL DO
```

Listado 3.7: Ciclos para recorrer todas las celdas de ambos dominios

Para continuar con el análisis se cambio la ubicación de la directiva `$OMP PARALLEL DO` en los ciclos `DO` correspondientes a recorrer las celdas del dominio de interpolación y como resultado el recorrido de las celdas de ambos dominios, la obtención del área común y la asignación de los contaminantes de realzarán en paralelo. En el listado listado 3.7 se muestran parte del código después de agregar la directiva `OpenMP` y en la tabla 3.4 se reportan los tiempos obtenidos.

En la imagen 3.3 se puede observar el reporte de perfilado del programa de interpolación después de agregar la directiva `OpenMP` como se muestra en el listado 3.7

Samples	Overhead	Symbol	Shared Object
391907	96.42%	[.] conversion_._omp_fn.0	interpola.exe
6111	1.48%	[.] reads_emision_	interpola.exe
1380	0.33%	[.] swapn4b	libnetcdf.so.15.1.0
1026	0.24%	[.] file_out_	interpola.exe
101	0.02%	[.] reads_grid_	interpola.exe
101	0.02%	[.] H5S__hyper_get_seq_list	libhdf5.so.103.1.0
72	0.02%	[.] fill_NC_var	libnetcdf.so.15.1.0
68	0.02%	[.] H5VM_memcpyvv	libhdf5.so.103.1.0
30	0.01%	[.] H5SL_search	libhdf5.so.103.1.0
18	0.00%	[.] H5SL_insert_common	libhdf5.so.103.1.0
15	0.00%	[.] H5FL_reg_malloc	libhdf5.so.103.1.0
8	0.00%	[.] H5FL_reg_free	libhdf5.so.103.1.0
8	0.00%	[.] H5S__hyper_free_span_info	libhdf5.so.103.1.0
7	0.00%	[.] memcpy@plt	libnetcdf.so.15.1.0
7	0.00%	[.] H5SL_remove	libhdf5.so.103.1.0
6	0.00%	[.] H5S_extent_release	libhdf5.so.103.1.0
6	0.00%	[.] conversion_	interpola.exe
6	0.00%	[.] H5C_protect	libhdf5.so.103.1.0
5	0.00%	[.] H5S__hyper_iter_init	libhdf5.so.103.1.0
5	0.00%	[.] H5S__hyper_release	libhdf5.so.103.1.0
4	0.00%	[.] H5D__select_io	libhdf5.so.103.1.0
4	0.00%	[.] H5S_select_copy	libhdf5.so.103.1.0

Figura 3.3: Perfilado del programa de interpolación después de agregar la directiva `OMP PARALLEL DO`

La siguiente parte del código que se analizó fue el correspondiente a la asignación de emisiones que se muestra en el listado 3.8 en donde se utilizan 3 ciclos para asignar los valores para cada conjunto de emisiones `kl`, cuando dicha asignación puede realizarse con notación de arreglos `A(:) = B(:)` soportada por el lenguaje Fortan90.

El cambio consistió en reescribir la asignación realizada en la línea 6 del listado 3.8, se sustituyo por `ed(i,j,:,:,kl) = ed(i,j,:,:,kl)+ei(ii,jj,:,:,kl)*area` eliminando con esto el uso los ciclos para recorrer la variable `l` y `ih`. El tiempo de ejecución obtenido para este cambio fue 104.83 segundos, en comparación con el tiempo obtenido antes del cambio 105.45 segundos, no representó una mejora significativa.

La última parte del código que se modificó fue la correspondiente al cálculo de la fracción del área común entre celdas listado 3.9, esta operación se realiza siempre para todas las celdas independientemente que tengan o no un área en común (línea 8). Para no realizar estas operaciones adicionales, se reescribió el código para que solo se haga el cálculo del área cuando ésta exista, cuando las dos condiciones `if` sean verdaderas lineas 4 y 6 listado 3.9. Si alguno de los 2 `if` es falso con la instrucción `cycle` de `Fortran` se pasa a la siguiente iteración del ciclo `DO` llevadose a cabo la comparación de otro par de celdas. El código modificado se presenta en el listado 3.10 y en la tabla 3.5 los tiempos obtenidos.

```
1  if( area.gt.0.) then
2   do l=1,size(ed,dim=3) ! altura
3    do  ih=1,size(ed,dim=4) !hora
4     do  kl=1,size(ed,dim=5) ! emision
5     if (tvar(kl))
6      ed(i,j,l,ih,kl)=ed(i,j,l,ih,kl)+ei(ii,jj,l,ih,kl)*area
7     end do ! kl
8    end do ! ih
9   end do  ! l
10 end if
```

Listado 3.8: Ciclos involucrados en la asignación de las emisiones.

```
1 alat=0.0
2 alon=0.0
3 tot=(elat2-elat1)*(elon2-elon1)/((ylat2-ylat1)*(xlon2-xlon1))
4 if(ylat1.le.elat2.and. ylat2.ge.elat1)
5      alat=(min(ylat2,elat2)-max(ylat1,elat1))/(elat2-elat1)
6 if(xlon1.le.elon2 .and. xlon2.ge.elon1)
7      alon=(min(xlon2,elon2)-max(xlon1,elon1))/(elon2-elon1)
8 area=max(0.,alat*alon)* tot
```

Listado 3.9: Fragmento del código correspondiente a determinar la fracción del área común entre 2 celdas.

```
1  alat=0.0
2  alon=0.0
3  if(ylat1.le.elat2.and. ylat2.ge.elat1) then
4   if(xlon1.le.elon2 .and. xlon2.ge.elon1) then
5     alat=(min(ylat2,elat2)-max(ylat1,elat1))/(elat2-elat1)
6     alon=(min(xlon2,elon2)-max(xlon1,elon1))/(elon2-elon1)
7     tot=(elat2-elat1)*(elon2-elon1)/((ylat2-ylat1)*(xlon2-xlon1))
8     area=max(0.,alat*alon)* tot
9   else
10    cycle
11  end if
12 else
13     cycle
14 end if
```

Listado 3.10: Cambios realizados para que el calculo de la fracción del área común se realice solo en aquellas celdas que si tengan dicha área.

número de hilos	tiempo segundos	Aceleración
2	30.41	3.47
4	17.26	6.11
8	19.21	11.45
16	7.89	17.90

Tabla 3.5: Tiempos de ejecución y aceleración observada (con base en el tiempo 105.45) después de reescribir la condición para el cálculo de la fracción de área común entre celdas

3.3. Evaluación de los cambios implementados

La evaluación de la refactorización consistió en la comparación del tiempo de ejecución del programa original con el tiempo de ejecución después de realizar cada uno de los cambios, los correspondientes a la técnica de refactorización y la inserción de directivas `OpenMP`. Posteriormente con el uso de herramientas de consulta y visualización de archivos `netCDF` verificar el contenido del archivo resultante de la ejecución del programa modificado.

El programa original tardaba en promedio 133 segundos y el primer cambio fue la reescritura de del código fuente de la subrutina `read_emision` para eliminar uno de los 2 ciclos utilizados para la lectura del archivo de entrada el resultado de este cambio fue la reducción de 12 segundos. Como uno de los resultados más importantes obtenidos durante la etapa de la refactorización fue identificar con la ayuda del perfilado que el formato para uno de los archivos de entrada no era el adecuado para la arquitectura `Intel (little-endian)` donde se ejecuta el programa de interpolación. Se cambio el formato orignal `CDF-1` a el formato `CDF-4` logrando con esto una reducción del 16 segundos durante la lectura del archivo. Esta información se resume en la tabla 3.6

Cambio	tiempo segundos	Aceleración `speedup`
Programa original	133.05	1
Ciclos de lectura	121.18	1.1
Formato CDF-4	105.45	1.26

Tabla 3.6: Tiempos de ejecución final del programa de interpolación después de la refactorización

Llevar a cabo la comparación de celdas para obtener el área común en caso de haberla, y la asignación del contaminante a partir de la fracción de área se detecto mediante la técnica de refactorización y perfilado como las operaciones que más consumían tiempo durante la ejecución del programa por lo que insertar directivas `OpenMP` para llevar a cabo estas operaciones de forma paralela redujo el tiempo de la subrutina `conversion` con el uso de 2 hilos de 98 a 45 segundos para un total de tiempo de ejecución de 52.89 y una aceleración de $1.99x$, una disminución de 98 a 8 segundos con el uso de 16 hilos para un tiempo total de ejecución de 14.83 segundos y una aceleración de $7.11X$.

La última etapa de la refactorización fue la modificación del código correspondiente a determinar el área común entre celdas debido a que las operaciones siempre de realizaban

aún cuando las celdas no tenían área común. Hacer este pequeño cambio en el código redujo aún más el tiempo obtenido con la ejecución en varios hilos por ejemplo en el caso de usar 16 hilos el tiempo total de ejecución paso de 14.83 a 7.89 lo que significó una aceleración del $17.90x$ en comparación con el tiempo del programa original que era de 133.05. El resumen de los tiempos obtenidos en cada etapa se muestran en la tabla 3.7

Cambio	tiempo segundos	Aceleración `speedup`
Programa original	133.05	1
Ciclos de lectura	121.18	1.1
Formato CDF-4	105.46	1.26
Interpolación 2 hilos	30.41	3.47
Interpolación 4 hilos	17.26	6.11
Interpolación 8 hilos	9.21	11.45
Interpolación 16 hilos	7.89	17.90

Tabla 3.7: Tiempos de ejecución final del programa de interpolación utilizando 16 cores

Para verificar que la aceleración de $17.90x$ obtenida en la versión final del programa de interpolación generará el archivo con el dominio de interpolación `wrfchemi_00z_d01` conteniendo la información de los diferentes conjuntos de contaminantes y compararlo con un archivo obtenido con el programa original.

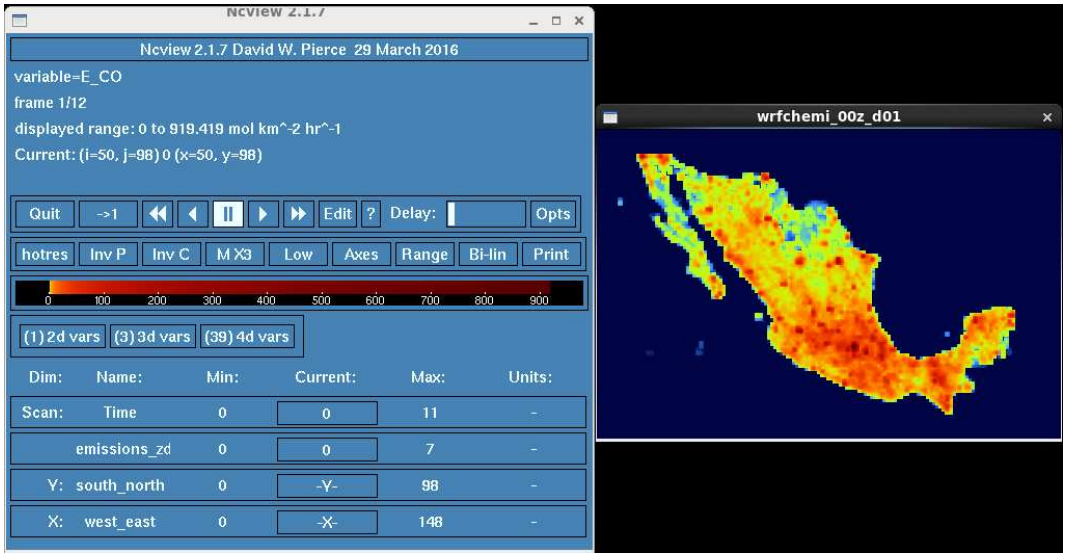

Figura 3.4: Visualización de la variable `E_CO` en el dominio de interpolación, en este caso el dominio de interpolación es más grande que el dominio de emisiones abarcando parte de EUA, Belice y Guatemala.

Esta comparación fue realizada visualmente con el programa `ncview` (figura 3.4) y numéricamente con la herramientas de línea de comandos proporcionadas por el software `nco`, en específico `ncbo` y sus alias `ncadd, ncdiff, ncmult, ncdivide`[11].

[11]La herramienta `ncbo netCDF Binary Operator` realiza operaciones binarias sobre variables en un ar-

```
data:

 E_CO =
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0,
```

Figura 3.5: Parte de la salida del comando `ncdiff` done se observa que la diferencia entre ambos archivos es cero lo que indica que son numéricamente iguales. El comando utilizado para la consulta anterior fue `ncdump -v E_CO wrfchemi-3.nc`

Para comparar ambos archivos utilizamos `ncdiff` para realizar la resta del archivo resultado de la interpolación con el programa original y el archivo resultado del programa de interpolación con las modificaciones, la línea de comando utilizada fue:

```
ncdiff -v E_CO wrfchemi-1.nc  wrfchemi-2.nc wrfchemi-3.nc
```

donde `wrfchemi-1.nc` corresponde al archivo resultado de ejecutar el programa original, `wrfchemi-2.nc` es el archivo obtenido de ejecutar el programa modificado y `wrfchemi-3.nc` es el archivo resultado de restar los valores de la variable indicada `E_CO`. La operación resta se lleva a cabo elemento a elemento y si ambos archivos son iguales entonces el resultado de la resta en el archivo `wrfchemi-3.nc` debe contener valores cero [12]. Lo anterior se verificó con la herramienta `ncdump` y el resultado se puede observar en la en la imagen 3.5

3.4. Resumen

En este capítulo se analizaron las subrutinas `read_emision e inversion` identificadas como las que consumen más tiempo durante la ejecución del programa de interpolación. Se obtuvo una reducción de 21 segundos después de refactorizar el código de la subrutina `read_emision` eliminando uno de los ciclos realizados durante la lectura del archivo con el dominio de emisiones, se identificó que al hacer el cambio de formato en el que se encuentra el archivo de emisiones globales `cerCDF-1 (classic)` al formato `netCDF-4` se reduce el tiempo de ejecución en 16 segundos obteniendo una aceleración de $1.26x$

La inclusión de directivas `OpenMP` para ejecutar en paralelo la búsqueda de áreas comunes entre cada una de las celdas del dominio global con cada una de las celdas del dominio de

chivo y las correspondientes variables con el mismo nombre en otro guardando el resultado en un tercer archivo. Las operaciones soportadas son suma `ncadd 1.nc 2.nc 3.nc`, resta `ncdiff 1.nc 2.nc 3.nc`, multiplicación `ncmult 1.nc 2.nc 3.nc` y división `ncdivide 1.nc 2.nc 3.nc`. Para mayor información http://nco.sourceforge.net/nco.html#ncbo-netCDF-Binary-Operator

[12] Podemos resumir el comportamiento de la herramienta ncdiff como la resta de 2 matrices y el resultado se almacena en una tercera $C = A - B$

Figura 3.6: tiempo aceleración

interpolación y la asignación de las emisiones correspondientes arrojo como resultado aceleraciones de $2x$ y hasta $7x$ para el uso de 2 y 16 hilos respectivamente. Por último se realizo un pequeño cambio para no calcular el área común en celdas que geográficamente no comparten latitud o longitud logrando con esto una aceleración más que llego al $17.90x$ con el uso de 16 hilos. Se puede observar un resumen de la reducción del tiempo y la aceleración obtenida en cada etapa en la imagen 3.6.

Capítulo 4

Conclusiones y Trabajo Futuro

4.1. Conclusiones

La técnica de refactorización permitió realizar pequeños cambios al código del programa de interpolación y obtener una reducción en el tiempo de lectura de los archivos de entrada, adicionalmente obtener el perfilado permitió identificar que la subrutina de interpolación consume el 96 % y agregar directivas `OpenMP` para realizar las operaciones de forma paralela tuvo como resultado una aceleración de $17.90x$ en la ejecución utilizando 16 hilos.

Uno de los principales hallazgos durante el desarrollo de este trabajo fue identificar por medio del análisis de tiempos ejecución provistos por la herramienta de perfilado que la subrutina `read_emision` encargada de lectura consumía un 23 % del tiempo y la subrutina externa al programa de interpolación `swapn4b` consumía más del 50 % de este tiempo. Después de implementar cambios al código fuente del programa combinando 2 ciclos de lectura en la subrutina `read_emision` se obtuvo una aceleración de $1.1x$ pero aun así la subrutina `swapn4b` consumía el 90 % de la ejecución. Se encontró que dicha rutina realizaba una conversión de formato `big-endian` a `little-endian` durante el proceso de lectura debido a que uno de los archivo de entrada se encuentra en formato clásico `netCDF-1` cuyo formato de almacenamiento es `big-endian`, al llevar a cabo la conversión del archivo de entrada con la herramienta `nccopy` a formato `netCDF-4` cuya característica principal es utilizar el formato de almacenamiento con base en la arquitectura donde se genere el archivo que en este caso es `little-endian`), se obtuvo una aceleración de $1.26x$.

Aunque el programa de interpolación se encuentra escrito en lenguaje de programación `Fortran 90` hace uso de un formato vigente para la lectura y escritura como lo es `netCDF` cuya versión más reciente al momento de escribir el presente trabajo es 4.8.1 liberada en agosto de 2021, no obstante aunque se trata de un lenguaje de programación eficiente y diseñado para cómputo científico es considerado un lenguaje poco atractivo para nuevas generaciones de alumnos, teniendo como consecuencia que el el mantenimiento de programas escritos en este lenguaje sea en ocasiones difícil o no pueda realizarse. Por otro lado reescribir estos programas con lenguajes modernos, rápidos y eficientes puede tener como consecuencia la generación de mayor interés en las nuevas generaciones y en consecuencia se tengan los alumnos para realizar su mantenimiento e inclusive agregar nuevas funcionalidades.

4.1.1. Recomendaciones

Llevar a cabo el cambio del formato para el archivo de emisiones de clásico `CDF-1` al formato `CDF-4` para evitar el uso de la subrutina `swap4n` para la conversión de `big-endian` a

`little-endian` para los valores enteros y de punto flotantes.

Analizar la posibilidad del cambio del lenguaje `fortran` por otro con el mismo rendimiento pero más actual, permitiendo con esto que nuevas generaciones de programadores puedan llevar a cabo el mantenimiento y modificaciones en el futuro.

4.1.2. Trabajo futuro

El proceso de interpolación es un problema que puede resolverse por medio del enfoque de la paralelización por datos y es susceptible a ser implementada en memoria distribuida mediante el uso de MPI.

Apéndices

Apéndice A

Estructura del archivo wrfchemin.nc

Resultado de la ejecución del comando `ncdump -h wrfchemin.nc`

```
netcdf wrfchemin {
dimensions:
Time = UNLIMITED ; // (12 currently)
DateStrLen = 19 ;
west_east = 1059 ;
south_north = 677 ;
bottom_top = 1 ;
emissions_zdim_stag = 8 ;
variables:
char Times(Time, DateStrLen) ;
float XLONG(Time, south_north, west_east) ;
XLONG:FieldType = 104 ;
XLONG:MemoryOrder = "XYZ" ;
XLONG:description = "LONGITUDE, WEST IS NEGATIVE" ;
XLONG:units = "degree_east" ;
XLONG:axis = "X" ;
float XLAT(Time, south_north, west_east) ;
XLAT:FieldType = 104 ;
XLAT:MemoryOrder = "XYZ" ;
XLAT:description = "LATITUDE, SOUTH IS NEGATIVE" ;
XLAT:units = "degree_north" ;
XLAT:axis = "Y" ;
float POB(Time, south_north, west_east) ;
POB:FieldType = 104 ;
POB:MemoryOrder = "XYZ" ;
POB:description = "Population in each grid" ;
POB:units = "number" ;
float UTMx(south_north, west_east) ;
UTMx:FieldType = 104 ;
UTMx:MemoryOrder = "XYZ" ;
UTMx:description = "UTM coordinate west-east" ;
UTMx:units = "km" ;
UTMx:axis = "X" ;
```

```
float UTMy(south_north, west_east) ;
UTMy:FieldType = 104 ;
UTMy:MemoryOrder = "XYZ" ;
UTMy:description = "UTM coordinate sotuth-north" ;
UTMy:units = "km" ;
UTMy:axis = "Y" ;
int UTMz(south_north, west_east) ;
UTMz:FieldType = 104 ;
UTMz:MemoryOrder = "XYZ" ;
UTMz:description = "UTM Zone" ;
UTMz:units = "None" ;
float E_CO(Time, emissions_zdim_stag, south_north, west_east) ;
E_CO:FieldType = 104 ;
E_CO:MemoryOrder = "XYZ" ;
E_CO:description = "Emissions rate of Carbon Monoxide" ;
E_CO:units = "mol km^-2 hr^-1" ;
E_CO:stagger = "Z" ;
E_CO:coordinates = "XLONG XLAT" ;
float E_NH3(Time, emissions_zdim_stag, south_north, west_east) ;
E_NH3:FieldType = 104 ;
E_NH3:MemoryOrder = "XYZ" ;
E_NH3:description = "Emissions rate of NH3" ;
E_NH3:units = "mol km^-2 hr^-1" ;
E_NH3:stagger = "Z" ;
E_NH3:coordinates = "XLONG XLAT" ;
float E_NO(Time, emissions_zdim_stag, south_north, west_east) ;
E_NO:FieldType = 104 ;
E_NO:MemoryOrder = "XYZ" ;
E_NO:description = "Emissions rate of NO" ;
E_NO:units = "mol km^-2 hr^-1" ;
E_NO:stagger = "Z" ;
E_NO:coordinates = "XLONG XLAT" ;
float E_NO2(Time, emissions_zdim_stag, south_north, west_east) ;
E_NO2:FieldType = 104 ;
E_NO2:MemoryOrder = "XYZ" ;
E_NO2:description = "Emissions rate of NO2" ;
E_NO2:units = "mol km^-2 hr^-1" ;
E_NO2:stagger = "Z" ;
E_NO2:coordinates = "XLONG XLAT" ;
float E_SO2(Time, emissions_zdim_stag, south_north, west_east) ;
E_SO2:FieldType = 104 ;
E_SO2:MemoryOrder = "XYZ" ;
E_SO2:description = "Emissions rate of SO2" ;
E_SO2:units = "mol km^-2 hr^-1" ;
E_SO2:stagger = "Z" ;
E_SO2:coordinates = "XLONG XLAT" ;
float E_ALD(Time, emissions_zdim_stag, south_north, west_east) ;
E_ALD:FieldType = 104 ;
```

```
E_ALD:MemoryOrder = "XYZ" ;
E_ALD:description = "Emissions rate of ALDEHYDES" ;
E_ALD:units = "mol km^-2 hr^-1" ;
E_ALD:stagger = "Z" ;
E_ALD:coordinates = "XLONG XLAT" ;
float E_CH4(Time, emissions_zdim_stag, south_north, west_east) ;
E_CH4:FieldType = 104 ;
E_CH4:MemoryOrder = "XYZ" ;
E_CH4:description = "Emissions rate of METHANE" ;
E_CH4:units = "mol km^-2 hr^-1" ;
E_CH4:stagger = "Z" ;
E_CH4:coordinates = "XLONG XLAT" ;
float E_CSL(Time, emissions_zdim_stag, south_north, west_east) ;
E_CSL:FieldType = 104 ;
E_CSL:MemoryOrder = "XYZ" ;
E_CSL:description = "Emissions rate of CRESOL" ;
E_CSL:units = "mol km^-2 hr^-1" ;
E_CSL:stagger = "Z" ;
E_CSL:coordinates = "XLONG XLAT" ;
float E_ETH(Time, emissions_zdim_stag, south_north, west_east) ;
E_ETH:FieldType = 104 ;
E_ETH:MemoryOrder = "XYZ" ;
E_ETH:description = "Emissions rate of Ethane" ;
E_ETH:units = "mol km^-2 hr^-1" ;
E_ETH:stagger = "Z" ;
E_ETH:coordinates = "XLONG XLAT" ;
float E_GLY(Time, emissions_zdim_stag, south_north, west_east) ;
E_GLY:FieldType = 104 ;
E_GLY:MemoryOrder = "XYZ" ;
E_GLY:description = "Emissions rate of Glyoxal" ;
E_GLY:units = "mol km^-2 hr^-1" ;
E_GLY:stagger = "Z" ;
E_GLY:coordinates = "XLONG XLAT" ;
float E_HC3(Time, emissions_zdim_stag, south_north, west_east) ;
E_HC3:FieldType = 104 ;
E_HC3:MemoryOrder = "XYZ" ;
E_HC3:description = "Emissions rate of HC3" ;
E_HC3:units = "mol km^-2 hr^-1" ;
E_HC3:stagger = "Z" ;
E_HC3:coordinates = "XLONG XLAT" ;
float E_HC5(Time, emissions_zdim_stag, south_north, west_east) ;
E_HC5:FieldType = 104 ;
E_HC5:MemoryOrder = "XYZ" ;
E_HC5:description = "Emissions rate of HC5" ;
E_HC5:units = "mol km^-2 hr^-1" ;
E_HC5:stagger = "Z" ;
E_HC5:coordinates = "XLONG XLAT" ;
float E_HC8(Time, emissions_zdim_stag, south_north, west_east) ;
```

```
E_HC8:FieldType = 104 ;
E_HC8:MemoryOrder = "XYZ" ;
E_HC8:description = "Emissions rate of HC8" ;
E_HC8:units = "mol km^-2 hr^-1" ;
E_HC8:stagger = "Z" ;
E_HC8:coordinates = "XLONG XLAT" ;
float E_HCHO(Time, emissions_zdim_stag, south_north, west_east) ;
E_HCHO:FieldType = 104 ;
E_HCHO:MemoryOrder = "XYZ" ;
E_HCHO:description = "Emissions rate of HCHO" ;
E_HCHO:units = "mol km^-2 hr^-1" ;
E_HCHO:stagger = "Z" ;
E_HCHO:coordinates = "XLONG XLAT" ;
float E_ISO(Time, emissions_zdim_stag, south_north, west_east) ;
E_ISO:FieldType = 104 ;
E_ISO:MemoryOrder = "XYZ" ;
E_ISO:description = "Emissions rate of ISOPRENE" ;
E_ISO:units = "mol km^-2 hr^-1" ;
E_ISO:stagger = "Z" ;
E_ISO:coordinates = "XLONG XLAT" ;
float E_KET(Time, emissions_zdim_stag, south_north, west_east) ;
E_KET:FieldType = 104 ;
E_KET:MemoryOrder = "XYZ" ;
E_KET:description = "Emissions rate of Acetone" ;
E_KET:units = "mol km^-2 hr^-1" ;
E_KET:stagger = "Z" ;
E_KET:coordinates = "XLONG XLAT" ;
float E_MACR(Time, emissions_zdim_stag, south_north, west_east) ;
E_MACR:FieldType = 104 ;
E_MACR:MemoryOrder = "XYZ" ;
E_MACR:description = "Emissions rate of Acrolein" ;
E_MACR:units = "mol km^-2 hr^-1" ;
E_MACR:stagger = "Z" ;
E_MACR:coordinates = "XLONG XLAT" ;
float E_MGLY(Time, emissions_zdim_stag, south_north, west_east) ;
E_MGLY:FieldType = 104 ;
E_MGLY:MemoryOrder = "XYZ" ;
E_MGLY:description = "Emissions rate of MGLY" ;
E_MGLY:units = "mol km^-2 hr^-1" ;
E_MGLY:stagger = "Z" ;
E_MGLY:coordinates = "XLONG XLAT" ;
float E_MVK(Time, emissions_zdim_stag, south_north, west_east) ;
E_MVK:FieldType = 104 ;
E_MVK:MemoryOrder = "XYZ" ;
E_MVK:description = "Emissions rate of Methyl Vinil Ket" ;
E_MVK:units = "mol km^-2 hr^-1" ;
E_MVK:stagger = "Z" ;
E_MVK:coordinates = "XLONG XLAT" ;
```

```
float E_OL2(Time, emissions_zdim_stag, south_north, west_east) ;
E_OL2:FieldType = 104 ;
E_OL2:MemoryOrder = "XYZ" ;
E_OL2:description = "Emissions rate of Alkenes" ;
E_OL2:units = "mol km^-2 hr^-1" ;
E_OL2:stagger = "Z" ;
E_OL2:coordinates = "XLONG XLAT" ;
float E_OLI(Time, emissions_zdim_stag, south_north, west_east) ;
E_OLI:FieldType = 104 ;
E_OLI:MemoryOrder = "XYZ" ;
E_OLI:description = "Emissions rate of alkenes" ;
E_OLI:units = "mol km^-2 hr^-1" ;
E_OLI:stagger = "Z" ;
E_OLI:coordinates = "XLONG XLAT" ;
float E_OLT(Time, emissions_zdim_stag, south_north, west_east) ;
E_OLT:FieldType = 104 ;
E_OLT:MemoryOrder = "XYZ" ;
E_OLT:description = "Emissions rate of Terminal Alkynes" ;
E_OLT:units = "mol km^-2 hr^-1" ;
E_OLT:stagger = "Z" ;
E_OLT:coordinates = "XLONG XLAT" ;
float E_ORA1(Time, emissions_zdim_stag, south_north, west_east) ;
E_ORA1:FieldType = 104 ;
E_ORA1:MemoryOrder = "XYZ" ;
E_ORA1:description = "Emissions rate of Formic Acid" ;
E_ORA1:units = "mol km^-2 hr^-1" ;
E_ORA1:stagger = "Z" ;
E_ORA1:coordinates = "XLONG XLAT" ;
float E_ORA2(Time, emissions_zdim_stag, south_north, west_east) ;
E_ORA2:FieldType = 104 ;
E_ORA2:MemoryOrder = "XYZ" ;
E_ORA2:description = "Emissions rate of Acetic Acid" ;
E_ORA2:units = "mol km^-2 hr^-1" ;
E_ORA2:stagger = "Z" ;
E_ORA2:coordinates = "XLONG XLAT" ;
float E_TOL(Time, emissions_zdim_stag, south_north, west_east) ;
E_TOL:FieldType = 104 ;
E_TOL:MemoryOrder = "XYZ" ;
E_TOL:description = "Emissions rate of TOLUENE" ;
E_TOL:units = "mol km^-2 hr^-1" ;
E_TOL:stagger = "Z" ;
E_TOL:coordinates = "XLONG XLAT" ;
float E_XYL(Time, emissions_zdim_stag, south_north, west_east) ;
E_XYL:FieldType = 104 ;
E_XYL:MemoryOrder = "XYZ" ;
E_XYL:description = "Emissions rate of XYLENE" ;
E_XYL:units = "mol km^-2 hr^-1" ;
E_XYL:stagger = "Z" ;
```

```
E_XYL:coordinates = "XLONG XLAT" ;
float E_CO2(Time, emissions_zdim_stag, south_north, west_east) ;
E_CO2:FieldType = 104 ;
E_CO2:MemoryOrder = "XYZ" ;
E_CO2:description = "Emissions rate of Carbon Dioxide" ;
E_CO2:units = "mol km^-2 hr^-1" ;
E_CO2:stagger = "Z" ;
E_CO2:coordinates = "XLONG XLAT" ;
float E_PM_10(Time, emissions_zdim_stag, south_north, west_east) ;
E_PM_10:FieldType = 104 ;
E_PM_10:MemoryOrder = "XYZ" ;
E_PM_10:description = "EMISSIONS RATE OF PM_10" ;
E_PM_10:units = "ug m-2 s-1" ;
E_PM_10:stagger = "Z" ;
E_PM_10:coordinates = "XLONG XLAT" ;
float E_PM25(Time, emissions_zdim_stag, south_north, west_east) ;
E_PM25:FieldType = 104 ;
E_PM25:MemoryOrder = "XYZ" ;
E_PM25:description = "EMISSIONS RATE OF PM_25" ;
E_PM25:units = "ug m-2 s-1" ;
E_PM25:stagger = "Z" ;
E_PM25:coordinates = "XLONG XLAT" ;
float E_SO4I(Time, emissions_zdim_stag, south_north, west_east) ;
E_SO4I:FieldType = 104 ;
E_SO4I:MemoryOrder = "XYZ" ;
E_SO4I:description = "EMISSIONS RATE OF Sulfates" ;
E_SO4I:units = "ug m-2 s-1" ;
E_SO4I:stagger = "Z" ;
E_SO4I:coordinates = "XLONG XLAT" ;
float E_NO3I(Time, emissions_zdim_stag, south_north, west_east) ;
E_NO3I:FieldType = 104 ;
E_NO3I:MemoryOrder = "XYZ" ;
E_NO3I:description = "EMISSIONS RATE OF Nitrates" ;
E_NO3I:units = "ug m-2 s-1" ;
E_NO3I:stagger = "Z" ;
E_NO3I:coordinates = "XLONG XLAT" ;
float E_PM25I(Time, emissions_zdim_stag, south_north, west_east) ;
E_PM25I:FieldType = 104 ;
E_PM25I:MemoryOrder = "XYZ" ;
E_PM25I:description = "EMISSIONS RATE OF PM25I" ;
E_PM25I:units = "ug m-2 s-1" ;
E_PM25I:stagger = "Z" ;
E_PM25I:coordinates = "XLONG XLAT" ;
float E_ORGI(Time, emissions_zdim_stag, south_north, west_east) ;
E_ORGI:FieldType = 104 ;
E_ORGI:MemoryOrder = "XYZ" ;
E_ORGI:description = "EMISSIONS RATE OF Organic" ;
E_ORGI:units = "ug m-2 s-1" ;
```

```
E_ORGI:stagger = "Z" ;
E_ORGI:coordinates = "XLONG XLAT" ;
float E_ECI(Time, emissions_zdim_stag, south_north, west_east) ;
E_ECI:FieldType = 104 ;
E_ECI:MemoryOrder = "XYZ" ;
E_ECI:description = "EMISSIONS RATE OF Elemental Carbon" ;
E_ECI:units = "ug m-2 s-1" ;
E_ECI:stagger = "Z" ;
E_ECI:coordinates = "XLONG XLAT" ;
float E_SO4J(Time, emissions_zdim_stag, south_north, west_east) ;
E_SO4J:FieldType = 104 ;
E_SO4J:MemoryOrder = "XYZ" ;
E_SO4J:description = "EMISSIONS RATE OF SulfatesJ" ;
E_SO4J:units = "ug m-2 s-1" ;
E_SO4J:stagger = "Z" ;
E_SO4J:coordinates = "XLONG XLAT" ;
float E_NO3J(Time, emissions_zdim_stag, south_north, west_east) ;
E_NO3J:FieldType = 104 ;
E_NO3J:MemoryOrder = "XYZ" ;
E_NO3J:description = "EMISSIONS RATE OF NitratesJ" ;
E_NO3J:units = "ug m-2 s-1" ;
E_NO3J:stagger = "Z" ;
E_NO3J:coordinates = "XLONG XLAT" ;
float E_PM25J(Time, emissions_zdim_stag, south_north, west_east) ;
E_PM25J:FieldType = 104 ;
E_PM25J:MemoryOrder = "XYZ" ;
E_PM25J:description = "EMISSIONS RATE OF PM25J" ;
E_PM25J:units = "ug m-2 s-1" ;
E_PM25J:stagger = "Z" ;
E_PM25J:coordinates = "XLONG XLAT" ;
float E_ORGJ(Time, emissions_zdim_stag, south_north, west_east) ;
E_ORGJ:FieldType = 104 ;
E_ORGJ:MemoryOrder = "XYZ" ;
E_ORGJ:description = "EMISSIONS RATE OF Organic" ;
E_ORGJ:units = "ug m-2 s-1" ;
E_ORGJ:stagger = "Z" ;
E_ORGJ:coordinates = "XLONG XLAT" ;
float E_ECJ(Time, emissions_zdim_stag, south_north, west_east) ;
E_ECJ:FieldType = 104 ;
E_ECJ:MemoryOrder = "XYZ" ;
E_ECJ:description = "EMISSIONS RATE OF Elemental Carbon" ;
E_ECJ:units = "ug m-2 s-1" ;
E_ECJ:stagger = "Z" ;
E_ECJ:coordinates = "XLONG XLAT" ;

// global attributes:
:TITLE = "EI 2014 emissions for Mexico Area" ;
:START_DATE = "2016-02-09_00:00:00" ;
```

```
:DAY = "TUE" ;
:SIMULATION_START_DATE = "2016-02-09_00:00:00" ;
:WEST-EAST_GRID_DIMENSION = 1059 ;
:SOUTH-NORTH_GRID_DIMENSION = 677 ;
:BOTTOM-TOP_GRID_DIMENSION = 1 ;
:DX = 3000.f ;
:DY = 3000.f ;
:CEN_LAT = 24.02021f ;
:CEN_LON = -102.066f ;
:TRUELAT1 = 17.5f ;
:TRUELAT2 = 29.5f ;
:MOAD_CEN_LAT = 24.02022f ;
:STAND_LON = -102.0364f ;
:POLE_LAT = 90.f ;
:POLE_LON = 0.f ;
:GRIDTYPE = "C" ;
:GMT = 12.f ;
:JULYR = 2016 ;
:JULDAY = 40 ;
:MAP_PROJ = 1 ;
:MMINLU = "USGS" ;
:MECHANISM = "RADM2" ;
:CREATION_DATE = "26-Feb-2020 14:08:33.098" ;
}
```

Se puede observar en el apartado *dimensions* que el archivo **wrfchemin.nc** contiene 12 registros para la dimensión *Time* que corresponen a las 12 horas de emisiones, y en el apartado *global attributes* que las datos corresponden al año **juliano** 2016 y al día **juliano** 40 (9 de febrero de 2016), así como la fecha y hora de creación del archivo: **26-Feb-2020 14:08:33.098**.

Variable en archivo netCDF	Nombre del contaminante	Unidades
1 E_CO	Carbon Monoxide	mol km^2/hr
2 E_NH3	NH3	mol km^2/hr
3 E_NO	NO	mol km^2/hr
4 E_NO2	NO2	mol km^2/hr
5 E_SO2	SO2	mol km^2/hr
6 E_ALD	ALDEHYDES	mol km^2/hr
7 E_CH4	METHANE	mol km^2/hr
8 E_CSL	CRESOL	mol km^2/hr
9 E_ETH	Ethane	mol km^2/hr
10 E_GLY	Glyoxal	mol km^2/hr
11 E_HC3	HC3	mol km^2/hr
12 E_HC5	HC5	mol km^2/hr
13 E_HC8	HC8	mol km^2/hr
14 E_HCHO	HCHO	mol km^2/hr
15 E_ISO	ISOPRENE	mol km^2/hr
16 E_KET	Acetone	mol km^2/hr
17 E_MACR	Acrolein	mol km^2/hr
18 E_MGLY	MGLY	mol km^2/hr
19 E_MVK	Methyl Vinil Ket	mol km^2/hr
20 E_OL2	Alkenes	mol km^2/hr
21 E_OLI	alkenes	mol km^2/hr
22 E_OLT	Terminal Alkynes	mol km^2/hr
23 E_ORA1	Formic Acid	mol km^2/hr
24 E_ORA2	Acetic Acid	mol km^2/hr
25 E_TOL	TOLUENE	mol km^2/hr
26 E_XYL	XYLENE	mol km^2/hr
27 E_CO2	Carbon Dioxide	mol km^2/hr
28 E_PM_10	PM_10	ug m^2/s
29 E_PM25	PM_25	ug m^2/s
30 E_SO4I	Sulfates	ug m^2/s
31 E_NO3I	Nitrates	ug m^2/s
32 E_PM25I	PM25I	ug m^2/s
33 E_ORGI	Organic	ug m^2/s
34 E_ECI	Elemental Carbon	ug m^2/s
35 E_SO4J	SulfatesJ	ug m^2/s
36 E_NO3J	NitratesJ	ug m^2/s
37 E_PM25J	PM25J	ug m^2/s
38 E_ORGJ	Organic	ug m^2/s
39 E_ECJ	Elemental Carbon	ug m^2/s

Tabla A.1: El archivo contiene 39 variables asociadas a 39 elementos con mediciones cada hora por 12 horas.

Apéndice B

Instalación del programa interpola

La instalación del programa interpola consiste en los siguientes pasos:

- 1. Instalar biblioteca HDF5
- 2. Instalar biblioteca netCDF-C con soporte del formato CDF-4 utilizando la biblioteca HDF5 compilada en el punto 1
- 3. Instalar biblioteca netCDF-Fortran para el uso de la biblioteca netcdf-c compilada en el punto2
- 4. Compilar el programa interpola para el uso de netCDF-Fortran compilada en el punto3

El programa interpola puede ser descargado del repositorio: https://github.com/JoseAgustin/interpola

Su instalación requiere de las bibliotecas `netcdf-c`[1] y `netcdf-fortran`[2], ambos configurados para el soporte del formato mejorado `CDF-4 (HDF5)`[3], siguiente paso es clonar el repositorio del programa interpola y ejecutar el script de configuración del código çonfigureïndicando en la variable de entorno `NETCDF\_ROOT` la ubicación de la biblioteca nefcdf-fortran.

```
$ git clone https://github.com/JoseAgustin/interpola.git
$ cd interpola/src
$ env NETCDF_ROOT=../netcdf ./configure
...
configure: ------------------------------------------------------------
configure: Configuration complete - interpola-3.2 (serial)
configure:
configure: Single precision:           --with-r4=no
configure: Fortran compiler:           FC=gfortran
configure: gfortran compiler:           GFOR=
configure: Enable parallel version:    --enable-parallel=no
configure: Fortran flags:              FCFLAGS=-ffree-form -O2
configure: Fortran OPENMPI:            OPENMP_FCFLAGS=
```

[1] https://github.com/Unidata/netcdf-c
[2] https://github.com/Unidata/netcdf-fortran
[3] https://github.com/HDFGroup/hdf5

```
configure: Root directory of netcdf:   NETCDF=../netcdf
configure: Compiler flags for netcdf:  NC_INC=-I../netcdf/include
configure: Linker flags for netcdf:    NC_LIB=-L../netcdf/lib \
 -lnetcdff -lnetcdf -lhdf5_hl -lhdf5 -lcurl -lm
configure: Install prefix:             --prefix=./src
configure: Executables install prefix: --exec_prefix=${prefix}
configure: Binary directory:           --bindir=${exec_prefix}/bin
configure: ----------------------------------------------------------
```

```
$ make
$ ls
autoconf/     config.status*  interpola.exe*  Makefile.am       s_check.o
testsuite/    configure*      Interpola.F90   Makefile.in       vars_dat.mod
aclocal.m4    configure.ac    Interpola.o     mod_vars_dat.F90  wrfchemi_00z_d01
AUTHORS       COPYING         lee_emis.F90    mod_vars_dat.o    wrfchemin.nc
autogen.sh*   depcomp*        lee_emis.o      NEWS              wrfchemin.nc4
calculos.F90  Doxyfile2       lee_malla.F90   README@           wrfinput
calculos.o    indices.F90     lee_malla.o     salidas.F90
ChangeLog     indices.o       LICENSE         salidas.o
config.log    INSTALL         Makefile        s_check.F90
```

Como resultado se obtiene el archivo ejecutable `interpola.exe`
Las versiones de bibliotecas y compiladores utilizados son los siguientes:

- hdf5-1.10.5
- netcdf-c-4.7.1
- netcdf-fortran-4.4.5
- gcc version 9.2.0

Apéndice C

Herramienta de perfilado Perf

Debido a que muchos eventos relacionados con la ejecución de un programa incluyen eventos que ocurren a nivel del kernel, es necesario ejecutar el programa `perf` como usuario `root` para tener los permisos y tener acceso a espacios de memoria en uso por el kernel linux.

En el siguiente listado se muestra la ejecución del programa `interpola.exe` bajo el control de `perf` como usuario `no root`:

```
$ perf record ./interpola.exe
WARNING: Kernel address maps (/proc/{kallsyms,modules}) are restricted,
check /proc/sys/kernel/kptr_restrict.

Samples in kernel functions may not be resolved if a suitable vmlinux
file is not found in the buildid cache or in the vmlinux path.

Samples in kernel modules won't be resolved at all.

If some relocation was applied (e.g. kexec) symbols may be misresolved
even with a suitable vmlinux or kallsyms file.

Cannot read kernel map
Couldn't record kernel reference relocation symbol
Symbol resolution may be skewed if relocation was used (e.g. kexec).
Check /proc/kallsyms permission or run as root.

 READS_EMISION::    41.4656982
 READS_GRIDS::     0.116981506
 CONVERSION::       94.2296753
 FILE_OUT::         1.04383850
 TOTAL:: seg   136.856186     (min    2.28093648     )
[ perf record: Woken up 83 times to write data ]
[ perf record: Captured and wrote 20.893 MB perf.data (547533 samples) ]
```

La herramienta perf genera por default el archivo **perf.data** con la información de los eventos registrados durante la ejecución del programa, en este caso perf capturo 547 533

eventos en el archivo **perf.data** de tamaño 20.893 MB.

Para generar un reporte se debe invocar el perf con el subcomando **report**. Imagen C.1

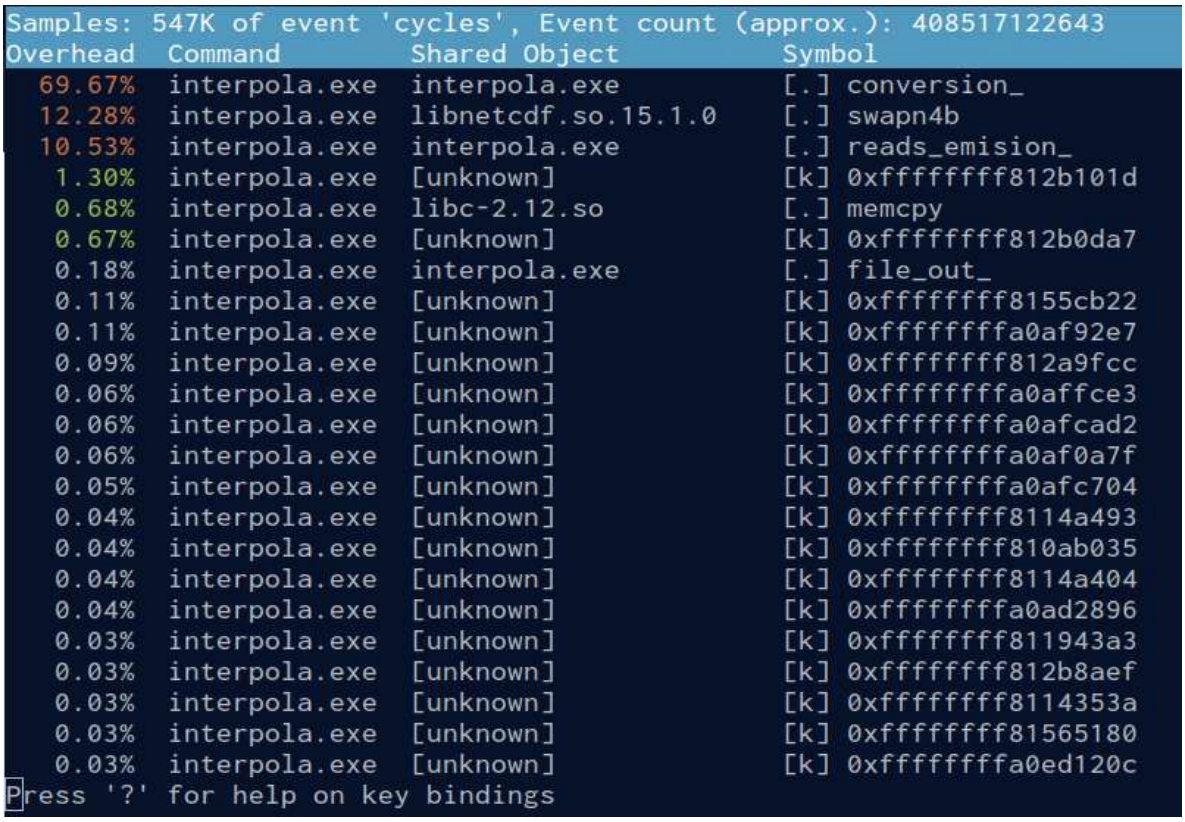

Figura C.1: perf report genera reporte de todos los eventos en espacio kernel y usuario registrados durante la ejecución del programa interporla

Debido a que perf se ejecuto como usuario no **root** no se tienen los permisos suficientes para tener acceso a los símbolos del kernel y se reportan solo las direcciones en memoria. Podemos observar también que el 69.67 % de los eventos fueron capturados durante la ejecución de la subrutina **conversion_**, el 12.28 % durante la ejecución de la subrutina o función **swapn4b**, el 10.53 % durante la ejecución de la subrutina **reads_emision_**.

Para indicar a perf que reporte solo eventos donde tuvo los permisos para consultar los nombres de las funciones, se debe invocar con la opción **-U**. Imagen C.2

Para refinar aun más el reporte debemos solicitar a perf que solo muestre los eventos relacionados directamente con el las subrutinas del programa interpola o de las bibliotecas que fueron compiladas como dependencias, en este caso netcdf, netcdff y hdf5. En la imagen C.3 se puede observar el resultado del siguiente comando:

```
perf report -U --dsos interpola.exe,libnetcdf.so.15.1.0,\
libnetcdff.so.6.2.1,libhdf5.so.103.1.0
```

```
Samples: 509K of event 'cycles', Event count (approx.): 381663521405
Overhead  Command        Shared Object          Symbol
  74.57%  interpola.exe  interpola.exe          [.] conversion_
  13.15%  interpola.exe  libnetcdf.so.15.1.0    [.] swapn4b
  11.27%  interpola.exe  interpola.exe          [.] reads_emision_
   0.73%  interpola.exe  libc-2.12.so           [.] memcpy
   0.19%  interpola.exe  interpola.exe          [.] file_out_
   0.02%  interpola.exe  interpola.exe          [.] reads_grid_
   0.02%  interpola.exe  libc-2.12.so           [.] __memset_sse2
   0.01%  interpola.exe  libnetcdf.so.15.1.0    [.] fill_NC_var
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_search
   0.00%  interpola.exe  interpola.exe          [.] indices_
   0.00%  interpola.exe  libc-2.12.so           [.] _int_malloc
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] memcpy@plt
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_insert_common
   0.00%  interpola.exe  interpola.exe          [.] __vars_dat_MOD_obtiene
   0.00%  interpola.exe  libc-2.12.so           [.] malloc
   0.00%  interpola.exe  libc-2.12.so           [.] malloc_consolidate
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5C_protect
   0.00%  interpola.exe  ld-2.12.so             [.] do_lookup_x
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] nc_unsafe_get_property
   0.00%  interpola.exe  ld-2.12.so             [.] check_match.12447
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] px_get
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] NC3_get_vara
```

Figura C.2: Para filtrar los simbolos que no fueron resueltos por perf, utilizar la opción -U, al filtra estos eventos los porcentajes de cada función o subrutina son recalculados

```
Samples: 509K of event 'cycles', Event count (approx.): 381663521405
Overhead  Command        Shared Object          Symbol
  74.57%  interpola.exe  interpola.exe          [.] conversion_
  13.15%  interpola.exe  libnetcdf.so.15.1.0    [.] swapn4b
  11.27%  interpola.exe  interpola.exe          [.] reads_emision_
   0.19%  interpola.exe  interpola.exe          [.] file_out_
   0.02%  interpola.exe  interpola.exe          [.] reads_grid_
   0.01%  interpola.exe  libnetcdf.so.15.1.0    [.] fill_NC_var
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_search
   0.00%  interpola.exe  interpola.exe          [.] indices_
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] memcpy@plt
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_insert_common
   0.00%  interpola.exe  interpola.exe          [.] __vars_dat_MOD_obtiene
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5C_protect
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] nc_unsafe_get_property
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] px_get
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] NC3_get_vara
   0.00%  interpola.exe  libnetcdff.so.6.2.1    [.] __netcdf_MOD_nf90_put_var_4d_fourbytereal
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5FL__reg_gc_list
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_remove
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] NC3_inq_var
   0.00%  interpola.exe  libnetcdf.so.15.1.0    [.] ncx_howmany
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5S_copy
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5SL_first
   0.00%  interpola.exe  libhdf5.so.103.1.0     [.] H5O_sdspace_shared_decode
```

Figura C.3: Se pueden filtrar solo los eventos ocurridos en la ejecución de funciones o subrutinas de archivos objeto que especifiquemos.

Bibliografía

[1] D. B. Chirila and G. Lohmann. Introduction to Modern Fortran for the Earth System Sciences. Springer Berlin Heidelberg, 2015.

[2] R. Fischer, S. Nowicki, M. Kelley, and G. A. Schmidt. A system of conservative regridding for ice–atmosphere coupling in a General Circulation Model (Gcm). Geoscientific Model Development, 7(3):883–907, May 2014.

[3] M. Fowler and K. Beck. Refactoring improving the design of existing code. 2019.

[4] J. García-Reynoso, B. E. Mar-Morales, and R.-S. L.G. Modelo de distribución espacial, temporal y de especiación del inventario de emisiones de méxico (año base 2008) para su uso en modelización de calidad del aire (diete). Revista Internacional de Contaminación Ambiental, 34(4):635–649, Nov 2018.

[5] G. A. Grell, S. E. Peckham, R. Schmitz, S. A. McKeen, G. Frost, W. C. Skamarock, and B. Eder. Fully coupled "online" chemistry within the wrf model. Atmospheric Environment, 39(37):6957–6975, Dec 2005.

[6] G. Hager and G. Wellein. Introduction to high performance computing for scientists and engineers. Chapman Hall/CRC computational science series; 7. CRC Press, 2011.

[7] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. Communications of the ACM (Association of Computing Machinery); (USA), 335(0), 5 1990.

[8] E. Pärt-Enander and B. Sjögreen. Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic problems. Computers Fluids, 23(3):551–574, 1994.

[9] R. Rew and G. Davis. Netcdf: an interface for scientific data access. IEEE Computer Graphics and Applications, 10(4):76–82, Jul 1990.

[10] S. Roland. Practical Meteorology: An Algebra-based Survey of Atmospheric Science. Dept. of Earth, Ocean Atmospheric Sciences University of British Columbia, 2017.

[11] J. H. Seinfeld and S. N. Pandis. Atmospheric chemistry and physics: from air pollution to climate change. Wiley, 1998.