OpenMP

Trabajo práctico Nº 2

Sistemas Operativos II
DC - FCEFyN - UNC

Abril de 2019

1 Introducción

Los niveles de integración electrónica han permitido la generación de procesadores de arquitecturas multiprocesos, multicore y ahora many integrated core (MIC). Este avance hace necesario que los programadores cuenten con un profundo conocimiento del hardware sobre el que se ejecutan sus programas, y que dichos programas ya no pueden ser monoproceso.

Entre las técnicas y estándares más utilizados para sistemas de memoria compartida y memoria distribuida, se encuentra OpenMP y MPI respectivamente.

2 Objetivo

El objetivo del presente trabajo práctico es que el estudiante sea capaz diseñar una solución que utilice el paradigma de memoria distribuida, utilizando OpenMP.

3 Desarrollo

3.1 Requerimientos

Para realizar el presente trabajo práctico, es necesario instalar la librerías *NetCDF4* [01] en el sistema sobre el cual se diseñe, desarrolle y pruebe la solución al problema. Estas librerías permiten compartir información tecnológica y científica en un formato auto definido, independiente de la arquitectura del sistema y también definen un formato de datos que se transformó en estándar abierto. La librería tiene versiones en Fortran, Python, Java y C, siendo estas últimas las que vamos a utilizar en este trabajo.

Como ejemplo del uso de este formato de datos se tiene la red de de radares meteorológicos NexRad[2] y la constelación de satélites GOES[3], ambos disponibles públicamente.

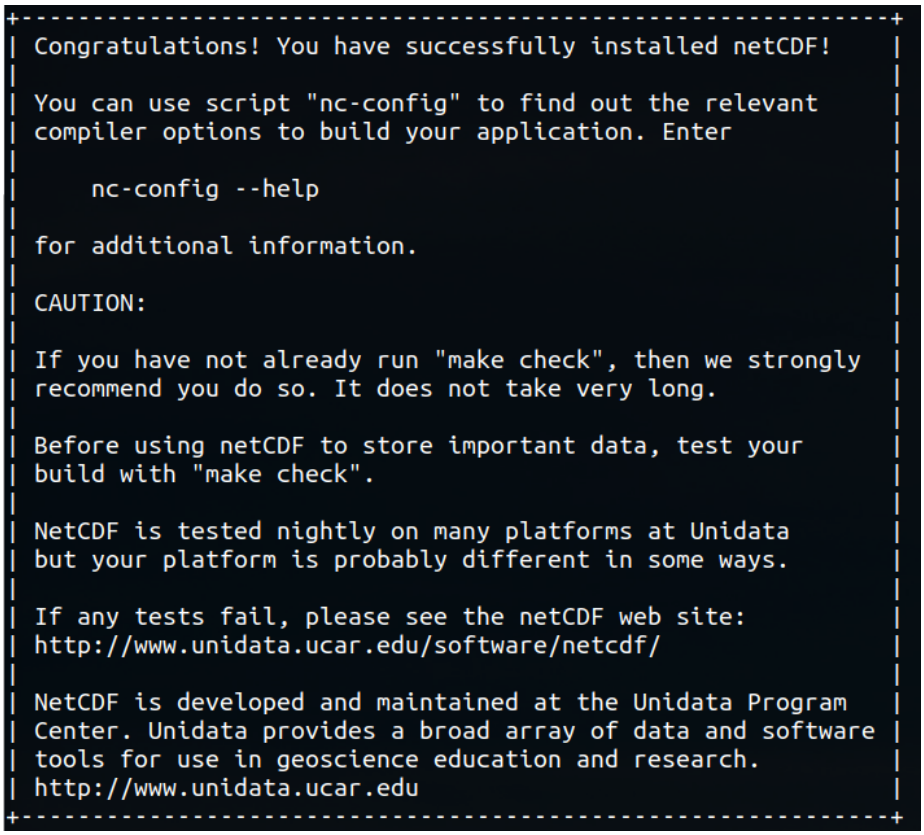

Figure 1: Instalación correcta de Netcdf4

A los fines de facilitar la instalación de estas librería y sus dependencias, y si el manejador de paquetes de su Sistema Operativo no las provee, junto al presente trabajo se entrega un *script*, llamado *install_netcdf4.sh*.

De haberse instalado correctamente la librería, aparecerá un mensaje como el de la 1 indicándolo. En caso de no aparecer dicha figura, por favor revisar los *path* del script y *logs* de cada dependencia. Se les recomienda leer el procedimiento de construcción de estas librerías y como se *linkean* en un programa [4]. También se provee de un archivo base para el uso y lectura del archivo NetCDF[5].

3.2 Problema a desarrollar

El satélite GOES16, usando su canal 2, obtuvo la información almacenada en *OR_ABI-L2-CMIPF-M6C02_G16_s20191011800206_e20191011809514_c20191011809591.nc*.

El nombre del archivo nos informa del instrumento que se utilizó (*ABI-L2-CMIPF-M6C02*), el canal (*C02*), el satélite que lo creó (*G16*), el *timestamp* de

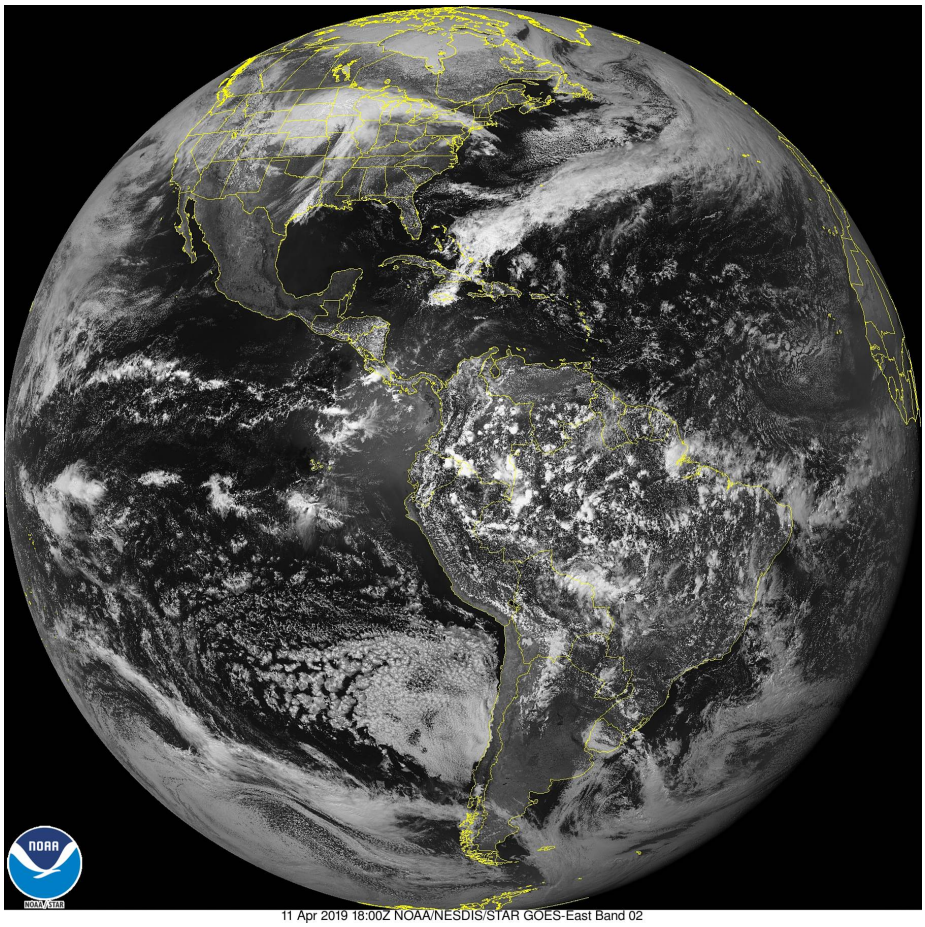

Figure 2: Imagen del canal 2 de GOES 16

la creación del archivo, el inicio del barrido y el final del mismo. El canal 2, corresponde a la longitud de onda de 0.64 µm, correspondiente al espectro de luz visible y posee una resolución de *500 m*, es decir, cada pixel de la imagen, corresponde a 500 metros de la superficie aproximadamente. Una imagen de la salida de este archivo se puede observar en la 2.

Dentro del archivo, hay una variable llamada *CMI*, de una matriz de *float* de 21696 filas y 21696 columnas, con la que se generó esa imagen. Cada punto de la matriz, corresponde a un pixel de la imagen y posee el valor del *"brillo"* en ese punto. Fuera del planeta, todos los valores son *nan*.

Se pide que, se implemente un algoritmo que aplique un filtro de borde (*edge filter*)[6] sobre toda la matriz de la imagen del planeta. La imagen filtrada se obtiene a partir de la convolución

$$g(y,z) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \quad (1)$$

Donde $g(x,y)$ es la imagen filtrada, $f(x,y)$ es la imagen original y ω es la matriz que se define a continuación, con límites $-a <= s <= a$ y $-b <= t <= b$.

$$\omega = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

El resultado de este procesamiento,se debe guardar en un archivo binario, indicando en la documentación, la estructura del mismo. También se debe graficar.

Como metodología para resolver este problema, se solicita que, primero, se realice un diseño que sea solución al problema sin explotar el paralelismo (procedural, que puede ser implementado en Python). Luego, a partir de este, realizar una nueva implementación que realice el proceso mediante el uso de la librería OpenMP, explotando el paralelismo del problema. Para ello, se requiere reconocer qué tipo de paralelismo exhibe el problema en cuestión y luego, diseñar la solución del mismo determinando cuáles son los datos/operaciones paralelizables. Se tendrá en cuenta, el nivel de paralelismo alcanzado.

Además, el informe debe contener gráficos/tablas de los datos recopilados de varias ejecuciones del programa (30, estadístico suficiente), tanto en el clúster de la facultad como localmente en una sola PC, indicando qué ganancia en pérformance existe al distribuir este proceso en comparación a la ejecución local. Comparar la ejecución procedural y de memoria compartida, explicando la diferencias observadas. También se deberá investigar acerca de qué utilidades de profiling gratuitas existen y que brinda cada una (un capítulo del informe), optando por una para realizar las mediciones de tiempo de ejecución de la aplicación diseñada

Al igual que en el trabajo anterior, se exige el uso de Cppcheck y la compilación con el uso de las flags de warning -Werror, -Wall y -pedantic. Se pide utilizar el estilo de escritura de código de GNU o el estilo de escritura del kernel de Linux.

4 Entrega

Se deberá proveer los archivos fuente, así como cualquier otro archivo asociado a la compilación, archivos de proyecto "Makefile" y el código correctamente documentado. También se debe entregar un informe, con el formato adjunto. Gráficos y análisis de comparación entre la ejecución procedural y la distribuida. El informe además debe contener el diseño de la solución y la comparativa de profilers. Se debe asumir que las pruebas de compilación se realizarán en un equipo que cuenta con las herramientas típicas de consola para el desarrollo de

programas (Ejemplo: gcc, make), y NO se cuenta con herramientas "GUI" para la compilación de los mismos (Ej: eclipse).

NOTA: todavía no se encuentra habilitada la cuenta de acceso al clúster de la facultad.

Cuando esté disponible la misma, se informará por LEV los datos necesarios.

5 Evaluación

El presente trabajo práctico es individual deberá entregarse antes de las 23:50 ART del día 11 de Abril de 2019 mediante el LEV. Será corregido y luego deberá coordinar una fecha para la defensa oral del mismo.

6 Referencias

1. Network Common Data Form (NetCDF), https://www.unidata.ucar.edu/software/netcdf/
2. NEXRAD, https://www.ncdc.noaa.gov/data-access/radar-data/nexrad
3. GOES-R Series Satellites, https://www.ncdc.noaa.gov/data-access/satellite-data/goes-r-series-satellites
4. Getting and Building netCDF, https://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_netcdf.html
5. Archivos y códigos, http://200.16.30.250/so2019/
6. https://en.wikipedia.org/wiki/Kernel_(image_processing)